

Bachelorprojekt: Nova Spot Micro

von Tammo Werner, Valentino Gaudeck, Neele Kohle
Betreuer: Prof. Dr. rer. nat. habil. Floris Ernst
Universität zu Lübeck



Inhaltsverzeichnis

Inhaltsverzeichnis	1
Einführung und Hintergründe	2
Vorhandene Dokumentation	3
Organisation und Ablauf	3
3D-Druck und Aufbau	5
Anmerkungen zu Druckeinstellungen	5
Schwierigkeit: Beine zusammenbauen	6
Elektronik	8
MOSFET Problem	11
Software	13
Bedienung	14
Zukünftige Veränderungen und Verbesserungen	15
Überarbeitung der Frame-Teile im Bauchbereich	15
Servos aufrüsten	15
Remote Controller	16
Änderung des Arduino und Teensy zu Raspberry Pi	16
LiDAR Erweiterung	16
ROS-Anwendung	16
Greifarm	16
Fazit	17

Einführung und Hintergründe

Das Ziel von diesem Bachelor-Projekt ist, den vierbeinigen Laufroboter "Spot", ehemals Spot Mini, von Boston Dynamics vereinfacht nachzubauen und die Fähigkeit zum Laufen zu implementieren.

Der Laufroboter Spot wurde erstmal 2015 vorgestellt, das zusätzliche Modell Spot Mini folgte 2016, wobei es heute dem Standardmodell entspricht. Die ursprüngliche Intention bei der Entwicklung von Spot war es, einen agilen Roboter, der diverse Terrains im In- und Outdoorbereich bewältigen kann, zu entwickeln. Die aktuelle Version hat eine Größe von 74cm und wiegt 25kg, wobei dieser auch noch 14kg Lastgewicht tragen kann. Spot ist mittlerweile kommerziell erhältlich, mit verschiedenen Zusätzen modifizierbar und findet in diversen Branchen Einsatzmöglichkeiten, was seiner Vielseitigkeit zugrunde liegt.

Die Motivation für dieses Projekt stammt aus dem Fach "Mobile Roboter", bei dem verschiedene Aspekte der mobilen Robotik behandelt werden und entsprechende Roboter an Beispielen vorgeführt werden. Das Open-Source-Projekt "Nova Spot Micro" von Chris Locke wurde Anfang 2021 offiziell gestartet und ist eine Zusammenführung vieler verschiedener Ansätze von nicht-kommerziellen Nachbauten, welche kontinuierlich weiterentwickelt und aktualisiert werden. Im Laufe des Open-Source-Projekts gab es diverse Änderungen und Weiterentwicklungen an den gedruckten Teilen, der Elektronik und der Software.

Die Fähigkeiten des Nachbaus sind nicht mit dem Original vergleichbar. Dieser bietet aber eine gute Grundlage und dieser hat Potential für die Zukunft.

Links: Original Spot von Boston Dynamics

Rechts: Nova Spot Micro inklusive selbstdesigntem Controller



Vorhandene Dokumentation

Bei diesem Projekt ist ein Teil der Dokumentation schon vorhanden auf der Webseite¹, dem Git Repository², verschiedenen YouTube-Videos³ und dem Discord-Server⁴.

Auf der zugehörigen Website existiert eine grobe Dokumentation der meisten Teilespekte des Projekts, wozu unter anderem eine ausführliche Liste der Komponenten und eine 3D-Darstellung des Roboters gehören.

Bei den Komponenten kann mittels einer Verlinkung das Produkt direkt auf Amazon aufgerufen werden, wobei zu beachten ist, dass es sich hierbei um das amerikanische Amazon handelt. Um Kosten und Zeit zu sparen wurden, für dieses Projekt, äquivalente Produkte von der deutschen Website ausgewählt.

Die Platine für die Version 5.2 kann über eine weitere externe Webseite bestellt werden, welche ebenfalls aus den USA liefert. Da hier keine Informationen über Kosten und die Dauer des Versands vorlagen, wurde die Version 5.1, für die im Git-Repository entsprechende Pläne vorhanden sind, gewählt, wodurch das Board in Deutschland hergestellt werden konnten.

Schaltpläne, Code und STL-Dateien für den 3D-Druck sind ebenfalls im Git-Repository gespeichert und können für das Projekt genutzt werden. Die Schaltpläne sind aber nicht immer korrekt und müssen dementsprechend mit Vorsicht genutzt werden.

Auf dem YouTube-Kanal werden regelmäßig Videos hochgeladen, die entweder den Aufbau des Roboters anleiten oder Aktualisierungen erläutern. Diese sind zwar oft nützlich, bieten aber keine genaue Übersicht, mit welcher Version in den Videos gearbeitet wird. Leider wurde die Dokumentation des Projektes nicht von Anfang an mitbegleitet, sodass die ersten Videos den Roboter in einem fortgeschritten Zustand zeigen.

Der Discord-Server dient eher als Diskussionsforum. Sollten Probleme auftreten kann hier entweder der Entwickler selbst angeschrieben oder die Community gefragt werden.

Außerdem werden teils individuelle Veränderungen, Ideen oder Tipps mitgeteilt, die man aus der offiziellen Dokumentation nicht entnehmen kann.

Im Großen und Ganzen bietet die bereits vorhandene Dokumentation eine gute Basis für dieses Projekt, ist aber an vielen Stellen lückenhaft und unübersichtlich, was auch bei unserem Nachbau Probleme nach sich gezogen hat. Daher empfehlen wir stark, sich bereits von vorneherein gut mit dieser auseinanderzusetzen. Jene werden im Laufe des Berichts auch erläutert und mögliche Lösungen dazu genannt.

Organisation und Ablauf

Zu Beginn des Projekts wurde zunächst eine Übersicht der benötigten elektronischen Teile erstellt, mit Inhalten bezüglich benötigter Menge, Kosten und ob das Originalprodukt lieferbar ist oder auf eine Alternative umgestiegen werden musste. Einige der benötigten Bauteile befanden sich auch bereits im Institut.

Des Weiteren wurde eine weitere Liste angefertigt, die Überblick über die zu druckenden Teile geben sollte. Der Bearbeitungsstand dieser Liste wurde im Laufe des Projekts jedoch nicht immer aktuell gehalten, was teils für Verwirrung und vereinzelt auch zu mehrfachem

¹ <https://novaspotmicro.com/>

² <https://github.com/cguweb-com/Arduino-Projects/tree/main/Nova-SM3>

³ <https://www.youtube.com/channel/UCiB6E0DLSXcLUyziYIWdbXA>

⁴ <https://discord.com/invite/Fj8NsHED>

Druck einiger Teile führte. Dies stellt jedoch kein großes Problem dar, da sie für einen zweiten Roboter verwendet werden können.

Anschließend wurden die entsprechenden Bestellungen getätigt und die ersten Drucke aufgesetzt. Nachdem die zentralen "Frame" Bauteile gedruckt wurden, wurde damit begonnen, diese zu bearbeiten und, wenn möglich, direkt zusammenzusetzen. Dieser stetige Prozess wurde des Öfteren unterbrochen, um neue Drucke vorzubereiten und zu drucken.

Zwischenzeitliche Versuche bereits an der Elektronik zu arbeiten stellten sich als schwierig dar, weil die Grundstruktur dafür grundlegend fehlte. Dies führte dazu, dass erst die 3D-gedruckten Teile fertig bearbeitet und zusammengebaut werden mussten. Der Einbau der elektronischen Teile und die zugehörige Verkabelung konnten erst im Anschluss stattfinden.

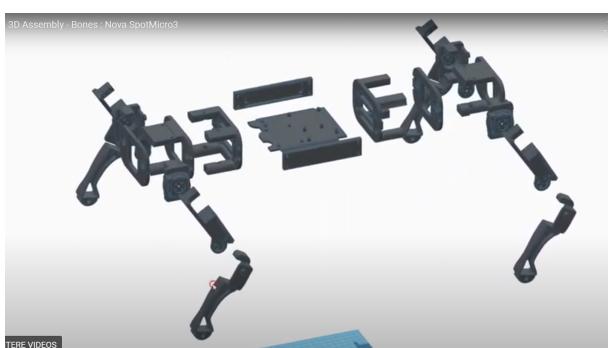
Die Kalibrierung der Servos und das implementieren, testen und umschreiben der Software hat als letztes stattgefunden.

Über den Verlauf des Projekts fanden regelmäßige Besprechungen mit dem Betreuer statt, bei denen der aktuelle Stand und Probleme besprochen wurden.

3D-Druck und Aufbau

Der Körper des Roboters besteht komplett aus 3D-gedruckten Teilen. Die meisten dieser bestehen aus Polylactide (PLA) und konnten an den 3D-Druckern der Uni angefertigt werden. Lediglich die Füße sind aus Polyurethane (TPU) und benötigten andere Drucker, welche von einem Mitarbeiter des Instituts privat gedruckt wurden.

Die Gesamt-Druckzeit beträgt insgesamt über 100 Stunden.



Die Teile werden in sogenannte Frame-Parts und Cover-Parts unterteilt. Frame-Parts (links) stellen hierbei das Grundgerüst dar, und soll dem Roboter die nötige Stabilität geben. Des Weiteren ist anzumerken, dass alleine mit den Frame-Parts schon ein Aufbauen der Elektronik nahezu vollständig möglich ist und diese im Torso komplett eingebaut werden kann. Nur für den Zusammenbau

der Beine sind Cover Teile notwendig, um ein funktionstüchtiges Gestell zusammenzubauen zu können.

Die Cover-Parts sind überwiegend Teile, die hauptsächlich die Elektronik abdecken und dementsprechend nicht maßgeblich für die Stabilität verantwortlich sind.

In der Originalversion von Chris Locke sind die Teile gut unterscheidbar, da die Cover-Parts im Allgemeinen mit gelben Filament gedruckt wurden und die Frame-Parts mit schwarzem. Elektronische Komponenten sind hierbei lediglich im Front- (Ultraschallsensoren), Top- (Infrarotsensoren) und Rear- (Anzeige und Buttons) Bereich verbaut. Durch die Cover-Parts der Beine führt ein Kabelkanal, um die Kabel der Servomotoren in den Kniegelenken zum Controller zu führen.

Zum Testen des Roboters empfiehlt es sich, den Standfuß, welcher auch als STL-Files im Git ist, zu drucken und zu verwenden. Dies hat den Vorteil, dass die Beine nicht den Boden berühren und man somit unbeschwert die Servo Motoren kalibrieren kann.

Anmerkungen zu Druckeinstellungen

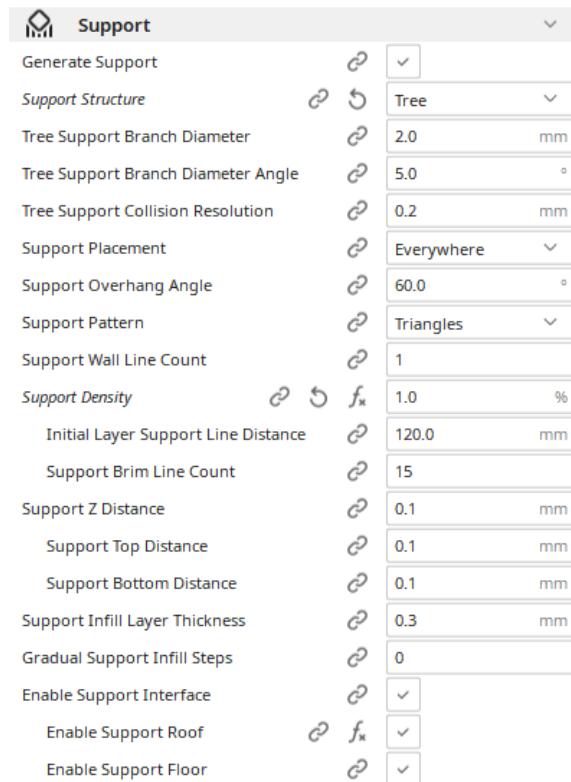
Zum Slicen der STL Files bei der Vorbereitung für den 3D-Druck wurde das Programm Ultimaker Cura benutzt. Dieses bietet die Möglichkeit, mit einer relativ einfachen Bedienoberfläche die einzelnen Teile vor dem Druck zu veranschaulichen und mehrere STL Files zu einem größeren Druck zu kombinieren. Hierbei ist zu beachten, dass die gewählten Druckeinstellungen maßgeblich für die Qualität und Nutzbarkeit der Teile entscheidend ist.

Es gab zwei besonders wichtige Änderungen, welche an den Standard-Druckeinstellungen während des Projekts vorgenommen wurden. Zum einen eine Verringerung des "Infill Density" Werts auf ca 15% zum Sparen von Druckzeit und Ressourcen. Sowie einige Veränderungen an den Einstellungen zur generierten Support Struktur bei überstehenden Bereichen der Teile. Der Wechsel zu einer baumartigen Struktur ist hilfreich, um das anschließende Bearbeiten der Teile deutlich zu erleichtern oder überhaupt erst zu

ermöglichen. Ohne diese Veränderung bildet die Support Struktur einen festen Block, der sich nur sehr schwer entfernen lässt. Auch hier kann die "Support Density" wieder deutlich reduziert werden.

Auf die korrekte Temperatureinstellung in der Software, sowie am Drucker ist zu achten, damit diese zu den empfohlenen Angaben des Filament-Typs passen.

Für exakte Werte aller Einstellungen lohnt es sich, diese zu testen und notfalls anzupassen. Im folgenden Bild sind die Support Einstellungen zu sehen, welche zuletzt in diesem Projekt verwendet wurden.



Selbst bei den letztlich gewählten Einstellungen gab es jedoch teils große Schwierigkeiten die Teile zu bearbeiten, sodass diese letztendlich genutzt werden konnten. Das Druckergebnis kann allerdings von der Qualität des gewählten 3D-Druckers beeinflusst werden.

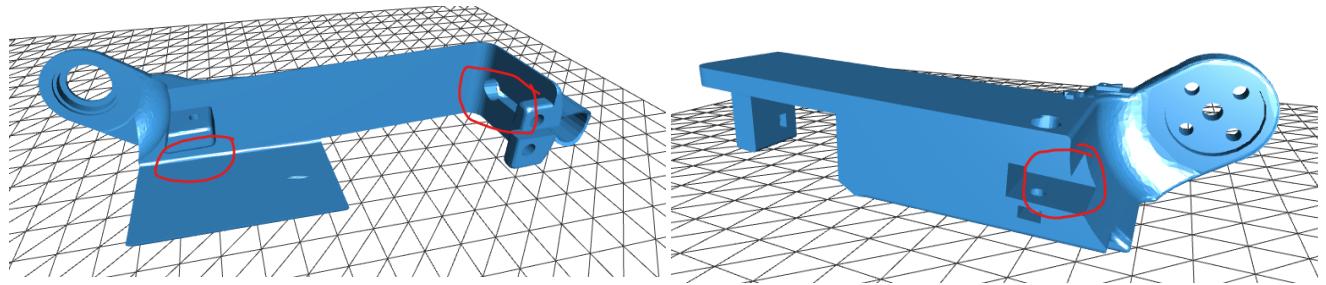
Schwierigkeit: Beine zusammenbauen

Einer der schwierigsten Parts beim Zusammenbau des Roboters, stellt das Zusammensetzen der Beine dar. Diese bestehen im Wesentlichen aus den Gelenken Coax, Femur und Tibia, worin die Servos verbaut sind.

Die gedruckten Beinteile benötigen mit am meisten Bearbeitung, da diese ineinander gefasst werden und dementsprechend möglichst millimetergenau abgestimmt werden sollten. Neben den Mulden für die Muttern, welche oft mit sehr festem Stützmaterial gefüllt waren, ist an zwei Bauteilen (Femur, MidArmCover) das Stützmaterial besonders fest.

Beim MidArmCover ist außerdem der Kabelkanal mit Druckmaterial gefüllt und muss vom inneren Support Material befreit werden. Hierbei bricht dieser auf der Außenseite schnell auf, dennoch kann dieses Teil noch genutzt werden, benötigt nur vorsichtigeren Umgang.

Die genannten Stellen sind auch auf den Bildern unten markiert (links Mid Arm Cover, rechts Femur).



Der Tibia Servo Motor, muss bevor der Coax Servo Motor eingebaut wird, mit einer Schraube durch das Servohorn des Coax-Drucks festgezogen werden.

Das Bein muss als Ganzes zusammengesetzt und anschließend eingebaut werden. Zuerst werden die Muttern mit den Servos sowie die Kugellager und Servo-Arm-Horns eingesetzt. Die Tibia, also das Unterbein, kann fertig zusammengeschraubt werden. Anschließend sollten MidArmCover und Femur zusammengesteckt werden, wobei das Kabel des Servos am Tibia bereits mit in den dafür vorgesehenen Zwischenraum eingesetzt werden muss und die Tibia selbst beim Zusammenstecken schon zwischen Kugellager und Servo-Arm sitzt. Beim Coax muss darauf geachtet werden, dass dieser erst am Bein befestigt wird bevor das gesamte Konstrukt zwischen dem Hauptframe Teilen eingesetzt wird.

Die eben beschrieben Prozedur wird auch im Video "Leg Assembly Tips" von Chris Locke etwas erläutert.⁵

⁵ <https://www.youtube.com/watch?v=AKOmoGPIIJ0>

Elektronik

Die Platine ist bei diesem Roboter von der Version 5.1, wobei andere Komponenten bereits von der Version 5.2 stammen. Um Kosten und zusätzlich auch Zeit zu sparen, wurden alte Pläne verwendet, die im Nachhinein noch an die aktuelle Version angepasst werden mussten. So musste unter anderem ein Stück der Platine nachträglich gebaut werden, wobei der Aufbau den Schaltplänen der Elektronik entnommen wurde. Im aktuellen Projekt ist dieses Teil jedoch nicht verbaut, da es nicht die korrekte Funktion erfüllte. Ursprünglich sollte es dem Benutzer ermöglichen, den Teensy nach dem Einschalten des Hauptschalters separat ein- und ausschalten zu können. Allerdings schaltete diese Platine den Teensy direkt aus und nicht wieder ein, weshalb ein Ausbau notwendig war. Diese fehlende Funktion ist kein erheblicher Nachteil beim aktuellen Roboter.

Die Stromversorgung des Roboters wird über eine Lipo-Batterie (11,1V und 2200mAh) gesichert und mittels zweier Umwandler auf die jeweils benötigten Spannungen angepasst. Hierbei war zudem ein weiterer Adapter nötig, um die Batterie und einen der Schalter zu verbinden.

Der eine Step Down Buck Converter wird auf die passende Spannung für die Servo Motoren eingestellt (6V) während der andere Converter zur Versorgung des Boards, Teensys, Arduinos und anderer Komponenten auf 5V eingestellt werden muss.

Da für den Roboter die Kabel selbst zugeschnitten werden müssen, sollte stets darauf geachtet werden, wo sich die einzelnen Komponenten befinden und final eingebaut sind, da auf Ober- und Unterseite der mittleren Platine sich verschiedene Objekte befinden. An vielen Stellen wurden Wago-Klemmen benutzt, um einerseits eine bessere Übersicht zu erhalten und andererseits den Innenraum flexibler für Veränderungen zu halten und dementsprechend auch Zeit zu sparen. Im Original sind an vielen Stellen Schraubklemmblöcke vorgesehen, die aber aufgrund von Platzmangel und Aufbau des Roboters oft kompliziert und umständlich sind.

Für die Sensorik werden zwei Ultraschallsensoren, drei Infrarotsensoren und ein Gyroskop genutzt.

Die zwei Ultraschallsensoren befinden sich in der Front und haben laut Hersteller eine Reichweite von 2cm - 300cm. Die Ausrichtung ist in Laufrichtung, wobei diese leicht nach innen gedreht sind, um einen komplexeren Bereich abzudecken.

Drei Infrarotsensoren befinden sich auf der Oberseite des Roboters, wobei einer vorne und zwei mittig, jeweils einer rechts und links sitzen. Durch 3D-gedruckte Abdeckungen wird der Sichtbereich eingeschränkt und lässt nur jeweils einen Winkel von ~120° zu, damit auf Bewegungen gezielter reagiert werden kann. Laut Hersteller haben die Sensoren eine Reichweite von bis zu 5 Metern, wobei nur bis zu 3,5 Metern empfohlen wird.

Das Gyroskop befindet sich im Bauchraum des Roboters auf dem Hauptboard und dient der Stabilität und dem Gleichgewicht des Roboters. Bei unserem Modell gibt es allerdings Schwierigkeiten dahingehend, dass das Gyroskop selbstständig Werte verändert über Zeit, obwohl keinerlei Veränderung der Roboterrotation stattgefunden hat. Dies führt zu fehlerhaften Bewegungen, weshalb es in unserem Code zurzeit deaktiviert ist.

Die Datenkabel der Sensoren werden an den vorgegebenen Steckplätzen auf der Platine eingesteckt und die Sensoren benötigen noch zusätzliche Kabel zur Stromversorgung.

In dem aktuellen Code ist die Sensorik noch nicht aktiviert, dementsprechend reagiert der Roboter aktuell nicht auf diese, wobei Infrarotsensoren funktionstüchtig wären. Die Ultraschallsensoren geben nicht interpretierbare Werte für den Teensy aus, wie später erläutert.

Für die Bewegung der Gelenke werden insgesamt 12 Servomotoren, 3 pro Bein, mit einem maximalen Drehmoment von 21,5 kg und einer maximalen Spannung von 6,8V verwendet. Die Spannung wurde bei diesem Projekt auf 6V reduziert, da der PWM Servo Motor Controller, also die Steuereinheit für alle 12 Servos, nur 6V maximal für die Servos verträgt. Dieser Wert kann eventuell auf die 6,8V erhöht werden, aber dann ist die Sicherheit des Boards nicht sichergestellt und es kann durchbrennen. Der Original Ersteller benutzt 6,8V, jedoch besitzt er einen leicht anderen Controller.

Die Motoren sollten im Idealfall vor dem Einbau kalibriert werden (z.B. mit einem "Probe-Bein"), was auch der Empfehlung des Entwicklers entspricht. Dies ist aber nicht zwanghaft notwendig und wurden in diesem Projekt erst nach dem festen Einbau der Beine vollzogen, was mit entsprechender Vorsicht geschehen muss. Da bei diesem Projekt einer der initial eingebauten Servos sowie der PWM Driver fehlerhaft waren, mussten diese im Nachhinein ausgetauscht werden. Da gerade der Aufbau der Beine sehr komplex ist, wie weiter oben erläutert, bedeutete dies einen nahezu komplettes auseinander- und wieder zusammenbauen. Hier hätte es sich angeboten, die Servos zu kalibrieren oder wenigstens auf die Funktionalität zu testen.

Gesteuert werden diese von einem Servo Controller mit 16 Pins mit folgenden Anschlüssen:

- 0 = Right Front Coax
- 1 = Right Front Femur
- 2 = Right Front Tibia
- 4 = Left Front Coax
- 5 = Left Front Femur
- 6 = Left Front Tibia
- 8 = Right Rear Coax
- 9 = Right Rear Femur
- 10 = Right Rear Tibia
- 12 = Left Rear Coax
- 13 = Left Rear Femur
- 14 = Left Rear Tibia

Hierbei zu beachten ist, dass die Pins auf dem Controller strukturell in vier Blöcke mit je vier Pins aufgeteilt wurden. Der Übersichtlichkeitshalber wurde jedem Bein einer dieser Blöcke zugewiesen, und der letzte übrig gebliebene Pin wurde freigelassen. Somit sind die Pins nicht von 0 bis 11 aufsteigend zugewiesen.

Bei der Kalibrierung der Servos müssen eine Home Position bestimmt werden, so wie eine Min- und Max Bewegung. Dafür muss jeder Servo zunächst einzeln getestet werden und im Anschluss als ganzes Bein. Die jeweils resultierenden Werte werden im Code zur Laufbewegung benötigt und müssen dementsprechend passend gewählt werden. Hierbei muss darauf geachtet werden, dass Vorder- und Hinterbeine volle Bewegungsfreiheit in ihrem Bewegungsradius haben. Dies beinhaltet auch eine Kollisionsvermeidung mit dem oben beschriebenen Standfuß zum Kalibrieren der Beine. Sollten diese Werte nicht korrekt eingestellt werden, erhöhen die Motoren ihren Stromverbrauch bei einer Blockade enorm, und dies könnte den Servo Driver sehr schnell durchbrennen lassen. Dies kommt daher,

dass die Motoren messen können, dass sie sich nicht mehr drehen können, sich jedoch weiter drehen sollten. Hierbei kann aber keine Unterscheidung gemacht werden, ob sie einfach nur eine schwerere Last bewegen müssen oder blockiert werden. Anschließend versuchen sie der vermeintlich schwere Last entgegenzuwirken, durch einer Erhöhung des Stromverbrauchs. Hier würde es sich eventuell anbieten zusätzlich zum Display der Spannungsmessung im hinteren Bereich des Roboters noch ein Display zur **Strommessung** einzubauen, um eventuelle Spitzen zu erkennen und den Roboter notfalls abschalten zu können.

Zur Übersicht hat der Entwickler eine Excel-Tabelle erstellt, in der die entsprechenden Werte eingetragen werden können. In der vorgegebenen Version sind die genannten Motor-Pins andere, als die im oberen Code, was auf leicht unterschiedliche Versionen andeutet.⁶ Des Weiteren wird die Kalibrierung der Beine auch in den Videos “Motors,PWM & Calibration”⁷ und “Recalibrating Motors”⁸ erläutert, wobei zu erwähnen ist, dass es sich hierbei noch um den Nova Spot Micro 2 handelt und dementsprechend der genutzte Code leichter Veränderungen aufweist.

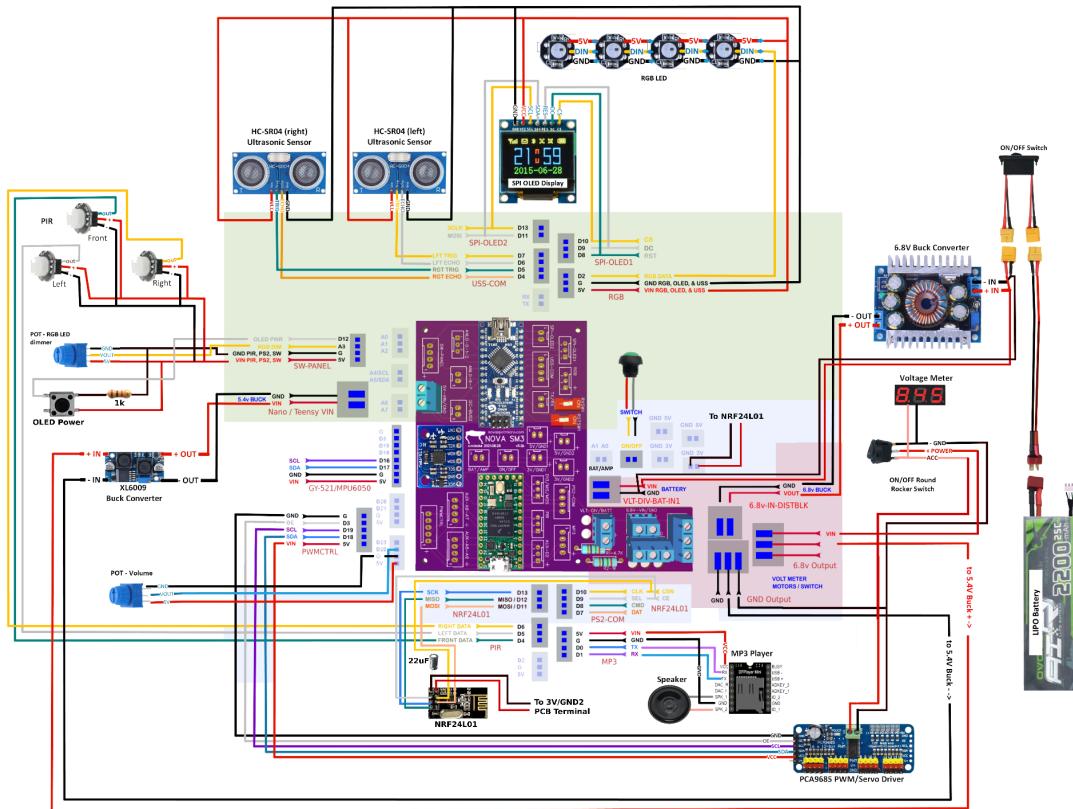
Beim Einbau der Komponenten sollte auf das Gravizentrum geachtet werden und, wenn nötig, versucht werden, Gewicht umzulagern. Sollte hierauf nicht geachtet werden, kann es passieren, dass durch das Übergewicht vorne oder hinten, der Körper zu weit abfällt und die Beine nicht korrekt arbeiten können.

⁶<https://github.com/cguweb-com/Arduino-Projects/blob/main/Nova-SM3/NovaSM3%20Motor%20Vars.xlsx>

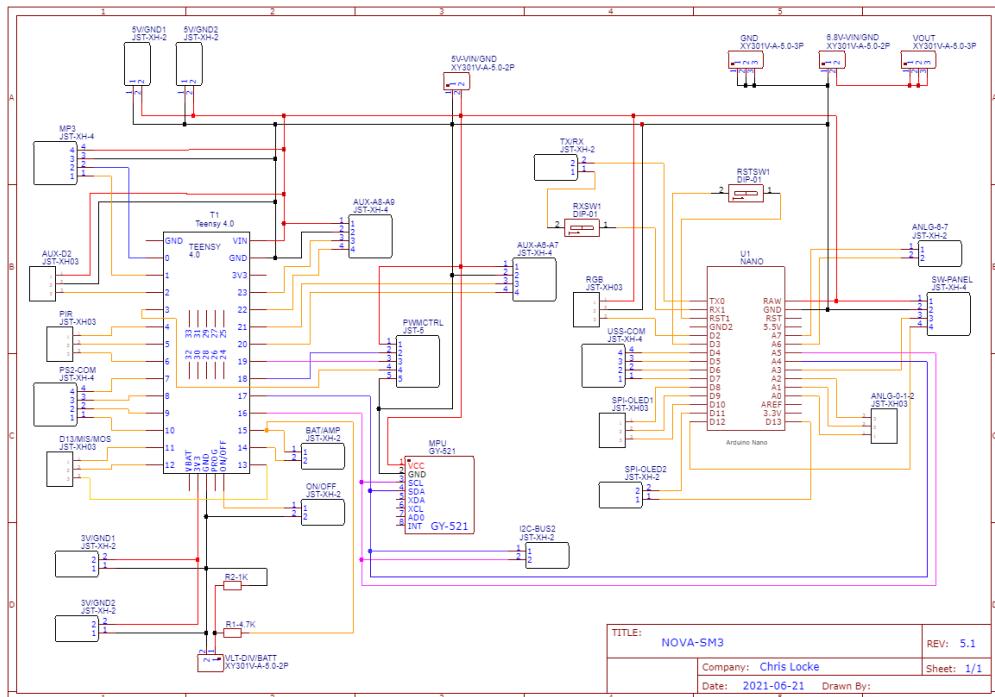
⁷ <https://www.youtube.com/watch?v=IGd8OIBIXnM&t=1035s>

⁸ <https://www.youtube.com/watch?v=PeF2PaJTmic>

Im Folgenden ist ein Schaltplan zu sehen, welcher auch im offiziellen Git Repository von Nova Spot Micro zu finden ist:



Im Folgenden ist eine Belegung der Pins auf dem Hauptboard in schematischer Darstellung: (auf dieser Darstellung befindet sich ein Fehler. Unten links wird die Verbindung der Widerstände gezeigt, wobei der 1k Ohm Widerstand eine falsche Verbindung aufweist.)



Hier zeigt sich deutlich, weshalb es schwierig wäre, die beiden Computer, wie später unter Software beschrieben, zu einem Computer zusammenzufassen.

MOSFET Problem

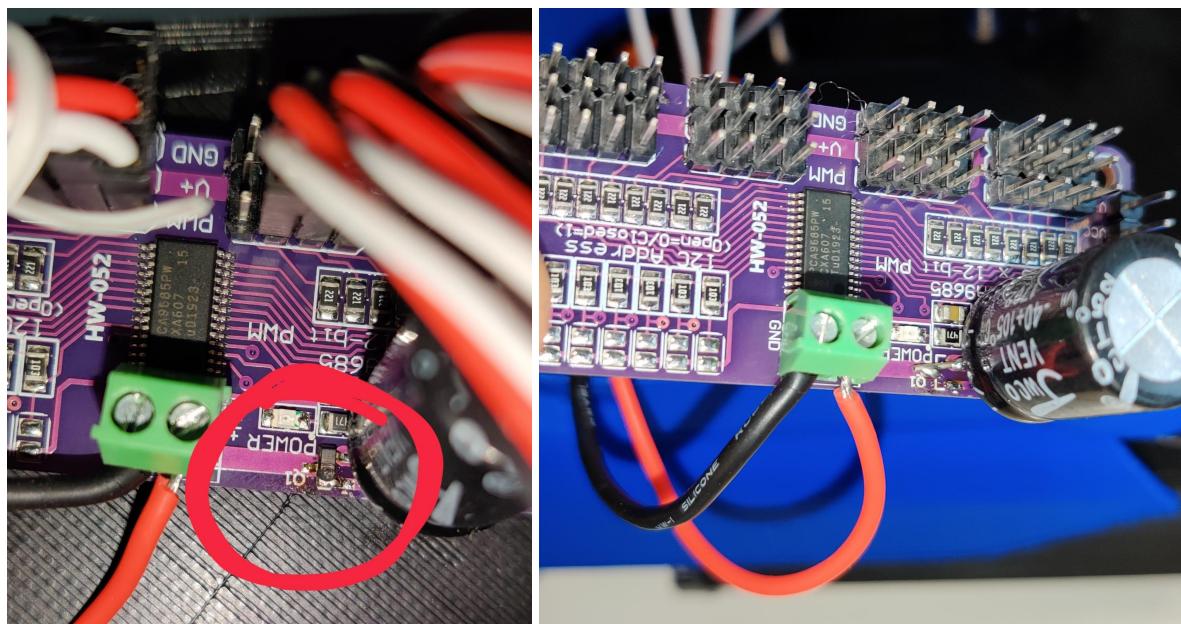
Nach dem erfolgreichen Zusammenbau konnte mit dem im Git zur Verfügung gestellten calibrate-Code fortgefahren werden. Nach dem Herunterladen und Einrichten konnten die Motoren eingestellt werden. Das einzelne Testen der Motoren war kein Problem, aber sobald mehr als 3 Motoren gleichzeitig kalibriert wurden, konnte ein unangenehmer und verbrannter Geruch wahrgenommen werden. Allerdings konnte keine genaue Stelle oder Bauteil ausgemacht werden, das den Geruch verursacht. Nach weiteren Tests und Versuchen der Lokalisierung des Problems ist der Servo Controller schließlich durchgebrannt. Daher war eine Vermutung eine falsche Verkabelung mit dem Servo Controller oder ein defekter Controller. Doch auch nach einem Austausch mit einem nagelneuen Controller und einer kompletten Überprüfung und Neuverkabelung aller Leitungen blieb das Problem bestehen.

Nach langer Recherche hat dieses YouTube⁹ Video Klarheit geschafft, wo jemand genau das aufgetretene Problem erklärt:

Die bestellten Servo Controller besitzen einen Verpolungsschutz in Form eines MOSFETs. Wenn die Spannungseingänge verkehrt herum angeschlossen sind, fließt keine Spannung über den Gate-Eingang des MOSFETS und dieser blockiert den Stromfluss. Das heißt, dass der gesamte Strom des Controllers an die Motoren aus Sicherheitsgründen durch diesen einen MOSFET geführt wird. Das Problem hierbei ist, dass jeder Servo je nach Last zwischen 1 - 2A an Strom benötigt und bei 16 möglichen Anschlusspins 20A nicht ungewöhnlich sind, die dann durch den Sicherheits-MOSFET fließen. Dieser verträgt aber nur bis 4A und geht somit bei der gleichzeitigen Benutzung von mehreren Servos kaputt. Die Lösung war aber ziemlich einfach:

Nach dem Sicherstellen, dass die Polarität korrekt ist, konnte der MOSFET ausgebaut und durch eine direkte Verbindung ersetzt werden, da diese die Funktion des Bauteils ersetzt. Nach kurzer Lötarbeit wurde dies erfolgreich umgesetzt.

Auf dem Bild links ist der durchgebrannte MOSFET klar zu erkennen, rechts sieht man den Ersatz in Form eines kleinen Metalldrahtes.



⁹ <https://www.youtube.com/watch?v=EqVhpilm3sw>

Software

Aufgrund von mangelnder Dokumentation stellte die Inbetriebnahme der Software eine recht anspruchsvolle Aufgabe dar. Es waren zwar vereinzelte Videos vorhanden, in denen die verschiedenen Codes erklärt wurden, allerdings gab es keine allumfassende Erklärung zu grundlegenden Problemen, wie z.B. der Einrichtung oder Installation der benötigten Komponenten und die Implementierung auf dem Roboter.

Nach dem Austesten verschiedener Ansätze zur Programmierung und der Implementierung der schon vorhandenen Codes zeigten sich, dass es am einfachsten ist, wenn das gesamte Repository aus dem Git heruntergeladen wird. Zusätzlich dazu mussten noch unnötige Dateien gelöscht werden und anschließend mit dem übrig gebliebenen Code gearbeitet werden.

Der calibrate-Code muss als Erstes auf den Teensy mithilfe des Teensyduino geladen werden und anschließend stellt man alle Motoren bezüglich der Home-, sowie Min- und Max Position ein.

Überraschend war, dass die Motoren sich beim Starten des Codes einmalig mit der maximalen Geschwindigkeit auf die Home Position bewegen, egal welchen Wert die Geschwindigkeitsvariable im Code annahm. Dies lässt sich dadurch erklären, dass die Motoren in ihre Startposition (Home Position) fahren wollen, was den Motoren durch den Servo Controller mitgeteilt wird. Dementsprechend kann dieses Verhalten nicht unterbunden werden und ist unabhängig vom Code. Daher ist beim ersten Einstellen enorme Vorsicht geboten, da es durchaus sein kann, dass die Motoren sich und den Roboter beschädigen.

Nach dem ausführlichen Testen und Einstellen der Motorwerte, konnte mit dem Code fortgefahren werden, welcher dann auf Roboter laufen wird. Zwar gab es einfache Möglichkeiten, die von uns nicht verbauten Bauteile zu deaktivieren, dennoch verbraucht dieser überflüssige Code viel Speicher und Rechenleistung. Die Erstellung des eigenen Codes, welcher genau auf unser Projekt zugeschnitten ist, durch die Abwandlung des vorhandenen Codes, lag damit nahe. Hier zeichnet sich deutlich ab, dass der Roboter in unserer Ausführung eigentlich nicht beide Computer brauchte, aber aufgrund der internen Verkabelung des Hauptboards es keine einfache Möglichkeit gibt, ihn auf einen einzelnen Computer umzubauen.

Die Löschungen des überflüssigen Codes ersparten für den Teensy ca. 1000 Zeilen und ist somit nur noch ca. 5200 Zeilen lang. Der Code für den Arduino Nano konnte um ca. 1200 Zeilen auf etwas über 150 Zeilen reduziert werden. Der Grund für diese proportional große Ersparnis beim Arduino ist, dass dieser im alten Code für die Ultraschall Sensoren, das OLED-Display, die RGB-Streifen und den MP3-Player zuständig ist. Nachdem der Code für von uns nicht verbauten Bauteile gelöscht wurden, ist er nur noch für die Ultraschall Sensoren zuständig.

Zwei selbst geschriebene Test-Codes, um die Funktionalität der Ultraschall- und Infrarotsensoren zu testen, folgte kurz darauf. Erstere haben unlogische Werte beim debuggen wiedergegeben und daher war es notwendig diese auf Korrektheit zu überprüfen. In der Verbindung mit dem Teensy waren diese Werte ausschließlich zwischen 0 bis 10 und 1100 bis 1150 und nichts dazwischen, was aus verschiedenen Gründen nicht sein kann. Der offensichtlichste ist, dass die Sensoren nur bis 3m, also dem Wert 300, messen können. Durch den Testcode, welcher ausschließlich auf dem Arduino lief und nicht mit dem Teensy verbunden war, konnten wir eine korrekte Funktionsweise der Ultraschallsensoren

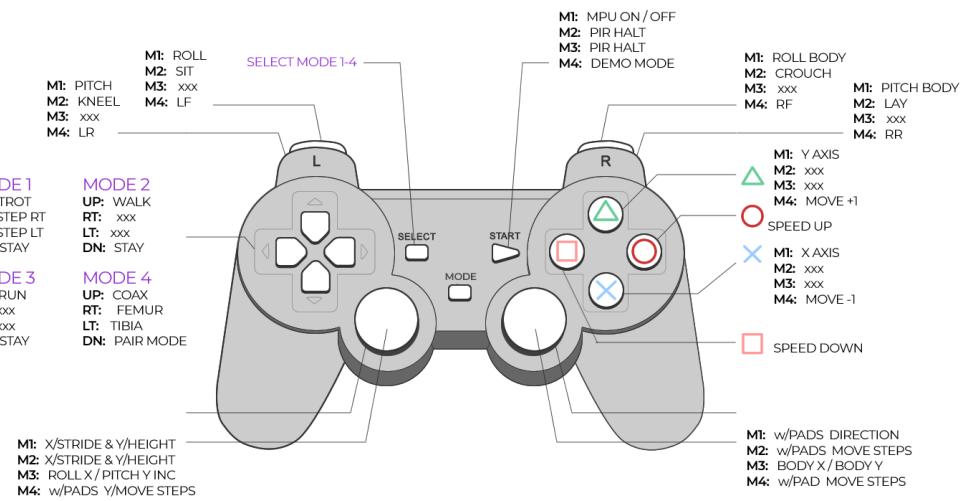
feststellen. Die falschen Werte kamen daher aus der Übertragung zwischen dem Arduino und dem Teensy.

Der Teensy und Arduino stehen in einer sogenannten Master-Slave-Beziehung zueinander. Im Allgemeinen heißt das, dass der Master (hier der Teensy) dem Slave (hier der Arduino) bestimmte Aufgaben zuweist. Der Arduino entscheidet nie durch seine eigene Programmierung, was er als Nächstes rechnet, sondern bekommt dies ausschließlich von dem Teensy mitgeteilt. Im Konkreten wurde dies hier über eine Vielzahl an festgelegten Buchstaben realisiert. Der Teensy schickt zu bestimmten Zeitpunkten einen Buchstaben an den Arduino, welcher diesen interpretiert und die gewünschte Aufgabe ausführt. Bei einigen Befehlen wird noch eine Antwort zurückgeschickt, wie z.B. eine Messung der Ultraschallsensoren, welche der Teensy interpretiert und in Abhängigkeit dieser Aktionen plant und durchführt. Andere Befehle, wie z.B. das (re-)booten des Arduinos erfolgen ohne Antwort.

Bedienung

Die Stromversorgung mit dem Board und den verbauten Computern wird mit dem Einschalten des Schieberegler hergestellt. Bei Bedarf können alle Servos mit dem beleuchteten Kippschalter aktiviert werden. Zusätzlich sollte das Display zur Spannungsmessung dann 6V anzeigen.

Anschließend erfolgt die Bedienung des Roboters über einen PS2-Controller, dessen Empfänger in dem Roboter verbaut wurde. Die Steuerung erfolgt nach dem unten dargestellten Diagramm. Wichtig hierbei, sind die vier verschiedenen Modi, durch welche jeder Knopf vier verschiedene Funktionen haben kann.



Der aktuell ausgewählte Modus muss sich zu jeder Zeit im Kopf behalten werden, da es keine Möglichkeit gibt diese nach außen darzustellen. Einer der großen Vorteile dieser Vielzahl an Funktionen sind die Einstellbarkeit des Roboters, während der Benutzung, wie z.B. die

Bewegungsgeschwindigkeit. Zusätzlich besitzt der Roboter noch recht unwichtige Funktionen, wie z.B. das hinsetzen oder -legen, welches keinen praktischen Nutzen hat.

Die Steuerung könnte in unseren Augen noch optimiert werden, sodass die wichtigsten Bewegungen im Mode 1 angesteuert werden können. Zudem fehlt jegliche Möglichkeit den Roboter zu drehen. Aktuell ist die einzige Bewegung, welche den Roboter nach links und rechts bewegt, eine wiederholte schwunghafte Bewegung des gesamten Torsos. Durch diese rutscht er somit leicht zur ausgewählten Seite. Diese Bewegung kann natürlich noch besser implementiert werden.

Zukünftige Veränderungen und Verbesserungen

Bei diesem Projekt gibt es zahlreiche Möglichkeiten für zukünftige Veränderungen, die entweder komplett neue Funktionsweisen bieten oder bereits vorhandenen verbessern.

Überarbeitung der Frame-Teile im Bauchbereich

Eine der simpelsten Veränderungen, wäre den Aufbau des Bauchbereichs zu verändern. Die 3 mittleren Frame-Teile können zu einem langen zusammengefasst werden, was zu einer deutlichen Vergrößerung der nutzbaren Fläche führt.

Hierbei würde sich dann nur noch der Akku auf der Unterseite befinden. Der Servo Controller sowie der kleine Buck Converter könnten auf der Oberseite platziert werden, was Kabelführung und Aufbau erleichtern würde.

Diese Variante wurde bereits von einem Nutzer auf dem Discord-Server angewandt, welcher potentiell für die STL Dateien angefragt werden könnte.

Anschließend sind einmal die veränderte Version zu sehen (oben), sowie den aktuellen Aufbau (zwei Bilder unten).



Servos aufrüsten

Der Entwickler Chris Locke hat die Möglichkeit vorgestellt, die 20kg Servomotoren in den Schultern durch 35kg Motoren zu ersetzen, um die Stabilität zu erhöhen. Zusätzlich hilft diese Veränderung dem Gravizentrum entgegenzuwirken. Die Wahl der Schulter Servos im Gegensatz zu den anderen Motoren kommt daher, dass diese die meiste Last tragen, und somit am meisten Kraft aufbringen müssen, um auf dieselbe Beschleunigung zu kommen ($F = m * a$).

Auch die anderen Servos können bei Bedarf ausgetauscht werden. Allerdings muss hier auf die leicht unterschiedlichen Dimensionen geachtet und die STL-Dateien evtl. angepasst werden.

Remote Controller

In der Dokumentation sind auch Pläne für einen selbstgebauten Controller vorhanden, welcher in den aktuelleren Versionen auch standardmäßig genutzt wird. Im Rahmen dieses Projektes wurde dieser aufgrund der fehlenden Notwendigkeit nicht gebaut, kann aber noch nachgerüstet werden. Wir benutzen einen PS2-Controller, der dieselben Funktionen bereitstellt.



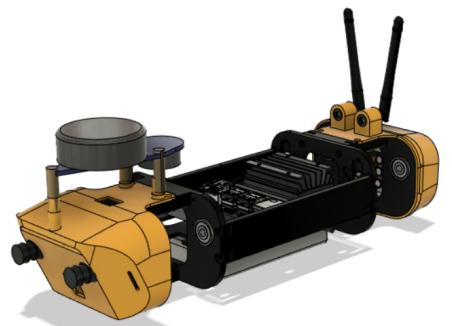
Änderung des Arduino und Teensy zu Raspberry Pi

Eine weitere mögliche Veränderung wäre es, den Arduino Nano und den Teensy gegen einen Raspberry Pi auszutauschen. Dieser stellt eine höhere Prozessorleistung, sowie Speichergröße zur Verfügung und kann somit mehr Anwendungsgebiete erschließen. Wie in vielen anderen Fällen muss auch hier die Stromversorgung, die Drucke und die Anschlüsse überprüft und gegebenenfalls angepasst werden.

Der bedeutende Vorteil, ist, dass der Raspberry Pi stark genug ist um rechenintensive Verfahren wie z.B. das Mapping und die Wegplanung selbstständig zu berechnen. Ohne diesen (also nach aktuellem Stand) müsste die Verbindung mit einem Laptop hergestellt werden, alle Sensordaten an diesen geschickt werden, und die berechnenden Motoreinstellungen zurückgeschickt werden. Der Raspberry Pi würde somit Schwierigkeiten bei der Verbindung entgegenwirken, indem die Berechnungen lokal kalkuliert werden würden.

LiDAR Erweiterung

Zur besseren Erfassung der Umgebung wäre eine LiDAR-Erweiterung eine Möglichkeit. Hierfür müssten verschiedene Aspekte der Elektronik, der Codierung als auch der 3D-gedruckten Rückenteile überarbeitet werden. Auf der Website des Projekts wird auch eine LiDAR-Erweiterung als zukünftig anvisiertes Ziel genannt.



ROS-Anwendung

Eine weitere Möglichkeit besteht darin, eine Anwendung in ROS zu schreiben, um die Automatisierung des Roboters zu vereinfachen und Erweiterungen, sowie bestimmte Verhaltensweisen leichter zu implementieren. Dies geht Hand in Hand mit der Raspberry Pi Erweiterung, die sich dafür optimal anbietet.

Greifarm

Eine komplexere Erweiterung, für die noch keine Pläne existieren, wäre das Anbauen eines Greifarms zum Beispiel auf der Oberseite. Hierfür müssten aber STL-Files als auch die Elektronik des Roboters komplett überarbeitet und neu aufgezogen werden, um einen erfolgreichen Aufbau zu erreichen. Wahrscheinlich wären hierfür unter anderem weitere Servos und eine höhere Akku-Leistung nötig.



Auch für dieses Projekt gibt es Anreize auf dem Discord-Server und Beispielbilder, wie diese Realisierung aussehen könnte.

Außerdem gibt es auch beim Originalroboter von Boston Dynamics¹⁰ eine Version, die einen Greifarm beinhaltet. An dieser könnte sich auch orientiert werden.



Fazit

Die anfangs aufgestellten Ziele, der Aufbau eines funktionstüchtigen Nova Spot Micro, wurden erreicht, wobei es an einigen Stellen noch Verbesserungspotential gibt. So kann der Roboter zum Beispiel laufen, rutscht aber auf glatten Oberflächen schnell weg.

Auch die Reaktion auf Detektionen der Sensoren kann noch angepasst und entsprechend eingebaut werden. Aktuell wird diese bei der manuellen Bewegung des Roboters nicht beachtet und er hat somit keine Kollisionsvermeidung. Dies stellt aber kein Problem dar, da der Roboter keine hohe Geschwindigkeit besitzt, um unerwartet gegen Wände zu laufen.

Des Weiteren gibt es zahlreiche Verbesserungs Ideen und Visionen für die Zukunft, die, wie weiter oben bereits erläutert, auch noch abgewogen und auf Wunsch eingebaut werden könnten. Wenn in Zukunft eine automatische Navigation geplant ist, sind die Sensoren natürlich zu benutzen. Ein Austausch der Infrarotsensoren für einen LiDAR bietet sich hier an, da somit eine genauere Messung in 360° erfolgen kann. Die Ultraschallsensoren als zusätzlichen Kollisionsschutz zu behalten, ist empfehlenswert.

Im Nachhinein ist zu sagen, dass die vielen kleinen und unerwarteten Probleme mehr Zeit gekostet haben als erwartet, wobei hier insbesondere die fehlerhaften Drucke zum Tragen kommen. Zusätzlich war die interne Organisationsstruktur der Druck und Bestelltabellen oft nicht strukturiert genug, was zu Misskommunikation und Zeitverlusten geführt hat.

Abschließend ist anzumerken, dass uns dieses Projekt viel in verschiedenen Bereichen gelehrt hat, wie z.B. der Elektrotechnik, dem 3D-Druck sowie in der Mechanik. Insbesondere hat es uns gut gefallen, dass dieses Projekt ein breites Spektrum an eben genannten Bereichen abdeckt und dass es sehr praxisnah ist, da man die Chance hatte von Grund auf einen Roboter zusammenzubauen und zu programmieren.

¹⁰

https://www.wevolver.com/wevolver.staff/spot.mini&sa=D&source=docs&ust=1651225771942557&usg=AOvVaw2NFwbQpq_sDalhJ7g6i4zB