Bachelorarbeit

Entwicklung einer auf Gewichtsverlagerung basierenden Steuerung für ein elektrisches Longboard

vorgelegt von

Mirko Hartung

MIN-Fakultät

Fachbereich Informatik

Arbeitsbereich Arbeitsbereich Technische Aspekte Multimodaler Systeme

Studiengang: Informatik

Matrikelnummer: 6947844

Abgabedatum: 22. März 2020

Erstgutachter: Dr. Norman Hendrich

Zweitgutachter: Florens Wasserfall

Abstract

In dieser Bachelorarbeit wird die Entwicklung eines neuen Mechanismus zur Steuerung eines elektrisch angetriebenes Longboards beschrieben. Die Steuerung arbeitet mit der Gewichtsverlagerung der Nutzer und setzt diese in Beschleunigungs- oder Bremsanweisungen um. Die Lage der Nutzer wird mit der Hilfe von an der Achse montierten Wägezellen ermittelt. Für die Auswertung der Sensoren und Ansteuerung der Elektromotoren wird ein Mikrocontroller verwendet. Die Montage von Sensoren und Mikrocontroller als auch die Verbindung von Sensoren, Motoren und Controller stellen den ersten Teil der Arbeit dar. Im zweiten Teil werden Auswertungsstrategien der Sensoren auf dem Mikrocontroller diskutiert. Zusätzlich wird die Diagnosesoftware vorgestellt, mit welcher Programmfehler und Messdaten ausgewertet wurden, und die kabellose Kommunikation zwischen dem Controller und einem Laptop erläutert. Als Basis für das Projekt diente kommerziell verfügbares elektrisches Longboard, welches mit einer Fernbedienung angesteuert wird.

Inhaltsverzeichnis

1 Einleitung

1.1 Aufgabenstellung und Zielsetzung

Ziel der Arbeit ist die Modifikation eines mittels Fernbedienung gesteuerten elektrischen angetrieben Longboards. Die Fernbedienung soll durch eine alternative Steuerungsmethode abgelöst werden, in welcher durch Gewichtsverlagerung ein Beschleunigen und Bremsen bewirkt wird. Im Mittelpunkt der Arbeit steht die Entwicklung der Software für den Mikrocontroller, welcher die Sensoren auswertet und den Motor des elektrischen Longboards ansteuert. Besonders hervorzuheben ist hierbei die Auswertung von den Wägezellen 2.3, mit welchen die Gewichtsverlagerung eines Nutzers interpretiert wird. Weitere Ziele sind die Auswahl von Sensoren, eines Mikrocontrollers und der Befestigung aller Komponenten am Longboard.

Die Durchführung dieses Projektes ist aufgrund der Thematik nicht von Beginn an festgelegt worden. Durch Experimente der ursprünglichen auf einer Fernbedienung basierenden Steuerung und später mit einem Proof-of-Concept Prototypen sollte Erfahrung gesammelt werden. Besonders ein optimales Steurungsschema und das Ansprechverhalten der Software 4.3 sollten experimentell erarbeitet werden.

Neben der Umsetzung der Steurung steht die Evaluation der Leistungsfähigkeit dieser. Die Software kann die Außenwelt nur in der Form von Sensorwerten erfassen. Eine Fehlinterpretationen von Störfaktoren als vom Nutzer explizit verwendete Gesten 2.1 dieser wird als Defizit bewertet. Um dies zu vermeiden werden verschiedene Mechanismen eingesetzt um das Signal zu filtern 4.3 und eine Fehlinterpretationen zu vermeiden.

1.2 Bestehende Ansätze

Die Idee für das Thema dieser Arbeit entstand als bei einer beiläufige Betrachtung des Marktes bemerkt wurde, dass elektrische Longboards und Skateboards zwar käuflich sind, die verwendete Steuerungsmethode dieser aber mit einer Fernbedienung arbeitet. Es stellte sich die Frage, ob es auch Produkte gebe, die eine Steuerung ohne Fernbedienung arbeiten. Nach einer gezielten Suche konnte keine kommerziellen Produkte finden, die das Kriterium erfüllen. Auch die Suche nach experimentellen Implementation durch Forscher oder Bastler blieb erfolglos.

Da die Suche nach einem Thema für Bachelorarbeit gesucht wurde, bat es sich an, an dieser Thematik zu forschen. Teil der Idee war die Vorstellung, wie eine Steurung ohne Fernbedienung 2.1 umgesetzt werden könne. Die Auswahl fiehl auf die Auswertung der Gewichtsverlagerung der Nutzer. Zu diesem Thema wurden bereits kommerziell verbreitete Fahrzeuge die über für ihre Steuerung die Gewichtsverlagerung auswerten genauer Betrachtet.

Zunächst werden der Segway Human Transporter und sogenannte Hoverboards betrachtet. Bei beiden Produkten handelt es sich um Fahrzeuge, die über nur eine Achse Verfügen. Um nicht zu kippen, muss der Motor permanent gegensteuern um das Gleichgewicht zu halten. Gleichzeitig steht der Nutzer aufrecht und kann mit einer Verlagerung des eigenen Körpers entsprechend eine Beschleunigung oder ein Bremsen erreichen. Dies steht im Kontrast zu einem auf zwei Achsen fahrenden Longboard. Durch 4 Kontaktpunkte, je zwei Rollen beider Achsen, sind selbst bei Passivität des Motors Nutzer nicht der Gefahr ausgeliefert, mit dem Fahrzeug zu kippen. Daher sind die Erkenntnisse über Hoverboards bezüglich der Balance nur wenig relevant für das in dieser Arbeit besprochene Projekt.

Abbildung 1.1: Kommerzielles Hoverboard,
Bildquelle: https://www.ptpro.de/Hoverboard-KSR-SBS-3000-schwarz-neu (Stand 19. Februar 2020)

2 Herstellen der Voraussetzungen

Dieser Abschnitt diskutiert die Entscheidungen die getroffen werden mussten, um die Entwicklung der Software und Integration der Hardware zu ermöglichen.

2.1 Festlegung des Steuerungsschemas

Bevor ermittelt werden konnte, wie eine Steuerung implementiert werden kann, musste ein Konzept für die Steuerung durch einen Nutzer aufgestellt werden. Vor Beginn des Projektes wurde überlegt, wie die Position einer auf einem Longboard stehenden Person ermittelt werden kann. Aus diesen Überlegungen stammt die Festlegung auf die verwendeten Sensorentypen und somit auch welche Gesten überhaupt vom Longboard erkannt werden können 2.3.

In der Grundposition steht Nutzer während der Fahrt mit beiden Füße auf dem Longboard. Die Füße sind dabei in einem 90 Grad Winkel zu der Länge des Longboards ausgerichtet. In dieser Position ist ein Fuß relativ zur Fahrtrichtung vor dem anderen Fuß.

Wenn Nutzer in der Grundposition stehen, ist die Gewichtsverteilung auf den vorderen und den hinteren Fuß annähernd gleich. Die Geste um eine Beschleunigung zu erreichen ist eine Verlagerung des Körpergewichtes auf das vordere Bein. Um eine Geschwindigkeit zu halten, muss die Körperposition der Anwender die Verlagerung nach vorne beibehalten. Mit Intensität der Verlagerung wird die aktuelle zu erreichende Zielgeschwindigkeit definiert. Es können also Geschwindigkeiten gehalten werden, die nicht der maximalen Geschwindigkeit entsprechen.

Bewegt sich der Nutzer zurück in die Grundposition wechselt der Motor in einen passiven Zustand. Das Longboard rollt mit der aktuellen Bewegungsenergie aus. Wenn Nutzer aufgrund eines Hindernisses die Geschwindigkeit schnell verringern wollen, muss die Geste für das aktive Bremsen ausgeführt werden. Eine Verlagerung des Gewichtes nach hinten bewirkt analog zu dem Befehl für die Beschleunigung, dass die Elektromotoren in mit starker Verlagerung zunehmender Intensität abbremsen.

Das Steuerungsschema wurde nicht ohne Abwägungen der Konsequenzen für die Umsetzung erarbeitet. Exemplarisch wird hervorgehoben, weshalb in den finalen Steuerung die Beschleunigung durch Verlagerung in die Fahrtrichtung angezeigt wird. Grund hierfür ist die Trägheit der Masse der Anwender. Bei jeder Beschleunigung und bei jeden Bremsvorgang beschleunigt nur das Board selbst. Da ein Anwender nicht fest mit dem Longboard verbunden ist, fährt dieses bei Beschleunigung unter den Füßen von dem Anwender weg. Als Resultat fällt der Nutzer leicht nach hinten, belastet also das hintere Bein mehr. Eine naive Auswertung der Sensordaten bricht also die Beschleunigung sofort ab, da der Nutzer wieder in die neutrale Position zurück fällt. Eventuell fällt der Nutzer so weit nach hinten, dass ein Bremsmanöver eingeleitet wird. Ein analoges Phänomen tritt auch bei dem Bremsen auf. Schlechte Auswertung der Messdaten führt also im schlimmsten Fall zu Stottern in hoher Frequenz während des Bremsen und der Beschleunigung. In dem Abschnitt 4.3 werden Maßnahmen diskutiert, die diese ungewollten Effekte abschwächen oder gar eliminieren.

Eine Beschleunigung bei Belastung der hinteren Hälfte des Longboards führt jedoch zu Problemen. Der durch die Beschleunigung nach Hinten fallende Körper der Nutzer verstärkt die Neigung des Nutzers. Dieser Rückkopplungseffekt verhindert, dass effektiv eine spezifische Geschwindigkeit gehalten werden kann. Im Extremfall ist die plötzliche Beschleunigung nicht mehr durch eine Gleichgewichtsverteilung des Nutzers auszugleichen und führt zum Sturz.

Ein zusätzlicher Vorteil des gewählten Steuerungsschemas zeigt sich beim Aufsteigen auf das Longboard. Setzt der Nutzer zuerst den Fuß auf die hintere Seite des Boards, so interpretiert die Steurung dies als Anweisung zum aktiven Bremsen. Um Fehlern bei der Bedienung vorzubeugen sollte hier erkannt werden, wenn unbeabsichtigt die falsche Seite des Longboards zuerst belastet

werden sollte. Problematisch ist , dass auch wenn nur ein Fuß auf der vorderen Seite des Board steht, der Hecksensor eine Gewichtsbelastung misst. In dem Abschnitt 5.2 wird diskutiert, wie das Problem behoben werden könnte.

2.2 Beschaffung eines geigneten elektrisch angetriebenen Longboards

Von zentraler Bedeutung für die Erfüllung der Aufgabenstellung ist, dass die benutzte Hardware die Möglichkeit bietet ohne größeren Aufwand modifiziert zu werden. Im Falle des Longbards sind die zwei essentiellen Kriterien hierbei der Zugang zur Motorsteuerung und die Möglichkeit zur Aufhängung von Sensoren, wie in 2.3 beschrieben. Bei vielen der kommerziell verfügbaren elektrischen Boards handelte es sich vollständig montierte Systeme. Es war daher nicht klar, ob ein direkter Zugriff auf die Motorsteuerung möglich ist. Die Wahl fiehl auf ein vielversprechendes Bausatzsystem, siehe Abbildung 2.1. Mit den Bildern auf der Website des Händlers war erkennbar, dass ein Steuerungmodul physisch zugänglich war. Nach Erhalt des Produktes stellte sich heraus, dass der Zugriff nicht wie geplant durchgeführt werden konnte. Es wurde erwartet, dass eine Komponente das Funksignal der Fernbedienung verarbeitet und danach ein Signal an einen sogenannten Electric Speed Controller weitergibt. Ein Electric Speed Controller (ESC) ist ein Chip zur Steuerung der als Eingabe ein Signal erhält und Anhand diesen den Strom zu einem Elektromotor steuert. Als Eingabe nutzten hierbei viele ESC Pulsweitenmodulation (PWM).

Abbildung 2.1: Das ausgewählte Longboard.

PWM ist eine Technik, bei welcher ein Signal codiert wird indem zwischen zwei diskreten Spannungswerten gewechselt wird. Je Periode fester Länge findet je ein Wechsel zwischen hohen Pegel

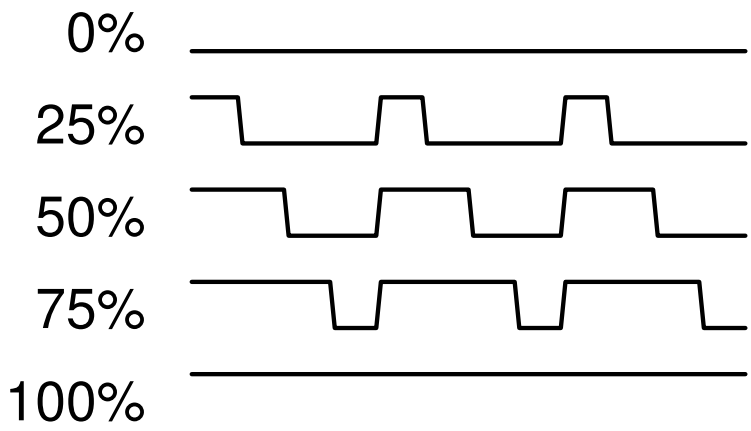


Abbildung 2.2: Darstellung verschiedener PWM-Signale, wobei der Prozentsatz die Verweildauer in dem oberen Spannungsniveau einer Periode angibt.

und niederen Pegel und umgekehrt statt. Das Verhältnis zwischen der Verweildauer im Hohen und niederen Pegel codiert das eigentliche Signal, während die Summe der beiden Verweilzeiten der Länge der Periode entspricht [6].

Viele Mikrocontroller verfügen über zahlreiche Pins mit der Fähigkeit PWM-Signale zu produzieren. So erhalten die auf zwei Rückgabewerte beschränkten digitalen Pin die Möglichkeit eine größere Menge voneinander unterscheidbarer Signale zu produzieren.

So hätte sich ein Mikrocontroller mit Unterstützung dieser sich direkt zwischen Funkverarbeitung und ESC schalten können und so direkt den Motor ohne Fernbedienung steuern können. In der Praxis war dies aber nicht möglich, da bei diesem Produkt Funkverarbeitung und ESC direkt auf einer Platine integriert sind. Die Größe der verwendeten SMD Bausteine auf der Platine erschweren ein manuelles Suchen nach und Löten an relevanten Komponenten. Der Aufwand würde den Rahmen dieser Arbeit übersteigen und das Risiko beinhalten, die Platine des Longboards bei dem Versuch der Erweiterung zu beschädigen.

Die alternative Methode lässt die Motorsteuerung des Boards unverändert und steuert das Longboard über eine modifizierte Fernbedienung. Während des normalen Betriebs ist das Potentiometer mit einem Schieberegler verbunden. Die von diesem ausgegebene Spannung zwischen 0V und 3,3V wird von der Fernbedienung gelesen und als Beschleunigungs- oder Bremssignal an das Longboard übertragen. Nach der Modifikation der Fernbedienung wird diese Eingabespannung nicht mehr durch ein Potentiometer erzeugt, sondern durch den analogen Ausgabepin des Mikrocontrollers. Da in dieser Konfiguration die Fernbedienung mit dem Mikrocontroller verschaltet werden muss, ist diese in dem selben Gehäuse wie der Controller untergebracht 3.2.

2.3 Auswahl der Sensoren

Um die Lageverschiebung der Nutzer zu messen sind verschiedene Sensorentypen ausgewählt worden. Die Basis bilden hierbei zwei sogenannte Wägezellen. Bei der Einwirkung von Kraft verformen diese. Diese Verformung kann elektrisch ausgewertet werden und so die einwirkende Kraft ermittelt werden. Eine Montage an der Vorder- und Hinterachse ermöglicht so, die Gewichtsbelastung auf das Board zu messen und somit auf die Position der Nutzer zu schließen. Die Bauform der Sensoren wird in 3.1 diskutiert. Zusätzlich zu der Form müssen die Sensoren in der Lage sein, den Kräften denen sie ausgesetzt sind standzuhalten. Durch Stöße und andere Beschleunigungen wirkt in Randfällen ein vielfaches der vom Nutzer ausgehende Gewichtskraft auf das Longboard. Relevant ist deshalb die zu erwartene dynamische Last die maximal auf eine Achse wirken kann. 200Kg je Sensor wurden als außreichend befunden, da durch die Montage die Last auf beide Sensoren verteilt wird.

Unter Beachtung der gewählten Bauform und der maximalen dynamischen Last wurde nach entsprechenden Sensoren gesucht. Zunächst wurde ein Modell bestellt, welches alle Kriterien erfüllt.

Da die Lieferung aus China mehrere Wochen dauerte, wurden Sensoren mit ähnlicher Form aus Deutschland bestellt, um die Abschlussarbeit nicht zu verzögern. Die aus Deutschland bestellten Sensoren sind TAS606 [9] Wägezellen und sind die aktuell im Board verbauten Sensoren.

Der Messmechanismus von Wägezellen basiert darauf, die Verformung der Zelle mit einem elektrischen Wiederstands messbar zu machen [1]. Auf einer Wägezellen sind feine Leiterbahnen aufgebracht, die bei Verformung der Zelle in der Länge gestreckt oder gestaucht werden. Eine Verlängerung bewirkt hierbei, dass der Leiter länger und gleichzeitig dünner wird. So nimmt der elektrische Widerstand zu. Analog bewirkt eine Stauchung eine Abnahme des elektrischen Widerstandes. Da die durch die Verformung erreichte Veränderung des Wiederstandes äußerst gering sind, muss eine Technologie verwendet werden, mit welcher diese geringe Veränderung quantifiziert werden kann. Mithilfe einer wheatstoneschen Messbrücke können die Größenverhältnisse zwischen Widerständen als Spannung wiedergegeben werden. Durch die Darstellung des Verhältnis kann durch Wahl eines genügend kleinen Vergleichswiderstandes die Veränderung dennoch wahrgenommen werden.

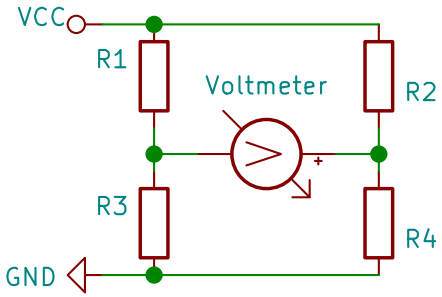


Abbildung 2.3: Schaltplan einer wheatstonschen Messbrücke

Die Wheatstonesche Messbrücke verschaltet vier Wiederstände wie in der Grafik 2.3. Durch die zwei parallelen Leiter hat der Strom zwei mögliche Wege zu fließen. Man spricht von einer einer balancierten Wheatstonschen Brücke, wenn die Verhältnisse der Wiederstände in den parallelen Pfaden identisch ist. In diesem Fall gilt $R_1 \div R_3 = R_2 \div R_4$. Die vom Voltmeter gemessene Spannung ist hier 0.

Wenn nun die Widerstandswerte von drei der Wiederstände bekannt sind, kann anhand der gemessenen Spannung der Wert des vierten Wiederstands ermittelt werden. Durch Einsetzen der bekannten Werte in die Gleichung 2.1 kann diese für den Wert des unbekannten Wiederstandes gelöst werden [5][4].

$$V_{voltmeter} = (\frac{R_3}{R_3 + R_1} - \frac{R_4}{R_4 + R_2}) \cdot VCC \tag{2.1}$$

Da die Verformung der aus Stahl gefertigten Wägezellen minimal ist, ist auch die Veränderung des Wiederstandes äußerst gering. Um die daraus resultierende Spannung zu messen wird daher ein geeigneter Messverstärker benötigt. Während der Suche nach passender Hardware wurden Mitglieder aus der RoboCup AG der Fakultät Informatik befragt, da in deren humanoiden Robotern ebenfalls Wägezellen verbaut sind. Es wurde eine vorgefertigte Platine zu Verfügung gestellt, welche für den Gebrauch in Robotern entwickelt wurde. Da diese über genügend Steckverbindungen und einen aufgelöteten Messverstärkerchip verfügt, wurde diese unverändert übernommen. für der Zwecke dieses Projekt verfügt. In diesem Projekt wird daher ein ADS1262 verwendet.

Durch die Messung der Beschleunigung kann zudem gemessen werden, welche Kräfte auf die Nutzer wirken müssen. Die verfälschten Messdaten bezüglich Gewichtsverteilung können mit diesem Wissen besser verarbeitet werden. Ein hierfür eingesetzter Beschleunigungssensor sollte zudem in der Lage sein, die Erdbeschleunigung zu messen. Der Vektor der Erdbeschleunigungs kann genutzt werden, um die Neigung des Untergrundes auf welchen das Board fährt zu ermitteln. Auf der Platine des ausgewählten Mikrocontrollers ist ein LSM6DS3-Sensor verbaut, welcher die genannten Anforderungen erfüllt.

2.4 Auswahl des Mikrocontrollers

Der Mikrocontroller muss viele Anforderungen erfüllen. Zur Ansteurung des Motor über die Fernbedienung sollte der Controller etwa ein Potentiometer emulieren können. Hierfür benötigt der Controller die Möglichkeit, echte analoge Spannungen auszugeben. Zudem werden genügend serielle Schnittstellen benötigt, damit alle Komponenten mit dem Controller verbunden werden können.

Beim der Entwicklung und Optimierung von den Filterungsalgorithmen 4.3 wird eine Ausgabe der Sensordaten in Echtzeit benötigt. Eine Kommunikation über ein USB-Kabel ist hierbei nicht praktikabel, da sonst während der Fahrt ein Laptop mit dem Controller verbunden sein müsste. Die Kommunikation muss also kabellos erfolgen.

Da keine Verbindung zu einem Computer besteht, muss ein Betrieb über eine portable Stromquelle muss ebenfalls möglich sein.

Alle Kriterien werden durch den Arduino Nano 33 IOT erfüllt. Auf dem Controller-Board sind ein WiFi-Modul und IMU-Sensoren integriert und mit dem Controller verbunden. Eine weitere SPI-Schnittstelle ist auf Logik-Pins extern ansprechbar und kann so für die Verbindung zum Messverstärker der Wägezellen genutzt werden. Dennoch ist der Controller kompakt genug um unter dem Longboard montiert zu werden.

2.5 Stromversorgung

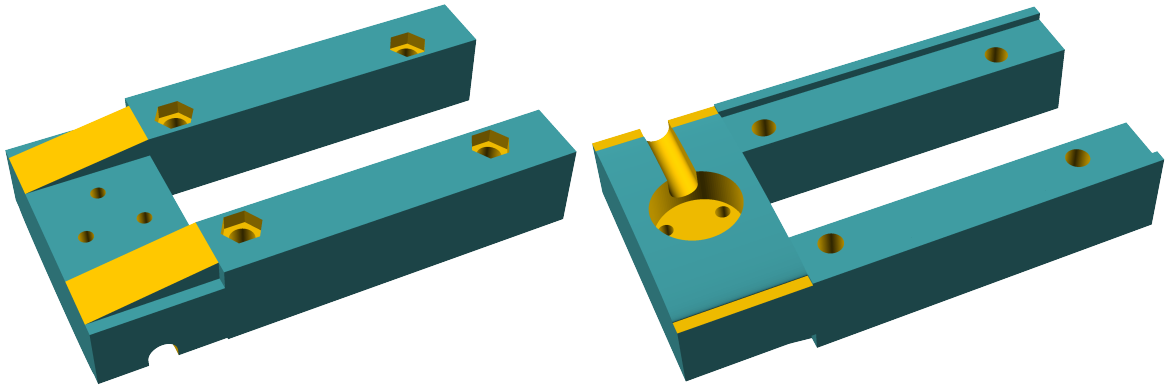
Die Stromversorgung für den Mikrocontroller und die Messverstärkerplatine wird über einen zusätzlichen USB-Akku bezogen. Diese Lösung ist trotz einer zusätzlichen Komponente in der Implementation trivial. Alternativ den Akku des Longboards mittels eines Spannungswandlers für den Mikrocontroller nutzbar zu machen ist im Vergleich technisch zu aufwendig.

3 Montage der Hardware

In diesem Kapitel wird diskutiert wie die verschiedenen Bauteile am Board montiert wurden und der Mikrocontroller mit allen relevanten Komponenten verschaltet ist.

3.1 Entwicklung der Sensorenhalterung

Aufgrund der Funktionsweise von Wägezellen ist es essentiell, dass die Sensoren in einer Position montiert werden, in welcher diese einen möglichst großen Teil der Gewichtskraft ausgesetzt sind. Die Entscheidung fiehl deshalb auf die Montage jeweils einer Wägezelle an jeder der beiden Achsen. Eine Montage eines Sensors zwischen Achsenaufhängung und Brett des Longboards wurde ausgeschlossen. Grund hierfür war, dass die Bauhöhe möglichst gering ausfallen soll.

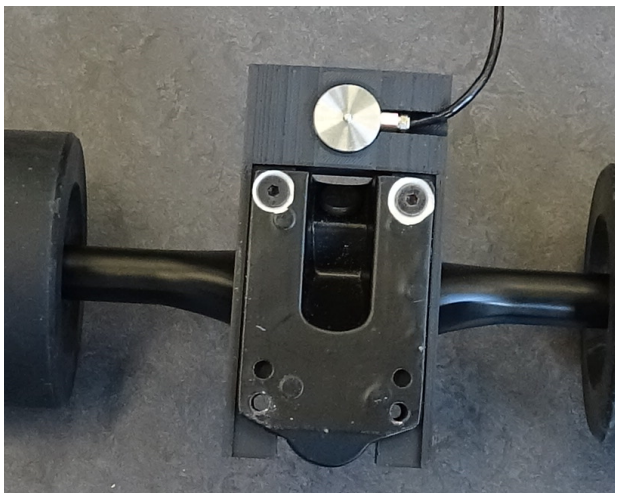


(a) (b)

Abbildung 3.1: Modell der Halterung für Wägezellen. Ansicht (b) zeigt die Fläche mit Kontakt zum Longboard oben

Stattdessen wird der Sensor neben der Achse montiert. Hierbei werden nur 2 der 4 Schrauben der Achsenhalterung mit dem Longboard verbunden. Dabei liegen die montieren Schrauben entweder beide vorne oder hinten. Bei der vorderen Achse liegen auch die Schrauben in den vorderen Löchern, während die hintere Achse analog mit den hinteren Löchern im Board verschraubt ist 3.2(b). Zwischen Achse und Board liegt eine auf den Schrauben aufgesteckte Metallplatte. Dies erwirkt einen Höhenunterschied zwischen der Seite mit und ohne Metallplatte. Durch Höhenunterschied und einseitiger Verschraubung kippt die Achsenhalterung bei Belastung ab. Die Kraftübertragung auf den Sensor erfolgt mittels einer speziellen Sensorhalterung 3.1. Diese wird von der Longboard abgewandten Seite aufgesteckt und verlängert so die Scherbewegung. In dem über die Achse hinausragenden Block befindet sich auf der dem Brett zugewandten Seite eine Aussparung. Diese fasst die Wägezelle, welche so auf das Brett gedrückt wird. Damit sich die aus Stahl gefertigte Wägezelle nicht in das weiche Holz des Bretts eindrückt, ist unter dem Sensor eine dünne Metallplatte befestigt. In der Abbildung 3.3 ist erkennbar wie der Pin auf der Oberfläche auf die Metallplatte drückt.

Während der Entwicklung der Software werden diese Sensorenhalterungen mithilfe eines 3D-Druckers gefertigt. Nach Abschluss des Projektes wäre es denkbar die Halterung aus einem weniger elastischen Material zu fertigen. So würde die Komponente weniger wie eine Feder wirken und die Messung weniger verfälschen.



(a) Sensor in Halterung eingefasst

(b) Sensorhalterung, montiert

Abbildung 3.2: Auf die Achse aufgeschraubte Sensorhalterung

3.2 Mikrocontroller und Gehäuse

Die Unterseite des Longboards bietet für alle elektrischen Komponenten genügend Platz, würde diese aber bei Fahrten unter freien Himmel der Witterung aussetzen. Daher werden diese innerhalb eines mit einem 3D-Drucker gefertigten Gehäuses 3.4 montiert.

Die im oder an dem Gehäuse Montierten Komponenten sind:

- Mikrocontroller
- Messverstärker-Platine für die Wägezellen
- USB-Akku
- Platine der Fernbedienung für das Longboard
- Breakout-Board
- 3 LED-Lampen
- 3 Taster
- Kippschalter

Zentraler Bestandteil der Installation ist ein speziell angefertigtes Breakout-Board. Auf diesem wird der Mikrocontroller in einen Sockel gefasst und so mit den anderen Komponenten verbunden. Das Breakout-Board selbst wird auf das im Boden des Gehäuses verschraubten Messverstärker-Boards aufgesteckt. Die Grafik 3.5 zeigt, welche Verbindungen durch das Breakout-Board realisiert werden.

3.2.1 Fernbedienung

Da die Fernbedienung darauf ausgelegt ist, während der Benutzung in der Hand des Nutzers gehalten zu werden, wurden für zum Betrieb innerhalb des Gehäuses einige Anpassungen vorgenommen. Von der Fernbedienung selbst wurde nur die Platine verbaut, während die Plastikhülle und das Potentiometer entfernt wurden. Letzteres wurde durch ein Kabelverbindung zum Breakout-Board ersetzt und ermöglicht so die Steuerung durch den Mikrocontroller. Um überwachen zu können, ob die Fernbedienung eingenschaltet und mit dem Board verbunden ist, besitzt die Fernbedienung LEDs. Deshalb besitzt das Gehäuse im Boden eine Öffnung, welche mit einem Stück durchsichtigen Plastik vor Schmutz geschützt ist. Die Platine wurde so orientiert und im Gehäuse verschraubt, dass die alle

Abbildung 3.3: Seitliche Ansicht der montierten Sensorhalterung.

LEDs von Außen sichtbar sind. Seitlich am Gehäuse sind 2 Taster befestigt, welche direkt mit den Tastern auf der Fernbedienung verbunden sind. Da die Fernbedienung auf kabellosen Betrieb ausgelegt ist und über einen Akku betrieben wird, muss regelmäßig ein Ladevorgang angestoßen werden. Der Kippschalter an der Außenseite des Gehäuses verbindet die MicroUSB-Buchse der Fernbedienung mit der 5V Spannungsquelle des Arduino Mikrocontrollers und ermöglicht so das Laden.

3.2.2 Messverstärkerplatine

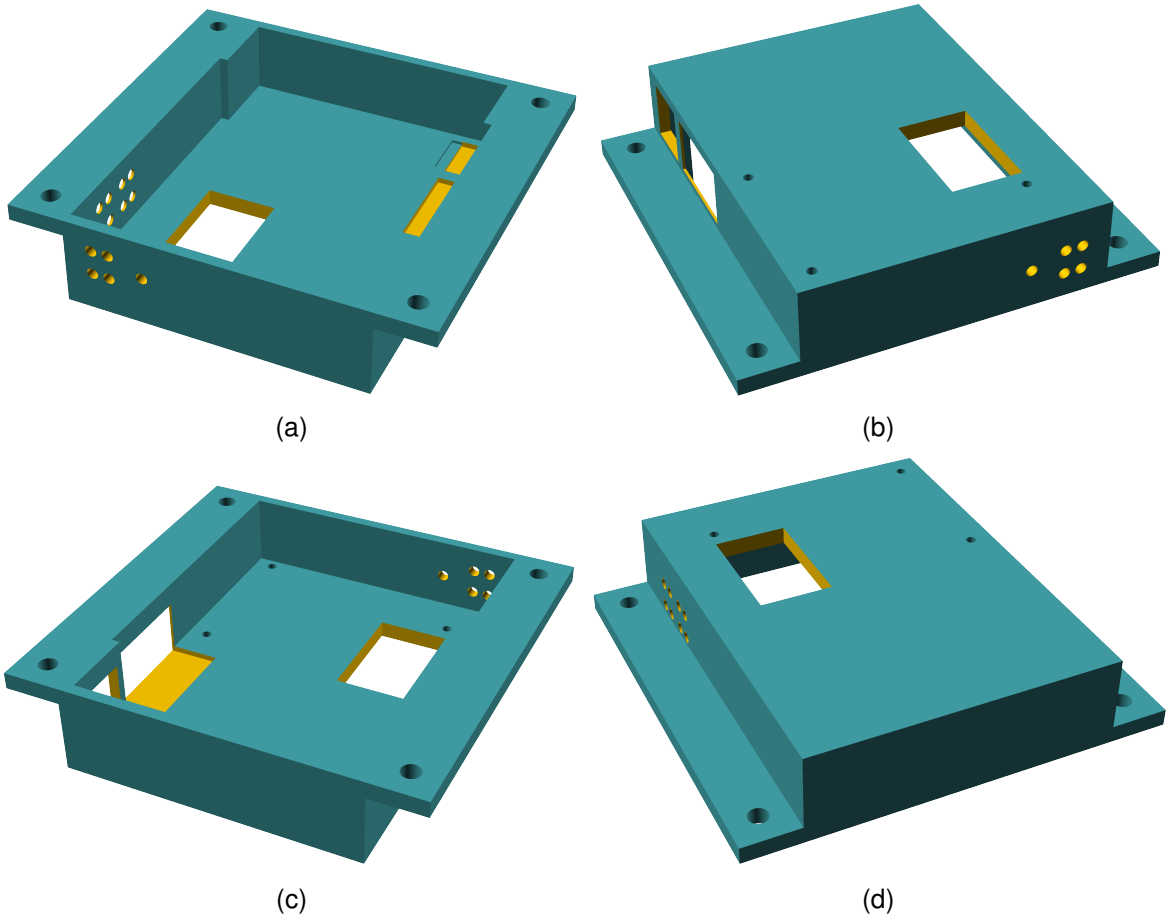
Die Messverstärkerplatine dient als Ankerpunkt für das Breakout-Board, welches mithilfe von 2 Pfostenleisten aufgesteckt wird. Um den ADS1262 mit Strom zu versorgen musste abgewogen werden, woher dieser bezogen werden kann. Das Design des Messverstärkerboard sieht vor, dass eine 12V Eingangsspannung angelegt werden muss. Da der verwendeten Mikrocontroller keine Möglichkeit bietet eine solche Spannung zu liefern, wurde eine Alternative angewandt. Der Arduino Nano 33 IOT besitzt während des Betriebs über USB die Fähigkeit die Versorgungsspannung weiterzuleiten. Die simpelste Lösung war daher, die 5V Versorgungsspannung und Masse hinter den Spannungswandler des Messverstärkerboards anzuschliessen. Über eine Anschlussbuchse für eine Wägezelle war es möglich die Spannungswandler zu umgehen. Die Kabel für die Wägezellen werden über eine Öffnung in der Seite in das Gehäuse geführt.

3.2.3 USB-Akku

Der Akku zur Stromversorgung des Mikrocontrollers ist so orientiert, dass die Anschlüsse durch eine Ausparung im Gehäuse von Außen zugänglich sind. Durch ein kurzes Kabel und einer zweiten Aussparung wird der Akku direkt mit dem Mikrocontroller verbunden.

3.2.4 Taster

Insgesamt sind auf der äußeren Seite des Gehäuses 3 Taster befestigt. Zwei von diesem dienen zur Steuerung der Fernbedienung, während der Dritte den den Reset-Pin des Mikrocontrollers kontrolliert. Die Kabel der Taster wird über einen gemeinsamen Stecker auf das Breakout-Board aufgesteckt, um die Möglichkeit zu bewahren, die Komponenten zu einem späteren Zeitpunkt zu trennen.



(a) (b)

(c) (d)

Abbildung 3.4: Modell des Gehäuses für Mikrocontroller, Fernbedienung und Messverstärker. Ansicht (a) zeigt die Fläche mit Kontakt zum Longboard oben.

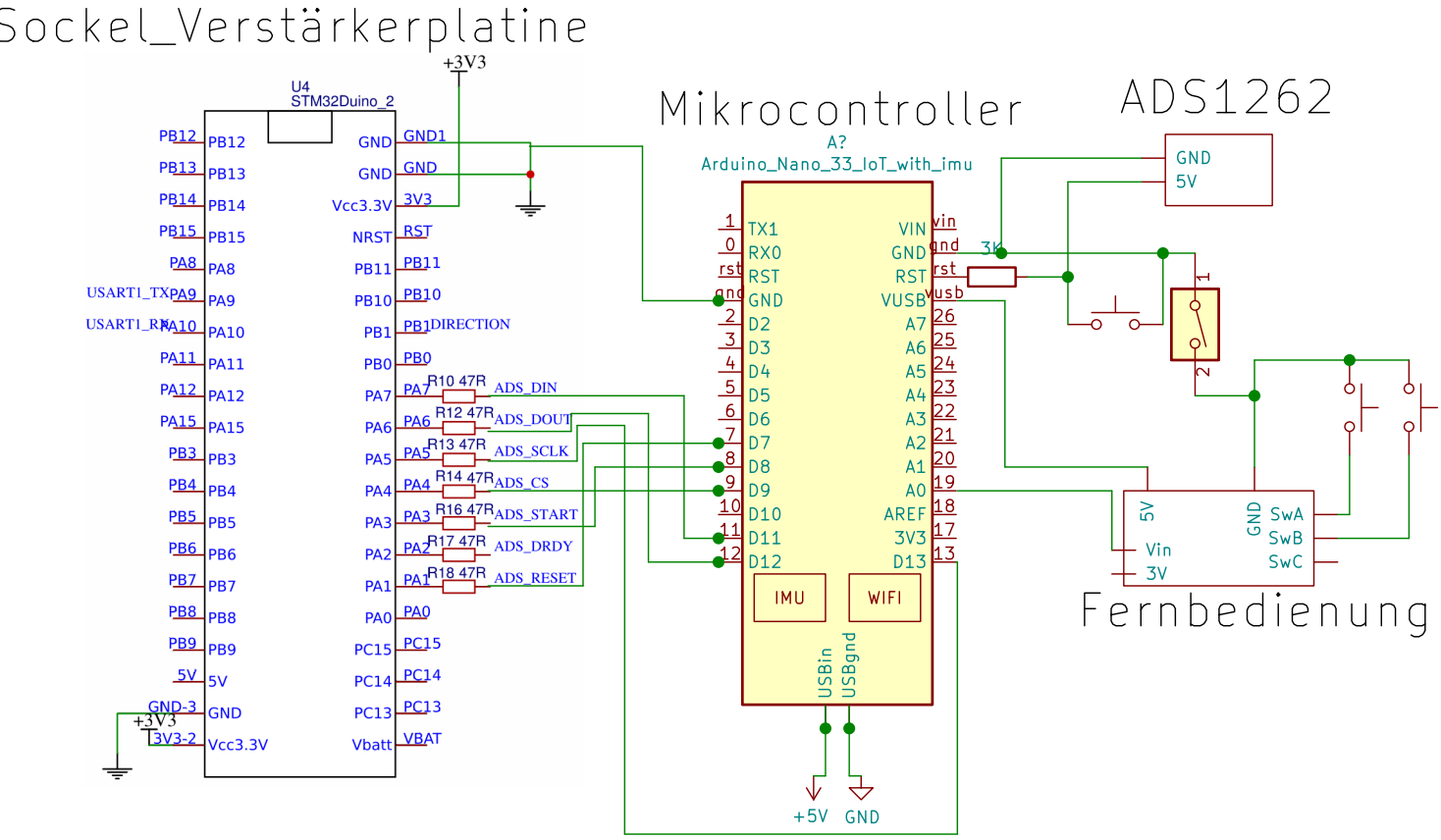


Abbildung 3.5: Verschaltung von Mikrocontroller, Messverstärker und der Fernbedienung

4 Entwicklung der Software

Dieses Kapitel beschreibt die im Rahmen dieser Arbeit entwickelten Softwarekomponenten. Die Softwareartefakte sind sowohl Routinen für den Mikrocontroller als auch Programme die auf einem unterstützenden Computer, wie einen Laptop, ausgeführt werden. Im Kern stehen die Komponenten die auf dem Mikrocontroller laufen und das Longboard steuern. Zudem wurden zur Entwicklung und Optimierung Werkzeuge entwickelt.

4.1 Auswertung der Sensoren

Wägezellen In diesem Projekt werden verschiedene Typen von Sensoren mit dem Mikrocontroller kommunizieren. Die beiden verwendeten Sensortypen werden zudem nicht direkt verarbeitet. Für die Wägezellen muss aufgrund des Messmechanismus ein Signalverstärker benutzt werden. Dieses Projekt nutzt eine Platine der RoboCup-AG 2.3, welche auf einer Schaltplatine einen ADS1262 Signalverstärker mit Buchsen für die Sensoren verbindet. Insgesamt 6 digitale Pins werden vom Arduino verwendet, um das Messverstärkerboard anzusteuern. Drei dieser Pins übernehmen dabei die Rolle von MISO, MOSI und SCK für eine SPI-Kommunikation. Deren Position kann daher nicht frei gewählt werden, da alle an den entsprechenden Pins eines gemeinsamen SERCOM liegen müssen. Ein SERCOM ist eine Komponente des Mikrocontrollers, welche die Kommunikation über verschiedene serielle Protokolle ermöglichen. Jeder SERCOM des Arduino Nano 33 IOT ist nur über spezifische Pins ansteuerbar [2]. Die restlichen Pins werden in der für dieses Projekt verwendeten Konfiguration zunächst in der Initialisierungsphase verwendet. Der Chip wird zunächst mittels eines Reset-Pins in einem deterministischen Zustand gebracht und danach mit dem Start-Pin initialisiert. Der ADS1262 wird mit ADS162X Bibliothek angesteuert. Die Bibliothek abstrahiert den Austausch von SPI-Nachrichten zwischen Mikrocontroller und Messverstärker auf eine Reihe von Funktionen. Kommunikation mit dem Chip mittels SPI wird durch eine Bibliothek übernommen. Als Konfigurationsparameter werden der Start- und der ChipSelect-Pin übergeben. Nach der Konfiguration der Bibliothek kann begonnen werden die Sensoren auszulesen. Durch die Abstraktion der Bibliothek wird wird verschleiert, dass der Signalverstärkerchip nicht in der Lage ist mehrere Sensoren gleichzeitig zu lesen. Um eine andere Wägezelle zu messen, muss der Chip mittels SPI angesteuert werden und dazu veranlasst werden, den internen Multiplexer in den angeforderten Zustand zu versetzen. Real dauert das Umschalten des Multiplexers gut eine Millisekunde und Messwerte die innerhalb dieses Zeitraums erhoben werden sind verfälscht. Da der Anwendungsfall dieses Projektes erfordert kontinuierlich die beiden Wägezellen miteinander zu vergleichen, muss darauf geachtet werden, ein Zeitlichen Sicherheitsabstand zu halten und vorherige Ergebnisse zu verwerfen. In der Implementation löst eine Funktion das Problem. Exemplarisch die Arbeitsweise von sensor_read_front() 4.2. Der erste Aufruf von *adc.readADC1()* dient lediglich dazu, dass die Bibliothek den Multiplexer des Verstärkers modifiziert. Nach dem Verstreichen der Wartezeit ergibt ein zweites Lesen einen korrekten Wert.

Beschleunigungs- und Drehratensensor Auf dem Arduino Nano 33 IOT ist LSM6DS3 verbaut. Mit diesem Sensor können die Beschleunigung in drei Achsen und mittels eines Gyroskops Rotationsbewegungen messen. Angesteuert wird der LSM6DS3 mittels der gleichnamigen Bibliothek über ein I2C-Interface.

Abbildung 4.1: Auszug aus Longboard-Steuerung: Funktion zum auslesen der vorderen Wägezelle

```

int32_t sensor_read_front() {
  delayMicroseconds(DELAY_ADC_READ);

  adc.readADC1(pos_pin[PIN_LOAD_CELL_FRONT],
               neg_pin[PIN_LOAD_CELL_FRONT]);

  delayMicroseconds(DELAY_ADC_READ);

  return adc.readADC1(pos_pin[PIN_LOAD_CELL_FRONT],
                      neg_pin[PIN_LOAD_CELL_FRONT]);
}
```

Abbildung 4.2: Beispiel zur Ansteurung der IMU

```
#include <Arduino_LSM6DS3.h>

setup() {
  IMU.begin();
}

loop() {
  float x, y, z;

  if (IMU.accelerationAvailable()) {
    IMU.readAcceleration(x, y, z);

    /* Arbeiten mit den Beschleunigungswerten */
  }

  if (IMU.gyroscopeAvailable()) {
    IMU.readGyroscope(x, y, z);
    /* Arbeiten mit der Rotationsgeschwindigkeit */
  }
}
```

4.2 Interaktion mit dem Motor

Zur Interaktion mit der Motorsteuerung des Longboards wird eine mit diesem gekoppelte Fernbedienung genutzt. Der Mikrocontroller ist in der Lage die Funktion des Potetiometers der Fernbedienung zu emulieren, indem dieser eine analoge Spannung ausgibt. Durch Messungen an der Fernbedienung konnte der Werteberreich des Potentiometers ermittelt werden. Spannungen zwischen 0V und 1,65V bewirken, dass der Motor aktiv bremst. Die Intensität steigt hierbei mit der Annäherung der Spannung an 0V. Werte ab 1,65V bis zu 3,3V bewirken analog ein Beschleunigen des Motors. Um 1,65V befindet sich der Leerlauf, in welchem das Longboard mit der aktuellen kinetischen Energie ausrollt.

Die Fernbedienung des Longboards verfügt über eine Gangschaltung mit 4 Stufen. Ein höherer Gang bewirkt eine entsprechende Steigerung der Maximalgeschwindigkeit. Für einen spezifischen Gang wird der gesammte Werteberreich des Potetiometers genutzt. So ist beispielhaft bei der geringsten Stufe die Maximalgeschwindigkeit erst erreicht, wenn der Spannungswert 3,3V beträgt.

Der Arduino Nano 33 IOT unterstützt die Ausgabe analoger Spannungswerte nur auf dem Pin *A0*. Um den Digital-Analog-Konverter dieses Pins zu nutzen darf dieser nicht mit *pinMode* initialisiert werden. In der Implementation wird der aktuelle Motorzustand als 10 Bit Integer ohne Vorzeichen modelliert. Der Wert 1023 entspricht so 3,3V und 0 übersetzt zu 0V.

```
1 setup() {
2   analogWriteResolution(10);
3 }
4
5 loop() {
6   /* Versetze den Motor in den Leerlauf */
7   analogWrite(A0, 512);
8 }
```

Die Kommunikation mit dem Motor des Longboards erfolgt unilateral 4.5.1. Für den Mikrocontroller besteht keine Möglichkeit die Geschwindigkeit oder Drehzahl der Motoren zu messen. Da die Software zur Evaluation der aktuell zu fahrenden Geschwindigkeit versucht zu starke Beschleunigungen zu vermeiden, wird ein Verlauf der zuletzt dem Motor übermittelten Geschwindigkeiten gespeichert.

Aufgrund des fehlenden Feedbacks durch den Motor besteht eine Diskrepanz zwischen dem in Software modellierten Zustand des Motors und er Realität. Dies ist der Fall, wenn der Controller einem neuen Geschwindigkeitswert auf den Ausgabepin schreibt, welcher stark vom Vorausgegangenen abweicht. Der Übergang vom Leerlauf (512) zu maximaler Motorkraft (1023) in der Software ist unmittelbar, dauert jedoch in der Realität einige Millisekunden.

4.3 Filterungsalgorithmen

Die Messungen der Wägezellen sind zunächst Rohdaten. In Abbildung 4.3 ist eine Messreihe der Rohdaten dargestellt, in welcher das Longboard wird montierten Sensoren auf einer Werkbank liegt. Um den Motor ansteuern zu können müssen zunächst technische Störfaktoren, wie u.A. Rauschen, entfernt werden und das Verhalten des Menschen anhand der Messdaten gedeutet werden. Dieser Abschnitt diskutiert die implementierten Filterungsverfahren die das rohe Signal aufarbeiten.

4.3.1 Glätten der Messdaten

Die in der Abbildung 4.3 zeigt das Signal der Wägezellen ohne äußere Einwirkung. Eine genauere Betrachtung offenbart, dass die Messungen sowohl zufällig wirkendes Rauschen als auch oszillierende Pulse enthält.

Für das Rauschen kommen verschiedene Störquellen in Frage. Es ist zu erwarten, dass die Sensoren selber und auch die Verbindungsdrähte als Antenne wirken. So würden Funkübertragungen unbekannter Frequenz als Signal in der Messung auftauchen. Der größte Teil des Drahtes welcher von Sensor zum Messverstärker führt ist speziell gegen solche Wechselwirkungen abgeschirmt. Doch der Teil nahe des Steckers für das Messverstärkerboard besitzt keine Abschirmung.

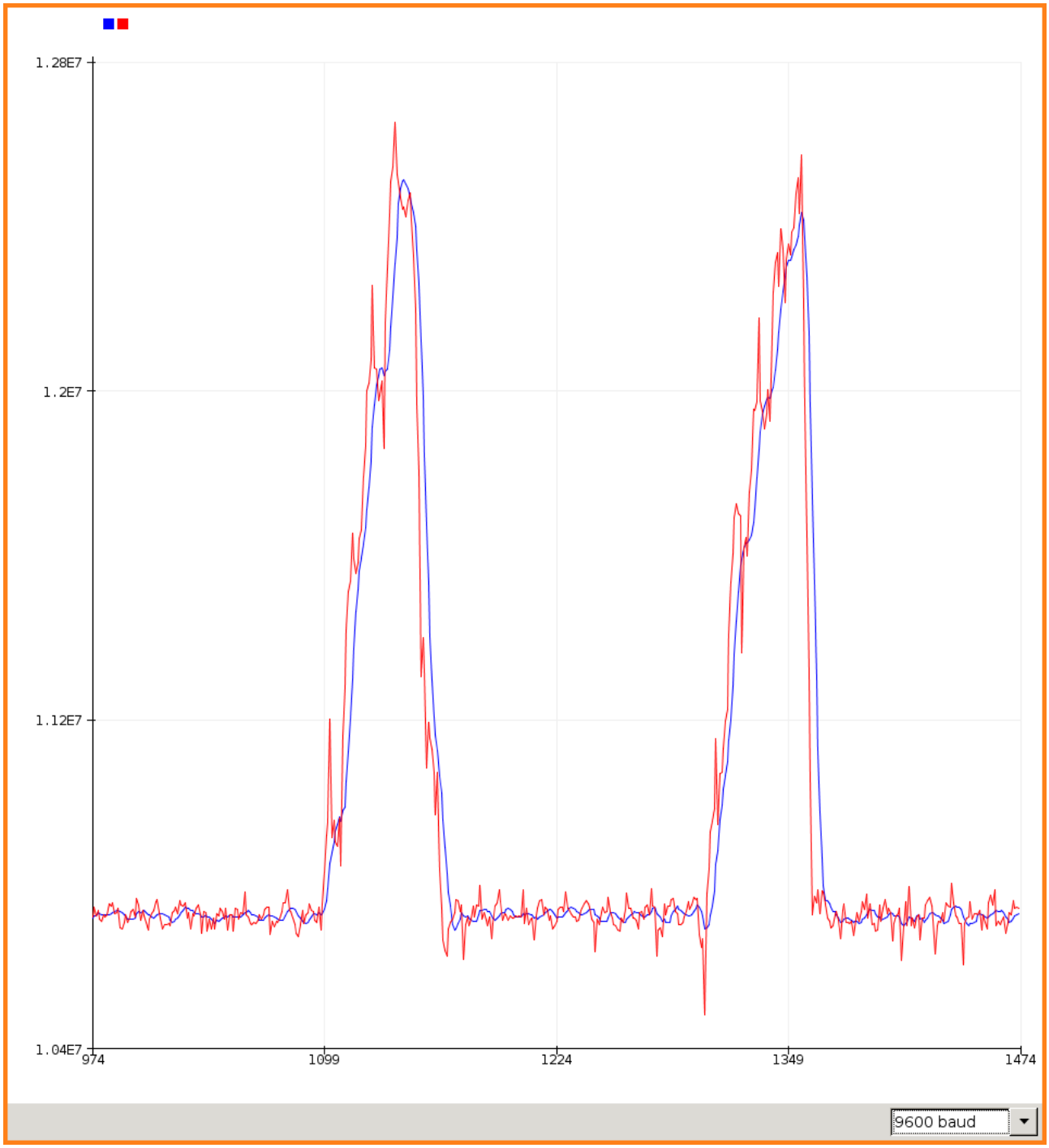


Abbildung 4.3: Rohe Messdaten ohne Filterung (rot) im Vergleich mit der Filterung mittels einfachen Durchschnittswert (blau) über die letzten 8 Messungen.

Der Ursprung der Oszillation scheint auch durch äußere Einflüsse verursacht. Allerdings können hierbei nur Quellen berücksichtigt werden, welche sehr regulär Arbeiten. Ein Beispiel ist der Einfluss elektrische Geräte in der Umgebung. Exemplarisch könnte das Stromnetz mit einer Frequenz von 50 Hertz in einer Messung erscheinen.

Auch der Einfluss einer elektrischen Verbindung zu einem Laptop wurde analysiert. Hierfür wurde das ungefilterte Signal der hinteren Wägezelle zwei Mal aufgezeichnet. Im ersten Durchlauf wurde der Mikrocontroller über den Laptop mit Strom versorgt, während im Zweiten der Controller mit einem USB-Akku verbunden wurde. Die Abbildungen 4.5 und 4.4 zeigen die Messungen über 13 Sekunden mit dem Beginn der Y-Achse auf 0. Für ein besseres Verständnis zeigen die Abbildungen 4.7 und 4.6 die selben Messungen mit einer angepassten Y-Achse. In den Grafiken ist erkennbar, dass es einen Unterschied in der Intensität des Rauschens gibt. Bei Stromversorgung durch den Laptops ist das Signal deutlich weniger gestört. Die Mögliche Ursache könnte eine stabilere Spannungsversorgung durch den Laptop sein.

Um den Einfluss der genannten Störfaktoren möglichst gering zu halten werden die Sensordaten gefiltert. Der Ansatz basiert hierbei auf der Datenerhebung über einen Zeitraum hinweg mit der Bildung eines geigneten Mittelwerts.

Eine Variable bei diesem Ansatz zur Filterung ist die Länge des beobachteten Zeitraums. Mit der Abbildung 4.8 wird ersichtlich, dass ein längerer Zeitraum Rauschen und Oszillationen effektiver filtert. Jedoch führt ein langes Beobachtungsintervall zu einer deutlichen Verzögerung von zwischen Eingabe eines Nutzers und der Wiederspiegelung dieser in dem gefilterten Signal. Dies hat Auswirkungen auf Fahrkomfort und Sicherheit. Bei dem Versuch zu beschleunigen wird die Gewichtsbelastung der Frontseite erhöht, bis eine Reaktion in gewünschter Intensität erreicht ist. Durch einen zu langen Zeitraum der Filterung wirkt der Motor erst nachdem eine ausreichende Belastung erzielt wurde. Nutzer würden in dem Zeitraum der Verzögerung bereits die Belastung erhöht haben und so eine unerwartet starke Beschleunigung auslösen. Praktische Experimente zeigen, dass es bei zu großer Verzögerung schwierig ist, eine konstante Geschwindigkeit zu halten. Den Zeitraum, über welchen der Durchschnittswert gebildet wird, lässt sich berechnen, indem gemessen wird, wie lange die Controllersoftware benötigt, um die Hauptschleife einmal zu durchlaufen. Praktische Erfahrung zeigt, dass die durchgeführten Berechnungen aber auch das Senden von den Messdaten einen minimalen Anteil an der Durchlaufzeit hatten. Vielmehr nimmt die Messung der Wägezellen einen Großteil der Durchlaufzeit in Anspruch. Wie in dem Abschnitt 4.1 erläutert muss während der Messung auf die Hardware des Messverstärkers gewartet werden. Je Schleifendurchlauf ergeben sich so etwas mehr als 4 mal eine Millisekunde. Eine Durchschnittsbildung über 16 Messwerte beachtet also Daten innerhalb der letzten 64 Millisekunden.

Die durch Nutzer wahrgenommene Verzögerung wird in dem Abschnitt 5 diskutiert.

Ein Mechanismus um das Ansprechverhalten der Steuerung zu verbessern ist es, Messwerte innerhalb des Filterungszeitraums zu gewichten. Aktuellere Werte haben hierbei ein höheres Gewicht als Alte und haben so schnell Einfluss auf die Steuerung. In der Grafik 4.9 werden der ungewichtete Durchschnitt und mit dem Verfahren Gewichtung neuerer Werte gegenübergestellt. Durch Experimente kann hierbei ermittelt werden, welche Gewichtung die besten Ergebnisse liefert.

4.3.2 Abstraktion der Sensorwerte

Nach dem Glätten 4.3.1 liegen die Messdaten noch immer in einer absoluten Messgröße vor. Es ist also die gesammte Gewichtsbelastung auf den Wägezellen in der Messung enthalten. Zudem ist der Werteberreich entsprechend einer 32-Bit Ganzzahl groß.

Der erste Schritt in der Abstraktion ist die durch das eigene Gewicht verursachte Belastung aus der Messung herauszurechnen. Um dies zu gewährleisten, muss diese Belastung zunächst gemessen werden. Die Implementation für diese Arbeit sieht vor, dass bei Start des Mikrocontrollers über einen kurzen Zeitraum eine Messungreihe erhoben wird. Durch die Position und des hohen Gewichtes des Akkupaketes des Longboards wird in der Ruhepositoin die hintere Achse deutlich stärker belastet. Für beide Achsen wird deshalb ein Mittelwert berechnet, um den Einfluss durch die eigentliche Belastung zu eliminieren. Um die Kalibration nicht zu verfälschen, darf während der Messreihe keine

Abbildung 4.4: Rauschen bei Stromversorgung durch USB-Akku.

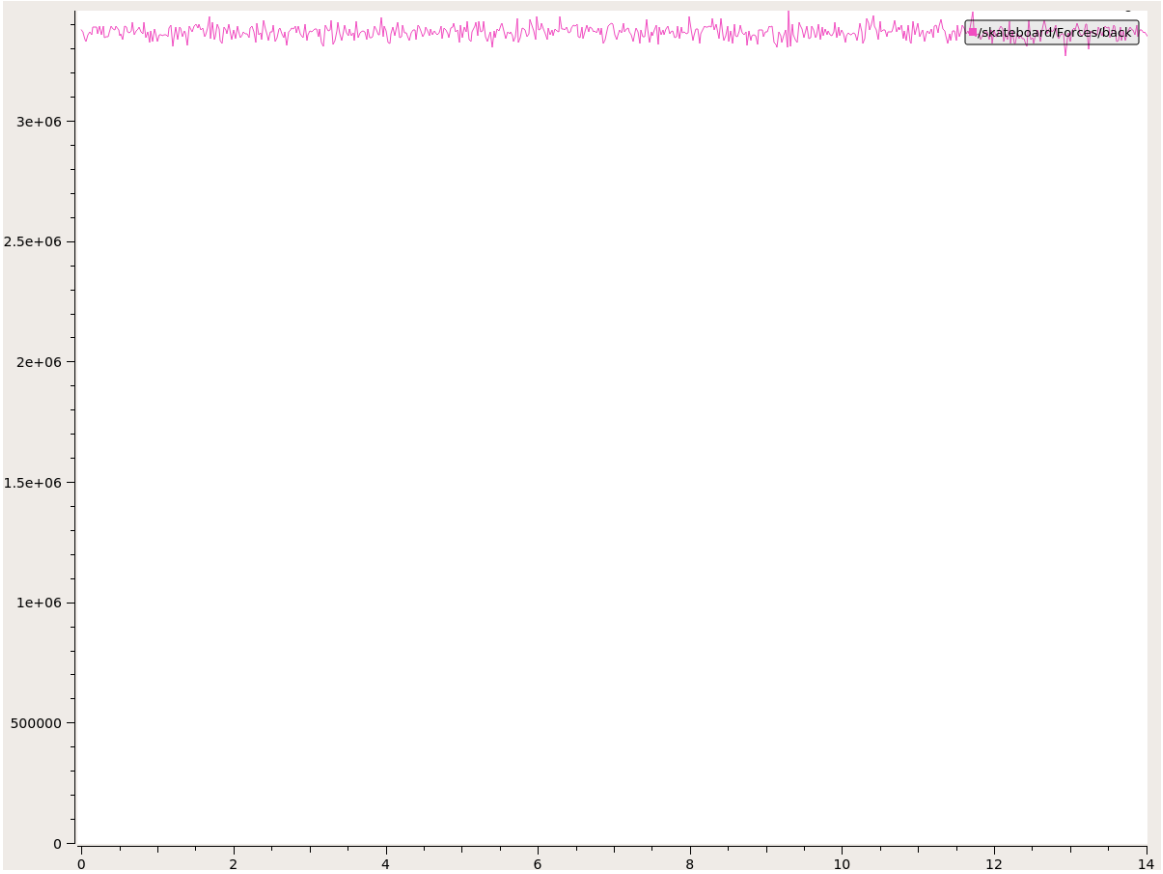


Abbildung 4.5: Rauschen bei Stromversorgung durch Laptop.

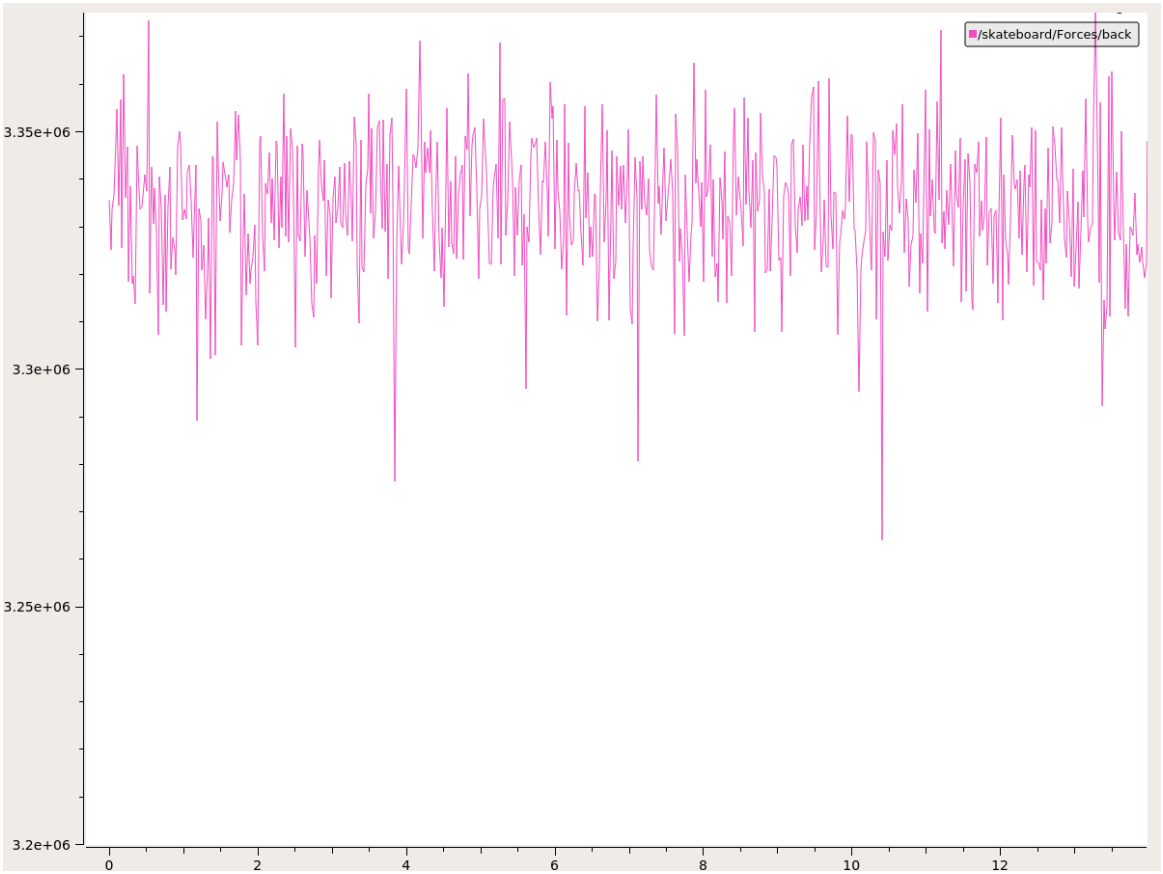


Abbildung 4.6: Rauschen bei Stromversorgung durch USB-Akku.

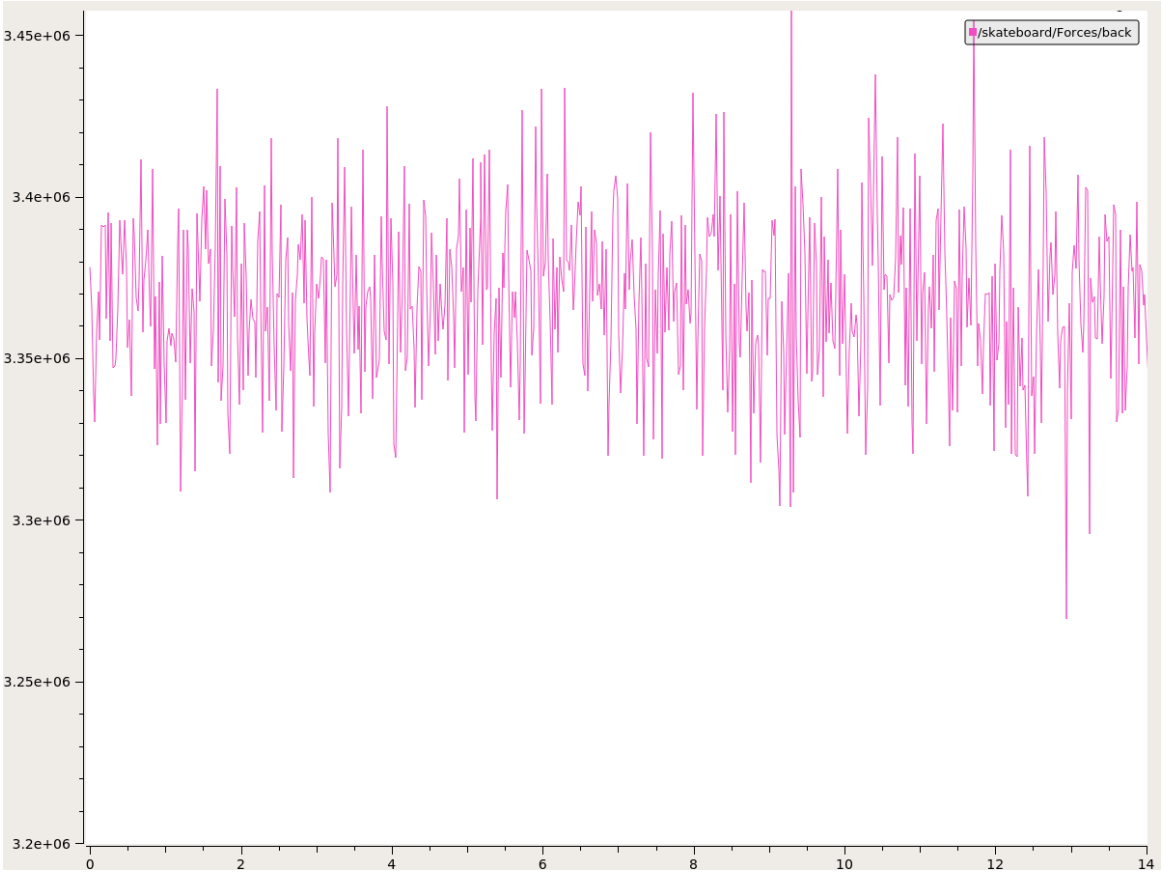


Abbildung 4.7: Rauschen bei Stromversorgung durch Laptop.

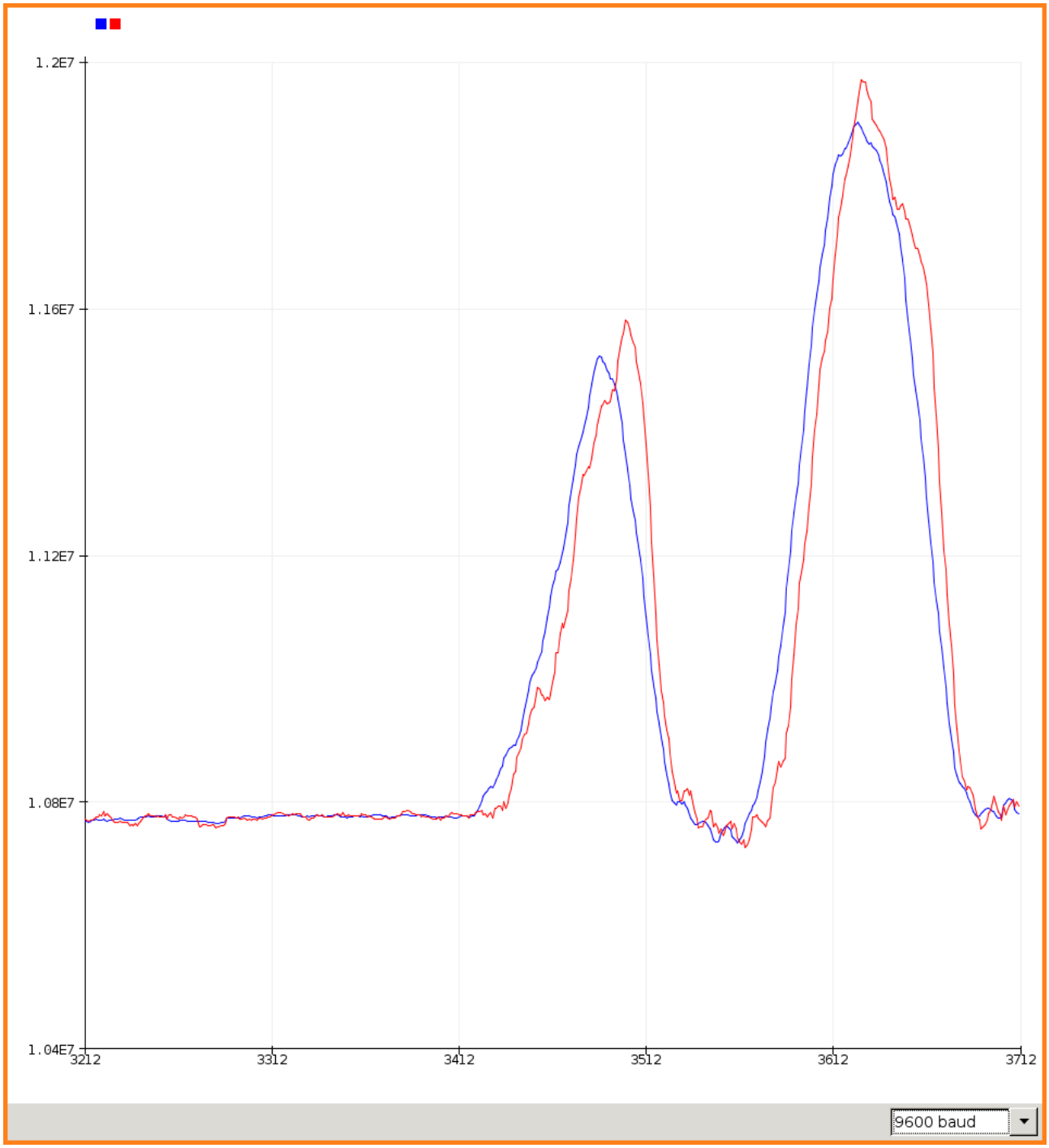


Abbildung 4.8: Vergleich des Signals nach bilden eines Mittelswertes über verschieden lange Zeiträume. 16 Messungen in Rot und 32 Messungen in Blau.

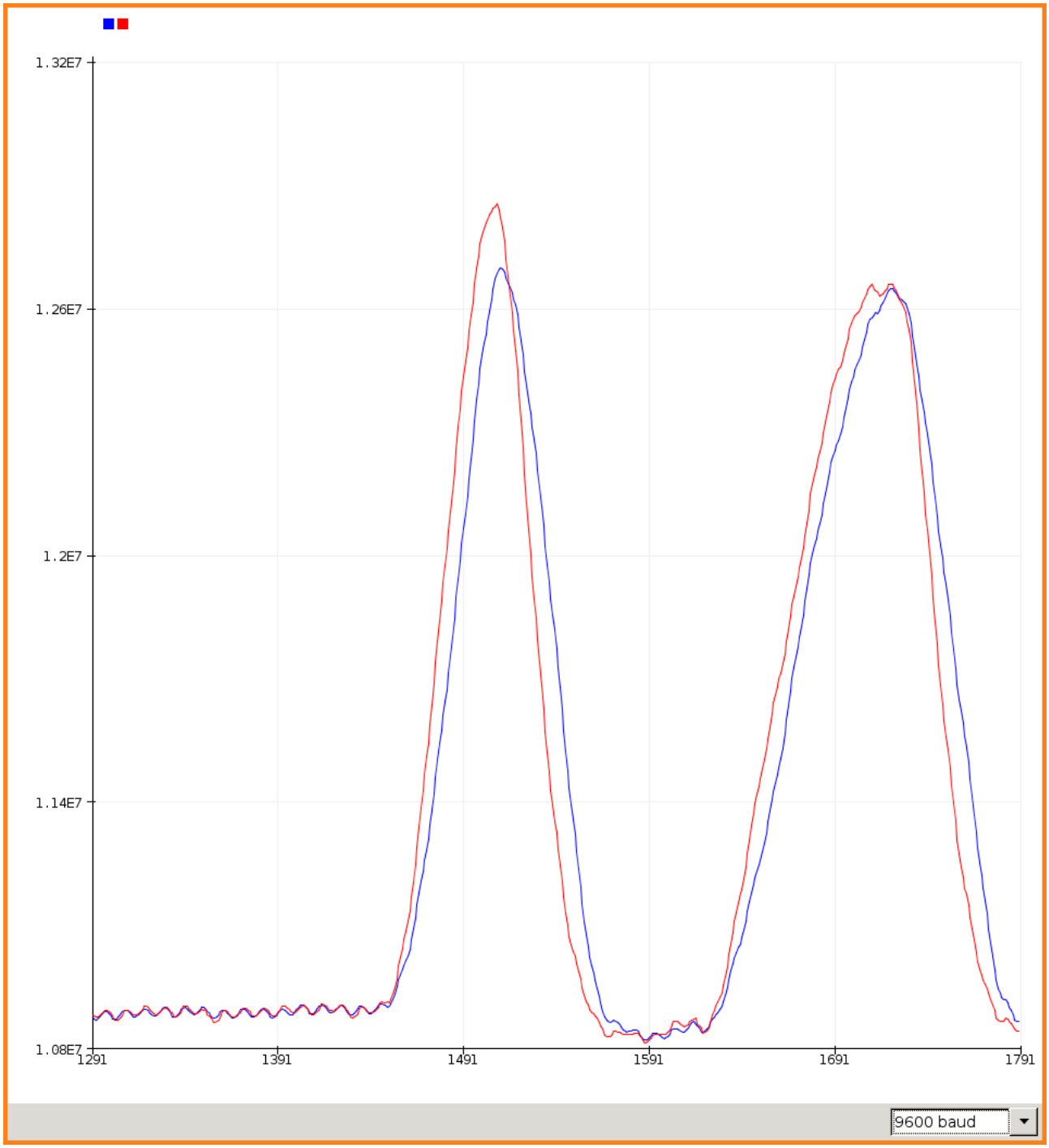


Abbildung 4.9: Vergleich der Mittelswerte generiert durch das gewichtete (rot) und das ungewichtete Verfahren (blau).

weitere Belastung auf das Board einwirken. Nach einer solchen Eichung sind die Messwerte ohne Belastung sehr nahe 0.

Es ist auch denkbar, das Eigengewicht des Boards zu messen und fest in den Quellcode einzubetten. Durch den Entwicklungsprozess ist dieses aber nicht konstant festzulegen. Dies liegen an zwei Faktoren. Zunächst besteht die Sensorhalterung aus Kunststoff und wird durch die Belastungen beim Fahren verformt. Der Anpressdruck der Sensoren auf die Unterseite des Longboards bleibt so nicht konstant. Analog werden die Sensorhalterungen mit Schrauben an das Board montiert, wodurch nach Montage der Achsen eine erneute Kalibration erforderlich ist.

Um die Sensordaten zu nutzen wird eine Differenz zwischen dem vorderen und hinteren Sensor gebildet. So entsteht ein einzelner Zahlenwert, durch welchen auf die Gewichtsverlagerung der Nutzer geschlossen werden kann. Der Werteberreich kann positive und negative Werte annehmen und nimmt Werte im Größenbereich der Messwerte an. Da der Motor mit Werten zwischen 0 und 1023 angesteuert wird, muss der Werteberreich der Messung auf den der Steurung abgebildet werden. Um dies zu erwirken muss ein Intervall auf dem Werteberreich der Messungen definiert werden. Die Begrenzungen geben jeweils an, wie stark sich ein Nutzer zum Bremsen beziehungsweise Beschleunigen bereit ist, in eine Richtung zu neigen. Die Grenzwerte des Intervalls sind also abhängig von dem Gewicht und Komfortzone eines spezifischen Anwenders. Wie beim Eigengewicht des Boards ist daher eine Konfiguration nötig. Besonders gilt dies, da verschiedene Personen ein individuelles Körpergewicht besitzen.

Ein Werkzeug, mit welchem Parameter wie der erwarteten Belastung interaktiv eingestellt werden, wird in dem Abschnitt 4.5.2 vorgestellt.

4.3.3 Fenstermechanismus

In Abschnitt 2.1 wird der Effekt diskutiert, welcher sich auf die Trägheit des Körpers einer auf dem Board fahrenden Person bezieht. Durch diesen folgt auf eine Reaktion des Motors ein sensorischer Input, welcher als Anweisung für eine der vorherigen entgegengesetzten Reaktion interpretiert werden kann. Die Folge ist eine Rückkopplung. Mit dem Fenstermechanismus wird versucht diesen Effekt zu minimieren.

Die grundsätzliche Idee ist auf eine Änderung der gemessenen Gewichtsverlagerung nur zu reagieren, wenn diese deutlich vom bisherigen Tempo abweicht. Schwache Schwankungen führen so nicht zu einer Änderung der Geschwindigkeit. Um diesen Effekt zu erreichen, wird der Werteberreich, auf welchen der Differenzwert skaliert wird 4.3.2 angepasst. Bei einer Fenstergröße von 200 würde die obere Grenze 1223 statt 1023 und die untere -200 anstatt 0 betragen. Sei die aktuelle Geschwindigkeit 512, also genau entsprechend der neutralen Position. Verlagerungen des Gewichts bewirken nun keine Änderungen der Geschwindigkeit, solange diese nicht einen Differenzwert verursachen, der 200 größer oder kleiner als die aktuelle Geschwindigkeit ist. Bei einem Input von 700 errechnet der Fenstermechanismus so eine Zielgeschwindigkeit von 512. Mit der genannten Fenstergröße ergeben sich so die Grenzen 312 und 712 die das Fenster aufspannen. Die Eingabe von 900 liegt außerhalb des Fensters und bewirkt so eine Reaktion. Da 900 oberhalb des Fensters liegt, wird zunächst die Differenz δd zwischen der oberen Grenze und der Eingabe berechnet. Beide Grenzen werden um δd in positiver Richtung verschoben. Die neue Obergrenze lautet so 900 und die Untergrenze 500. Um die Geschwindigkeit zu bestimmen bestimmen, wird der Mittelpunkt zwischen den Grenzen bestimmt. In diesem Fall ergibt sich also eine Geschwindigkeit von 700. Aufgrund dieser Logik ergeben sich die angepassten Grenzen des Wertebereichs. Der Maximalwert 1223 ergibt so nach Abziehen der Fenstergröße den maximal als Eingabe für den Motor geeigneten Wert 1023. Das Verhalten ist für das Bremsen analog durch Verwendung der unteren Grenze des Fensters.

Aufgrund der Verschiebung der Grenzen muss, um eine Reduktion der Geschwindigkeit zu bewirken, die Eingabe unter den Wert von 500 fallen. Mit Probefahrten wurde der Einfluss der Fenstergröße auf das Fahrverhalten beobachtet und angemessene Werte ermittelt. Der einfache Fenstermechanismus erschwert die neutrale Geschwindigkeit einzustellen. Entgegen der Erwartung muss das vorderen oder hintere Bein stärker belastet werden, abhängig davon, ob zuvor gebremst oder beschleunigt wurde. Daher werden, wenn der ermittelte Mittelpunkt weniger als die Fenstergröße von der neutralen Position 512 entfernt ist, die Grenzen langsam verschoben. Der Wert der Verschiebung ist so

gewählt, dass sich der Mittelpunkt stetig der neutralen Position annähert. In dem Abschnitt 5 wird die Wirkung des Fensteralgorithmus ausgewertet.

Bremsfunktion	
a	-0.00208
b	2.066
c	0

Beschleunigungsfunktion	
a	0.00253
b	-2.8884
c	1326.82982

Tabelle 4.1: Standardeinstellung der Koeffizienten für beide Funktionen

4.3.4 Reaktionskurve

Die Reaktionskurve ist ein Mechanismus, mit dem Ziel ein ruhigeres Fahrerlebnis zu ermöglichen. Verlagerungen nahe der Extremwerte, also der 4.3.2 vom Nutzer als noch als komfortabel definierten Werte, sollen eine proportional stärkere Wirkung erzielen, als die nahe der neutralen Position. Der Gedanke ist hierbei, dass bei geringeren Geschwindigkeiten eine höhere Präzision benötigt wird.

Ausgangspunkt für die Berechnung ist die Definition zweier quadratischen Funktionen. Diese bestimmten je das Ansprechverhalten im Intervall zum Bremsen $[v < 512]$ und dem zum Beschleunigen $[v \geq 512]$. Die Funktionen werden über jeweils 3 Koeffizienten definiert und. Mit dem Mittelpunkt aus als Eingabe wird der Funktionswert mit $f(x) = ax^2 + bx + c$ ermittelt. Zusätzlich werden die Funktionswerte auf das Intervall der Eingabe beschränkt. So kann für ein x aus dem Beschleunigungsintervall $[512, 1023]$ keine Werte über 1023 oder unterhalb 512 generieren. Das Bremsintervall ist analog beschränkt. Die Koeffizienten aus der Tabelle 4.1 generieren die in 4.10 dargestellten Kurven.

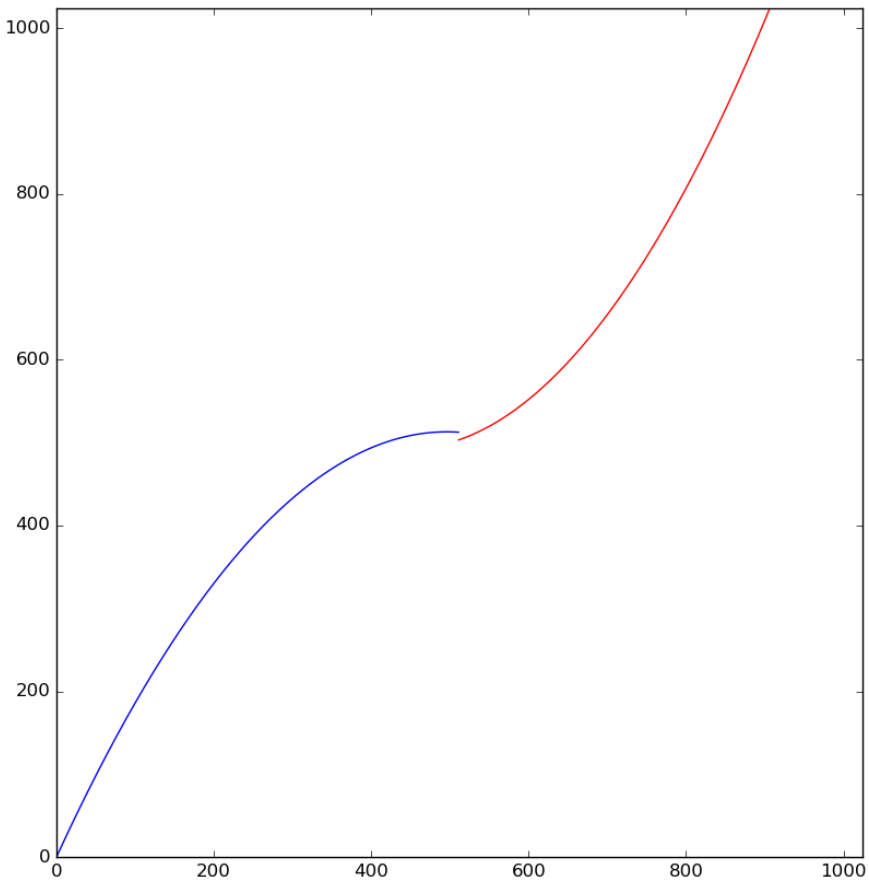


Abbildung 4.10: Visualisierung der beiden Ansprechkurven durch selbst entwickeltes Werkzeug. Die Beschleunigungskurve in Rot und die Bremskurve in Blau

```
12:14:28.924 -> datapoints sent -> 5
12:14:28.924 -> datapoints sent -> 5
12:14:28.957 -> datapoints sent -> 5

```

Abbildung 4.11: Teile der Ausgabe bei Aktivierung der print-Befehle in der WiFi-Komponente. Die Zeitstempel wurden durch die IDE generiert. Vergleiche 4.12

4.4 Unterstützende Software

Um die Entwicklung der Filteralgorithmen zu beschleunigen wurden mehrere unterstützende Programme geschrieben. Für die verwendete Software müssen Daten zwischen der Software auf einem Laptop und dem Mikrocontroller ausgetauscht werden. Zusätzlich zu der seriellen Kommunikation mit einem Kabel wurde eine auf WiFi basierende Funkschnittstelle entwickelt.

4.4.1 Debugging mittles Arduino IDE

Wie bei jeder anderen Software war die Software für den Mikrocontroller zunächst nicht fehlerfrei. Um semantische und logische Fehler Quellcode ausfindig zu machen wurde die Arduino IDE genutzt. Diese macht es trivial eine auf Text basierte Kommunikation mit dem Mikrocontroller mittels einer seriellen Schnittstelle aufzunehmen. So kann zu beliebigen Zeitpunkten der Inhalt von Variablen ausgegeben werden und so Kenntnis über den Zustand des Controllers gewonnen werden.

Um diese Funktionalität nutzen zu können muss einmalig die Funktion `Serial.begin(<Rate>)` aufgerufen werden. Dabei ist `<Rate>` die Baudrate, mit welcher der Mikrocontroller die Daten ausgibt. Die eigentlichen Daten werden mit den Funktionen `Serial.print(<Daten>)` und `Serial.println(<Daten>)` verschickt.

Der Quellcode der Mikrocontrollersoftware wurde mit zahlreichen Anweisungen versehen, welche für die Behebung eines speziellen Fehlers benötigt wurden und nur noch auskommentiert vorhanden sind. Zudem existieren Schreibanweisungen, die mit Hilfe des `#ifdef`-Mechanismus der Programmiersprache C++ aktiviert oder deaktiviert werden können [3]. Anweisungen dieser Art werden bei deaktivierten Zustand schon vor dem Kompilieren aus dem Quellcode entfernt und so nicht mit übersetzt. Dies hilft dabei, bei wachsender Menge an Programmzeilen nur die aktuell wichtigen Fehlermeldungen anzuzeigen. Die Grafik 4.11 zeigt exemplarisch einen Teil der Ausgabe bei aktivierten print-Befehlen innerhalb der WiFi-Routine.

Die Funktion aus dem Beispiel 4.12 ist ein Ausschnitt aus der Mikrocontrollersoftware. Sie koordiniert die drahtlose Kommunikation mit dem per WiFi verbundenen Laptop. Bei der Analyse von Verbindungsproblemen wurden mehrere Schreibanweisungen eingefügt, die die Vorgänge innerhalb des Mikrocontrollers protokollieren. Ein gutes Beispiel ist die Ausgabe *"parameter change received"*. Durch Anweisungen dieser Art war es bei der Entwicklung leichter nachvollziehbar wo Fehler ihren Ursprung hatten. Wenn der Mikrocontroller auf eine Anfrage nicht reagiert und zusätzlich keine serielle Ausgabe liefert, legt dies Nahe, dass die Anfrage vom Laptop nicht korrekt übertragen wurde.

4.5 Kommuniktion zwischen Mikrocontroller und Computer

Um die Wirksamkeit der gewählten Algorithmen 4.3 und deren Parameter zu prüfen, werden Messdaten der Sensoren und andere interessante Kennwerte über eine Funkschnitstelle auf einen unterstüzenden Computer überschrieben.

Dies wird realisiert durch das Versenden von UDP-Paketen über eine mittels einer WiFi-Verbindung zwischen Mikrocontroller und einem Computer. Um die Verbindung zu realisieren, öffnet der mit einem WiFi-Chip ausgestattete Mikrocontroller einen Accesspoint. Aus Gründen der Mobilität wird als Empfänger ein Laptop verwendet, damit dieser bei Testversuchen innerhalb der Sendereichweite bleibt. Über diese Verbindungen laufen zwei Dienstleistungen.

Abbildung 4.12: Funktion mit Debug-Statements um den Austausch von Paketen nachzuvollziehen.

```
void wifi_action() {
  sinkUdp.beginPacket(sink_addr, sink_port);
  for (int p = 0; p < sensor_buf_usage; ++p) {
    sinkUdp.write((char*) &sensor_buf[p], sizeof (struct sdp_single));
  }
  sinkUdp.endPacket();

#ifdef DEBUG_WIFI
  Serial.print("datapoints sent -> ");
  Serial.println(sensor_buf_usage);
#endif

  sensor_buf_usage = 0;

  struct parameter_change recv_buf = {0, 0, 0};
  int num_recv = parameterUdp.parsePacket();

  if (num_recv == sizeof(struct parameter_change)) {
#ifdef DEBUG_WIFI
    Serial.println("parameter change received");
#endif

    int bytes_read = parameterUdp.read((char*) &recv_buf, sizeof(struct
    parameter_change));

    if (bytes_read == sizeof(struct parameter_change)) {
      if (recv_buf.par_id < PARAMETER_COUNT &&
          recv_buf.par_id >= 0) {

        parameter_react(&recv_buf);
        recv_buf.ack_p = 1;

      } else {
        recv_buf.ack_p = 3;
      }

      parameterUdp.beginPacket(sink_addr, parameter_port);
      parameterUdp.write((char*) &recv_buf, sizeof(struct parameter_change));
      parameterUdp.endPacket();
    }
  }
  else {
#ifdef DEBUG_WIFI
    Serial.println("ecv_parameter: num_recv != sizeof(struct parameter_change)");
    Serial.print("  expected -> ");
    Serial.println(sizeof(struct parameter_change));
    Serial.print("  received -> ");
    Serial.println(num_recv);
#endif
  }
}
```

```
1   typedef union {
2     int32_t i[4];
3     float f[4];
4   } sdp_data;
5
6   struct sdp_single { // Sensor Data Point uint8_t id; sdp_data data; };
7
```

Abbildung 4.13: Definition eines einzelnen Messdatenpunktes

Zum ersten sendet der Mikrocontroller in hoher Frequenz UDP-Pakete an den Computer, die Messdaten und Informationen die Aufschluss über den Status des Controllers enthalten. Auf dem Computer werden diese in den im Abschnitt 4.5.1 beschriebenen System weiter verarbeitet.

Die Zweite Dienstleistung ermöglicht Optionen und Parameter am Mikrocontroller zur Laufzeit zu verändern. Durch senden spezieller UDP-Pakete kann der Computer die Werte spezieller Parameter im Mikrocontroller setzen und so das Verhalten ändern. Zusätzlich werden um die Wirksamkeit der gewählten Algorithmen 4.3 und deren Parameter zu prüfen, Messdaten der Sensoren und andere interessante Kennwerte über eine Funkschnitstelle auf einen unterstüzenden Computer überschrieben.

Dies wird realisiert durch das Versenden von UDP-Paketen über eine mittels einer WiFi-Verbindung zwischen Mikrocontroller und einem Computer. Um die Verbindung zu realisieren, öffnet der mit einem WiFi-Chip ausgestattete Mikrocontroller einen Accesspoint. Aus Gründen der Mobilität wird als Empfänger ein Laptop verwendet, damit dieser bei Testversuchen innerhalb der Sendereichweite bleibt. ich können über die selbe Schnittstelle die aktuellen Werte dieser abgefragt werden, ohne diese zu verändern. Der Abschnitt geht auf die Anwendung dieser Funktionalität ein.

4.5.1 Verarbeitung und Darstellung der Messdaten

Die Controllersoftware verwaltet einen Absendepuffer der mit einzelnen Datenpunkten gefüllt wird. Regelmäßig sendet der Controller diese in einem Paket gebündelt ab. So wird die Zahl der versendeten Pakete minimiert.

Alle Routinen die Messdaten erheben, schreiben diese in dem Format eines Messdatenpunktes auf den Absendepuffer. Zusätzlich werden bestimmte Variablen protokolliert, um die Arbeit der in Algorithmen 4.3 nachvollziehbar zu machen. In der Definition des Formats 4.13 ist erkennbar, dass zusätzlich zu den eigentlichen Daten eine Messung noch eine *id* enthält. Diese *id* wird genutzt um die Daten einem spezifischen Sensor zuordnen zu können. Innerhalb des Formates ist erkennbar, dass die Datenfelder als Fließkommazahlen oder ganze Zahlen interpretiert werden können. Deshalb wird die *id* zusätzlich genutzt um den Datentyp der in der Messung enthaltenen Daten zu bestimmen. Exemplarisch geben der Beschleunigungsmesser und das Gyroskop der IMU ihre Daten in der Form von Fließkommazahlen aus.

Auf der Seite des Empfängers läuft ein System des *Roboter Operating Systems (ROS)*. Dies ist eine Sammlung von Werkzeugen zur Verarbeitung, Aufzeichnung und Visualisierung von Messdaten in Echtzeit.

Zur Nutzung von ROS ist es nötig mehrere Applikationen zu schreiben, die mit einem zentralen Programm namens *roscore* kommunizieren. Daher ist es nötig die Programme in einer Sprache zu verfassen, in welcher die für die Kommunikation benötigten Bibliotheken verfügbar sind. Die Sprache Python ausgewählt um den Entwicklungsprozess zu beschleunigen.

Für ROS entwickelte Software muss in ein spezielles Arbeitsverzeichnis gespeichert werden um für die ROS-Werkzeuge wie *roslaunch* sichtbar zu sein.

Grundkonzepte von ROS

Von zentraler Bedeutung für um ROS zu verstehen sind die Topics. Einem jeden mit ROS verbundenen Programm ist es möglich, beliebige strukturierte Daten auf einem Topic zu veröffentlichen.

```
std_msgs/Header header
uint32 front
uint32 back
int32 diff
```

Abbildung 4.14: Spezifikationsdatei eines selbst erstellten Nachrichtentyps für ROS

```
import rospy

# importieren des selbsterstellen Nachrichtentyps
from skateboard.msg import Forces

rospy.init_node('longboard_talker')

publisher = rospy.Publisher('/skateboard/Forces', Forces, queue_size=10)

# Beispieldaten
front = 100
back = 200
diff = -100
velocity = 320

msg = Forces()
msg.header.frame_id = "board"
msg.header.stamp = rospy.Time.now()
msg.front = front
msg.back = back
msg.diff = diff

publisher.publish(msg)

```

Abbildung 4.15: Beispiel für ein Pythonskript welches Daten auf einem ROS-Topic veröffentlicht.

Wiederum kann jedes Programm Subscriber für ein bestimmtes Topic sein. Subscriber werden über auf dem Topic neu veröffentliche Daten mittels einer Callback-Funktion informiert. Bevor Daten auf einem Topic veröffentlicht werden könne, ist es nötig diesen einen Datentyp zuzuweisen.

Für den Anwendungsfall dieser Arbeit wurden zusätzlich Datentypen entworfen, die Speziell auf die entsprechenden Sensoren zugeschnitten sind. Zusätzlich zu dem Erstellen der Spezifikation 4.14 müssen diese dem System bekannt gemacht werden. Dies wird erreicht indem sie innerhalb der *CMakeLists.txt*-Datei für das Arbeitsverzeichnis eingetragen werden und dieses Buildscript ausgeführt wird.

Empfangen der Daten

Basis für die Verarbeitung der Daten ist eine Software, welche Daten in das System einspielt. *receiver* ist hierbei ein Programm, welches die vom Controller drahtlos an den Laptop übertragen wurden empfängt und in ROS einspielt. Der Code in 4.15 illustriert, wie Daten in auf einem ROS-Topic veröffentlicht werden können.

Aufzeichnung und Visualisierung der Daten

Um die Daten nach Erhebung für Menschen lesbar zu machen wird das Werkzeug *PlotJuggler* verwendet. Kann vom Mirkocontroller übertragenen Daten in Echtzeit visuell darstellen. So ist es möglich die Folgen einer Bewegung auf dem Longboard direkt zu sehen. Das Layout von *Plotjuggler* lässt sich

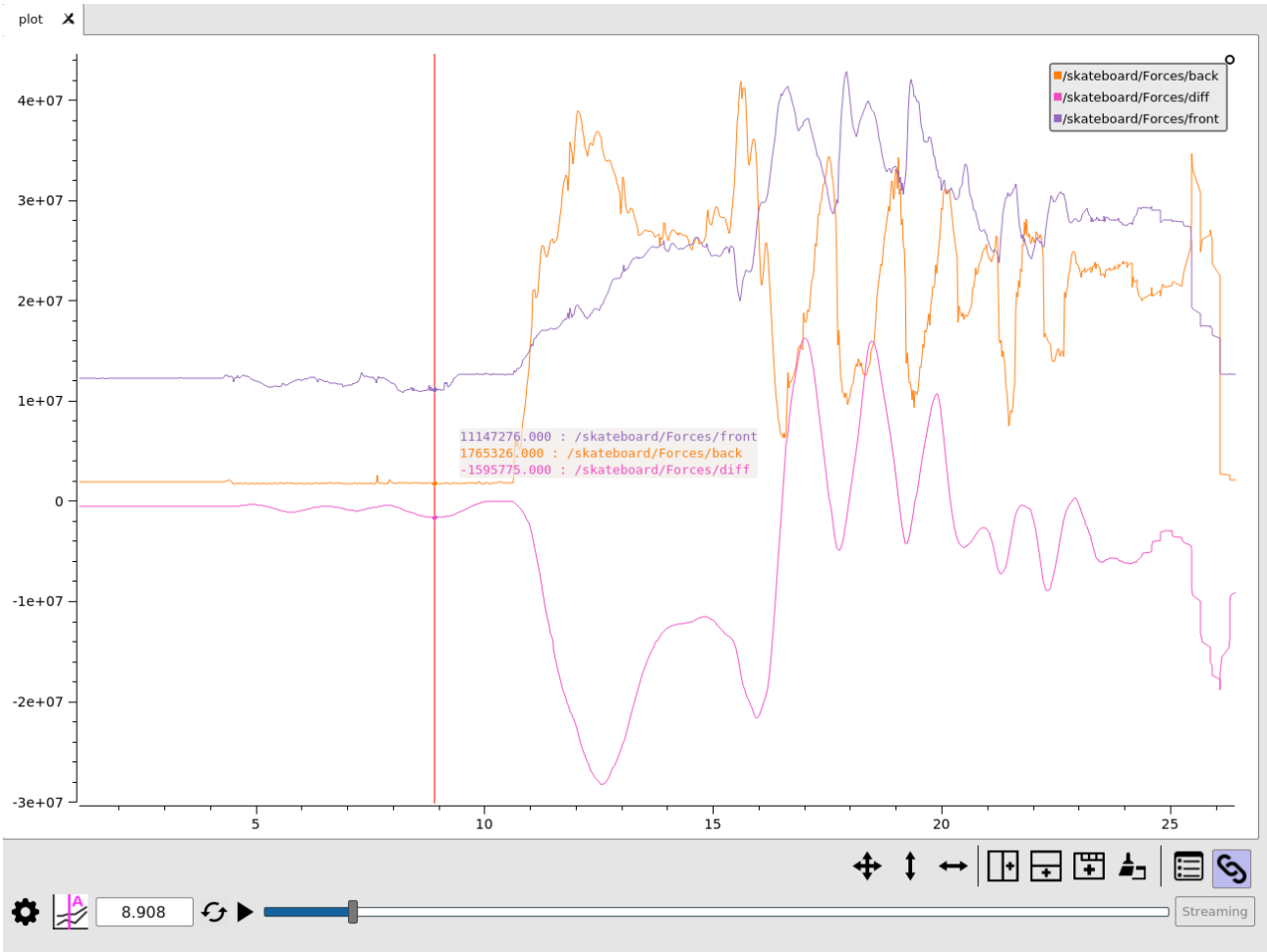


Abbildung 4.16: Vergleich des Signals nach bilden eines Mittelswertes über verschieden lange Zeiträume

vom Benutzer anpassen und speichern.

Um die Daten auch nach Fahrt auswerten zu können wird das Werkzeug *rosbag* [7] genutzt. Dieses wird Subscriber für alle beim Start angegebenen Topics und zeichnet alle übertragenen Nachrichten in einer Datei auf. Eine solche Datei lässt sich in Plotjuggler öffnen und so analysieren. Neben der Auswertung nach einer Fahrt kann durch die Speicherung der Messdurchläufe ein langfristiger Eindruck gewonnen werden. So können alte und neue Messungen verglichen werden und Verbesserungen aber auch Regressionen beobachtet werden.

Starten der Software mittels *roslaunch*

Um das Starten der ROS-Umgebung zu vereinfachen stellt ROS das Werkzeug *roslaunch* bereit. Nach erstellen einer *.lauch*-Datei können so alle dort eingetragenen Programme und der *roscore*

```
#!/bin/bash
rosbag record\
       /skateboard/Acc\
       /skateboard/Gyro\
       /skateboard/Forces\
       /skateboard/Vel\
       /skateboard/Window
```

Abbildung 4.17: Shellskript, welches die Beobachtung meherer ROS-Topics durch *rosbag* startet

```
<launch>
  <node name="receiver" pkg="skateboard" type="receiver" />
  <node name="plot_subscriber" pkg="skateboard" type="plot_subscriber" />
  <node name="parameter_changer" pkg="skateboard" type="parameter_changer" />
  <node name="rqt_reconfigure" pkg="rqt_reconfigure" type="rqt_reconfigure" />
  <node name="plotjuggler_with_layout"
  pkg="plotjuggler"
  type="PlotJuggler"
  args="-\-layout $(find skateboard)/config/PlotJuggler_layout.xml"
  />

  <node name="plotjuggler"
  pkg="plotjuggler"
  type="PlotJuggler"
  />
</launch>
```

Abbildung 4.18: Konfigurationsdatei mit welcher angegeben wird, welche Software *roslaunch* starten soll

selbst in der richtigen Reihenfolge mit nur einem einzigen Befehl gestartet werden. Die einzige Bedingung um die Programme erfolgreich zu starten ist, dass der verwendete Laptop mit WiFi-Netz des Mikrocontrollers verbunden sein muss.

4.5.2 Optimierung der Softwareparameter

Der Aufwand um während einer Testphase die Parameter für die Algorithmen des Mikrocontrollers anzupassen sollte möglichst gering sein. Der übliche Prozess den Quellcode zu modifizieren, neu zu kompilieren und anschließend mit einer Kabelverbindung auf den Controller zu flashen wird daher um eine Alternative ergänzt. Deshalb wurde eine Schnittstelle entwickelt, welche es ermöglicht von einem über WiFi zum Controller verbundenen Computer Parameter zu ändern. Hierfür werden einfache Pakete an den Controller übermittelt, welche die auszuführende Änderung spezifizieren. Es kann sich dabei um Änderungen einfacher Variablen handeln oder den Wechsel in in einen anderen Modus.

Der Mikrocontroller verfügt über 3 Modi, die dabei helfen die Parameter zu optimieren. Neben den Modus *drive*, welcher die Sensordaten interpretiert und den Motor ansteurt, existieren *active* und *active*. In beiden dieser Modi ist es einer Testperson möglich, auf dem Longboard zu stehen, ohne dass der Motor auf Bewegungen der Person reagiert. Im *active*-Modus bremst der Motor stets mit voller Kraft, während hingegen im *passive*-Modus sich der Motor im Leerlauf befindet.

Auf einem Laptop läuft ein Programm, welches mit dem ROS-Modul [8] Parameter vom Mikrocontroller auslesen und verändern kann. Ein entsprechendes Fenster erlaubt es Parameter interaktiv zu verändern. Veränderte Werte werden im Hintergrund auf den Mikrocontroller überspielt. Welche Parameter existieren muss zunächst in einem Pythonskript deklariert werden. Analog zu den benutzerdefinierten Nachrichten aus Abschnitt 4.5.1, muss die Deklaration der Parameter im Buildscript eingetragen werden. hinterlegt werden. Die komplette Konfiguration ist zu umfangreich um sie an dieser Stelle einzubetten. Aus Platzgründen zeigt daher Code in 4.20 zeigt exemplarisch wie Parameter deklariert werden. Zusätzlich zu der durch das Modul generierten grafischen Schnittstelle 4.21 öffnet das Programm ein zusätzliches Fenster 4.19. In diesem werden die Ansprechkurven 4.3.4 die über die Parameter definiert sind dargestellt. Dies erhöht während der Optimierung den Effekt von neuen Parametern die Nachvollziehbarkeit. Dass in dieser Darstellung die Funktionswerte auch außerhalb des gültigen Intervalls $[0, 1023]$ liegen können, dient dazu Parameter zu identifizieren diese Grenze verletzen würden. Innerhalb der Software auf dem Mikrocontroller werden die Werte auf das Intervall begrenzt. Bei schlechter Parameterwahl würde zum Beispiel ein großer Teil des Eingabeintervalls der Beschleunigungsfunktion auf 1023 abgebildet werden.

Im Hintergrund werden diese Änderungen von dem Programm *parameter_changer* in Pakete ge-

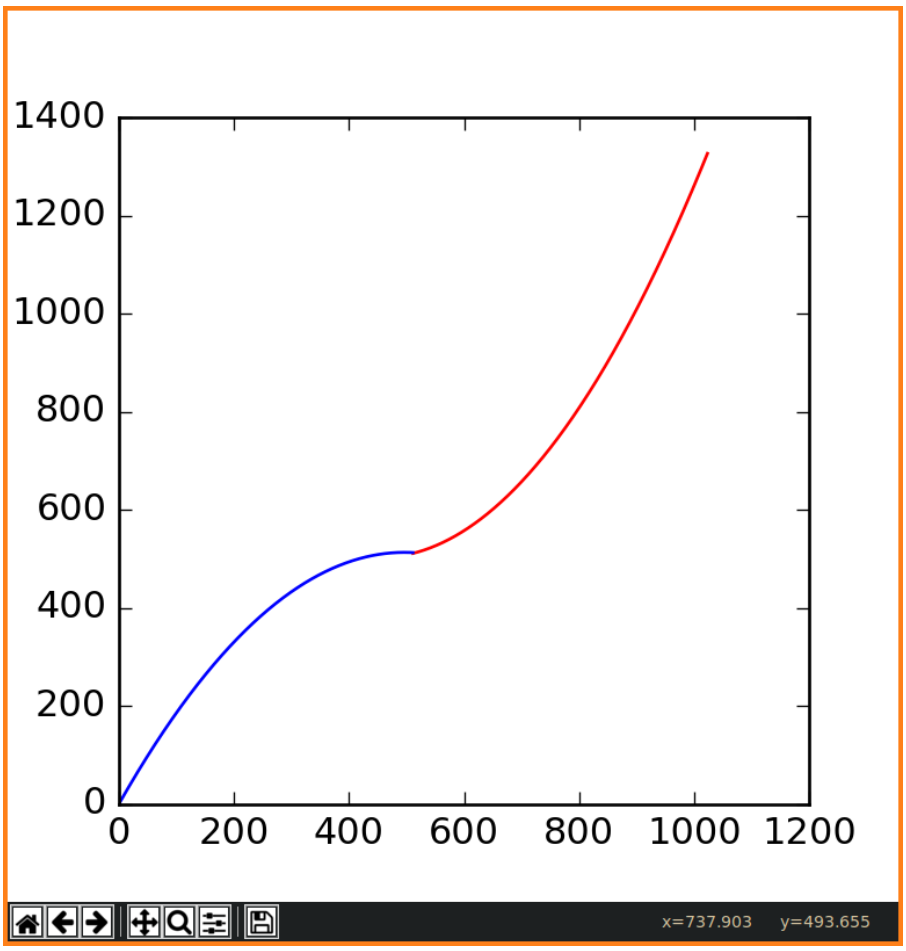


Abbildung 4.19: Grafische Ausgabe zur Kontrolle der Auswirkungen neuer Parameter.

packt und an den Mikrocontroller gesendet. Das Struct aus dem Ausschnitt des Quelltextes 4.22 definiert hierbei wie die Pakete aufgebaut sind. Die *par_id* gibt an, welcher Parameter geändert werden soll. *new_val* enthält den zu übernehmenden Wert für einen Parameter. Da die Kommunikation mit dem Mirkocontroller über eine UDP-Verbindung läuft, muss mit dem Verlust von Paketen gerechnet werden. Daher wird von dem Programm *parameter_changer* erwartet, dass der Mirkocontroller eine Antwort auf die Änderungsanfrage sendet. Diese entspricht ausgenommen *ack_p* genau der Anfrage. *ack_p* wird von *parameter_changer* zunächst mit 0 initialisiert. Bei erfolgreicher Änderung sendet der Mikrocontroller *ack_p* mit dem Wert 1 zurück. Der Wert 2 bedeutet, dass der gesendete *new_val* außerhalb des für den entsprechenden Parameter erlaubten Wertebereichs befindet. 3 gibt an, dass der Parameter mit der angefragten *par_id* nicht existiert.

Die auf dem Laptop laufende Anwendung hat zusätzlich die Möglichkeit die aktuelle Belegung der Parameter auszulesen. Ein *ack_p*-Wert von 4 veranlasst den Mikrocontroller den eingehenden *new_val* zu ignorieren und stattdessen den aktuellen Wert in das Antwortpaket zu schreiben.

```
PACKAGE = "skateboard"
from dynamic_reconfigure.parameter_generator_catkin import *

gen = ParameterGenerator()
mode_enum = gen.enum(
[
  gen.const("calib",    int_t, 1, "calibration mode"),
  gen.const("drive",    int_t, 2, "driving mode"),
  gen.const("active",   int_t, 3, "active mode"),
  gen.const("passive",  int_t, 4, "passive mode")
  ],
  "Enum to describe the possible driving modes")

gen.add("mode", int_t, 0, "current driving mode", 1, 1, 4, edit_method=mode_enum)

```

Abbildung 4.20: Spezifikation für Anfragen die Parameter des Mikrocontrollers ändern.

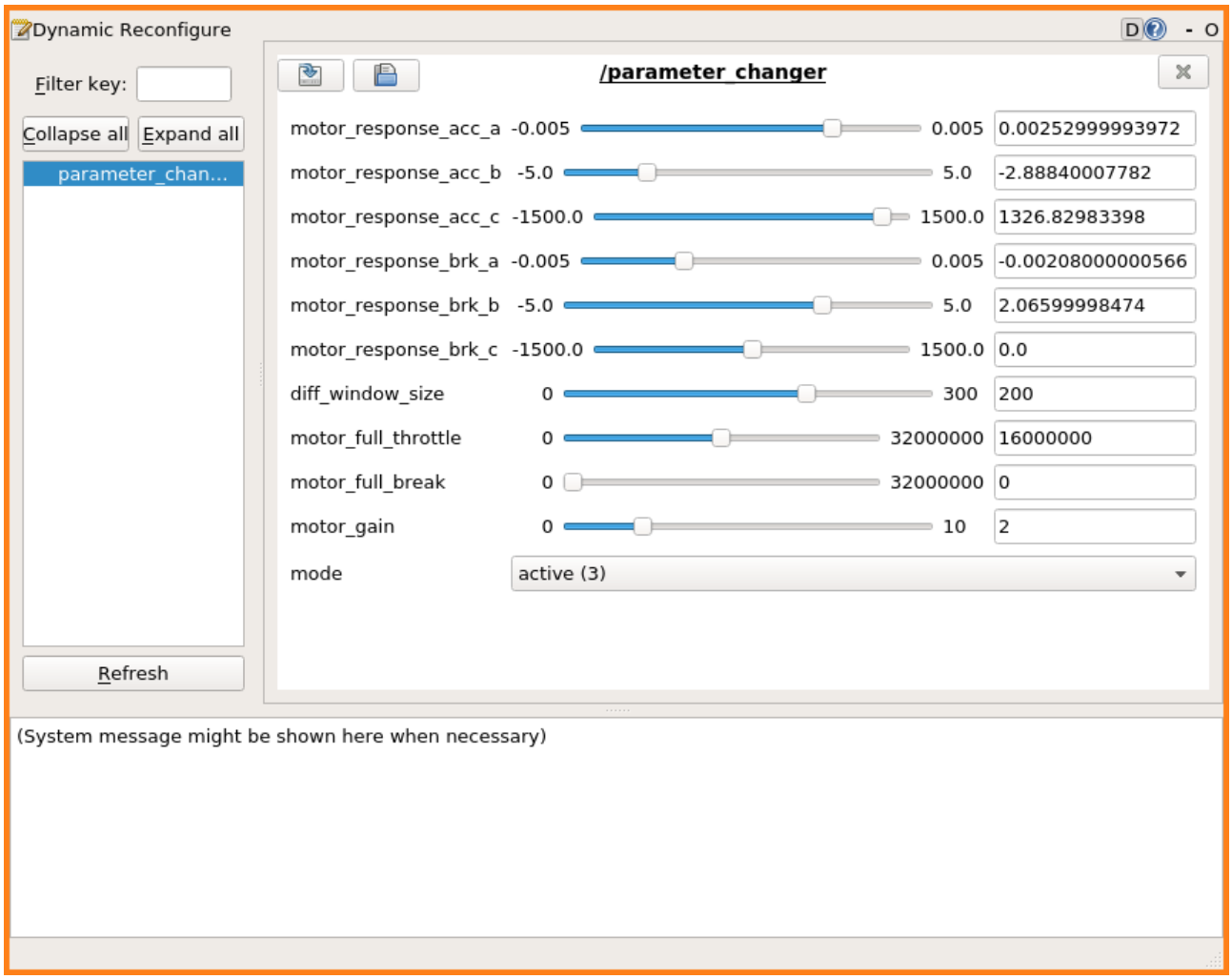


Abbildung 4.21: Mit rqt_reconfigure erstellte Oberfläche zur dynamischen Änderungen auf den Mikrocontroller

```
1 struct parameter_change {
2     int32_t par_id;
3     union {
4       int32_t i;
5       float f;
6     } new_val;
7     int32_t ack_p;
8 };
9
```

Abbildung 4.22: Spezifikation für Anfragen die Parameter des Mikrocontrollers ändern.

5 Zusammenfassung

Dieses Kapitel fasst die Funktionalität die im Laufe der Arbeit erfolgreich erarbeitet wurden zusammen und vergleicht diese mit den in der Zielsetzung 1.1 geplanten. Zusätzlich wird erwägt, wie der Funktionsumfang in einer hypothetischen Fortsetzung der Arbeit erweitert werden könnte.

Um einen schnellen Entwicklungszyklus zu ermöglichen sind Teile Hardware in der aktuellen Form Platzhalter. Die Sensorhalterung und das Gehäuse für den Mikrocontroller bestehen durch den Fertigungsprozess in einem 3D-Drucker aus Plastik 3.1. Besonders die die Sensorhalterung sollte bei einer Anwendung über eine längere Zeit oder stärkerer dynamischer Last aus einem besser geeigneten Material gefertigt werden. Eine nicht auf Tauglichkeit geprüfte Option wäre gefräster Stahl oder Aluminium. Auch die Verschraubung der Rollachsen mit dem Longboard müsste bei stärkerer Belastung verbessert werden. Die aktuelle Montage durch zwei lediglich zwei Schrauben wären langfristig nicht adäquat.

Aktuell ist die Implementation der Elektronik dem Nutzen angemessen. Ein Breakout-Board 3.2 verschaltet alle Komponenten miteinander. Zu diesen Komponenten gehören auch Schalter die an der äußeren Seite des Gehäuses befestigt sind mit welchen die Fernbedienung, die zum Zwecke der Motorsteuerung verbaut wurde, ansteuern lässt. Ein Neustarten des Mikrocontrollers und das Einspielen neuer Software auf diesen sind möglich, ohne das Gehäuse zu demontieren. Die Fertigung einer Leiterplatine ist nach Ansicht des Autors selbst bei auf dieser Arbeit aufbauenden Projekte nicht sinnvoll. Gründe hierfür sind der hohe Aufwand, der geringe Mehrwert und eventuelle Anpassungen an das neue Projekt.

Der Hauptteil des Arbeitsergebnis ist die Software, welche sowohl auf dem Mikrocontroller, als auch auf einem unterstüzenden Laptop ausgeführt wird. In Kombination ermöglichen beide Teilstücke die Erhebung, Aufzeichnung und Auswertung von Messdaten. Durch die drahtlose Verbindung ist es zudem möglich das die Steuerungssoftware neu zu konfigurieren.

5.1 Befragung von Nutzern bezüglich der Fahreigenschaften

Als Abschluss der Arbeit wurde eine Befragung durchgeführt, mit dem Ziel die Qualität des Fahrverhaltens zu quantifizieren

Der Mechanismus zur Änderung von Parametern über eine drahtlose Verbindung 4.5.2 wurde genutzt um schnell zwischen vorgefertigten Konfigurationen umzuschalten und zu vergleichen.

Der Aufbau der Befragung bestand hierbei aus einer Fahrt des Longboards mit einer Fernbedienung, also der Ursprünglichen Steuerung. Danach fanden Fahrten mit der im Rahmen dieser Arbeit implementierten Steurung statt. Diese Steuerung wurde in vier verschiedenen Konfigurationen getestet und gewertet. Dabei wurde verglichen, wie sich die Filteralgorithmen 4.3 auf das Fahrverhalten auswirken. Im Anhang 5.1 ist der Fragebogen selbst.

5.1.1 Methodik

Zu jeder Versuchskonfiguration muss die Testperson ihre Meinung zu je drei Wertungskritierien zu dem Beschleunigungsprozess und dem Bremsprozess abgeben. Während der Testphase wurden die Probanden darüber im Dunkeln gelassen, was eine spezifische Konfiguration bewirkt. Die Kriterien sind dabei wie folgt:

> **Verzögerung** Ist eine Verzögerung zwischen der Eingabe durch Nutzer und der Reaktion auf diese wahrnehmbar? Wertung zwischen *Nicht Wahrnehmbar* und *zu Hoch*. Optimalwert *Nicht Wahrnehmbar*.

Rückkopplung Ist der in 2.1 beschriebene Effekt wahrnehmbar? Zum Beispiel bricht eine Beschleunigung ist konstanter Frequenz ab und läuft wieder an? Wertung zwischen *Nicht Wahrnehmbar* und *zu Hoch*. Optimalwert *Nicht Wahrnehmbar*.

Wirkung Wie stark reagiert die Steurung auf die Eingabe des Nutzers? wird außreichend stark beschleunigt oder zu aggressiv beschleunigt? Wertung zwischen *Nicht Wahrnehmbar* und *zu Hoch*. Optimalwert ist der Mittelwert, also nicht zu stark und nicht zu schwach.

Zusätzlich ist ein optionales Feld für Freitext vorhanden.
Innerhalb der Versuchsphase haben 10 Personen den Fragebogen vollständig segfault.

5.1.2 Grafische Darstellung

5.1.3 Sensordaten während der Umfrage

Zusätzlich zu der Befragung wurden die Sensordaten gespeichert. Mit diesen kann beobachtet werden, wie verschiedene Fahrmanöver aus der Perspektive des Mikrocontrollers dargestellt werden.

Exemplarisch wird verglichen, wie zwei Personen mit unterschiedlicher Körpergröße und entsprechend abweichender Masse das Longboard in der trivialen Konfiguration nutzen. Die Abbildungen 5.4 und 5.5 zeigen die Messdaten. Zu Beginn ist erkennbar wie die Differenz, in rot dargestellt, deutlich in den negativen Bereich wandert. Ursache hierfür liegt in der in Abschnitt 2.1 beschriebenen Nutzungsrichtlinie. Die folgenden Oszillationen sind jeweils Beschleunigungen und Bremsmanöver.

Besonders interessant sind die unterschiede der beiden Testpersonen. Es fällt auf, dass die Person mit dem größeren Körpergewicht auch deutlich stärkere Ausschläge in den Messverwerten verursacht.

5.1.4 Ergebnis: Fernbedienung

Entsprechend der Erwartung sind die Effekte der Verzögerung und Rückkopplung nach Meinung der Nutzer nicht vorhanden. Dies gilt sowohl für die Beschleunigung als auch den Bremsvorgang.

Auch bei dem Aspekt der Wirkung sind sich die Nutzer weitestgehend einig. Die Wirkung ist sehr direkt, weshalb Nutzer mit zu weniger Erfahrung dazu neigen, diese als zu stark wahrzunehmen. Ein deutlicher Trend ist, dass die Empfindung der Bremswirkung schwächer ausfällt, als die der Beschleunigung.

Die positiven Bewertungen der Steuerung über die Fernbedienung war erwartet und dient als Kontrolle für die erarbeitete Steuerung. Zusätzlich war dieser Test als Aufwärmphase für die unerfahrenen Nutzer gedacht.

5.1.5 Ergebnis: Triviale Steuerung

Als triviale Steuerung wird die Konfiguration bezeichnet in welcher ein minimaler Filterungsaufwand betrieben wird, um die Sensorsignale in Bewegung umzusetzen. In dieser wird werden die rohen Daten lediglich über einen kleinen Zeitraum gemittelt, um Rauschen aus dem Signal zu entfernen 4.3.1.

Die Rückmeldung der Nutzer war gespalten. Ein Teil der Nutzer empfand die Rückkopplung als stark ausgeprägt und störend. Wiederum Andere schienen diese nicht zu bemerken. Die Wirkung wurde als mittig, also optimal, bis zu schwach bewertet. Auch in dieser Konfiguration wurde die Bremswirkung im Vergleich zum Beschleunigen als zu schwach empfunden.

5.1.6 Ergebnis: Fenstermechanismus

Diese Konfiguration basiert auf der trivialen Konfiguration, bei welcher der Fenstmechanismus 4.3.4 zur Filterung der Eingabesignale aktiviert ist.

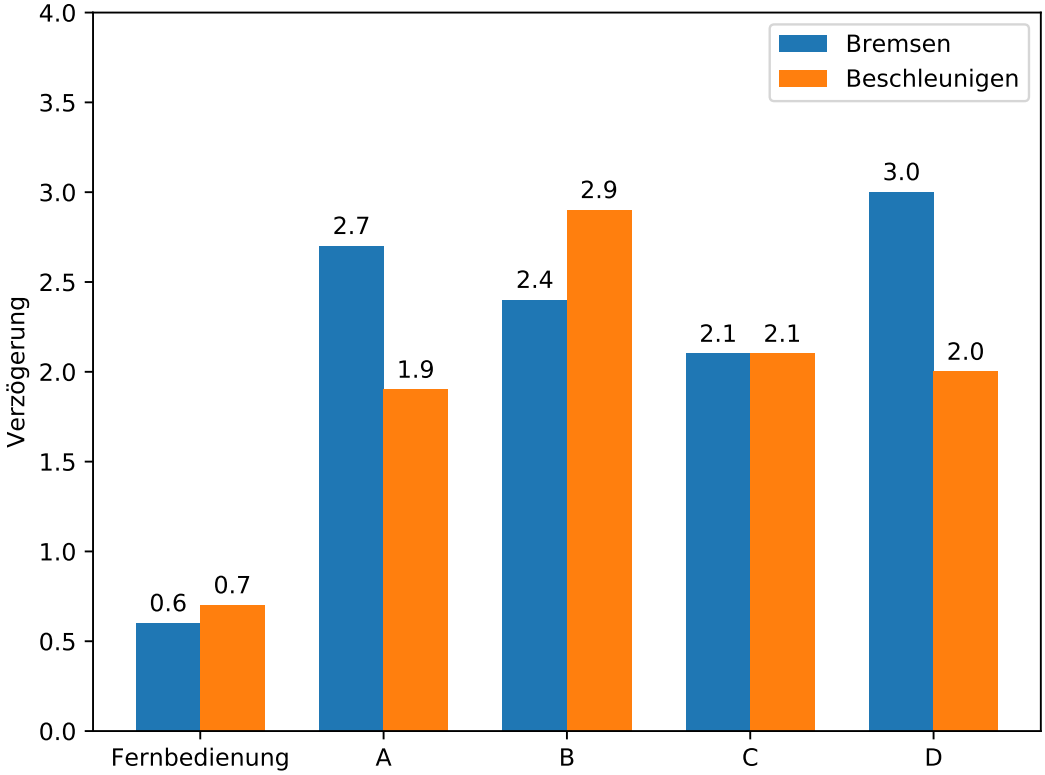


Abbildung 5.1: Mittelwerte aus der Befragung bezüglich der Verzögerung im Vergleich.

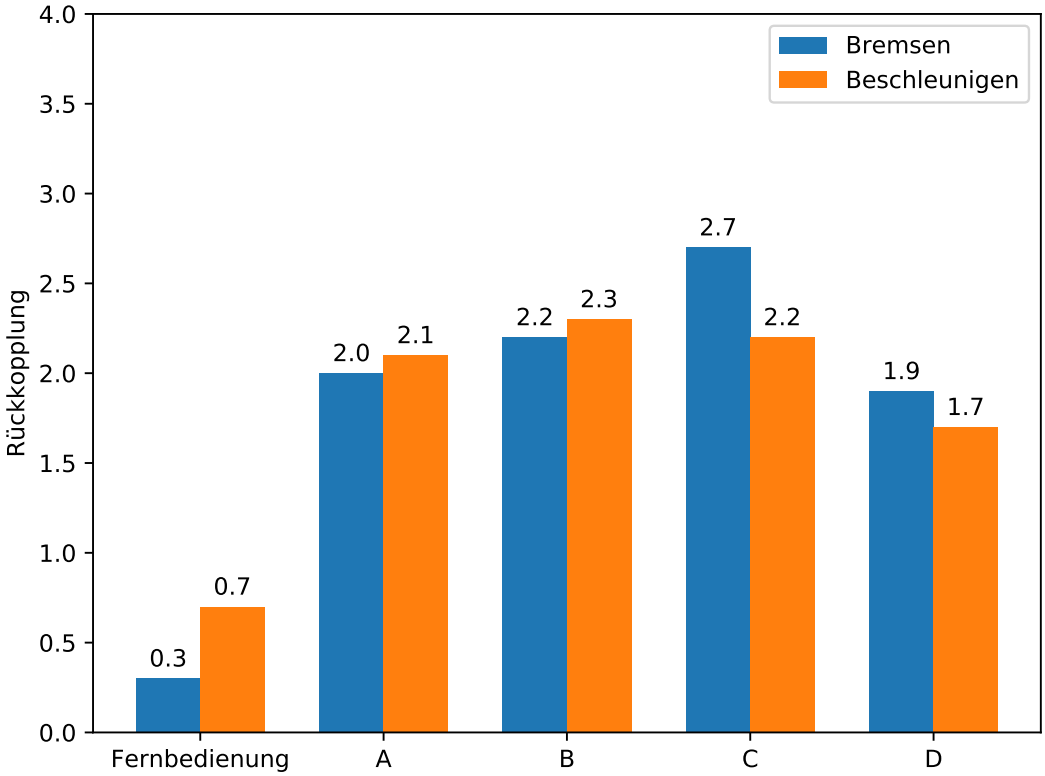


Abbildung 5.2: Mittelwerte aus der Befragung bezüglich der Wirkung im Vergleich.

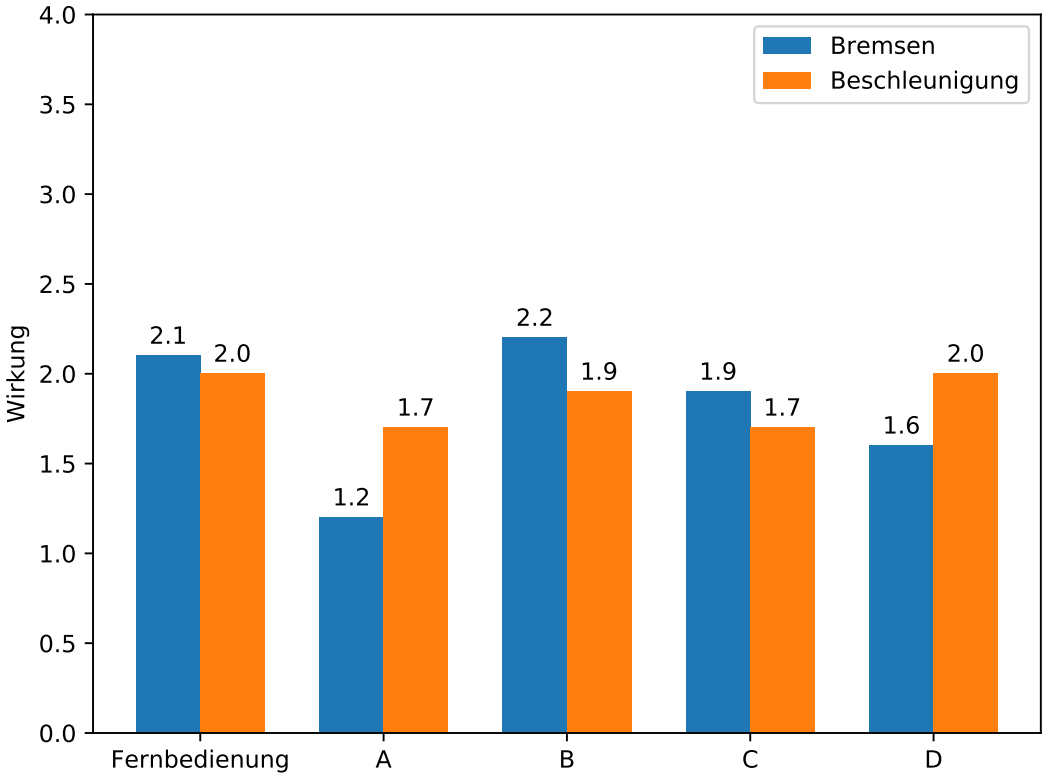


Abbildung 5.3: Mittelwerte aus der Befragung bezüglich der Wirkung im Vergleich.

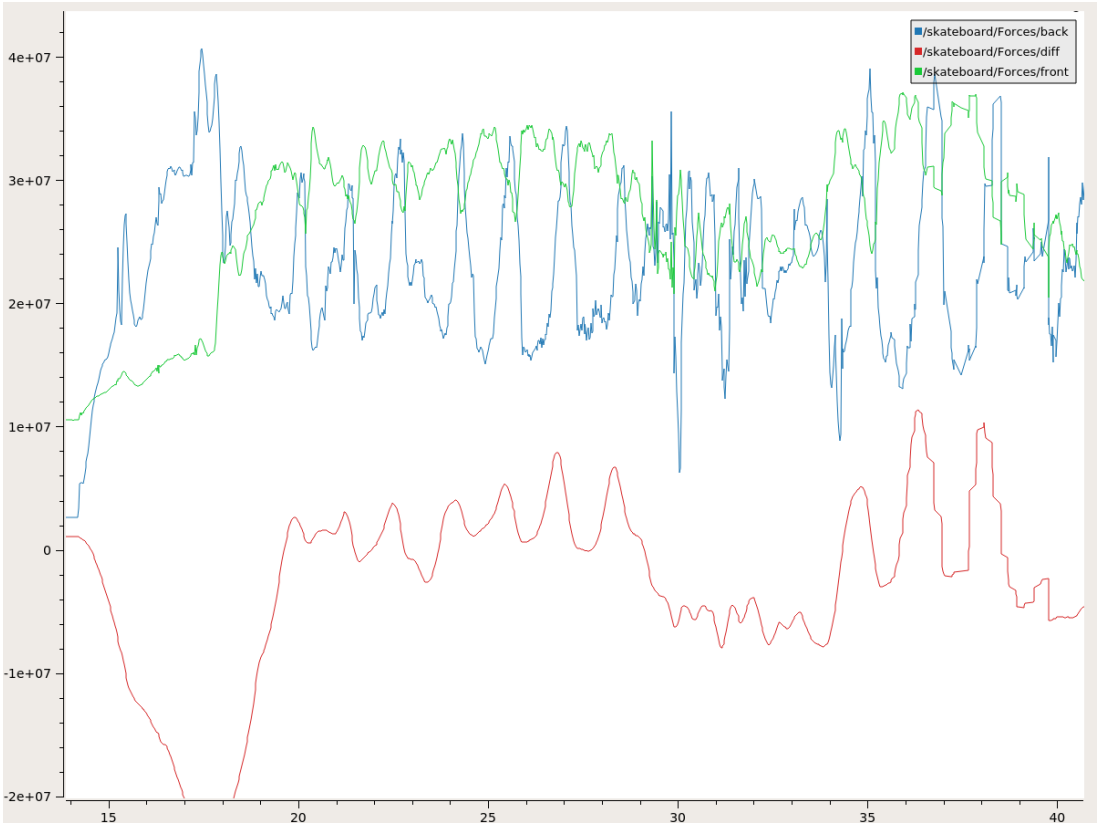


Abbildung 5.4: Sensordaten während der Fahrt mit der trivialen Konfiguration mit einer Testperson mit einem Gewicht von 79 Kg. Differenzwert in Rot, Frontsensor in Grün, Hecksensor in Blau.

Abbildung 5.5: Sensordaten während der Fahrt mit der trivialen Konfiguration mit einer Testperson mit einem Gewicht von 101 Kg. Differenzwert in Rot, Frontsensor in Grün, Hecksensor in Blau.

Zwei Trends wurden durch die Befragung festgestellt. Zum Einem erhöht sich die Empfindung der Verzögerung zwischen Eingabe und Reaktion deutlich. Zum Anderen Empfand ein Teil der Testpersonen, dass die Rückkopplung schlechter wurde während andere genau das Gegenteil empfunden haben. Eine Änderung bei der Wirkung ist im Mittel der Ergebnisse nicht aufgetreten.

5.1.7 Ergebnis: Reaktionskurve

In dieser Konfiguration sind alle Parameter identisch zu dem der trivalen Steuerung. Zudem ist der Mechanismus der in Abschnitt 4.3.4 beschriebene Reaktionskurve aktiviert.

In der Befragung gab ein großer Teil der Befragten an, dass die empfundene Intensität der Rückkopplung höher als die triviale Steuerung war. Zusätzlich empfanden die meisten Testpersonen, dass die Verzögerung relativ größer oder unverändert blieb. Diese beiden Erkenntnisse gelten sowohl für das Bremsen als auch das Beschleunigen.

Der Effekt der Wirkung ist nach dem Ergebnis der Befragung von allen Konfigurationen der Beste. Ein großer Teil der Werte lag auf dem Optimalwert.

5.1.8 Ergebnis: Fenstermechanismus mit Reaktionskurve

Die letzte Konfiguration kombiniert die Reaktionskurve mit dem Fenstermechanismus. Zwecks der Vergleichbarkeit sind wiederum alle anderen Parameter identisch zu der trivialen Steuerung.

Da die beiden Mechanismen einzeln jeweils das Fahrerlebnis bezüglich Verzögerung und Rückkopplung beeinträchtigen, sollte die Kombination ähnliches zeigen. Diese Annahme stellte sich als falsch heraus. In Kombination scheinen die Algorithmen keine Verschlechterung bezüglich der Verzögerung und eine Verbesserung relativ zu der trivialen Konfiguration.

Bezüglich der Wirkung wurde in fast allen Fällen das hohe Niveau der Reaktionskurve gehalten. Die Abweichungen von diesen Trend Werten die Wirkung der Beschleunigungs als leicht schwächer relativ zu der Reaktionskurve.

5.1.9 Schlussfolgerungen

In diesem Abschnitt werden mögliche Ursachen für die Rückmeldungen diskutiert. Im Anschluss wird erwägt, ob und wie diese Beobachtungen zur Verbesserung der Software beitragen könnten.

Die Reaktionskurve bewirkt, dass Eingabe an nahe der maximal Erwarteten deutlich stärker gewertet werden. Dies kann eine Ursache für eine stärker Empfundene Rückkopplung sein. Der durch zum Beispiel eine Beschleunigung bewirkte Impuls bewirkt, dass das Eingabesignal in die entgegengesetzte Richtung wandert. Wenn allerdings zur Werte nahe der Ränder einen großen Effekt haben, wird das Beschleunigen mit der Reaktionskurve schneller abgebrochen, als mit der trivialen Einstellung. Mit diesem Verhalten kann die als stärker empfundene Rückkopplung erklärt werden.

Ein subjektiv wahrgenommene Anstieg der Verzögerung kann durch die Funktionsweise des Fenstermechanismus erklärt werden. Um kleine Schwankungen zu filtern wird in Kauf genommen, dass auch kleine Anpassungen der Geschwindigkeit eine große Veränderung der Position verlangen.

Die Schwankungen in der Wahrnehmung bezüglich der Rückkopplung könnte auf die Parametrisierung des Algorithmus zurückführbar sein. Die Software ist bereit kleinere Schwankungen zu ignorieren, in der Erwartung, dass diese nur Messungen der Trägheit des Körpers der Testpersonen ist. Durch die Einstellung der Fenstergröße kann die Menge der erwarteten Trägheit definiert werden. Verschiedene Nutzer können durch abweichende Körpermasse oder Erfahrung auf Long- oder Skateboards unterschiedliche Schwankungen produzieren. Fallen diese innerhalb der Fenstergröße, ist während des Tests eine Verbesserung spürbar. Im Falle das dies nicht der Fall ist verstärkt das Fenster mit der Verzögerung vorhandene Rückkopplung. Als Konsequenz aus dieser Beobachtung steht die Abwegung zwischen erhöhter Fenstergröße, welche mehr Nutzer abdeckt, und der daraus resultierenden Verzögerung. Auch eine individuelle Anpassung an Nutzer ist denkbar.

Die Rückmeldung zu der Kombination aus Reaktionskurve und Fenstermechanismus sagt, dass die Verzögerung besser als die der Algorithmen alleine ist. Hierfür kommt sowohl eine technische als

auch eine psychologische Begründung infrage. Erstere ist, dass die während der Testphase verwendeten Parameter auf eben die Kombination der Algorithmen abgestimmt sind. Die Parameter wurden während der Entwicklung der Software empirisch ermittelt. Eine andere Begründung ist, dass die Nutzer im Laufe der Testreihe besser im Umgang mit den Longboard wurden. Um den Lernfaktor aus der Messung zu eliminieren könnten etwa verschiedene Testpersonen in die Konfigurationen in zufälliger Reihenfolge absolvieren. Für die im Rahmen dieser Arbeit absolvierten Umfrage waren dafür aber nicht genügen Teilnehmer vorhanden. Ein dritter Grund für diese Beobachtung wäre also, dass die Methodik der Umfrage fehlerhaft war um statistisch aussagekräftige Ergebnisse zu produzieren.

5.2 Ausblick

Dieser Abschnitt soll betrachten, welche Probleme innerhalb dieser Abschlussarbeit nicht behoben oder neu entdeckt wurden. Zusätzlich wird betrachtet, wie zukünftige Arbeiten das Ergebnis dieser erweitern können.

Ein Mangel, der erst während der Durchführung der Arbeit auftrat, ist die Verwendung einer modifizierten Fernbedienung in der Motorsteuerung. Besonders problematisch ist der Wechsel zwischen der Gewichtssteuerung und der Steuerung über die Fernbedienung. Für den Wechsel ist musst jeweils eine Fernbedienung mit dem Longboard gepairt werden. Es wäre sinnvoll Zeit zu investieren, um den Motor direkt anzusteuern. Eventuell könnte so auch eine parallele Nutzung von beiden Methoden möglich werden.

Ein großes Thema für die möglichen Verbesserungen ist die Entwicklung der Hardware. Für die noch in Plastik gefertigten Komponenten könnte ein Ersatz aus Metall gefunden werden.

Auch die aktuelle Auswertung der Sensoren ist nicht optimal. Das Umschalten des Multiplexers (Abschnitt 4.1) führt zu unangebrachter Verzögerung. Daher wäre die Verwendung eines Messverstärkers, der ein solches Verhalten nicht benötigt, geboten.

Wie in dem Abschnitt 2.1 geschrieben, ist das Aufsteigen auf das Longboard aktuell bei falscher Benutzung gefährlich. Da die Auswertung der Kraftsensoren alleine keine verlässlichen Daten liefert, um ein fehlerhaftes Aufsteigen zu deuten, muss eine Alternative gefunden. Eventuell können sensoren gefunden werden, mit welchen die Präsenz eines Fußes gemessen werden kann. Dabei muss nur eine binäre Aussage getroffen werden.

Während der Arbeit und dem Testen des Longboards war dieses auch in kleinere Unfälle verwickelt. Bei einer Kollision mit einer Wand verbog sich eine Sensorhalterung so stark, dass der Sensor völlig falsche Werte lieferte. Es ist denkbar, dass Unfälle aber auch Stöße und Kantsteinen Fehlerproduzieren, die das Board veranlassen, mit permanent vollem Tempo zu fahren.. Während einer Fahrt unter freiem Himmel würde dieser Umstand riskieren, dass das Board sich von den Nutzern entfernt und nicht mehr eingeholt werden kann. Personen und Sachschäden sind so denkbar. Bevor das Fahrzeug auf einem nicht begrentzen Gelände gefahren werden kann, sollte dringend über die Implementation einer Totmanneinrichtung nachgedacht werden.

Literatur

[1] Sarah Al-Mutlaq. *Getting started with load cells*. 2017.

[2] *Arduino Nano 33 IOT, technische Informationen*. URL: https://store.arduino.cc/arduino-nano-33-iot (besucht am 10. 10. 2019).

[3] *C++ preprocessor conditional inclusion*. URL: https://en.cppreference.com/w/cpp/preprocessor/conditional (besucht am 13. 12. 2019).

[4] *Lessons In Electric Circuits – Volume I Chapter 8 DC METERING CIRCUITS - Bridge Circuits*. URL: http://www.ibiblio.org/kuphaldt/electricCircuits/DC/DC_8.html (besucht am 24. 10. 2019).

[5] *Load Cell and Strain Gauge Basics*. URL: https://www.800loadcel.com/load-cell-and-strain-gauge-basics.html (besucht am 24. 10. 2019).

[6] *Pulsweitenmodulation - RN-Wissen.de*. URL: http://www.htc-sensor.com/products/151.html (besucht am 24. 10. 2019).

[7] *rosbag documentation - ROS Wiki*. URL: http://wiki.ros.org/rosbag (besucht am 13. 12. 2019).

[8] *rqt_reconfigure documentation - ROS Wiki*. URL: http://wiki.ros.org/rqt_reconfigure (besucht am 13. 12. 2019).

[9] *TAS606 Button type compression load cell*. URL: http://www.htc-sensor.com/products/151.html (besucht am 11. 10. 2019).

Git-Repository mit Quelldateien dieser Arbeit.
https://github.com/hmirko/longboard-shared-files

- Definitionen der Sensor- und Controllerhalterung in dem OpenSCAD-Format
- Quellcode der Software des Mikrocontrollers
- Schaltplan des Projekts als KiCad-Projekt
- ROS Arbeitsumgebung mit den Entwickelten Werkzeugen zur Datenerfassung
- Ausgewählte mit ROS-Bag erhobenen Messreihen

Fragebogen zu den Fahrreigenschaften

Auf diesem Dokument befindet sich eine Reihe von Fragen bezüglich der Fahreigenschaften eines elektrischen Longboards. Dieses wurde im Rahmen einer Bachelorarbeit mit einer alternativen Steuermethode ausgestattet, bei welcher durch die Verlagerung des Gewichts Beschleunigt und gebremst wird.

Allgemeine Informationen zu der Steuerung
Grundsätzlich wertet die Steuerung die Differenz der Belastung auf der Vorder- und Hinterseite des Boards aus.

Angababen zur Person und Parameter

Name: _____ **full_throttle**: _____

Größe: _____ **full_break**: _____

Gewicht: _____ **gain**: _____

1 Eigenschaften der Konfiguration Fernbedienung

Bremsen

1a. Verzögerung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

1b. Rückkopplung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

1c. Wirkung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

Beschleunigen

1d. Verzögerung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

1e. Rückkopplung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

1f. Wirkung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

Besondere Anmerkungen

1g. _____

_____

_____

2 Eigenschaften der Konfiguration A

Bremsen

2a. Verzögerung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

2b. Rückkopplung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

2c. Wirkung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

Beschleunigen

2d. Verzögerung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

2e. Rückkopplung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

2f. Wirkung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

Besondere Anmerkungen

2g. _____

_____

_____

3 Eigenschaften der Konfiguration B

Bremsen

3a. Verzögerung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

3b. Rückkopplung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

3c. Wirkung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

Beschleunigen

3d. Verzögerung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

3e. Rückkopplung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

3f. Wirkung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

Besondere Anmerkungen

3g. _____

_____

_____

4 Eigenschaften der Konfiguration C

Bremsen

4a. Verzögerung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

4b. Rückkopplung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

4c. Wirkung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

Beschleunigen

4d. Verzögerung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

4e. Rückkopplung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

4f. Wirkung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

Besondere Anmerkungen

4g. _____

_____

_____

5 Eigenschaften der Konfiguration D

Bremsen

5a. Verzögerung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

5b. Rückkopplung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

5c. Wirkung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

Beschleunigen

5d. Verzögerung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

5e. Rückkopplung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

5f. Wirkung Nicht Wahrnehmbar □—□—□—□—□ Zu Hoch

Besondere Anmerkungen

5g. _____

_____

_____