Studiengang Computervisualistik und Design

Ball Balancing Robot

Autoren:
Susanne Totz
Matr.-Nr.: 2140866
Claudia-Susanne.Totz@stud.hshl.de
Schäferkemperweg 45
59597 Bad Westernkotten

Jannis Becker
Matr.-Nr.: 2140689
Jannis.Becker@stud.hshl.de
Goerdelerstr. 7
59557 Lippstadt

Inhaltsverzeichnis

1 Einleitung

Im Rahmen des Wahlpflichtfaches „Ubiquitous Computing“ im 6. Semester des Studiengangs Computervisualistik und Design ist das Projekt „Ball Balancing Robot“ entstanden. Die Aufgabenstellung besagte, dass die Studenten zusammen in Gruppen einen Hardwareprototypen planen, bauen und letztlich mit einer eigens entwickelten Software lauffähig machen sollen. Die Wahl des Themas und der Systemumgebung war den Studenten dabei selbst überlassen. Als Hilfestellungen konnten Internetquellen und Literatur verwendet werden.

Das Ziel dieses Projektes war es, einen mobilen Roboter zu entwickeln, der eigenständig auf einem Ball balancieren und sich bei einer externen Krafteinwirkung auch möglichst selbst wieder ausrichten kann. Inspiriert wurden wir hierbei durch ein Projekt an der Tohoku Gakuin Universität in Tagajo, Japan, geleitet von Dr. Masaaki Kumagai, der zusammen mit einem Studententeam selbst einen solchen Roboter entwickelt hat. [1]

Dieses entwarf eine Plattform, die, angetrieben von drei mit sogenannten Allseitenrädern bestückten Schrittmotoren, auf einer mit Gummi beschichteten Bowlingkugel balancieren und sich auch selbst oder per Steuerbefehl im Raum bewegen kann. Dieser Roboter ist außerdem in der Lage, Objekte mit bis zu 10kg Gewicht zu transportieren und kann damit sowohl eigenständig oder als maschinelle Unterstützung für den Lastentransport eingesetzt werden.

Die zugrunde liegende Funktionsweise eines solchen, sogenannten omnidirektionalen Roboters wurde in den 70er Jahren entwickelt. Heutzutage haben bereits viele Studentengruppen, aber auch Hobbyentwickler sich einem solchen Projekt angenommen und es existieren zahlreiche Berichte und Dokumentationen über diese im Internet. So ist auch dieses Team über einige Internetquellen sowohl auf das Projekt von Dr. Kumagai, als auch andere Nachbauten gestoßen und wurde somit inspiriert, selbst einen solchen Roboter zu entwickeln.

[1] http://spectrum.ieee.org/automaton/robotics/robotics-software/042910-a-robot-that-balances-on-a-ball

Gegenüber anderen beweglichen Robotiksystemen hat eine omnidirektionale Plattform einige entscheidende Vorteile. So können diese ohne Verzögerung und jegliche Änderung der Blickrichtung direkt in alle Richtungen bewegt werden und sind somit äußerst agil. Zudem können diese Roboter je nach Bauweise sehr klein und mobil gestaltet werden, weshalb sie für zahlreiche Aufgaben in den unterschiedlichsten Bereichen infrage kommen.

Neben den zu implementierenden Funktionen der Balancehaltung und der Bewegung kann dieses System außerdem noch um viele Funktionen erweitert werden, wie z.B. das ferngesteuerte oder autonome Fahren, sowie weitere Sensorik und Aktorik.

2 Problembeschreibung

In diesen Abschnitt wird kurz dargestellt, welche Quellen das Team nutzt, um den Balancing Robot zu realisieren. Ebenso werden die Risiken erläutert, die während der Implementierung Hard- und Software auftreten können.

Die grundlegende Architektur lehnt sich an das Projekt von Dr. Kumagai an, da diese eine sehr gut funktionierende, aber auch vergleichsweise einfach zu implementierende Basis bildet. So werden drei Motoren und Räder verwendet, angeordnet in einem 120° Winkel zueinander. Es gibt durchaus auch einige Ansätze, welche zwei oder vier Antriebe verwenden würden, wobei wir ersteres aufgrund der fehlenden Stabilität und letzteres wegen dem zusätzlichen Kostenaufwand sowie Stromverbrauch für ungeeignet hielten. Auch die restliche Hardware, wie z.B. die benötigte Sensorik und der Mikrocontroller wurden für dieses Projekt eigens ausgewählt, um optimale Kompatibilität mit unserer Plattform und der gewählten Umgebung zu gewährleisten.

Die Verarbeitung und Ansteuerung in der Software funktioniert grundlegend wie folgt: Zuerst wird ein Lagesensor nach der aktuellen Rotation des Roboters gefragt. Diese Daten werden dann in einem Regler weiter verarbeitet und zuletzt an die Motoren gegeben, die eine Schieflage dann ausgleichen. Dieser Ablauf findet viele Male pro Sekunde statt, sodass der Roboter seine Position und Rotation kontinuierlich ausgleicht.

Die Risiken in der Implementierung der Software stecken vor allem darin, dass der Datenfluss nicht einwandfrei funktionieren könnte. Die Messdaten enthalten Schwankungen und können von der Nullposition abdriften. Dies gilt es zu erkennen und zu beseitigen. Außerdem müssen mathematische Konventionen, Wertebereiche und Standards definiert werden, welche in allen Datenströmen und Interfaces zwischen der Sensorik, Verarbeitung und Aktorik eingehalten werden müssen. Ein falscher Wertebereich oder ein Vertauschen von Datenströmen kann hier bereits zu einem sehr willkürlich wirkenden Verhalten des Roboters und daraus folgend einem sehr aufwendigen und zeitintensivem Debugprozess führen.

Da diese Roboter kontinuierlich gegen Schwankungen in jegliche Richtungen, als auch gegen die Schwerkraft arbeiten muss, sollte bei der Gestaltung des Gerüsts und dem Einbau der Hardware insbesondere auf die radiale Gewichtsverteilung, aber auch die Verteilung in unterschiedlichen Höhen geachtet werden. Eine falsche Aufteilung kann hier entweder zu einem nicht genügend trägen Verhalten führen, da eine gewisse Trägheit für das System benötigt wird, oder aber zu einem radialen Ungleichgewicht führen, welches die Steuerung des Roboters prinzipiell sehr viel schwieriger, wenn nicht sogar unmöglich machen würde.

Das Zeitmanagement ist gerade besonders bei Projekten wichtig, in dem vorwiegend Studenten zusammen arbeiten, die unterschiedliche Erfahrungen in Umgang mit Software und eingebetteten System mitbringen. Die Implementationsphase der Software könnte sich schon aufgrund der Tatsache verzögern, dass die Programmiersprache C/C++ eingesetzt werden muss und diese mit einem gewissen Lernaufwand verbunden ist. Die Beschaffung der Hardwarebestandteile sowie das Design des Robotergerüsts benötigen zusätzlich Zeit, weshalb die Entwicklung des Roboters in unterschiedlichen Phasen unterteilt wird, damit das gesamte Projekt fristgerecht abgeschlossen werden kann.

3 Realisierung des Ball Balancing Robot

In diesem Teil der Ausarbeitung werden die Umsetzungsphasen im Projekt "Balancing Robot" aufgegriffen und beschrieben. Die Phasen fanden teilweise parallel statt und können somit nur eingeschränkt unabhängig von einander betrachtet werden.

3.1 Entwicklung der Gestalt des Roboters

3.1.1 Allgemein

Folgende Punkte sind bei der Gestaltung des Ball Balancing Robot zwingend zu beachten, um die Funktionsfähigkeit des Roboters zu gewährleisten:

- Der Roboter benötigt in alle Richtungen eine gleichmäßige Gewichtsverteilung, damit er seine eigene Masse auf dem Ball ausbalancieren kann.
- Die Einbettung des Hardwaresystem mit seinen gesamten Bestandteilen benötigt viel Platz, daher ist dieser Sachverhalt bei der Erstellung des Robotergerüsts zu berücksichtigen. Idealerweise sollte der Roboter nicht in die Breite, sondern in die Höhe, mithilfe mehrerer Ebenen für die verschiedenen Komponenten, gestaltet werden. Somit wird auch die Verwendung kleinerer Bälle ermöglicht und der Roboter ist allgemein mobiler und agiler.
- Die Gewichtsverteilung auf den Ebenen spielt ebenfalls eine wichtige Rolle, um den Roboter vor dem Umkippen zu bewahren. Eine möglichst hohe Trägheit im Vergleich zum Ball gewährleistet, dass der Ball sich unter dem Roboter bewegen kann, und somit einer Übersteuerung der Motoren entgegen gewirkt wird. Außerdem ließe sich eine externe Steuerung des Roboters andernfalls nicht realisieren. Dazu müssen die schweren Akkus möglichst weit unten nahe am Ball platziert werden und die leichte Elektronik und Sensorik finden weiter oben ihren Platz. Zusätzlich bewirkt ein Platzieren des Lagesensors in größerer Entfernung zum Ball einen größeren Ausschlag der Werte, was eine einfachere Auswertung und eine direktere Steuerung ermöglicht.

3.1.2 Prototyp 1

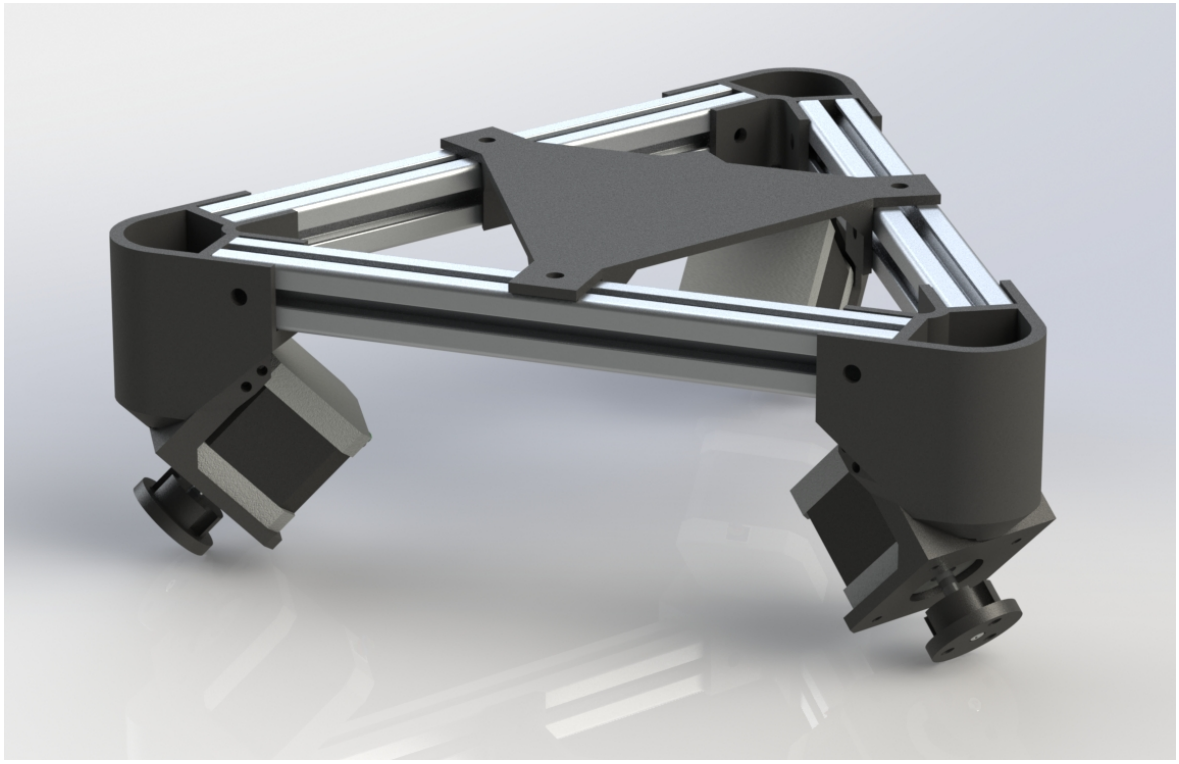

Abbildung 1: Der nicht vollendete erste Prototyp

Der erste Prototyp (Abb. 1) basierte einer dreieckigen Basis, die aus 20*20 mm großen Extrusionsprofilen aus Aluminium bestand. Dieses Grundgerüst war an die sehr ähnliche, übliche Bauweise von “Delta 3D-Druckern“ angelehnt. Der Prototyp war optimiert für den Einsatz von Schrittmotoren (siehe Abschnitt Hardware). Die Motoren wurden dabei auf drei Schlitten montiert, die in einem 45° Winkel an das Robotergehäuse angebracht sind. Der Vorteil dieser Montage ist die einfache Wartbarkeit. (Abb. 2) Das Hardwaresystem, sowie alle anderen notwendigen Bestandteile wie die Stromversorgung finden ihren Platz in der mittleren Plattform und der darunter liegenden Ebene.

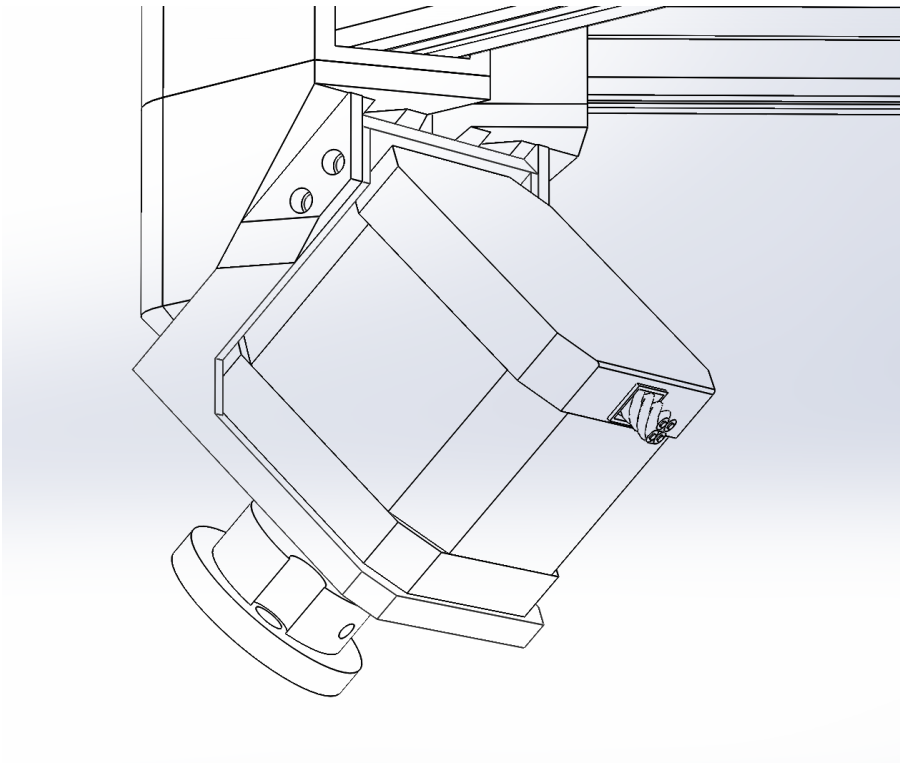

Abbildung 2: Die Befestigung des Motors beim ersten Prototyp

Folgende Vorteile erhofften wir uns von diesem Prototypen:

- Die Aluminiumprofile waren bereits vorhanden, somit hätten weniger Teile nachproduziert (gedruckt) werden müssen.
- Das Gehäuse bestünde aus Aluminium und wäre damit sehr stabil und widerstandsfähig.
- Durch den Einsatz von starken Schrittmotoren wäre ein Lastentransport möglich gewesen.

Durch die folgenden Nachteile haben wir uns jedoch gegen die Nutzung dieses Prototyps entschieden:

- Die gesamte Konstruktion wäre sehr groß gewesen. Die vorhandenen Profile sind 20cm lang, woraus sich ein Gesamtdurchmesser von etwa 30cm ergibt. Da der Ball somit ebenfalls größer ausfallen muss, gäbe es einen großen Einbußen in der Mobilität des Roboters.
- Das hohe Gesamtgewicht der Konstruktion und der Schrittmotoren u.A. hätten die Bewegung erschwert und das System weniger reaktionsfähig gemacht.
- Da der Fokus bei dieser Variante auf der Breite statt auf der Höhe liegt, wären die Trägheit und die Gewichtsverteilung des Systems nicht optimal gewesen und die Ausbalancierung der Komponenten würde erschwert.

3.1.3 Prototyp 2

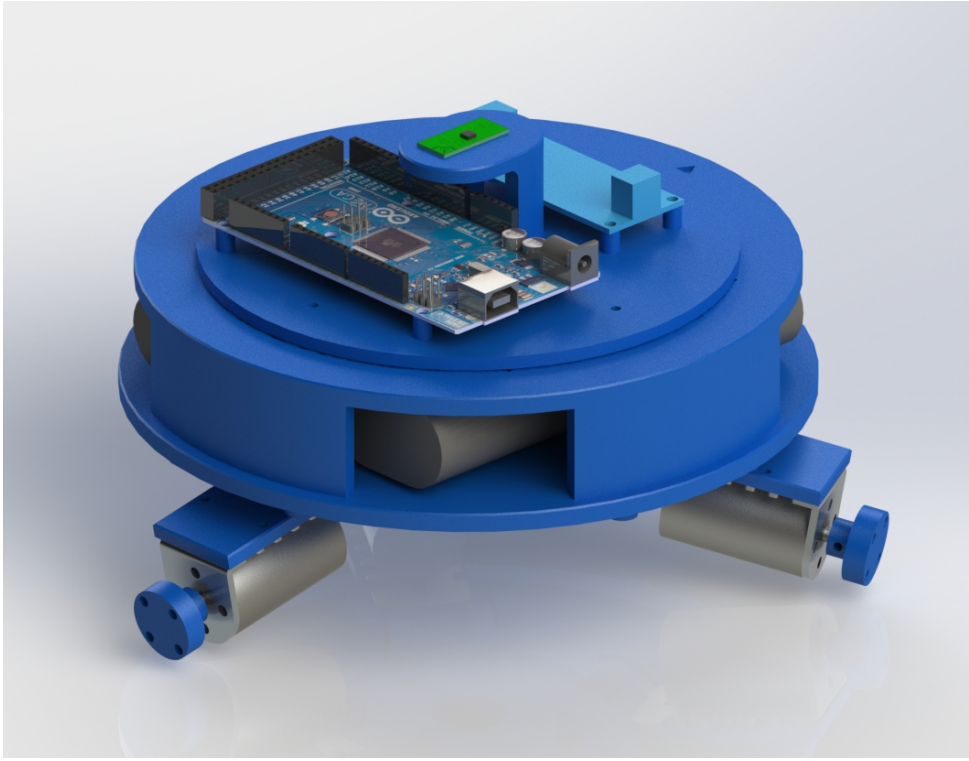

Abbildung 3: Das Rendering des vollendeten zweiten Prototyps, inklusive der meisten Hardware

Aufgrund der Nachteile der ersten Variante ist die Entscheidung gefallen, ein kreisförmiges Grundgerüst zu entwerfen (Abb. 3), bestehend aus leichtem, jedoch stabilem Kunststoff, welches vollständig mithilfe eines 3D-Druckers hergestellt werden kann. Die direkte Verfügbarkeit dessen begünstigte diese Entscheidung wesentlich.
Der Prototyp Nr. 2 hat die folgenden Vorteile gegenüber dem ersten:

- Die Grundform des Gehäuses eignet sich besser zum Anordnen der Hardwarebestandteile und der Stromversorgung.
- Die mehrschichtige Konstruktion lässt eine präzisere, vertikale Gewichtsverteilung zu (Trägheitsoptimierung) und ermöglicht eine bessere Balance in der Horizontalen.

- Sehr viel schmaler, mit eng aneinander liegenden und justierbaren Motoren, daher ist dieser Prototyp für viele Ballgrößen geeignet.
- Die Konstruktion ist insgesamt leichter, kleiner und agiler, dies sorgt dafür, dass die Reaktionen schneller und die Bewegung flüssiger sind.

Jedoch gibt es auch hier den folgenden, jedoch vernachlässigbaren Nachteil:

- Die Konstruktion ist aufgrund der gewählten Materialien etwas instabiler. Somit ist eventuell der Lastentransport nicht gewährleistet. Trotz der Kunststoffkonstruktion ist aber mit Hilfe von Verstärkungsrippen, Verrundungen u.Ä. für ausreichende Stabilität gesorgt.

3.1.4 Aufbau des Prototyp 2 im Detail

Im Folgenden ist der grundlegende Aufbau der einzelnen Ebenen des zweiten Prototyps beschrieben. Die Ebenen werden hier einzeln von unten nach oben betrachtet.

1. Abb. 4: Drei Motoren inklusive Halterungen und angebrachten Adaptern für die Räder, befestigt mit drehbaren Verbindungen an der unteren Grundplatte. Zu sehen sind hier außerdem die Verstärkungsrippen, die Schraubenlöcher zur Befestigung der weiteren Schichten, sowie das zentrale Loch für die Kabelführung.
2. Abb. 5: Obere Grundplatte, inklusive äußeren und inneren "Stelzen", damit zwischen oberer und unterer Platte Hardware untergebracht werden kann. Auch hier sieht man das zentrale Loch für die Kabelführung, die benötigten Schraublöcher, sowie weitere, eckige Löcher, welche später Kabelbinder führen, um die Akkupacks zu befestigen.
3. Abb. 6: Weiterhin die obere Grundplatte, Ansicht von oben. Hier sieht man sehr gut die Vertiefungen für die Kabelbinder, damit direkt darauf aufliegend die weiteren Ebenen befestigt werden können. Eine pfeilförmige Vertiefung symbolisiert die mathematische "Vorderseite" des Roboters.
4. Abb. 7: Befestigungsplatte mit Abstandshaltern, auf dem der Arduino Mega und der Spannungsregler befestigt werden. Der Kabelkanal wird hier zu einem Langloch, um das seitliche Herausführen der Kabel auf dieser obersten Ebene zu ermöglichen. Nicht alle Abstandshalter sind vorgebohrt, da die entsprechenden Schraubenlöcher auf dem Arduino

durch zu kleine Abstände den SMD-Bauteilen nicht für Schrauben genutzt werden können. Hier wird der Arduino dementsprechend nur aufgelegt. Die beiden Löcher neben der Kabelführung verbinden später den sogenannten Gyroturm mit dieser Platte.

5. Abb. 8: Der Gyroturm dient der Befestigung des Gyrosensors auf einer erhöhten Ebene, um dessen Präzision und die Reaktionsfreudigkeit des Gesamtsystems zu erhöhen. Das rechteckige Loch auf der Oberseite besitzt ähnliche Maße wie die Stiftleiste unter der Gyrosensorplatine und hält diesen somit in Position. Die entsprechenden Pins stehen nun aus der Unterseite des Turms hervor, wo sie per Jumperkabel mit dem darunterliegenden Arduino verbunden werden können. Das Loch und der Turm selbst sind so platziert, dass der eigentliche Gyrosensor IC auf seiner Platine möglichst exakt mittig über dem Roboter liegt, sodass X und Y-Rotation des Sensors bei einer Bewegung gleichmäßig ausschlagen.

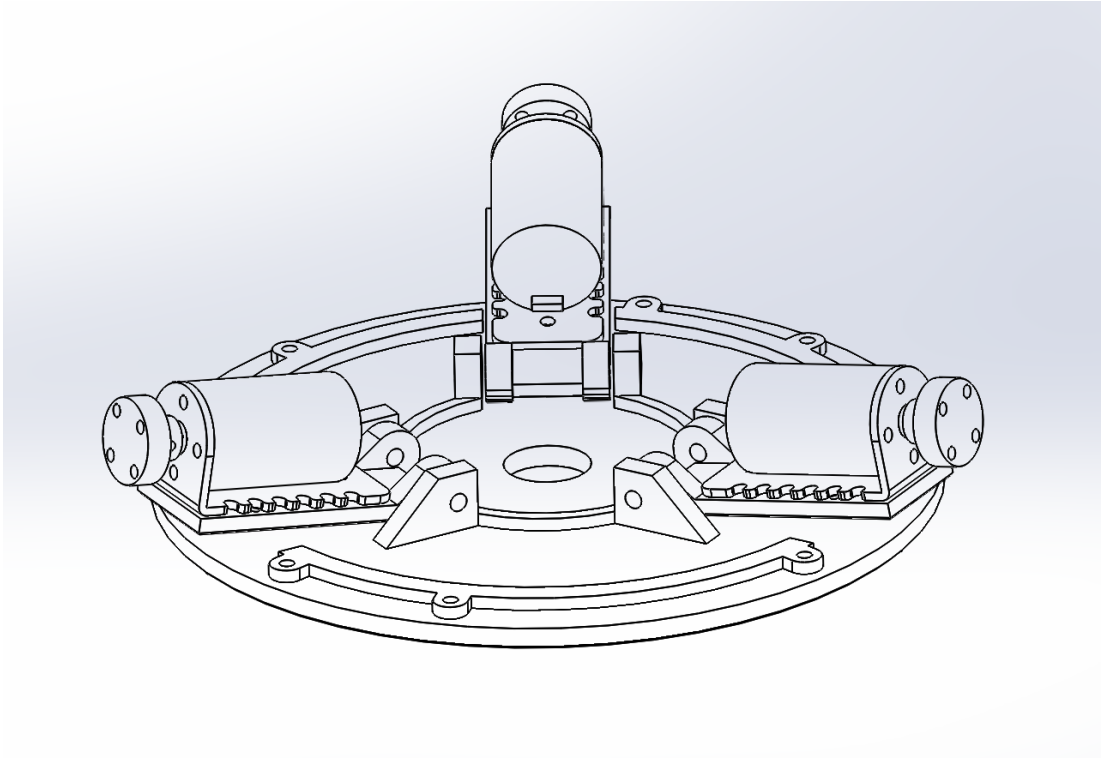

Abbildung 4: Eine Skizze der unteren Grundplatte, inklusive Motoren und Halterungen, Ansicht von Unten

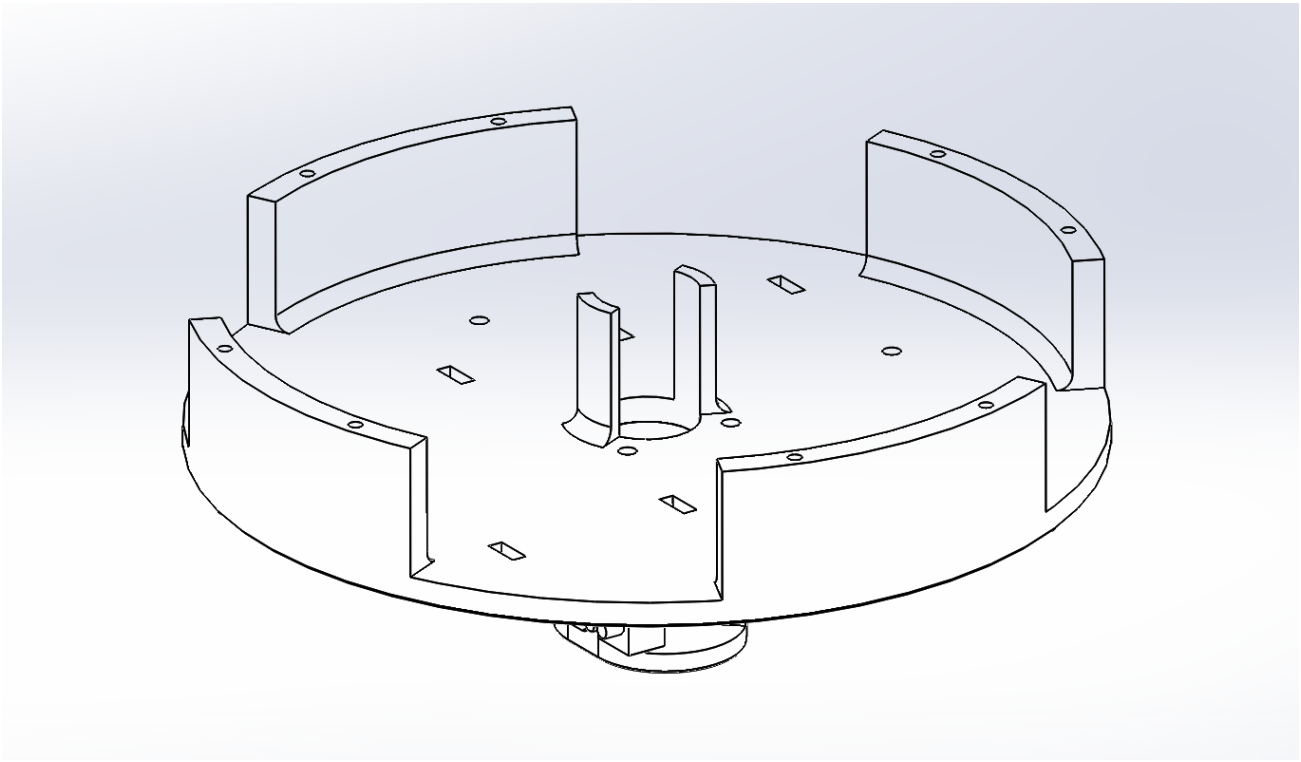

Abbildung 5: Eine Skizze der oberen Grundplatte, Ansicht von Unten

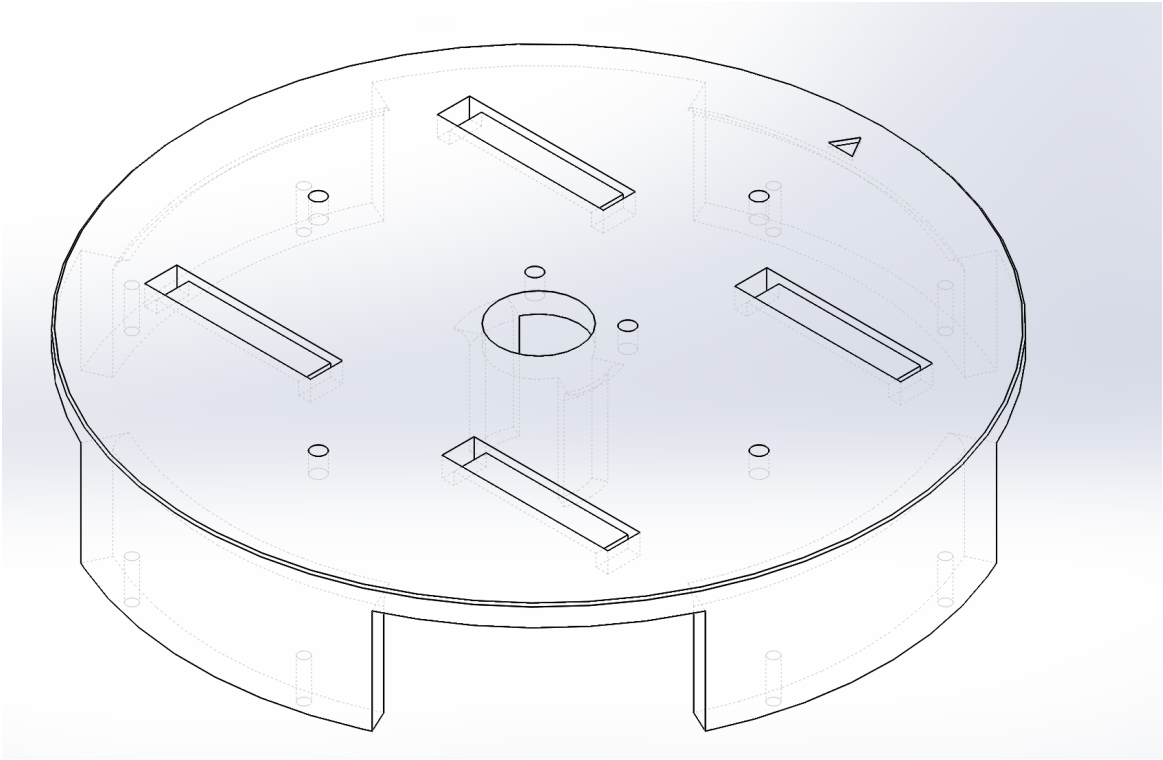

Abbildung 6: Eine Skizze der oberen Grundplatte, Ansicht von Oben

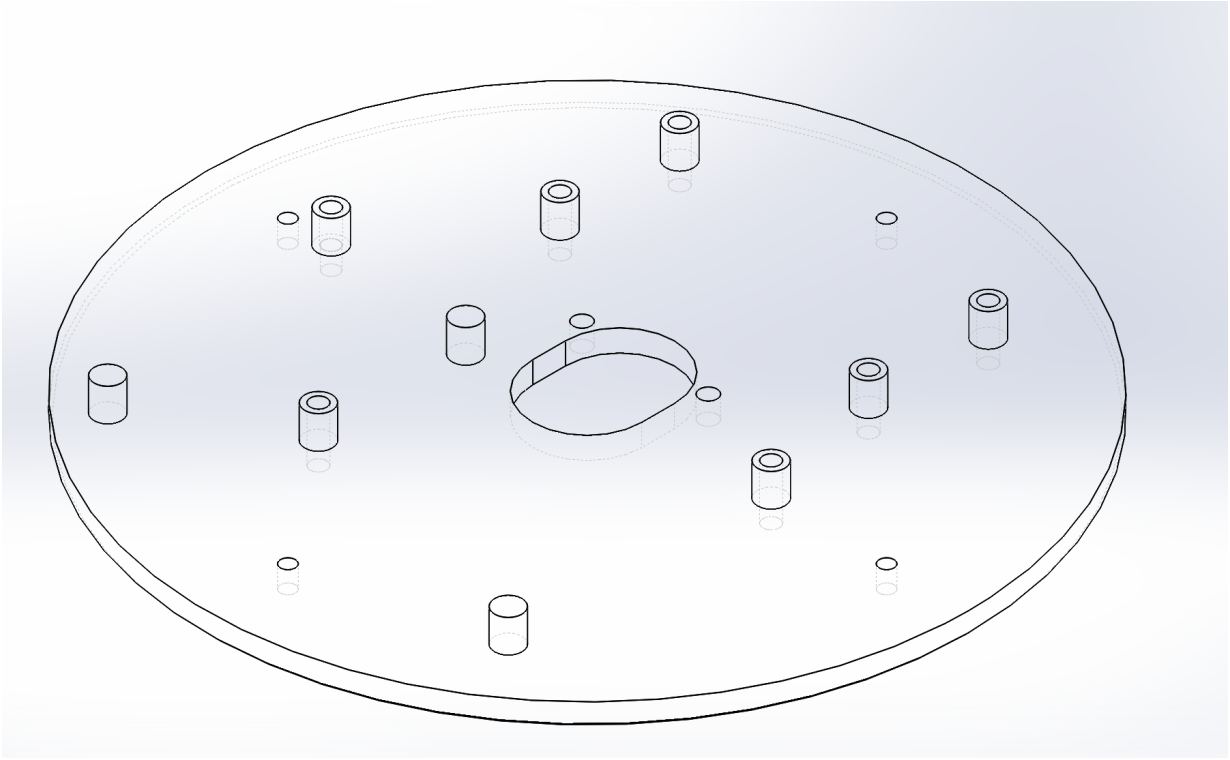

Abbildung 7: Eine Skizze der Befestigungsplatte für Platinen, Ansicht von Oben

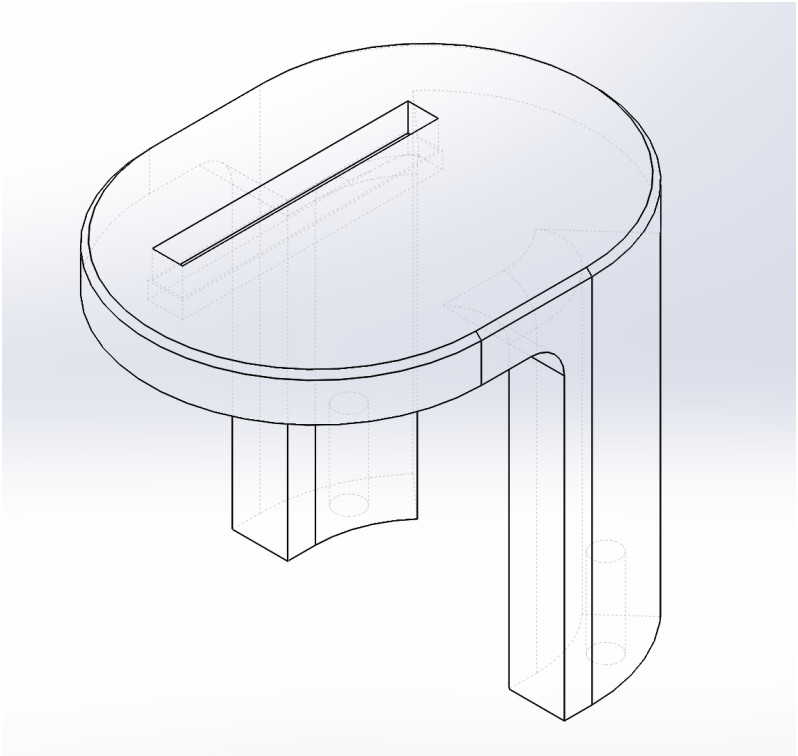

Abbildung 8: Eine Skizze des Gyroturms, Ansicht von Oben

Einige weitere Ansichten, um den Zusammenbau zu verdeutlichen:

6. Abb. 9: Gesamtaufbau, Ansicht von unten. Hier deutlich zu sehen sind die Verbindungen von oberer und unterer Grundplatte, die Positionen der Akkus und der Motoren zueinander. Auch interessant ist der Wechsel von den dreieckig angeordneten Motoren auf der Unterseite zu einem viereckigem System im Mittelteil, dass durch die sperrigen Akkus erzwungen wird.

7. Abb. 10: Gyroturm, befestigt auf der Platinenebene sowie der oberen Grundplatte, inklusive Arduino Mega und Stepdown Wandler. Hier ist die genaue Position des Gyro ICs über dem kreisförmigen Kabelkanal gut zu sehen, sowie die Austragung zum Langloch in Richtung der Vorderseite, da der Arduino etwa die Hälfte des Kabelkanals durch seine Größe bereits verdeckt.

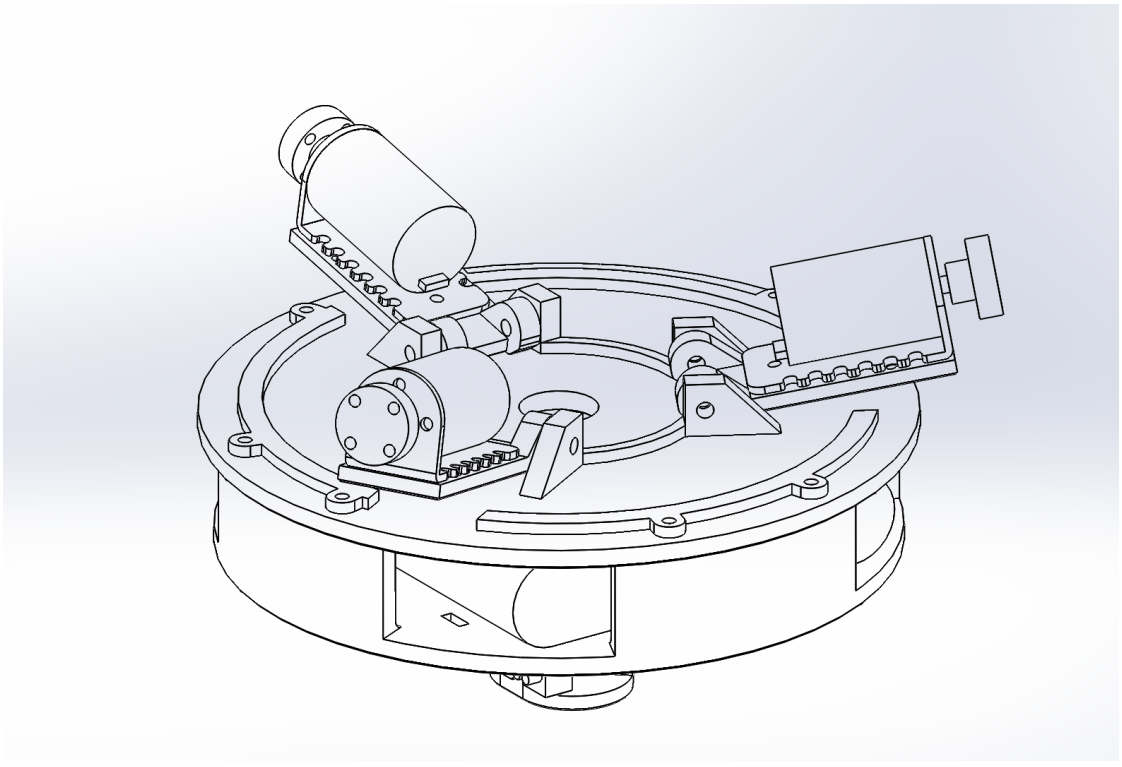

Abbildung 9: Eine Skizze des Gesamtaufbaus, Ansicht von Unten

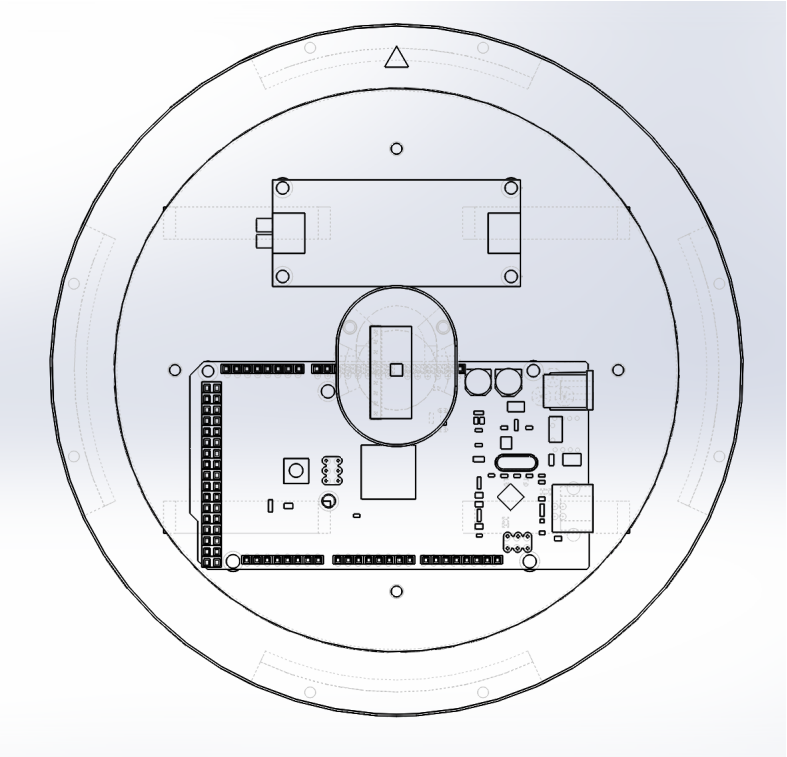

Abbildung 10: Eine Skizze des befestigten Gyroturms auf der PCB Platte und oberer Grundplatte, Ansicht von Oben

3.2 Aufbau des Hardwaresystems

3.2.1 Allgemeines

Zeitgleich mit der Entwicklung der Gestalt kam auch die Frage nach der Hardware auf. Beim ersten Prototypen plante man mit der Verwendung von NEMA 17 Schrittmotoren, weil diese aus dem Bereich des 3D-Druck bereits bekannt waren und ein hohes Drehmoment und sehr präzise Bewegungen erzielen können. Nach einigen Tests mithilfe eines DRV8825 Schrittmotortreibers (ebenfalls üblich im 3D-Druck) fielen aber sofort die Probleme des Systems auf. Zum einen sind Schrittmotoren nicht unbedingt dafür ausgelegt, sehr viele, schnelle Bewegungswechsel zu vollziehen, insbesondere, wenn ein Wechsel von einer hohen Drehzahl aus stattfinden soll. Andererseits verlieren diese Schrittmotoren bei hohen Drehzahlen gerne einzelne Schritte, oder drehen sich gar nicht mehr. Ein plötzliches Auftreten dieses Problems im endgültigen Roboter wäre fatal, weshalb das Team entschied, diesen Ansatz nicht weiter zu verfolgen. Da aber der Roboter von Dr. Kumagai ebenfalls erfolgreich Schrittmotoren (NEMA 23) verwendet, stellt sich die Frage, ob es nicht vielleicht an den getesteten Schrittmotoren, oder aber an der benutzten AccelStepper Library liegt.

Mit dem Wechsel zum zweiten Prototypen mit seinm kleineren Kunststoffgerüst hatte sich die Frage nach den Motoren sowieso erübrigt. Nun kommen DC (Gleichstrom) Motoren zum Einsatz, welche kleiner, leichter und trotz dessen für ihre Größe stark und schnell genug sind. DC-Motoren bieten den Vorteil, dass sie für eine Maximalgeschwindigkeit ausgelegt sind, die sie immer und sicher erreichen können, somit sind zu hohe Geschwindigkeiten kein Problem. Des weiteren sind sie sehr agil und für schnelle Richtungswechsel ausgelegt.

3.2.2 Bestandteile der Hardware

3.2.2.1 Mikrocontroller

Das Rechensystem des Roboters bildet ein Arduino Mega 2560 rev 3. Dieser bietet ausreichende Leistung für den Roboter, da dieser etwa 100 Mal pro Sekunde einige ausführliche Berechnungen vollführen muss. Zudem sind alle nötigen Anschlüsse in genügender Zahl vorhanden und die zusätzlichen Anschlüsse, sowie die reichlich vorhandene Leistung bieten genug Spielraum für Erweiterungen.

3.2.2.2 Gyrosensor

Die Rotation des Roboters wird durch einen sogenannten Gyrosensor, genauer gesagt dem Modell LSM6DS33 von Polulu erfasst. Dieser kann per Serial oder SPI Schnittstelle mit dem Arduino ohne großen Aufwand verbunden werden und besitzt einen integrierten Spannungswandler, sodass das Betreiben an einem 5V Arduino kein Problem darstellt.

3.2.2.3 Motoren und Räder

Der im Ball Balancing Robot verwendete Motor, mit der Nummer FIT0441 von DFRobot, bietet neben den bereits genannten Vorzügen der DC-Motoren zusätzlich den Vorteil, dass der benötigte Motortreiber, als auch ein (in diesem Projekt nicht verwendeter) Encoder direkt im Motor integriert sind, so dass diese ausschließlich eine 12V Versorgung und einen Microcontroller benötigen.

Unverständlich war dagegen der Aspekt, dass diese, eigentlich für das Prototyping ausgelegten Motoren mit JST-Steckern versehen waren, welche nicht mit den für das Prototyping üblichen Dupont Stecksystemen kompatibel sind. Hier wurden kurzerhand in mühevoller Handarbeit die Buchsen gegen Dupont Varianten ausgetauscht, um den weiteren Zusammenbau erheblich zu erleichtern.
An die Motoren angeschlossen sind insgesamt drei Allseitenräder, genannt Omni Wheels, in der 38mm Ausführung von noDNA. Es ist zwar eine 60mm Ausführung erhältlich, diese war aber zum Zeitpunkt des Einkaufs nicht verfügbar. Es könnte aber gut möglich sein, dass die Motoren mit den 60mm Rädern zu wenig Kraft aufbringen können, deshalb sollte lieber auf die kleinere Variante zurückgegriffen werden.

3.2.2.4 Akkus und Stromversorgung

Versorgt wird die Elektronik von zwei 7.2V 2400 mAh NiMh Akkupacks aus der Modellbau Branche, welche in Reihe geschaltet etwa 14-15V liefern. Diese Spannung wird durch einen XL4015 Spannungswandler (Step Down) auf stabile 12V reduziert und dann an den Arduino und die Motoren gegeben.

3.2.3 Hardware in der Gestalt

3.2.3.1 Aufteilung

Wie im Kapitel der Gestalt bereits erwähnt, ist eine intelligente Aufteilung essentiell für die einwandfreie Funktion des Roboters. Einerseits sollten die einzelnen Komponenten, insbesondere jedoch die schweren Motoren und Akkupacks mit Bedacht ausbalanciert werden, damit der Roboter sein Gleichgewicht halten kann und andererseits sollte auch auf die Aufteilung in der Vertikale geachtet werden, denn bspw. ein Einbau der Akkupacks in höheren Ebenen würde den Roboter sehr viel anfälliger für das Umkippen auf dem Ball machen, während ein Einbau möglichst weit unten dies erstens verhindert und zweitens für eine bessere Haftung auf dem Ball und ein optimales Trägheitsmoment des Roboters sorgt.

In unserem aktuellen Prototyp 2 befinden sich die Motoren und Akkus in den untersten Ebenen, nahe am Ball, und die Elektronik findet weiter oben ihren Platz, wobei insbesondere der Gyrosensor ganz oben, idealerweise noch erhöht platziert wird, um sein Reaktionsverhalten zu verbessern. Außerdem sollte dieser möglichst exakt mittig platziert werden (mit Fokus auf dem eigentlichen IC, nicht der Platine), sodass die Messdaten des Sensors gleichmäßig ansteigen.

3.2.3.2 Kabelführung und Anschluss

In diesem Kapitel sei einmal die Kabelführung im Roboter, von unten nach oben betrachtet, erklärt.

1. Ganz unten im Roboter befinden sich die drei Motoren, mit jeweils einer zweipoligen Stromverbindung und einem weiteren dreipoligen Kabel für die Übertragung der Geschwindigkeit, Richtung sowie dem Encoderfeedback (in Prototyp 2 unbenutzt). Diese werden nun durch das Loch der unteren Grundplatte in die zweite Ebene geführt.

2. Dort befinden sich die beiden 7.2V Akkupacks, per Y-Kabel in Reihe geschaltet. Das daraus entspringende Kabel (nun mit 14-15V) wird, zusammen mit den Motorkabeln, durch das nächste Loch, weiter nach oben geführt.

3. Durch die obere Grundplatte und die PCB Befestigungsplatte hindurch werden diese Kabel nun in Richtung Vorderseite (entlang des Langlochs) und unter dem Stepdown Wandler hindurch geführt und die Kabel der Akkus werden hier mit diesem verbunden. Die Stromkabel der Motoren werden an den Ausgang (12V) des Wandlers geschaltet, selbiges gilt für den Arduino. Zuletzt werden noch die Datenverbindungen der Motoren an den Arduino gesteckt, und der Gyrosensor ebenfalls mit diesem verbunden.

3.3 Implementierung der Software

Die Software ist in mehreren Schritten entstanden, da die benötigten Hardwarekomponenten nicht alle gleichzeitig eingetroffen sind. Zuerst kümmerten wir uns um den Gyrosensor und einige Tests mit den nun verworfenen Schrittmotoren, danach kamen die neuen Motoren und zuletzt kam die Datenverarbeitung sowie der Zusammenschluss der einzelnen Softwarebestandteile. Insgesamt lässt sich die Software in 3 Modulen beschreiben, jedes besitzt in der Implementierung auch eine eigene Klasse: Der Gyrosensor und dessen Verarbeitung, das PID und dessen Regelung und zuletzt die Motoransteuerung und die Kräfteverteilung

3.3.1 Gyrosensor und Filter

Erfahrungen mit Gyrosensoren waren praktisch gar nicht vorhanden. Um ein Gefühl für die Empfindlichkeit und den Wertebereich des Sensors zu bekommen, wurde zuerst ein Minimalprogramm für den Arduino Mega programmiert, welches die gelesenen Rotations- und Beschleunigungsdaten über die serielle Schnittstelle an den Steuerungsrechner überträgt. Um ein ausreichend schnelles Steuerungsverhalten zu erhalten, muss der Gyro mindestens alle 100ms ausgelesen werden. Die dabei anfallende Datenflut ist jedoch in Textform nicht mehr zu überblicken. Daher suchten wir eine Bibliothek, um die serielle Schnittstelle und somit die Sensordaten unabhängig von der Arduino IDE auslesen und visualisieren zu können. Am geeignetsten erschien uns ein Visualisierungsprogramm in Java, da wir mit dieser Sprache schon in früheren Semestern reichlich Erfahrung in der GUI Programmierung sammeln konnten.
Für die Anbindung an den seriellen Bus gibt es unter Java verschiedene Bibliotheken, die jedoch alle eine native, in C++ programmierte Komponente benötigen. Wir haben uns für nrjavaserial entschieden, da diese die nativen Bibliotheken bereits mit bringt und auch selbstständig verwaltet. Mithilfe des EventListener Pattern wurde eine grafische Anzeige der jeweils über Schnittstelle empfangenen Daten des Sensors implementiert (Abb. 11). [2]

[2] https://github.com/NeuronRobotics/nrjavaserial

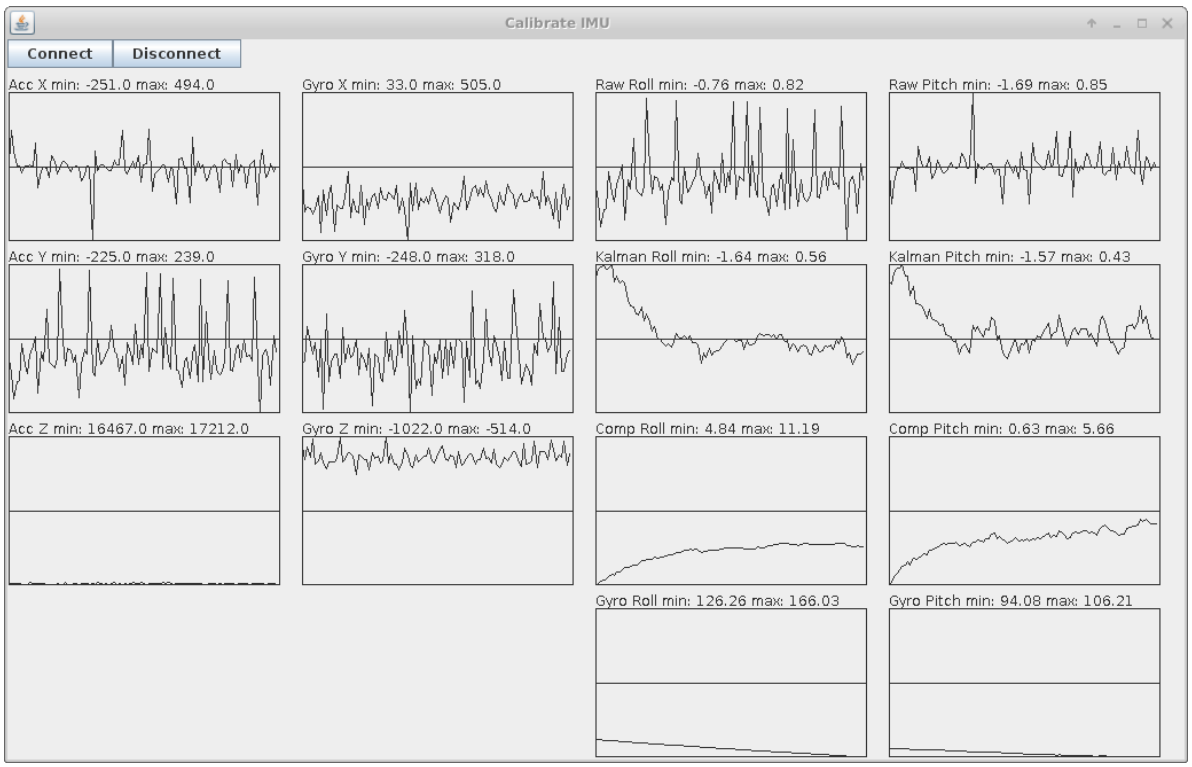

Abbildung 11: Kalibrierung des Gyrosensors

Die genaue Ausrichtung des Messplatine genau waagerecht oder senkrecht war händisch nicht möglich. Daher schraubten wir den Sensor temporär auf einen rechtwinkligen Holzklotz. Somit konnte eine Messung in jede 90° Richtung angemessen werden. Dabei stellte sich schnell heraus, dass die Daten des Gyrosensor definitiv weiter verarbeitet werden müssen, da sonst ganz schnell ein Abdriften der Rotationsdaten stattfindet. Außerdem wurde eine weitere Eigenschaft des von uns gewählten Sensors durch die Visualisierung deutlich: Die hohe Empfindlichkeit erzeugte auch ein starkes Messrauschen, welches schon durch kleinste, unbemerkte Vibrationen, durch Wärmeunterschiede, oder aber durch Spannungsschwankungen entstehen kann.

Es musste also ein Filter eingebaut werden, der die Messwerte glättet und einen Datendrift verhindert, bevor diese an die Motorregelung weitergegeben werden können. Auch hier gab es bereits verschiedene Bibliotheken mit unterschiedlich komplexen Filtern. Die Ergebnisse wurden ebenso wie die Messwerte zur Visualisierung an den Steuerungscomputer übertragen und bewertet. Nach eingehender Betrachtung schien der Komplementärfilter eine ausreichende Glättung bei beherrschbarer Komplexität aufzuweisen und wurde daher im weiteren Verlauf verwendet.

Nach der Implementierung des Komplementärfilters ist jedoch ein neues Problem ersichtlich geworden. Da der Komplementärfilter die Messwerte über die Zeit integriert, sind etwa die ersten 200 Messwerte (die ersten 2 Sekunden der Laufzeit) nicht zu gebrauchen, da sich die Messwerte erst nach und nach der richtigen Kurve annähern. Außerdem wurde eine Möglichkeit gesucht, den Nullpunkt des Sensors neu setzen zu können, so dass sich dieser beim Start bzw. beim Aufsetzen des Roboters auf den Ball präzise festlegen lässt. Beide Probleme ließen sich mit einer Kalibrierungsmethode lösen, welche zu erst die genannten 200 Messwerte liest und überspringt, um dann etwa 300 Messproben zu nehmen, den Durchschnitt zu errechnen, und dies von den, im späteren Programmablauf folgenden Messungen abzuziehen. Dieses Methode wird zu Beginn des Programmablaufs aufgerufen, womit der Nullpunkt beliebig neu gesetzt werden kann, und damit die gemessenen Werte nahe dem Nullpunkt auch wirklich nahe Null liegen.

3.3.2 PID Regelung

Den nächsten Programmabschnitt bildet das PID. PID steht für "Proportional, Integral, Derivative" und bezeichnet eine der am universellsten einsetzbaren Regelungsimplementierungen. Der PID Regler wird in unserem Fall verwendet, damit sich der Roboter in einem möglichst weichen Verlauf an den Nullpunkt annähert. Zusätzlich lässt sich dieser durch drei Variablen beliebig konfigurieren, um das Ansteuerungsverhalten des Roboters an die Umgebung anpassen zu können. In der Implementierung existieren zwei unabhängige, aber identisch konfigurierte PID Regler, jeweils einer für die Rotationen (Abweichungen von der Null-Lage) in X- und Y-Richtung.

Hier gab es im Projekt lange Zeit Probleme. Zu Anfang gab die Regelung ausschließlich Nullwerte aus, aber dies hatte sich schnell erübrigt. Insbesondere die Findung der richtigen Parameterwerte gestaltete sich über den Großteil des Projektzeitraumes als schwierig, denn der Regler war oft am Übersteuern, oder hatte ein hohes Maß an Rauschen und viele Datenspitzen (peaks) im Datenstrom, wobei beides für die Robotersteuerung ungeeignet wäre und zu unerwünschten Verhalten führen konnte.

Letztlich haben sich sehr schwache Werte als ideal erwiesen. Stattdessen wurde dem Regler die Datenglättung überlassen, während die Stärke des Motoroutputs über eine weitere Variable linear und ideal angepasst wurde.

3.3.3 Ansteuerung der Motoren

Der letzte Programmteil kümmert sich um die Auswertung der PID Werte und das letztliche Ansteuern der Motoren. Da die PID-Regler zwei Werte liefern, die das gewünschte Ansteuerverhalten der Motoren in X und Y-Richtung beschreiben, es aber drei Motoren gibt, müssen diese Koordinateninformationen in einer durchaus komplexen Verarbeitung auf die drei Motoren aufgeteilt werden. Das Grundprinzip ist hierbei aber relativ einfach: Die PID-Regler liefern zwei Werte, X und Y, die auch als Vektor aufgefasst werden können. In einem Koordinatensystem kann so nun ein Winkel ausgerechnet werden. Den festgelegten Standards des Projekts zufolge ergibt ein Vektor in Y-Richtung einen Winkel von 0° und der Winkel erhöht sich gegen den Uhrzeigersinn. Die Y+ Richtung ist als “Vorderseite“ (Siehe Kapitel 3.1.3) des Roboters definiert, deshalb ergibt es Sinn, hier einen eindeutigen Wert zu erhalten.

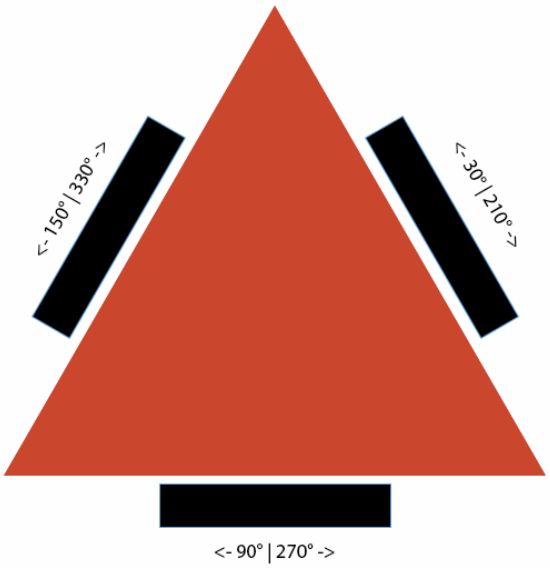

Abbildung 12: Die einzelnen Stellwinkel der Räder und jeweils beiden Bewegungsrichtungen im Koordinatensystem

In Abb. 12 wird gezeigt, in welche Richtungen die beiden Drehrichtungen jedes Rades im oben definierten Koordinatensystem zeigen. Sobald nun aus den PID-Daten ein Vektor erstellt und dessen Winkel im selben System ausgerechnet wurde, kann dieser Winkel nun mit jedem der gezeigten Winkel verglichen werden. Sollte der Unterschied kleiner als 90° ausfallen, so muss sich das jeweilige Rad in diese Richtung bewegen. Je kleiner der Unterschied ausfällt, desto stärker wird sich das Rad drehen. Wenn sich z.B. eine Bewegungsrichtung von 330° ergibt, wird sich das Rad vorne links mit der maximalen Geschwindigkeit gegen den Uhrzeigersinn drehen, das Rad vorne rechts wird sich etwas langsamer (60° Differenz) mit dem Uhrzeigersinn und das hintere Rad wird sich ähnlich langsam ebenfalls mit dem Uhrzeigersinn drehen (ebenfalls 60° Differenz). Im Programmcode sind dies normalisierte Werte, die nun mit der Länge (Stärke) des PID Vektors multipliziert und zusätzlich mit einem weiteren einstellbaren Motor Gain Wert verrechnet werden. So ergeben sich in etwa Werte von 0-255 für jeden Motor, welche dann die Motorsteuerung per Pulsweitenmodulation an die Motorpins weiterleitet.

4 Fazit

Was haben wir aus dem Projekt gelernt? Während der Entwicklungsphase des Ball Balancing Robot ist das Team mit einigen Problemen konfrontiert worden, die es zu lösen galt. In diesen Abschnitt der technischen Dokumentation sollen diese Probleme diskutiert werden und alternative Ansätze kurz dargestellt werden. In der ersten Phase des Prototyping sind zwei Entwürfe entstanden (siehe Kapitel 3.1.2 und 3.1.3), die mit ihren Vor- und Nachteilen beschrieben wurden. Die Wahl fiel auf den Prototyp 2, der mit Hilfe der 3D Drucktechnik realisiert wurde. Bereits bei der Anfertigung ist dem Team aufgefallen, dass der Druck der Plattformen mit der gewünschten Stärke zu lange dauern würde und dass es allgemein Probleme mit dem Druck gab. Aus diesem Grund wurden die Platten 3mm dünner gestaltet und die Dicke mit 3mm starken Sperrholzplatten kompensiert. Bei der Montage fiel die geringe Festigkeit des Kunststoffs negativ auf, da es schwierig war, stabile Gewinde für die Montage der Adapter für die Räder in das Plastik einzuarbeiten. Das Gewinde wies nicht die Festigkeit auf, um die Räder stabil auf der Achse der Motoren zu fixieren. Bei einer zweiten Entwicklungsphase würde das Team als Alternative den Prototyp mit Hilfe einer CNC-Fräse aus Metall fertigen, um so die benötigte Stabilität und Festigkeit zu gewährleisten.

Der Entwurf des Prototyp hat sich als erstaunlich einsatzfähig gezeigt und es gab insgesamt wenig Probleme bei Montage der Hardware und des Gehäuses. Dennoch sind einige Anpassungen empfehlenswert, um das Gehäuse zu optimieren. Zum Beispiel wäre die Kabelführung zu optimieren, da diese gerade ausreichend Platz bot, um alle Jumperkabel und Stromkabel des Akkus anzuordnen. Ein Bündelung der Kabel je Motor erweist als ebenfalls praktisch, da das Stecken der Pins somit sehr viel einfacher ist. Hier könnte z.B. auch der Gyroturm etwas erhöht werden, damit darunter mehr Platz für die Kabelverlegung vorhanden ist. Die Halterung der Räder wurde so entwickelt, das man diese flexibel einstellen kann. Es hat sich dennoch bei der Testphase des Prototypen herausgestellt, dass sich ein Abstandhalter zwischen Gehäuse und Motor-Radhalterung als Vorteil erweist. Alternativ hätte man auch eine fixe Halterung montieren können, die auf einem festen Balldurchmesser angepasst ist. Zusätzlich hätte man einen Ein/Aus Schalter und einen dedizierten Ladeanschluss verbauen können. Das Ein/Ausschalten funktioniert zu diesem Zeitpunkt noch über das Anschließen eines der Stromkabel, und für das Laden der Akkus muss der Roboter zum Großteil demontiert und nachher wieder zusammengebaut werden.

Die Softwareentwicklung ohne die Hardwarebestandteile hat sich als schwierig erwiesen, da ein kontinuierliches System einen Datenfluss beinhaltet, den es am Anfang zu simulieren galt. Nach kurzer Zeit wurde jedoch festgestellt, dass der Einsatz der Hardwarebestandteile zwingend notwendig war, um die Software zu entwickeln, sodass diese auch auf das Hardwaresystem des Roboters übertragen werden kann. Insbesondere das PID System braucht realistische Änderungswerte des realen Systems, sonst lässt sich dieses nicht wirklich testen. Der Einsatz vorhandener Bibliotheken erleichterte zwar die Programmierung, aber dennoch waren in der Testphase einige Anpassung nötig um die Probleme zu beseitigen. Der Gyrosensor verfügt über eine eigene Bibliothek, deren Funktionen korrekt sind, dennoch gab es Probleme bei der Kalibrierung, da die Koordinaten teilweise vertauscht waren. Beim Testen des Gyrosensors an sich ist dieser Sachverhalt noch nicht aufgefallen. Allerdings wurden die ersten Test mit einem anderen, ähnlichen Sensor durchgeführt. Bei den ersten Tests hat es sich zudem herausgestellt, dass es sinnvoll ist, die Filter für die Angleichung der Koordinaten zu testen, in dem man Plotter

für diese einsetzt. Aus diesem Grund wurde ein Java Plottertool entwickelt, dass zur visuellen Überwachung des Datenstroms dient.

Bereits in einer frühen Phase hat sich herausgestellt, dass es beim Einsatz des Komplementärfilters zu einer Abflachung kommt, sodass man diesen alle paar Sekunden zurücksetzen musste. Der Datenfluss wies nach eine Weile regelmäßig solche Abweichungen auf, so dass ein anderer Komplementärfilter zum Einsatz kam. Der PID-Regler war in seiner Implementation und Kalibrierung am komplexesten. Die Stellwerte des Reglers waren am Anfang zu hoch eingestellt, dadurch neigte der Roboter stark zum übersteuern und damit zu einem ruckartigen Fahrverhalten. Nach der schrittweisen Anpassung der Regelparameter sowie der Anpassung der Motoransteuerung konnte auch dieses Problem gelöst werden. Zudem gab es noch die Probleme mit verdrehten Koordinaten und großen Wertebereichen, die dazu führten, dass die Räder des Roboters sich nicht in die gewünschte Richtung und nicht mit der berechneten Geschwindigkeit gedreht haben. Nach dem Abschluss der Testphase hat sich herausgestellt, dass der PID-Regler für diesen Prototypen nicht zwingend notwendig gewesen wäre. Der Kehrwert der Rotation, die mit Hilfe des Gyrosensors ermittelt wird, ist zur Steuerung der Motoren ausreichend. Dennoch ist das PID sehr hilfreich, gerade wenn der Prototyp weiterentwickelt, oder auf andere Umgebungen angepasst werden soll.

Der Zeitaufwand für die Prototyping- und die Testingphase wurde von unserem Team etwas unterschätzt. Bei der nächsten Projektdurchführung wäre es sinnvoll, mehr Zeit für diese Phasen einzuplanen, da wir den Zusammenhang zwischen der Interaktion des Prototypen mit der Umwelt unterschätzt haben und immer wieder Anpassungen notwendig waren. Als Fazit lässt sich ziehen, dass wir als Team viel über den Bereich Ubiquitous Computing gelernt haben und vor allem unser Wissen im Bereich der Elektrotechnik, der Softwareentwicklung für Hardwaresysteme und die Grundlagen der Robotik ausbauen und erweitern konnten. Das Projekt hat uns sehr viel Spaß gemacht und wir sind mit den ersten Erfolgen sehr zufrieden.

5 Sonstiges

5.1 Inbetriebname des Systems

1. Im ausgeschalteten Zustand liegt eines der Stromkabel frei. Schließen Sie dieses an das korrekte Eingangsterminal des Stepdown Wandlers an (Rot auf +, Schwarz auf -). In der Regel sollte das andere der beiden Kabel noch in seinem vorgesehenen Terminal stecken, schieben Sie also das übrige Kabel in das anliegende Terminal und schrauben Sie es im Terminal fest.

2. Der Roboter wird nun durch mehrere LEDs signalisieren, dass er Strom bekommt. Stellen Sie den Roboter auf den vorgesehenen Ball und drücken Sie einmal auf den Reset Knopf des Arduino Controllers, damit der Roboter eine korrekte Kalibrierung vollführt.

3. Der Roboter wird sich nun kalibrieren. In der Standardeinstellung dauert dies etwa 5 Sekunden, halten Sie den Roboter so lange still in der Ruheposition fest.

4. Nun können Sie versuchen, den Roboter langsam zu einer Seite des Balls zu kippen. Wenn er auf diese Neigung ruhig, aber mit genügend Kraft reagiert, ist der Roboter nun funktionsbereit!

Anhang

Beigefügt zu diesem Dokument befindet sich der Ordner “Anhang“, in dem Sie den Quellcode des Java Plotting Tools vorfinden. Des momentanen fehlenden Zugriffs halber werden die restlichen Quellcode Snippets, wie z.B. die Testcodes für die Motoraufteilung, das PID, den Gyrosensor und weiteres in naher Zukunft in einer github Repository zu finden sein. Bei Bedarf eines Links zu diesem Repository, kontaktieren Sie uns einfach unter den Email Adressen auf der Titelseite.