

BACHELOR ARBEIT

Steuerung eines Elektrofahrzeugs durch die Drehmomente der Räder und Regelung im Anhängerbetrieb

Victor Maier

Gutachter: Prof. Dr. Christoph Scholl

Betreuer: Andrea Ghezzi



Vorgelegt am 17.08.2023

ERKLÄRUNG

Hiermit erkläre ich, dass ich diese Abschlussarbeit selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet habe und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen wurden, als solche kenntlich gemacht habe. Darüber hinaus erkläre ich, dass diese Abschlussarbeit nicht, auch nicht auszugsweise, bereits für eine andere Prüfung angefertigt wurde.

Freiburg 17.08.2023

Ort, Datum



Unterschrift

Inhaltsverzeichnis

1.	Einleitung.....	5
2.	Verwandte Arbeiten.....	5
3.	Aufbau des Fahrzeuges	6
3.1.	Hardware.....	6
3.1.1.	Achsen	6
3.1.3.	Regler.....	12
3.1.3.	Akku.....	12
3.1.4.	Sensoren.....	13
3.1.5.	Hauptprozessor	13
3.2.	Software	13
3.2.1.	Regler.....	13
3.2.2.	Hauptprozessor	14
3.3.	Eingabekonzepte	14
3.3.1.	API.....	14
3.3.2.	ADC	14
3.3.3.	Konsole	14
3.3.4.	Gamepad	15
3.3.5.	RC.....	15
3.4.	Regelungen.....	15
3.4.1.	Eingabefilterung (ADC)	15
3.4.2.	Allrad Regelung.....	16
3.4.3.	Lenkregelung	17
4.	Erweiterung des Fahrzeugs mit Anhänger	20
4.1.	Mechanische Teile.....	20
4.2.	Simulation.....	21
4.2.1.	Modellierungen des Anhängers	22
4.2.2.	Modellierung der Zugmaschine.....	25
4.3.	Stabiler Anhängerwinkel	26
4.3.1.	Lineare Näherung des stabilen Winkels	27
4.4.	Schutzschaltung für den Hänger	27
4.5.	Rückfahrregelung	28

4.5.1.	Detektierung für den Anhänger	30
4.6.	Modi im Anhängerbetrieb.....	30
4.6.1.	Kein Anhänger	30
4.6.2.	Anhänger Schutz.....	30
4.6.3.	Anhängersteuerung.....	30
4.7.	Messwerte der Linearregelung	30
4.8.	Entwicklungprobleme mit dem ESP32	32
5.	Fazit	34
6.	Ausblick.....	34
7.	Quellen	34
	Abbildungsverzeichnis.....	36

1. Einleitung

In dieser Arbeit geht es darum, ein neues Steuerungskonzept für Elektrofahrzeuge zu entwickeln und zu testen.

Zunächst wird ein kurzer Blick auf ähnliche Arbeiten geworfen.

Dann wird in Teil 3 die Konzeption der Lenksteuerung des Elektrofahrzeugs beschrieben. Dafür wurde ein Bobbycar an den Rädern der Lenkachse mit einzeln ansteuerbaren Elektromotoren versehen. Zusätzlich wurde die Lenkachse noch mit einem Winkelsensor ausgerüstet. Mit diesem Aufbau lässt sich über eine gezielte Ansteuerung der Räder und das Regulieren des Drehmoments ein Lenkwinkel anfahren ohne das Einwirken eines weiteren Agenten. Neu an diesem System ist, dass trotz des Verzichts auf einen gesonderten Aktuator an der Lenkmechanik zu jedem Zeitpunkt ein beliebiger Lenkwinkel angesteuert werden kann.

Das Fahrzeug wird dann mit einer Fernsteuerung erweitert. Dafür wird eine Jumper T-Pro mit ELRS eingesetzt.

Der vierte Teil befasst sich damit, wie mittels Fernbedienung das Fahrzeug beim Rückwärtsfahren gesteuert werden kann. Dafür wird eine Sensorik entwickelt, die den Einschlagwinkel des Anhängers misst. Verschiedene Ansätze zur Simulation des Gespanns werden vorgestellt und dann eine Näherung des Fahrverhaltens über Dreiecke umgesetzt. Zusätzlich wird eine Schutzschaltung für den Hänger notwendig, mit der Schäden durch Kollisionen mit dem Deichsel verhindert werden. Diese wird entwickelt und vorgestellt. Verschiedene Ansätze zur Rückfahrregelung werden vorgestellt und der Lösungsweg dieser Arbeit mittels Lookup-Tabelle und Nutzung des Stabilen Winkels ausgeführt. Die gesamte Arbeit war mit vielerlei Schwierigkeiten mit dem verwendeten Mikrocontroller und dem SDK von diesem verbunden. Dem ist ein eigener kleiner Absatz gewidmet ist.

Die Arbeit schließt mit einem Fazit und Ausblick auf die noch zu lösenden Probleme.

2. Verwandte Arbeiten

Die Recherche ergab nur wenige Arbeiten, die eine ähnliche Problemstellung zum Thema hatten.

In der Arbeit Motion planning for a reversing general 2-trailer configuration using Closed-Loop RRT [1] wurde ein Fahrzeug gebaut und geregelt, das einen Drehschemelanhänger rückwärts regelt und an einen definierten Punkt lenkt. Bei der Arbeit wird eine Lenkmechanik verwendet, die per Servomotor gesteuert wird. Dies unterscheidet sich wesentlich von der Ausgangslage dieser Arbeit mit einer Lenksteuerung durch Drehmomentregelung der Einzelräder.

In der Arbeit Reversing Control of a Car with a Trailer Using the Driver Assistance System [2] wird ein Anhänger rückwärts geregelt. Dabei verwenden die Forscher hier ein Fahrzeug mit einer Lenkregelung, die über ein Servomotor gesteuert wird. Diese Arbeit nutzt somit eine andere Art der Lenkregelung und verfolgt einen anderen Ansatz.

3. Aufbau des Fahrzeuges

Es handelt sich bei dem Fahrzeug um ein umgebautes VW Beetle Bobbycar.



Abbildung 1: Orginales VW Beetle Bobbycar mit Hänger

Bei diesem wurde die Vorderachse abgeschraubt, die Hinterachse großzügig ausgesägt und die Lenkstangenhalterung aufgebohrt. Später wurden neue Aufnahmepunkte für die Vorderachse in die Kunststoffkarosserie eingeklebt.

3.1. Hardware

3.1.1. Achsen

Hinterachse

Die Hinterachse besteht aus zwei großen Kunststoffteilen, die mit 2 ca. 20cm langen 25x25x2mm Aluminiumprofilen verbunden sind. An diesen sind sie mit Epoxidharz und je 2 M8x40 Schrauben mit Muttern befestigt. Die Motoren wurden pro Motor mit einem speziellen Kunststoffteil und je 4 M8x40 Schrauben mit Muttern befestigt.



Abbildung 2: Foto der verbauten Hinterachse

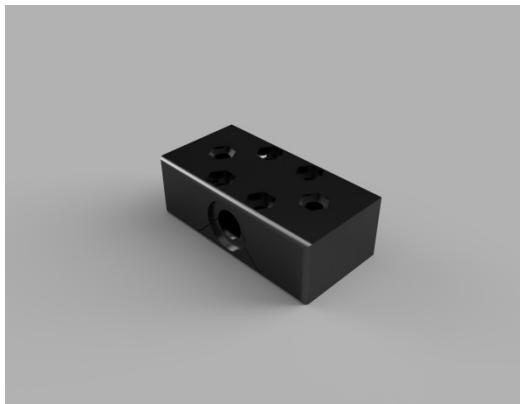


Abbildung 3: Rendering des Seitlichen Hinterachsteil

Die Hinterachse hat keine Überarbeitung und Verstärkung erfahren. Die Achse hat 1.2mm Wandstärke und 15% Infill. Die unteren (auf dem Foto schwarzen) Lagerböcke sind etwas stabiler gedruckt. Die Kunststoffteile haben über die Zeit Materialermüdung erfahren und die Böcke wurden rissig (Abbildung 6 und 7). Diese konnten rechtzeitig getauscht werden und es kam dabei zu keinem Defekt. An der Oberseite der Hinterachse sah der Fall etwas anders aus, da diese nicht getauscht werden konnte, da zu wenig Zeit für einen Ersatz war. Diese ist gebrochen und wurde notdürftig mit Heißkleber repariert, damit das Fahrzeug transportierbar blieb (Abbildung 4 und 5). Die neue Achse ist wieder aus PLA gedruckt dieses Mal aber mit einer Wandstärke vom 6mm und 20% Infill. Damit ist die Stelle über dem Motor massiv und der Motor kann die innere Wand nicht mehr nach oben zerbrechen und somit die Achse schwächen. Die Böcke sind jetzt massiv aus PETG gefertigt.



Abbildung 4: Beschriebener Hinterachschaeden mit notdürftiger Reparatur



Abbildung 5: Oberseite der notdürftig reparierten Achse



Abbildung 6: Die rissigen Achsböcke



Abbildung 7: Die rissigen Achsböcke

Vorderachse

Die Vorderachse besteht aus acht Kunststoffteilen (Abbildung 8-15). Bei der ersten Version sind die unteren Teile der Motoraufnahme gebrochen. Daraufhin wurden der Achsschenkel und die Radaufnahmen neu designet. Der Adapter für die Verbreiterung (Abbildung 8) ist gebrochen, das ist aber kein Problem, da es geklebt wurde. Das Teil ist aber sehr massiv gedruckt, da es in einem Vorversuch schon einmal gebrochen ist. Die Verbreiterung ist mit CA-Sekundenkleber verklebt.

Die Radaufnahme besteht aus zwei Kunststoffteilen, in denen ein Aluminiumeinsatz ist, der die Achse verstärkt. Die Kunststoffteile werden von 10 M8x45 Schrauben zusammengespannt. Das Anzugsdrehmoment liegt bei $5Nm$.

Die Vorderachse wurde komplett überarbeitet, da bei der ersten Version die Motorhalter nachgegeben haben.



Abbildung 8: Rendering des verklebten Adapters zur Verbreiterung

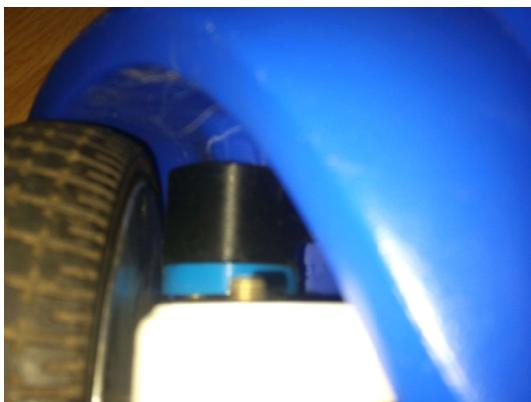


Abbildung 9: Einklebeteil der Vorderachsre (schwarz) mit Spacer (blau)



Abbildung 10: Vorderachse Radaufnahmen, Spurstange und Achsschenkel



Abbildung 11: Rendering der zweiteiligen Radaufnahme



Abbildung 12: Rendering von der Oberseite der Radaufnahme



Abbildung 13: Rendering von der Unterseite der Radaufnahme



Abbildung 14: Rendering des Teils, das als Spurstangen verwendet wird



Abbildung 15: Rendering des Achsschenkels

Benötigtes Material

1,2-1,5Kg 3D Drucker Filament (PLA, ABS oder PETG), je nach Infill auch weniger. Die Teile sind aber mechanisch hoch beansprucht, deshalb ist es nicht ratsam, geringe Wandstärken und wenig Infill zu drucken.

Fertigungsdauer

Ca. 1 Woche Dauerbetrieb mit einem handelsüblichen 3D Drucker (Anycubic Kossel). Dabei wurde eine Geschwindigkeit von 30-45mm/min gewählt. Es wurde ein 0,4mm Nozzle verwendet. Zusätzlich wurde mit einer Wandstärke von 3.2mm und einer Decken und Bodenhöhe von 2.4mm gearbeitet. Für die Füllung wurde 25-30% Dreiecksmuster gewählt. Nur die Datei Spurstangen.stl [1] wurde sehr dünn gedruckt mit 1.2mm Wandstärke und 0.8mm Boden und Deckenhöhe zusammen mit einem Füllungsgrad von 15%. Der Grund ist, dass das Teil keinen Kontakt mit den Motoren hat und somit nicht so stabil ausgelegt sein muss. Die Teile der Lenkstrange sind massiv gedruckt, da sie sehr knapp ausgelegt sind.

3.1.3. Regler

Bei den Motorreglern handelt es sich um Hoverboardregler mit STM32F103 Prozessoren aus der ersten Generation (Abbildung 16). Diese Boards können je 2 bürstenlose Motoren ansteuern. Sie sind für 36V ausgelegt und sind für Ströme bis zu 12A sicher. Sie lassen sich aber auch mit 54V betreiben und halten auch Strömen von 30A stand. Dabei steigt das Ausfallrisiko aber sehr. Aktuell laufen sie mit 20A Strombegrenzung auf 36V Nennspannung.

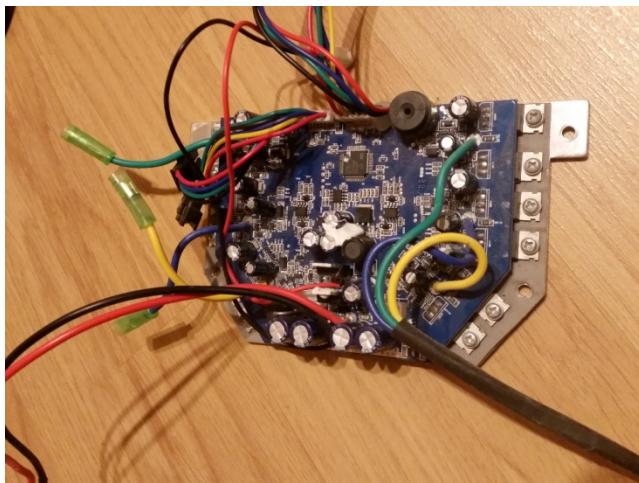


Abbildung 16: Hoverboardregelboard



Abbildung 17: Die aktuell verbauten Regler zusammen mit dem Hoverboardakku (zwei mit Paketband isolierte Hoverboardregler)

3.1.3. Akku

Der Akku war ein problematisches Bauteil an dem Fahrzeug, da zu Beginn der Arbeit ein Hoverboardakku (Abbildung 17) von 2016-2018 montiert war. Dieser musste auf Grund von einem zu großen Verlust der Belastbarkeit auch mehrmals getauscht werden. Am Ende wurde ein E-Bike Akku (Abbildung 18) mit 10s5p verbaut. Dieser leistet 15Ah bei einer Nennspannung von 36V. Der Balancer wurde von einem Hoverboard Akku genommen.



Abbildung 18: Umgebauter Panasonic E-Bike Akku

3.1.4. Sensoren

Das Fahrzeug hat mehrere Sensoren, von denen die Werte abrufbar sind. Es hat Sensoren für die Akkuspannung, den Stromfluss durch die Motoren, die Position der Motoren, die Drehzahl der Motoren und die Temperatur der Regelboards. Zusätzlich ist an der Lenkstange ein Sensor montiert, der den Lenkwinkel messen kann.

3.1.5. Hauptprozessor

Bei dem verbauten Hauptprozessor handelt sich es um einen ESP32 C3 Microcontroller mit einem XTensia Dualcore Prozessor, der auf 160MHz getaktet ist. Die Taktung kann nicht angehoben werden, da der Chip sonst heiß wird und die Befestigung mit Kleber an einem 3D Druck Kunststoffteil nicht besonders viel Hitze aushält. Zusätzlich sind 4MB Flash an den Controller angebunden, um den Programmcode und sonstige relevante Werte zu speichern.

3.2. Software

3.2.1. Regler

Die Regler laufen mit einer modifizierten Version[5], dem hoverboard-firmware-hack-FOC [4] von eferu, der auf der Software [3] von lucysrusch basiert, auf Github. An diesem wurden im Rahmen der Arbeit folgende Änderungen vorgenommen:

1. bldc.c current init
 - a. Problem: Der alte Algorithmus nimmt effektiv nur die letzten 5 Werte des ADC bei 2000 Initialisierungsmessungen
 - b. Erste Lösung: Der neue Algorithmus errechnet den Durchschnitt von 1024 ADC-Messungen. Dieser Lösungsansatz hatte weiterhin das Problem, dass die Lernwerte für die Errechnung des Stromes unpräzise sind, wenn die Motoren in Bewegung sind. Um das zu verbessern gab es eine weiter Überarbeitung.
 - c. Letztendlich eingesetzte Lösung: Es werden jetzt 4096 Messpunkte verwendet und die Messung wird verworfen, wenn sich der Motor mehr wie 4 Grad in dem gegebenen Zeitraum gedreht hat.
2. Eingabeverzögerung (USART Eingabe)

- a. Problem: Die Eingabe wird nur sehr träge an die Motoren übergeben und damit ist die Treibersoftware nicht agil regelbar.
 - b. Lösung: Alle Nachbearbeitungen der Eingabewerte für den USART Eingang an dem Hoverboardmainboard werden deaktiviert und die Motoren werden direkt nach den Eingangswerten geregelt.
3. USART Feedback Speed
 - a. Problem: In der Rückmeldung vom Regelboard hat der Geschwindigkeitswert ein anderes Vorzeichen als der PWM Wert
 - b. Lösung: Die Vorzeichenkorrektur von den Eingabewerten auch auf die Speedwerte im Feedback anwenden.
 4. USART Security
 - a. Problem: Bei Timings-Problemen vom Hauptcontroller konnte es dazu kommen, dass das Fahrzeug die Signale falsch ausgewertet hat und Vollgas gegeben hat.
 - b. Lösung: Die Software auf CRC32-Chechsums umgestellt und gleichzeitig werden Pakete mit Gaswerten größer 1000 und kleiner -1000 ignoriert. (Davor wurden die Werte, die über 1000 waren einfach als 1000 interpretiert und bei Werten kleiner -1000 wurde -1000 als Wert genommen.)

3.2.2. Hauptprozessor

Der Steuerungscontroller hat eine vollständig eigenentwickelte Software [6], die die Regelboards verwaltet und steuert. Dabei ist der Chip mit USART mit den Reglern verbunden und kann damit die Sensoren von dem Board auslesen und gleichzeitig die Motoren steuern. Wichtige Statusinformationen gibt der Controller an ein 0,91 Zoll OLED-Display weiter, damit der Fahrer über die Vorgänge im Fahrzeug informiert ist und mögliche Hardwarefehler schnell erkennen und beheben kann, ohne dass Folgeschäden am Fahrzeug entstehen.

3.3. Eingabekonzepte

3.3.1. API

Softwareintern gibt es Vorbereitungen, um neue Eingabemethoden zu definieren. Mit diesen wird gesichert, dass jede neue Eingabemethode garantiert funktioniert und es keine Fehler beim Setzen, der für die Fahrt relevanten Werten gibt. Die Eingabemethoden sind aber nicht auf die vorgesehenen Funktionen festgelegt, sondern können auf die gesamte Software zugreifen (siehe Konsole).

3.3.2. ADC

Die ADC Eingabeviante hat ein Potenziometer als Gas und Bremse. Bei ADC wird der PID Lenkregler deaktiviert, damit der Fahrer selbst das Lenkrad bewegen kann. Desweiteren wird der Lenkwinkelsensorwert für die Berechnung der Gaswerte für die einzelnen Motoren genutzt.

3.3.3. Konsole

Bei der Konsole handelt sich es um eine selbstprogrammierte Konsole, die über USB oder Bluetooth funktioniert. Diese hat Befehle zum Setzen von Gas- und Lenkwerten. Der PID Regler kann mit dem Werkzeug auch eingestellt werden. Debuggingwerte und kleine Logs können auch ausgelesen werden. Die Befehle vom Setzen der Werte sind aus der API, funktionieren absolut fehlerfrei und sind mehrfach getestet.

Befehlsliste:

help: gibt die komplette Befehlsliste aus
 echo: gibt aus was dahinter steht: „echo hi“ gibt „hi“ aus.
 sets: Setzt einen Lenkwert, der angesteuert wird, wenn „Console“ als Eingang gewählt ist. Genutzt wird der Befehl wie folgt: „sets 11.1“ setzt den Lenkwert α auf $11,1^\circ$. Der Wert muss dabei $-27.5^\circ < \alpha < 27.5^\circ$ sein. Der Parameter ist eine Fließkommazahl.
 gets: Gibt den aktuellen Sensorwert der Lenkung aus.
 getds: Gibt den Lenkwert an, der angesteuert werden soll.
 sett: setzt den Gaswert, wenn „Console“ als Eingang gewählt ist. „sett 51“ setzt den Gaswert t auf 51. Dabei muss $-1000 < t < 1000$ eingehalten werden. Der Parameter ist eine ganze Zahl.
 gett: gibt den Gaswert aus.
 seti: Setzt die Eingabequelle fest. Die Optionen sind: ADC, Gamepad, Console und RC.
 geti: Gibt die aktuelle Eingabequelle aus.
 setm: Setzt den Anhängermodus. Die Optionen sind 0-4.
 getm: Gibt den aktuell aktiven Anhängermodus aus.
 getf: Gibt den Winkel des Anhängers aus.
 getb prüft welche Motorregelungsmainboards verbunden sind und zeigt das an.
 setkp, setki, setkd, getkp, getki, getkd: sind die Befehle, um die Kalibrierungswerte der Lenkung zu ändern und auch wieder auszulesen. Damit können k_p , k_i und k_d im laufenden Betrieb geändert werden.
 getpo: Gibt die Ausgabe des PID Reglers der Lenkung aus.
 getlog: Dumpst die Daten in die Konsole
 setlog: setlog 1 startet die Aufzeichnung, setlog 0 stoppt die Aufzeichnung wieder.
 startlog: Startet die Aufzeichnung von einem Log, um Messerwerte zu gewinnen.
 autolvl: erstellt kp, ki und kd für die Lenkung.
 exec: Führt den Befehl als Parameter aus (für Tests der Konsole).
 reset: Startet den Microcontroller neu.

3.3.4. Gamepad

Bei der Steuerungsmethode wird ein Bluetooth Gamepad mit dem Microcontroller verbunden. Dieser liest von dem Eingabegerät dann regelmäßig die Werte der Knöpfe und Achsen und setzt dann Gas und Lenkwert entsprechend. Da die verwendete Library aktuell leider das Problem hat, dass sie sich sehr unzuverlässig mit einem Gamepad verbindet.

3.3.5. RC

Hier wird ein ELRS Empfänger in das Bobbycar eingebaut. Der sendet per CRSF die Eingabewerte (CH1 Lenkung, CH3 Gas, CH6 Eingangsaktivschaltung) an die Steuerung. Das wird dann per API gesetzt und damit ist das Fahrzeug schon steuerbar. Dabei werden keine Filter für das Signal verwendet, da das die Fernbedienung übernimmt. Das Signal wird direkt verwendet.

3.4. Regelungen

3.4.1. Eingabefilterung (ADC)

Die Eingabefilterung wird über einen Durchschnittswert über $N_{SAMPLES}$ Sensorwerte errechnet. V_{out_x} stellt den Ausgabewert dar. $V_{in_{x+i}}$ bezeichnet einen Sensorwert.

$$V_{out_x} = \frac{\sum_{i=0}^{N_{SAMPLES}} V_{in_{x+i}}}{N_{SAMPLES}}$$

Zur Optimierung und Beschleunigung des Algorithmus wird es mit einer Variablen gemacht, die die Summe enthält.

Initialisiert wird der Wert mit der Summe von $N_{SAMPLES}$ Werten

$$V_{sum} = \sum_{i=0}^{N_{SAMPLES}} V_{tmp_i}$$

und

$$V_{tmp_x} = V_{in_x}$$

Danach wird bei jedem neuen Wert:

$$V_{sum} = V_{sum} - V_{tmp_x}$$

$$V_{sum} = V_{sum} + V_{in}$$

$$V_{tmp_x} = V_{in}$$

$$x = (x + 1) \bmod N_{SAMPLES}$$

$$V_{out} = \frac{V_{sum}}{N_{SAMPLES}}$$

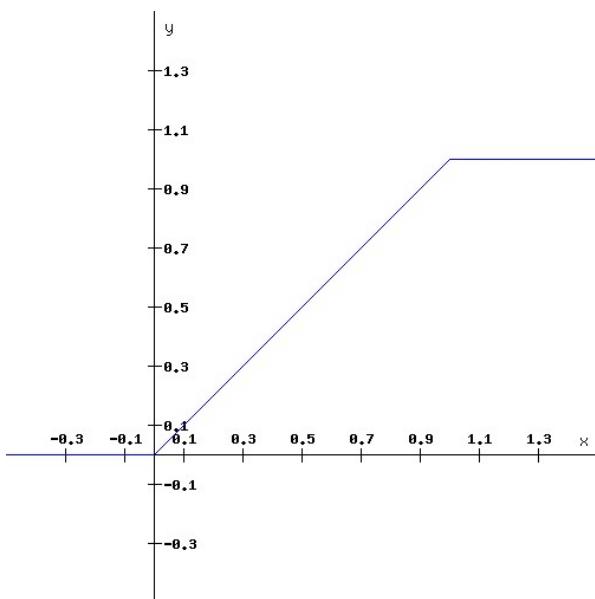


Abbildung 19: Sprungantwort auf eine Änderung des Eingangssignals des Gaspedals

3.4.2. Allrad Regelung

Der Allrad soll danach mit folgenden Funktionen die Wegstrecke der einzelnen Räder errechnen und danach regeln.

$L_{WHEELBASE}$ ist der Radstand des Fahrzeugs. L_{WIDTH} ist die Breite der Hinterachse, diese wird zwischen den Mittelpunkten der beiden Hinterrädern gemessen. $L_{WIDTH_{STEER}}$ ist der Abstand zwischen den beiden Lenkpunkten. $L_{WIDTH_{STEER2WHEEL}}$ ist der Abstand vom Lenkpunkt zum Mittelpunkt des Rads. Diese Werte sind alle konstant.

α ist der Lenkwinkel.

$$V_{bw} = \frac{L_{WHEELBASE}}{\tan(\alpha)}$$

$$V_{bw_0} = \frac{V_{bw} - \frac{L_{WIDTH}}{2} * sign(\alpha)}{V_{bw}}$$

$$V_{bw_1} = \frac{V_{bw} + \frac{L_{WIDTH}}{2} * sign(\alpha)}{V_{bw}}$$

$$V_{fw_0} = \frac{\sqrt{(V_{bw} + L_{WIDTH_{STEER}} * sign(\alpha))^2 + L_{WHEELBASE}^2} + L_{WIDTH_{STEER2WHEEL}}}{V_{bw}}$$

$$V_{fw_1} = \frac{\sqrt{(V_{bw} - L_{WIDTH_{STEER}} * sign(\alpha))^2 + L_{WHEELBASE}^2} + L_{WIDTH_{STEER2WHEEL}}}{V_{bw}}$$

Falls $\alpha = 0$ $V_{fw_0} = V_{fw_1} = V_{bw_0} = V_{bw_1} = 1$

Danach will man noch, dass die Summe der vier Motoren normiert wird. Dabei kann man einen Korrekturfaktor ausrechnen, den man später auf die Gaswerte auf multipliziert.

$$\chi_{correction} = \frac{4}{V_{fw_0} + V_{fw_1} + V_{bw_0} + V_{bw_1}}$$

3.4.3. Lenkregelung

Die Lenkregelung hat folgende Herausforderungen zu meistern:

Die Motoren haben ein Losbrechmoment, das eine Regelung erschwert. Sie bewegen sich nach dem initialen Losbrechen um einiges leichtgängiger. Die Lenkmechanik ist auch schwergängig, da hier gefettete Kunststoffgleitlager verwendet werden. Um das Lenkrad loszubrechen benötigt, man 2 Nm. In der Bewegung fällt der Widerstand, der für die Drehung benötigt wird, ab. Das stellt die Regelung vor das Problem, dass eine kleine Änderung des Lenkwinkels sehr kurze Impulse der Motoren benötigt.

Eine vollständige Bewegung der Lenkung von dem rechten Anschlag zu dem linken Anschlag benötigt lediglich eine Drehung der Motoren von nur 44°. Das macht 22° von der Mitte zum seitlichen Anschlag. Dies kommt daher, dass es sich um BLDC Motoren handelt, die mit 18 Polen eine minimale Regelung in 4° Schritten zulassen. Damit fällt die Möglichkeit weg, über den Motorregler und die Sensoren der Motoren den Lenkwinkel anzufahren. Ein zusätzliches Problem sind die ungünstigen Hebelverhältnisse der Lenkmechanik, die somit auch etwas Schlupf bekommen kann.

Eine weitere Möglichkeit wäre ein PID Regler. Diese Möglichkeit scheitert aber an den Filterfunktionen des EFERU-hoverboard-hacks. Um das Problem zu umgehen, wurde die Software angepasst, alle Eingabefilter deaktiviert und auf eine Filterung im ESP32 Steuerungscontroller zurückgegriffen. Der Regler nutzt als Eingabe den Lenkwinkel α und macht als Ausgabe einen Faktor, der mit dem maximalen Gaswert multipliziert wird und danach auf die Gaswerte der Motoren gerechnet wird.

V_{fw_0} ist der Gaswert des linken Rads, V_{fw_1} ist der Gaswert des rechten Rads.

$$V_{fw_0} = V_{fw_0} + V_{PID}$$

$$V_{fw_1} = V_{fw_1} - V_{PID}$$

Die Werte des PID Reglers wurden auf $k_p = 0,125$, $k_i = 0$ und $k_d = 0$ festgelegt, da diese Werte in der Testreihe am besten funktioniert haben. Um das Losbrechmoment zu überwinden, wird, falls der Ausgangswert $> 0,01$ ist, nochmal 0,05 dazu addiert. Der Ausgang wird auf $\pm 0,15$ limitiert, damit die Lenkung nicht zu schnell in den Anschlag fahren kann und die Mechanik komplett zerstört.

Eine noch vorhandene Schwachstelle ist der Lenkwinkelsensor. Dieser kann Schaden nehmen, wenn die Lenkung und in den Anschlag fährt, und muss getauscht werden. Die Kunststoffmechanik überlebt so einen Fall ohne Probleme.

Bei dem Regelungsprozess entsteht folgende Sprungantwort:

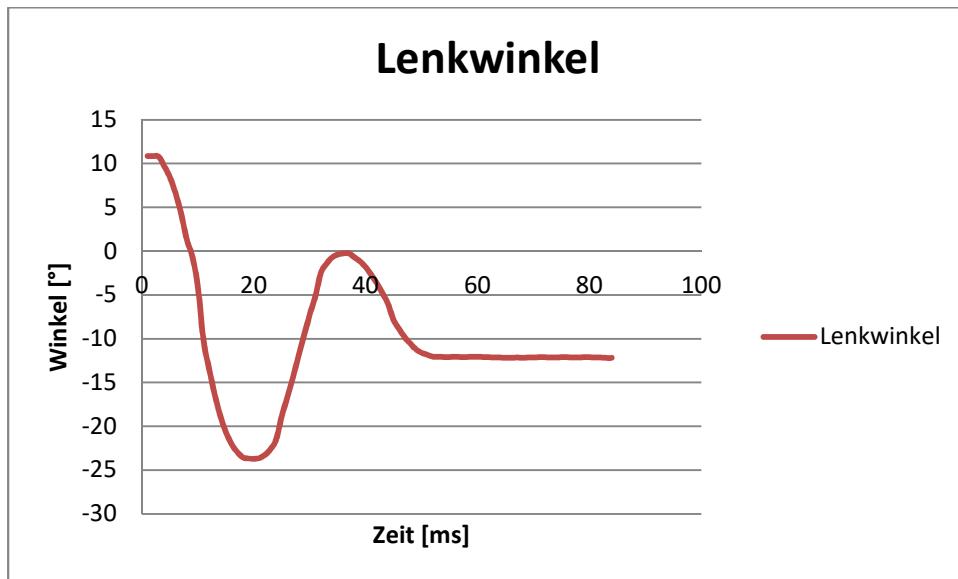


Abbildung 20: Sprungantwort des Lenkwinkels

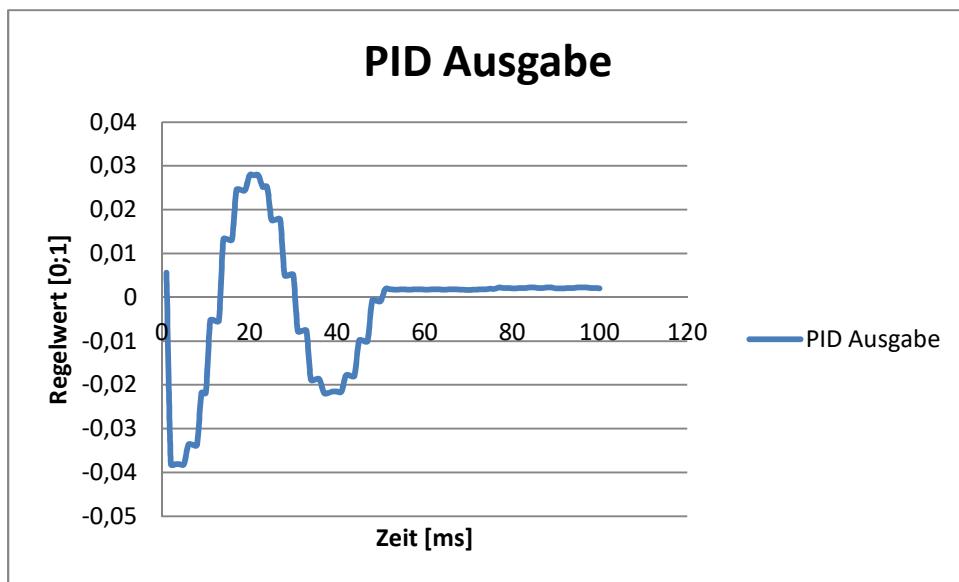


Abbildung 21: Sprungantwort des Regelwertes des P-Reglers

Aus dem Messerwerten lässt sich ablesen, dass der Lenkwinkel nicht perfekt getroffen wird.

4. Erweiterung des Fahrzeugs mit Anhänger

Das Ziel ist es, den Anhänger intuitiv steuerbar zu machen. Das Rückwärtsfahren mit Anhänger ist ohne spezielle Ausbildung nur schwer zu meistern. Für eine benutzerfreundliche Lösung mittels Fernsteuerung soll das Gespann (Antriebsfahrzeug und Anhänger) so reagieren, als ob der Anhänger das Zugfahrzeug wäre. Damit könnte dem Benutzer das komplizierte Regeln des Anhängers abgenommen werden.

Als Anhänger kommt ein Tandemanhänger zum Einsatz, dieser hat eine Distanz von Drehpunkt im Zugmaul zum Mittelpunkt der Achse von 0,6m. Der Anhänger nutzt dabei ein spezielles 3D gedrucktes Zugmaul, das Sensorik beinhaltet zum Messen des Einschlagwinkels des Hängers. Das Zugmaul ist außerdem noch in der Lage, Höhendifferenzen auszugleichen. Das ist notwendig, da ansonsten die Winkelmessung des Zugmauls äußerst komplex und ungenau werden könnte. Der Anhänger kann einen Einschlag von $\pm 15^\circ$ relativ zu dem Zugfahrzeug haben. Eine Überschreitung des Winkels hat einen physischen Schaden am Zugmaul zur Folge.

Der Anhänger wird in das Zugmaul des Bobbycars eingehängt. Dieses wurde davor genau auf 15.0mm aufgebohrt, da bei dem Serienbobbycar das Zugmaul nicht präzise und in der Mitte verjüngt ist. Damit lassen sich keine präzisen Messeinrichtungen entwickeln und anbringen.

4.1. Mechanische Teile

Alle Teile sind aus PETG gefertigt.



Abbildung 22: Rendering des Deichsels mit Sensor und Bolzen

Die Führungsschiene (Abbildung 22) wird an das Bobbycar geklebt und hält damit das Potenziometer ausgerichtet, damit die Schleifscheibe des Potenziometers relativ zum Bobbycar fixiert ist. In der Kunststoffpfanne wird der Sensor fixiert.



Abbildung 23: Rendering der verklebten Führungsschiene

Unterhalb der Pfanne befindet sich der Steckbolzen, der den Anhänger in dem Zugmaul hält. Dieser ist auf die Welle mit dem Schleifkontakt von dem Potenziometer aufgepresst. Da der Bolzen eine Nase hat, die ihn in der Anhängekupplung fixiert, dreht sich der Bolzen mit dem Anhänger mit.

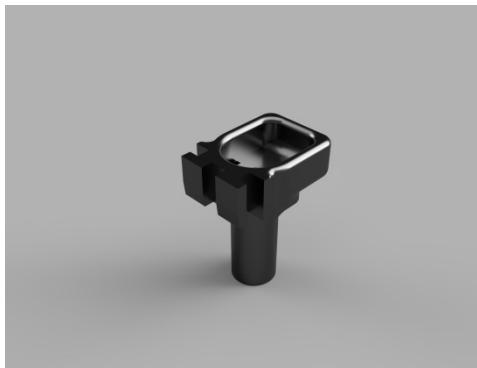


Abbildung 24: Rendering von Kunststoffpfanne und Bolzen



Abbildung 25: Rendering vom Deichsel ohne den Bolzen



Abbildung 26: Rendering vom Bolzen im Deichsel

4.2. Simulation

Der Simulator [7] soll ein Fahrzeug mit einer Lenkachse und einem Tandemanhänger simulieren. Der Simulator soll die Bewegungen in einem 2D kartesischen Koordinatensystem darstellen. Diese Simulation muss nicht nur die Wege des Anhängers präzise genug aufzeigen, sondern auch noch sehr ressourcensparend laufen, damit sie auf einem ESP32 Microcontroller nebenher ausgeführt werden kann. Diese muss in Echtzeit mögliche Lenkwinkel überprüfen und damit einen geeigneten Lenkwinkel finden.

4.2.1. Modellierungen des Anhängers

Um die Bewegung des Anhängers in der Simulation später darzustellen, muss ein mathematisches Modell gewählt werden, was die Bewegung präzise genug schätzen kann und gleichzeitig mit begrenzten Ressourcen ausgeführt werden kann. Dies ist notwendig, da sich das Fahrzeug kontinuierlich bewegt und sich damit die Eingabewerte verändern und bei einem Zeitversatz die Ausgabe an Präzision verliert.

Die Simulation wird schrittweise ausgeführt.

Folgende Werte stehen zur Verfügung:

Die alte Position des Anhängers inklusive dem Winkel mit $x_{Hänger}, y_{Hänger}$ und $\gamma_{Hänger}$.

Sowie die alte und neue Position des Zugmauls mit

$x_{Anhängekupplung_n}, x_{Anhängekupplung_{n-1}}, y_{Anhängekupplung_n}$ und $y_{Anhängekupplung_{n-1}}$.

Anhand dieser Daten soll eine neue voraussichtliche Position für den Anhänger berechnet werden.

Für dieses Problem gibt es mehrere Lösungsansätze.

Modellierung über Splices

Die präziseste Näherung geht über Splices. Damit kann man ein perfektes Modell erstellen, das ist aber mathematisch zu komplex für den ESP32 Microcontroller. Es lässt sich nicht in Echtzeit ausführen. Die Modellierung nimmt die Fahrstrecke des Bobbycars und zieht von jedem Punkt der Fahrstrecke eine Strecke mit der Länge L , die am Endpunkt die Fahrstrecke des Anhängers tangiert. Dabei ist die Fahrstrecke des Hängers kontinuierlich und über die besagten Strecken definiert.

Näherung über Kreisbögen

Bei dieser Näherung ist der Ansatz, dass man davon ausgeht dass sich der Hänger mit einer Aneinanderreihung von Kreisbögen bewegt. Dabei entstehen minimale Abweichungen, da der Hänger spiralförmige Bewegungen macht.

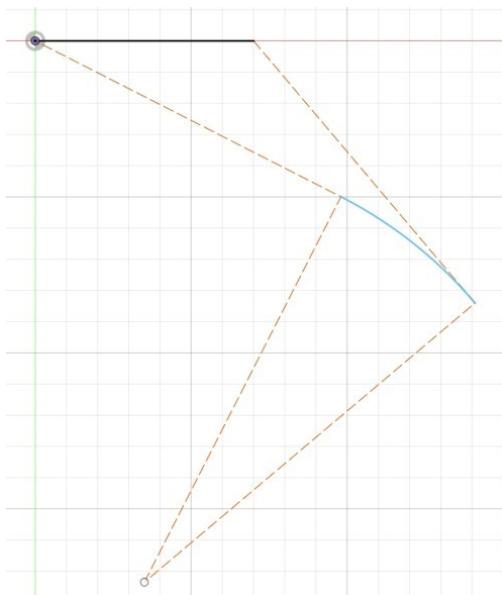


Abbildung 27: Kreisbogen-Näherung Skizze, die Fahrstrecke des Bobbycars ist schwarz dargestellt, die des Anhängers blau.

Näherung über Dreiecke

Bei dieser mathematisch einfachen Näherung liegt die Annahme zu Grunde, dass der Hänger ein kleines Stück gerade zurücklegt und im nächsten Schritt der Winkel korrigiert wird. Dabei ist es wichtig, dass die Schrittweite relativ klein gewählt wird. Das ist aber kein Problem, da die mathematische Komplexität des Modells gering ist. Es gilt auch zu beachten, dass, wenn die Schritte zu klein werden, Abweichungen über die Präzision der Fließkommazahlen entstehen. Aufgrund der technischen Gegebenheiten wurde diese Art der Simulation gewählt.

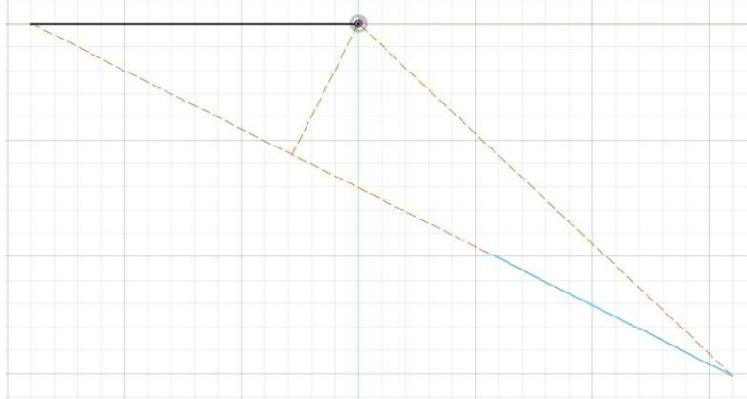


Abbildung 28: Dreiecksnäherung Skizze, die Fahrstrecke des Bobbycars ist schwarz dargestellt, die des Anhängers blau.

Das Programm unterscheidet drei Fälle. Diese werden über den Winkel β_{gerade} unterschieden.

$$\Delta x_n = x_{Anhängerkupplung_n} - x_{Anhängerkupplung_{n-1}}$$

$$\Delta y_n = y_{Anhängerkupplung_n} - y_{Anhängerkupplung_{n-1}}$$

$$\gamma_{xy} = \begin{cases} sign(\Delta y) * \frac{\pi}{2} & \Delta x = 0 \\ \pi & \Delta y = 0 \wedge \Delta x < 0 \\ \tan^{-1}\left(\frac{\Delta y}{\Delta x}\right) & \end{cases}$$

$$\beta_{gerade} = \gamma_{Hänger} - \gamma_{xy}$$

$$s_n = \sqrt{\Delta x^2 + \Delta y^2}$$

$$s_{kurz_n} = \cos(\beta_{gerade}) * L_{Deichsel}$$

Wenn $s_n = 0$ dann hat keine Bewegung stattgefunden, deshalb wird hier abgebrochen.

Fall 1: $|\beta_{gerade}| < 90^\circ$

Wenn $|\beta_{gerade}| < 90^\circ$ ist muss man wieder drei Situationen anhand von s_{kurz_n} unterscheiden.

Situation 1

$$\frac{s_n}{2} < s_{kurz_n}$$

$$h = \sin(\beta_{gerade}) * s_n$$

$$\beta_{gerad\ n+1} = \beta_{gerade_n} + \cos^{-1}\left(\frac{h}{L_{Deichsel}}\right) - \frac{\pi}{2}$$

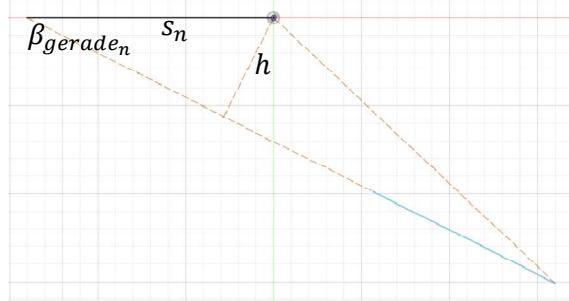


Abbildung 29: Fall 1 Situation 1

Fall 1 Situation 2

$$\frac{s_n}{2} > s_{kurz_n}$$

$$h = \sin(\beta_{gerade_n}) * L_{Deichsel}$$

$$x_{tempLen} = s_n - s_{kurz_n}$$

$$\beta_{gerad\ n+1} = \pi - \tan^{-1}\left(\frac{h}{x_{tempLen}}\right)$$

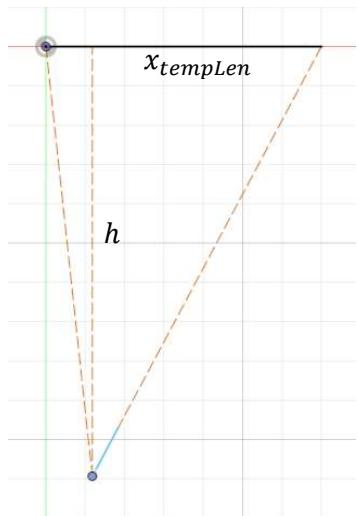


Abbildung 30: Fall 1 Situation 2

Fall 1 Situation 3

$$\frac{s_n}{2} = s_{kurz_n}$$

$$\beta_{gerad\ n+1} = \pi * sign(\beta_{gerad\ n}) - \beta_{gerad\ n}$$

Der Anhänger fährt nicht, sondern ändert nur seine Ausrichtung.

Fall 2: $|\beta_{gerade}| > 90^\circ$

$$h = \cos(\beta_{gerade_n}) * L_{Deichsel} + s_n$$

$$x_{tempLen} = \sin(\beta_{gerade_n}) * L_{Deichsel}$$

$$\beta_{gerade_{n+1}} = -\pi + \tan^{-1}\left(\frac{h}{x_{tempLen}}\right)$$

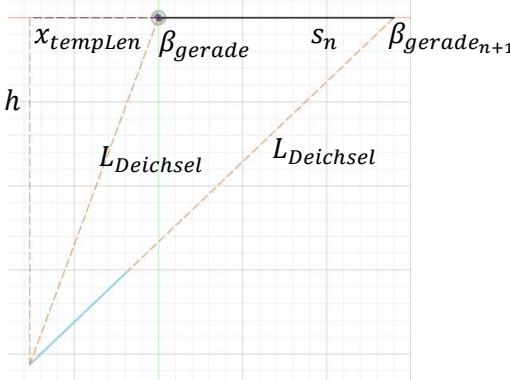


Abbildung 31: Fall 2

Fall 3: $|\beta_{gerade}| = 90^\circ$

$$\beta_{gerade_{n+1}} = \pi - \tan^{-1}\left(\frac{L_{Deichsel}}{s_n}\right) * sign(\beta_{gerade_n})$$

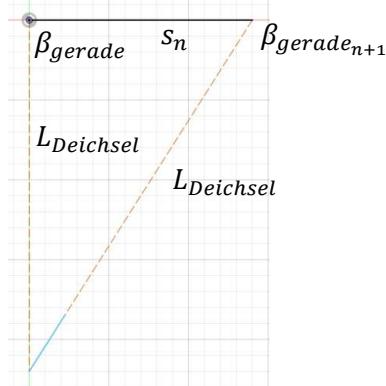


Abbildung 32: Fall 3

4.2.2. Modellierung der Zugmaschine

Das Zugfahrzeug bekommt einen α -Winkel für die Lenkung zugewiesen. Anhand des Lenkwinkels kann das Zugfahrzeug berechnen, an welchem Punkt es steht, nachdem es sich über eine definierte Distanz bewegt hat. Dabei errechnet es auch die neue Ausrichtung im Koordinatensystem. Dabei ist der Eingabewert die Strecke s , die das Fahrzeug zurücklegt.

$$r_{bw} = L_{WHEELBASE}/\tan(\alpha)$$

$$\Delta\gamma = s/r_{bw}$$

$$x_{relativ} = \sin(\Delta\gamma) * r_{bw}$$

$$y_{relativ} = (1 - \cos(\Delta\gamma)) * r_{bw}$$

Am Ende werden die relativen Koordinaten noch auf die realen umgerechnet

$$x = x + \cos(\gamma) * x_{relative} + \sin(\gamma) * y_{relative}$$

$$y = y + \sin(\gamma) * x_{relative} + \cos(\gamma) * y_{relative}$$

$$\gamma = \gamma + \Delta\gamma$$

4.3. Stabiler Anhängerwinkel

Errechnung des stabilen Winkels. Dieser Winkel ist der Zugmaulwinkel β , auf dem sich der Anhänger bei der Vorwärtsfahrt stabilisiert und gleichzeitig der Winkel, auf dem das System bei der Rückwärtsfahrt stabil ist (Abbildung 33).

$$V_{bw}(\alpha) = L_{wheelbase}/\tan(\alpha)$$

$$V_{fbw}(\alpha) = \sqrt{V_{bw}(\alpha)^2 + L_{car2hitch}^2}$$

$$\beta_{const}(\alpha) = \sin^{-1}\left(\frac{L_{car2hitch}}{V_{fbw}(\alpha)}\right) + \sin^{-1}\left(\frac{L_{hitch2axle}}{V_{fbw}(\alpha)}\right)$$

Da die Funktion relativ komplex, aber trotzdem quasi linear ist, lässt sie sich sehr gut über eine Gerade nähern.

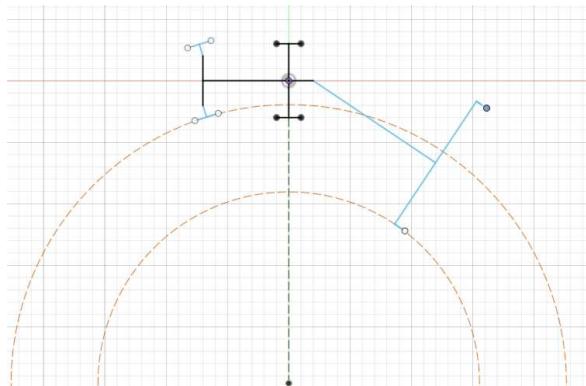


Abbildung 33: Schaubild zum stabilen Winkel

Funktion für den stabilen Winkel

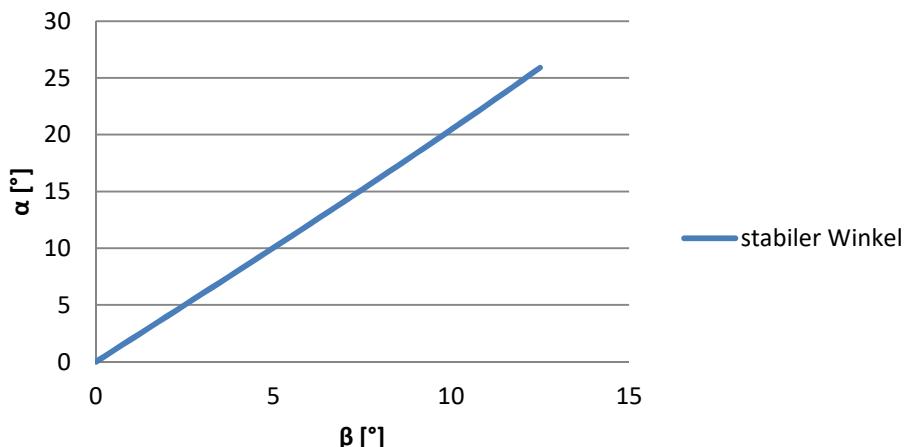


Abbildung 34: Grafik zur Funktion des stabilen Winkels des Anhängers

4.3.1. Lineare Näherung des stabilen Winkels

Für die lineare Näherung wird ein α -Wert, der im Definitionsbereich liegt, gewählt und es wird für diesen Wert der stabile Winkel β ausgerechnet. Mit α und $\beta(\alpha)$ lässt sich eine Konstante berechnen mit der man zu jedem α -Wert einen β -Wert annähern kann.

$$c_{\alpha\beta} = \frac{\beta(\alpha)}{\alpha}$$

$$\sim \alpha_{const}(\beta) = \frac{\beta}{c_{\alpha\beta}}$$

$$\sim \beta_{const}(\alpha) = c_{\alpha\beta} * \alpha$$

lineare Näherung

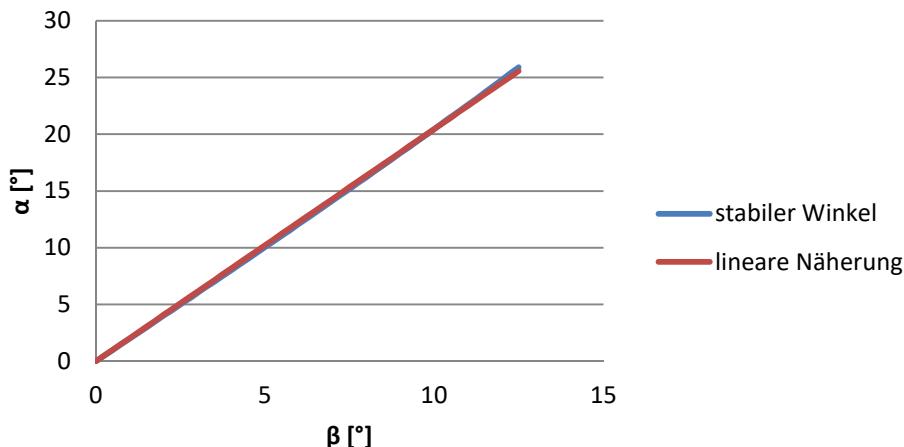


Abbildung 35: Grafik zur linearen Näherung des stabilen Winkels

4.4. Schutzschaltung für den Hänger

Die Funktion bildet \mathbb{R}^3 auf $\{0,1\}$ ab.

α , β und v sind die drei Eingabewerte. Dabei interessiert bei der Geschwindigkeit v nur die Fahrrichtung, da die Funktion dafür da ist, eine Kollision mit dem Deichsel zu verhindern, das heißt, die Weiterfahrt wird unterbunden. Dabei muss natürlich zuerst geprüft werden, ob

$$|\beta| > \beta_{max}$$

Falls das nicht eintritt, ist die Ausgabe 1 und somit kann das Auto weiterfahren. Ist das nicht der Fall, müssen weitere Tests gemacht werden. Falls

$$sign(v) = 1$$

wird geprüft, ob $\beta_{const}(\alpha) < \beta_{max}$ & $sign(\beta_{const}(\alpha)) = sign(\beta)$

In diesem Fall darf das Fahrzeug nicht fahren und die Ausgabe ist 0. Ist dies nicht der Fall, darf das Fahrzeug fahren.

$$sign(v) = -1$$

Wenn $sign(\beta) = sign(\beta_{const}(\alpha))$ & $|\beta| < |\beta_{const}(\alpha)|$ eintrifft, darf das Fahrzeug fahren, da damit garantiert ist, dass der Winkel β wieder kleiner wird und somit eine Kollision vermieden wird. Die Funktion gibt 1 aus. Trifft es nicht ein, ist die Ausgabe 0 und eine Fahrt wird verhindert.

4.5. Rückfahrregelung

Für die Regelung der Fahrt gibt es mehrere Ansätze.

Der komplexeste Ansatz ist es, das mathematische Modell komplett auf dem Microcontroller des Fahrzeugs auszuführen und somit live zu berechnen, was der Hänger voraussichtlich tun wird bei einer definierten Eingabe. Dabei muss mit einer Vorwärtssimulation für jede Nutzereingabe eine Reihe von Simulationen ausgeführt werden, da über das Testen ein neuer Lenkwert genähert und ausgegeben wird. Die Laufzeit ist bei dem Testen von einer Reihe Lenkwinkel linear, kann aber über Optimierung logarithmisch werden. Zusätzlich ist es möglich die Präzision der anfänglichen Simulationen zu senken, um Rechenzeit zu sparen.

Die Optimierung sieht wie folgt aus:

Initialisiert wird der Wert $\Delta\epsilon$. Dies ist die Schrittgrößenänderung.

Der Wert ϵ_{tmp} ist die Schrittgröße.

$$\Delta\epsilon = \frac{\Delta\beta}{2}$$

$$\epsilon_{tmp} = \epsilon$$

Mit c_β und c_ϵ wird die Präzision eingestellt, mit der der Algorithmus arbeitet.

Danach werden folgende Schritte wiederholt.

$$\Delta\epsilon = \frac{\Delta\epsilon}{2}$$

Wenn $\beta_{sim}(\beta_{min} + \epsilon) < \beta_{des}$ $\epsilon = \epsilon_{tmp} + \frac{\epsilon}{2}$ sonst $\epsilon_{tmp} = \epsilon_{tmp} - \Delta\epsilon$. Das wird so lange wiederholt, bis entweder $\epsilon < c_\epsilon$ oder $|\beta_{des} - \beta_{sim}(\beta_{min} + \epsilon)| < c_\beta$.

Damit schrumpft die Laufzeit auf $O(\log(n))$ und ist nicht mehr $O(n)$ wobei $n = \beta_{max} * 2/c_\beta$

Ein weiterer Ansatz ist es, die Simulation für einen Teil der möglichen Eingaben auszuführen. Damit kann eine Tabelle erstellt werden, in der während der Fahrt einfach nachgeschaut werden kann, bei welcher Eingabe welche Ausgabe zu erwarten ist. Da die Tabelle unendlich groß sein müsste, ist dieser Ansatz natürlich nur eine Näherung, um jeden möglichen Eingabewert abzudecken. Das ist aber gar nicht notwendig, da die verwendeten Sensoren nur eine begrenzte Präzision haben und die Tabelle nur so präzise sein muss wie der Sensor.

Ein präzisionsverbessernde Maßnahme wäre, eine Funktion anzunehmen, und zwischen zwei Tabellenwerten zu interpolieren. Dieser Ansatz wurde nicht verfolgt, da genügend Speicher für eine große, fein abgestufte Tabelle vorhanden ist. Die folgenden Formeln sind für die Umrechnung der Werte in die Indexwerte für die Tabelle und wieder zurück. Die Rechnung zurück ist notwendig für die Generierung der Tabelle.

In diesem Fall ist die Tabelle symmetrisch. Dabei lässt sich die Größe der Tabelle halbieren, indem man einen der beiden Indexwerte nur aus positiven Werten generiert. Am Ende lässt sich durch das umdrehen der Vorzeichen das Ergebnis wieder korrigieren.

$T(i_0, i_1)$ ist die Lookuptable.

$$i_{full}(x) = \frac{x * i_{max}}{x_{max} * 2} + \frac{i_{max}}{2}$$

$$i_{half}(x) = \frac{|x| * i_{max}}{x_{max}}$$

$$x_{full}(i) = \left(i - \frac{i_{max}}{2} \right) * \frac{x_{max} * 2}{i_{max}}$$

$$x_{half}(i) = i * \frac{x_{max}}{i_{max}}$$

Wenn $x_0 < 0$ dann $-T(i_{half}(-x_0), i_{full}(-x_1))$ sonst $T(i_{half}(x_0), i_{full}(x_1))$.

Der letzte Ansatz ist der simpelste von allen. Über den stabilen Winkel lässt sich vorhersagen, in welche Richtung der Anhänger wandern wird. Damit lässt sich mit einem P-Regler ein passender Lenkwinkel generieren. Der Vorteil dieses Ansatzes ist, dass wenn K_p abhängig von der Deichsellänge L ist, die Länge der Regelstrecke konstant bleibt.

$$\Delta\beta = \beta_{new} - \beta_{old}$$

$$\alpha_{new} = \alpha_{const}(\beta_{old}) + c_{p_L} * \Delta\beta$$

Wenn $|\alpha_{new}| > \alpha_{max}$ $\alpha = \alpha_{max} * sign(\alpha_{new})$ sonst $\alpha = \alpha_{new}$. Damit wird verhindert, dass das Fahrzeug einen Lenkwinkel einstellen soll, der nicht möglich ist und die Lenkmechanik beschädigt.

c_{p_L} berechnet sich aus L und einen Proportionalfaktor, der bestimmen soll wie stark die Lenkung reagiert.

Die Echtzeitsimulation braucht eine erhebliche Rechenleistung, da der ESP32 bei Fließkommaberechnungen verhältnismäßig langsam ist. Vor allem, da die Winkel teilweise sehr spitz sind und die Strecken sehr kurz sein können. Über die spitzen Winkel und kurzen Kanten wird eine hohe Präzision gefordert. Diese wird über die Verwendung von 64 Bit Fließkommazahlen garantiert, der Preis dafür ist aber nochmals ein erheblich höherer Rechenaufwand, da die FPU des Chips nur 32 Bit Fließkommazahlen unterstützt.

4.5.1. Detektierung für den Anhänger

Für die Erkennung des Anhängers wurde Initial ein $100\text{k}\Omega$ Pulldownwiderstand verwendet. Das Problem war jedoch, dass die Konstruktion mechanisch nicht stabil war und eine sonderangefertigte Platine den Zeitrahmen und den Aufwand der Arbeit schnell gesprengt hätte. Zusätzlich verliert das $10\text{k}\Omega$ Potentiometer einiges an Linearität, durch die Parallelschaltung eines $100\text{k}\Omega$ Widerstands auf den Ausgang. Die $10\text{k}\Omega$ Potentiometer sind die aktuell verwendet Sensoren an dem Fahrzeug. Ein anderes ($1\text{k}\Omega$) an dem Hänger hätte den Nachteil, dass die Wartung problematisch wird, da mehrere verschiedene Potentiometer auf Lager gehalten werden müssen. Als Pulldownwiderstand ein höheren Widerstandswert wie $100\text{k}\Omega$ zu nehmen hat zum Nachteil, dass es schlechter funktioniert, da der Controller einen Eingangswiderstand von $1\text{M}\Omega$ hat.

Die aktuelle Lösung sieht vor, dass das Fahrzeug per ELRS Fernbedienung oder USB-Serial Konsole eingestellt wird und damit eingestellt bekommt, ob ein Hänger verbunden ist und wie mit dem Hänger gearbeitet werden soll.

Dafür gibt es verschiedene wählbare Modi gibt.

4.6. Modi im Anhängerbetrieb

4.6.1. Kein Anhänger

In diesem Modus wird der Anhängerwinkelsensor ignoriert und es besteht kein Schutz vor Schäden am Zugmaul. Dieser Modus ist ausschließlich für das Fahren ohne Anhänger gedacht. Die Lenkregelung ist aktiv, damit das Fahrzeug komplett steuerbar bleibt.

4.6.2. Anhänger Schutz

In diesem Modus wird die Schutzschaltung zugeschalten, die sich darum kümmert, die Weiterfahrt zu untersagen, wenn Anhängerwinkel eintreten, die Schaden hervorrufen können. Die Schutzschaltung rechnet nach, in welche Richtung sich der Hänger bewegen wird und erlaubt weiterhin Bewegungen, die die Situation entschärfen und das Problem auflösen. Somit ist kein äußeres Eingreifen erforderlich.

4.6.3. Anhängersteuerung

Hier wird die Rückfahrregelung zugeschalten. Das heißt, vorwärts steuert der Fahrer wie gewohnt das Bobbycar. Rückwärts ändert sich dann das Lenkverhalten und es wird nichtmehr der Lenkwinkel α kontrolliert, sondern der Anhängerwinkel β . Dabei bleibt die Schutzschaltung weiterhin aktiv, da es sonst möglich wäre, das Zugmaul in der Vorfahrtsfahrt zu beschädigen.

4.7. Messwerte der Linearregelung

Der Versuchsaufbau war wie folgt:

Das Fahrzeug wurde aus einer beliebigen Ausgangsposition heraus auf einer Strecke von 3m mit konstanter Geschwindigkeit rückwärts bewegt und dabei wurden Messwerte erhoben. Aufgezeichnet auf dem Mikrocontroller wurden Fahrzeit, Lenkwinkel, Anhängerwinkel und der Regelungslenkwinkel, den die Steuerung versucht zu erreichen. Dazu wurden für die unterschiedlichen Messreihen unterschiedliche Winkel als Regelungsziel eingestellt. Am Anfang der Fahrt wurde die Aufzeichnung gestartet am Ende wurden die Messwerte von Mikrocontroller auf einen PC übertragen.

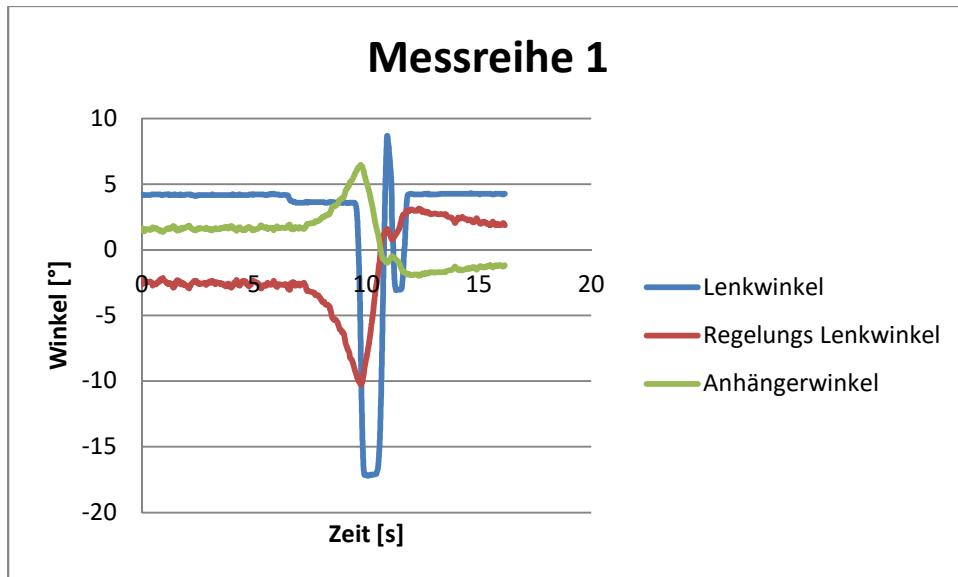


Abbildung 36: Diagramm von den Messwerten der ersten Fahrtaufzeichnung mit 0° angepeiltem Anhängerwinkel

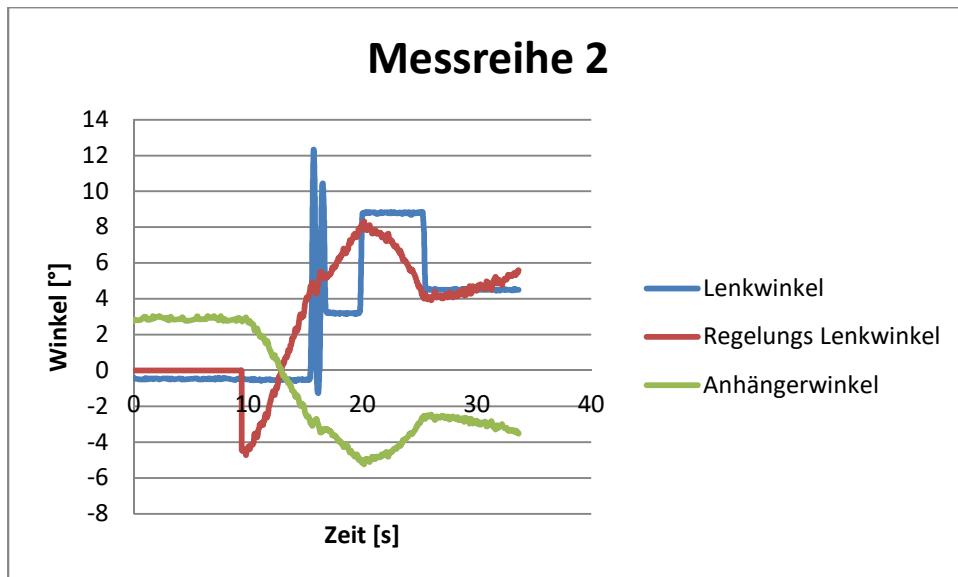


Abbildung 37: Diagramm von den Messwerten der zweiten Fahrtaufzeichnung mit 4° angepeiltem Anhängerwinkel

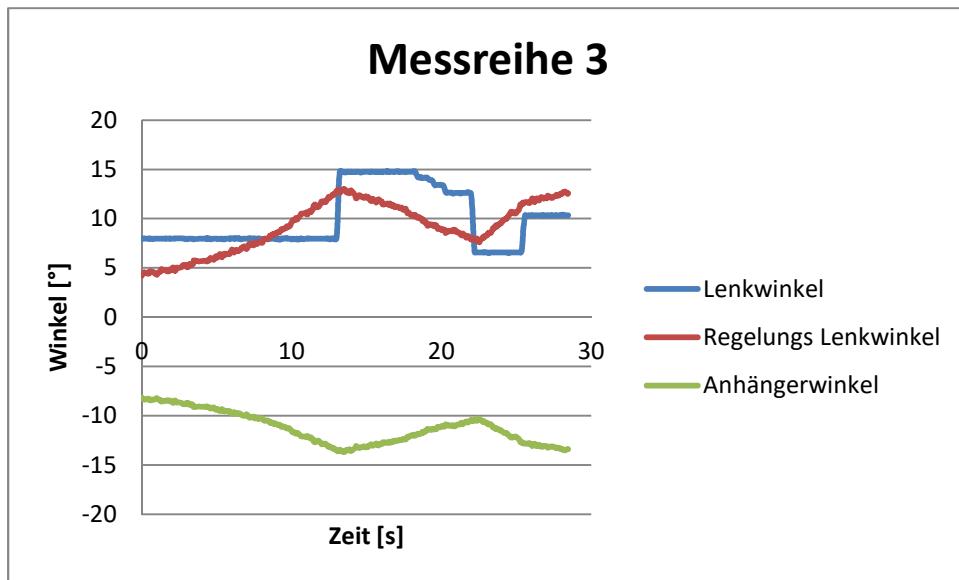


Abbildung 38: Diagramm von den Messwerten der dritten Fahrtaufzeichnung mit 12° angepeiltem Anhängerwinkel

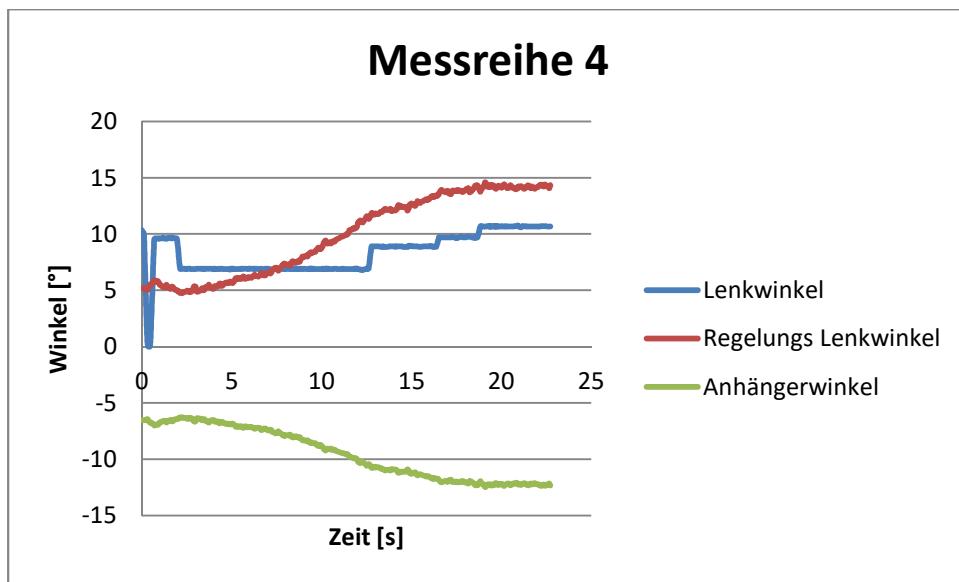


Abbildung 39: Diagramm von den Messwerten der vierten Fahrtaufzeichnung mit 10° angepeiltem Anhängerwinkel

Interpretation der Messdaten:

Bei einer idealen Lenkwinkelregelung liegen Lenkwinkel und Regelungs-Lenkinkel übereinander. Bei der hier gemessenen Regelung zeigt sich, dass diese Steuerung in der Lage ist, einen Lenkwinkel mit einer Genauigkeit von 3° anzusteuern und einzuregeln. Es ist ersichtlich, dass der Winkel aber bis zu einer Differenz von 7° Abweichung zwischen Regelungs-Lenkinkel und tatsächlichem Lenkwinkel stehen bleibt. Erst wenn die Abweichung darüber ist, wird neu eingeregelt.

4.8. Entwicklungprobleme mit dem ESP32

Während dem Projekt gab es mehrmals Probleme mit dem ESP32 Mikrocontroller. Es kam zu Hardwaredefekten. Zusätzlich gab es Probleme mit den verwendeten Bibliotheken. Dazu kamen noch

unerwartete Verhaltensweisen der C Standardbibliothek, die auf Probleme im ESP32-SDK von espressif schließen lassen.

Der Mikrochip wurde mehrfach auf Grund verschiedener Hardwaredefekte ausgetauscht. Bei einem Hardwaredefekt ist der 36V Akku auf das USB Kabel gefallen und hat die USB Buchse von dem Developmentboard abgerissen. Danach wurde das Board ausgebaut, die Buche neu aufgelötet und wieder eingebaut. 5 Tage später hat der Mikrochip nicht mehr zuverlässig gearbeitet und das Developmentboard wurde erneuert. Ein weiteres Mal ist der Chip ohne erkennbaren Grund durchgebrannt. Er hat nicht mehr reagiert und wurde ungewöhnlich heiß.

Bei den verwendeten Bibliotheken handelt es sich bei den meisten um Open Source Bibliotheken von Github.com. Das ganze Projekt baut auf dem esp-idf-arduino-bluepad32-template von ricardoquesada auf. Dies ist notwendig, damit Bluetooth Gamepads verwendet werden können. Das erste Problem, das in diesem Zusammenhang auftrat, war, dass der verwendete Xbox One Controller sich nur einmal alle 2-4 Wochen mit dem Mikroprozessor verbindet. Das Problem wurde durch einen Workaround gelöst, indem ein ELRS Empfänger verbaut wurde und das entsprechende Protokoll implementiert wurde. Im Verlauf der Arbeit wurden weiterhin die Bibliotheken aktualisiert, in der Hoffnung, dass es von ricardoquesada noch eine Fehlerbehebung gibt. Da der Workaround gut funktioniert, wurden keine weiteren Tests unternommen. Während der Arbeit kam hinzu, dass in nach einer Aktualisierung der arduino Bibliothek die Funktionen loop() und setup() nicht ausgeführt wurden. Die Lösung davon war, auf die vorherige arduino Version zurückzufahren und später das Template zu aktualisieren. Ein Problem, das auch viel Zeit in Anspruch genommen hat, war, dass nach der Aktualisierung des Templates der ESP32 nicht mehr starten wollte und sich durchgehend zurückgesetzt hat. Das lag daran, dass über die neue Version der Bibliotheken im Template die Projektkonfiguration geändert werden musste, da sich die USB Konsole anders verhalten hat. Die Lösung war, die Option Enable UART output von der Bluepad32 Konfiguration einzuschalten. Die Fehlersuche davon hat mehrere Tage in Anspruch genommen.

Das Problem mit den Standardbibliotheken ist in der Programmierung der USB Konsole sichtbar geworden. Die Ausgabe über die Funktion printf() hat im Durchschnitt eine Verlustrate einzelner Zeichen von 1,75 – 2,5%. Diese Verlustrate wurde über einen Zeichenzähler gemessen. Dabei werden die Zeichen, die das Programm ausgeben soll, gemessen und am Ende der Ausgabe wird das Ergebnis der Zählung mit exportiert. Wenn es Abweichungen von der Zeichenmenge in der Ausgabe und der angegebenen Zeichenanzahl gibt, liegt offensichtlich ein Fehler vor. Das ist beim Exportieren der Sensorwerte ein großes Problem, da nicht ersichtlich ist, wo ein Zeichen fehlt. Ein Lösungsansatz war, kritische Teile des Programms beim Export der Daten zu deaktivieren. Leider hat das nichts geändert und die Implementierung der printf() Funktion lässt keine Deaktivierung der Interrupts zu, um die Sendung der Zeichen zu garantieren. Ein Workaround ist, mit der Ausgabe der Zeilenlänge zu prüfen, ob eine Zeile plausibel ist und somit genommen werden kann. Bei dieser Art der Fehlererkennung ließen sich 42% Verlust messen. Das schmälert die Datenmenge leider, da nur alle 20ms ein Datensatz erstellt wird.

Der Dual Core Prozessor vom ESP32 hat auch Probleme mit Multitasking. Auf dem Chip laufen 5 Tasks parallel. Dabei ist anzumerken, dass das Timing von den beiden UARTs für die Hoverboardmainboards unzuverlässig wurde. Die Lösung für das Problem war es, die Funktion zum Senden der Pakete vor Unterbrechungen zu schützen und die Interrupts in dieser Zeit zu deaktivieren. Aktuell ist das Feedback von den Hoverboardmainboards auch nicht schlüssig, da der

ESP nicht liest, was mit dem Oszilloskop auf dem Bus messbar ist. Eine Ursache dafür kann sein, dass der Mikrocontroller eine verhältnismäßig schlechte FPU hat und in Messungen bei der Leistung dieser um den gleichen Faktor länger braucht für Berechnungen mit Kommazahlen wie andere Controller, die keine FPU haben und das per Software machen [8].

5. Fazit

Die Anhängerregelung schafft es, das System mit Regeleingriffen zu stabilisieren. Zu der Stabilisierung fährt der Häger in die grob vorgegebene Richtung. Dabei wird sichtbar, dass die Regelung noch unpräzise ist. Diese Problematik kommt von der Lenkregelung und könnte in späteren Arbeiten behoben werden. Zusätzlich empfiehlt sich eine Überarbeitung der Lenkmechanik. Wenn die Mechanik eine leichtgängigere Lagerung bekommt, kann die Regelung präziser mit dieser arbeiten. Dabei bietet sich eine Einarbeitung einer Wälzlagerung an.

Bei der Regelung kommen mehrere Probleme auf. Erstens die Motoren reagieren nichtlinear auf höhere Gaswerte. Damit beschleunigt das Fahrzeug beim Ändern des Lenkwinkels. Damit verhält sich das Fahrzeug nicht, wie ein normaler Nutzer dies erwartet. Da die Regelungen aber komplett getrennt sind für das Lenken und für das Fahren, lässt sich das beheben.

Damit ist das Steuerungskonzept entwickelt und auch getestet, bleibt aber in der Präzision hinter der Erwartung zurück. Für eine höhere Präzision ist Optimierung notwendig

6. Ausblick

In Zukunft kann das Projekt weiterentwickelt und die regelungstechnischen Funktionen für die Lenkung verbessert werden, z.B. über das Verlegen der Lenkregelung auf die Motorcontroller, damit lassen sich höhere Abfrageraten erreichen und somit eine präzisere Regelung entwickeln. Diese Art der Implementierung ermöglicht durch die niedrigere Reaktionszeit auch eine Änderung des Ansteuerungsmodus für die Motoren und ermöglicht damit ein intuitiveres Fahrverhalten. Eine weitere Möglichkeit besteht darin, die Rückgabewerte der Positionssensoren der Motoren in das Programm mit einzubinden, da sich damit Schluß und Fahrstrecke messen und errechnen lassen. Die Sensoren messen die Wegstrecke der 6.5 Zoll Motoren in 6mm Schritten. Dabei erkennen sie auch die Drehrichtung. Mit dem Übereinanderlegen der Daten mit einem Beschleunigungssensor lassen sich sehr genau Fahrsituationen analysieren und eine präzise Vorhersage über die gefahrene Strecke machen. Damit wäre es möglich, eine vordefinierte Strecke automatisiert abzufahren. Damit lassen sich auch die Parameter des Hängers messen, um ihn präziser zu simulieren. Über die Fahrstrecke, den Lenkwinkel und die Winkeländerung des Hängers kann berechnet werden, wie lange der Deichsel ist. Über diese gemessenen Parameter lassen sich dann genaue Modelle simulieren und der Häger kann präziser navigiert werden.

7. Quellen

- [1] <https://liu.diva-portal.org/smash/get/diva2:1066727/FULLTEXT01.pdf>
- [2] https://www.researchgate.net/publication/221912629_Reversing_Control_of_a_Car_with_a_Trailer_Using_the_Driver_Assistance_System
- [3] <https://github.com/lucysrausch/hoverboard-firmware-hack>

- [4] <https://github.com/EFeru/hoverboard-firmware-hack-FOC>
- [5] <https://github.com/BobbyIndustries/hoverboard-firmware-hack-FOC>
- [6] <https://github.com/LayerDE/bobbycar-project>
- [7] <https://github.com/LayerDE/simulator>
- [8] https://www.researchgate.net/publication/316173015_Analysis_of_the_performance_of_the_new_generation_of_32-bit_Microcontrollers_for_IoT_and_Big_Data_Application/link/58f54f1f458515ff23b570c3/download

Abbildungsverzeichnis

Abbildung 1: Orginales VW Beetle Bobbycar mit Hänger	6
Abbildung 2: Foto der verbauten Hinterachse	6
Abbildung 3: Rendering des Seitlichen Hinterachsteil	7
Abbildung 4: Beschriebener Hinterachschenkel mit nötdürftiger Reparatur.....	7
Abbildung 5: Oberseite der notdürftig reparierten Achse	7
Abbildung 6: Die rissigen Achsböcke.....	8
Abbildung 7: Die rissigen Achsböcke.....	8
Abbildung 8: Rendering des verklebten Adapters zur Verbreiterung.....	9
Abbildung 9: Einklebeteil der Vorderachsre (schwarz) mit Spacer (blau).....	9
Abbildung 10: Vorderachse Radaufnahmen, Spurstange und Achsschenkel	9
Abbildung 11: Rendering der zweiteiligen Radaufnahme.....	10
Abbildung 12: Rendering von der Oberseite der Radaufnahme.....	10
Abbildung 13: Rendering von der Unterseite der Radaufnahme.....	10
Abbildung 14: Rendering des Teils, das als Spurstangen verwendet wird.....	11
Abbildung 15: Rendering des Achsschenkels	11
Abbildung 16: Hoverboardregelboard	12
Abbildung 17: Die aktuell verbauten Regler zusammen mit dem Hoverboardakku (zwei mit Paketband isolierte Hoverboardregler).....	12
Abbildung 18: Umgebauter Panasonic E-Bike Akku	13
Abbildung 19: Sprungantwort auf eine Änderung des Eingangsignals des Gaspedals	16
Abbildung 20: Sprungantwort des Lenkwinkels	18
Abbildung 21: Sprungantwort des Regelwertes des P-Reglers	19
Abbildung 22: Rendering des Deichsels mit Sensor und Bolzen	20
Abbildung 23: Rendering der verklebten Führungsschiene	20
Abbildung 24: Rendering von Kunststoffpfanne und Bolzen	21
Abbildung 25: Rendering vom Deichsel ohne den Bolzen	21
Abbildung 26: Rendering vom Bolzen im Deichsel.....	21
Abbildung 27: Kreisbogennäherung Skizze, die Fahrstrecke des Bobbycars ist schwarz dargestellt, die des Anhängers blau.	22
Abbildung 28: Dreicksnäherung Skizze, die Fahrstrecke des Bobbycars ist schwarz dargestellt, die des Anhängers blau.....	23
Abbildung 29: Fall 1 Situation 1.....	24
Abbildung 30: Fall 1 Situation 2.....	24
Abbildung 31: Fall 2	25
Abbildung 32: Fall 3	25
Abbildung 33: Schaubild zum stabilen Winkel	26
Abbildung 34: Grafik zur Funktion des stabilen Winkels des Anhängers.....	27
Abbildung 35: Grafik zur linearen Näherung des stabilen Winkels.....	27
Abbildung 36: Diagramm von den Messwerten der ersten Fahrtaufzeichnung mit 0° angepeiltem Anhängerwinkel.....	31
Abbildung 37: Diagramm von den Messwerten der zweiten Fahrtaufzeichnung mit 4° angepeiltem Anhängerwinkel.....	31
Abbildung 38: Diagramm von den Messwerten der dritten Fahrtaufzeichnung mit 12° angepeiltem Anhängerwinkel.....	32

Abbildung 39: Diagramm von den Messwerten der vierten Fahrtaufzeichnung mit 10° angepeiltem Anhängerwinkel..... 32