|  |
| --- |
| Albert-Ludwigs-Universität Freiburg |
| Bachelor Projekt |
| **Lenkung eines Elektrofahrzeugs durch die Steuerung der Drehmomente der Räder** |
|  |
| **Victor Maier (4337015)** |
| **19.12.2022** |



|  |
| --- |
|  |

## Gliederung

* Aufbau des Fahrzeugs
  + Hardware
    - Achsen
      * Hinterachse
      * Vorderachse
      * Benötigtes Material
      * Fertigungsdauer
    - Regler
    - Sensoren
      * Lenkwinkelsensor
    - Hauptprozessor
  + Software
    - Regler
      * Probleme
        + Current init
        + USART Lag
        + USART Speed
        + Feedback Speed
    - Hauptprozessor
* Eingabekonzepte
  + API
  + ADC
  + Console
  + Gamepad
  + RC
* Regelungen
  + Eingabefilterung (ADC)
    - Sprungantwort
  + Allrad Regelung
  + Lenkregelung
    - PID
    - Sprungantwort Winkel
    - Sprungantwort Regelung
* Simulator
  + Modelierungen
    - Modelierung über Splices
    - Näherung über Kreisbögen
    - Näherung über Dreiecke
  + Abläufe in der Simulation
    - Position
      * move\_straight
    - Car
      * move
    - Follower
      * move…
* Anhänger
  + Errechnung des stabilen Winkels
  + Lineare Näherung des stabilen Winkels
  + Anwendung der Simulation
  + Anwendung einer Lookuptabelle
    - Rechenzeit zur Erstellung der Tabelle
    - Schritte zum Auslesen eines Wertes

## Aufbau des Fahrzeuges

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



Abbildung : 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.

## Hardware

### Achsen

#### Hinterachse

Die Hinterachse besteht aus zwei großen Kunststoffteilen, die mit 2 ca. 20cm langen 25x25x2mm Aluminiumprofilen verbunden sind. An diesen sind sie mit Kleber 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.

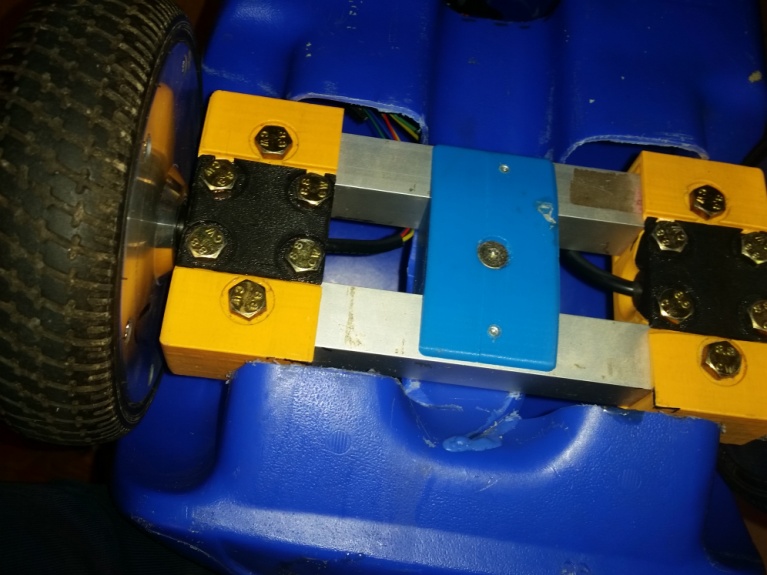


Abbildung : Erste Version der Hinterachse (aktuell verbaut in dem Bobbycar)

#### Vorderachse

Die Vorderachse besteht aus acht Kunststoffteilen. Bei der ersten Version sind die unteren Teile der Motoraufnahme gebrochen. Daraufhin wurden der Achsschenkel und die Radaufnahmen neu designt. Die Verbreiterung ist auch schon gebrochen, das ist aber kein Problem, da das Teil jetzt stabiler gefertigt ist und mit einer M3 Schraube verstärkt wurde. 

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



Abbildung : Vorderachse Radaufnahmen, Spurstange und Achsschenkel

#### Benötigtes Material

1,2-1,5Kg 3D Drucker Filament (PLA, ABS oder PETG)

#### Fertigungsdauer

Ca. 1 Woche Dauerbetrieb mit einem handelsüblichen 3D Drucker.

### Regler

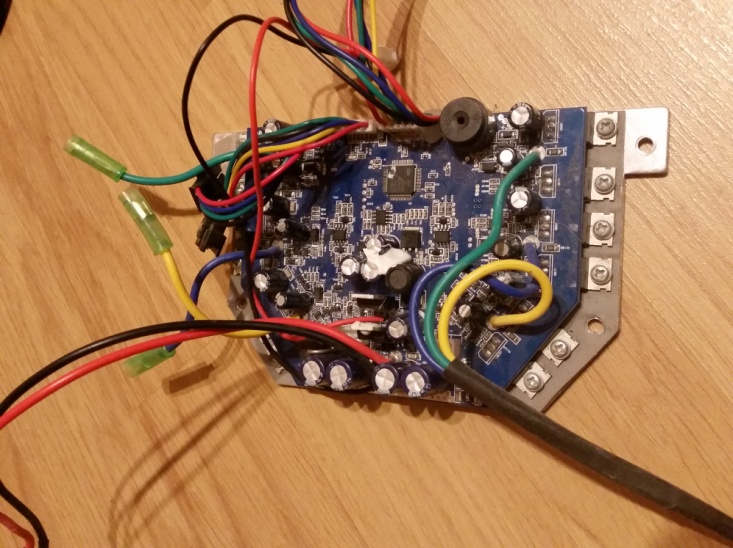
Bei den Motorreglern handelt sich es um Hoverboardregler mit STM32F103 Prozessoren aus der ersten Generation. Diese Boards können je 2 bürstenlose Motoren ansteuern. Sie sind für 36V ausgelegt und sind für Ströme bis zu 15A sicher. Sie lassen sich aber auch mit 54V betreiben und halten auch Strömen von 30A stand. Dabei steigt das Ausfallrisiko aber sehr. 

Abbildung : Hoverboardregelboard



Abbildung : Die aktuell verbaute Elektronik (36V Akku und zwei mit Packetband isolierte Hoverboardregler)

### 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.

### Hauptprozessor

Bei dem verbauten Hauptprozessor handelt sich es um einen ESP32 C3 Microcontroller mit einem XTensia Dualcore Prozessor, der auf 160MHz getaktet ist. Zusätzlich sind 4MB Flash an den Controller angebunden, um den Programmcode und sonstige relevante Werte zu speichern.

## Software

### Regler

Die Regler laufen mit einer modifizierten Version, dem hoverboard-firmware-hack-FOC von eferu auf Github. Dieser hat folgende Änderungen:

1. bldc.c current init
   1. Problem: Der alte Algorithmus nimmt effektiv nur die letzten 5 Werte des ADC bei 2000 Initialisierungsmessungen
   2. Lösung: Der neue Algorithmus errechnet den Durchschnitt von 1024 ADC-Messungen.
   3. Lösung Nr. 2: 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. USART Lag
   1. Problem: Die Eingabe wird nur sehr träge an die Motoren übergeben und damit ist die Treibersoftware sehr schwer regelbar.
   2. 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
   1. Problem: In der Rückmeldung vom Regelboard hat der Geschwindigkeitswert ein anderes Vorzeichen wie der PWM Wert
   2. Lösung: Die Vorzeichenkorrektur von den Eingabewerten auch auf die Speedwerte im Feedback anwenden.
4. USART Security
   1. Problem: Bei Timings-Problemen vom Hauptcontroller konnte es dazu kommen, dass das Fahrzeug die Signale falsch ausgewertet hat und Vollgas gegeben hat.
   2. 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.)

### Hauptprozessor

Der Steuerungscontroller hat eine vollständig eigenentwickelte Software, 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 kann und beheben kann, ohne dass Folgeschäden am Fahrzeug entstehen.

## Eingabekonzepte

### 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).

### ADC

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

### Console

Bei der Console handelt sich es um eine selbstprogrammierte Console, die über USB oder Bluetooth funktioniert. Diese hat Befehle zum Setzen von Gas- und Lenkwerten und dabei auch noch, um den PID Regler einzustellen. Debuggingwerte und kleine Logs können auch ausgelesen werden. Die Befehle vom Setzen der Werte sind aus der API, funktionieren absolut fehlerfrei und sind auch mehrfach getestet.

### 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, ist die Eingabemethode noch nicht wirklich nutzbar.

### RC

Hier wird ein ELRS Empfänger in das Bobbycar eingebaut. Der sendet per CRSF danach 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.

## Regelungen

### Eingabefilterung (ADC)

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

Initialisiert wird der Wert mit der Summe von Werten

Und

Und danach wird bei jedem neuen Wert:

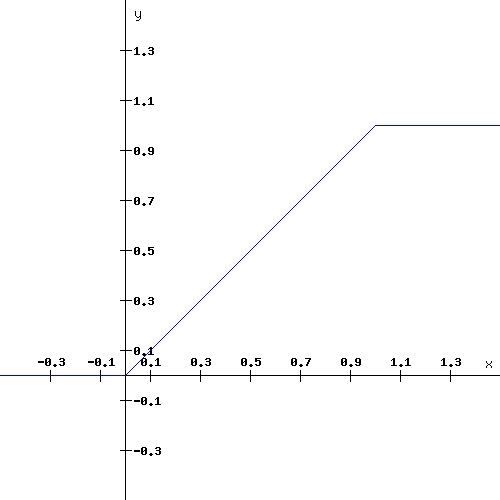


Abbildung : Sprungantwort auf eine Änderung des Eingangsignals des Gaspedals

### Allrad Regelung

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

Falls

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

### Lenkregelung

Hier wird ein PID-Regler (P-Regler) verwendet, bei dem , und ist. Falls der Ausgangswert ist, wird nochmal dazu addiert, um das Losbrechmoment der Motoren zu überwinden. Der Regelausgang wird auf limitiert. Die Regelroutine wird alle 20ms aufgerufen um etwas Ressourcen zu sparen aber trotzdem schnell genug zu reagieren, um starkes Überschwingen zu vermeiden. Dies ist notwendig, da sich der Motor nur um wenige Grad drehen muss, um einmal von dem rechten zum linken Endanschlagpunkt der Lenkung zu gelangen.

Bei dem Regelungsprozess entsteht folgende Sprungantwort:

Abbildung : Sprungantwort des Lenkwinkels

Abbildung : Sprungantwort des Regelwertes des P-Reglers

Simulation

Der Simulator ist simuliert ein Fahrzeug mit einer Lenkachse und einem Tandemanhänger. Sie simuliert die Bewegungen in einem 2D kartesischen Koordinatensystem. Diese Simulation muss nicht nur die Wege des Anhängers präzise genug aufzeigen sondern auch noch sehr performant laufen, damit sie auf einem ESP32 Microcontroller nebenher ausgeführt werden kann. Diese muss in Echtzeit Möglichkeiten überprüfen und damit mögliche Lenkwinkel verifizieren.

Modelierungen

Die Modellierung des Autos lässt sich darstellen indem über einen Kreisbögen die neue Position errechnet wird. Für den Anhänger gibt es mehrere Ansätze:

Modelierung 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 und lässt sich nicht in Echtzeit ausführen.

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.

Näherung über Dreiecke

Bei dieser Mathematisch einfachen Näherung liegt die Annahme zu Grunde, dass der Hänger sich ein kleines Stück als 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 Models gering ist. Es gilt auch zu beachten, dass wenn die Schritte zu klein werden Abweichungen über die Präzision der kommazahlen entstehen. Aufgrund der Technischen Einschränkungen wurde diese Art der Simulation gewählt.

In diesem Fall unterscheidet mein Programm 3 Situationen

Anhänger

Errechnung des stabilen Winkels

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

Lineare Näherung des stabilen Winkels

Errechung des Alphawinkels für eine definierte Änderung des Betawinkels nach einer definierten Stecke