Höhere Technische Bundeslehranstalt Wien 3, Rennweg
IT & Mechatronik
HTL Rennweg :: Rennweg 89b
A-1030 Wien :: Tel +43 1 24215-10 :: Fax DW 18

Diplomarbeit

CAiRO

Crawling Insectoid Robot

ausgeführt an der
Höheren Abteilung für Mechatronik
der Höheren Technischen Lehranstalt Wien 3 Rennweg

im Schuljahr 2016/2017

durch

Manuel Federanko

Moritz Merlin Mießkes

Marc Oppolzer

Nicolas Oswald

unter der Anleitung von
DI Wolfgang Steinwender
DI Herbert Fleck
DI Dr. Christoph Grinschgl
DI Martin Meschik

Wien, 07. April 2017

Wir möchten uns recht herzlich bei unseren Betreuern bedanken, die uns immer unterstützt und jede Frage beantwortet haben.

Wolfgang Steinwender für die Hilfe mit dem 3D-Drucker.
Christoph Grinschgl für die mentale Unterstützung, besonders zur Abschlusspräsentation
Martin Meschik für die Hilfe mit der RS232 Schnittstelle und dem am besten sortierten Chaos und dessen Inhalt
Herbert Fleck für das Erklären von Neuronalen Netzwerken, auch wenn das am Schluss nicht implementiert wurde.

Zusätzlich wollen wir uns bei folgenden Personen bedanken:

Johannes Stehlik für die immer unterhaltsamen Stunden bedanken, die uns sehr bei der Entwicklung unterschiedlicher Teile hilfreich waren.
Klaus Goger, der mehrmals Überstunden machte, ohne die das Projekt niemals zu diesem Ende gekommen wäre.

Danke

Kurzfassung

Der CrAwling Insectoid Robot, oder Cairo, ist ein sechsbeiniger, fernsteuerbarer Roboter. Jedes Bein wird durch zwei direkt nebeneinander angebrachten Servomotoren in Bewegung versetzt. Hierbei übernimmt ein Servo das Heben des Beines, der andere die Schwenkbewegung. Die Kontrolle des Roboters geschieht über ein A31-µQ7-Board von Theobroma Systems.

Um die Motoren anzusteuern werden drei kleine Mikroprozessoren vom Typ PIC16F1827 verwendet, da der Hauptprozessor nicht über ausreichend PWM-fähige Ausgänge verfügt. Jeder dieser drei Coprozessoren übernimmt die Steuerung von vier Servomotoren, die Befehle für die einzustellenden Positionen werden über eine RS232-Schnittstelle übertragen.

Da auf dem Hauptprozessor Linux läuft kann ein WLAN-Stick per USB verbunden werden. Über diesen werden Befehle der Fernsteuerung entgegengenommen und bei Bedarf Updates der Systemsoftware durchgeführt.

Abstract

The Crawling Insectoid Robot, or Cairo, is a six-legged, remote controlled robot. Each leg is steered and moved by two servo motors, where each motor is responsible for either lifting, or panning the leg. The robot is controlled by an A31-µQ7-Board from Theobroma Systems.

The servo motors are controlled by three microprocessors of the type PIC16F1827 by Microchip, since the main processor does not own enough PWM outputs. Each coprocessor takes over the control of four motors. They get their commands over a RS232 connection.

The main processor runs a full Linux system and can therefore use a WLAN-stick, which is connected via USB. Over this connection commands to steer the robot can be issued und updates deployed.

Ehrenwörtliche Erklärung

Ich erkläre an Eides statt, dass ich die individuelle Themenstellung selbstständig und ohne fremde Hilfe verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche erkenntlich gemacht habe.

Wien am 07. April 2017

_____

Manuel Federanko

_____

Moritz Merlin Mießkes

_____

Marc Oppolzer

_____

Nicolas Oswald

Präambel

Die Inhalte dieser Diplomarbeit entsprechen den Qualitätsnormen für „Ingenieurprojekte“ gemäß § 29 der Verordnung des Bundesministers für Unterricht und kulturelle Angelegenheiten über die Reife- und Diplomprüfung in den berufsbildenden höheren Schulen, BGBl. Nr. 847/1992, in der Fassung der Verordnung BGBl. Nr. 269/1993, Nr. 467/1996 und BGBl. II Nr. 123/97.

Betreuende Personen:

DI Wolfgang Steinwender
DI Herbert Fleck
DI Dr. Christoph Grinschgl
DI Martin Meschik

Kooperationspartner:

Elternverein HTL Rennweg
HTL Rennweg
Modellbau G. Kirchert
Theobroma Systems

Verzeichnisse

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Vorwort

Ursprünglich sollte bei dieser Diplomarbeit eine Drohne selbst entwickelt und hergestellt werden, die durch den Einsatz eines Neuronalen Netzwerkes selbstständig fliegen, und in weiterer Folge auch komplexere Manöver wie abheben/landen und Wind selbstständig ausgleichen lernen sollte. Diese Idee wurde jedoch aufgrund akuter Sicherheitsbedenken, wie einer mehrere Kilo schweren, nicht steuerbaren, mit schnell rotierenden Teilen ausgestatten Plattform, verworfen. Da uns die allgemeine Idee jedoch sehr gut gefiel wurde sie so abgeändert, dass selbst bei einem katastrophalen Unfall keine Menschen zu Schaden kommen können. Da die Anwendung von Rädern aus offensichtlichen Gründen zu wenig Herausforderung bat entschieden wir uns für das Modell eines sechsbeinigen Insekts, auch aus dem einfachen Grund, dass es über die Bewegung von Insekten bereits ausführliche Studien gibt. Außerdem gab es an der Schule wegen der Maturaarbeit „Spiderbot“ aus dem Jahr 2005 bereits Erfahrung mit ebensolchen Robotern.

2 Projektziele

2.1 Hauptziele

RE-M 1 Sechsbeiniger Roboter

Ein Roboter mit sechs beweglichen Beinen

Der Roboter soll über sechs bewegliche Beine mit je mindestens zwei Motoren verfügen. Diese sollen unabhängig voneinander ansteuerbar sein, um verschiedene Gangarten zu ermöglichen.

RE-M 2 Zwei Gangarten

Zwei verschiedene Gangarten für unterschiedliche Anforderungen

Die Gangarten sollen unterschiedliche Zwecke erfüllen: 1.) langsam, aber alle Motoren gleichzeitig (maximale Kraft), 2.) schnell, mit einer Mindestgeschwindigkeit von 5 cm/s.

RE-M 3 Fernsteuerbar

Jede der Gangarten soll über einen Computer auswählbar und steuerbar sein

Die Fernsteuerung soll kabellos mit geringer Latenz (unter 100ms) ausgelegt werden. Der Nutzer soll direkt in die momentane Bewegung des Roboters eingreifen und diese verändern können. Wenn die Verbindung zur Fernsteuerung verloren geht, soll (je nach Modus) auf eine Wiederherstellung gewartet, oder eigenständig weiternavigiert werden.

RE-M 4 Entwicklung der Steuerelektronik

Die Steuerelektronik soll neu entwickelt und an die Bedürfnisse angepasst werden

Die bestehende Steuerelektronik soll durch eine neu entwickelte ausgetauscht werden. Diese soll drahtlos kommunizieren können und genügend Rechenleistung für komplexe Bewegungsmuster bereitstellen. Weiters sollen mindestens zwölf Servomotoren angesteuert werden können.

RE-M 5 Mechanische Auslegung

Sicherstellung der Haltbarkeit und Stabilität.

Darunter fällt vorwiegend die Auslegung und Konstruktion des Roboters, da hier Berechnungen gemacht werden müssen, um sicherzugehen, dass die einzelnen Bauteile den Belastungen standhalten. Weiters müssen die Bauteile dimensioniert werden.

Weiters fällt in diesen Bereich auch die Auslegung der Motoren, welche die Beine bewegen werden.

Der Roboter soll ohne Probleme zumindest Lasten von 200 Gramm tragen können. Weiters soll er stabil genug sein, um Objekte ziehen zu können.

RE-M 6 Selbst designtes Gehäuse

Das bestehende Gehäuse soll überarbeitet und mit neueren Produktionsmöglichkeiten effizienter gestaltet werden. Weiters soll an einer Verbesserung des Schwerpunktes und der Bewegungsfreiheit gearbeitet werden.

2.2 Optionale Ziele

RE-O 1 Neuronales Netzwerk

Der Roboter soll auf Wunsch auf die Leistung eines neuronalen Netzwerkes zurückgreifen können. Dieses soll für verschiedene Anforderungen das optimale Muster ermitteln und an die Motorsteuerung übermitteln.

RE-O 2 Android-App

Die Steuerung des Roboters soll nicht nur über einen Computer, sondern auch über ein Android-Smartphone erfolgen. Außerdem können auf diesem Weg auch weitere Daten wie der Akkustand, Beinpositionen und mehr ausgelesen werden.

RE-O 3 Eleganteres Material

Der Roboter soll größtenteils aus Metall bestehen. Dies soll mithilfe eines speziellen 3D-Druck Verfahrens geschehen, welches einen 3D-Druck mit Metallen erlaubt. Dies wertet den Roboter nicht nur optisch auf, sondern erlaubt auch höhere Stabilität.

RE-O 4 Leichter als 4 kg

Der Roboter sollte nicht nur die gewünschten Funktionen bieten, sondern auch einfach zu transportieren sein. Aus diesem Grund muss der Roboter leicht sein, um ihn besser tragen zu können.

RE-O 5 Kleiner als 500 mm x 500 mm x 500 mm

Um platzsparender zu sein, soll der Roboter kleiner werden, ohne Leistung zu verlieren. Somit wäre er auch einfacher zu transportieren.

RE-O 6 Unter 800€

Die Entwicklungskosten des gesamten Projekts sollen durch regelmäßige Preisvergleiche und Recherche weniger als 800€ betragen.

2.3 NICHT Ziele

RE-N 1 Über das Internet steuerbar

Der Roboter soll ausschließlich von einem autorisierten Computer oder Smartphone über kurze Distanzen gesteuert werden können.

RE-N 2 Implementation einer KI

Der Roboter soll nicht während dem Betrieb dazulernen oder selbst neue Funktionen entwickeln. Dieser Schritt soll ausschließlich während der Trainingsphase durchgeführt werden.

RE-N 3 Schwerer als 10 kg

Der Roboter soll nicht schwerer als 10 kg sein, da es eine Zumutung wäre, diesen per Hand zu transportieren.

RE-N 4 Größer als 1 m x 1 m x 1 m

Da so ein riesiger Roboter unpraktisch wäre, soll er nicht größer als 1 m^3 sein. Dieser ist dann zu groß um transportiert zu werden und verbraucht auch so zu viel Platz.

RE-N 5 Normen

Der Roboter wird nach keinen Normen designt. Auch die Sicherheitsmaßnahmen werden nicht Normen entwickelt, sondern situationsbedingt festgelegt.

RE-N 6 Beachtung möglicher Folgearbeiten

Während des gesamten Projekts wird keine Rücksicht auf eventuelle nachfolgende Projekte genommen, welche den Roboter erneut verbessern oder verwenden.

RE-N 7 Testen verschiedener Projektmanagementmethoden

Während des gesamten Projekts soll sich die Leitung und Organisation nicht im bemerkbaren Ausmaß verändern. Es werden also auch keinen verschiedenen Projektmanagementmethoden ausgetestet. Dies soll zu einem koordinierten Arbeiten beitragen.

3 Projektmanagement

Das Projektmanagement befasst sich mit der ständigen Kontrolle und Führung des Projekts. Zudem müssen Risiken und Chancen frühzeitig erkannt und in den Plan, nach welchem das Projekt abläuft, eingearbeitet werden. Eine weitere Aufgabe des Projektmanagers ist es die Teammitglieder einzuteilen, so, dass alle Aufgaben erledigt werden, und das Projekt zu einem erfolgreichen Abschluss kommt.

3.1 Vorprojektphase

In der Vorprojektphase wurde die Idee zu dem Projekt geboren und das Team gebildet. Ebenfalls befinden sich in diesem Abschnitt die Einteilung des Projektteams und die meisten der erstellten Analysen.

3.1.1 Themenfindung

Die Idee entstand aus der Kombination verschiedener Teilziele, welche aus Vorschlägen der Teammitglieder hervorgingen. Diese Ziele waren folgende:

- Das fertige Produkt soll sich bewegen.
- Das fertige Produkt soll fernsteuerbar sein.
- Das fertige Produkt soll ein neuronales Netzwerk enthalten.

Zu Beginn lag der Fokus auf eine einem Quadkopter, da Drohnen viele Herausforderungen und Möglichkeiten bieten. Diese Idee wurde aber aufgrund ihrer Komplexität und anderen Bedenken, wie die Sicherstellung von Sicherheit und der Stromversorgung, verworfen. Da Drohnen unbedingt fliegen müssen wanderte der Fokus in Richtung landbasierter autonomer Roboter. Etwas zuvor wurde auch ein Artikel veröffentlicht, der beschreibt, wie neuronale Netzwerke Robotern helfen können, wenn diese ein Bein verloren haben (Pöppe, 2016). Außerdem gab es schon eine Diplomarbeit die ebenfalls einen Sechsbeinigen Roboter gebaut hatte (Jekl & Zillek, 2005). Dieser würde als Körper fungieren, in welchen nur noch die Elektronik eingepasst werden müsste. Daraus entstand Idee welche auch im Antrag verfasst wurde. Dabei ist darauf zu achten, dass das Ziel des neuronalen Netzwerks als optionales Ziel eingestuft wurde, da dessen gelingen fragwürdig war.

3.1.2 Namensfindung

Der Name Crawling Insectoid Robot beschreibt die Eigenschaften des Roboters.

- Crawling umschreibt die Bewegung die der Roboter bei der Fortbewegung vollführt. Sie erinnert stark an ein Kriechen.
- Insectoid beschreibt den anatomischen Aufbau des Roboters. Da er sechs Beine hat erinnert er an ein Insekt

Aus Diesem Namen ergab sich das Akronym CAIRO, wobei das A aus Crawling entnommen wurde.

3.1.3 Projektorganisation

Die am Projekt beteiligten Personen lassen sich in die Kategorien: Team, Betreuer und Externe einteilen. Jedes Teammitglied hat einen individuellen Betreuer. Als „Externe“ zählen Lehrer, welche nicht im Betreuerteam vertreten sind, Sponsoren und andere Personen welche Einfluss auf das Projekt haben.

3.1.3.1 Teammitglieder

Jedes Teammitglied ist einem bestimmten Bereich, nach ihren Präferenzen im Projekt, zugeteilt.

Moritz Mießkes **Projektleiter**	Bereich	Projektmanagement Mathematische Auslegung
	Mail	miesskes.m@gmail.com
	Betreuer	DI Dr. Christoph Grinschgl
	Ziele	M2, M6 O1, O6 N5, N6, N7
Nicolas Oswald **Stellvertretender** **Projektleiter**	Bereich	Elektronik Projektmanagement
	Mail	oswald.nicolas@gmail.com
	Betreuer	DI Martin Meschik
	Ziele	M1, M2, M3, M4 O1 N2
Manuel Federanko **Teammitglied**	Bereich	Firm- und Software
	Mail	federanko.manuel@gmail.com
	Betreuer	DI Herbert Fleck
	Ziele	M2, M3 O1, O2 N1, N2
Marc Oppolzer **Teammitglied**	Bereich	Mechanische Auslegung
	Mail	marc.oppolzer@gmx.at
	Betreuer	DI Wolfgang Steinwender
	Ziele	M5, M6 O3, O4, O5 N3, N4

Tabelle 1 Teammitglieder

3.1.4 OSP

Der Objektstrukturplan (OSP) dient der Veranschaulichung der einzelnen technischen Komponenten, welche hergestellt oder bearbeitet werden müssen um das Projekt zu einem erfolgreichen Abschluss zu bringen.

3.1.4.1 OSP Aufbau

Ein OSP wird als Baumdiagramm aufgebaut. Dabei wird das Projekt in einige größere Teilgebiete gegliedert, welche dann konkrete Unterpunkte erhalten. Ein umfangreicheres Projekt benötigt auch einen ausführlicheren OSP.

3.1.4.2 OSP CAiRO

Unterhalb ist der OSP zu sehen, welcher für das Projekt CAIRO erstellt wurde. Er besteht aus fünf Unterpunkten und insgesamt dreizehn Objekten.

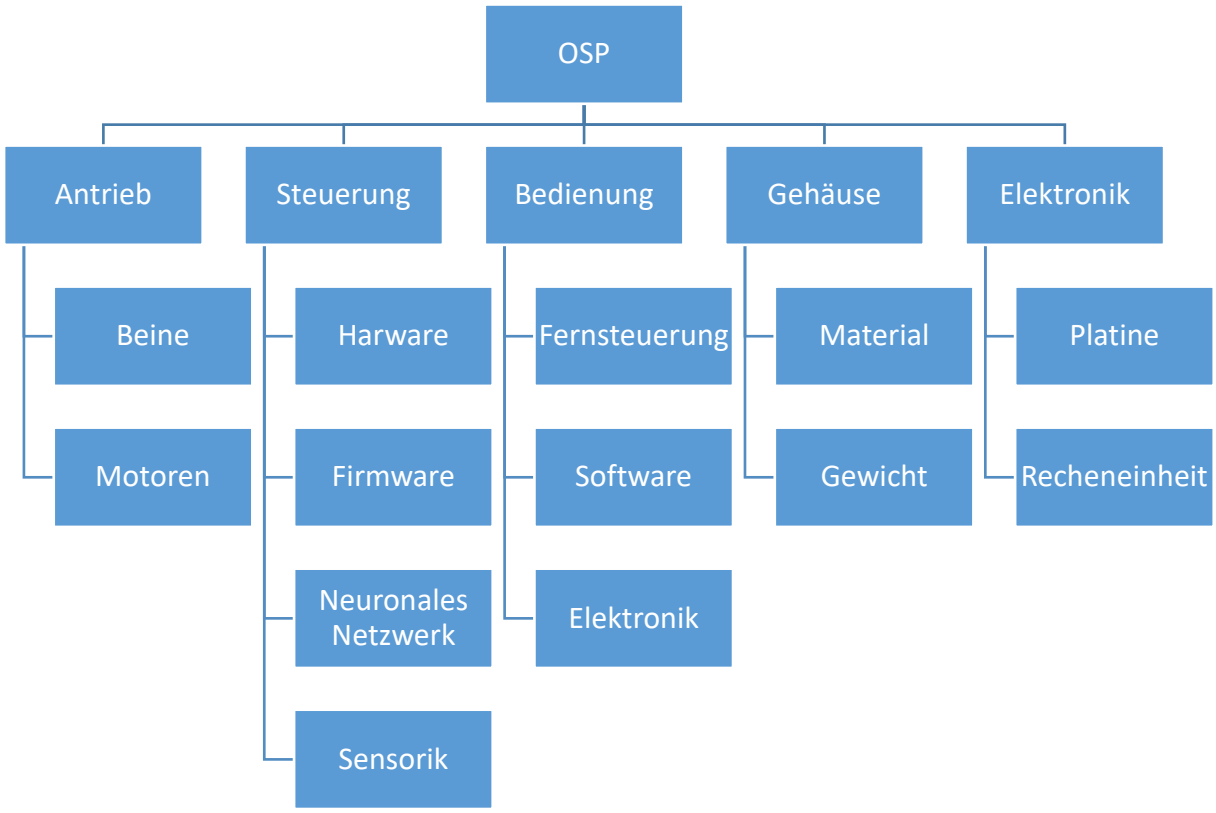

Abbildung 1 Objekt Struktur Plan

3.1.4.3 OSP Fazit

Der OSP zeigt einige zu erwartende Punkte, welche die Grundbausteine eines Roboters wiederlegen. So zum Beispiel Motoren, Material und Platine. Allerdings auch speziellere Unterpunkte, wie beispielsweise das Gewicht.

3.1.5 PSP

Der Projektstrukturplan verwendet die im OSP definierten Objekte und definiert sie als Schritte, welche ausgeführt werden müssen, um das Projekt zu einem erfolgreichen Abschluss zu bringen. Diese Schritte werden Arbeitspakete genannt und einzelnen Personen, oder in größeren Teams, einzelnen Arbeitsgruppen zugeteilt, welche dann für die Vervollständigung verantwortlich sind.

3.1.5.1 PSP Aufbau

Der PSP wird, ähnlich dem OSP, in Unterkategorien eingeteilt. Diese sind allerdings eher so aufgeteilt, dass sie auf einzelne Abteilungen oder Personen zugeschnitten sind. Am Ende jeder Kategorie ist ein Meilenstein, welcher die erfolgreiche Beendigung dieses Abschnittes kennzeichnet.

3.1.5.2 PSP dieses Projektes

Im auf Abbildung 2 zu sehenden PSP sind alle Arbeitspakete zu erkennen. Meilensteine, die einen Unterpunkt abschließen sind dunkler gehalten um die Lesbarkeit zu verbessern.

3.1.5.3 PSP Fazit

Wie in diesem PSP zu erkennen ist, gibt es einige Arbeitspakete, welche anschließend gut auf die Teammitglieder aufzuteilen sind. Auch lässt sich erkennen, dass sich manche Unterpunkte aus Arbeitspaketen zusammensetzt, die unabhängig voneinander sind (5 Elektronische Umsetzung), während andere aufeinander aufbauen (8 Marketing).

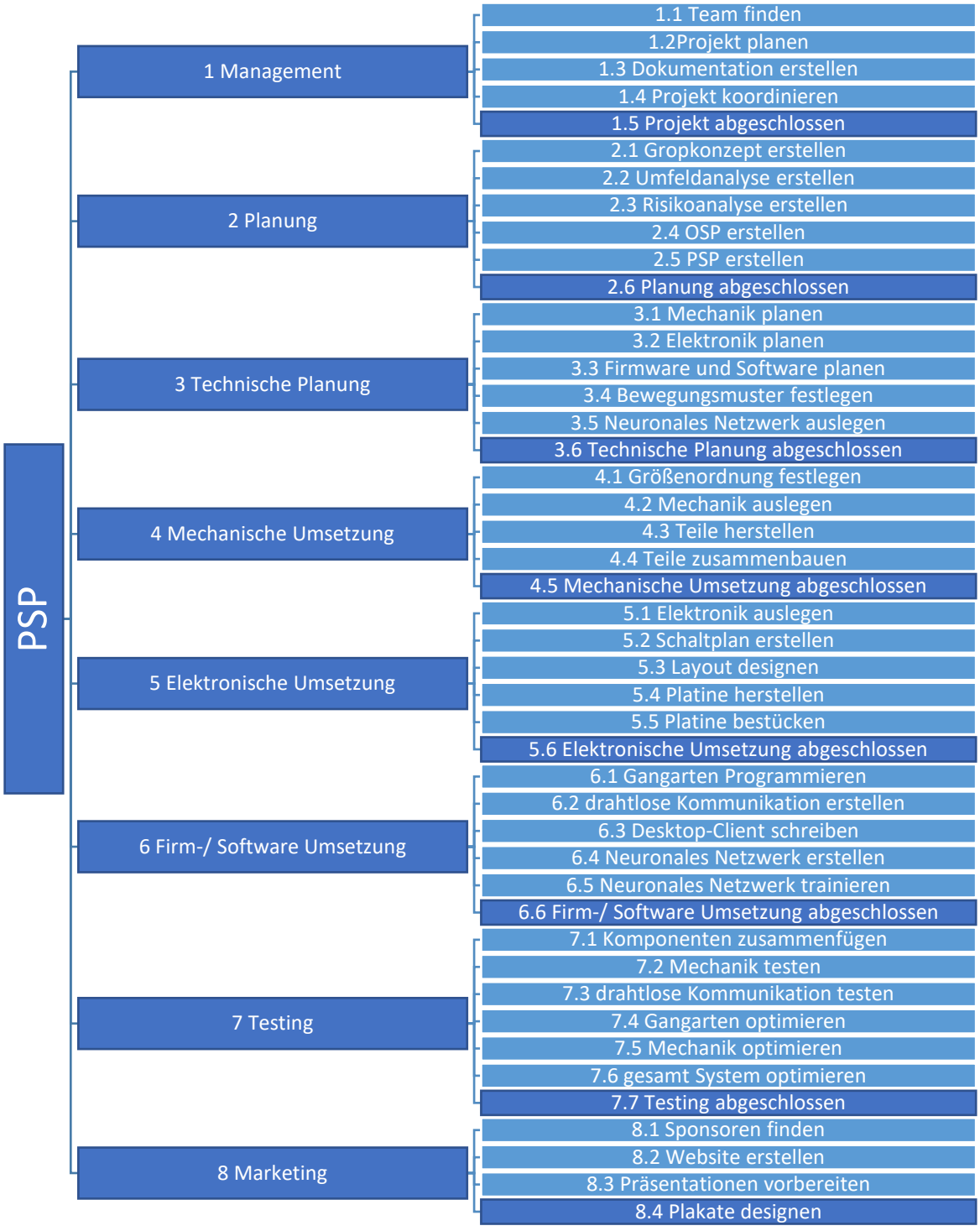

Abbildung 2 Projekt Struktur Plan

3.1.6 Umfeldanalyse

Eine Umfeldanalyse begutachtet alle Parteien, welche einen Einfluss auf das Projekt haben können. Dabei ist es nicht von Bedeutung ob dieser Einfluss positiv oder negativ ist. Auch werden technologische Einflüsse veranschaulicht, dies kann zum Beispiel eine noch unbekannte Technik oder erwartete Entwicklung sein.

3.1.6.1 Umfeldanalyse Aufbau

Die Umfeldanalyse besteht aus einem Feld, in welchem der Name des Projekts steht. Von diesem Feld gehen drei Linien aus, welche die umliegende Fläche in drei gleiche Teile teilt. Diese Teile werden mit den Namen „Intern“, „Extern“ und „Technisch“ versehen. Alle Einflüsse werden nun in das zugehörige Feld eingetragen, dabei können einzelne Parteien mehrfach vorkommen. Als letzter Schritt werden die Parteien nummeriert und bewertet welchen Einfluss sie auf das Projekt haben können: positiv, negativ, oder beides.

3.1.6.2 Umfeldanalyse CAiRO

Unterhalb ist die Umfeldanalyse des Projekts CAIRO zu sehen. In grün sind alle Einflüsse des Projekts angeführt, die nur positive Auswirkungen auf das Projekt haben. Gelb sind solche, die positive als auch negative Auswirkungen haben können. Rein negative Einflüsse wurden rot gehalten.

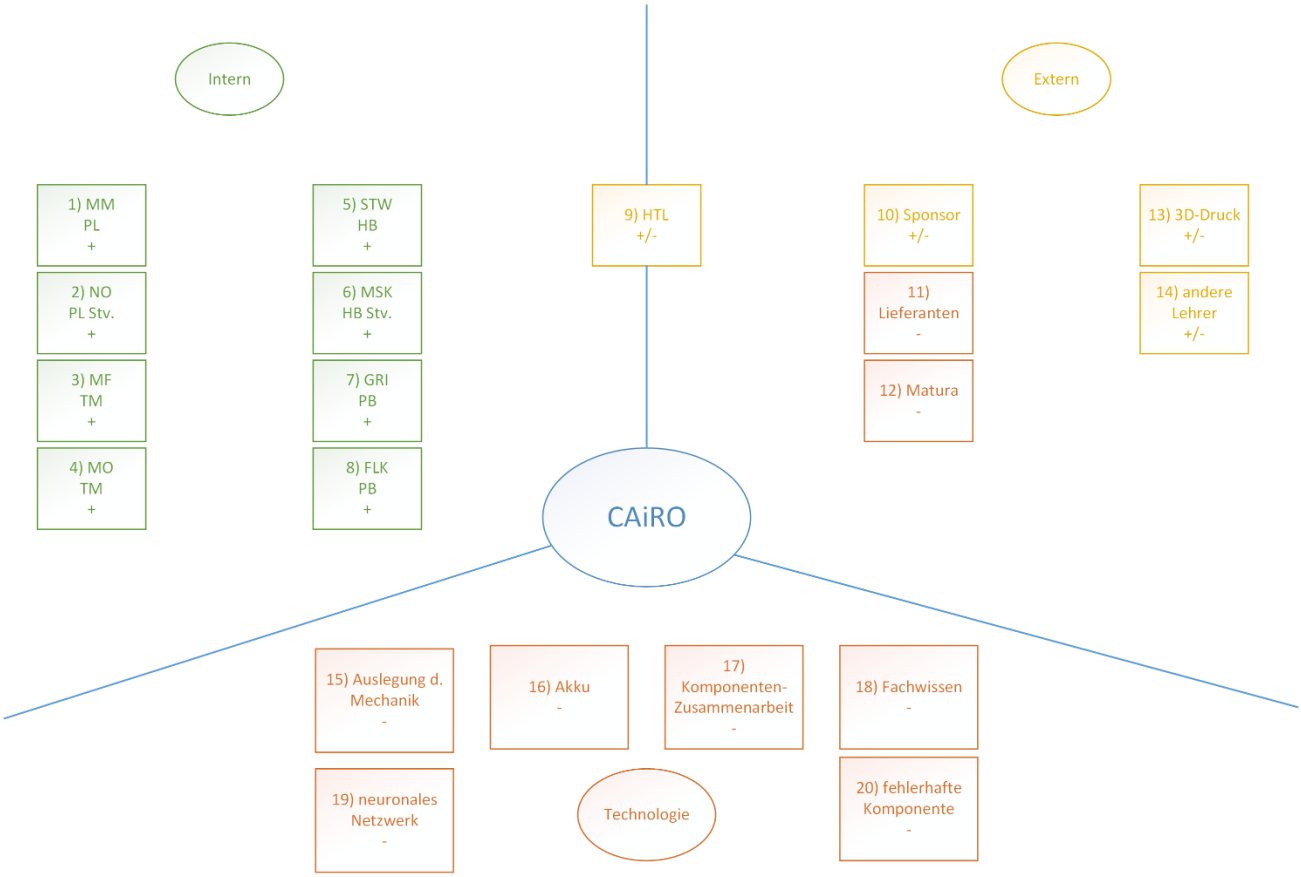

Abbildung 3 Umfeldanalyse

3.1.6.3 Umfeldanalyse Fazit

Aus der Umfeldanalyse konnten einige Informationen gewonnen werden. Eine dieser waren die Ausmaße des Projekts. Es konnte zum ersten Mal ein grober Überblick gewonnen werden, wie groß das Projekt wird, von wie vielen Parteien der Erfolg abhängt und wer aller daran beteiligt ist.

3.1.7 Risikoanalyse

Die Risikoanalyse beschäftigt sich mit den möglichen Problemen, welche während dem Projekt auftreten können, wie wahrscheinlich deren Erscheinen ist und was die Folgen für das Projekt sind. Auch werden Gegenmaßnahmen definiert, welche im Fall des Eintretens eines dieser Probleme ergriffen werden.

3.1.7.1 Risikoanalyse Aufbau

Eine Risikoanalyse wird in den folgenden Schritten erstellt:

1. Die Einflüsse aus der Umfeldanalyse werden bewertet. Dazu gehört die Wahrscheinlichkeit des Eintretens (P) und wie wahrscheinlich ein Misslingen des Projekts aufgrund dieses Problems (A) ist. Diese Werte sollen ein Prozentsatz sein, mit welcher Wahrscheinlichkeit das Problem auftritt/das Projekt dadurch scheitert.
2. Daraufhin werden diese beiden Werte multipliziert und ergeben einen Risikofaktor (RF). Außerdem werden die beiden Werte in ein Diagramm mit den Achsen „A“ und „P“ eingetragen. Je nach Projekt werden eine „grüne“ und eine „rote“ Zone definiert. Befindet sich ein Wert in der grünen Zone ist es nicht dringend für dieses Problem eine ausgefeilte Gegenmaßnahmen zu erarbeiten, da ein Eintritt einerseits unwahrscheinlich und andererseits nicht gefährdend für das Projekt ist. Befindet sich hingegen ein Wert in der roten Zone ist eine ernsthafte Gefährdung für das Projekt gegeben. Wenn keine entsprechenden Gegenmaßnahen entwickelt oder präventive Aktionen eingeleitet werden um das Risiko zu senken, muss die Frage gestellt werden ob das Projekt in diesem Zustand durchgeführt werden soll.

3. Für alle Risiken muss eine Gegenmaßnahme getroffen werden. Gegen Probleme mit geringer Eintrittswahrscheinlichkeit aber starken Auswirkungen empfiehlt sich zum Beispiel eine Versicherung.

3.1.7.2 Risikoanalyse CAiRO

Unterhalb befindet sich die Risikoanalyse dieses Projekts als Tabelle und als Grafik.

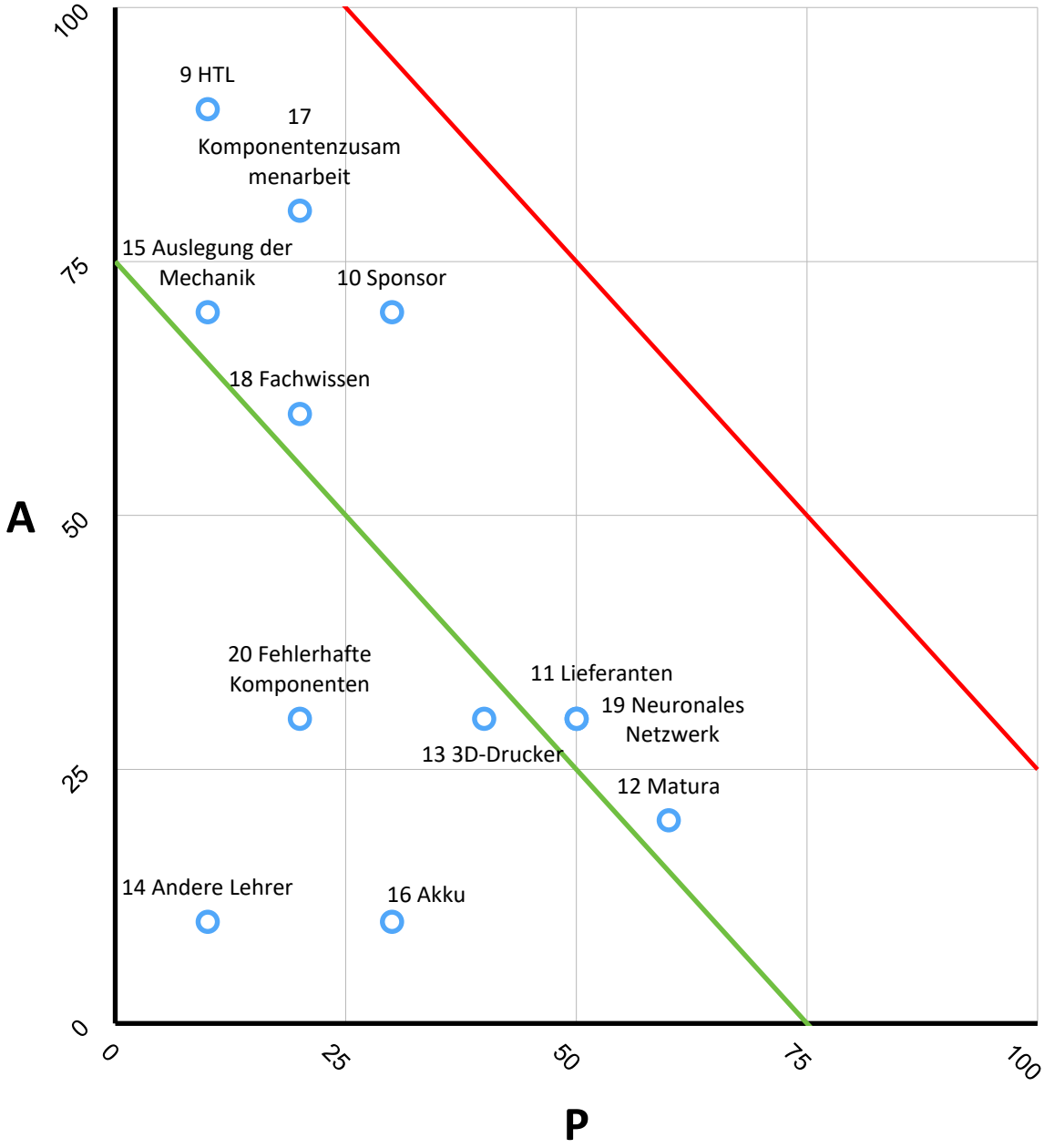

Abbildung 4 Risikoanalyse Graphisch

#	Bezeichnung	Beschreibung des Risikos	P	A	RF
10	Sponsor	Wenig Unterstützung.	30	70	2100
17	Komponenten-zusammenarbeit	Die Kommunikation und Interaktion der vorhandenen Komponenten ist nicht, oder unzureichend gegeben. Fehler können von der Firmware und Elektronik herrühren.	20	80	1600
11	Lieferanten	Späte Lieferung	50	30	1500
19	Neuronales Netzwerk	Das Neuronale Netzwerk kann nicht programmiert und trainiert werden.	50	30	1500
12	Matura	Stresseinwirkung	60	20	1200
14	3D-Drucker	Der 3D-Drucker ist von anderen Projektteams belegt.	40	30	1200
18	Fachwissen	Zu wenig Fachwissen zur effizienten Umsetzung des Projekts	20	60	1200
9	HTL	Bestimmte Einrichtungen stehen nicht zur Verfügung. Das Projekt wird nicht angenommen	10	90	900
15	Auslegung der Mechanik	Die mechanischen Elemente, vor allem die Beine, sind den herrschenden Belastungen nicht gerecht ausgelegt.	10	70	700
20	Fehlerhafte Komponenten	Eine oder mehrere Komponenten sind fehlerhaft oder defekt und können dadurch nicht verwendet werden.	20	30	600
16	Akku	Kapazität, Gewicht zu klein, bzw. zu groß	30	10	300
13	Andere Lehrer	Einer oder mehrere Lehrer, die nicht zum Betreuerteam gehören, finden keine Zeit bei der Lösung möglicher Probleme behilflich zu sein.	10	10	100

Tabelle 2 Risikoanalyse potenzielle Probleme

Als Gegenmaßnahmen wurde folgendes definiert.

#	Bezeichnung	Gegenmaßnahme
9	HTL	Private Räumlichkeiten der Mitarbeiter werden verwendet
10	Sponsor	Die anfallenden Kosten werden zu gleichen Teilen von den Teammitgliedern übernommen
11	Lieferanten	Lieferungen werden so früh als möglich getätigt, oder werden direkt beim Händler gekauft.
12	Matura	Um genügend Zeit für das Lernen für die Matura zu haben, werden mehrere Puffer eingeplant.
15	Auslegung Mechanik	Andere Materialien, wie zum Beispiel Plexiglas werden verwendet
17	Komponenten-zusammenarbeit	Die Elektronik wird mit der Absicht entworfen, eventuelle Änderungen der Bauteile vorzunehmen. Es wird eine verlängerte Testphase eingeplant
18	Fachwissen	Betreuende Lehrer werden um Beistand gebeten
19	Neuronales Netzwerk	Das Ziel wird als Optionales Ziel definiert. Dadurch ist das Projekt nicht gefährdet.

Tabelle 3Risikoanalyse Geenmaßnahmen

3.1.7.3 Risikoanalyse Fazit

Aus der Risikoanalyse lassen sich einige wichtige Daten ablesen. Durch Risiko 17 wurde eine verlängerte Testphase eingeplant, welche zwar nicht benötigt wurde aber als guter Zeitpuffer diente.

3.1.8 Meilensteinplan

Der Meilensteinplan gibt einen zeitlichen Rahmen vor, durch den das Projekt besser geleitet werden kann, da es eine Referenz für den Abschluss bestimmter Aktionen gibt. Er wirkt auch als Checkliste und hilft zu erkennen, wie viel des Projekts schon erledigt wurde.

3.1.8.1 Meilensteinplan Aufbau

Für den Meilensteinplan wurden die Arbeitspakete aus dem PSP weiterverwendet. Jedem Arbeitspaket wurden Durchführungsdauer und Abhängigkeiten zugeteilt. So kann zum Beispiel eine Platine nicht ohne Layout geätzt werden. Das Layout wiederum kann nicht

ohne Schaltplan erstellt werden. Aus den Zeiten der einzelnen Arbeitspakete und deren Abhängigkeiten ergibt sich ein kritischer Pfad: verzögert sich dieser, verzögert sich das gesamte Projekt. Die Meilensteine aus dem PSP werden natürlich auch in den Plan eingetragen und dienen als gute Referenz.

3.1.8.2 Meilensteinplan CAiRO

Unterhalb ist der Meilensteinplan des Projekts CAIRO angeführt. Die zugehörige Grafik mit allen Arbeitspaketen befindet sich im Anhang und auf der CD.

Datum	Meilenstein
15.09.2016	Projektplanung abgeschlossen
16.09.2016	Diplomarbeitsantrag abgegeben
29.09.2016	Website Online
12.01.2017	Elektronik fertig
18.01.2017	Mechanik fertig
20.01.2017	Firmware fertig
18.11.2016	Jugendinnovativ Anmeldung fertig
01.12.2016	Interne Präsentation fertig
21.02.2017	Testphase abgeschlossen
28.03.2017	Endpräsentation
04.04.2017	Abnahme Diplomarbeit
22.05.2017	Defensio

Tabelle 4 Meilensteinplan

3.1.8.3 Meilensteinplan Fazit

Der Meilensteinplan ist ein gutes Mittel um den groben Projektablauf gut darzustellen und ein Gefühl für die Zeiten zu geben. Dieser muss allerdings auch regelmäßig überprüft und eventuell überarbeitet werden.

3.1.9 Kostenplanung

Die Kostenplanung ist eine notwendige Analyse um die ungefähren Kosten des Projekts abschätzen zu können.

3.1.9.1 Kostenplanung Aufbau

Die Kostenplanung ist eine Liste, in die alle zu erwarteten Kostenstellen samt dem Betrag eingetragen werden. Dabei ist es nicht wichtig, die genauen Kosten bestimmter Bauteile, zum Beispiel eines Servomotors zu verwenden, da immer mit Mehrkosten gerechnet werden kann. Alle Kosten werden addiert um die Gesamtkosten zu erhalten. Danach muss ein Plan ausgearbeitet werden um diese Kosten zu decken.

3.1.9.2 Kostenplanung CAiRO

Unterhalb ist die Kostenplanung des Projekts CAIRO angeführt.

Pos.	Bezeichnung des Aufwands	Kosten	Kumuliert
1	Website	EUR 20	EUR 20
2	Büroutensilien	EUR 20	EUR 40
3	3D-Druck Material	EUR 30	EUR 70
4	12x Servomotoren	EUR 180	EUR 250
5	Elektronische Bauteile	EUR 150	EUR 400
6	Sonstiges	EUR 50	EUR 450
-	Gesamtkosten		EUR 450

Tabelle 5 Kostenplanung

Zur Deckung der Kosten wurde vereinbart Sponsoren um Mithilfe anzufragen. Die übrigen Kosten würden vom Team zu gleichen Teilen getragen werden.

3.1.9.3 Tatsächliche Kosten

In der folgenden Tabelle werden die tatsächlichen Kosten des Projekts aufgeführt.

Produkt	Stk.	Einzelpreis	Gesamtpreis
HS-225BB Servo	12	EUR 16,50	EUR 198,00
Blechschrauben	1	EUR 5,95	EUR 5,95
Linsenkopfschrauben	1	EUR 3,49	EUR 3,49
Website	1	EUR 14,00	EUR 14,00

3D-Druck	1	EUR 0,00	EUR 0,00
PIC16F1827	6	EUR 2,19	EUR 13,14
Platine	1	EUR 0,00	EUR 0,00
Summe			EUR 216,58

Tabelle 6 Ausgaben

3.1.9.4 Kostenplanung Fazit

Aus der Kostenplanung konnte entnommen werden, dass keine zu hohen Kosten anfallen würden und diese auch gut von allen Teammitgliedern getragen werden konnten, selbst wenn keine Sponsoren aufkommen würden.

3.1.10 Zeitplanung

Um dem Team eine bessere Vorahnung zu geben wie lange jedes Teammitglied am Projekt arbeiten muss, wird eine Zeitplanung durchgeführt. Dies hilft bei der Planung und Koordination.

3.1.10.1 Zeitplanung Aufbau

Um die Zeitplanung möglichst akkurat zu gestalten ist es von Vorteil, die schon im PSP erstellten Arbeitspakete und Zuständigkeiten zu übernehmen. Jedem Arbeitspaket muss eine Dauer zugewiesen werden, die voraussichtlich aufgewendet werden muss um das Arbeitspaket zu vervollständigen. Daraus folgt eine Tabelle mit der Dauer die jedes Teammitglied für jedes Arbeitspaket aufbringen muss. Daraufhin müssen nur noch die Zeiten addiert werden. Aufgrund dieser Analyse kann es sein, dass erkannt wird das eine oder mehrere Parteien mehr Arbeit zu verrichten haben als andere. Daraufhin kann es von Vorteil sein die Arbeitspakete neu zu verteilen.

3.1.10.2 Zeitplanung CAiRO

Im Projekt CAIRO wurden manche Arbeitspakete mit der Zuständigkeit „Alle“ gekennzeichnet. Dies sind Pakete an denen das gesamte Team gemeinsam arbeiten muss. Bei diesen wurde die geplante Zeit zu gleichen Teilen auf die Teammitglieder aufgeteilt. Es wurde angenommen, dass an einem Tag durchschnittlich 2,5 Stunden am Projekt gearbeitet wird. Daraus ergeben sich dann, wie zum Beispiel im Arbeitspaket 1.1, aus 20 Stunden acht Tage an denen das gearbeitet werden muss. Dies war auch der Wert der für die Berechnung des Meilensteinplans verwendet wurde.

Arbeitspaket	Zuständiger	Dauer in Stunden	Dauer in Tagen
1.1	Alle	20	8
1.2	Alle	1	1
1.3	Alle	20	8
1.4	Moritz Mießkes	35	14
2.1	Alle	10	4
2.2	Moritz Mießkes	4	2
2.3	Moritz Mießkes	4	2
2.4	Moritz Mießkes	4	2
2.5	Moritz Mießkes	4	2
3.1	Marc Oppolzer	15	6
3.2	Nicolas Oswald	15	6
3.3	Manuel Federanko	10	4
3.4	Nicolas Oswald	7	3
3.5	Moritz Mießkes	20	8
4.1	Marc Oppolzer	8	4
4.2	Marc Oppolzer	12	5
4.3	Marc Oppolzer	15	6
4.4	Marc Oppolzer	10	4
5.1	Nicolas Oswald	10	4
5.2	Nicolas Oswald	10	4
5.3	Nicolas Oswald	10	4
5.4	Nicolas Oswald	1	1
5.5	Nicolas Oswald	4	2
6.1	Manuel Federanko	20	8
6.2	Manuel Federanko	15	6
6.3	Manuel Federanko	20	8
6.4	Manuel Federanko	10	4
6.4	Nicolas Oswald	10	4
6.4	Moritz Mießkes	15	6
6.5	Nicolas Oswald	20	8
7.1	Alle	5	2
7.2	Marc Oppolzer	10	4

7.3	Manuel Federanko	8	4
7.4	Moritz Mießkes	7	3
7.5	Marc Oppolzer	10	4
7.6	Alle	40	16
8.1	Moritz Mießkes	20	8
8.2	Alle	15	6
8.3	Alle	10	4
8.4	Moritz Mießkes	10	4

Tabelle 7 Zeitaufstellung genau

Zusammengefasst ergibt das den folgenden Stundenaufwand.

	Moritz Mießkes	Nicolas Oswald	Marc Oppolzer	Manuel Federanko
Alleine	123	87	80	83
Gemeinsam	121	121	121	121
Summe	244	208	201	204

Tabelle 8 Zeitaufstellung zusammengefasst

Zur Veranschaulichung wurden diese Daten auch in ein Diagramm übernommen.

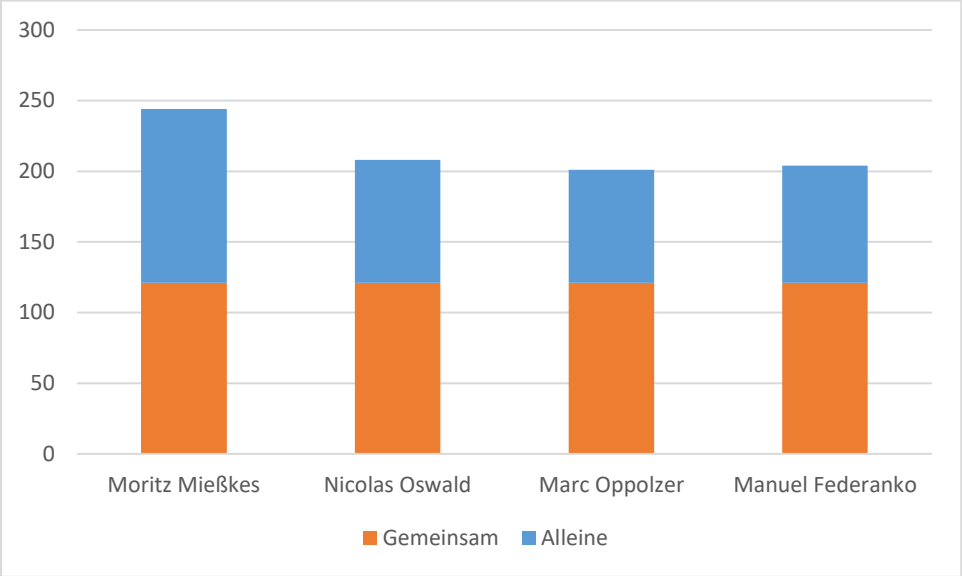

Abbildung 5 Zeitaufstellung Graphisch

3.1.10.3 Zeitplanung Fazit

Aus dieser Zeitplanung ist zu erkennen, dass die Teammitglieder nahe an der vorgegebenen maximalen Arbeitszeit sind. Diese ist für den Projektleiter höher als für den anderen. Zudem zeigt das Diagramm auch, dass ein Großteil der aufgewendeten Zeit an Arbeitspaketen verwendet wird, die gemeinsam erledigt werden müssen.

3.1.11 Datenverwaltung

Es ist notwendig die Datenverwaltung zu planen, da im Laufe eines Projekts viele verschiedene Dateien erstellt werden und diese vom gesamten Team einsehbar sein sollen. Deswegen wurde ein zentraler Speicherplatz definiert. Außerdem musste eine gute Kommunikation zwischen den einzelnen Teammitgliedern sichergestellt werden.

3.1.11.1 Data-Sharing

Als Datenspeicherplatz wurde OneDrive (Microsoft, 2017) von Microsoft verwendet. Durch diese Plattform wurden alle Daten der Teammitglieder miteinander synchronisiert. Außerdem erlaubt es Microsoft Office (Microsoft, 2017) Produkte wie Word (Microsoft, 2017) gleichzeitig zu bearbeiten.

3.1.11.2 Protokoll Software

Als Protokoll Software wurde die Android Applikation Time Recording (Android Time Recording, 2017) verwendet. Diese erlaubt das einfache Eintragen von Zeiten zu denen gearbeitet wurde. Außerdem bietet sie die Möglichkeit die Daten als Backup zu speichern. Das Erfassen der Zeiten ist notwendig, um die tatsächlich aufgewendete Arbeit mit der veranschlagten zu vergleichen. Dies hilft spätere Projekte besser zu planen.

3.1.11.3 Kommunikationsmedium

Als Kommunikationsmedium wurden die Applikationen WhatsApp (Whatsapp, 2017) und Allo (Google, 2017) verwendet. Diese bieten einfache Möglichkeiten um schnell mit dem gesamten Team zu kommunizieren.

3.2 Vertretung nach Außen

In diesem Kapitel geht es um die Kommunikation zu Personen außerhalb des Teams. Dazu gehören unter anderem die Sponsoren und Wettbewerbe an denen teilgenommen wurde.

3.2.1 Sponsoren

Es wurden Sponsoren gesucht um das Projekt zu finanzieren. Dazu wurde im Umfeld der Teammitglieder und bei schon bekannten Sponsoren anderer und ehemaliger Projekte angefragt.

3.2.1.1 Sponsoren finden

Es wurden vier verschiedene Parteien kontaktiert und um ein Sponsoring angefragt, wovon mit dreien auch eines eingegangen werden konnte. Angeschrieben wurden Theobroma Systems (Theobroma Systems, 2017), Modellbaugeschäft G. Kirchert (Modellbau Kirchert, 2017), der Elternverein HTL Rennweg und Conrad Electronics (Conrad, 2017), welche sich allerdings nach kurzem Mailverkehr nicht mehr meldeten.

3.2.1.2 Theobroma Systems

Theobroma Systems ist eine Firma welche System-on-Modules herstellt. Theobroma Systems unterstützte das Projekt durch Bereitstellung von Knowhow, ihrer Räumlichkeiten und der Finanzierung der Platine. Außerdem wurde ein von Theobroma Systems entwickelter Chip als Prozessor für den Roboter verwendet.

3.2.1.3 Modellbaugeschäft G. Kirchert

Das Modellbaugeschäft G. Kirchert unterstützte das Projekt durch einen großzügigen Rabatt bei den Servomotoren. Außerdem stellte es Informationen für die richtigen Produkte zu Verfügung.

3.2.1.4 Elternverein HTL Rennweg

Der Elternverein Unterstützte das Projekt mit einer Finanziellen Unterstützung von €150.

3.2.2 Website

Das Team erstellte eine Website unter der Webadresse www.cairobot.at[1]. Wodurch Aufmerksamkeit auf das Projekt gelenkt werden konnte.

3.2.2.1 Warum eine Website

Eine Website wurde erstellt um dem Projekt ein professionaleres Erscheinen zu verleihen. Auch sollte die Reichweite des Projekts zu vergrößern, um eventuelle Sponsoren zu finden

3.2.2.2 Kooperation mit anderen Teams

Um dem eigenen Projekt als auch den anderen Projekten der 5BM eine größere Reichweite zu erlangen, wurde eine Kooperation zwischen den einzelnen Teams geschlossen. Jedes Projekt mit einer Website präsentierte auf dieser alle anderen Projekte.

3.2.3 Tage der offenen Tür

Um die Bekanntheit des Projekts zu steigern und der HTL Rennweg einen besseren Tag der offenen Tür zu gewähren, wurde an diesen Tagen das Projekt den Besuchern präsentiert. Hierzu wurde auch ein Plakat entworfen. Dieses ist im Anhang zu finden.

[1] Diese Webadresse ist nur bis 12.09.2017 erreichbar.

3.2.4 Wettbewerbe

Das Projekt CAIRO nahm an einigen Wettbewerben teil[2], da die Preisgelder unterstützend bei der Tilgung der Kosten wären. Zudem war auch bei den Wettbewerben der Hintergrund vorhanden, dem Projekt eine größere Plattform bieten zu können

3.2.4.1 Jugend Innovativ

Der Jugend Innovativ Wettbewerb ist ein alljährlicher Wettbewerb, bei dem verschiedene Projekte teilnehmen können. Dafür musste eine Zusammenfassung des Projekts verfasst werden.

3.2.4.2 Bosch Technik fürs Leben Preis

Auch am Bosch Technik fürs Leben Preis wurde teilgenommen. Hierzu musste nur ein Formular ausgefüllt werden.

3.2.4.3 AXAward

Der AXAward ist ein neuer Wettbewerb an dem das Team teilnahm.

3.2.5 Präsentation

Es wurden zwei verpflichtende Präsentationen gehalten. Während die zweite Präsentation eher einer Verkaufsvorstellung entsprach wurde bei der ersten das Projekt auch von der technischen Seite erklärt. Marc Oppolzer war bei der ersten Präsentation nicht anwesend wodurch sein Teil von den anderen Teammitgliedern übernommen werden musste. Die Präsentationen befinden sich im Anhang.

3.3 Teaminternes Management

Da jedes Teammitglied seine eigenen Aufgaben bearbeitet, kommt es oft dazu, dass an bestimmten Bereichen des Projekts gearbeitet wird, während andere vernachlässigt werden. Darum muss das Team gemanagt werden.

3.3.1 Meilenstein-Neuplanung

Ein wichtiger Teil im Management ist die wiederholte Überprüfung der Einhaltung des Zeitplans. Es wurde allerdings schnell erkannt, dass der erste Entwurf dieses Plans viel zu strickt war und wenig Platz für Fehler oder Verzögerungen bat. Darum wurden alle

[2] Mit dem Verfassen dieser Arbeit sind noch keine Platzierungen bekannt.

Meilensteine verschoben. Dadurch wurde ein Arbeiten nach Plan ermöglicht. Trotz dieser Umplanung war das Projekt oft in Verzug, dieser konnte aber durch eine ausgesprochen kurze Testphase eingeholt werden.

3.3.2 Change Request

Da zu Beginn des Projekts nicht geplant wurde den Roboter vollständig neu zu bauen, waren die Ziele anders definiert. So war zum Beispiel Marc Oppolzer ursprünglich für die Elektronik zuständig. Dies wurde aber durch einen Change Request so geändert, dass Marc Oppolzer für die mechanische Auslegung das Design zuständig wurde. Ein weiterer Punkt der in diesem Change-Request behandelt wurde, war die minimale Geschwindigkeit des Roboters. Ursprünglich waren 1,5 km/h geplant, dies entspricht 42 cm/s. Da nach den ersten Berechnungen und Designentwürfen so eine Geschwindigkeit nicht zu erwarten war, wurde sie auf 5 cm/s abgeändert. Diese konnte leicht erreicht werden.

3.3.3 Termineinhaltung

Die notwendigen Termine die einzuhalten waren, wurden rechtzeitig bekanntgegeben und in die Planung des Projekts eingegliedert. Um rechtzeitig mit allen notwendigen Vorbereitungen fertig zu sein wurde mit den Arbeiten für diesen speziellen Termin immer spätestens eine Woche vor diesem begonnen. Dadurch konnten alle Termine eingehalten werden.

3.3.4 Protokoll

Es war die Vorgabe alle zwei Wochen ein Protokoll abzugeben, in welchem der Stand des Projekts wiedergegeben wurde.

3.3.5 Betreuergespräche

Abwechselnd mit den Protokollen musste alle zwei Wochen ein Gespräch mit dem Hauptbetreuer gehalten werden. Diese waren manchmal nur kurze Erklärungen, dass alles nach Plan ging, aber manchmal auch ausführliche Berichte über ein bestimmtes Problem.

3.4 Endprojektphase

In der Endprojektphase werden alle offenen Aufgaben des Projekts abgeschlossen. Dazu gehören unter anderem die Defensio und die Abgabe des Korrekturexemplars der Dokumentation.

4 Mechanik

4.1 Materialwahl/Herstellungsverfahren

Eine große Frage war, aus welchem Material der Roboter hergestellt werden soll. Diese Frage ist insofern schwer zu beantworten, da sowohl Preis, Fertigung und Gewicht, als auch Design und Notwendigkeit beachtet werden müssen. Um die richtige Entscheidung zu treffen, wurde eine Pro/Contra-Liste erstellt.

	Pro	Contra	Hinweise
Stahl	Sieht gut aus, stabil, einfach zu bearbeiten, in Schule bearbeitbar	Teuer, schwer, fehleranfällig bei Bearbeitung	Mit Stahl ist gemeint, dass Stahlbleche verwendet werden, welche dann in die Formen gebogen werden können.
Aluminium	Sieht gut aus, stabil, einfach zu bearbeiten, in Schule bearbeitbar	Teuer, fehleranfällig bei Bearbeitung, leicht zerkratzbar	Mit Aluminium ist gemeint, dass Aluminiumbleche verwendet werden, welche dann in Formen gebogen werden können.
Plexiglas	Sieht gut aus, stabil, in Schule bearbeitbar, verschiedene Farben	Schwer zu bearbeiten (da nur Platten hergestellt werden können, müssten komplizierte Teile aus mehreren Platten geklebt werden)	Mit Plexiglas ist gemeint, dass Bauteile aus Plexiglas gelasert werden.
PLA	Billig, sehr genau aufgrund des 3D-Drucks, leicht, verschiedene Farben	Einfach zu biegen/zerbrechen, nicht in Schule bearbeitbar, nicht möglich Überhänge zu drucken	Mit PLA ist die chemische Verbindung „Polylactid“ gemeint, welche mittels 3D-Druck-Verfahren bearbeitet werden kann.

ABS	Billig, sehr genau aufgrund des 3D-Drucks, leicht, verschiedene Farben, in Schule bearbeitbar	Einfach zu biegen/zerbrechen, nicht möglich Überhänge zu drucken	Mit ABS ist die chemische Verbindung „Acrylnitril-Butadien-Styrol“ gemeint, welche mittels 3D-Druck-Verfahren bearbeitet werden kann.

Tabelle 9: Material Auswahl

Ausgewählt wurde dann „ABS“ da es für dieses Projekt, die meisten Vorteile bietet. Da Servomotoren mit großer Leistung teuer sind, musste ein Material gewählt werden, welches leicht ist. ABS eignet sich somit besser als Metalle, da es ein Kunststoff ist. Des Weiteren lassen sich relativ einfach, komplizierte Formen herstellen. Zum Beispiel muss nicht gebohrt werden, da Bohrungen einfach als „Loch“ gedruckt werden können und somit an dieser Stelle kein Material ist. Im Vergleich zu „PLA“ ist es außerdem in der Schule bearbeitbar. Da es möglich ist, fast jede Form herzustellen gibt es designtechnisch keine Einschränkungen. Das ist sehr hilfreich, da der Roboter im Endeffekt auch gut aussehen sollte. Zusätzlich gibt es verschiedene Farben, mit denen gedruckt werden kann.

Ein Nachteil ist allerdings, dass ABS nicht sehr stabil, verglichen zu anderen Materialien, ist. Somit darf der Roboter keinen zu großen Belastungen ausgesetzt sein, da die Beine sonst brechen würden. Ein weiteres Problem ist die Bearbeitung. Da der 3D-Drucker maximal einen Überhang von 45° drucken kann, gibt es gewisse Einschränkungen. Wenn, zum Beispiel, ein „L-Profil“ hergestellt werden soll, kann es nur liegend oder aufrechtstehend gedruckt werden. Wenn es kopfüber gedruckt werden soll, wäre ein Überhang von mehr als 45° der Fall (90°), was der Drucker nicht herstellen kann. Dieses Problem muss umgangen werden, indem der Überhang separat hergestellt und anschließend entweder mit Schrauben, etc... befestigt oder angeklebt wird. Dies ist zwar eine mühsamere und weniger genaue Methode, allerdings funktioniert sie.

4.2 Mechanische Auslegung

Die mechanische Auslegung oder auch Berechnung, ist jener Teil eines Projektes, welcher gewährleistet, dass das Produkt den ausgesetzten Belastungen standhält. Hierfür werden die Teile betrachtet, die entweder sehr großen Kräften ausgesetzt sind

oder selbst sehr klein, dünn, filigran, etc… sind. Für diese Berechnung wurde das Programm PTC Mathcad Prime 3.1 (PTC, 2017) verwendet. Dieses Programm wurde im Zuge des Unterrichts im Fach „Konstruktionsübungen“ verwendet, weshalb Kenntnis darüber bestand.

Weiters musste eine Möglichkeit gefunden werden, alle Einzelteile so zusammenzusetzen, dass alles funktioniert. Beispielsweise mussten die Servomotoren so angebracht werden, dass sie genug Platz haben sich zu bewegen, keine anderen Bauteile zu behindern und zusätzlich die Beine in die richtige Richtung bewegen zu können (siehe Konstruktion).

Der Bereich „Mechanische Auslegung“ steht in engem Zusammenhang mit dem Bereich „Konstruktion“. Er gibt der Konstruktion maximale Werte vor (Länge, Volumen, Gewicht, etc…), während jede auch noch so kleine Änderung der Konstruktion rechnerisch überprüft werden muss, um gewährleisten zu können, dass nichts beschädigt wird.

4.2.1 Bedingungen

Anfangs steht natürlich die Frage, wie der Roboter sich eigentlich fortbewegen soll. Die zwei großen Fragen sind:

- Wie viele Beine werden verwendet?
- Wie werden die Beine angetrieben?

Um dies zu beantworten, wurde zuerst überlegt, wie viele Beine benötigt werden, um Gleichgewicht halten zu können. Die Antwort lautete vier. Allerdings ist das Problem mit vier Beinen, dass beim Gehen, durch die Bewegung, das Gleichgewicht nicht mehr so einfach gehalten werden kann. Also wurde die Anzahl der Beine auf sechs erhöht. Da Insekten sich ebenfalls auf sechs Beinen bewegen, wurde der Roboter „…Insectoid (Insektenähnlich)…“ genannt. Um genauere Einblicke über das Verhalten bei der Fortbewegung zu bekommen siehe Design der Walkfiles.

Die zweite Frage war nun, wie die Beine angetrieben werden sollen. Hierfür gibt es verschiedene Möglichkeiten. Die beste ist jedoch eindeutig, sie per Motor anzusteuern. Nachdem Motoren entweder viel zu groß, stark oder teuer wären, wurden ganz einfach Servomotoren verwendet. Diese Art der Motoren ist zwar nicht besonders

Leistungsstark, jedoch klein, leicht und billig. Die nächste Frage war also wie viele Servomotoren benötigt werden, um jedes dieser sechs Beine in die nötigen Richtungen zu bewegen. Nachdem jedes Bein nach vorne, hinten und nach oben und unten bewegt werden muss, ergibt sich, dass pro Bein zwei Servomotoren benutzt werden müssen. Für die Auslegung der Servomotoren siehe Servomotoren-Auslegung.

4.2.2 Handskizze

Im Normalfall wird bei einem Projekt mit einer oder mehreren Handskizzen begonnen. Diese Handskizzen werden, wie der Name schon sagt, per Hand gezeichnet. In ihnen werden die ersten Ideen visualisiert. Zum Beispiel ist es in diesem Projekt so, dass die Anzahl der Beine, die Anzahl der Servomotoren, ihre Position und die allgemeine Form des Roboters festgelegt wurden.

Diese Handskizzen sind sehr hilfreich, da man so eine gute Übersicht darüber bekommt, wie das Endprodukt aussehen wird. Oft werden aufgrund dieser Zeichnung Fehler oder Unstimmigkeiten im Team bemerkt. Somit können Schäden, obwohl noch nicht zu konstruieren begonnen wurde, minimiert werden. Außerdem kann bei jeder Änderung einfach ausgebessert werden, was hilfreich ist, da nicht jedes Mal eine neue Zeichnung gefertigt werden muss.

Diese Handskizzen (Abbildungen 6+7) sind das erste Ergebnis einer Auslegung. Man sieht den Grundkörper, welcher designtechnisch in achteckiger Form gezeichnet ist. Zudem ist festgelegt, wo sich die Beine und Servomotoren befinden. Als Deckel wurde die Platine definiert. Das hat den simplen Grund, dass der Roboter im Prinzip keinen Deckel benötigt und die Platine interessant aussieht. Sie soll anschließend mit Schrauben fixiert werden, damit sie fest und sicher sitzt.

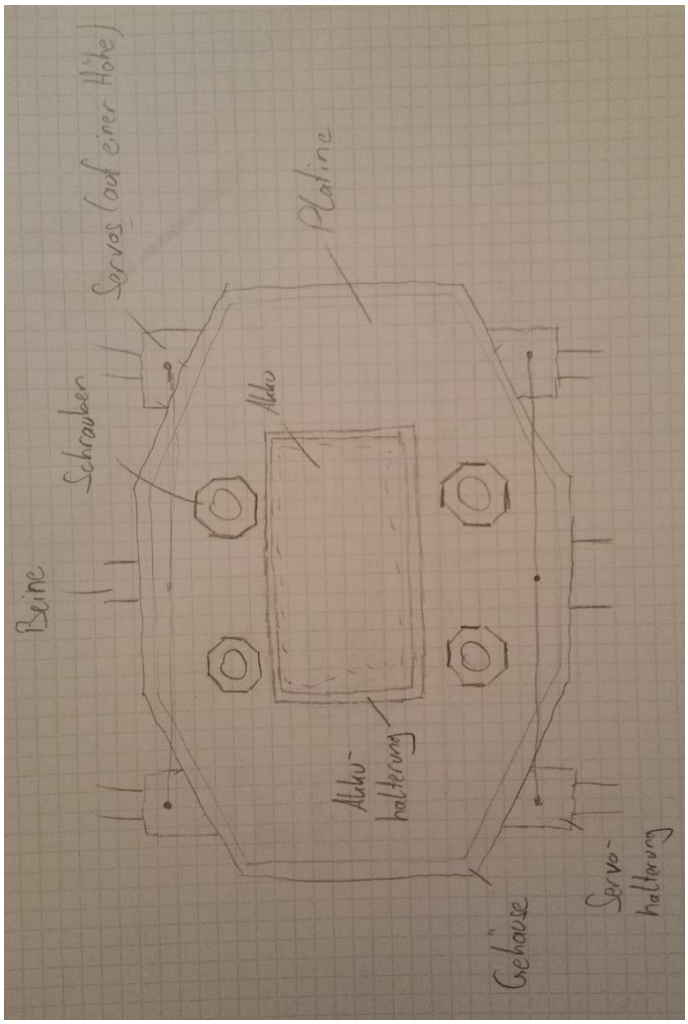

Abbildung 7: Handskizze Gehäuse von oben

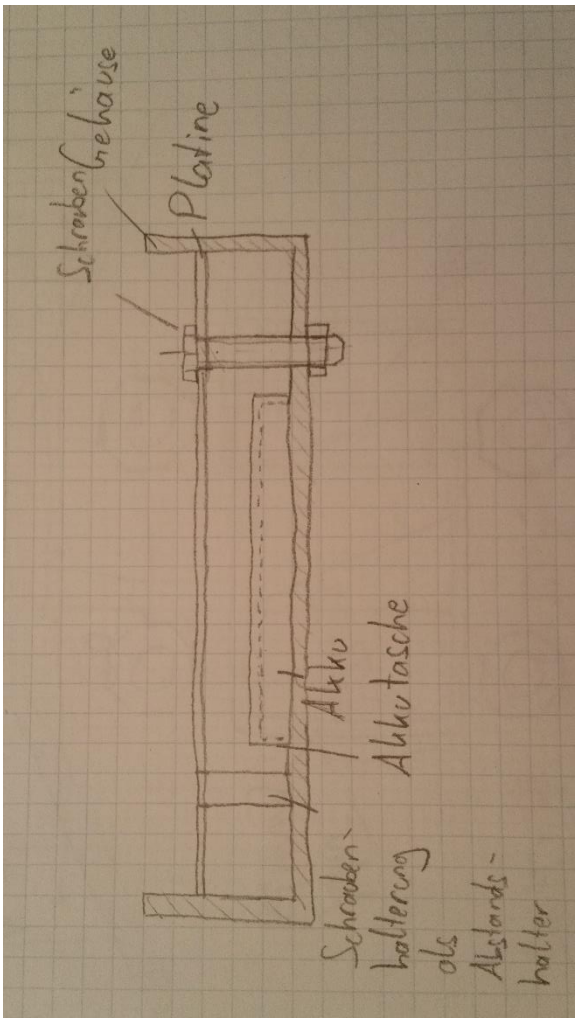

Abbildung 6: Handskizze Gehäuse von rechts

Im folgenden Bild ist dargestellt, wie sich die Beine bewegen werden. Die Pfeile symbolisieren dabei die Bewegungsrichtung.

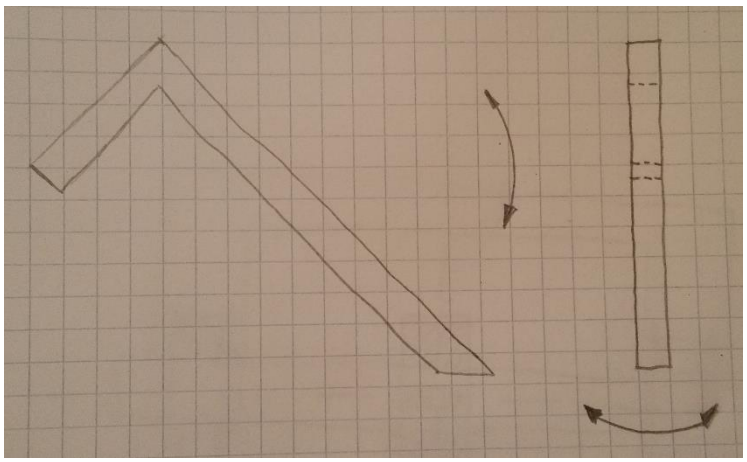

Abbildung 8: Handskizze Beinbewegung

Die linke Ansicht ist die Ansicht von vorne. Dieses Bein wird nach oben und unten schwenken.

Rechts im Bild ist die Ansicht von der Seite. Dieses Bein schwenkt nach rechts und links.

4.2.3 Servomotoren-Auslegung

Bei dieser Diplomarbeit war ein weiterer essenzieller Teil die Auswahl der Servomotoren. Es könnte zum Beispiel ein dicker und somit stabiler Roboter gebaut werden, welcher nur schwer zerstört werden kann. Das Problem ist aber, dass der Roboter damit schwerer wird. Wenn der Roboter schwerer wird, müssen die Servomotoren mehr Leistung aufbringen, um im Gleichgewicht zu bleiben und die Beine zu bewegen. Mehr Leistung heißt aber auch teurer. Um das Projekt so billig wie möglich zu halten, wurde ein Servomotor ausgewählt, welcher der gewünschten Leistung und dem Preis entspricht.

4.2.3.1 Auswahl

Um den besten Servomotor zu finden, wurden drei verschiedene Servos aus einem Katalog (HiTEC, 2015) gesucht und verglichen. In diesem Katalog befinden sich gängige, im Modellbau verwendete Servomotoren.

	Pro	Contra	Preis
HS-225BB	Billig, klein, leicht	Schwach	€ 21,90
HS-225MG	Klein, leicht, Metallgetriebe	Teuer, schwach	€ 29,90
HS-635HB	Sehr stark, Karbonitegetriebe	Groß, sehr teuer, schwer	€ 37,90

Tabelle 10: Servomotor Auswahl

Anhand dieser Liste kann erkannt werden, dass für gewisse Leistungen, sehr viel gezahlt werden muss. Da nicht viel Geld für die Servos gezahlt werden wollte, fiel der teuerste weg. Im Anschluss musste nur noch zwischen den ersten beiden entschieden werden. Der Unterschied liegt einzig im Getriebe. Da ein Metallgetriebe nicht unbedingt notwendig war, konnte darauf verzichtet werden und der Servomotor HS-225BB (HiTEC, 2017), hergestellt von „HiTEC“, wurde gewählt. Er stammt aus der Reihe „Mighty Mini“ und wie der Name schon vermuten lässt, hat er für seine Größe erstaunlich viel Kraft. Außerdem wurde dieser Servomotor ausgewählt und verwendet, da er von einem Sponsor (siehe Sponsoren) empfohlen wurde.

4.2.3.2 Berechnung

Da noch nicht sicher war, ob dieser Servo überhaupt die benötigte Leistung aufbringen kann, musste das Ganze noch rechnerisch überprüft werden.

Da der Antrieb schon gewählt wurde, war eine klare Grenze gesetzt, wie schwer der Roboter sein darf und wie lang die Beine sein können. Grund dafür ist, dass die Servomotoren nur ein gewisses Moment (eine Kraft) aufbringen können, um etwas zu bewegen. Nun gibt es zwei Faktoren die wichtig sind. Zum einen das Gewicht (die Gewichtskraft), die gegen das Moment des Servos wirkt und zum anderen der Abstand, den diese Kraft von der Wirkungslinie entfernt ist.

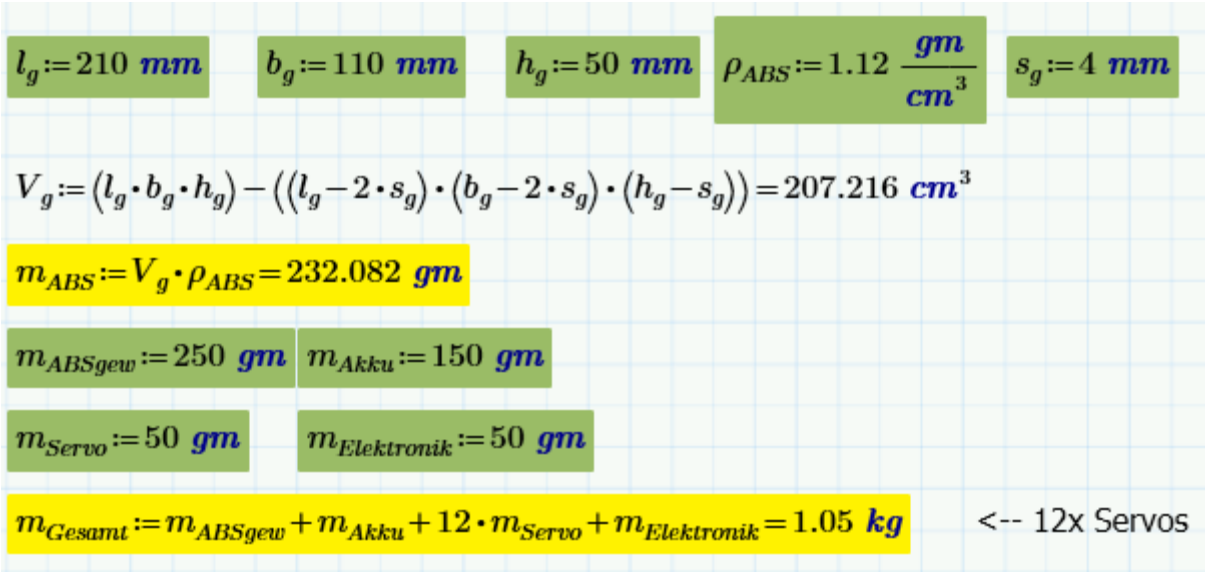

Abbildung 9: Gewichtsberechnung

Um die Gewichtskraft zu ermitteln, musste zuerst das Gesamtgewicht des Roboters geschätzt werden. Dafür mussten die Abmessungen des Gehäuses geschätzt werden. Um die Masse von etwas zu berechnen, werden das Volumen und die Dichte benötigt.

Um das Volumen zu bestimmen wurde l_g als Länge, b_g als Breite, h_g als Höhe und s_g als Wandstärke des Gehäuses definiert. Die Werte wurden zuerst nur geschätzt. Um auf das Volumen zu kommen, wurden nun also zwei Quader definiert. Einer, mit den Maßen 210mm x 110mm x 50mm und der zweite mit den Maßen 202mm x 102mm x 46mm. Der kleine Quader beschreibt den Hohlraum im Gehäuse. Nun wurde der kleinere Quader vom größeren subtrahiert und das Volumen war berechnet. Um Sicherheit zu erlangen wurde eine größere Masse gewählt.

Die Dichte bezieht sich auf das Material ABS. ABS hat eine Dichte von ~1,12 Gramm pro Kubikzentimeter. Genau kann man die Dichte nicht bestimmen, da sie stark von der Qualität des Materials abhängt. Um kein Risiko einzugehen, wurde der größtmögliche Wert gewählt.

Da der Roboter aus mehr als nur dem Gehäuse besteht, muss natürlich noch einiges an Gewicht hinzugerechnet werden. Die größten Bestandteile sind zusätzlich noch der Akku, die Servos und die Elektronik. Da der Servomotor schon ausgewählt wurde, konnte die Masse der Servos einfach bestimmt werden. Ein Servomotor wiegt 27 Gramm. Somit wurde die Masse auf 50 Gramm festgelegt. Um das Gewicht des Akkus zu bestimmen, wurden verschiedene Akkumulatoren verglichen. Aus diesem Vergleich wurde ein Mittelwert bestimmt. Somit beläuft sich das Gewicht des Akkus auf ~150 Gramm. Elektronische Bauteile wie Mikroprozessoren, Widerstände und Platinen wiegen sehr wenig. Um diesen Faktor jedoch nicht zu vernachlässigen, wurden 50 Gramm geschätzt.

Die Gesamtmasse errechnet sich nun aus der Masse des Gehäuses, des Akkus, der Elektronik und zwölfmal der Masse der Servomotoren.

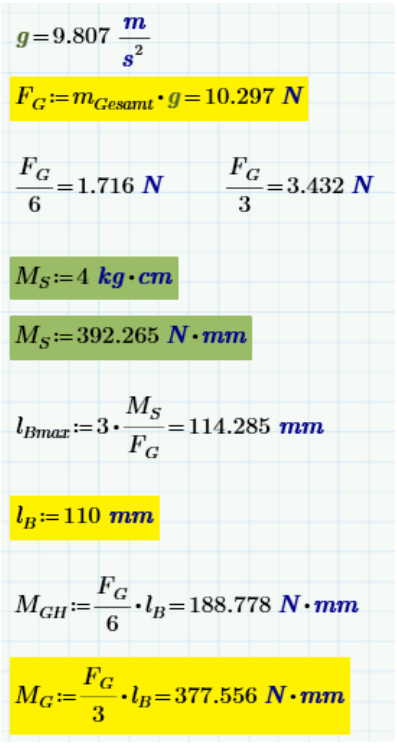

Abbildung 10: Gewichtskraft und Beinlänge

Da die Masse allerdings keine Kraft ist, muss diese umgerechnet werden. Eine Kraft ist definiert durch Masse mal Beschleunigung. Somit muss diese Masse mit der Erdbeschleunigung multipliziert werden und man erhält die sogenannte Gewichtskraft.

Da sechs Beine vorhanden sind, kann die Kraft, welche auf jedes Bein wirkt, errechnet werden. Diese wird als $\frac{FG}{6}$ ausgedrückt. Diese Kraft wird später beim Haltemoment benötigt.

Die wesentlich wichtigere Kraft ist $\frac{FG}{3}$. Diese ist wichtig, da der Roboter beim Gehen immer mindestens drei Beine am Boden haben wird. Somit wirkt die gesamte Gewichtskraft auf drei Beinen.

Um nun also die Beinlänge zu ermitteln, wird die Momenten-Gleichung benutzt. $Moment = Kraft * Normalabstand$.

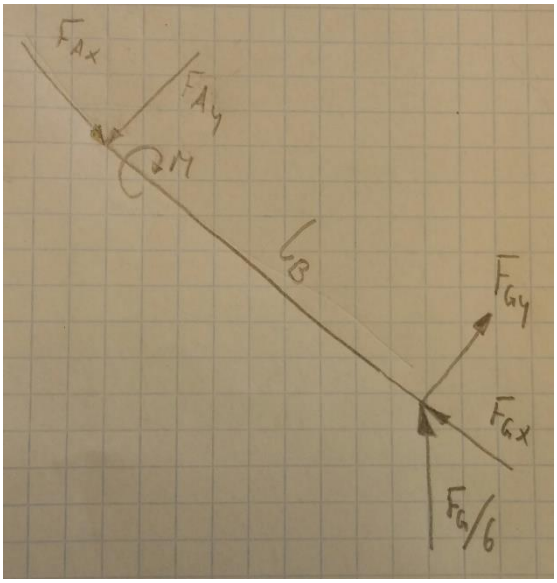

Abbildung 11: Lagerung Bein

Dieser „Normalabstand“ ist die Beinlänge. Das Moment ist jenes Moment, welches der Servo übertragen kann, und die Kraft ist die Gewichtskraft. Somit kann durch Umformen die maximale Beinlänge ermittelt werden.

Da das Bein maximal 114,285mm lang sein darf, wurde die Beinlänge auf 110mm festgelegt. Mit dieser Beinlänge kann man anhand der gleichen Formel errechnen, welches Moment also tatsächlich an den Servos wirkt. Dieses Moment muss kleiner als das Moment, welches der Servomotor übertragen kann, sein.

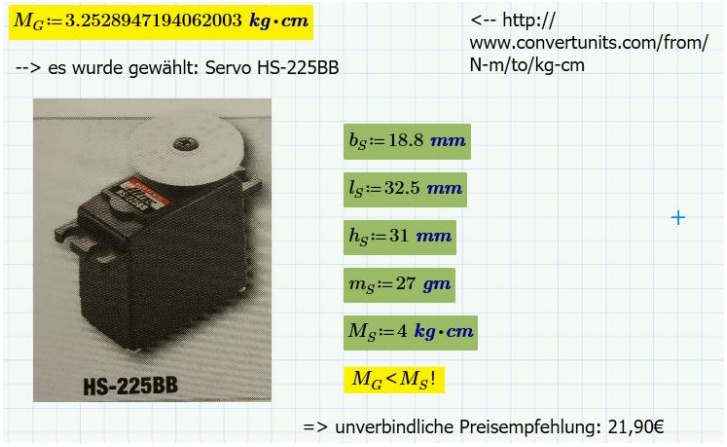

Abbildung 12: Servo HS-255BB

Dieses Moment wird in der Einheit Kilogrammcentimeter angegeben. Um von den Newtonmillimetern, mit denen gerechnet wurde auf diese Einheit zu kommen, wurde ein Umrechner im Internet benutzt.

Somit war der Servo rundum berechnet und konnte gewählt werden. Alle Daten wurden zusammengefasst und in die Berechnung eingetragen.

4.2.4 Biegemoment

Das Biegemoment ist ein essentieller Teil jeder Berechnung. Wie der Name schon sagt, berechnet man damit, das Moment, das eine Biegung eines Bauteils auslöst. Dieses Moment wird mit der Formel $Mb = F * l$ berechnet, wobei M_b für das Biegemoment, F für die Kraft und l für den Normalabstand steht. Dieses Moment ist anschließend für weitere Berechnungen sehr wichtig.

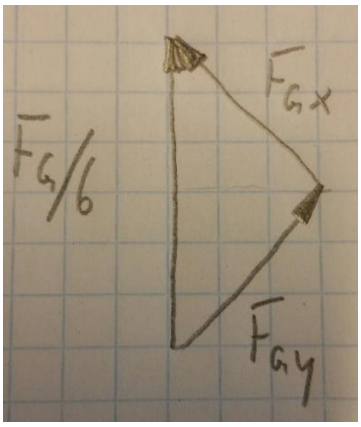

Abbildung 13: Kräftedreieck

Da $\frac{FG}{6}$ aus der Sicht des Beines schräg steht und bei einem Moment nur eine Kraft, die im rechten Winkel auf die Wirkungslinie steht, verwendet werden kann, muss diese Kraft in auf eine x- und eine y-Komponente aufgeteilt werden. Dies kann getan werden, da jede Kraft eine Resultierende ihrer beiden Komponenten ist. Um die Komponenten zu bestimmen, muss ein sogenanntes Kräftedreieck gebildet werden. Anschließend kann mithilfe von Winkelfunktionen (Sinus, Cosinus und Tangens) die Stärke dieser Kräfte bestimmt werden.

In diesem Fall ist der Winkel 45°. Dies bewirkt, dass die x- und y-Komponenten gleich groß sind. Die für das Biegemoment relevante Kraft ist die y-Komponente F_{Gy}. Nun wird die Momenten- Gleichung angewandt und das Biegemoment des Beins bestimmt.

$$\alpha := 45°$$

$$F_{Gx} := \frac{F_G}{6} \cdot \sin(\alpha) = 1.214\ N$$

$$F_{Gy} := F_{Gx}$$

$$M_b := F_{Gy} \cdot l_B = 133.486\ N \cdot mm$$

Abbildung 14: Biegemoment

Um das Moment graphisch darzustellen, wurden Verläufe erstellt. Das bedeutet, dass die angegebene Größe über die Länge dargestellt wird. Um den Biegemomentenverlauf darzustellen, ist es wichtig zuerst den Querkraftverlauf zu erstellen. Die Querkraft ist jene Kraft, die innerhalb eines Bauteils wirkt. In diesem Fall ist es die Gewichtskraft. Dieser Verlauf ist wichtig für den Biegemomentenverlauf, da jeder Wechsel von positiv zu negativ auf der x-Achse dieses Verlaufs, einen lokalen Extremwert bei dem anderen Verlauf bedeutet.

$$Q(x) := \begin{cases} \text{if } x = 0 \\ \quad 0 \\ \text{if } 0 < x \le l_B \\ \quad F_{Gy} \\ \text{if } x = l_B \\ \quad 0 \end{cases}$$

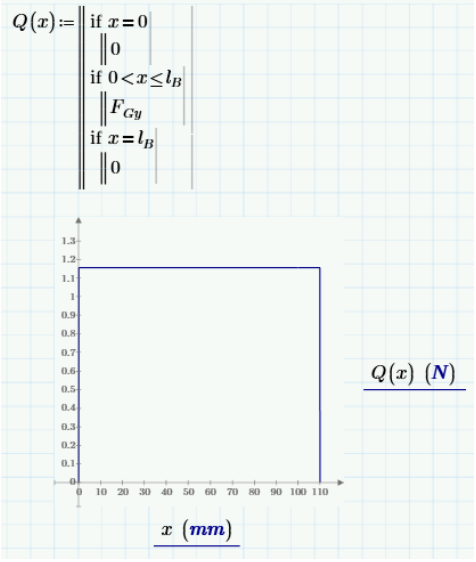

Abbildung 15: Querkraftverlauf

Q steht in diesem Graphen für die Querkraft und ist auf der y-Achse zu sehen. Q wird in Newton angegeben. Auf der x-Achse ist x zu sehen. x beschreibt den Abstand vom Nullpunkt und wird in Millimetern angegeben.

Dieser Graph zeigt die Veränderung des Biegemoments über die Länge des Beins. Da im Querkraftverlauf kein Übergang von positiv zu negativ besteht, ist hier kein lokaler Extremwert (außer Beginn und Schluss) vorhanden.

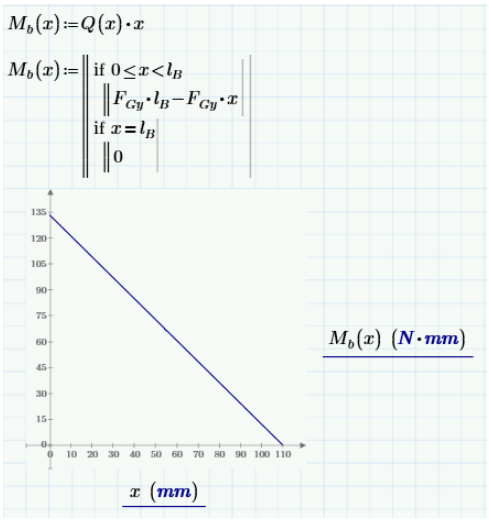

Abbildung 16: Biegemomentenverlauf

M_b steht in diesem Graphen für das Biegemoment und ist auf der y-Achse zu sehen. Das Moment wird in Newtonmillimeter angegeben. x steht wie im oberen Verlauf für den Abstand zum Nullpunkt und wird in Millimetern angegeben.

Um nun das maximale Biegemoment zu bestimmen, kann man einfach eine Funktion des Programmes benutzen. Einfach das Biegemoment an der höchsten Stelle des Graphs anzeigen lassen. In diesem Fall wäre das an der Stelle „0 Millimeter“.

Dadurch erhält man M_{bmax} = 133,486 Newtonmillimeter.

$$M_{bmax} := M_b(0\ mm) = 133.486\ N \cdot mm$$

Abbildung 17: Maximales Biegemoment

4.2.5 Bester Querschnitt des Beins

Obwohl das Biegemoment errechnet wurde, sagt das noch nichts darüber aus, ob ein Bein diese Belastung aushält. Wichtig sind sowohl Form- und Materialwahl. Der Grund dafür ist, dass verschiedene formbasierte Querschnitte unterschiedlich, auf eine Belastung reagieren. Da das Material bereits gewählt wurde, bleibt also nur noch übrig, den Querschnitt zu finden, welcher die Kraft am besten überträgt.

Begonnen wurde nun unterschiedliche Querschnitte zu wählen. Hierfür wurden drei verschiedene Formen gewählt:

- Quadrat: Ein Quadrat hat den Vorteil, dass es einfach herzustellen ist und gut aussieht. Allerdings ist es weniger belastbar als ein aufrecht stehendes Rechteck.

- Rechteck: Ein Rechteck (welches höher als breit ist) sieht nicht unbedingt gut aus, eignet sich allerdings sehr gut für Belastungen.
- Kreis: Ein Kreis als Querschnitt sieht sehr gut aus, ist allerdings nicht nur schlecht für diese Art der Belastung geeignet, sondern ist auch schwerer zu fertigen.

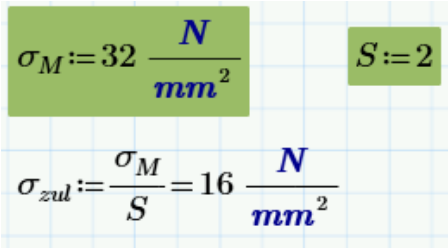

Abbildung 18: Zulässige Belastbarkeit

Die Belastbarkeit eines Materials erfährt man aus einem Tabellenbuch (Wittel, Muhs, Jannasch, & Voßiek, Roloff/Matek Maschinenelemente Tabellenbuch, 2011). Zu finden ist dieser auf der Seite 13 in Tabelle 1-4. Diese Tabelle ist für Kunststoffe bestimmt. ABS ist in dieser Tabelle in der Kategorie Thermoplaste zu finden und wird als schlagzäh, kratzfest und mit hoher Formbeständigkeit beschrieben. Weiters ist es perfekt für Gehäuse geeignet. Der gesuchte Wert ist unter Festigkeit zu finden. Dieser wurde durch etwaige Zugversuche in Erfahrung gebracht.

Um sicherzustellen, dass das Material gegen Belastungen geschützt ist, die nicht eingerechnet werden können (Roboter fällt vom Tisch auf den Boden, etc...) wurde eine Sicherheit miteingerechnet. Im Normalfall wird in etwa eine Sicherheit von 1,5 bestimmt. In diesem Fall wurde 2 als Sicherheit gewählt, da sichergestellt werden wollte, dass tatsächlich nichts bricht. Wenn nun also der Tabellenwert durch die Sicherheit dividiert wird, erhält man die zulässige Spannung, welche von den auftretenden Spannungen nicht überschritten werden darf.

4.2.5.1 Quadrat

Als erstes wurde ein Quadrat als Querschnitt bestimmt. Die Seitenlänge betrug 5 Millimeter. Dieser Wert ist nur geschätzt, würde aber verwendet werden, wenn dieser Querschnitt gewählt werden würde.

Nun muss die gesamte Spannung ermittelt werden, die auf dieses Bauteil, mit diesem Querschnitt, wirkt. Hierfür musste zuerst die Biegespannung bestimmt werden. Um diese zu errechnen, muss das maximale Biegemoment durch das Widerstandsmoment des Quadrats, dividiert werden. Dieses errechnet sich durch: Seitenlänge hoch drei, dividiert durch sechs (für genauere Information siehe Biegespannung). Da aber auch

$a_Q := 5\ mm$

$W_{xQ} := \frac{a_Q^3}{6} = 20.833\ mm^3$

$\sigma_{bQ} := \frac{M_{bmax}}{W_{xQ}} = 6.407\ \frac{N}{mm^2}$

$A_Q := a_Q \cdot a_Q = 25\ mm^2$

$\sigma_{DQ} := \frac{F_{Gx}}{A_Q} = 0.049\ \frac{N}{mm^2}$

$\sigma_{VQ} := \sigma_{bQ} + \sigma_{DQ} = 6.456\ \frac{N}{mm^2}$

$\sigma_{VQ} < \sigma_{zul}$

Abbildung 19: Querschnitt Quadrat

Druck auf das Bauteil wirkt, muss die Druckspannung ebenfalls berechnet werden. Um diese zu errechnen muss nur die wirkende Kraft durch die Querschnittsfläche dividiert werden (für genauere Information siehe Druck). Um diese Spannungen zusammenzufügen, muss die Vergleichsspannung gebildet werden (für genauere Information siehe Vergleichsspannung). Die Vergleichsspannung beläuft sich auf 6,456 Newton pro Quadratmillimeter und ist somit kleiner als die maximal zulässige Spannung. Dies bedeutet, dass das Bein der Belastung standhält.

4.2.5.2 Rechteck

Als zweites wurde ein Rechteck als Querschnitt bestimmt. Die Höhe dieses Rechtecks wurde auf 10 und die Breite auf 5 Millimeter festgelegt. Wie schon beim Quadrat, wurden diese Werte nur geschätzt und würden, wenn ein Rechteck als Querschnitt gewählt werden würde, verwendet werden.

$a_R := 10\ mm$

$b_R := 5\ mm$

$W_{xR} := \frac{b_R \cdot a_R^2}{6} = 83.333\ mm^3$

$\sigma_{bR} := \frac{M_{bmax}}{W_{xR}} = 1.602\ \frac{N}{mm^2}$

$A_R := a_R \cdot b_R = 50\ mm^2$

$\sigma_{DR} := \frac{F_{Gx}}{A_R} = 0.024\ \frac{N}{mm^2}$

$\sigma_{VR} := \sigma_{bR} + \sigma_{DR} = 1.626\ \frac{N}{mm^2}$

$\sigma_{VR} < \sigma_{zul}$

Abbildung 20: Querschnitt Rechteck

Die Grundlegende Formel, um die auftretende Spannung zu errechnen ist wieder die gleiche. Allerdings ist das Widerstandsmoment eines Rechtecks anders. Der Grund dafür ist an einem Beispiel zu erklären.

Ganz einfach ist es an einem Blatt Papier zu erkennen. Nun muss der Querschnitt der Breite, mit den Maßen 210x1mm, betrachtet werden. Das Papier muss nun an einer der kürzeren Seiten gehalten werden. Wird es horizontal gehalten, wird es sich sofort durchbiegen. Wenn es allerdings vertikal gehalten wird, wird es sich nicht durchbiegen. Also ist einfach zu sehen: Wenn der Querschnitt viel höher als breit ist, wird es mehr Kraft übertragen können.

Somit muss nun die Formel: $Widerstandsmoment = \frac{Breite * Höhe^2}{6}$ verwendet werden. Wenn man die Berechnung weiterführt, erhält man eine Vergleichsspannung von 1,626

Newton pro Quadratmillimeter. Diese Spannung ist ebenfalls kleiner als die maximal zulässige Spannung. Ein Bein mit diesem Querschnitt würde den Belastungen also auch standhalten.

4.2.5.3 Kreis

$$d_K := 5\ mm$$

$$W_{xK} := \frac{d_K^3 \cdot \pi}{32} = 12.272\ mm^3$$

$$\sigma_{bK} := \frac{M_{bmax}}{W_{xK}} = 10.877\ \frac{N}{mm^2}$$

$$A_K := \frac{d_K^2 \cdot \pi}{4} = 19.635\ mm^2$$

$$\sigma_{DK} := \frac{F_{Gx}}{A_K} = 0.062\ \frac{N}{mm^2}$$

$$\sigma_{VK} := \sigma_{bK} + \sigma_{DK} = 10.939\ \frac{N}{mm^2}$$

$$\sigma_{VK} < \sigma_{zul}$$

Abbildung 21: Querschnitt Kreis

Als drittmöglicher Querschnitt wurde ein Kreis gewählt. Als Durchmesser wurden fünf Millimeter bestimmt. Dieser Wert wurde, wie auch schon bei den vorherigen Berechnungen, nur geschätzt und würde verwendet werden, wenn ein Kreis als Querschnitt bestimmt werden würde.

Die Berechnung läuft im Grunde wieder gleich ab. Die einzigen Unterschiede sind das Widerstandsmoment, welches mit der Formel: $Widerstandsmoment = \frac{Durchmesser^3 * Pi}{32}$ errechnet wird, und die Fläche, welche nun die Fläche eines Kreises ist.

Die Vergleichsspannung ergibt nun 10,939 N/mm^2, was kleiner als die maximal zulässige Spannung ist. Somit würde ein Bein mit diesem Querschnitt der Belastung ebenfalls standhalten.

4.2.6 Biegespannung

Da nun die Querschnitte bestimmt wurden, muss die Biegespannung für ein Bein ermittelt werden. Um diese zu ermitteln muss ein Querschnitt gewählt werden. In der Berechnung kann man sehen, dass jeder dieser Querschnitte der Belastung standhält. Dies bedeutet nun, dass es mehr oder weniger egal ist, welcher gewählt wird. In diesem Fall wurde ein Quadrat gewählt. Die Gründe dafür sind, dass ein Kreis ist im 3D-Drucker relativ schwer und mühsam zu produzieren ist. Dementsprechend ist es klüger keinen Kreis zu wählen. Ein Rechteck würde zwar größeren Belastungen besser standhalten und wäre somit die beste Wahl. Da ein Quadrat allerdings (in den Augen des Teams) besser aussieht und den auftretenden Belastungen locker gewachsen ist, wurde es gewählt.

$$a_B := 5\ mm$$

$$b_B := 5\ mm$$

$$W_y := \frac{a_B \cdot b_B^2}{6} = 20.833\ mm^3$$

$$\sigma_b := \frac{M_{bmax}}{W_y} = 6.407\ \frac{N}{mm^2}$$

Abbildung 22: Biegespannung

Nun muss also das Widerstandsmoment ermittelt werden. Das Widerstandsmoment beschreibt den Widerstand gegenüber einer Belastung, der aus der Geometrie (Form) eines Bauteils entsteht. Für ein praktisches Beispiel siehe Rechteck. Bei Viereckigen Körpern wird diese Formel verwendet: $Widerstandsmoment = \frac{Breite * Höhe^2}{6}$. Da bei einem Quadrat, die Höhe und Breite ident sind, könnte man auch Seitenlänge^3 verwenden. Aufgrund dieser Formel ergeben sich für das Widerstandsmoment pro Bein 20,833mm³.

Um nun die Biegespannung zu ermitteln muss das Auftretende Biegemoment durch das Widerstandsmoment dividiert werden. Da nicht die Spannung an einem gewissen Punkt, sondern die maximal auftretende Spannung ermittelt werden soll, wird auch das maximal auftretende Biegemoment verwendet. Somit ergibt sich eine Biegespannung von 6,407 N/mm².

4.2.7 Biegelinie

$$M_b = F \cdot (l_B - x) \qquad y(0) = 0 \qquad => C_2 = 0$$

$$y'' = -\frac{M_b}{E \cdot I} \qquad y'(0) = 0 \qquad => \quad E \cdot I \cdot 0 = -F \cdot l_B \cdot 0 + F \cdot \frac{0^2}{2} + C_1$$

$$E \cdot I \cdot y'' = -F \cdot (l_B - x) = -F \cdot l_B + F \cdot x \qquad 0 = -0 + 0 + C_1 \quad => C_1 = 0$$

$$E \cdot I \cdot y' = -F \cdot l_B \cdot x + F \cdot \frac{x^2}{2} + C_1$$

$$E \cdot I \cdot y = -F \cdot l_B \cdot \frac{x^2}{2} + \frac{1}{2} \cdot F \cdot \frac{x^3}{3} + C_1 \cdot x + C_2$$

$$E \cdot I \cdot y = -\frac{3}{6} \cdot F \cdot l_B \cdot x^2 + \frac{1}{6} \cdot F \cdot x^3$$

$$6 \cdot E \cdot I \cdot y = F \cdot (-3 \cdot l_B \cdot x^2 + x^3)$$

$$y \cdot \frac{6 \cdot E \cdot I}{F \cdot l_B} = -3 \cdot x^2 + \frac{x^3}{l_B}$$

$$y \cdot \frac{6 \cdot E \cdot I}{F \cdot l_B^3} = -3 \cdot \left(\frac{x}{l_B}\right)^2 + \left(\frac{x}{l_B}\right)^3$$

$$y = -\frac{F \cdot l_B^3}{6 \cdot E \cdot I} \cdot \left[3 \cdot \left(\frac{x}{l_B}\right)^2 - \left(\frac{x}{l_B}\right)^3\right]$$

Abbildung 24:Biegelinie Berechnung

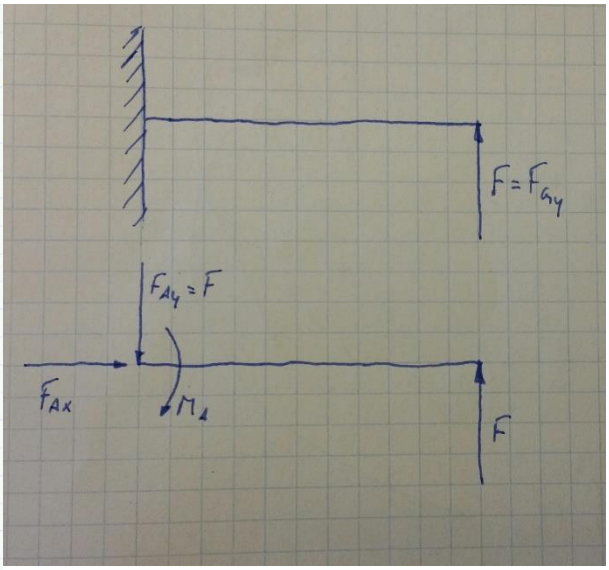

Abbildung 23: Biegelinie Skizze

Um die genaue Durchbiegung zu ermitteln müsste eine andere, kompliziertere Formel verwendet werden. Da diese weit über das Wissen eines Schülers hinausgehen würde, wurde eine Annäherung verwendet.

Um nun zu ermitteln wie, beziehungsweise wie weit, sich ein Bein durchbiegt, muss die Biegelinie ermittelt werden. Die Biegelinie wird durch die Krümmung ermittelt. Somit ergibt sich die Formel für y“. Da aber y benötigt wird, muss zweimal integriert werden.

Nachdem dies erledigt ist, müssen die Anfangsbedingungen bestimmt werden. Wichtig hierfür ist die Einbausituation. In diesem Fall ist es eine einseitige Einspannung mit einfacher Belastung. Daraus ergibt sich, dass die Durchbiegung an der Stelle „0“ gleich null sein muss, da eine Einspannung vorhanden ist. Weiters ist die Steigung an der Stelle „0“ auch null, da dort keine Steigung vorhanden ist.

Durch weiteres Umformen ergibt sich dann die Formel für die Kurve der Durchbiegung. Die übrigen Unbekannten sind „E“ und „I“. „E“ steht in diesem Fall für das E-Modul. Das Elastizitätsmodul hat mit der Verformung eines Materials zu tun. In diesem Fall beträgt dieser Wert für ABS 2300 N/mm². „I“ steht in diesem Fall für das Flächenträgheitsmoment. Dies wird ähnlich wie das Widerstandsmoment für die formabhängige Berechnung von Belastungen verwendet. Die Formel: $I = \frac{Breite * Höhe^3}{12}$, ergibt dementsprechend 52,083 mm⁴. Graphisch dargestellt sieht das dann so aus:

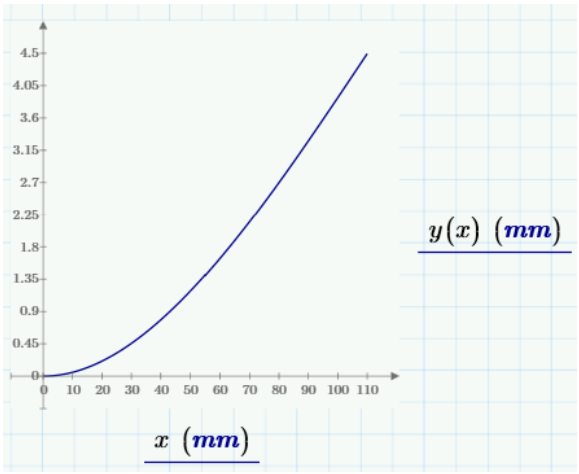

Abbildung 25: Biegelinie Graph

Auf der y-Achse sieht man die Durchbiegung und auf der x-Achse die Länge des Beins.

Die maximale Durchbiegung (also bei der Gesamtlänge des Beins) beträgt 4.494 mm. Das Bein wird sich also bis zu diesem Wert (ohne zu brechen!) durchbiegen können.

4.2.8 Knickung

Die Knickung ist jener Wert, welchen ein Material aushält, ohne zu zerbrechen. Für diese Knickung werden (nach Euler) vier Fälle festgelegt. Hier zutreffend ist der erste Fall, da eine einseitige Einspannung vorliegt.

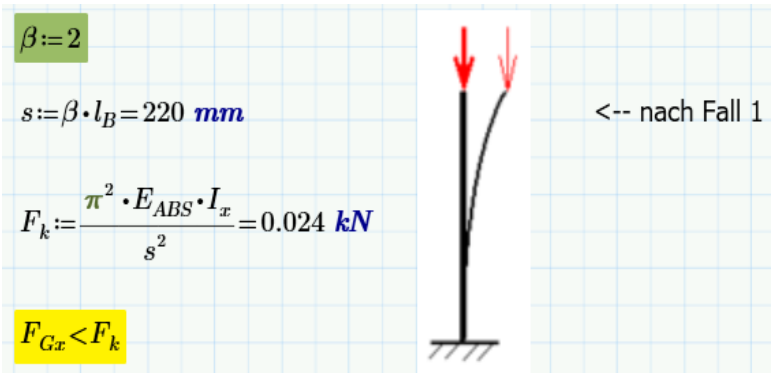

Abbildung 26: Knickung

Aus diesem Grund beträgt β gleich 2. β wird für die Knicklänge „s“ benötigt. Die Knicklänge beträgt in diesem Fall 220mm.

Um nun die Knickkraft F_k ausrechnen zu können, werden wieder das E-Modul und das Flächenträgheitsmoment benötigt. Daraus ergibt sich für die Knickkraft der Wert 0,024 kN. Solange also F_{Gx} kleiner ist als die Knickkraft, wird kein Bein brechen.

4.2.9 Druck

$$A_B := a_B \cdot b_B = 25\ mm^2$$

$$\sigma_D := \frac{F_{Gx}}{A_B} = 0.049\ \frac{N}{mm^2}$$

Abbildung 27: Druck

Druck beschreibt im Allgemeinen den Widerstand, den das Material gegen das Material der Kraft entgegensetzt, um nicht zusammengedrückt zu werden. In diesem Fall bedeutet das, dass das Bein versucht nicht zusammengedrückt zu werden.

Diese Spannung errechnet sich durch: Wirkende Kraft dividiert durch Auftrittsfläche. Also Gewichtskraft in x-Richtung durch Querschnittsfläche des Beins. Dies ergibt anschließend 49 mN/mm².

4.2.10 Vergleichsspannung

$$\sigma_V := \sigma_b + \sigma_D = 6.456\ \frac{N}{mm^2}$$

$$\sigma_{zul} > \sigma_V$$

Abbildung 28: Vergleichsspannung

Eine Vergleichsspannung muss immer dann gebildet werden, wenn mehr als eine Spannung vorhanden ist, die auf ein Bauteil wirkt. Üblicherweise wird sie dementsprechend verwendet, wenn verschiedene Spannungsarten auftreten (Torsion, Druck, Biegung, ...).

In diesem Fall ist es wichtig zu sehen, ob das Bauteil nicht nur den Einzelspannungen standhält, sondern auch der Gesamtbelastung.

Wenn Spannungen gleicher Art (die zwei verschiedenen Spannungsarten sind Normal- und Scherspannung) zusammengerechnet werden, kann man sie einfach addieren. Eine Normalspannung zeichnet sich dadurch aus, dass die Spannungskomponenten senkrecht auf die betrachtete Fläche wirken. Da dies sowohl bei der Biege- als auch bei der Druckspannung der Fall ist, werden diese addiert. Somit ergibt sich eine Vergleichsspannung von 6,4 N/mm². Um zu überprüfen, ob dieser Wert innerhalb der

Toleranz liegt, muss er mit der zulässigen Spannung verglichen werden. Da sie in diesem Fall größer als die vorhandene Spannung ist, kann davon ausgegangen werden, dass das Bein jeglicher vorhandenen Belastung standhalten wird.

4.3 Konstruktion

Die Konstruktion ist nun der Schritt, bei welchem auch tatsächlich etwas entsteht, das, nach der Fertigung, berührt werden kann. Es geht also vom Theoretischen, zum Praktischen.

Grundsätzlich ist es nun die Aufgabe der Konstruktion, die errechnet Maße, zu einem Bauteil umzuwandeln, welches herstellbar ist und möglichst gut aussieht. Wie schon erwähnt, hängt sie sehr eng mit der mechanischen Auslegung zusammen. Oft wird erst in der Konstruktion bemerkt, dass etwas nicht notwendig ist, oder eventuell nicht passt, da etwas erstellt wird, das auch tatsächlich gesehen werden kann.

Eine der größten Herausforderungen der Konstruktion bei diesem Projekt war, die Servomotoren so zu platzieren, dass die Beine die nötige Bewegung durchführen können. Die Motoren mussten nämlich:

- Genug Platz haben um die Beine zu bewegen,
- Eine Aufhängung haben (also nirgends aufliegen),
- Richtig ausgerichtet sein,
- Ein- und ausbaubar sein.

Nachdem sich im Laufe der Konstruktion und der gesamten Diplomarbeit noch einiges bei dem Modell ändern wird, ist es sinnvoll, verschiedene Versionen des Roboters zu erstellen. Zum Beispiel wird die erste Idee erstellt und als Version_1 abgespeichert. Wenn sich entweder die Idee selbst, oder einfach nur das Aussehen ändert, kann die Datei nun als Version_2 gespeichert werden. Falls also nochmal auf die erste Idee oder das erste Design zurückgegriffen werden will, muss nur eine ältere Version geöffnet werden. Diese Modelle wurden mit Creo Parametric 2.0 (PTC, 2017) erstellt.

4.3.1 Version 1

Begonnen wurde damit, das Gehäuse aufzubauen. Da die Rohmaße bereits aus der Berechnung gegeben waren, war zumindest ein Anfang gesetzt. Nachdem ein rechteckiger Kasten allerdings nicht sonderlich gut aussieht wurde das Gehäuse

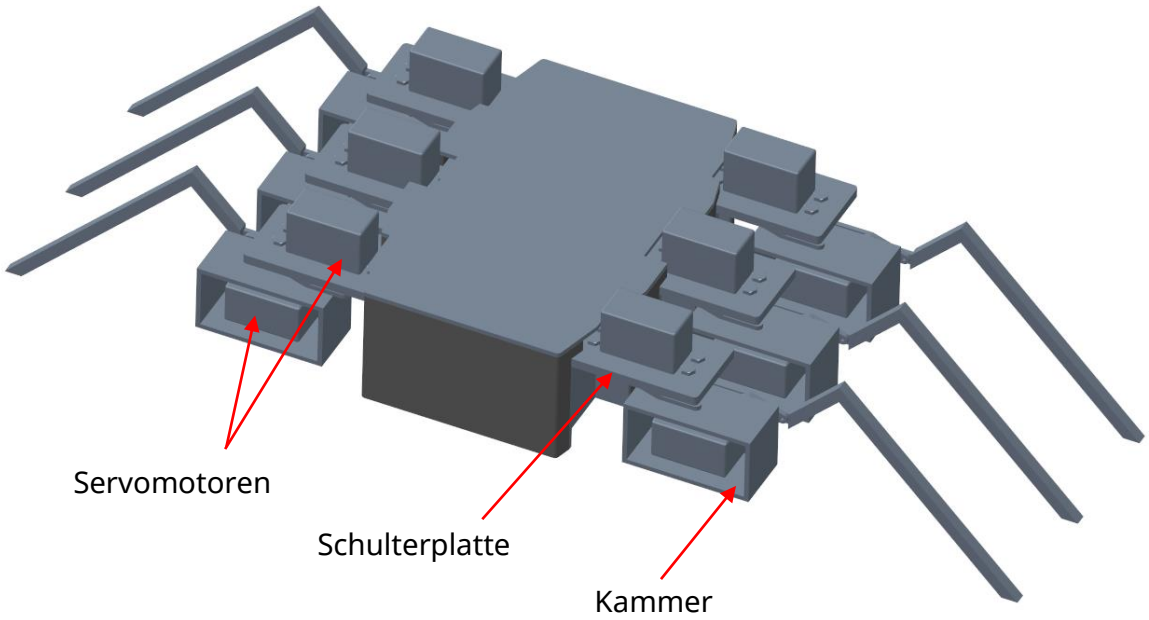

Abbildung 29: Roboter Version 1

designtechnisch zu einem Achteck geändert. Die schrägen Seitenwände haben einen Winkel von 70°.

Da dies nicht die finale Version war, sondern eine Art Versuchsaufbau, wurden noch nicht die richtigen Schrauben verwendet. Trotzdem werden für die Befestigung der Servos M3 Schrauben verwendet, welche mithilfe von Beilagscheiben und Muttern festgeschraubt werden.

Die erste Idee war es, die Servos mittels Schulterplatten zu platzieren. Somit würden die schwenkenden Servos weit vom Gehäuse entfernt sein und Bewegungsfreiheit garantieren. Die hebenden

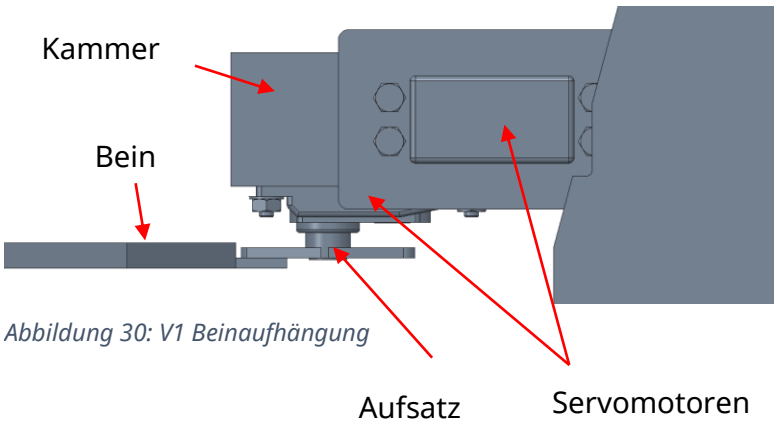

Abbildung 30: V1 Beinaufhängung

Motoren wären dann in Kammern angebracht worden, welche an den anderen Servomotoren angeschraubt sind.

Zum Erstellungszeitpunkt dieses Modells waren die Servos allerdings noch nicht geliefert. Deswegen konnte nicht mit den verschiedenen Aufsätzen experimentiert werden. Somit wurde der Standardaufsatz gewählt. Das Bein sollte einfach mit Schrauben am Servo befestigt werden.

Der Nachteil davon war allerdings, dass sehr viel Material benötigt wird und somit das Modell auch schwer ist. Da das Aussehen keinem der Teammitglieder gefallen hat, wurde die Übersetzungsvariante geändert. Des Weiteren ist der Roboter viel zu flach und breit. Dies hätte seine Bewegungsfreiheit einschränken können, da er zu knapp beim Boden ist. Außerdem hat das Gehäuse einen relativ komplizierten Aufbau (ähnlich einem menschlichen Brustkorb), welcher zwar ohne Probleme hergestellt werden kann, aber einfach nur Verschwendung wäre.

4.3.2 Version 2

Diese Version wurde in erster Linie erstellt, um die Fehler der ersten Version auszubessern und etwaige neue Fehler zu zeigen. Das Ziel der neuen Version war also anhand der durchgeführten Verbesserungen bzw. des Ergebnisses zu sehen, was alles möglich ist und wie das Modell aussehen kann. Dies sollte anschließend als Inspiration für die dritte Version dienen.

Als Modifizierung wurden die Schulterplatten nicht oben vom Roboter weggeführt, sondern unten, auf gleicher Linie wie der Boden des Gehäuses. Diese Verschiebung hat den Vorteil, dass der Roboter weiter vom Boden entfernt ist. Daraus ergab sich, dass das Gehäuse eine simple Außenkontur bekommen kann.

Da die Kammern auf diese Weise auch unpraktisch sind, musste daran etwas geändert werden. Die einfachste Lösung ist es, einfach alle ungenutzten Flächen zu entfernen. In diesem Fall wurden dementsprechend jene Flächen entfernt, mit denen kein Servomotor in Kontakt ist. Also blieben nur noch die Oberseite und die Seitenwand übrig.

Des Weiteren wurde an der Beinaufhängung gearbeitet. Die Verbindung zwischen Bein und Motor war nicht perfekt gelöst. Um mehr Stabilität zu erreichen und um sicherzustellen, dass nichts bricht, wurde die Form des Beinendes verändert. Das Ziel des neuen Designs ist es, den Servo-Aufsatz zu umschließen. Im Unterschied zu vorher sollen die Schrauben jetzt nicht im herkömmlichen Sinne, sondern als Bolzen verwendet werden. Diese werden ganz einfach auf einer Seite mittels Mutter festgeschraubt.

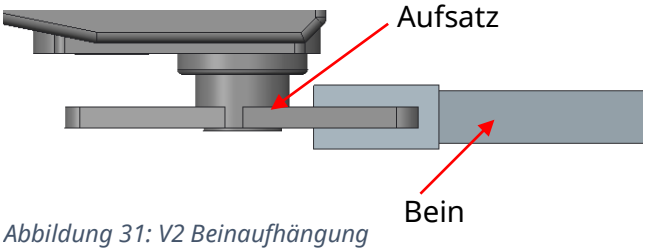

Abbildung 31: V2 Beinaufhängung

4.3.3 Version 3

Wie schon im letzten Punkt erwähnt, ist diese dritte Version die endgültige. Trotzdem musste in dieser Version noch einiges verbessert werden. Da viel an Gewicht eingespart

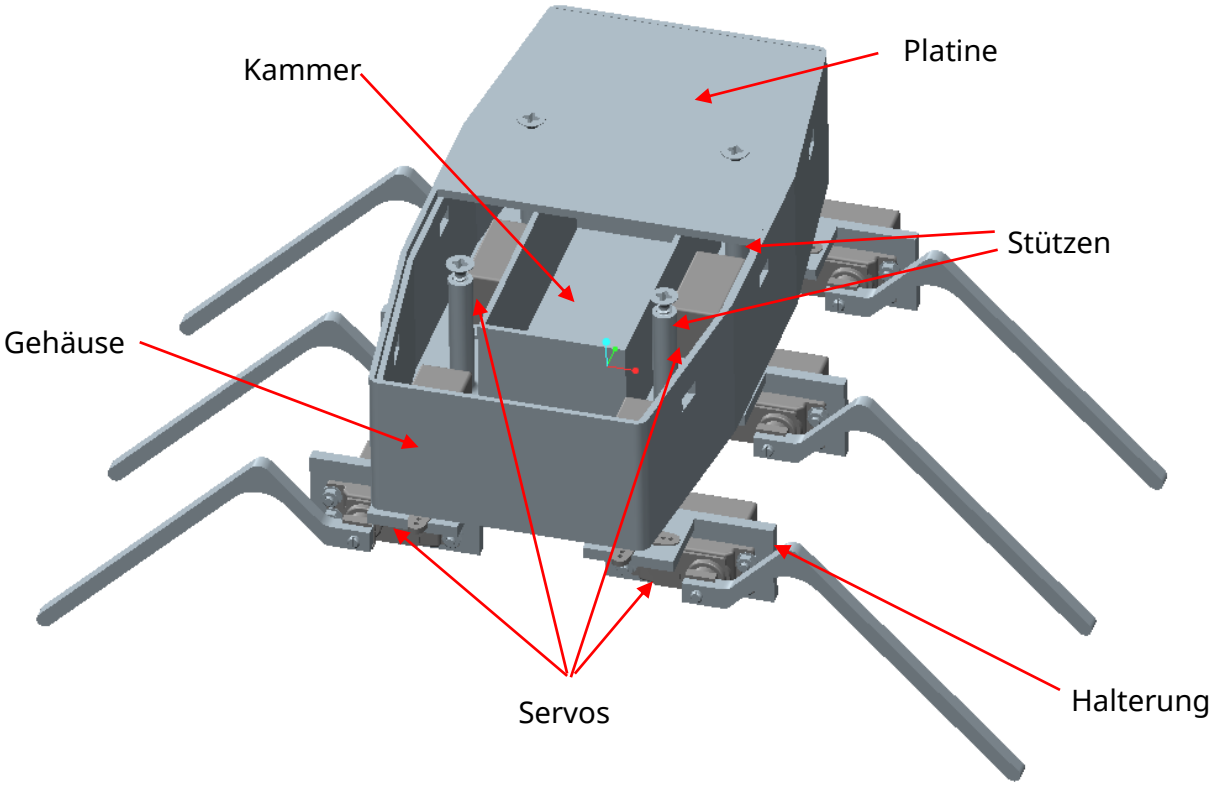

Abbildung 32: Roboter Version 3

werden konnte, war es möglich das Gehäuse noch zu vergrößern. Bisher war der Roboter nämlich kleiner als erwartet. Endgültig erreicht er die Maße 210x110x50mm. Das bedeutet, dass er in etwa so lang, wie ein Papier im A4 Format breit ist.

Auf Abbildung 32 ist die Platine in der Mitte aufgeschnitten, damit man den Innenteil sehen kann. In Wirklichkeit ist das Gehäuse oben geschlossen.

Das Gehäuse ist in drei, gleich lange, Teile geteilt, welche jeweils 70mm lang sind. Die Wandstärke beträgt 4mm.

Der große Unterschied zu der letzten Version ist, dass die Servos nun in den Innenteil verschoben wurden. Das bringt mehrere große Vorteile:

- Der Roboter wird um einiges schmäler und sieht somit besser aus,
- Die Motoren haben mehr Bewegungsfreiheit, da sie sich unterhalb des Gehäuses bewegen und somit uneingeschränkt sind,
- Es wird kein zusätzliches Material benötigt, sondern sogar gespart,
- Der Einbau ist einfacher gestaltet.

Im Innenteil sind die Servos mittels M3x16 Schrauben befestigt, welche an der Rückseite mithilfe von Muttern festgeschraubt werden. Ursprünglich war geplant, Sechskantschrauben zu verwenden. Diese Schrauben sind in dieser Größenordnung allerdings sehr schwer zu bekommen und außerdem ziemlich teuer. Als Ersatz wurden Kreuz-Schlitz Schrauben mit Halbrundkopf verwendet.

Die Besonderheit an der Einbausituation ist, dass die Servos nicht nur auf einer Linie liegen, sondern die Achsen auch gleich weit entfernt sind. Das erleichtert das Bewegungsmuster zu erstellen.

Die Kammer in der Mitte hat den Zweck, den Akku zum einen von den anderen Bauteilen zu trennen und zum anderen festzuhalten, damit er nicht verrutscht. Weiters wurden in die Wände des Gehäuses viereckige Aussparungen eingearbeitet, damit sowohl die Kabel des Akkus, als auch die der Servomotoren vom Innenteil, nach außen auf die Platine reichen können.

Um die Platine vor dem Verrutschen zu sichern, wurden zwei Mechanismen eingebaut. Zum einen wurde eine Stufe in die Außenwände des Gehäuses eingebaut, damit die Platine einen festen Sitz hat. Da sie allerdings noch immer, bei zum Beispiel einem Ruck, aus dieser Halterung herausspringen kann, musste die Platine durch etwas hinuntergedrückt werden. Dies wurde gelöst, indem beim 3D-Druck Stützen

miteingearbeitet wurden, in welche selbstschneidende Schrauben gedreht werden können. In die Platine selbst mussten nur noch Bohrungen gemacht werden und somit ist sie gesichert.

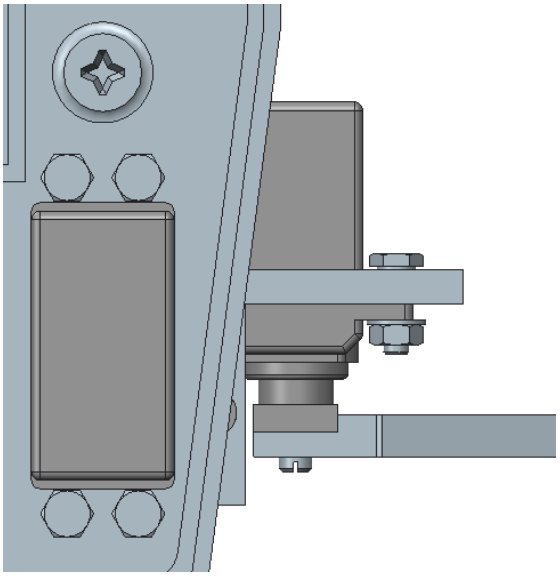

Abbildung 33: V3 Beinaufhängung

Nachdem die zweite Version der Beinaufhängung zwar ganz gut war, war das Problem, dass sie schwer zu drucken war. Als Problemlösung wurde diesmal nicht die Form des Beins, sondern des Servos betrachtet. Anstatt des ursprünglichen Aufsatzes, wird nun einer verwendet, welcher einem U-Profil ähnelt. Dieser Aufsatz hat den Vorteil, dass das Bein nur mit einer Schraube befestigt werden muss und trotzdem gegen Verrutschen und Momente gesichert ist. Das Ende des Beins muss bei diesem Aufsatz nur ein simpler Quader sein.

In Abbildung 34 kann man sehr gut erkennen, dass auch der Aufsatz der Servomotoren geändert wurde, die in dem Gehäuse sitzen. Der neue Aufsatz erinnert, die Form betreffend, an ein Plus-Zeichen. Er wurde gewählt, da er, wenn er ins Material hineingesetzt wird, auch Querkräfte übertragen kann. Das bedeutet, dass bei einer Drehung, die Form mithilft die Beine zu bewegen und nicht die gesamte Belastung auf den Schrauben lastet.

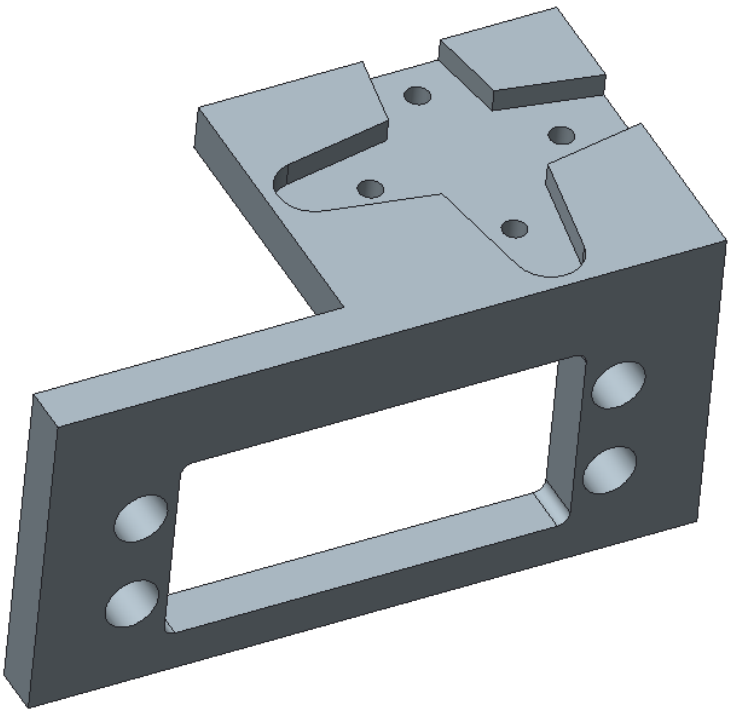

Abbildung 34: V3 Servohalterung

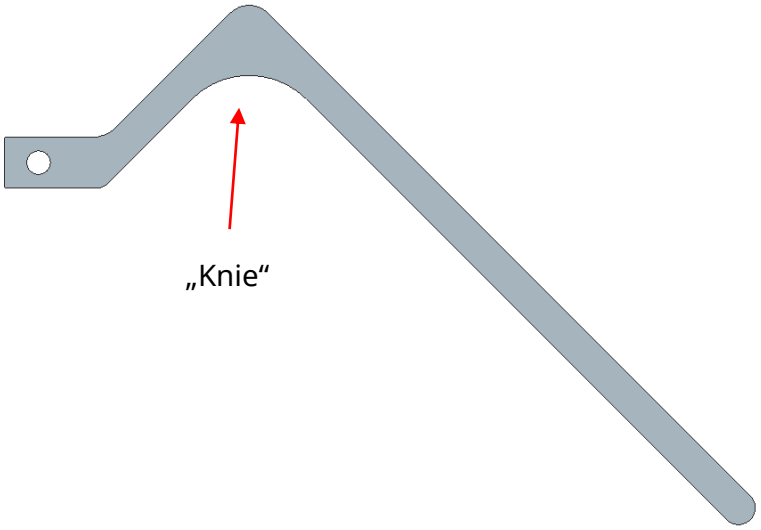

Abbildung 35: V3 Bein

Aufgrund zahlreicher Versuche, wurde evident, dass eine Formänderung, die Lebenszeit der Beine erheblich verlängern würde. Da im „Knie“ aufgrund des Knickes, Spannungsspitzen auftreten, ist es nur eine Frage der Zeit, bis sie brechen. Um dem entgegenzuwirken, wurde eine Rundung eingebaut. Diese verteilt die Kräfte besser und vermeidet zu hohe Spannungen an einem Punkt.

4.4 3D-Drucker

Für den 3D-Druck wurde ein 3D-Drucker der HTL Rennweg benutzt. Bei diesem Gerät handelt es sich um den „EVOlizer (Evo-Tech, 2017)“ von EVO-tech. Dieser Drucker wurde verwendet, da Firmen für das Benutzen ihrer Geräte viel Geld verlangen, wohingegen das Drucken in der Schule gratis ist.

Der Drucker arbeitet mit dem sogenannten „G-Code“. Dieser Code ist eine Anreihung an Befehlen wie G89 oder G54. Der 3D-Drucker versteht diese Befehle und führt demnach Operationen, wie zum Beispiel die Bewegung um eine Achse, aus. Um diesen G-Code zu erstellen, gibt es mehrere Möglichkeiten. Die schwierigste ist es, jeden Befehl per Hand zu schreiben. Eine einfachere Methode ist es, diesen G-Code anhand einer Vorlage, von einem Programm erstellen zu lassen. Um dies umsetzen zu können, muss zuerst ein 3D-Model erstellt, und in eine Stereolithographie (stl) umgewandelt werden. Diese Datei gibt mithilfe eines Dreiecksnetzes die Oberfläche eines Objekts an. Somit kann also die Form, aber nicht das Volumen erkannt werden. Diese stl-Datei kann nun von einem Programm gelesen werden. Anschließend können die nötigen Arbeitsschritte (zu fahrenden

Bahnen, benötigtes Stützmaterial) berechnet werden. Diese Information kann schlussendlich in einen G-Code gewandelt werden.

Bei diesem Projekt wurde der Slic3r (Slic3r, 2017) verwendet. Slic3r kann entweder einfach so verwendet, oder in das Programm Repetier-Host (Repetier, 2017) integriert werden. Der Unterschied ist, dass Slic3r allein Detailgenauer ist, Repetier-Host allerdings auch die benötigte Zeit errechnen kann.

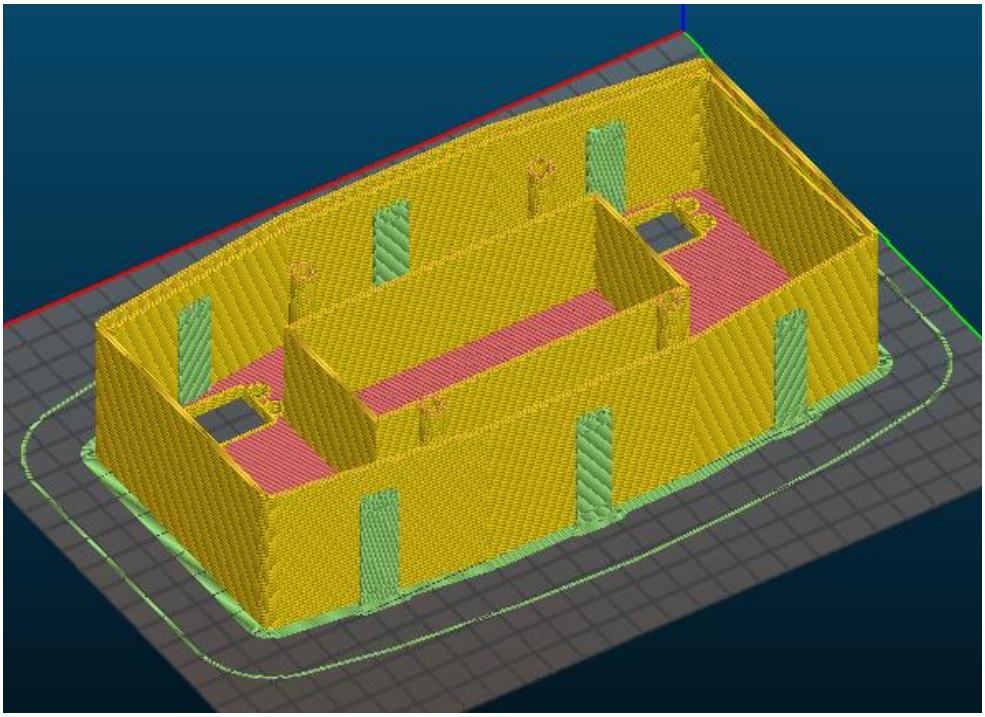

Abbildung 36: 3D-Druck Gehäuse

In Abbildung 36 können die verschiedenen Materialien erkannt werden. In rot und gelb zu sehen ist das Hauptmaterial, während das Stützmaterial in grün zu erkennen ist.

Um zu prüfen, ob der 3D-Druck wie gewünscht verläuft, wurde ein Bein als Probestück hergestellt. Anhand dieses Tests wurde festgestellt, dass der 3D-Drucker die Maße nicht perfekt, sondern zu groß druckt. Das wäre grundsätzlich kein Problem, allerdings ist es nicht mehr möglich, die Schrauben zu platzieren. Da die Maße zu groß werden, werden die Löcher zu klein. Um dieses Problem zu beheben, wurde diese Übergröße einkalkuliert.

Für das Gehäuse war als Farbe schwarz vorgesehen. Grün würde optisch nicht zur Platine und rot oder orange nicht zu den Beinen passen. Da allerdings kein schwarzes Material vorhanden war, wurde stattdessen weißes verwendet. Die Beine und die Halterungen wurden in Rot gedruckt. Somit hat der Roboter, von oben betrachtet, die Farben rot-weiß-rot.

5 Elektronik

Aufgabe der Elektronik ist es, die Zusammenarbeit aller erforderlichen Teile zu ermöglichen und diese zu versorgen. Hierzu muss festgelegt werden, welche Teile verwendet werden. Anschließend können die wichtigsten Peripherieteile, wie Stromversorgung oder Kommunikationswege, geplant werden. Als letzter Schritt wurden nützliche Zusätze und Erweiterungen eingeplant. Sobald alle erforderlichen/erwünschten Features vorhanden sind kann mit der Erstellung des Schaltplanes begonnen werden. Hierfür wurde das Programm „Altium Designer“ verwendet, da die erforderlichen Teile/Connectoren bereits als Library zur Verfügung standen. Weiters ist es dasselbe Programm, das unser Sponsor „Theobroma Systems“, was den Datenaustausch und die Zusammenarbeit deutlich vereinfachte. Nach dem Fertigstellen des Schaltplanes und einer Kontrolle seitens unseres Sponsors musste der Schaltplan erstellt werden. Hierfür wurde ebenfalls Altium Designer verwendet.

5.1 Auswahl des Hauptprozessors

Die Auswahl des Hauptprozessors hing von mehreren Faktoren wie der Rechenleistung und der Größe ab.

5.1.1 Anforderungen

Einer der größten Faktoren in der Auswahl des Prozessors war aufgrund des neuronalen Netzwerks die verfügbare Rechenleistung, sowie die Möglichkeit Python Code zu verarbeiten. Weiters muss sich der Prozessor ohne spezielle Bauteile von einem Akku mit Strom versorgen lassen, sowie alle erforderlichen Schnittstellen für die Kommunikation mit der Fernbedienung und den Servos zur Verfügung stellen. Außerdem benötigt er die Möglichkeit, mindestens ein USB 2.0-Gerät anzusteuern. Ein weiterer wichtiger Kritikpunkt war die Größe des Prozessors, da er sich zu jeder Zeit auf dem Roboter befinden sollte.

5.1.2 Der Qseven-Standard

Der Qseven-Standard ist ein offener Standard der SgET (Standardization Group for Embedded Technologies). Er beschreibt die Nutzung eines 230 Pin MXM2 SMT Connectors mit herstellerübergreifenden hochintegrierten Einplatinenrechnern bis 12W.

Abbildung 37 MXM2 - Connector

Unser Maturaprojekt basiert auf Version 2.0 des Standards. Der Standard sieht Module mit entweder 70x70mm oder 70x40mm (µQseven) vor. Seitens der Konnektivität gibt es Pins für USB 2.0, GBit Ethernet, HDMI, I2C und GPIO-Pins, die auch für eine RS232-Schnittstelle genutzt werden können. Weitere Schnittstellen wie CAN-Bus, PCIEx1 oder SPI sind ebenfalls vorhanden, werden jedoch in diesem Projekt nicht benötigt. Die Versorgung des Moduls wird über eine einzelne 5V Zuleitung sichergestellt, alle anderen benötigten Spannungen werden direkt auf dem Modul erzeugt.

5.1.3 Gewähltes Modul

Die Wahl des Moduls fiel auf das A64-µQ7 von Theobroma Systems, da Teammitglied Nicolas Oswald aus einem Ferialpraktikum bereits Erfahrung mit dem Vorgänger A31-µQ7 hat. Es bietet einen Quad-Core ARM Cortex-A53 Prozessor von Allwinner Technologies, den namensgebenden A64, mit einer Maximalfrequenz von 2,0 GHz, zusammen mit 2GB RAM und 8GB eMMC Flash Speicher. Weiters bietet es bis zu 7 USB 2.0 Anschlüsse, Gigabit Ethernet, HDMI, I2C sowie zahlreiche GPIOs mit RS232. Zum

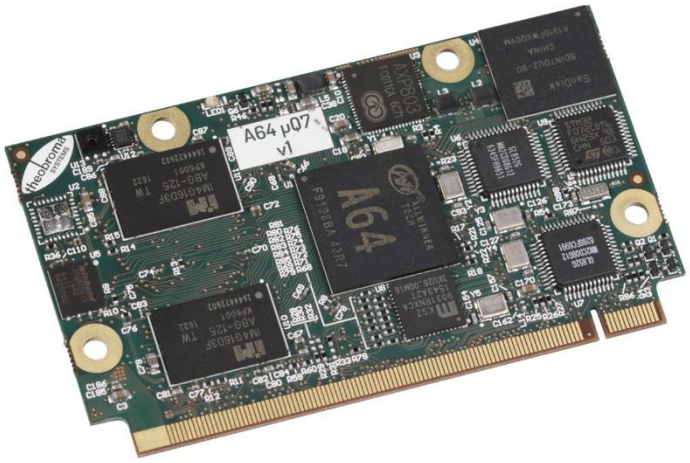

Abbildung 38 A64-µQ7

Abbildung 39 A31-µQ7

momentanen Zeitpunkt ist dieses Modul jedoch noch nicht verfügbar, weswegen die bisherige Entwicklung mit dem A31-µQ7 geschah. Dieser ist ebenfalls eine Quad-Core CPU von Allwinner, basiert jedoch auf der alten 32-Bit ARM Cortex-A7 Architektur. Dadurch, und durch die damit einhergehende Reduktion des Maximaltaktes auf 1,2 GHz, sinkt die Gesamtleistung des Moduls deutlich, was sich auch in der Perfomance des Python-Codes bemerkbar macht. Abgesehen davon ist die Ausstattung sehr ähnlich, mit ebenfalls 2GB RAM und 8GB onboard eMMC. Der einzige größere Unterschied bei der I/O ist das Vorhandensein von 8 USB 2.0 Schnittstellen, da allerdings nur vier verwendet werden hat das keinen Einfluss auf das Projekt.

5.2 Festlegung der erforderlichen Features der Platine

Bevor die Platine designt wurde, mussten alle erforderlichen Features festgelegt werden.

5.2.1 USB

Für die Fertigstellung des Projekts wird mindestens ein USB 2.0 Gerät. Da genügend Platz vorhanden ist wurden vier Stecker eingeplant. Diese dienen zur Verbindung mit dem WLAN-Modul, das für Updates und die Fernsteuerung verwendet wird, sowie weitere Geräte wie Webcams (für eine Live-Übertragung der aktuellen Position) oder Maus und Tastatur zur Nutzung der verfügbaren grafischen Benutzeroberfläche von Linux (Xfce 4).

5.2.2 5V Spannungsversorgung

Der 5V Spannungsregler versorgt den Hauptprozessor, sowie die USB-Anschlüsse der Platine. Damit auch stärkere USB-Geräte versorgt werden können, ohne das der Prozessor abstürzt wurde der Step-Down Converter TPS62133 von Texas Instruments gewählt. Dieser operiert mit eine Schaltfrequenz von 2,5MHz und kann bis zu 3A Dauerstrom liefern. Weiters kann er mit dem „WeBench“ Tool berechnet werden, was die Dimensionierung enorm vereinfacht. Dies hat eine hohe Effizienz von über 90% zur Folge. Der Regler kann mit einer Eingangsspannung zwischen der Ausgangsspannung (in diesem Fall 5V) und 17V betrieben werden. Dies ermöglicht theoretisch die Verwendung eines stärkeren Akkus um die Laufzeit zu

Abbildung 40 TI TPS62133

erhöhen. Weiters biete er einen Überstromschutz und eine automatische Abschaltung bei Überhitzung.

5.2.3 6V Spannungsversorgung

Für die Erzeugung der 6V wurde der Step-Down Converter TPS54623 gewählt, da dieser bis zu 6A Dauerstrom bietet. Dies ist für den Betrieb der Servos nötig, da sie ansonsten an Kraft verlieren. Der TPS54623 wird ebenfalls von Texas Instruments produziert und ist auch über das „WeBench“ Tool berechenbar. Die Effizienz liegt bei Betrieb der Servos über 90%. Ein weiterer großer Vorteil ist die schnelle Nachregelung bei einem Spannungseinbruch, da so die gleichbleibende Leistung der Servos garantiert werden kann. Die 6V Versorgung wurde bewusst komplett von der 5V Versorgung getrennt, da so Spannungsabfälle (wie bei großer Belastung der Servos) keine Probleme mit dem Hauptprozessor verursachen können. Die Eingangsspannung kann zwischen der Betriebsspannung (in diesem Fall 6V) und 17V liegen. Als weiteres Sicherheitsfeature bietet er einen automatischen Überstromschutz.

Abbildung 41 TI TPS54623

5.2.4 3,3V Spannungsversorgung

Abbildung 42 ST LD1117 3,3V

Die 3,3V Spannungsversorgung wird über einen LD1117 von STMicroelectronics realisiert. Dieser liefert fixe 3,3V bei bis zu 800mA Dauerstrom und hat eine geringe Dropout-Spannung von 1V. Er wird von der 5V-Versorgung gespeist, da auf diese Art auch höhere Batteriespannungen verwendet werden können. Da nur die PIC-µController und der HDMI-Connector für Powerdelivery (500mA) versorgt werden müssen ist der vergleichsweise geringe Ausgangsstrom ausreichend.

5.2.5 µC für PWM

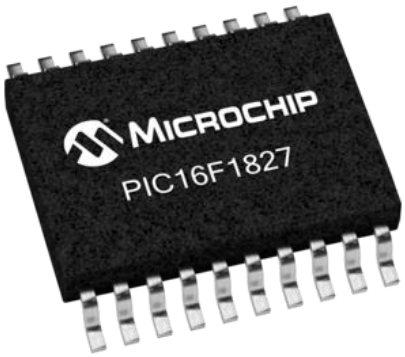

Abbildung 43 Microchip PIC16F1827

Da jeder einzelne Servo-Motor eine PWM zur Ansteuerung benötigt wird die Erzeugung dieser an Mikroprozessoren abgegeben. Jeder der drei verwendeten PIC16F1827 Prozessoren kann bis zu vier voneinander unabhängige PWM-Signale erzeugen.

5.2.6 Kommunikation via RS232

Da RS232 ursprünglich nur für zwei Geräte vorgesehen ist musste ein Trick verwendet werden, um alle µController mit derselben serielle Schnittstelle zu verbinden. Dieser sieht eine Diode in Sperrrichtung an jeder der Tx-Leitungen der PICs vor, zusammen mit einem Pull-up-Widerstand auf 3,3V zwischen den PICs und dem Hautprozessor. Das verhindert Kurzschlüsse zwischen den PICs und ermöglicht dadurch die Kommunikation zwischen mehreren Geräten. Zusammen mit dem Anschluss für einen externen RS232-Adapter können so bis zu fünf Geräte gleichzeitig kommunizieren.

5.3 Weitere Features

Diese Features sind zwar nicht unbedingt für die Funktion der Platine erforderlich, bieten aber ohne großen Zusatzaufwand praktische Funktionen.

5.3.1 HDMI

Da Qseven-Module standardmäßig mit einer HDMI-Verbindung ausgestattet sind müssen nur die richtigen Pins des Connectors mit einer HDMI-Buchse verbunden werden. Nur die dedizierten 5V-Leitungen müssen über Level-Shifter verbunden werden.

5.3.2 Ethernet

Da Ethernet mit ±2V betrieben wird ist keinerlei aktive Elektronik nötig. Für Gigabit-Übertragung ist zwar eine genau definierte Induktion zwischen den Leitungen nötig, was eine mehrschichtige Platine nötig macht, ist diese jedoch nicht vorhanden fällt das System automatisch auf 100Mbit zurück.

5.4 Vorabtests

Die Vorabtests sollen sicherstellen, dass die auf der letzten Version der Platine geplanten Features auch funktionieren.

5.4.1 Testplatine für RS232 und PWM

Um sicherzustellen, dass die Kommunikation zwischen der Hauptplatine und den Mikroprozessoren funktioniert wurde vorab eine Testplatine hergestellt. Diese besteht lediglich aus einem Mikroprozessor und einigen Stiftleisten zur Verbindung mit den Motoren, der Spannungsversorgung, und dem PICkit-Programmiergerät. Hier konnte die Kommunikation zwischen den Prozessoren, sowie die Erzeugung des PWM-Steuersignals für die Servomotoren getestet werden.

5.5 Entwickeln des Schaltplanes

In die Entwicklung des Schaltplanes floss viel Zeit, da die Herstellung der Platine einer der größten Kostenpunkte des Projektes ist.

5.5.1 Dimensionieren der Spannungsregler

Die Spannungsregler gehören zu den wichtigsten Komponenten auf der Platine in unserem Projekt, da sie sehr hohen Anforderungen entsprechen müssen. So muss zum Beispiel der Spannungsregler für die 6V-Versorgung der Servomotoren bis zu 6A Strom liefern, damit es bei den Motoren nicht zu Leistungsverlust kommt. Der 5V-Regler hingegen muss auch bei sehr geringen Spannungsunterschieden und gelegentlichen

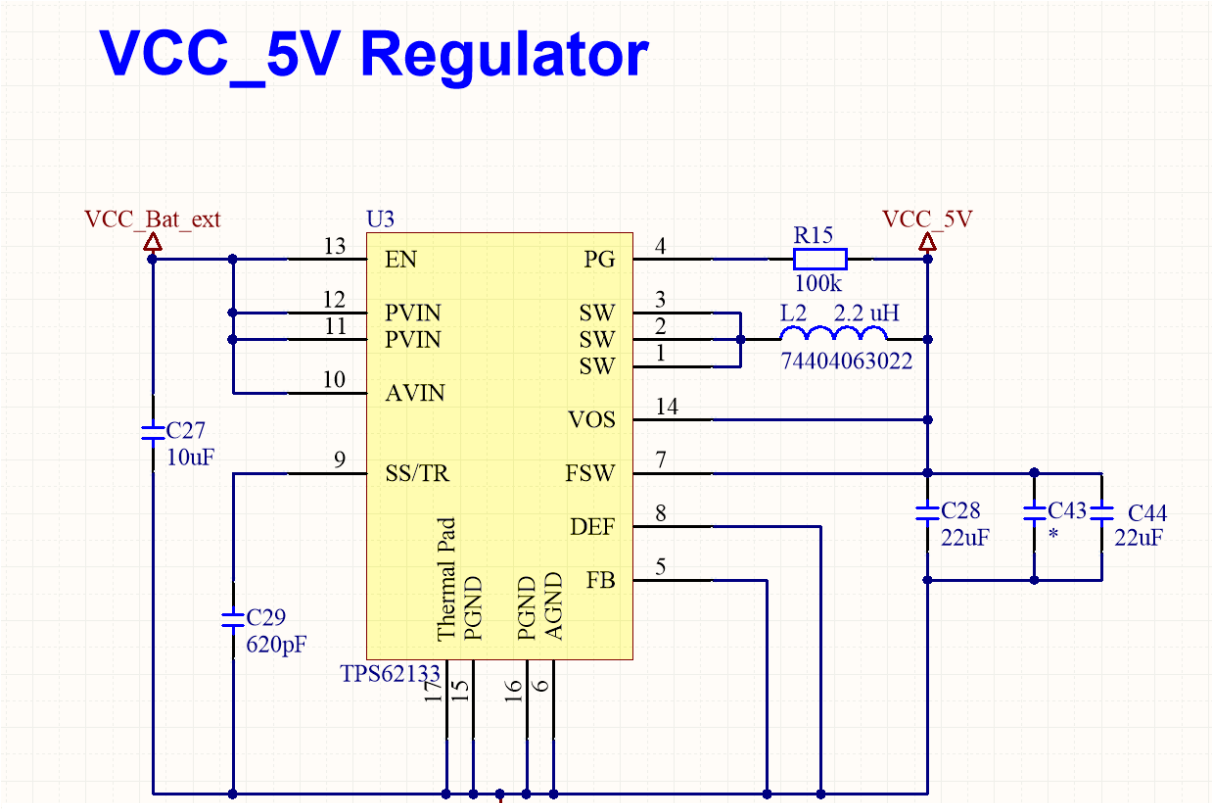

Abbildung 44 Schaltplan 5V Spannungsregler

Spannungsabfällen stabile 5V für die Versorgung des Hauptprozessors gewährleisten. Um diesen Anforderungen zu entsprechen wurden Hochleistungs-Step-Down Converter von Texas Instruments verwendet. Zu Beginn wurden Spannungsregler mit fixen Spannungen in Betracht gezogen, diese erwiesen sich jedoch schnell als ungeeignet, da sie nur vergleichsweise geringe Ströme liefern können und meistens 1-2V Spannungsunterschied benötigen, was bei einer Batteriespannung zwischen 6,8V und 8,4V nicht immer gewährleistet werden kann. Für die Dimensionierung der Spannungsregler wurde das „WeBench“-Tool von TI verwendet. In dieses werden die benötigte Spannung und Strom, sowie die Versorgungsspannung eingetragen, woraufhin die verfügbaren Regler durchsucht und auf Wunsch fertig dimensioniert werden. Die einzigen Anpassungen müssen bei den Werten einiger Widerstände vorgenommen werden, da diese nicht immer vorhanden sind.

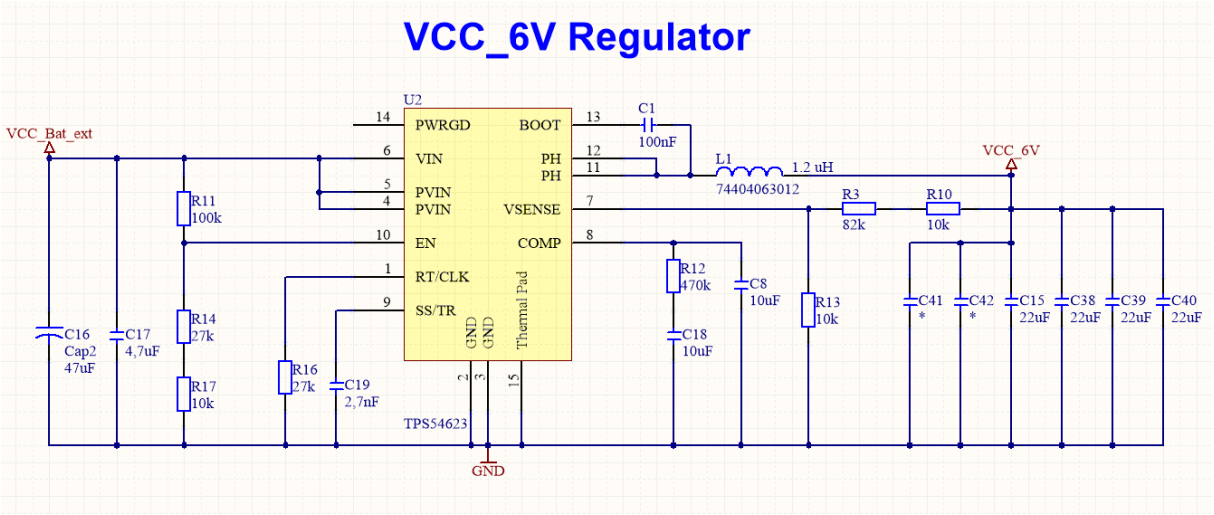

Abbildung 45 Schaltplan 6V Spannungsregler

HDMI

Die Schaltung für HDMI konnte von den Referenzdesigns von Theobroma Systems übernommen werden. Der optionale ESD-Schutz wurde nicht eingeplant, da die dafür nötigen Teile vergleichsweise teuer und schlecht lieferbar sind. Ein Nebeneffekt dieser Maßnahme ist, dass „billige“ HDMI-Kabel ohne extra differenziellen Ground-Leitungen verwendet werden können.

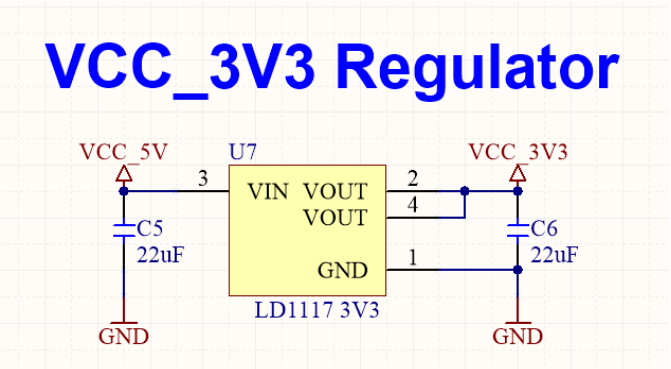

Abbildung 46 Schaltplan 3,3V Spannungsregler

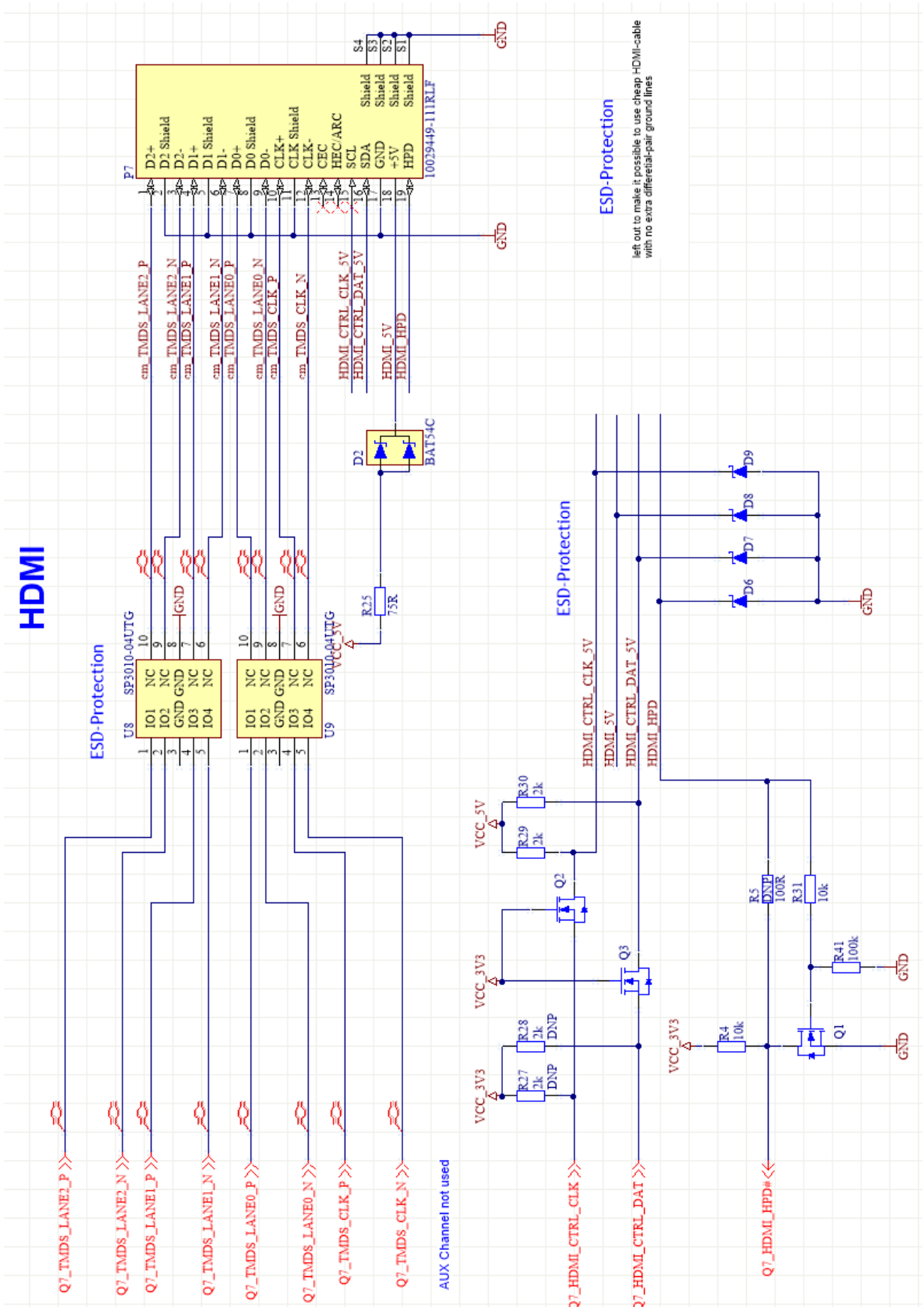

Abbildung 47 Schaltplan HDMI Anschluss

5.5.2 USB

Von den verfügbaren sieben USB-Schnittstellen (eine davon OTG-Fähig) wurden vier in den Schaltplan aufgenommen. Mehr Stecker hätten, neben deutlich mehr Platz, auch eine stärkere 5V Spannungsversorgung erfordert. Da jedoch nur selten mehr als zwei benötigt werden wurde entschieden nur vier zu verbauen. Der ESD-Schutz wurde nach den Richtlinien im von Cypress publizierten Dokument „Common USB Development Mistakes“ sichergestellt.

5.5.3 Ethernet

Als Ethernet-Stecker wurde der Würth Electronics 7499111221A gewählt, da dieser die für Ethernet benötigten Übertragern/Stromkompensierdrosseln bereits enthält. Bei einfacheren Steckern müssten diese als zusätzliches Teil eingeplant werden, was die genaue Dimensionierung komplizierter macht.

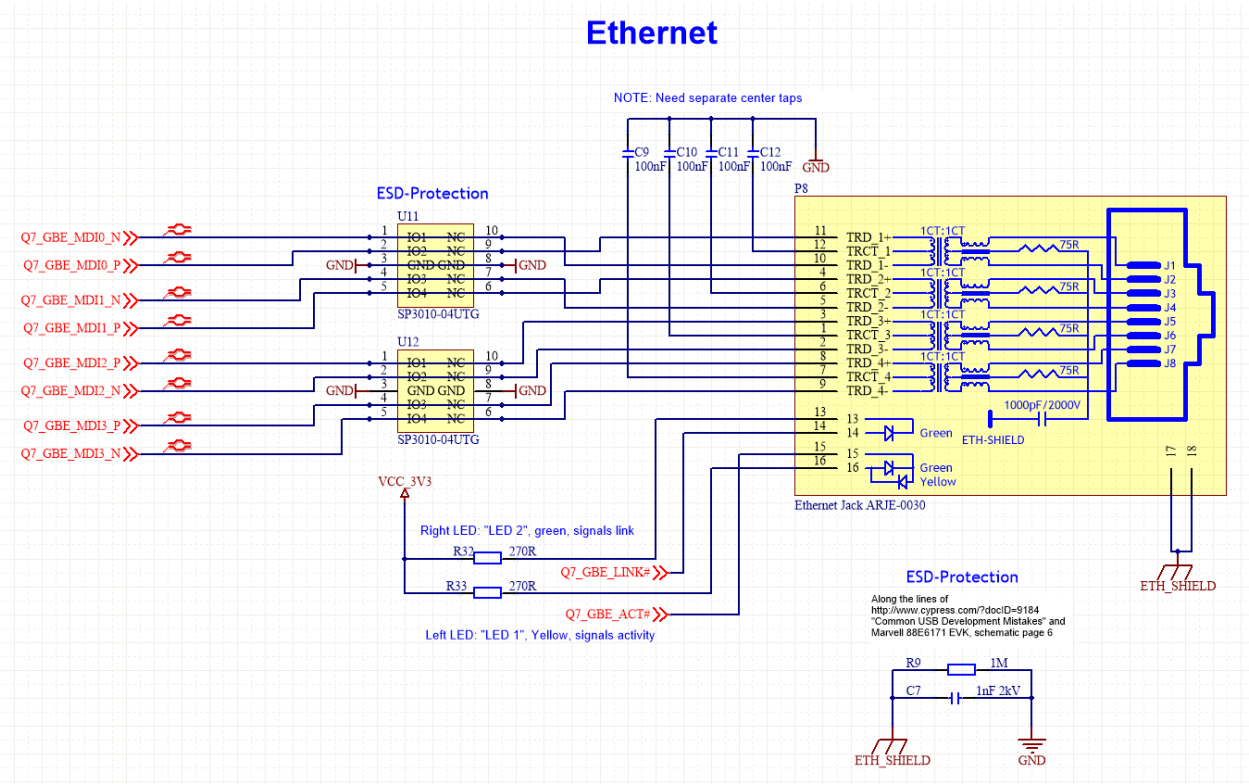

Abbildung 48 Schaltplan Ethernet Anschluss

5.5.4 MicroSD

Der Anschluss einer SD-Karte an das Hauptmodul ist durch das Vorhandensein dedizierter Pins auf dem Qseven-Connector sehr einfach, da lediglich jeder Pin der SD-Karte mit dem jeweiligen Pin am Connector verbunden werden. Zusätzlich wurde für alle

Leitungen ESD-Schutz mittels High-Bandwidth ESD Dioden sichergestellt. Hierfür werden die HSP061-4M10 von ST Microelectronics verwendet, welche eine Bandbreite von 8,7GHz aufweisen.

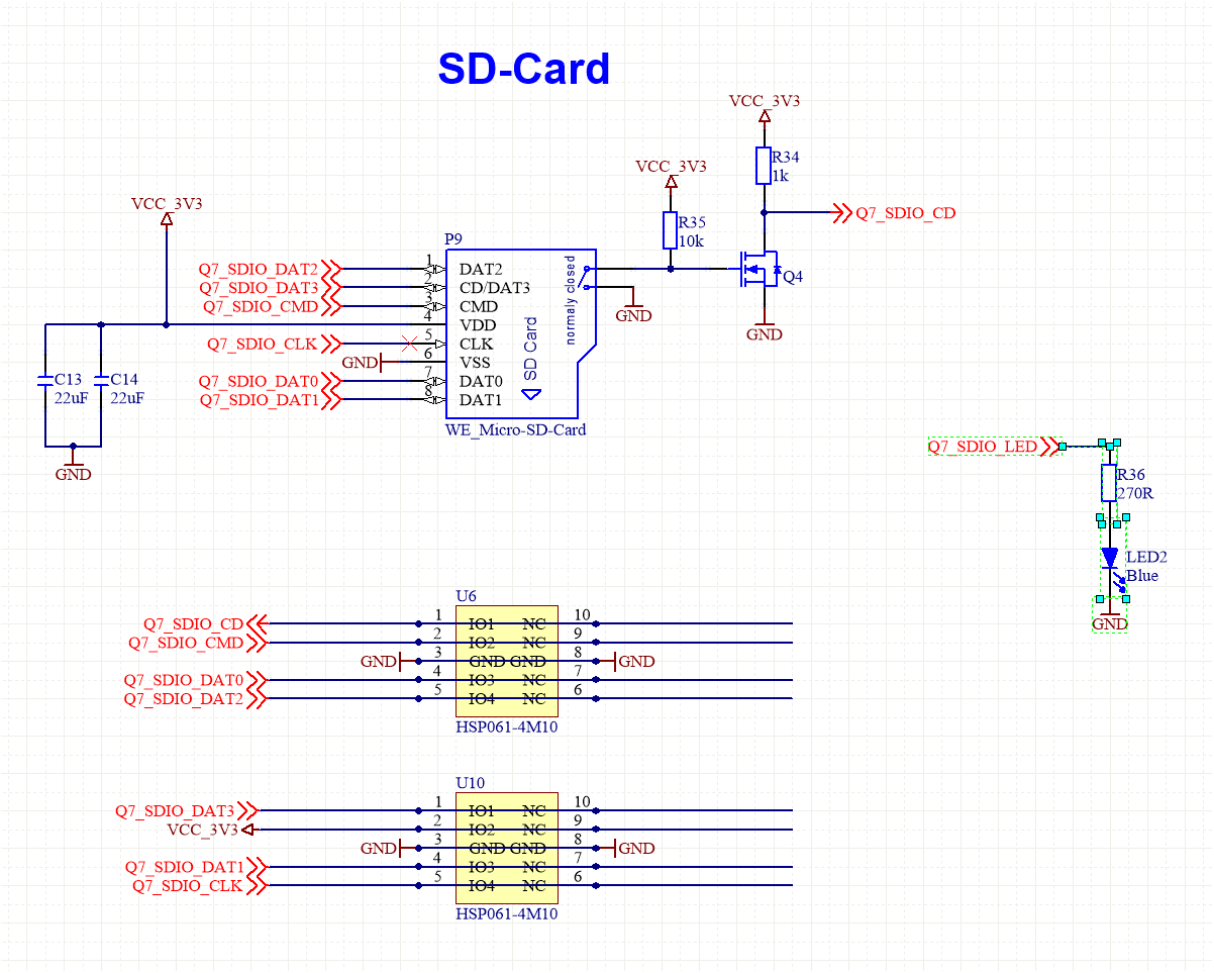

Abbildung 49 Schaltplan SD-Karte

5.5.5 PIC-µC

Da für jeden Servomotor ein eigenes PWM-Signal erforderlich ist müssen bei diesem Projekt zwölf unabhängige PWM-Ausgänge vorhanden sein. Dafür wurden drei PIC16F1827 der Firma Microchip verbaut. Jeder der Mikrocontroller hat Kontrolle über zwei der Beine, sowie eine LED. Des Weiteren ist ein Analogeingang eines PICs über einen Spannungsteiler mit dem Akku verbunden, wodurch die Akkuspannung direkt am Roboter gemessen werden kann. Dies ermöglicht sicheren Betrieb ohne den LiPo-Akku zu unterladen. Als Bauform wurde SOIC-18 gewählt, da diese einen guten Kompromiss zwischen einem kleinen Formfaktor und einfacher Verarbeitung bieten. SSOP wurde bewusst vermieden, da es nicht zum Aussehen der restlichen Platine passt und das

Routing aufgrund des deutlich größeren Footprints aufwendiger werden würde. Auch QFN wurde vermieden, da keine Leitungen unter dem Mikrocontroller verlegt werden können, was mehr Lagen auf der Platine nötig machen würde.

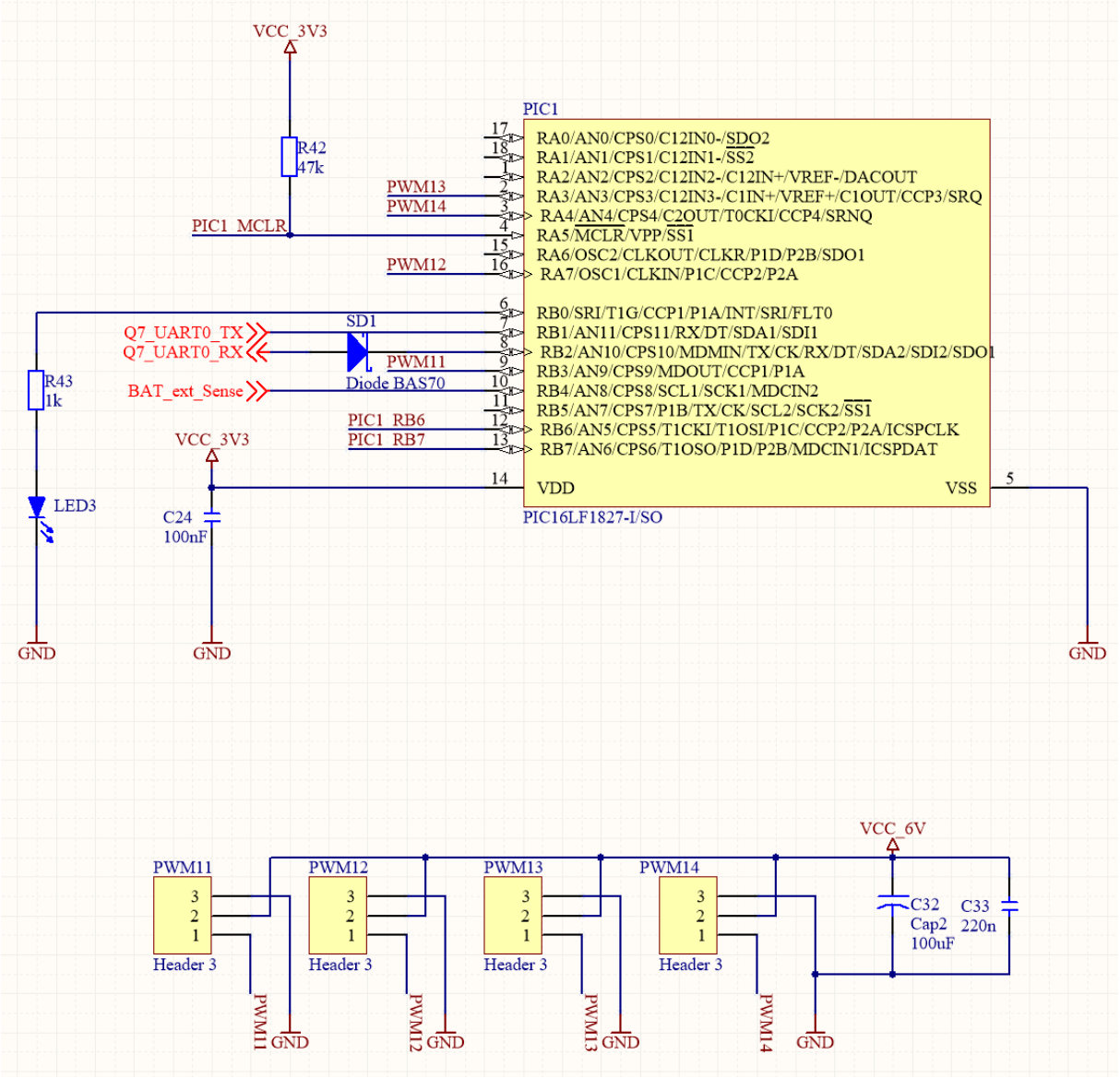

Abbildung 50 Schaltplan PIC16F1827 und Servomotoren

5.5.6 RS232

Um zwischen dem Hauptprozessor und den Motorcontrollern zu kommunizieren wurde RS232 gewählt, da es extrem einfach zu implementieren und mit einem eigenen Protokoll zu versehen ist. Um die Kommunikation zwischen mehreren Geräten mit einem Protokoll zu ermögliche das auf Point to Point Kommunikation ausgelegt ist müssen im Schaltplan zusätzliche Bauteile verbaut werden. Hierbei muss die Empfangsleitung des Hauptprozessors über Dioden mit den Sendeleitungen der PICs verbunden werden und

ein Pull-up Widerstand zwischen den Dioden und dem Hauptprozessor eingebaut werden. Hierdurch wird, da bei RS232 wenn keine Kommunikation stattfindet die Leitung auf logisch 1 gesetzt wird, verhindert, dass die logische 0 eines Signals von den anderen Mikroprozessoren „überstimmt“ wird.

Außer der Kommunikation zwischen den Prozessoren wird RS232 in diesem Projekt auch dazu genutzt, auf die Kommandozeile der Linux-Installation zuzugreifen, wenn dies nicht über anderen Wege wie SSH möglich ist. Da die zur Verfügung stehende Linux-Distribution jedoch nur einen RS232-Port vorsieht wurden im Schaltplan beide Funktionen auf ein Leiterpaar zusammengefasst. Hierzu wurde in die bereits für die PICs bestehenden Leitungen ein Stecker eingefügt. Weiters wurde das verwendete Protokoll so angepasst, dass es einfach von den für die Kommandozeile genutzten ASCII-Zeichen zu unterscheiden ist. (Nähere Informationen hierzu: Neudesign des UART Protokolls)

5.5.7 ICP

Um die Programmierung der Mikroprozessoren automatisieren zu können wurden die zur Programmierung benötigten Leitungen im Schaltplan eingeplant. Über sie kann jeder der Prozessoren einzeln oder auch alle gleichzeitig programmiert werden. Die hierzu verwendete Methode heißt LVP (Low Voltage Programming). Auf der Seite des Hauptprozessors wurden diese Leitungen an die normalerweise für SPI vorgesehenen Pins angeschlossen. Da in diesem Projekt jedoch kein SPI benötigt wird ist dies kein Problem. Die Leiterbahnen sind außerdem durch 0Ω Widerstände von den übrigen Prozessoren getrennt, sodass auch herkömmlich über das PICKit programmiert werden kann.

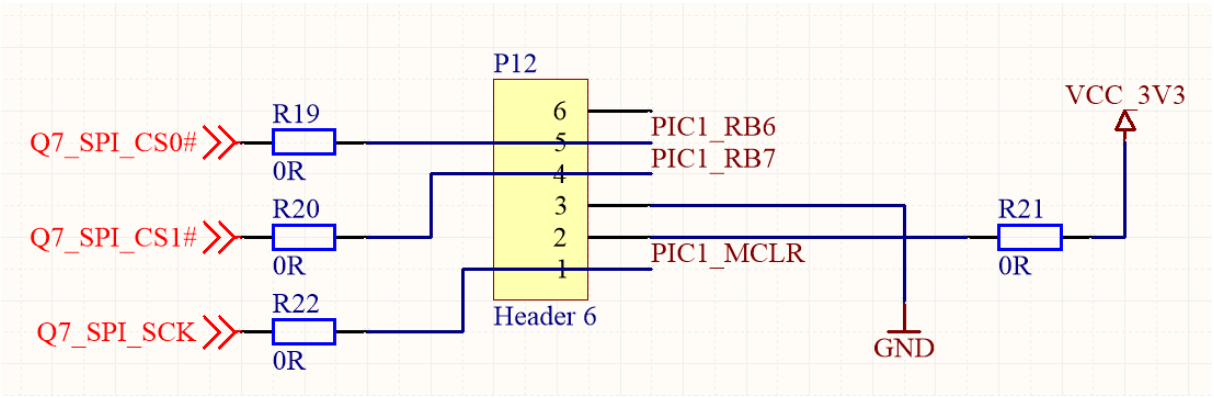

Abbildung 51 Programmieranschluss

5.5.8 I2C

Die I2C-Schnittselle wurde eingeplant um Entfernungssensoren mit dem Hauptprozessor verbinden zu können. Die hierfür benötigten Pins sind direkt auf dem A64-Modul vorgesehen. Da die Sensoren und das Hauptmodul jedoch auf verschiedenen Spannungslevels arbeiten ist zwischen ihnen ein Level-Shifter bestehend aus je einem Transistor pro Leitung (Data und Clock) eingeplant.

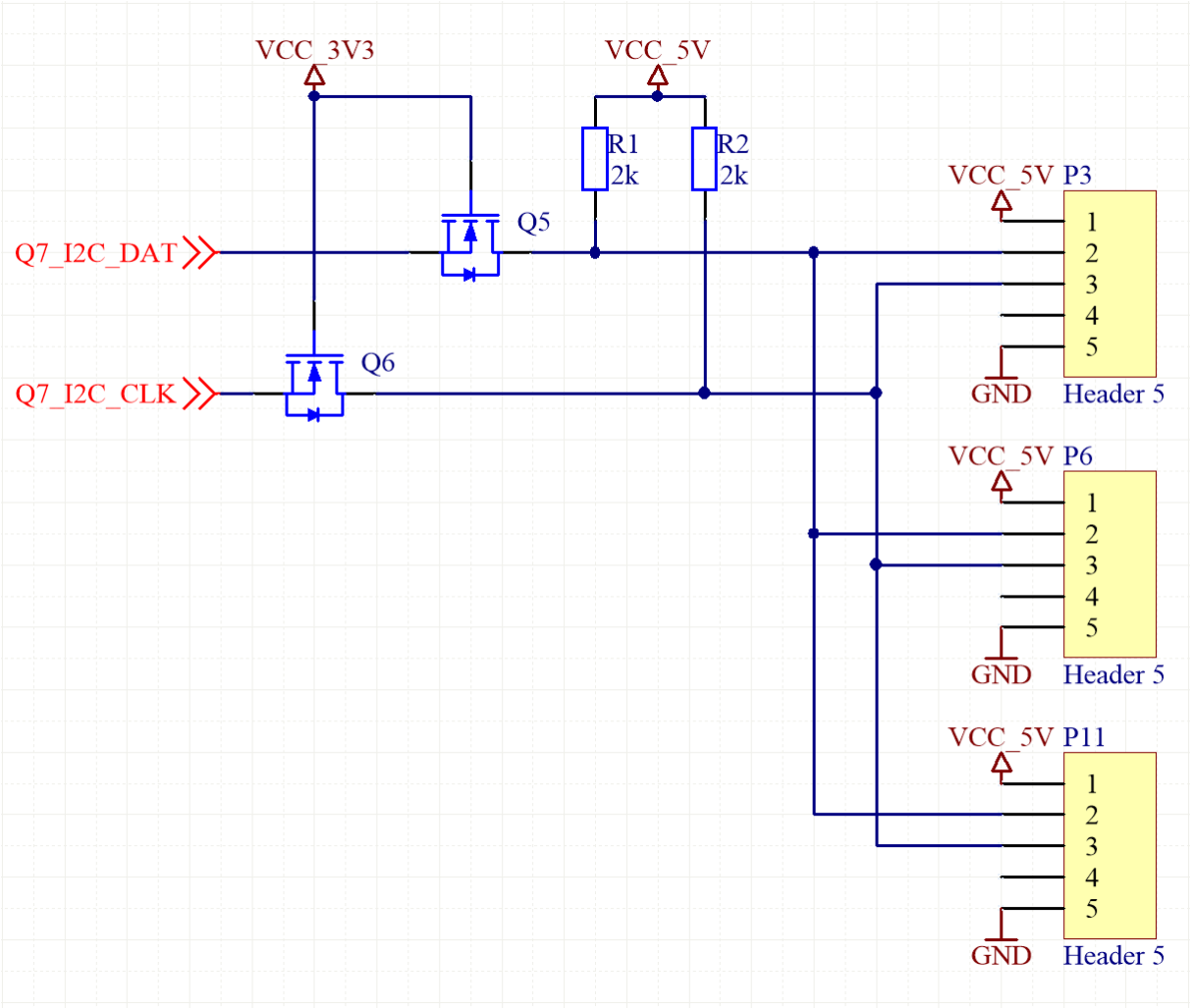

Abbildung 52 Anschluss Entfernuungssensoren

5.6 Platinendesign

Das Design der Platine stellt einen wichtigen Teil des Projektes dar, da sie sowohl einen großen ästhetischen Punkt darstellt, als auch das „Gehirn“ des Roboters beherbergt. Deswegen wurde auf eine zusätzliche Abdeckung der Platine mit einem Deckel verzichtet.

5.6.1 Power-Design

Beim Design der Spannungsregler musste vor allem darauf geachtet werden genügend Kühlfläche um das Bauteil zu haben. Hierzu wurden neben großen Kupferflächen bei den Versorgungspins unter dem Bauteil Vias gesetzt, welche die Hitze auf eine weitere Kühlfläche auf der anderen Seite der Platine leiten. Weiters musste sichergestellt werden, dass plötzliche Schwankungen im Verbrauch nicht in Spannungsschwankungen resultieren. Hierzu wurden mehrere Kondensatoren an jedem Spannungsausgang verbaut. Zusammen mit den verbauten Spulen sorgen sie für eine stabile Spannung.

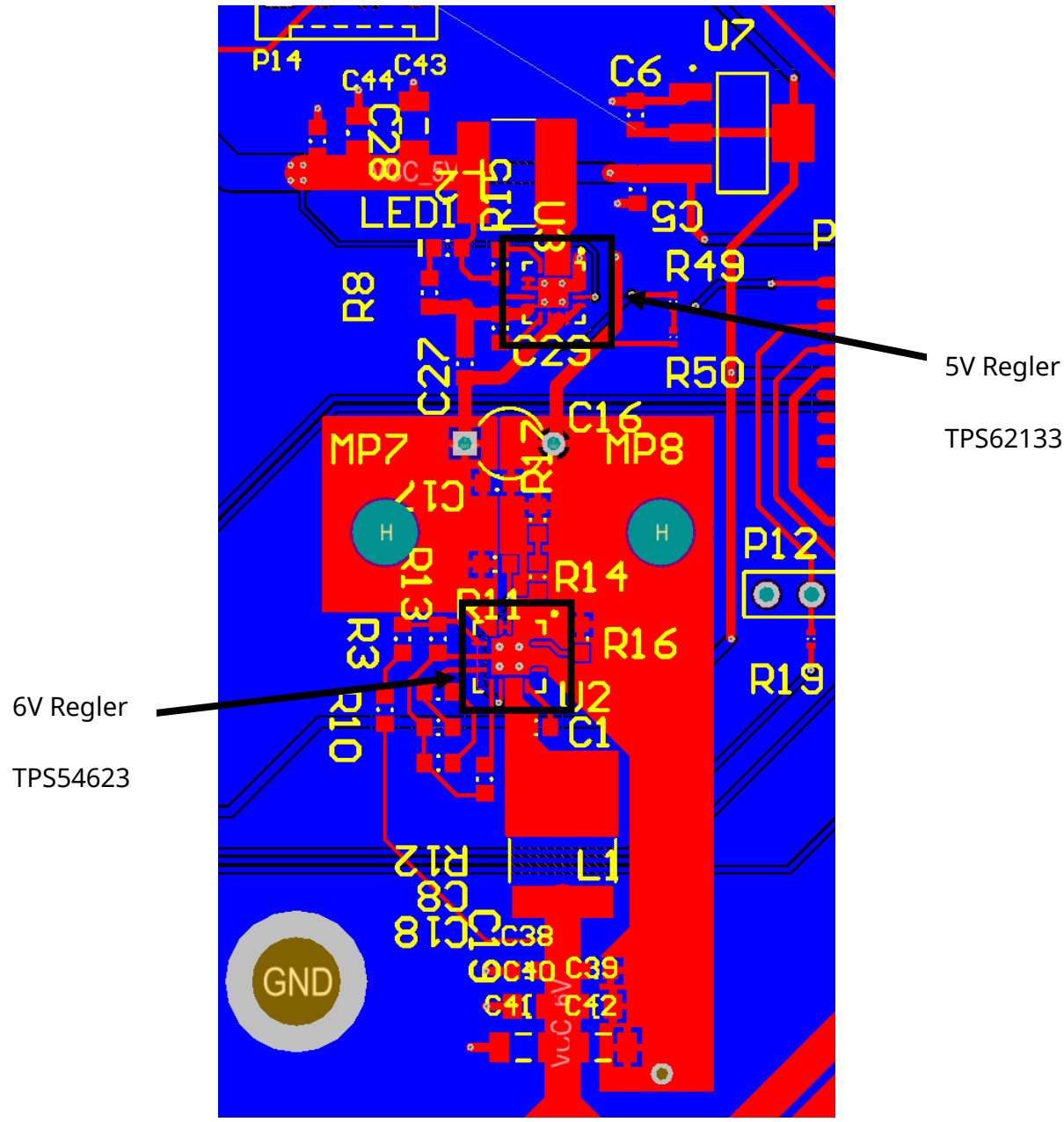

Abbildung 53 PCB Spannungsregler

5.6.2 Spezielle Anforderungen für HDMI und Ethernet

Sowohl HDMI, als auch Ethernet sind sehr sensibel gegenüber verschieden langen Leitungen zwischen den einzelnen Pins. Daher muss bei der Planung der Platine darauf geachtet werden, dass alle hierfür benötigten Leiterbahnen so gleichmäßig wie möglich verlaufen. Weiters sollte darauf geachtet werden keine der Versorgungsleitungen zu nahe an den Datenleitungen zu platzieren, da diese weitere Störungen hervorrufen können. Um allerhöchsten Anforderungen zu entsprechen müssen sowohl Ethernet-, als auch HDMI-Leitungen in genau definierten Abständen zu Kupferflächen liegen, um gleichmäßige Impedanz zu gewährleisten. Dies konnte bei der gefertigten Platine nicht eingehalten werden, da hierfür mehr als zwei Lagen nötig wären. Ohne diese exakten Impedanzen funktionieren die Schnittstellen zwar trotzdem, haben allerdings eine geringere Durchsatzrate. Bei Ethernet kann dies in einer Abstufung von Gigabit auf 100 Mbit resultieren, bei HDMI wird automatisch die maximale Auflösung und Bildrate angepasst. Beide Schnittstellen bleiben jedoch funktionsfähig.

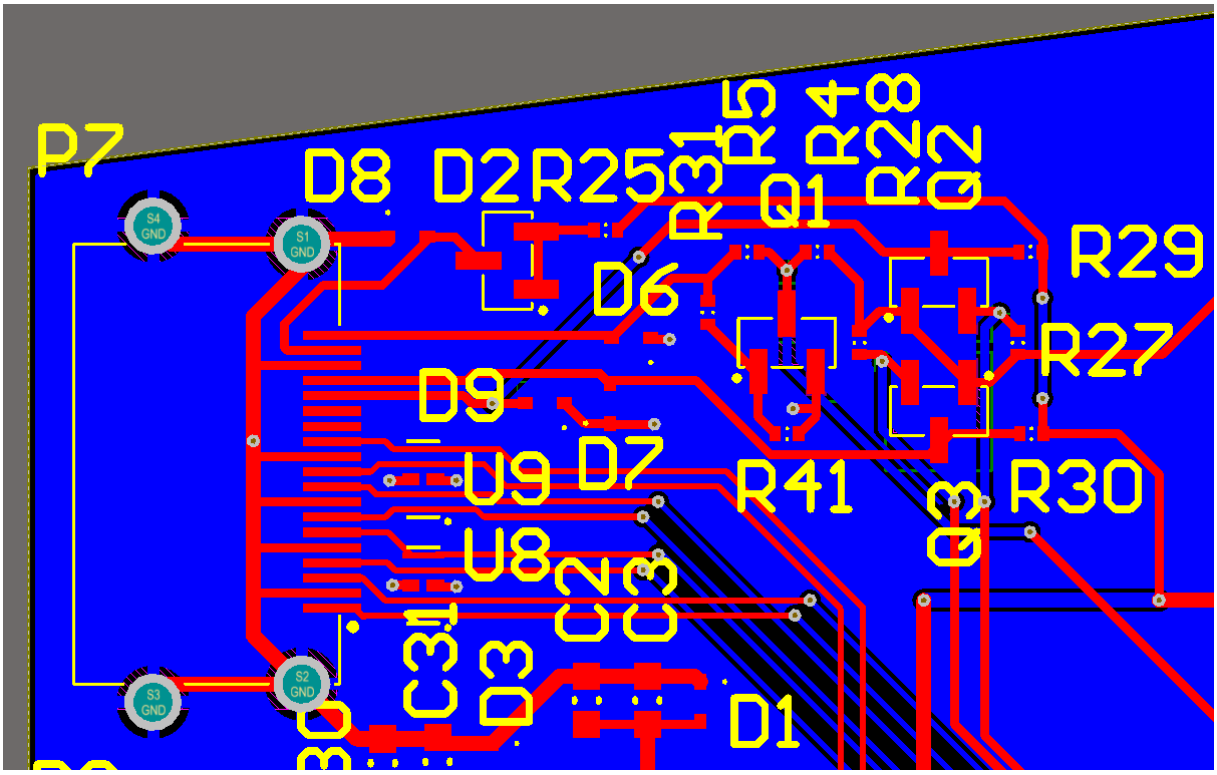

Abbildung 54 PCB HDMI Anschluss

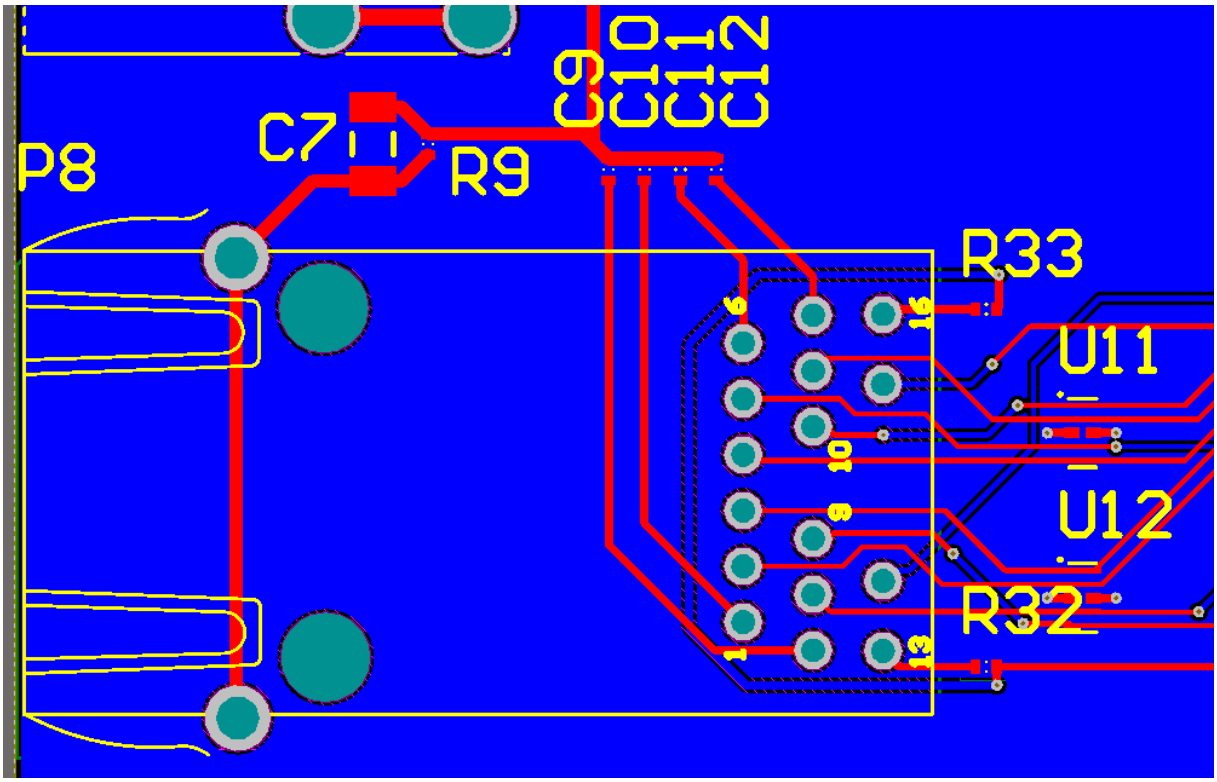

Abbildung 55 PCB Ethernet Anschluss

5.6.3 Q7-Edge Connector

Die Platzierung des Qseven-Connectors stellte eine unerwartete Herausforderung dar, da auf eine 70x70mm großen Fläche keine größeren Bauteile platziert werden können. Lediglich kleine SMD-Bauteile so wie Widerstände, Transistoren und kleine Kondensatoren lassen genügend Platz für das Modul. Die Knopfzelle für die Versorgung der Real Time Clock auf dem Modul, die ursprünglich unter diesem geplant war musste daher verschoben werden.

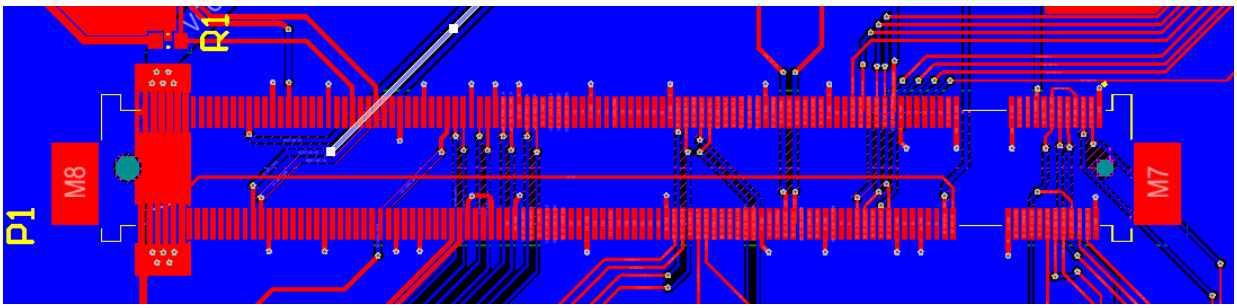

Abbildung 56 PCB MXM2 Anschluss

5.6.4 SD-Karte

Der SD-Reader wurde am Rand der Platine geplant, damit man auch im zusammengebauten Zustand des Roboters noch Zugang zur Speicherkarte hat. Dadurch

kann auch eine Beschädigung des Betriebssystems leicht von außen repariert werden. Die Aktivitäts-LED für die SD-Karte wurde direkt neben dem eigentlichen Reader platziert.

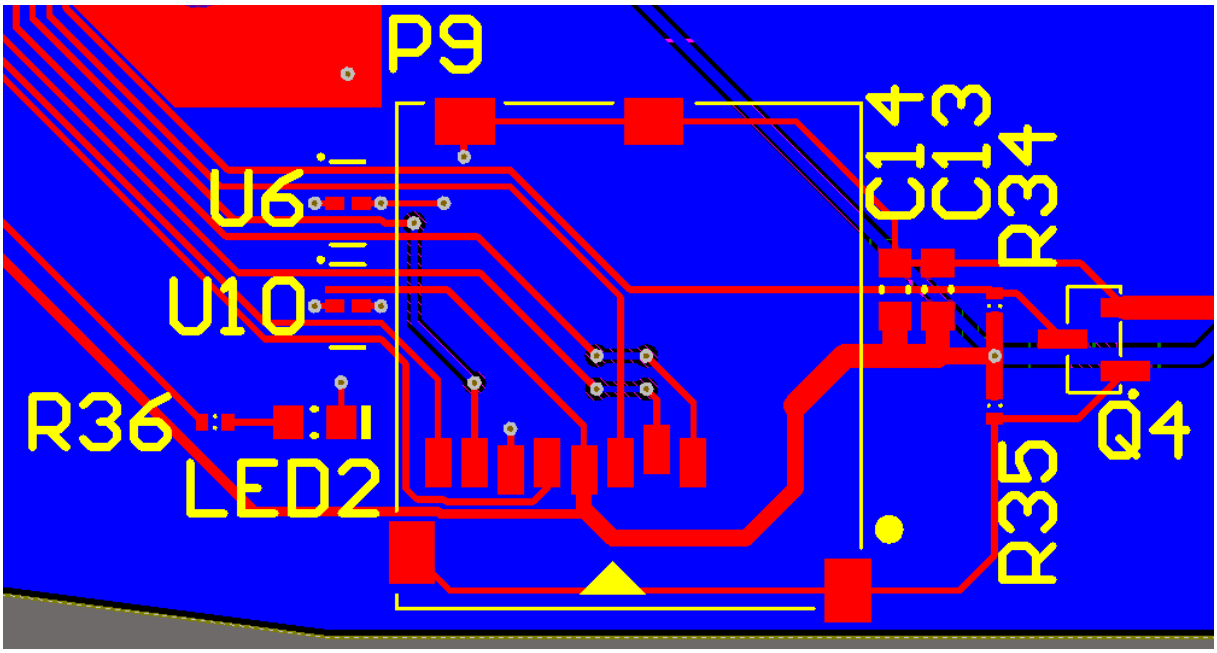

Abbildung 57 PCB SD-Karte

5.6.5 Dimensionierung der Leiterbahnenbreite

Da die Servomotoren unter Volllast viel Strom auf einmal benötigen musste bei der Entwicklung des Layouts darauf geachtet werden, dass die hierfür verwendeten Leiterbahnen nicht zu warm werden, beziehungsweise sogar schmelzen. Um die Breite zu bestimmen wurde ein Diagramm (Multi-CB Leiterplatten, 2017) verwendet. Aus diesem geht hervor, dass für 6A bei einer Kupferdicke von 35µm eine Breite von mindestens 1,75mm bei einer Erwärmung von 30K benötigt werden. Um unvorhergesehenen Stromspitzen zu genügen wurde eine Bahnbreite von 2mm gewählt.

5.6.6 Servo-Ansteuerung

Die Position der Anschlüsse für die Servos wurde so gewählt, dass einfach erkennbar ist, welcher Anschluss zu welchem Bein und welchem PIC gehört. Dies erwies sich in der Testphase als sehr praktisch. Außerdem wurde darauf geachtet, dass die Anschlüsse und alle dazugehörigen Leitungen so weit wie möglich von anderen Komponenten entfernt sind um Störungen zu vermeiden.

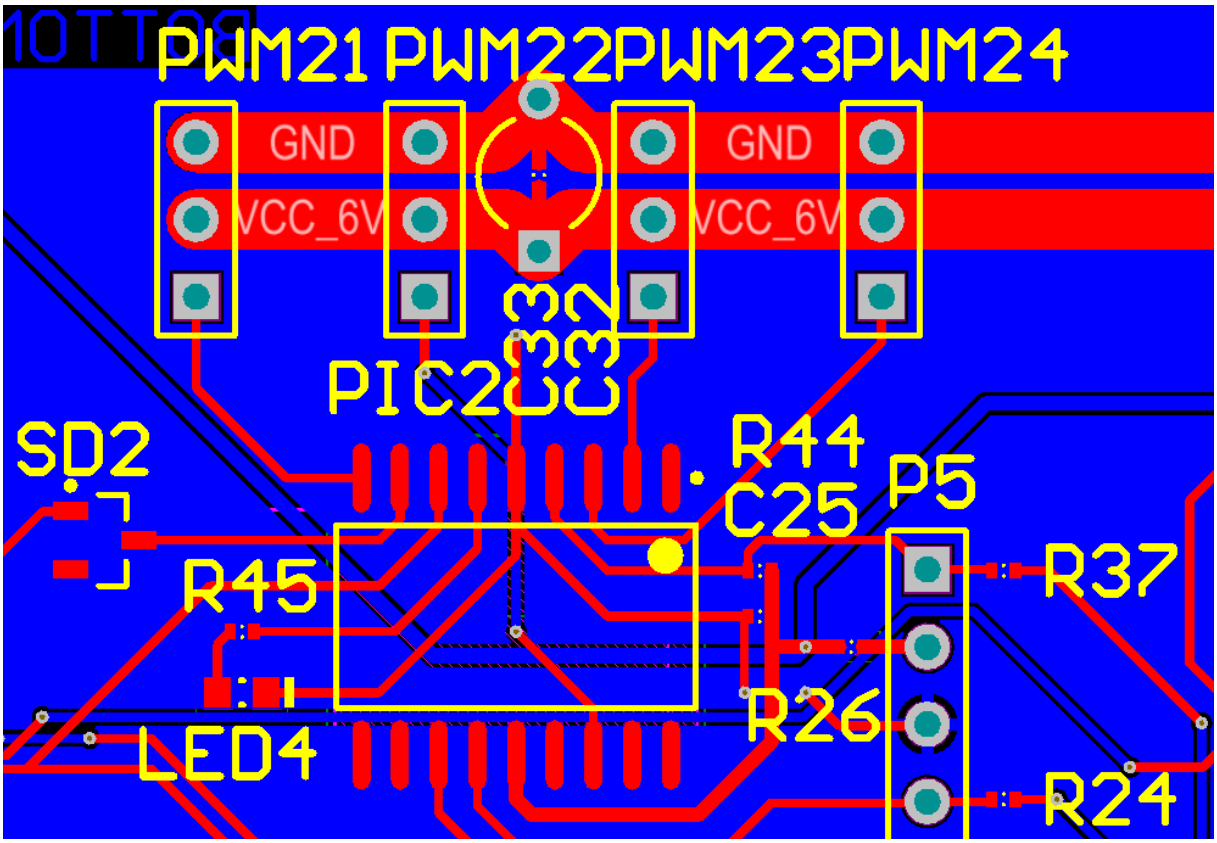

Abbildung 58 PCB PIC16F1827 und Servomotor Anschlüsse

5.7 Bestückung

Die Bestückung der Platine wurde aufgrund der teilweise sehr geringen Bauteilgrüßen nicht in der Schule, sondern direkt bei Theobroma Systems durchgeführt. Weiters konnten die Teile nicht mit einem Lötkolben bestückt werden, weswegen eine Lötmaske zusammen mit der Platine bestellt wurde. Durch diese wurde Lötpaste auf die benötigten Pads aufgetragen. Anschließend konnten die Teile platziert und die Platine in einem Lötofen gebacken werden. Nach einer Sichtkontrolle und einigen kleinen Verbesserungen beim Qseven-Connector konnte die Platine getestet werden.

5.7.1 ICP

Um zwischen verschiedenen Arten der Programmierung für die Mikrocontroller wählen zu können wurden zwar Pads für 0Ω Widerstände vorgesehen, diese allerdings noch nicht bestückt. Nun können bei Bedarf einfach drei Widerstände pro PIC eingelötet werden, um eine direkte Programmierung vom Hauptprozessor aus zu ermöglichen.

5.8 Hardware-Tests und Problemlösung

Die Hardware-Tests wurden zusammen mit den Integrationstests durchgeführt. Hierbei wurden alle für die Funktion des Roboters nötigen Teile ausprobiert und gegebenenfalls repariert.

5.8.1 6V Spannungsversorgung

Zu Beginn lieferte die 6V Versorgung lediglich 2V und reagierte nur sehr träge auf angelegte Lasten. Nach mehreren Tests bezüglich der Beschaltung und den umliegenden Komponenten wurde der Schaltplan kontrolliert. Hierbei wurde ein Fehler in der Versorgung der Steuerelektronik des Reglers bemerkt, welcher ihn davon abhielt komplett zu starten. Da die Pins jedoch direkt neben einer geeigneten Versorgung positioniert waren, war das Ausbessern dieses Fehlers sehr leicht. Es musste lediglich eine Leiterbahn weggekratzt und der Pin mit dem Nebenliegenden verbunden werden. Daraufhin funktionierte der Regler genau wie dimensioniert und liefert stabile 6,1V.

5.8.2 Programmieren der PICs

Das Programmieren der PICs über einen externen Programmer funktioniert wie erwartet einwandfrei. Das Einzige Detail auf das geachtet werden muss ist, dass die Batterie an die Platine angeschlossen sein muss, da die Versorgung nicht über das PICKit erfolgen kann. Dies wurde absichtlich im Schaltplan eingebaut um eine Querversorgung zwischen den PICs und der SD-Karte zu verhindern.

5.8.3 Zugriff auf die SD-Karte

Zu Beginn funktionierte der Zugriff auf die SD-Karte nicht richtig. Daher konnte nur der Bootloader, nicht aber das eigentliche Betriebssystem geladen werden. Mit der Hilfe von Theobroma Systems konnte der Fehler jedoch schnell auf einen zu neuen Beispiel-Schaltplan zurückgeführt werden. Durch diesen wurde der Idle-Pin der SD-Karte falsch angesteuert, was sie bei jedem Zugriff in den Stromsparmodus versetzte. Das Problem konnte entweder durch das ersetzen eines Transistors mit einer Drahtbrücke, oder durch das setzen einer Flag in der Konfiguration des Bootloaders gelöst werden. Wir entschieden uns für die Software-Lösung, da hierdurch weiterhin die Vorteile des Stromsparmodus genutzt werden können. Der daraus resultierende Nachteil ist, dass die

eingesetzte Version des Bootloaders nicht mehr auf anderen Basisboards verwendet werden kann. Dies ist jedoch während des Projektes kein Problem.

6 Neuronales Netzwerk

Ein Neuronales Netzwerk ist ein System von untereinander verbundenen „Neuronen“, die sich ähnlich verhalten wie die Neuronen im menschlichen Gehirn. Dazu werden sie in sogenannte „Layer“ aufgeteilt. Jeder dieser Layer kann eine theoretisch beliebige Anzahl von Neuronen enthalten, die jeweils mit allen Neuronen des vorherigen Layers verbunden sind. Diese Verbindungen können beliebige numerische Werte zwischen +1 und -1 annehmen. Diese Gewichtung bestimmt, wie das Ziel-Neuron von den Ausgangsneuronen beeinflusst wird. Die Gewichtungen der Verbindungen werden von einem Neuronalen Netzwerk während der Trainings-Phase automatisch von selbst angepasst, meistens in einem Verfahren namens „Reinforced Learning“. Hierzu werden dem System sowohl Trainings-, als auch Testdaten zur Verfügung gestellt. Mit den Trainingsdaten wird nun versucht, den Fehleranteil bei den darauffolgenden Tests zu minimieren. Nach jedem Test versucht das Netzwerk die Gewichtung (entsprechend vorgegebenen Parameter wie der maximalen Änderungsrate) anzupassen. Eine weitere Möglichkeit ist lernen nach einem „Trial and Error“ Prinzip, auch bekannt als „NEAT“. NEAT steht für „NeuroEvolution of Augmenting Topologies“, zu Deutsch „Neurale Entwicklung durch aufeinander aufbauenden Topologien“. Diese Variante führt ein Bewertungssystem ein, durch das jeder Durchlauf des Netzwerkes, beziehungsweise dessen Ergebnis, mit den vorherigen verglichen werden kann. Nun versucht das Netzwerk durch das anfangs zufällige verknüpfen von Ein- und Ausgängen eine höhere Punktezahl zu erreichen als die letzte Generation. Wirft ein Evolutionszweig zu lange keine vielversprechenden Ergebnisse ab wird einfach auf eine ältere Generation zurückgeschalten und es wir ein anderer Weg ausprobiert. Gibt es auch hier keine zufriedenstellenden Ergebnisse wird noch eine Generation weiter zurückgeschalten. Dies kann entweder so lange ausgeführt werden bis ein festgelegter Schwellwert unterboten wird, oder bis alle Kombinationen ausprobiert wurden. Diese Variante kann jedoch, je nach Anzahl von Ein- und Ausgängen, sehr viele Versuche und damit sehr viel Zeit in Anspruch nehmen. Für das Lernen des Roboters sollte ein solcher NEAT-

Algorithmus verwendet werden, da er keine bereits bekannten Testdaten benötigt. Als Bewertungssystem wurde die erreichte Durchschnittsgeschwindigkeit vorgesehen.

6.1 TensorFlow

TensorFlow ist ein von Google entwickeltes Framework zum Erstellen und Trainieren von Neuronalen Netzwerken. Es bietet eine ausführliche Python-API, zusammen mit leicht verständlichen Beispielen und einer guten Dokumentation. Ein weiterer Vorteil von TensorFlow ist die Möglichkeit, den Code auf einer Grafikkarte ausführen zu lassen, was aufgrund der deutlich höheren Anzahl an Kernen im Vergleich zu CPUs einen deutlichen Geschwindigkeitszuwachs bietet.

6.2 Anforderungen

Für die Verwendung von TensorFlow ist nicht weiteres als eine funktionierende Installation von Python 3.5.x erforderlich. Die Installation erfolgt durch das ausführen des Befehls `pip install tensorflow` . Falls GPU-Support gewünscht ist muss eine Nvidia-Grafikkarte vorhanden sein. Zusätzlich müssen CUDA 8.0, cuDNN 5.1 und ein aktueller Treiber von der Nvidia Homepage heruntergeladen und installiert werden. Nun kann der Support mit `pip install tensorflow-gpu` aktiviert werden.

6.2.1 Nvidia CUDA

CUDA ist eine von Nvidia entwickelte Abwandlung der C-Programmiersprache. Über sie können die Ressourcen moderner Nvidia-GPUs für parallelisierte Berechnungen genutzt werden.

6.3 Tests

Bevor das Neuronale Netzwerk auf dem Roboter zum Einsatz gebracht werden sollte wurden ausführliche Tests durchgeführt. Hierbei wurden Ablaufgeschwindigkeit, Prozessorauslastung und Speicherauslastung getestet. Die Zeit, in der das zum Testen verwendete Programm zur Handschrifterkennung variierte wie erwartet stark mit dem eingesetzten Prozessor.

Die Speicherauslastung war in allen Tests gering genug um außer Acht gelassen zu werden.

6.4 Ergebnisse

Unglücklicherweise ist die Anforderung des Netzwerkes zu hoch für den eingesetzten Prozessor, hauptsächlich weil die eingesetzte PowerVR-GPU keinen CUDA-Code ausführen kann. Hinzu kommt die relativ hohe Auslastung des Prozessors durch den benötigten Python Code. In Zukunft kann das Hauptmodul jedoch gegen ein stärkeres ausgetauscht werden, wodurch sich die Fähigkeit nachrüsten lässt. Eine weitere Möglichkeit wäre die Nutzung eines anderen Development Kits wie dem Nvidia Tegra X, welches eine vollwertige Grafikeinheit besitzt. Dieses Board verbraucht jedoch mehr Strom und kostet deutlich mehr.

7 Firm- und Software

7.1 Präambel

In den folgenden Abschnitten werden die Firm- und Software oft als eine Einheit betrachtet, da diese sehr nahe miteinander verbunden sind. Dies hat den Vorteil, dass es die Komplexität der einzelnen Bauteile reduziert und somit das größere Gesamtbild besser erkennen lässt. In Abschnitten, die spezifisch den Unterschied zwischen Firm- und Software hervorheben, werden diese jedoch normal als einzelne Teile betrachtet, welche auch auf unterschiedlicher Hardware sind.

7.2 Benötigtes Vorwissen

Um einige Konzepte der Programmierung zu verstehen ist es wichtig, dass ein gewisses Vorwissen zum OOP existiert. Ebenso sollte eine Gewisse Vertrautheit mit dem UNIX System bestehen, da GNU/Linux einfach ausgedrückt ein UNIX-Klon ist, sollte es reichen, wenn schon mit Linux gearbeitet wurde. Im Besonderen ist es wichtig das Konzept von Dateien verstanden zu haben.

Hier eine kurze Zusammenfassung dieses Konzepts: Auf UNIX oder GNU/Linux ist alles eine Datei, ein Ordner ist eigentlich auch eine Datei, ein Programm, ein USB-Stick, und auch angeschlossene Geräte, wie etwa ein Drucker oder eine Maus. Um mit dem Gerät zu kommunizieren wird einfach in diese Geräte-Datei geschrieben. Dies wird in der Software des CAiRO an zwei Kernstellen genutzt.

7.3 Anforderungen

Die Anforderungen an die Steuerung sind relativ simpel. Ziel ist es eine Verbindung zwischen Computer und Roboter herzustellen, um diesen dann fernsteuern zu können. Der Roboter wird dann die Kommandos verarbeiten und verschiedene Befehle ausführen, dazu gehört das Laden und Ausführen von Programmen, welche definieren wie sich der Roboter bewegt. Ebenso soll es möglich sein dieses Programm leicht um ein Neuronales Netzwerk zu erweitern, da dies eines der großen Nebenziele ist. Diese Steuerung soll dann die Servopositionen an die µController weitergeben, welche für die Bewegung der Beine zuständig sind. Dabei soll die gesamte Verarbeitung und Übermittlung der Daten eine Latenz von weniger als 200ms aufweisen.

Abbildung 59 Befehlskette unter den Einzelnen Hauptprogrammen.

Die Fernbedienung schickt die Befehle an den Main-Controller, welche dieser dann weiterverarbeitet und transformiert, um sie dann an die Servos weiter zu leiten (eigentlich die PICs, jedoch haben diese nur ein einfaches Steuerprogramm, welches in dieser simplen Darstellung ignoriert werden kann).

7.4 Programmiersprachen

Da das Projekt von Anfang an auf die Benützung eines Neuronalen Netzwerkes ausgelegt wurde, musste gewährleistet werden, dass dieses auch leicht eingebunden werden kann, wenn die Muss-Ziele erfüllt wurden. Da als API TensorFlow ausgewählt wurde, fiel die Entscheidung rasch auf Python für den Hauptkontroller.

Für die Programmierung der Fernbedienung fielen wir schon bald auf Java zurück, weil es auf verschiedenen System, und nicht ausschließlich auf einem OS (z.B. Windows, GNU/Linux, macOS), ausführbar ist. Dies erleichtert die Arbeit, da nicht drei verschiedene Versionen des Programmes für die drei großen Operating Systems geschrieben werden müssen. Ein weiterer Vorteil von Java ist es, dass es eine einfache und Umfangreiche GUI-Library hat, welche das rasche Prototyping ermöglicht.

Die Firmware der PiC-µController kann entweder in Assembly oder in C geschrieben werden. Microchip (der Hersteller der Controller) stellt einen Assembler (mpasm) und einen C-Compiler (XC8) zur Verfügung. Da es im Prinzip keinen Unterschied zwischen dem fertigen Programm gibt, wenn es in C oder in Assembly geschrieben wird, entschieden wir uns für die einfachere Variante, C.

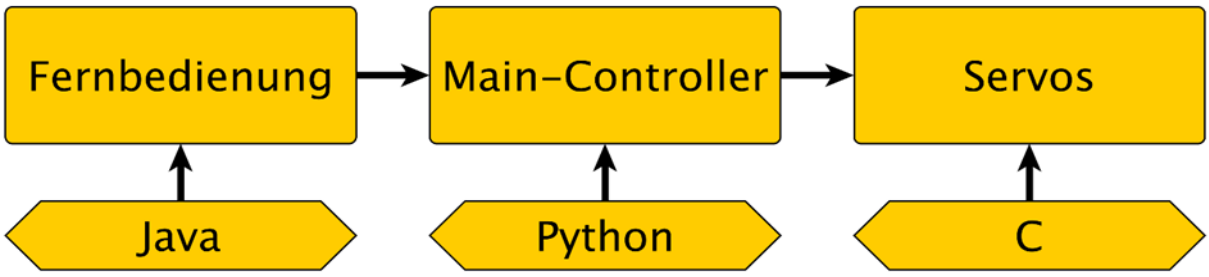

Abbildung 60 : Abbildung 59 um die Programmiersprachen pro Haupteinheit erweitert.

Zu bemerken ist, dass sowohl Java, als auch Python Objektorientierte Sprachen sind, während C eine rein prozedurale Sprache, ohne objektorientierten Einflüssen ist (es ist jedoch mithilfe von Makros auch in C möglich Objektorientiert zu schreiben, siehe das GTK+ UI Framework).

7.5 Schnittstellen

Die Verbindung zwischen den einzelnen Haupteinheiten wird über diverse Schnittstellen aufgebaut. So kommunizieren die Fernbedienung und der Main-Controller über WLAN mit dem Telnet-Protokoll, während der Main-Controller und die Servos über RS232 kommunizieren.

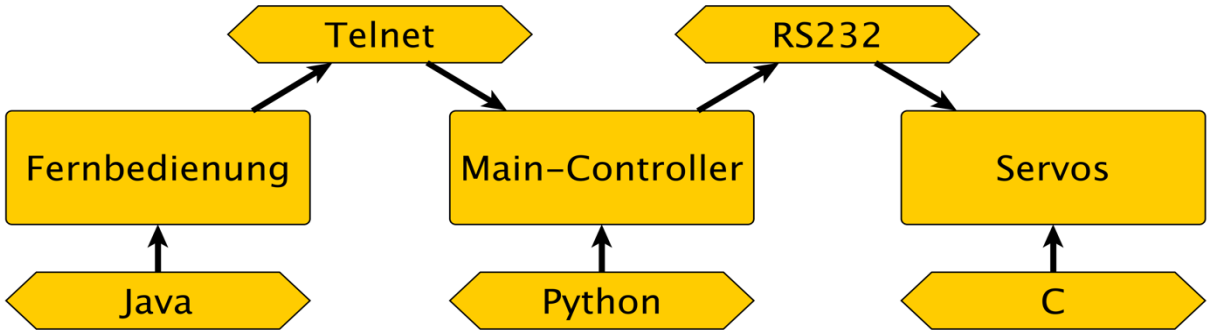

Abbildung 61 Die Verwendung der Schnittstellen im generellen Kommunikationsdiagramm.

Das Telnet-Protokoll ist eine einfache, ASCII-Text basierte Übertragungsmethode, bei der Strings mit '\r\n' am Ende übertragen werden. Dies ist vorteilhaft, da nur einzelne Kommandos übertragen werden müssen. So kann ein Kommando beispielsweise "fwstart" oder "fwselect forward" heißen, wobei das erste Wort das am Main-Controller registrierte Kommando ist und jedes weitere Wort ein Argument dieses Kommandos ist.

RS232 ist eine Schnittstelle die zur Datenübertragung zwischen genau zwei Geräten gedacht ist. Unsere Konfiguration hat jedoch drei PiCs als Slaves eingebaut. Dies erfordert eine spezielle Vorschaltung an der Hardware (#LINK HARDWAR) und hat ein eingeschränktes, oder zumindest komplizierteres Protokoll auf der Softwareseite zur Folge. Deshalb werden immer nur zwei Bytes an die Motorkontroller geschickt, diese enthalten sowohl Metadaten, als auch ein Byte an Information.

bit	7	6	5	4	3	2	1	0
byte 0	1	PiC addr.		servo addr.		mode		pos.msb
byte 1	0	pos						pos.lsb

Tabelle 11 Das Übertragungsprotokoll von Servopositionen.

7.6 Aufbau des Main-Controllers

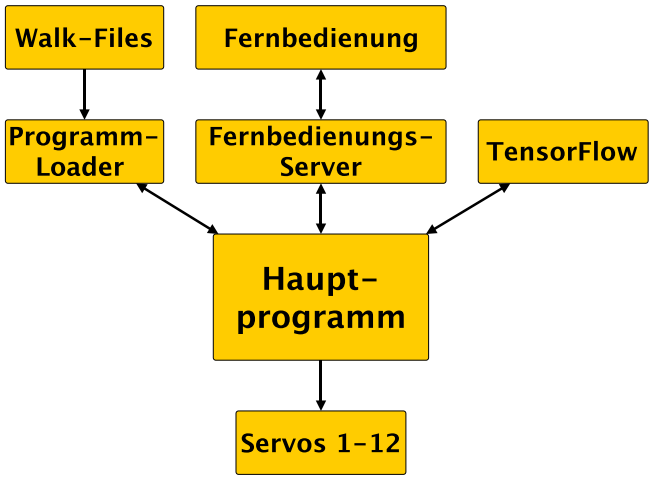

Abbildung 62 Übersicht der Module des Hauptprogrammes

Die Applikation wurde von unten aufgebaut, da es wichtig war so früh wie möglich erste Resultate zu sehen, um etwaige Fehler zu erkennen. Das erste Modul für den Main-Controller war das Walkie-Talkie welches für die Kommunikation mit dem Microcontroller zuständig war.

Der Einfachheit wegen wird in dieser Dokumentation jedoch von Oben nach Unten beschrieben. Zuerst generell mit vielen Abstraktionen, um es besser vermitteln zu können.

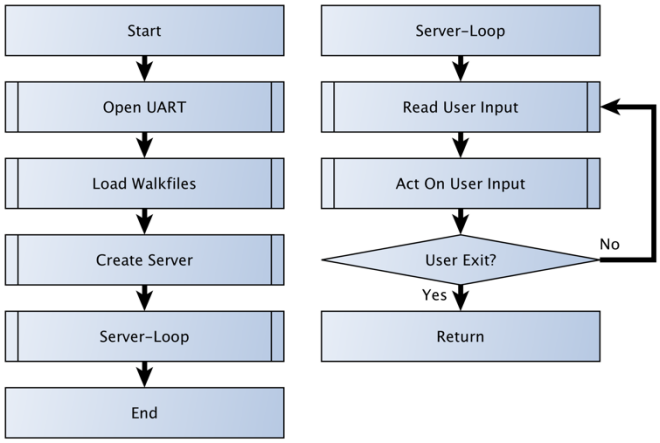

Abbildung 63 Grobe Übersicht des Main-Controllers.

Der Main-Controller beginnt damit alle Ressourcen zu laden. Dies beinhaltet das Öffnen des File-Deskriptors der UART-Verbindung und das Laden der Walkfiles. Danach wird der Kommunikationsserver gestartet, welcher dann auf Verbindungen wartet und Informationen verarbeitet, um diese dann an das darunterliegende Walkie-Talkie-Modul zu geben.

7.6.1 Walkfiles

Da die Walkfiles komplexe Datensätze sind, wurden gleich mehrere Klassen im Walkie-Talkie definiert, welche diese als Objekte darstellen können, um die Arbeit damit zu vereinfachen. Das ist auch der Grund, warum das Walkie-Talkie nahezu 900 Zeilen hat und mit Abstand der größte Teil der Software ist.

Im Folgenden werden alle Klassen des Walkie-Talkie-Moduls aufgeführt und deren Funktion genauer erklärt:

- Step: Die Step-Klasse stellt einen Schritt des Roboters dar, wobei einfach die Motorpositionen als Array abgespeichert werden. Da nicht jeder Schritt ohne eine zeitliche Verzögerung dem nächsten Folgt, wird auch noch eine Verzögerung pro Schritt definiert. Diese besagt wie lange nichts passieren soll, nachdem der Schritt gemacht wurde.
- Function: Diese Klasse stellt eine Funktion dar, welche die Bewegung eines Motors beschreibt. Sie stellt eine andere Möglichkeit dar, die Schritte zu definieren, wurde jedoch in keinem der Standard-Walkfiles benützt.
- MotorFunctions: MotorFunctions ist eine Erweiterung der Function-Klasse und beschreibt ebenfalls nur einen Motor, kann jedoch mehrere Funktionen beinhalten, wobei jede Funktion ein dazugehöriges Intervall haben, in denen sie gilt. So können komplexere Bewegungsabläufe mit den Funktionen programmiert werden.
- Program: Das Programm stellt das Walkfile als Python-Objekt dar. Dieses wird erstellt, wenn die Walkfiles beim Start des Main-Controllers geladen werden. Es beinhaltet eine Liste an nacheinander auszuführender Schritte (Steps) und/oder eine Liste an Motor-Funktionen (MotorFunctions), wobei jede einzelne dieser MotorFunctions für einen Servo zuständig ist. Je nachdem welchen Wert die Use-Variable im Programm hat, wird entweder die Step-, oder die MotorFunctions-List verwendet.
- Walker: Der Walker ist eine ABC (Abstract Base Class) zum Ausführen eines Programmes. Eine ABC wurde erstellt, da auch das Einbinden von TensorFlow ein optionales Ziel war und in Folge dessen ein TFWalker erstellt worden wäre. Zu dem kam es jedoch nicht, da es an Zeit für die Umsetzung eines Neuronalen Netzes mangelte. Es existiert nur der FileWalker, welche Programme ausführt.
- FileWalker: Der FileWalker wird dazu benutzt ein Programm auszuführen, dies tut er nach den Vorgaben des Programmes und wird periodisch mit einer doTick() Methode ausgeführt.
- MotorDistributor: Der MotorDistributor ist ein Modul, welches dazu zuständig ist die richtigen Werte an die Motoren zu versenden, um dies einfacher und abstrakter zu machen. Dies erwies sich als hilfreich, da im Entwicklungsprozess

die Methode zur Kommunikation mit den Microcontrollern dreimal geändert wurde und somit nur diese Klasse angepasst werden musste.

Um einen FileWalker zu erstellen könnte für Testzwecke folgender Code Verwendet werden:

```
862   ua = uart.Uart('./uart-out.bin')
863   print('  opening: ' + str(ua.open()))
864   motd = MotorDistributor(ua)
865   fw = FileWalker(motd)
866   print('  loading: ' + str(fw.loadProgram('./walkfiles/test.walk')))
867   print('  loading: ' + str(fw.loadProgram('./walkfiles/mot-ex.walk')))
868   print('selecting: ' + str(fw.selectProgram('Mot Test')))
869   while 1:
870           fw.doTick()
```

Abbildung 64 Test-Code aus den frühen Entwicklungsstufen des Main-Controllers

Wenn der Server gestartet wird, so beginnt er damit, alle nötigen Daten zu laden und zu initialisieren. Der Output sieht etwa wie folgt aus:

```
user src$ python main_controller.py -w ../../walkfiles/ -d /dev/null
set input file to stdin
set output file to stdout
set error file to stderr
[INFO][2017-04-03/15:17:36]:Opening uart connection on: /dev/null...
[INFO][2017-04-03/15:17:36]:Creating motor distributor...
[INFO][2017-04-03/15:17:36]:done
[INFO][2017-04-03/15:17:36]:Creating file walker...
[INFO][2017-04-03/15:17:36]:done
[INFO][2017-04-03/15:17:36]:Loading programs from '../../walkfiles/' ...
[INFO][2017-04-03/15:17:36]:found file: 000-idle.walk
[INFO][2017-04-03/15:17:36]:trying to load...
[INFO][2017-04-03/15:17:36]:loaded
[INFO][2017-04-03/15:17:36]:found file: 001-stand.walk
[INFO][2017-04-03/15:17:36]:trying to load...
[INFO][2017-04-03/15:17:36]:loaded
[INFO][2017-04-03/15:17:36]:found file: 102-jump.walk
[INFO][2017-04-03/15:17:36]:trying to load...
[INFO][2017-04-03/15:17:36]:loaded
[INFO][2017-04-03/15:17:36]:found file: 103-test.walk
[INFO][2017-04-03/15:17:36]:trying to load...
[WARN][2017-04-03/15:17:36]:mot is deprecated!
[INFO][2017-04-03/15:17:36]:loaded
exit
[DBUG][2017-04-03/15:17:39]:sending data: "['exit']"
[DBUG][2017-04-03/15:17:39]:command: ['exit']
[INFO][2017-04-03/15:17:39]:stopping server
[DBUG][2017-04-03/15:17:39]:sending data: 'stopping server'
[INFO][2017-04-03/15:17:39]:stopping filewalker
[DBUG][2017-04-03/15:17:39]:sending data: 'stopping filewalker'
user src$
```

Abbildung 65 Output des Main-Controlers wenn er gestartet wird.

7.7 Design der Walkfiles

Es stand schon relativ früh fest, dass ein eigenes Dateiformat für die Definition der Bewegung des Roboters eingeführt werden würde. Nicht nur bietet dies den Vorteil, dass schnell und einfach neue Programme geladen werden können, sondern trennt dieses Design auch das eigentliche Programm von den Daten, was bei einem fix im Code eingebundenen Programm nicht möglich wäre.

7.7.1 Anforderungen

Ein Walkfile muss eine Definition zur Initialisierung des Programms und eine Ausführsequenz enthalten. Die Initialisierung soll einen Befehl enthalten, welcher einmal zu Beginn des Programmes ausgeführt wird, während das Hauptprogramm mehrere Schritte ausführen kann und optional nach Ablauf wieder von vorne beginnt.

7.7.2 Umsetzung

Wir entschieden uns für eine leserliche und einfach zu parsende Lösung. Das Programm besteht aus mehreren Blöcken, welche durch eckige Klammern angegeben werden. Variablen werden mit dem '=' Operator bezeichnet und schritte als einfache Folge von Werten, wo jeder Wert die Position eines Motors angibt.

Die erste Version der Walkfiles definiert folgende Blöcke:

- info
- setup
- main
- mot0 - mot11

Info-Block

```
5   [info]
6           Name=idle
7           Id=000
8           Speed=0
9           Version=0.1
10          Tick=0
11          Looping=false
12          Use=prg
13  [end]
```

Abbildung 66 Der Infoblock des "idle" Programms, welches alle Servos ausschaltet.

Im Info-Block werden generelle Informationen zum Programm angegeben, wie etwa der Name, die Versionsnummer oder die Identifikationsnummer. Ebenfalls wird angegeben, ob eine Sequenz an Schritten zur Bestimmung der Servopositionen oder Funktionen verwendet werden sollen.

Setup-Block

```
15  [setup]
16          >0..12,
17  [end]
```

Abbildung 67 Setup-Block

Der Setup-Block definiert die Anfangspositionen eines Programmes, hier sieht man ein simples Beispiel für eine Anweisung zum Setzten der Servos. Diese Anweisung wird immer mit einer spitzen Klammer eingeleitet und besteht danach aus einer, durch Kommas separierten Liste, welche die Positionen eines Servos angibt. Der erste Eintrag bezieht sich auf Servo 1, der zweite auf Servo 2, etc. Da es jedoch oft vorkommt, dass mehrere Servos denselben Wert haben, ist es auch möglich an die Position zwei Punkte anzuhängen, dies ermöglicht die Spezifikation von den nächsten 'n' Servos, wobei die Anzahl der Servos direkt nach den Punkten angehängt wird.

Die Position wird immer als Winkel angegeben, diese kann in unterschiedlichen Einheiten angegeben werden, dazu zählt der simple PWM-Wert, welcher am µController eingestellt wird, der Winkel in Grad oder im Bogenmaß (dieser Wert wird als Integer in 1/1000 Radiant angegeben). Um die Angabeform auszuwählen wird ein Präfix an die Position vorgestellt. So wird 'd' für Grad verwendet, 'r' für Radiant und kein Präfix für den rohen PWM-Wert.

Wollte man nun alle Servos auf 90 Grad stellen, so könnte es wie folgt geschrieben werden:

Form 1	Form 2	Form 3
>d90..12,	>r1571..12,	>d90,d90,d90,d90,d90,d90,d90,d90,d90,d90,d90,d90,

Tabelle 12 Anschriftsformen der Servo-Befehle

Programm-Block

Der Programm-Block enthält Instruktionen für die Bewegung der Servos wie sie schon im Setup-Block gezeigt wurden. Um ein individuelles Timing zu bewerkstelligen kann an jede Instruktion noch mit einer Wartezeit ausgestattet werden, bevor die nächste Instruktion ausgeführt werden soll. Dieses Delay wird mit einem Doppelpunkt und dem Wert in Millisekunden an den Befehl angestellt. Falls keines Spezifiziert wird, so wird die Standardverzögerung verwendet, welche im Info-Block definiert ist.

Sollte das Ende des Programms erreicht worden sein, so wird dieses entweder beendet, oder noch einmal gestartet, je nachdem ob die Looping Variable im Info-Block gesetzt wurde.

```
24  [prg]
25          >d70,d50,d116,d94,d117,d126,d130,d77,d134,d126,d44,d92,:20
26          >d110,d50,d116,d94,d117,d126,d130,d117,d94,d126,d44,d92,:50
27          >d110,d50,d116,d134,d77,d126,d130,d117,d94,d126,d44,d132,:20
28          >d110,d90,d156,d134,d77,d86,d170,d117,d94,d86,d84,d132,:70
29          >d110,d90,d156,d94,d117,d86,d170,d117,d94,d86,d84,d92,:50
30          >d70,d90,d156,d94,d117,d86,d170,d77,d134,d86,d84,d92,:20
31          >d70,d50,d116,d94,d117,d126,d130,d77,d134,d126,d44,d92,:50
32  [end]
```

Abbildung 68 Die Ablaufdefinition des standardmäßig eingestellten Schrittverlaufes.

7.7.3 Ladevorgang

Das Laden der Programme ist relativ simpel gehalten, sie werden vor dem Start des Servers nacheinander eingelesen, wobei das Einlesen pro Block aufgeteilt ist. Dies macht es einfacher sie zu lesen, ohne eine hohe Codekomplexität einzuführen. Die Ladefunktion ist als State-Maschine aufgebaut, wobei jede Zeile einzeln gelesen wird. Gespeichert werden die Schritte als eine Liste von Schritten, wobei jeder Schritt im Prinzip ein Array von 12 Werten sind.

7.7.4 Generierung

Die Walkfiles von Hand zu schreiben ist mühsam, deshalb wurden die Positionen mit Excel ausgerechnet und visualisiert. Danach wurden sie als CSV-Dateien exportiert und mit zwei Shell-Scripts als Walkfiles konvertiert. Dies bietet den Vorteil, dass die Programme selbst schnell und leicht angepasst und veranschaulicht werden können, während sie immer noch einfach für den Main-Controller zu laden sind. Das erste Skript ist für die reine Konversion zuständig, während das zweite für das Zusammenfassen von aufeinanderfolgenden Doppeleinträgen zuständig ist. Da diese Scripts den Input vom Standardinput lesen ist es möglich sie mittels einer Pipe aneinanderzuketten.

7.8 Bewegungsmuster

Damit der Roboter sich fortbewegen kann wurden Bewegungsmuster festgelegt, an die er sich hält. Da sich das Team vornahm mindestens zwei Bewegungsmuster zu erstellen, wurde nach unterschiedlichen Möglichkeiten gesucht. Jedes Muster ist aus einer bestimmter Grundidee entstanden.

7.8.1 Ablauf der Bewegungen

In diesem Kapitel werden die erstellten Bewegungsmuster beschrieben. Dabei wird auch darauf eingegangen wie jedes funktioniert, und welche Eigenschaften es hat.

7.8.1.1 Natürliches Bewegungsmuster

Der Roboter ähnelt von der Anatomie her sehr der eines Insekts, darum wurde der Gang auch von dem einer Ameise inspiriert. Dabei folgt er einem einfachen Muster. Die Beine sind in zwei Gruppen unterteilt. Für die folgende Erklärung werden sie A und B genannt. A besteht aus dem linken vorderen und hinteren, sowie dem rechten mittleren Bein. Gruppe B setzt sich aus den restlichen Beinen zusammen. Also links Mitte, rechts Vorne und Hinten.

1. A ist gehoben und bewegt sich nach vorne, während B simultan, am Boden stehend, nach hinten bewegt. Das resultiert darin, dass der Roboter nach vorne geschoben wird.
2. In den Endpositionen angekommen senkt sich Gruppe A und Gruppe B hebt sich.
3. Es wiederholt sich der erste Schritt nur mit A und B vertauscht. A schiebt den Roboter nach vorne während B sich selbst nach vorne hebt.
4. A hebt sich und B senkt sich.

Der Vorgang wiederholt sich solange der Roboter gehen soll. Um rückwärts zu gehen wird der ganze Vorgang rückwärts abgearbeitet. Diese Gangart ermöglicht ein beinahe ständiges Fortbewegen und somit eine hohe Geschwindigkeit. So, dass die Gruppe die gerade am Boden steht, sich nach vorne bewegt. Dadurch wird der Roboter nach hinten geschoben. Um eine Kurve zu gehen, wird der normale Ablauf des Vorwärtsgangs ausgeführt, mit dem Unterschied, dass die Beine auf der rechten beziehungsweise linken Seite nur halb so große Schritte machen. Außerdem kann sich der Roboter am Stand drehen. Dies wird durch eine Kombination von Drehen und Rückwärtsgehen erreicht.

Das Programm sieht vor, dass der Ablauf wie beim Gehen einer Kurve normal ausgeführt wird. Der Unterschied liegt darin, dass anstatt auf einer Seite nur halb so große Schritte gemacht werden, wird diese wie beim Rückwärtsgehen bewegt. Das erzeugt eine Vorwärtsbewegung auf der einen Seite des Roboters und eine Rückwärtsbewegung auf der anderen. Daraus folgt unweigerlich eine Drehung. Da alle Schritte dieselbe Länge haben und auch gleichschnell durchgeführt werden, entsteht keine Translation des Körpers.

7.8.1.2 Langsames Bewegungsmuster

Es wurde auch eine zweite Gangart implementiert. Dazu werden die die Beine in drei Gruppen geteilt: Vorne, Mitte und Hinten. Diese Bewegungsart kann nicht direkt flüssig aus dem Stand angesteuert werden, da die Grundposition des Standes und der zweiten Gangart unterschiedlich ist. Allerdings haben Tests ergeben, dass es keinen negativen Effekt gibt, sollte dies trotzdem geschehen. Darum wird auch bei der Anwendung direkt vom Stand in die langsame Gangart gewechselt.

0. Alle Beine befinden sich in der hintersten Position.
1. Die hinteren beiden Beine heben sich und setzen sich hinter die mittleren.
2. Die mittleren beiden Beine heben sich und setzen sich hinter die vorderen.
3. Die vorderen beiden Beine heben sich und setzen sich ganz nach vorne.
4. Alle Beine bewegen sich simultan nach hinten um den Roboter nach vorne zu schieben.

Auch dieser Vorgang kann beliebig oft wiederholt werden. Der große Vorteil dieser Gangart ist, dass sie links und rechts immer symmetrisch ist. Dadurch kann keine ungewollte Drehung in die Fortbewegung mit einfließen. Es ist wichtig, dass sich zuerst die hinteren Beine bewegen und sich dann erst die andern. Würden zuerst die vorderen Beine weitersteigen verliert der Roboter während dem Schritt sein Gleichgewicht. Auch ist es zu beachten, dass sich die Beine nicht zu weit bewegen. Da auch dann das Gleichgewicht gefährdet ist. Der große Nachteil dieser Gangart ist die Geschwindigkeit. Ein großer Teil des Schrittablaufs wird für das Ausrichten der Beine gebraucht. Da aber alle Beine gleichzeitig den Roboter nach vorne schieben, arbeiten auch mehrere Motoren zusammen. Das resultiert in einer höheren Zugkraft.

7.8.2 Erstellung der Bewegungsmuster

Die Bewegungsmuster wurden aufgrund eines einfachen Prinzips erstellt. Es wurde eine Tabelle in Microsoft Excel (Microsoft, 2017) erstellt. Die erste Spalte bestimmt die Zeit in Millisekunden, die anderen die Winkel der einzelnen Servomotoren. Daraufhin werden diese Werte mit vordefinierten Kalibrierungswerten verändert. Diese Werte werden aus dem Aufbau es Roboters entnommen. Der Wert „Orientierung“ kehrt die Drehrichtung des Servos um und der Wert „Offset“ verschiebt den Nullpunkt. Durch das Anwenden dieser Werte kann ein Programm unabhängig des tatsächlichen Einbaus der Servomotoren designet werden. Daraus folgte eine Liste für die Position aller Servomotoren zu bestimmten Zeitpunkten. Diese wurden dann in eine .walk-Datei umgewandelt. Wichtig bei der Erstellung dieser Listen ist, dass bei einer Bewegung immer die Endposition angegeben wird um ein Stocken der Bewegung zu verhindern, sollte der Servomotor schneller sein als das Programm.

Position	Seite	Richtung	Name	Orientierung	Offset
Vorne	Links	Vertikal	AA	-1	100
Vorne	Links	Horizontal	AB	-1	70
Vorne	Rechts	Horizontal	AC	1	136
Vorne	Rechts	Vertikal	AD	1	104
Mitte	Links	Vertikal	BA	-1	107
Mitte	Links	Horizontal	BB	-1	106
Hinten	Links	Horizontal	BC	-1	150
Hinten	Links	Vertikal	BD	-1	107
Mitte	Rechts	Vertikal	CA	1	104
Mitte	Rechts	Horizontal	CB	1	106
Hinten	Rechts	Horizontal	CC	1	64
Hinten	Rechts	Vertikal	CD	1	102

Tabelle 13 Servobelegung und Kalibrierungswerte

T	AA	AB	AC	AD	BA	BB	BC	BD	CA	CB	CC	CD
10	30	20	-20	-10	-10	-20	20	30	30	20	-20	-10
20	30	20	-20	-10	-10	-20	20	30	30	20	-20	-10
30	-10	20	-20	-10	-10	-20	20	-10	-10	20	-20	-10
40	-10	20	-20	-10	-10	-20	20	-10	-10	20	-20	-10
50	-10	20	-20	-10	-10	-20	20	-10	-10	20	-20	-10
60	-10	20	-20	-10	-10	-20	20	-10	-10	20	-20	-10
70	-10	20	-20	-10	-10	-20	20	-10	-10	20	-20	-10
80	-10	20	-20	30	30	-20	20	-10	-10	20	-20	30
90	-10	20	-20	30	30	-20	20	-10	-10	20	-20	30
100	-10	-20	20	30	30	20	-20	-10	-10	-20	20	30
110	-10	-20	20	30	30	20	-20	-10	-10	-20	20	30
120	-10	-20	20	30	30	20	-20	-10	-10	-20	20	30
130	-10	-20	20	30	30	20	-20	-10	-10	-20	20	30
140	-10	-20	20	30	30	20	-20	-10	-10	-20	20	30
150	-10	-20	20	30	30	20	-20	-10	-10	-20	20	30
160	-10	-20	20	30	30	20	-20	-10	-10	-20	20	30
170	-10	-20	20	-10	-10	20	-20	-10	-10	-20	20	-10
180	-10	-20	20	-10	-10	20	-20	-10	-10	-20	20	-10
190	-10	-20	20	-10	-10	20	-20	-10	-10	-20	20	-10
200	-10	-20	20	-10	-10	20	-20	-10	-10	-20	20	-10
210	-10	-20	20	-10	-10	20	-20	-10	-10	-20	20	-10
220	30	-20	20	-10	-10	20	-20	30	30	-20	20	-10
230	30	-20	20	-10	-10	20	-20	30	30	-20	20	-10
240	30	20	-20	-10	-10	-20	20	30	30	20	-20	-10
250	30	20	-20	-10	-10	-20	20	30	30	20	-20	-10
260	30	20	-20	-10	-10	-20	20	30	30	20	-20	-10
270	30	20	-20	-10	-10	-20	20	30	30	20	-20	-10
280	30	20	-20	-10	-10	-20	20	30	30	20	-20	-10

Tabelle 14 Bewegungsablauf

7.9 Kommunikationsserver

Wie schon in vorhergehenden Kapiteln erwähnt wird für die Kommunikation Telnet verwendet, was den Aufbau des Servers simpler macht. Es wird periodisch der Input, der über die TCP-Verbindung eingeht ausgelesen und als String-Variable gespeichert. Wenn ein "\r\n" empfangen wird, werden die Charakters vom Anfang des Strings bis zum ersten "\r\n" als Kommando aufgefasst und weiterverarbeitet, der Rest der Informationen bleib in dem Stringbuffer zurück. Dieses Design hat den Vorteil, dass einzeln erhaltene Informationen, die im Netzwerk minimale Zeitabstände haben nicht zerschnitten werden und somit den Befehl zerstören. Eine weitere Bewusste Einschränkung ist es, dass immer nur ein Client eine Verbindung zur Fernbedienung haben kann. Dies verhindert mehrere Kommandos, welche gleichzeitig empfangen werden könnten und somit zur Unkontrollierbarkeit des Roboters führen könnte.

Um einfach eine Verbindung zwischen Fernbedienung und Server herstellen zu können, sendet der Server jede Sekunde ein UDP-Paket an die Broadcasting-Adresse des Routers. Dieses Paket ist somit für alle Geräte im Selben Netzwerk ersichtlich und beinhaltet den Port auf dem der Server zurzeit Clients annimmt. Durch zwei Klicks auf zwei Knöpfe auf der Fernbedienung kann sich der Bediener leicht mit dem Roboter verbinden und sofort anfangen zu steuern.

Aufgrund der geringen Anforderungen ist der Server sehr archaisch gehalten. Dies hat zur Folge, dass er nur wenig Code zur Umsetzung benötigt. Der Server ruft periodisch 3 Funktionen auf, welche für die Annahme einer neuen Verbindung und das Verarbeiten von Kommandozeileneingaben von dem Roboter, als auch von der Fernbedienung. Hierzu wird ein Hilfsmodul verwendet, welches Usereingaben vereinfacht und ein Interface für die Registrierung von Befehlen bietet. Somit muss nur noch ein Objekt erzeugt werden, welches einen Befehl ausführt, welches dann bei der Kommandozeile registriert wird.

Die Folgenden Befehle sind im Programm implementiert

- fwstart: Nimmt keine Parameter entgegen und startet den FileWalker, dies startet sofort ein Programm, wenn es zuvor mit fwselect ausgewählt wurde. Wenn sich

die Fernbedienung verbindet schickt sie diesen Befehl automatisch an den Main-Controller.

- fwstop: Nimmt ebenfalls keine Parameter entgegen und stoppt den FileWalker. Wenn zuvor ein Programm ausgeführt wurde, dann wirkt dies wie ein Pausieren, durch eingabe des Befehls fwstart wird dieses dann wieder an der letzten Position ausgeführt.
- fwselect: Wählt ein Programm aus und übernimmt daher ein Argument, den Namen des Programmes. Falls dieses nicht existiert wird eine Fehlermeldung ausgegeben und kein Programm wird geladen. Wenn kein Argument übergeben wird, dann wird das zurzeit ausgeführte Programm abgebrochen und abgewählt.
- fwdeselect: Ist ein alias für den Befehl fwselect, welcher ohne Argumente ausgeführt wird.
- servo: Das Kommando wird dazu benutzt einzelne Servos zu setzen und nimmt daher zwei Argumente entgegen. Das erste Argument gibt an welcher Servo verstellt werden soll, während das zweite Argument Stellung in Winkelgrad angibt. Dieser Befehl wurde alleine zum Debugging benutzt und ist für den Endbenutzer unwichtig.
- dostep: Führt einen Schritt eines durch fwselect ausgewählten Programmes aus, wenn der FileWalker nicht durch fwstart gestartet wurde. Auch dieser Befehl wurde nur zum Debugging der Gehmuster genutzt und ist nicht wichtig für den Endbenutzer.
- exit: Der Exit-Befehl ist ein theoretisch ungenutzter Befehl und schließt den Server.

7.10 Command Line

Die Command-Line ist ein Nebenmodul, welches vom Server genutzt wird und das Ausführen von Befehlen erleichtert. Sie bietet zwei Klassen, den Command, welcher eine ABC ist und den Command-Handler, welcher den Input des Benutzers verarbeitet und an die einzelnen registrierten Commands weitergibt. Da der Command-Handler Dateien als InputQuelle nutzt ist es möglich mehrere InputQuellen zu nutzen, so zum Beispiel eine einfache Datei, die Kommandozeile, oder auch die Client-Verbindung.

7.11 Erstellung eines Befehls

Um einen Command zu erstellen und danach im Command-Handler nutzen zu können, wird zuerst ein Command geschrieben, welcher dann beim Command-Handler registriert wird. Dies ist ein kleines Beispiel hierfür:

```python
#!/usr/bin/env python3

from cmd_line import *

class CommandName(Command):
        def __init__(self, hdlr):
                Command.__init__(self, hdlr)

        def do(self, argv):
                print("Hello from the command-line!")
                print("Arguments:")
                for x in range(0, len(argv)):
                        print("argument" + str(x) + ": " + argv[0])
                print("Bye!")

ch = DefaultCmdHandler
ch.regCmd('hello', CommandName(ch))
ch.overrideCmd('exit', CommandName(ch))
ch.run()
```

Abbildung 70 Testprogramm für die Command-Line

```
user$ python3 test.py

> hello
Hello from the command-line!
Arguments:
Bye!
> exit
Hello from the command-line!
Arguments:
Bye!
```

Abbildung 69 Output des Testprogrammes

7.12 Fernsteuerung

Die Fernsteuerung wurde in Java geschrieben und verbindet sich über das Transmission Control Protocol mit dem Python-Server. Aller Datenverkehr wird über das Telnet-Protokoll geregelt. Um den Roboter steuern zu können kann der Nutzer entweder im Gehtab die Knöpfe für die Entsprechende Bewegung drücken (und gedrückt halten), oder gleich mit WASD/QE die Bewegung steuern, oder direkt die richtigen Programme in der Kommandozeile laden. Sollte der Knopf oder die Taste losgelassen werden, so bleibt der Roboter automatisch stehen und wartet auf weitere Anweisungen.

7.12.1 Funktionsweise

Das Fenster der Fernbedienung ist auf vier Tabs aufgeteilt, wobei das erste Tab die anderen "freischält", also anwählbar für den User macht, nachdem dieser sich mit dem Roboter verbunden hat.

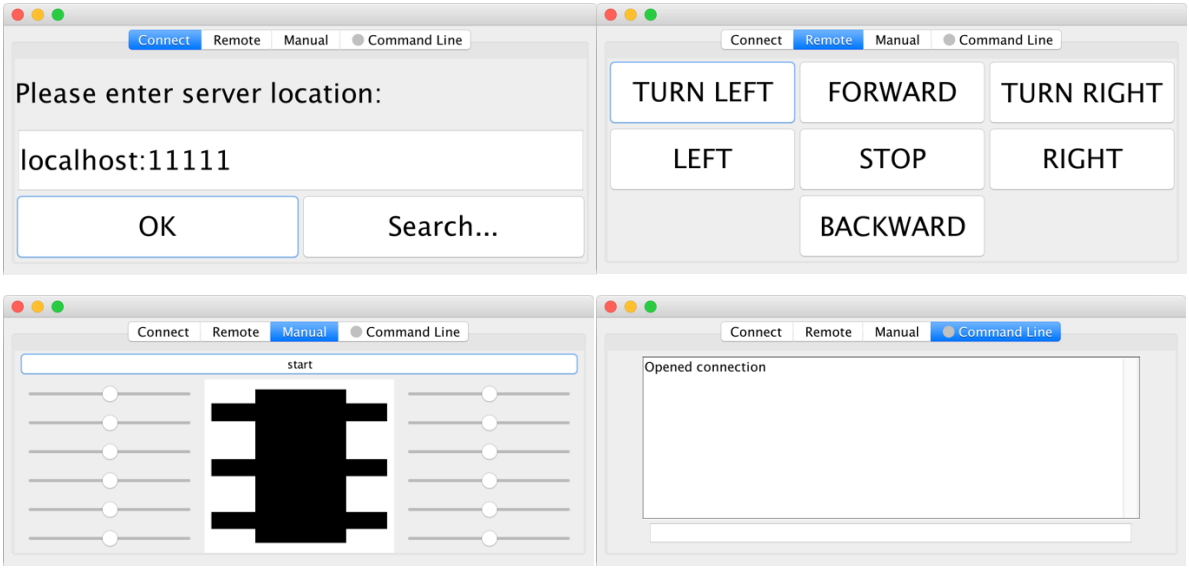

Abbildung 71 Übersicht aller verfügbaren Fenster der Fernbedienung.

Nachdem die Verbindung erfolgreich hergestellt wurde, muss der Nutzer den FileWalker starten. Dies geschieht im Manual Tab durch drücken des Start-Buttons. Danach kann in den Remote-Tab gewechselt werden um den Roboter zu steuern.

7.12.2 Umsetzung

Um die Steuerung durch Gedrückthalten der Knöpfe zu realisieren wurden ChangeListener auf den Button-Modellen registriert. Diese feuern immer dann ein Event, wenn der Knopf gedrückt/losgelassen wurde. Wenn ein Knopf-Gedrückt-Event gefeuert wird, so wird das entsprechende Programm geladen ("fwselect <program>"). Sobald ein Knopf losgelassen wird, wird der Befehl zum beendigen des Programmes gesendet ("fwdeselect").

Die Steuerung durch die Tastatur wird durch einen KeyListener, welcher an jedem Komponenten des Fensters registriert ist realisiert. Somit kommt dann auch hier ein Event für jeden Knopfdruck auf der Tastatur an. Sobald einer der Steuerungsknöpfe gedrückt wird, wird dann der Knopfdruck im Programm simuliert, um auch ein optisches Feedback zu geben.

7.13 Erster Integrationstest

Der erste Integrationstest beinhaltete den Kommunikationsserver, welche schon Walkfiles laden und nutzen sollte, ebenfalls sollten die Kommandos über die serielle Schnittstelle an die µController weitergegeben werden. Eine Verbindung über WLAN mit

einem WLAN-Stick war ebenfalls Ziel dieses Tests. Zu diesem Zeitpunkt wurden noch das Development-Board von Theobroma-Systems und ein serieller USB-Adapter genutzt, welcher an die Testplatine für die PiCs angeschlossen wurde.

Die Verbindung zum WLAN war sehr einfach, nachdem die richtigen Treiber für den WLAN-Stick installiert wurden. Da der selbst erstellte Linux-Kernel von Theobroma-Systems keine Kernelmodule dynamisch laden kann, war es nötig diese schon in dem Kernel zu kompilieren. Der Vorgang beinhaltet das laden der Kernelsource, deren Konfiguration und im Anschluss die Kompilation des Kernels. Resultat dieses Vorgangs ist eine binäre Datei, welche den Gesamten Kernel beinhaltet. Der zweite wichtige Baustein, der bei diesem Schritt entsteht, ist das initramfs. Dies ist ein fertiges Dateisystem, welches früh im Bootvorgang benötigt wird und im RAM des Computers lebt. Im Fall von CAiRO gab es schon ein initramfs, weshalb einfach nur der alte Kernel durch den neuen Kernel ersetzt wurde.

7.13.1 Generelle Struktur des Programmes

Nach einigen kleinen Anpassungen gelang die gesamte Prozedur der Kommunikation, vom Computer, von dem die Anweisungen kommen, zum µQ7-A64, welcher dann alle Anweisungen weiter an die PiC-Microcontroller weitergab. Zu diesem Zeitpunkt wurde noch die Terminal-Applikation "Telnet" verwendet, da unsere Fernbedienung noch nicht völlig ausgereift war.

7.13.2 Multithreading mit Python in eingebetteten Systemen

Die einzelnen Komponenten der Python Applikation liefen auf unterschiedlichen Threads, dies war durch die Entwicklungsschritte gegeben, welche es uns ermöglichten einzelne Komponenten zu testen, ohne die Anderen zu haben. So konnte etwa die UART-Verbindung über Python getestet werden, ohne eine WLAN Verbindung aufbauen zu müssen.

Bei der Zusammenführung der 3 Hauptteile kam es jedoch am integrierten Linux-Computer zu Problemen, da dieser nicht so viel Leistung wie das Entwicklungssystem besaß. Dies äußerte sich dadurch, dass die WLAN-Kommunikation sehr instabil war, weil der Python-Server nicht die hereinströmenden Daten verarbeiten konnte.

7.13.3 Lösung

Die Lösung war eine kleine Umgestaltung, welche die Anzahl der Threads von einem Hauptthread und zwei Nebenthreads auf einen Hauptthread reduzierte. Dieses Problem entstand jedoch nur dadurch, da Python alle Python-Threads auf einem Prozessorthread ausführte, anstatt, wie angenommen, auf mehreren Rechenkernen. Dieses Phänomen wurde jedoch erst auf dem Integrierten Computer ersichtlich, da hier wesentlich weniger Rechenkraft zur Verfügung stand als auf der Testmaschine.

7.14 Neudesign des UART Protokolls

Das UART-Übertragungsprotokoll wurde einmal umkonzeptioniert, da eine kommandozeilenähnliche Implementation auf dem Microcontroller platzverschwendend ist. Funktionen zum Bearbeiten und Analysieren von Strings, wie Beispielsweise 'strtok' sind nicht als Library vorhanden und verbrauchen somit wertvollen Programmspeicher. Ein weiterer Vorteil ist es, dass mit diesem Protokoll ein Multimaster-System mit mehreren Slaves auf einer einzigen RS232-Verbindung aufgebaut werden kann. Dies ist dadurch möglich, dass die eine Master-Slave Verbindung (zwischen Computer und Integriertem Linux) ausschließlich mittels ASCII kommuniziert, während die neue Implementation des Servo-Übertragungsprotokolls als ersten Charakter einen so genannte EASCII-Charakter benützt. Auch wenn diese 8-Bit-Integer von 128 bis 255 High-ASCII (oder Extended-ASCII) genannt werden sind sie doch nicht Teil des ASCII-Standards, welcher damals für TTYs entwickelt wurde, welche oft nur sieben Bit besaßen.

Diese Nicht-Standardisierung kann gut aufgezeigt werden, wenn eine Textdatei mit EASCII unter Windows erstellt wird und unter einem anderen Betriebssystem (wie macOS oder Linux) geöffnet wird.

Das Grundprinzip dieser neuen Version wurde schon unter Schnittstellen erklärt. Grund für dieses Design ist die Möglichkeit zwischen ASCII und dem eigenen Protokoll zu unterscheiden. Dies macht das komplizierte Multi-Master-Multi-Slave System auf der Schnittstelle möglich.

7.15 Erster Firmwaretest

Die Aufgabe jedes einzelnen PiC17F1827 ist es, jeweils 4 Servos zu kontrollieren. Dies wird mittels einer PWM erzielt. Für die Ansteuerung aller 4 PWM-Pins werden alle CCP-Module und ein Timer benötigt, da ein Timer für alle CCP-Module verwendet werden kann, andernfalls wäre es nicht möglich vier PWM-Ausgänge gleichzeitig anzusteuern.

7.15.1 PWM der Servos

Die PWM ist eine rechteckförmige Welle, bei welcher der Duty-Cycle geändert werden kann (Modulation). Dies wird bei Servos benutzt um die Position anzugeben, ein Duty-Cycle von 1ms ist generell bei 0°, während 2ms bei 180° liegen. Die Frequenz der PWM liegt standardmäßig bei 50Hz.

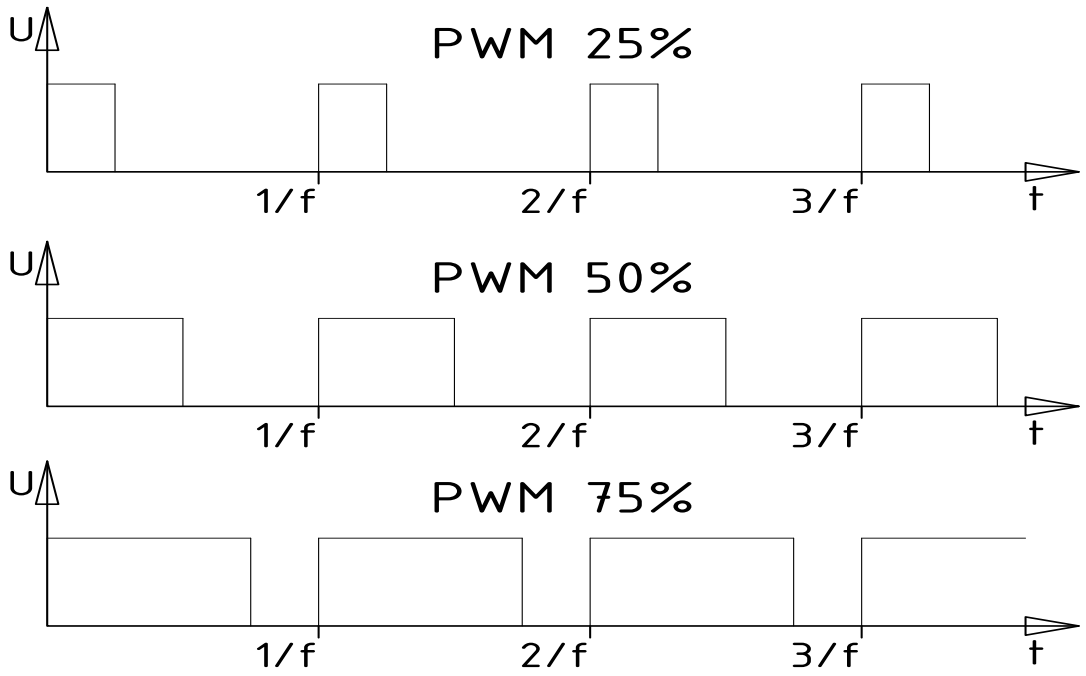

Abbildung 72 3 PWM mit jeweils 25, 50 und 75 Prozent Duty-Cycle.

7.15.2 Funktionsweise der Firmware

Die Firmware Initialisiert alle Module, welche benötigt werden, dazu zählt ein Timer (TMR6), 4 CCP-Module für die PWM und das UART-Modul für die Kommunikation. Danach wird auf den Befehl die Position eines Servos zu ändern gewartet. Der Befehl sieht wie folgt aus:

>servo [pic-id] [servo-id] [wert]<

Ausgesprochen könnte der Befehl "Fahre den Servo [servo-id], welcher an dem Prozessor [pic-id] angeschlossen ist auf die Position [wert]." lauten. Dieser Test wurde erfolgreich durchgeführt.

7.16 Kommunikationstests

Beim Kommunikationstest ging es darum, sicherzustellen, dass nicht nur über das serielle Terminalprogramm Picocom, sondern auch über Python kommuniziert werden kann.

7.16.1 Picocom

Picocom ist ein kleines Terminalprogramm, welches es erlaubt sich mit einem anderen Terminal, zu verbinden. Solche seriellen Terminals (VT-100 Terminals und co.) sind noch aus einer vergangenen Ära des Computers bekannt, heutzutage kommt der Normalbenutzer nicht mehr damit in Kontakt. Jedoch gibt es noch viele Tools und Schnittstellen, die damit kompatibel sind, nicht zuletzt die RS232 Schnittstelle und das Programm Picocom, welches damit kommunizieren kann.

Die richtige Verbindung über Picocom wurde schon beim vorhergehenden Test (Design der Walkfiles) überprüft und bestätigt.

7.16.2 Python - Pyserial

Da es uns jedoch auch möglich sein muss mit den µControllern über ein eigens geschriebenes Programm zu kommunizieren, um sie somit in einem größeren Programm steuern zu können, musste auch dies überprüft werden.

Die allgemein bekannte Bibliothek Pyserial erschien uns als richtiges Grundwerkzeug und wurde demnach getestet.

Die Kommunikation hat mit dieser Library jedoch nicht funktioniert. Trotz richtiger Konfiguration gelang es uns nicht zuverlässig eine Verbindung aufzubauen. Dies gelang in nur ca. 5% der Fälle, ohne eine Änderung des Programmes. Wir wissen nicht, ob Python oder Pyserial dies verursachten, entschlossen uns aber einen anderen Lösungsweg zu suchen.

7.16.3 C Bibliothek

Die Lösung für dieses Problem war eine eigens geschriebene C-Bibliothek, welche dann mit Ctypes eingebunden wurde. Die Bibliothek übernimmt die Low-Level Kommunikation und bietet ein minimalistisches Interface für Programme. Ihre Aufgabe ist es, das Device zu öffnen, vorzubereiten, Daten auszutauschen und schließlich wieder ordnungsgemäß zu schließen.

Der Header, welcher Vorgezeigt wird enthält die folgenden vier Funktionen:

```
int uart_open(const char *dev);

int uart_write(int hdl, uint8_t *dat, size_t len);

int uart_read(int hdl, uint8_t *dat, size_t len);

int uart_close(int hdl);
```

7.16.4 Datei

Die Letzte, finale Methode besteht darin ein schon konfiguriertes Gerät zu öffnen, was dazu führt, dass dieses wie eine Textdatei geöffnet und beschrieben werden kann. Grund für diese Änderung war es, dass beim Platinendesign die Kommunikationsschnittstelle auf den Terminal des OS' gelegt wurde, und somit beides über die selbe Leitung übertragen wird. In diesem Fall übernimmt das OS die Konfiguration des Devices. Ersichtlich wurde dieses Verhalten erst bei einem späteren Integrationstest.

8 Lessons Learned

In der Diskussion werden die Erfahrungen mitgeteilt welche jedes Teammitglied erhalten hat.

8.1 Mießkes

Das Projektmanagement ist, fasst man es zusammen, nur regelmäßiges Vergleichen und Anpassen. Keine der einzelnen Aufgaben ist weder besonders schwer noch kompliziert, allerdings wird von einem Projektmanagement viel abverlangt. Er muss immer wissen was gerade zu tun ist und wie weit jeder Schritt ist. Auch wenn das Team sehr selbständig

ist und kaum Überzeugungsarbeit geleistet werden muss um ein Teammitglied zum Weiterarbeiten zu bringen, ist es kompliziert alles zu koordinieren.

Das Team CAIRO war sehr selbstständig und engagiert. Meistens wurde schon an den nächsten Schritten gearbeitet bevor ich überhaupt wusste, dass diese notwendig sind. Trotzdem, oder gerade deswegen, war es sehr schwer den Überblick zu behalten. Zum Glück wurden Informationen immer miteinander geteilt, so dass Schwierigkeiten vermieden und Fortschritte schnell in die Planung eingebracht werden konnten.

Als Projektleiter und Manager muss man sehr geduldig und organisiert sein, da man auch das restliche Team organisieren muss. Ich habe herausgefunden, dass ich nicht diese Person bin. Trotzdem hat alles funktioniert und war sehr aufschlussreich.

8.2 Oppolzer

Die Diplomarbeit ist ein exzellenter Weg, zeigen zu können, was man in den letzten Jahren gelernt hat. Neben dem Wissen, dass benötigt wird um den technischen Teil zu lösen, musste auch das Management durchgeführt werden. Sehr interessant an dem ganzen Projekt war, dass wir es von Anfang bis Ende, komplett selbst durchgeführt haben, ohne Hilfe von Lehrern etc. Die Professoren haben zwar manches überwacht und waren immer für Fragen offen, waren aber trotzdem nicht direkt an der Umsetzung beteiligt. So war es also unsere Aufgabe, einzuteilen was wir wie schnell schaffen, wieviel es kostet und wie wir das alles in dieser begrenzten Zeit schaffen. Zusätzlich sind natürlich, wie in einer echten Firma, auch zwischenmenschliche Dinge zu regeln. In dem Team darf es keinen Streit geben und es muss gute Kommunikation herrschen. Dieses Projekt neben dem alltäglichen Schulstress und der Maturavorbereitung unterzubringen ist also keine leichte Aufgabe.

Als Teammitglied ist es meine Aufgabe, dem Projektleiter zu beschreiben, wie schnell ich meinen Teil umsetzen kann und was ich dafür benötige. Die Kosten waren schwer zu schätzen, da ich anfangs nicht genau wusste, was tatsächlich notwendig ist, um diesen Roboter zu erstellen. Durch wiederholte Diskussionen mit den Teammitgliedern konnte schlussendlich ein Managementplan erstellt werden.

Den Roboter zu erstellen hat sehr viel Spaß gemacht. Es war sehr cool alles nach eigenen Vorstellungen zu gestalten. Von der Auswahl des Materials, über die Berechnungen, bis

zum Design. Obwohl es zwischendurch natürlich auch oft schwer war, da Probleme zu lösen waren, die längere Zeit beansprucht haben, war es im Vergleich zum Unterricht eine ganz andere Erfahrung. Im Rahmen des KU-Unterrichts wurden schon einige Aufgaben gelöst, welche ähnlich waren. Allerdings ist es bei diesem Projekt so, dass man tatsächlich etwas herstellt, was auch produziert werden wird. Die Auslegung und das Design kann nach eigenen Wünschen kreiert werden und man kann langsam beobachten, wie der Roboter immer mehr Form annimmt. Bis zu dem Zeitpunkt wo man ihn tatsächlich in den Händen hält. Diese Erfahrung und dieser Stolz können einfach nicht mit dem Unterricht verglichen werden.

Im Großen und Ganzen kann ich sagen, dass ich im Laufe dieses Projekts einiges gelernt habe. Da ich alles von Beginn an ganz gut über die Bühne gebracht habe, hatte ich keinen Stress und konnte für die Tests und Schularbeit lernen, ohne dass ich Probleme bekommen habe. Außerdem hat es mir geholfen zu verstehen, wie es später in einem Job aussehen könnte welche Probleme entstehen, was alles „passieren“ kann, welche Lösungsansätze gewählt bzw. umgesetzt werden und wie man mit den verschiedensten Dingen umgehen kann.

8.3 Oswald

Die Diplomarbeit ist eine hervorragende Gelegenheit das über die letzten 4,5 Jahre angeeignete Wissen zu testen und einzusetzen. Von der Planung und dem Management über die Mechanik, die Elektrik und die Software bat jeder Teil seine eigenen Herausforderungen, was die Arbeit aber immer interessant hielt. Genauso interessant war der Austausch mit anderen Maturagruppen, zu sehen was sie für Fehler und Erfolge hatten, Wissen auszutauschen, und ab und zu sich gegenseitig aufzuziehen.

Das Ausarbeiten des Schaltplanes war definitiv der anspruchsvollste Teil meiner Aufgaben, einfach da viele Teile von einander abhängen und damit nacheinander dimensioniert werden müssen. Genauso kann schon ein kleiner Fehler Auslöser für große Probleme sein, was oftmaliges Kontrollieren praktisch zur Pflicht macht. Das Layouten der Platine war hingegen nicht so schlimm, einfach weil auf unserer Platine mehr als genug Platz für alle Bauteile und Leiterbahnen vorhanden ist. Die größte Herausforderung hier war die Leiterbahnen auf einem zumindest halbwegs effizienten

Weg zu verlegen, sowohl um potenzielle Störungen auszuschließen, als auch aus ästhetischen Gründen.

Das Bestücken der Platine war dank der Vorhandenen Ausrüstung bei Theobroma Systems ein Kinderspiel, lediglich das zusammensuchen der einiger Bauteile dauerte etwas länger. Aber was wäre ein Elektronik-Labor ohne Chaos. Die darauffolgende Testphase verlief glücklicherweise relativ Fehlerfrei, einzig der 6V Spannungsregler war ein echter Fehler im Schaltplan, und auch das war nur ein Detail. Alle anderen Fehler lagen entweder bei der Software, oder bei einer Kombination aus ungünstigen Zuständen.

Die Servo-Motoren und das Problem mit durchbrennenden Transistoren war bei weitem das kritischste und mühsamste Problem, vor allem weil nicht viel dagegen unternommen werden kann, außer immer einen Ersatz-Motor auf Lager zu haben.

Am meisten gelernt habe ich bei der Erstellung des Schaltplans und bei den ersten Tests. Das Wissen, wie man Bauteile effizient aus einem teilweise fast endlosen Angebot sucht, in einem Schaltplan einbaut und dann auch nutzt wird sich in Zukunft sicher noch als hilfreich erweisen.

8.4 Federanko

Das Projekt half dabei zu sehen wie ein geplantes technisches Gerät umgesetzt wird. Im Konstruktions-Unterricht wurden oft solche Geräte dimensioniert und geplant, jedoch nicht hergestellt. Demnach bot das Maturaprojekt eine neue Herausforderung. Eine dieser Herausforderungen war es, mit den schlecht gekauften Servos vorlieb zu nehmen. Diese Servos waren leider von einer schlechten Qualität und brannten schon bei 60 Prozentiger Auslastung durch, was für die Anwendung im Roboter total ungeeignet war. Es ist wichtig Kernbauteile nicht nur auf ihre Funktionalität, sondern auch auf ihre Widerstandskraft zu testen. Oder zumindest Kundenbewertungen zu Rate zu ziehen.

Die Gestaltung der Software war einer der einfachsten Teile der Diplomarbeit, einzig die Kommunikation über die diversen Schnittstellen war etwas komplexer, wobei vor allem Linux selbst dies wesentlich erschwerte. Dies ist darauf zurückzuführen, dass der benutzte Linux-Kernel keine Module unterstützte und somit die meisten Lösungswege nicht anwendbar waren.

9 Begleitprotokoll

DATUM	THEMA
15.09.2016	Planung
29.09.2016	Statusupdate
13.10.2016	Statusupdate
27.10.2016	Statusupdate
10.11.2016	Verzögerung Elektronik
24.11.2016	Changerequest
22.12.2016	Präsentation Nachbesprechung
19.01.2016	Statusupdate
16.02.2015	3D Druck Funktionsweise
02.03.2014	Statusupdate
16.03.2013	Präsentationsbesprechung

10 Anhang

10.1 Literaturverzeichnis

Android Time Recording. (02. April 2017). *Android Time Recording*. Von https://sites.google.com/site/androidtimerecording/home abgerufen

Conrad. (02. April 2017). *Conrad*. Von https://www.conrad.at/ abgerufen

ConvertUnits.com. (04. April 2017). *Convert Units*. Von http://www.convertunits.com/from/N-m/to/kg-cm abgerufen

Evo-Tech. (04. April 2017). *Evo-Lizer*. Von http://evo-tech.eu/de/evo-lizer abgerufen

Google. (02. April 2017). *Allo*. Von https://allo.google.com abgerufen

GrabCAD. (04. April 2017). *Grabcad.com*. Von https://grabcad.com/ abgerufen

HiTEC. (2015). Servos 12-17mm. *Minikatalog 2015*, S. 7.

HiTEC. (04. April 2017). *HiTEC rcd*. Von http://hitecrcd.com/products/servos/micro-and-mini-servos/analog-micro-and-mini-servos/hs-225bb-mighty-mini-servo/product abgerufen

Jekl, K., & Zillek, B. (2005). *Spiderbot.*

Microsoft. (04. April 2017). *Microsoft Excel*. Von https://products.office.com/de/excel abgerufen

Microsoft. (02. April 2017). *Microsoft Office*. Von https://www.office.com/ abgerufen

Microsoft. (02. April 2017). *Microsoft Word*. Von https://products.office.com/de/word abgerufen

Microsoft. (02. April 2017). *Onedrive*. Von https://onedrive.live.com abgerufen

Modellbau Kirchert. (02. April 2017). *Modellbau Kirchert*. Von http://www.kirchert.com/modellbau/ abgerufen

Multi-CB Leiterplatten. (04. April 2017). *Multi-Circuit-Board*. Von https://www.multi-circuit-boards.eu/leiterplatten-design-hilfe/oberflaeche/leiterbahn-strombelastbarkeit.html abgerufen

Pöppe, C. (Februar 2016). Verletzte Roboter lernen wieder laufen. *Spektrum der Wissenschaft*.

PTC. (04. April 2017). *Creo Parametric*. Von http://www.ptc-de.com/cad/creo/parametric abgerufen

PTC. (04. April 2017). *MathCAD Prime 3.1*. Von http://www.ptc-de.com/software-fuer-konstruktionsberechnungen/mathcad abgerufen

Python Software Foundation. (31. März 2017). *Python*. Von https://docs.python.org/3/ abgerufen

Repetier. (04. April 2017). *Repetier-Host*. Von https://www.repetier.com/ abgerufen

Slic3r. (04. April 2017). *Slic3r*. Von http://slic3r.org/ abgerufen

Theobroma Systems. (04. April 2017). *Theobroma Git*. Von https://git.theobroma-systems.com/ abgerufen

Theobroma Systems. (02. April 2017). *Theobroma-Systems Home*. Von https://www.theobroma-systems.com/home abgerufen

Theobroma Systems. (04. April 2017). *Theombroma Resources*. Von https://www.theobroma-systems.com/a31-uq7/resources abgerufen

Traceparts. (04. April 2017). *Traceparts.net*. Von http://www.tracepartsonline.net/ abgerufen

Ubuntu. (05. April 2017). *Ubuntu Forums*. Von https://ubuntuforums.org/ abgerufen

wallyk. (04. August 2011). *Stackoverflow.* Von http://stackoverflow.com/questions/6947413/how-to-open-read-and-write-from-serial-port-in-c abgerufen

Whatsapp. (02. April 2017). *Whatsapp*. Von https://www.whatsapp.com/ abgerufen

Wikimedia. (06. Februar 2017). *Wiki Archlinux*. Von https://wiki.archlinux.org/index.php/NetworkManager abgerufen

Wikipedia. (04. April 2017). *ABS*. Von https://de.wikipedia.org/wiki/Acrylnitril-Butadien-Styrol-Copolymer abgerufen

Wittel, H., Muhs, D., Jannasch, D., & Voßiek, J. (2011). *Roloff/Matek Maschinenelemente.* Vieweg+Teubner.

Wittel, H., Muhs, D., Jannasch, D., & Voßiek, J. (2011). *Roloff/Matek Maschinenelemente Tabellenbuch.* Vieweg+Teubner.

10.2 Zeitaufstellung

10.2.1 Mießkes

DATUM	DAUER	AKTIVITÄT
30.05.2016	1,50	Teamfindung
31.05.2016	2,50	OSP PSP
01.06.2016	3,50	Ansuchen Schreiben
07.09.2016	0,90	Antrag schreiben
08.09.2016	1,60	Antrag schreiben
13.09.2016	7,20	Gespräch Kirchert
14.09.2016	2,10	Antrag Abgabe
21.09.2016	0,80	Sponsoren anfragen
28.09.2016	0,70	Sponsoren anfragen
03.10.2016	0,60	Theobroma
11.10.2016	3,50	Website
18.10.2016	7,10	Neuronales Netzwerk lernen
25.10.2016	7,10	diverse Recherchen
08.11.2016	7,20	Vorbereitung Toft
11.11.2016	5,30	TOFT
12.11.2016	3,60	TOFT
21.11.2016	0,50	Sponsoren anfragen
22.11.2016	7,10	Change Request
29.11.2016	7,10	Jugend Innovativ und andere Wettbewerbe teinehmen
01.12.2016	4,30	Servos Holen
06.12.2016	3,50	Entwickeln Testplatine
11.12.2016	2,00	Website
13.12.2016	8,80	Entwickeln Testplatine + herstellen
14.12.2016	1,00	Tests Servo
20.12.2016	4,00	Website
24.01.2017	9,50	Vorbereitung Toft
25.01.2017	1,00	Vorbereitung Toft
27.01.2017	5,30	TOFT
28.01.2017	3,60	TOFT
31.01.2017	3,50	Tests servo
01.02.2017	3,70	Gespräch mit Journalistin
13.02.2017	0,90	Website
14.02.2017	3,20	diverse Tests
16.02.2017	0,90	Website
21.02.2017	3,80	Jugend Innovativ abgabe schreiben
28.02.2017	6,70	Platinen Design
07.03.2017	8,60	Bewegungsmuster
08.03.2017	1,70	Gesamt test
09.03.2017	1,50	Gesamt test
14.03.2017	8,40	Fehlerbehebung / Bewegungsmuster
20.03.2017	1,80	Bewegungsmuster

21.03.2017	9,50	Präsentation Vorbereiten
22.03.2017	2,00	Präsentation Vorbereiten
27.03.2017	8,00	Präsentation Vorbereiten
28.03.2017	6,00	Präsentation
29.03.2017	4,50	Dokumentation zusammen fügen
30.03.2017	2,60	Dokumentation zusammen fügen
01.04.2017	6,00	Dokumentation zusammen fügen
02.04.2017	8,00	Dokumentation zusammen fügen
03.04.2017	6,40	Dokumentation zusammen fügen / Abnahmeprotokoll erstellen
04.04.2017	1,40	Abnahme

10.2.2 Oppolzer

DATUM	DAUER	AKTIVITÄT
13.09.2016	15,00	Handskizze, Sponsorgespräch
14.09.2016	2,00	Website
21.09.2016	1,00	Website
27.09.2016	4,00	Handskizze
28.09.2016	2,00	Website
03.10.2016	1,50	Website
04.10.2016	9,00	Handskizze, Berechnung
11.10.2016	8,00	Berechnung
12.10.2016	1,00	Dokumentation
18.10.2016	8,00	Berechnung
20.10.2016	14,00	Berechnung, Sponsorgespräch (Servokauf)
24.10.2016	2,00	Dokumentation
25.10.2016	8,00	3D-Modellierung
08.11.2016	8,00	3D-Modellierung
09.11.2016	4,00	Skizzenänderung
11.11.2016	5,50	Tag der offenen Tür
12.11.2016	4,00	Tag der offenen Tür
21.11.2016	3,00	3D-Modellierung
22.11.2016	8,00	Berechnung/3D-Modellierung
26.11.2016	5,00	Dokumentation
27.11.2016	5,00	Dokumentation
29.11.2016	8,00	3D-Modellierung
01.12.2016	4,00	3D-Modellierung
04.12.2016	5,00	Dokumentation
05.12.2016	4,00	Dokumentation
06.12.2016	4,00	3D-Modellierung
20.12.2016	4,00	Planung
28.12.2016	6,00	Dokumentation
10.01.2017	5,00	3D-Druck Modellierung
21.01.2017	1,00	Dokumentation
24.01.2017	8,00	3D-Druck Test
25.01.2017	3,00	Berechnung

27.01.2017	5,50	Tag der offenen Tür
28.01.2017	4,00	Tag der offenen Tür
01.02.2017	4,00	Tests
18.02.2017	2,00	Dokumentation
19.02.2017	5,00	Dokumentation
21.02.2017	5,00	Tests
28.02.2017	7,00	3D-Modellierung, Tests
02.03.2017	2,00	3D-Druck Modellierung
07.03.2017	5,00	Tests
14.03.2017	8,00	Dokumentation
16.03.2017	2,00	Dokumentation
21.03.2017	8,00	Tests, 3D-Druck Modellierung, 3D-Druck
23.03.2017	2,00	Dokumentation
28.03.2017	4,00	Präsentation
01.04.2017	4,00	Dokumentation
03.04.2017	3,00	Dokumentation
04.04.2017	10,00	Abnahme, Dokumentation

10.2.3 Oswald

DATUM	DAUER	AKTIVITÄT
27.09.2016	2,00	
28.09.2016	1,00	
03.10.2016	3,00	Theobroma
08.11.2016	8,00	Vorbereitung TOFT
09.11.2016	3,00	
11.11.2016	6,00	TOFT
12.11.2016	4,00	TOFT
18.11.2016	4,50	Theobroma Erste Besprechung
22.11.2016	8,00	
23.11.2016	1,00	
29.11.2016	7,00	Jugend Innovativ Anmeldung
30.11.2016	1,00	Schaltplan
06.12.2016	3,50	Testplatine
07.12.2016	2,00	Schaltplan
13.12.2016	5,00	Testplatine
14.12.2016	2,00	Schaltplan
17.01.2017	6,00	Schalrplan
18.01.2017	1,50	Schaltplan
24.01.2017	7,50	Schaltplan
25.01.2017	2,00	SchaltpkanSchaltplan
27.01.2017	5,50	TOFT
28.01.2017	4,00	TOFT
31.01.2017	3,50	PCB
01.02.2017	4,00	Gespräch mit Journalistin
02.02.2017	5,00	Theobroma Beantworten genereller Fragen

07.02.2017	5,00	PCB
08.02.2017	2,00	PCB
11.02.2017	4,00	PCB
12.02.2017	5,00	PCB
14.02.2017	7,00	PCB
15.02.2017	1,50	PCB
16.02.2017	4,00	PCB
17.02.2017	7,00	Theobroma Schaltplan + Platine
21.02.2017	9,00	
22.02.2017	1,00	
28.02.2017	7,00	
01.03.2017	1,50	
06.03.2017	8,00	Theobroma Bestücken der Platine
07.03.2017	9,00	
08.03.2017	2,00	
14.03.2017	7,00	
15.03.2017	2,00	
21.03.2017	9,00	
22.03.2017	1,50	
28.03.2017	4,00	Maturapräsentation

10.2.4 Federanko

DATUM	DAUER	AKTIVITÄT
26.09.2016	2,50	Programmierung der Firmware in C
28.09.2016	2,00	Testen der Seriellen Schnittstelle
03.10.2016	1,00	Projektmeeting
04.10.2016	3,60	Implementieren der Seriellen Schnittstelle
05.10.2016	1,00	Firmware Fertiggestellt (String basierte Steuerung)
05.10.2016	2,00	Propgrammierung des Main-Controllers in Python
07.10.2016	1,00	Entwicklung des Walkfile Dateiformats
11.10.2016	7,00	Schreiben des Persers der Walkfiles in Python
11.10.2016	1,00	Kompilieren des OS
12.10.2016	4,00	Kompilieren und Aufsetzen des OS
18.10.2016	7,00	Testplatine für PWM entwickeln und Bestücken
25.10.2016	8,00	Inbetriebnahme des Development Boards
03.11.2016	4,00	Implementieren des Kommunikationsservers in Python
08.11.2016	10,00	Aufsetzen des rootfs und installieren benötigter Software
09.11.2016	4,00	Optimieren des Walkfile Parsers (Walkie-Talkie)
16.11.2016	8,50	Schreiben der UART Library in C und testen der Kommunikation
18.11.2016	6,00	Besprechung und Debugging des Kernels
19.11.2016	8,00	Neukompilieren d. Kernels, aufsetzen des WLAN
22.11.2016	10,50	Debugging der Seriellen Schnittstelle und Integrierung der Library in Python
23.11.2016	4,00	Bugfixing im Walkie-Talkie

24.11.2016	7,00	Dokumentation mittels Doxygen
29.11.2016	9,50	Testen des Walkie-Talkies und der Programmexekution
12.12.2016	3,50	Präsentation
13.12.2016	5,50	Schreiben der Fernbedienung
14.12.2016	7,00	Schreiben der Fernbedienung
20.12.2016	9,00	Dokumentation der Fernbedienung und des Main-Controllers
10.01.2017	5,00	Entwicklungsscripts zum Konvertieren von Walkfiles
24.01.2017	9,00	Test-Walkfiles und Debugging der WLAN-Verbindung
25.01.2017	6,00	Configuration des OS; Lizenz und Readme hinzugefügt
26.01.2017	4,00	cairocon startup-script geschrieben
27.01.2017	5,00	Neuschreiben des Servers (server2.py)
28.01.2017	4,00	Neuschreiben des Servers und Automatische IP erkennung; Alias für fwselect hinzugefügt
31.01.2017	6,00	Implementieren des Port broadcastings; Der Server kann nun automatisch erkannt werden
01.02.2017	3,00	Fixen eines Bugs, welcher die Übertragenen Nachrichten zerstörte; Broadcasting impl (Automatische Servererkennung)
21.02.2017	4,00	Port von Python 2.0 auf Python 3.x
07.03.2017	9,30	Überarbeitung der Infrastruktur; Neukompilieren des Kernels (ohne dyn. Module)
10.03.2017	2,00	Wechsel von UART-Lib auf Datei
13.03.2017	8,00	Manuelle Ansteuerung von Servos möglich (Remote)
14.03.2017	4,00	Manuelle Ansteuerung von Servos möglich (Main-Controller); Ersten 2 fertigen Walkfiles geschrieben; Bugfixes
20.03.2017	3,00	Implementieren einer Step-by-Step Ausführungsmethode von Walkfiles
21.03.2017	6,00	Übersetzen der Bewegungsmuster in Walkfiles (+ Batch Converter Script)
29.03.2017	2,00	Fixen eines Bugs beim Laden von Funktionen zur definition von Motorbewegungen im Walkie-Talkie