Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

Tien Hung Nguyen

Entwicklung und Aufbau eines Fahrsimulators mit Hilfe einer kostengünstigen Mikrocontroller-Plattform

Fakultät Technik und Informatik
Department Mechatronik

Faculty of Engineering and Computer Science
Department of Mechatronics

Tien Hung Nguyen

Entwicklung und Aufbau eines Fahrsimulators mit Hilfe einer kostengünstigen Mikrocontroller-Plattform

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung
im Studiengang Mechatronik
an der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

in Zusammenarbeit mit:
Jugend- und Bildungswerk der Arbeiterwohlfahrt Hamburg GmbH
Abteilung Elektronik
Auf dem Königslande 45
22041 Hamburg

Betreuender Prüfer: Prof. Frerk Haase
Zweiter Prüfer: Prof. Dr.-Ing. Jochen Maaß

Abgegeben am 25. November 2016

Tien Hung Nguyen

Thema der Bachelorarbeit

Entwicklung und Aufbau eines Fahrsimulators mit Hilfe einer kostengünstigen Mikrocontroller-Plattform

Stichworte

Fahrsimulator, Getriebemotor, Arduino UNO, Sabertooth, Monster Motor Shield, Parameteridentifikation des Gleichstrommotors, Positionsregelung

Kurzzusammenfassung

Das Ziel der vorliegenden Bachelorarbeit ist es, einen Fahrsimulator nach Kundenanforderungen zu entwickeln und aufzubauen. In dieser Arbeit wird ein Getriebemotor mit unbekannten Parametern identifiziert und ein PID-Regler für die Positionsregelung entworfen. Der Mikrocontroller empfängt die Game-Daten durch eine Motion Simulator Software, um zwei Motoren anzusteuern, die die Bewegungen des Fahrzeugs im Spiel auf den Fahrsimulator übertragen.

Tien Hung Nguyen

Title of the paper

Development and building of a driving simulator with the help of a cost-effective microcontroller-platform

Keywords

Driving simulator, gear motor, Arduino UNO, Sabertooth, Monster Motor shield, parameter identification of a gear motor, position control

Abstract

The objective of the present bachelor thesis is to develop and build a driving simulator according to customer requirements. In this word, a gear motor with unknown parameters is identified and a PID controller for the position control is designed. The microcontroller receives the game data through a Motion Simulator software to drive two motors, which transfer the vehicle's movements to the driving simulator.

Danksagung

An dieser Stelle möchte ich all jenen danken, die mich während der Anfertigung dieser Bachelorarbeit unterstützt und motiviert haben.

Ganz besonders gilt dieser Dank Prof. Frerk Haase, der meine Arbeit durch seine fachliche und persönliche Unterstützung begleitet hat. Daneben möchte ich mich bei Prof. Dr.-Ing. Jochen Maaß für die großartige Betreuung bedanken.

Danken muss ich außerdem Frau Britta Hinz (Geschäftsführerin) und Herrn Jürgen Witt (Projektleiter) von der Firma Jugend- und Bildungswerk der Arbeiterwohlfahrt Hamburg GmbH, in der ich mein Praktikum absolviert habe. Sie haben mir die Möglichkeit gegeben, bei Ihnen zu arbeiten. Vielen Dank für die Zeit und Mühen, die Sie für mich investiert haben.

Ebenso möchte ich mich bei meinen Eltern bedanken, die durch Ihre Unterstützung mein Studium ermöglicht haben.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Motivation

Der Betrieb, das Jugend- und Bildungswerk der AWO Hamburg GmbH (JBW), in dem der in dieser Arbeit entwickelte Fahrsimulator entsteht, ist ein gemeinnütziger Träger der Jugendberufshilfe. Das JBW bildet außerbetrieblich schulisch und sozial benachteiligte Jugendliche in Berufen der metallverarbeitenden Industrie sowie der Fachinformatik aus. Als Zuwendungsempfänger der öffentlichen Hand verfügt das JBW nur über ein schmales Budget und ist daher auf Spenden Hamburger Betriebe angewiesen. Die Berufsausbildung im Metall- und Informatikbereich beinhaltet den Bau von Projektarbeiten, auch Gewerke übergreifend. Um Jugendliche zur Teamarbeit zwischen den beiden Gewerken zu motivieren, wurden sog. Playseats gebaut. Diese optisch einem Auto nachempfundenen Sitze werden im Rennspiele-Bereich eingesetzt. Dafür notwendige Eingabegeräte wie z.B. Pedalerien, Buttonbox, Schaltung und Handbremse wurden von den angehenden Fachkräften der Metall Technik aus Recyclingmaterialien erstellt, die Anbindung an den PC und die Ansteuerung eines Cockpits mit Drehzahl- und Geschwindigkeitsanzeige haben die Fachinformatik –Auszubildenden übernommen.

Im nächsten Schritt soll ein beweglicher Playseat, ein Low-Budget-Fahrsimulator entstehen, der die Bewegungen eines Fahrzeugs im Spiel in reale Stuhlbewegungen übersetzt. Dies ist die Motivation zu dieser Bachelorarbeit.

1.2 Ziele der Arbeit

Die Arbeit beschäftig sich mit der Optimierung eines vorhandenen Fahrsimulators, der im Praktikum aufgebaut wurde. Der vorherige Fahrsimulator hat wegen fehlender mechanischer Berechnung nur mit Einschränkung funktioniert, bzw. das limitierte Gewicht des Spielers ist max. 50 kg, die Spielzeit dauert max. 15 Minuten und die Bewegung des Stuhls war verzögert. In dieser Arbeit werden die Anforderungen von dem Kunden (Firma JBW) gesammelt und durch die mechanische Berechnung werden alle Kräfte und das Drehmoment für den Antrieb berechnet. Danach kann man den Antrieb mit der entsprechenden Leistung auswählen.

Außerdem werden die Parameter von dem ausgewählten Antrieb mittels der stochastischen Methode der “kleinsten Quadrate“ in Matlab/Simulink identifiziert. Ein PID-Regler wird nach der Ziegler/Nichols-Methode entworfen, um mit diesem PID-Regler die Position des Motors mit hoher Genauigkeit anzusteuern. Mit der Motion Simulation Software wird eine Verbindung zwischen dem Mikrocontroller und den PC-Spielen mittels der seriellen Schnittstelle aufgebaut. Danach kann der Mikrocontroller alle Game-Daten empfangen, um den Antrieb anzusteuern

Zuerst wird die Funktion des Fahrsimulators durch ein Blockdiagramm beschreiben.

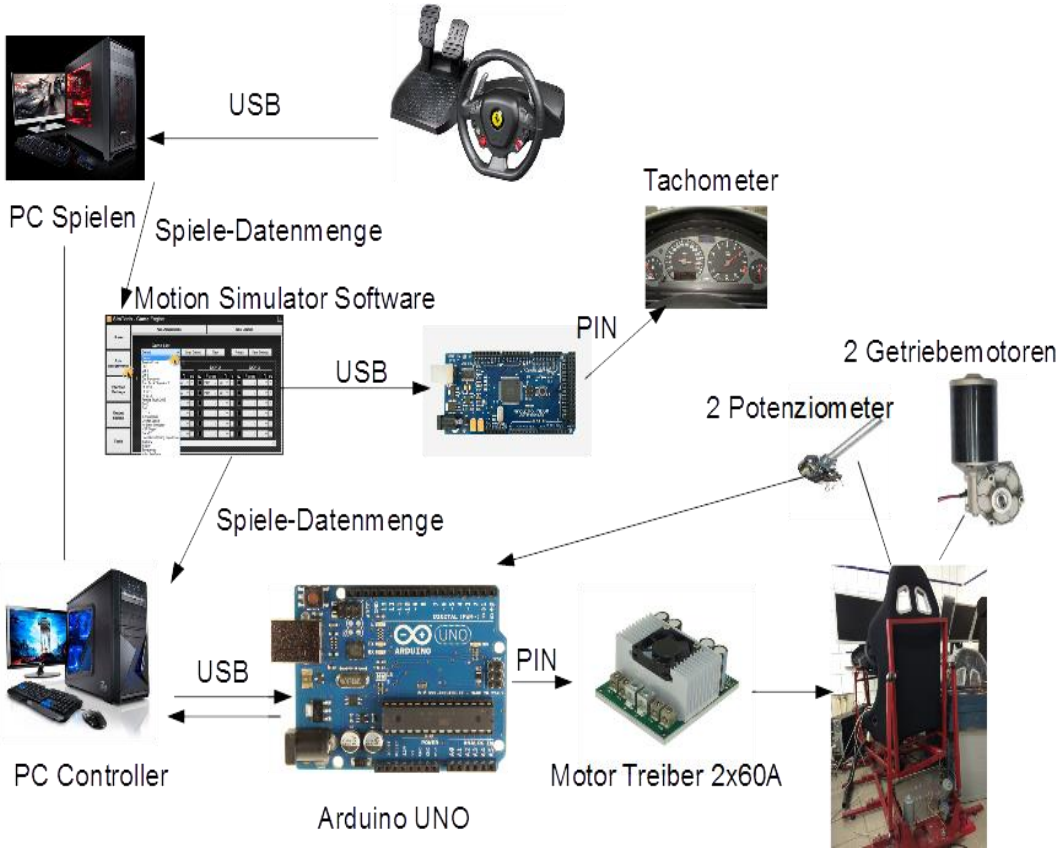


Abbildung 1: Blockdiagramm eines Fahrsimulators mit den Bauteilen

1.3 Gliederung der Arbeit

Die Arbeit ist in acht Kapitel aufgeteilt. In dem ersten Kapitel “Einleitung“ wird das Thema vorgestellt. Es beinhaltet die Motivation, die Ziele der Arbeit, die Beschreibung der Gliederung der Arbeit und die Anforderungen des Kunden. Das nächste Kapitel beschäftigt sich mit der mechanischen Berechnung des Fahrsimulators, die für die Auswahl des Getriebemotors benötigt wird. Im dritten Kapitel geht es um die Grundlagen eines Gleichstrommotor und das Erstellen des mathematischen Modells der GM mit Hilfe der Simulink Software Matlab. Das vierte Kapitel beinhaltet das Konzept zur Parameteridentifizierung der GM. Die bestimmten Parameter für die Arbeit werden zum Entwurf eines PID-Reglers im fünften Kapitel benötigt. Im Anschluss folgt das Kapitel “Fahrsimulator“. Dieses beinhaltet die Beschreibung des Fahrsimulators mit dem Sequenzdiagramm und die Erklärung der Komponenten des Systems. Das siebte Kapitel beschäftig sich mit den Sicherheitsmaßnahmen. Diese sind unterteilt in Software und Hardware. Das letzte Kapitel befasst sich mit einer Zusammenfassung dieser Arbeit. Außerdem wird ein Ausblick geschrieben, dieser gibt Ansätze zur Weiterentwickelung.

1.4 Anforderungen an den Fahrsimulator

Durch die Besprechung mit Frau Britta Hinz (Geschäftsführerin) und Herrn Jürgen Witt (Projektleiter) wird die Tabelle mit den Anforderungen des Systems dargestellt.

Tabelle 1: Anforderungsliste des Fahrsimulators mit E: Essentiell; N: Notwendig und W: Wünschenswert

ID	Titel	Priorität	Stakeholder	Anforderungen
FST-01	Positionsregelung	E	Nguyen	Zwei Motoren werden durch den Mikrocontroller angesteuert.
FST-02	mit PID Regler	N	Nguyen	Der Regelkreis soll mit dem bestimmten PID-Regler arbeiten
FST-03	Cockpit anzeigen	W	Nguyen	Die Gamedaten (Drehzahl, Geschwindigkeit...) sollen am Cockpit angezeigt werden
FST-04	Belastbarkeit	E	Nguyen	Das System soll mit einem Spieler mit einem Höchstgewicht von 80 kg arbeiten
FST-05	Spielzeit und Stabilität	N	Nguyen+Witt	Der Fahrsimulator muss mindestens 30 Minuten spielen
FST-06	Echtzeit	W	Nguyen	Das System soll mit der maximalen Verzögerung von 1 Sekunde funktionieren
FST-07	Modularität	W	Nguyen+Witt	Das System soll modular aufgebaut sein
FST-08	Gehäuse	N	Nguyen+Witt	Das Steuerungsmodul muss in einem Gehäuse montiert werden

ID	Titel	Priorität	Stakeholder	Anforderungen
FST-09	mehrere Rennspiele	N	Nguyen	Der Fahrsimulator soll mit mindestens zwei Rennspielen (rFactor und Dirt 2) arbeiten
FST-10	Gesamtkosten	W	Nguyen+Witt	Das System soll preisgünstig produziert werden können (700€)
FST-11	Austauschbarkeit	N	Nguyen+Witt	Das System soll wartungsfreundlich sein
FST-12	Sicherheit	E	Nguyen	Bei Überspannung oder Überhitzung soll das System Alarm geben
FST-13	Bedienbarkeit	N	Nguyen	Der Fahrsimulator soll einfach zu bedienen sein.

2 Auslegung des Elektroantriebs

Der Elektroantrieb ist das Herz des gesamten Systems. Im Fahrsimulator wird ein Gleichstrom-Getriebemotor als Drehmoment-Quelle verwendet. Der Getriebemotor muss genau das erforderliche Drehmoment sowie die erforderliche Drehzahl erzeugen, um eine realistische Bewegung und Simulation zu erreichen. Ein leistungsstarker Getriebemotor ist der größte Kostenpunkt im gesamten Simulator-Projekt. Außerdem ist die Auswahl der „mechanischen und elektrischen Elemente“ abhängig vom Motor bzw. Getriebe, wie beispielsweise die Fundamente, das Steuergerät usw. Deshalb spielt die Dimensionierung des Motors eine wichtige Rolle. Die Antriebsdimensionierung lässt sich in 2 Phasen: Analyse und Berechnung unterteilen.

2.1 Analyse

In dieser Phase werden die Antriebsanforderungen durch die Besprechung mit Frau Hinz (Geschäftsführung) und Herrn Witt (Ausbilder Metalltechnik) der Firma JBW erfasst. In der folgenden Tabelle wird der Vergleich zwischen einem Getriebemotor 12V und einer Asynchronmaschine gezeigt und in Bezug auf die Anforderungen bewertet:

Tabelle 2: Motoranforderungen

ID	Anforderung	Getriebemotor 12V	Asynchron-maschine
M01	niedrige Abtriebs Drehzahl ohne extra Getriebe	A	C
M02	Hohes Drehmoment am Abtrieb	B	A
M03	kleiner und kompakter Bauraum und geringes Motorgewicht	A	D
M04	Einfache Montage	A	C
M05	Mit niedriger Versorgungsspannung zu betreiben (wegen Sicherheit)	A	C
M06	Einfache Positionsregelung	A	C
M07	Steuerung durch Arduino UNO	A	B
M08	Niedrige Kosten	B	C

A: Sehr gut, **B**: Gut; **C**: Erreichbar, **D**: Nicht erreichbar

In der oberen Tabelle hat der Getriebemotor 6 Bewertungen mit A und die Asynchronmaschine hat nur eine Bewertung mit A. Deshalb wird der Getriebemotor 12V in diesem Fahrsimulator verwendet.

Die Bestimmung der Kräfte und der Lastmomente wird im nächsten Abschnitt „statische Berechnung“ behandelt.

2.2 Berechnung

2.2.1 Statische Berechnung

Die Basis der Antriebsdimensionierung ist die statische Berechnung. Es gilt die Bedingung, dass das Lastdrehmoment über den ganzen Drehzahlbereich nicht größer als das Antriebsmoment sein darf. Deswegen muss man zuerst das Lastdrehmoment des Systems bestimmen.

In der folgenden Abbildung wird die technische Zeichnung mit den äußeren und inneren Kräften des Fahrsimulators im kartesischen Koordinatensystem angezeigt.

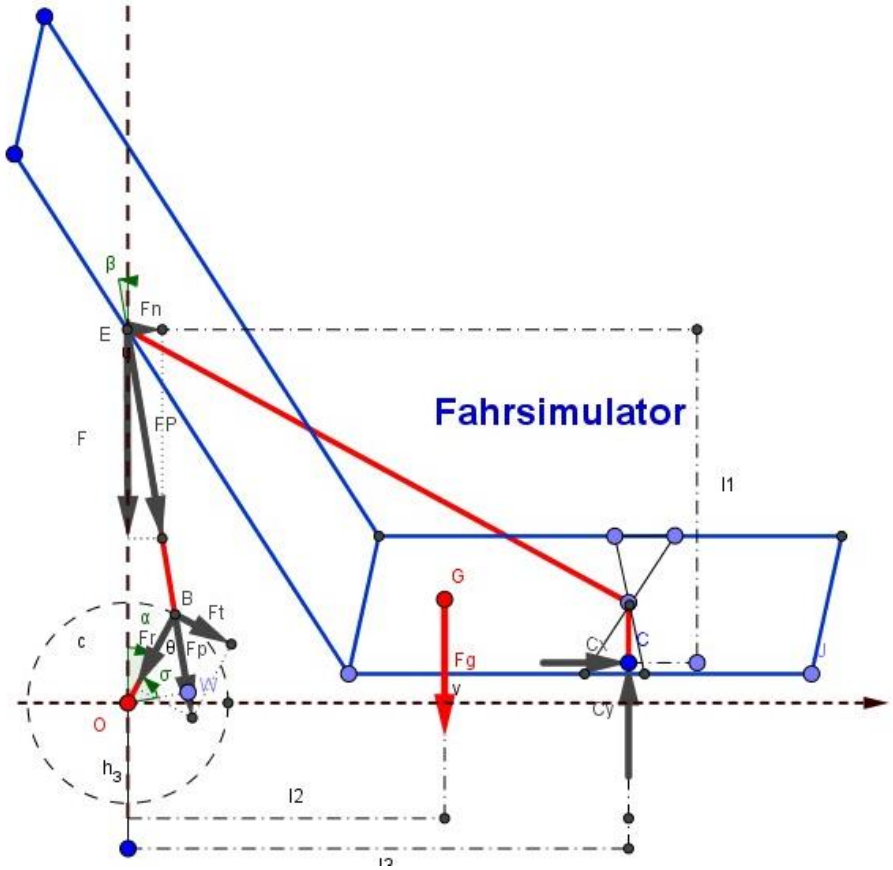


Abbildung 2: Einzeichnen der angreifenden Kräfte von Fahrsimulator

Die äußeren Kräfte:

$$\overrightarrow{F_N} : Normalkraft$$
$$\overrightarrow{F_R} : Radialkraft$$
$$\overrightarrow{F_T} : Tangentialkraft$$
$$\overrightarrow{F_P} : Pleuelstangenkraft$$
$$\overrightarrow{F_G} : Gewichtskraft$$

Die inneren Kräfte bei dem Festlager C

$$\overrightarrow{C_x} : horizontale\ Lagerkraft$$
$$\overrightarrow{C_y} : vertikale\ Lagerkraft$$

Die Koordinaten des bestimmten Schwerpunkts G im Koordinatensystem Oxyz

$$G\ (x_G, y_G, z_G) = G(0{,}40; 0{,}40; 0{,}25)$$

Die Vorberechnung des Winkels θ

Im rechteckigen Dreieck OWB:

$$\theta = \frac{\pi}{2} - \sigma \qquad \text{(Gl. 2.1)}$$

Im rechteckigen Dreieck OWE:

$$\sigma + \alpha = \frac{\pi}{2} - \beta \rightarrow \sigma = \frac{\pi}{2} - (\alpha + \beta) \qquad \text{(Gl. 2.2)}$$

Ersetzen (Gl. 2.2) in (Gl. 2.1)

$$\theta = \alpha + \beta \qquad \text{(Gl. 2.3)}$$

Um die Kraftberechnung zu vereinfachen, kann man das System in zwei Teile (Kurbelgetriebe und Sitzplatz) aufteilen:

- **Das Kurbelgetriebe**

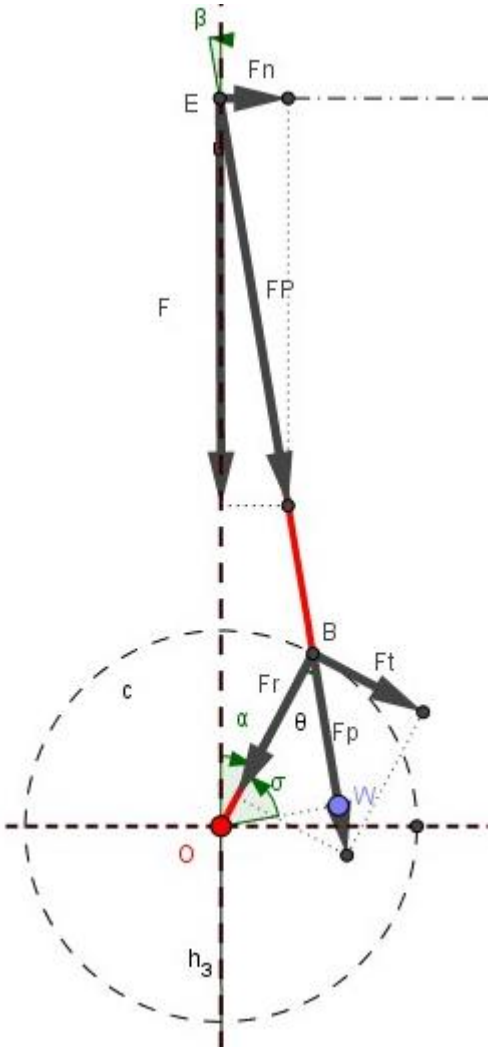


Abbildung 3: Die angreifenden Kräfte des Kurbelgetriebes

Nun wird der Winkel β betrachtet:

$$cos\beta = \frac{F}{F_P} \rightarrow F_P = \frac{F}{cos\beta} \quad \text{(Gl. 2.4)}$$

$$tan\beta = \frac{F_N}{F} \rightarrow F_N = F * tan\beta \quad \text{(Gl. 2.5)}$$

Die Beziehung zwischen der Tangentialkraft F_T und den Winkeln α und β

$$sin\theta = \sin(\alpha + \beta) = \frac{F_T}{F_P} \rightarrow F_T = F_P * \sin(\alpha + \beta) \quad \text{(Gl. 2.6)}$$

- **Der Sitzplatz**

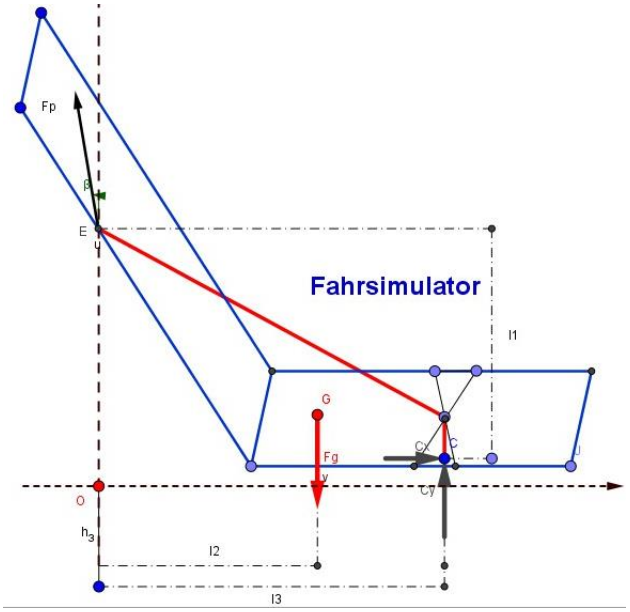


Abbildung 4: Die angreifenden Kräfte des Sitzplatzes

Kräftegleichgewicht in x-Richtung

$$\sum F_x = 0 = -F_P * sin\beta + Ax \rightarrow Ax = F_P * sin\beta \qquad \text{(Gl. 2.7)}$$

Kräftegleichgewicht in y-Richtung

$$\sum F_y = 0 = F_P * cos\beta - F_G + Ay \rightarrow Ay = F_G - F_P * cos\beta \qquad \text{(Gl. 2.8)}$$

Momenten-Gleichgewicht an dem Punkt E

$$\sum M_E = F_G * l2 - Ay * l3 - Ax * l1 = 0 \qquad \text{(Gl. 2.9)}$$

Die Gleichungen 2.7 und 2.8 werden in der Gleichung 2.9 ersetzt und danach kann man die Pleuelstangenkraft bestimmen:

$$F_P = F_G * \frac{l3-l2}{l3*cos\beta - l1*sin\beta} \qquad \text{(Gl. 2.10)}$$

Die Lagerkräfte Cx und Cy

$$C_x = F_G * sin\beta * \frac{l3-l2}{l3*cos\beta - l1*sin\beta} \qquad \text{(Gl. 2.11)}$$

$$C_y = F_G - F_G * cos\beta * \frac{l3-l2}{l3*cos\beta - l1*sin\beta} \qquad \text{(Gl. 2.12)}$$

Die Tangentialkraft wird durch das Ersetzen der Gleichung 2.10 in die Gleichung 2.6 berechnet:

$$F_T = m * g * \frac{l3-l2}{l3*cos\beta - l1*sin\beta} * \sin(\alpha + \beta) \qquad \text{(Gl. 2.13)}$$

Jetzt kann man das Lastdrehmoment des Fahrsimulators bestimmen:

$$M = m * g * r = m * g * \frac{r*(l3-l2)}{l3*cos\beta - l1*sin\beta} * \sin(\alpha + \beta) \qquad \text{(Gl. 2.14)}$$

Mit r: Länge des Hebelarms

Das Ergebnis mit den gemessenen Bemaßungen:

l1 [m] = 0,70

l2 [m] = 0,40

l3 [m] = 0,60

r [m] = 0,08

α [rad] = $\frac{5\pi}{18}$

β [rad] = $\frac{\pi}{18}$

Das Gewicht des Systems ist die Summe der Gewichte von Spieler und Stuhl

$m = m_{Spieler} + m_{Stuhl} = 140\ kg$

Mit $m_{Spieler} = 100\ [kg]\ und\ m_{Stuhl} = 40\ [kg]$

Die Stützkräfte und die Kräfte der Kurbel

Cx [N] = 100,88

Cy [N] = 801,27

$F_P\ [N] = 580{,}96$

$F_T\ [N] = 503{,}13$

Das errechnete Drehmoment

M [Nm] = 40

Im dem Fahrsimulator verwendet man zwei Motoren, d.h. das minimale Lastdrehmoment für jeden Motor wird halbiert.

$M_{min} = \frac{M}{2} = 20\ [Nm]$

2.2.2 Dynamische Berechnung

In dieser dynamischen Berechnung geht es um die grafische Ermittlung von den Geschwindigkeiten des Fahrsimulators. Durch die mathematische Software GeoGebra[1] (Open Source) wird die Konstruktion des Geschwindigkeitsplans von dem Fahrsimulator erleichtert. In dem Geschwindigkeitsplan nutzt man die Euler-Gleichung, um den Betrag und die Wirkungslinien aller Geschwindigkeitsvektoren zu bestimmen. Jetzt wird die Grundlage der Euler-Gleichung erklärt.

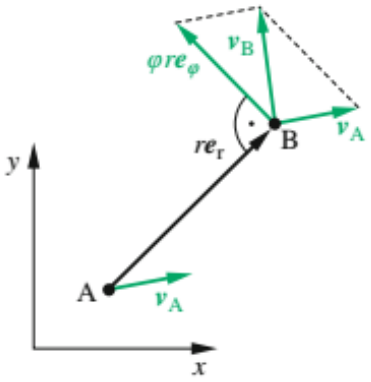


Abbildung 5: Geschwindigkeitsplan bei einer allgemein ebenen Bewegung[2]

Mit Hilfe der Euler-Gleichung wird die Geschwindigkeit im Punkt B bestimmt und der vorliegende Geschwindigkeitsvektor $\overrightarrow{V_B}$ resultiert folglich aus der vektoriellen Addition der Geschwindigkeiten der Teilbewegungen.

$\overrightarrow{V_B} = \overrightarrow{V_A} + re_\varphi \; oder \; \overrightarrow{V_B} = \overrightarrow{V_A} + \overrightarrow{V_{BA}}$

In folgender Abbildung wird der Geschwindigkeitsplan des Fahrsimulators gezeigt.

[1] Homepage: https://www.geogebra.org
[2] vgl. Werber Skolaut: Maschinenbau - Ein Lehrbuch für das ganze Bachelor-Studium, 2014, S.208

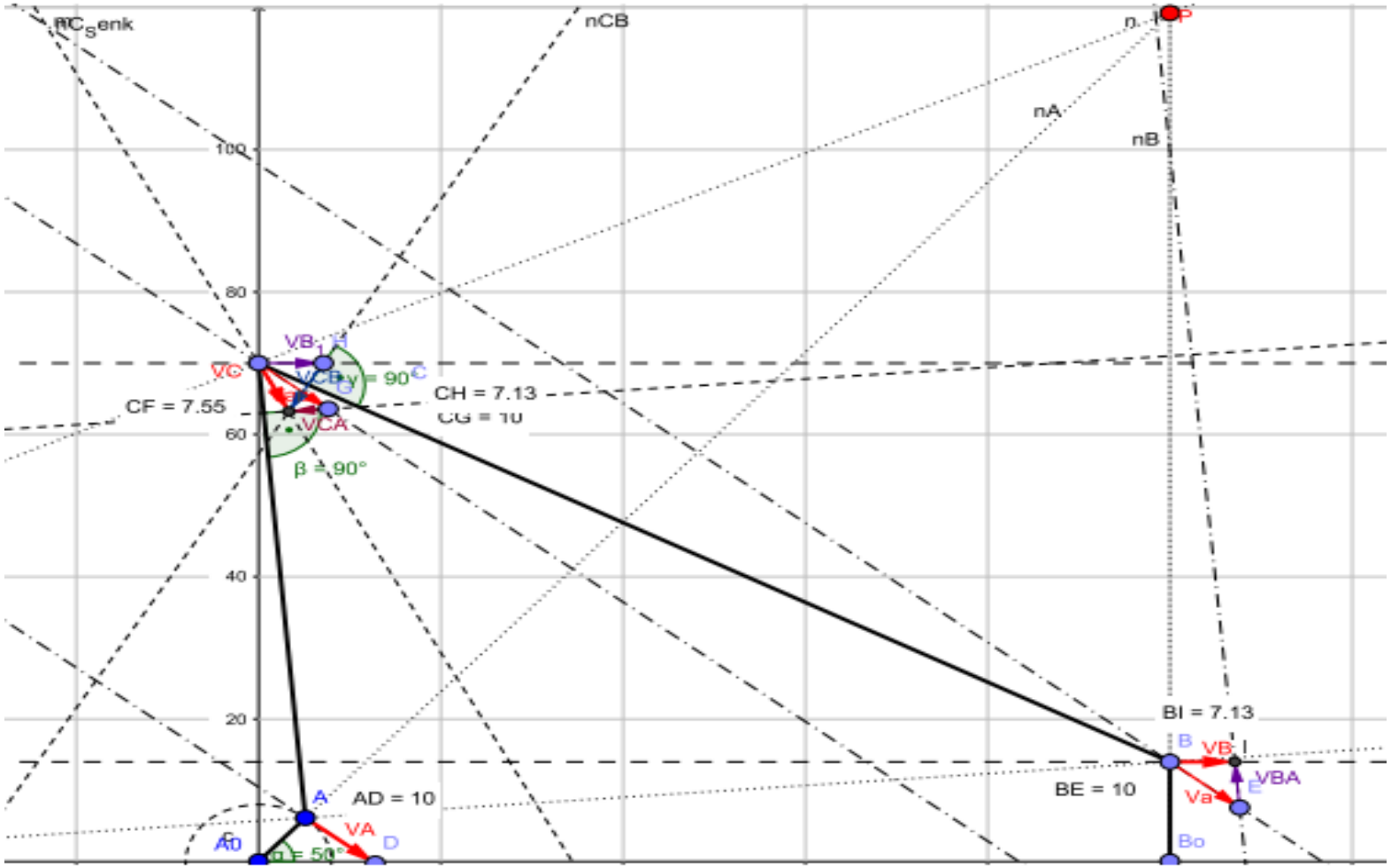


Abbildung 6: Geschwindigkeitsermittlung des Fahrsimulators im Maßstab 1:1

- **Geschwindigkeit im Punkt A**

Mit der vorgegebenen Antriebsdrehzahl n=95rpm der Kurbel kann man zunächst die Winkelgeschwindigkeit im Punkt A0 berechnen:

$w = 2\pi n = 2\pi * \frac{2\pi}{60} * 95rpm = 62{,}5\frac{rad}{s}$

Der Betrag der Geschwindigkeit im Punkt A ist zu ermitteln:

$V_A = AA0 * w = r * w = 0{,}08m * \frac{62{,}5rad}{s} = 5m/s$

Der Geschwindigkeitsvektor $\overrightarrow{V_A}$ steht senkrecht der Bahnnormalen nA und der Geschwindigkeitsplan wird mit dem Maßstab 1:1 konstruiert. Die Länge des dargestellten Vektors $\overrightarrow{V_A}$ mit dem Betrag 5 m/s ist 10 Zentimeter

- **Geschwindigkeit im Punkt B**

Mit Hilfe der Euler-Gleichung wird die Geschwindigkeit im Punkt B ermittelt:

$\overrightarrow{V_B} = \overrightarrow{V_A} + \overrightarrow{V_{BA}}$

Terme $\overrightarrow{V_A}$ ist vollständig gegeben (Betrag und Richtung) und bei den anderen Termen $\overrightarrow{V_B} und \overrightarrow{V_{BA}}$ sind nur die Richtungen bekannt. ($\overrightarrow{V_B}$ senkrecht der Bahnnormalen nB und $\overrightarrow{V_{BA}}$ senkrecht der Richtung $\overrightarrow{AB}$). Von der Spitze des Vektors $\overrightarrow{V_A}$ zeichnet man eine Linie senkrecht der Strecke AB. Durch den Schnittpunkt zwischen den Wirkungslinien $\overrightarrow{V_B} und \overrightarrow{V_{BA}}$ ist der Geschwindigkeitsvektor $\overrightarrow{V_B}$ mit dem bekannten Betrag zu ermitteln. Die Länge des Geschwindigkeitsvektors $\overrightarrow{V_B}$ beträgt 7,13 Zentimeter. Danach wird der Betrag von $\overrightarrow{V_B}$ durch den Maßstab von $\overrightarrow{V_A}$ (5 m/s ≔ 10 $Zentimeter$) berechnet:

$V_B = \frac{5\ m/s}{10\ cm} * 7{,}13cm = 3{,}6\ m/s$

- **Geschwindigkeit im Punkt C**

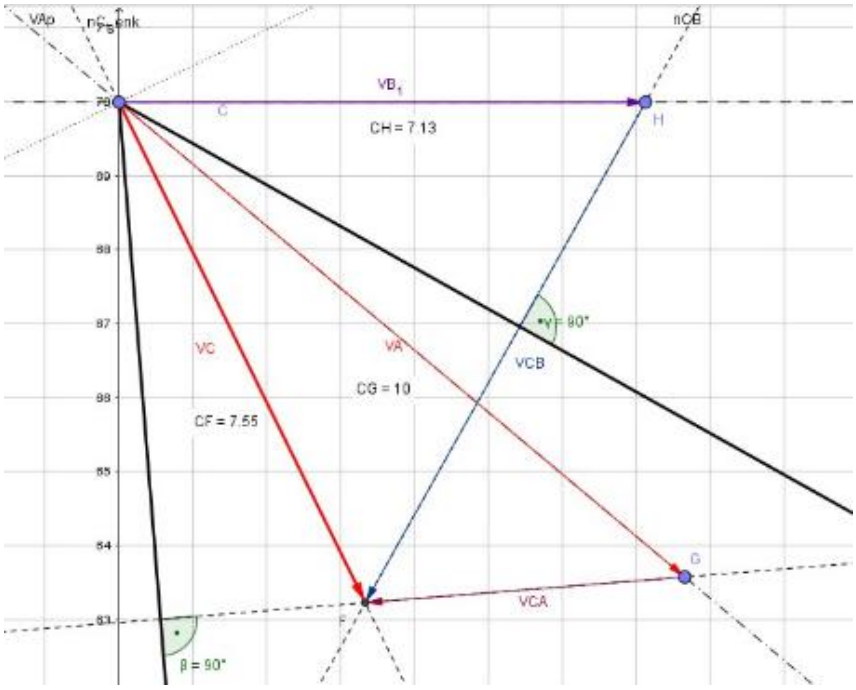


Abbildung 7: Geschwindigkeitsvektor im Punkt C

Die Geschwindigkeit im Punkt C ist mit Hilfe der Euler-Gleichung zu ermitteln:

$\overrightarrow{V_C} = \overrightarrow{V_A} + \overrightarrow{V_{CA}} \quad bzw. \quad \overrightarrow{V_C} = \overrightarrow{V_B} + \overrightarrow{V_{CB}}$

Durch den Schnittpunkt von zwei Bahnnormalen nA und nB kann man den Momentanpol P, der den Schnittpunkt der Bahnnormalen aller Punkte in einer Lage der bewegten Ebene ist, bestimmen. Die Verbindungslinie zwischen dem Punkt C und dem Momentanpol P entspricht der Bahnnormalen nC, auf der der Geschwindigkeitsvektor $\overrightarrow{V_C}$ senkrecht steht. An dem Punkt C wird der Geschwindigkeitsvektor $\overrightarrow{V_A}$ mit der Länge 10 Zentimeter dargestellt, von der Spitze dieses Vektors $\overrightarrow{V_A}$ wird der Vektor $\overrightarrow{V_{CA}}$ (senkrecht der Strecke AC) gezeichnet. Durch den Schnittpunkt zwischen den Wirkungslinien $\overrightarrow{V_C}$ und $\overrightarrow{V_{CA}}$ ist der bestimmte Geschwindigkeitsvektor $\overrightarrow{V_C}$ zu ermitteln. Die Länge des Vektors $\overrightarrow{V_C}$ beträgt 7,55 Zentimeter. Der Betrag des Vektors $\overrightarrow{V_C}$ wird berechnet:

$$V_C = \frac{5\frac{m}{s}}{10\ cm} * 7{,}55\ cm = 3{,}8\frac{m}{s}$$

Durch die dynamische Berechnung bestimmt man alle Geschwindigkeitsvektoren $\overrightarrow{V_A}$, $\overrightarrow{V_B}$ und $\overrightarrow{V_C}$ mit den bekannten Richtungen und Beträgen.

2.3 Auswahl des Motors

Aus den Ergebnissen der statischen Berechnung und dem Preisvergleich zwischen den Herstellern wurde der Getriebemotor WG88BD88-1 mit dem maximalen Drehmoment 60 Nm der Firma IMC-Drives ausgewählt.

Abbildung 8: Getriebemotor 12VDC

In folgender Tabelle befindet sich die technischen Daten des ausgewählten Motors:

Voltage (V)	Power (W)	No Load		Load		
		Speed (RPM)	Current (A)	Speed (RPM)	Current (A)	Torque (N.M)
36	100	20±2	≤2.5	16±2	≤5.5	≥5.5
28	200	200±5	≤2.5	185±5	≤6.0	≥5.0
24	150	180±5	≤2.0	185±5	≤6.0	≥5.0
12	120	100±5	≤3.2	95±5	≤8.0	≥4.5

Abbildung 9: Motordaten von WG88BD88-1

3 Grundlagen der GM

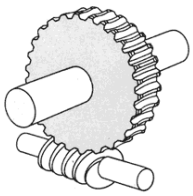
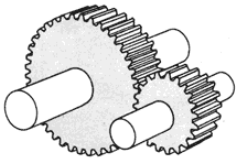
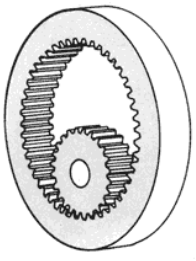
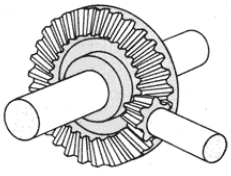
Heutzutage wird der Gleichstrommotor sehr häufig als Antriebsmotor eingesetzt. In dieser Arbeit wird der Gleichstrom Getriebemotor zur Erzeugung der Bewegung des Fahrsimulators verwendet. Der Gleichstrom Getriebemotor ist eine Kombination von einem Gleichstrommotor und einem Getriebe. In den meisten Fällen läuft der Getriebemotor mit geringer Geschwindigkeit und höherem Drehmoment. In dem Fahrsimulator-System sind nur das hohe Drehmoment und die niedrige Geschwindigkeit von Interesse. Aus obigem Grund spielt die Verwendung von einem Getriebemotor eine wichtige Rolle. Außerdem hat der Getriebemotor viele Vorteile:

- Die Richtungsumkehr ist problemfrei durch einfaches Umpolen im Betrieb möglich
- Die Ansteuerung ist sehr einfach und hat nur einen geringen regelungstechnischen Aufwand
- Er ist preiswert
- Die 12 VDC Gleichstrom Spannungsversorgung schützt den Spieler vor einer gefährlichen Hochspannung, ein Vorteil im Vergleich zum Wechselstrommotor

Aber der Gleichstrommotor hat auch einen großen Nachteil: Er erreicht nicht so hohe Wirkungsgrade, da in Folge der sehr großen Spulenpakete und der Verluste durch den Schleifkontakt der Bürsten ein erheblicher Teil der zugeführten elektrischen Energie in Wärme umgesetzt wird. Derzeit wird ein Antriebssystem ohne Kohlebürsten eingesetzt.

Je nach Einsatzbereich lässt sich der Getriebemotor in verschiedene Arten unterteilen. In folgender Tabelle wird der Getriebemotor mit den unterschiedlichen Getrieben aufgezeigt:

Tabelle 3: Getriebeart

Getriebe	**Schnecken**	**Stirngrad**	**Planeten**	**Kegelrad**
Form				
Zum Einsatz	Achse der Abtriebswelle senkrecht zur Motor-Drehachse	Achse der Abtriebswelle parallel zur Motor-Drehachse	Achse der Abtriebswelle gleich der Motor-Drehachse	Achse der Abtriebswelle senkrecht zur Motor-Drehachse

In dem Fahrsimulator wird der Getriebemotor mit dem Schneckengetriebe verwendet

Abbildung 10: Getriebemotor 12VDC WG88DB88-1

Im folgenden Abschnitt geht es um die theoretischen Grundlagen des allgemeinen Gleichstrommotors. Anschließend folgt eine mathematische Berechnung des Gleichstrommotor-Modells. Für dieses mathematische Modell wird die Simulation des Gleichstrommotors mit der Software Matlab/Simulink durchgeführt. Diese Simulation bereitet das nächste Kapitel „Parameteridentifikation für das Modell eines Gleichstrommotors “ vor.

3.1 Prinzipieller Aufbau

Ein Gleichstrommotor dient der Umwandlung von elektrischer Energie in mechanische Energie, mit der eine mechanische Arbeit verrichtet wird. Eine Drehbewegung, die man zum Antrieb anwendet, wird erzeugt. Die Funktion des Gleichstrommotors basiert auf dem Kraftwirkungsgesetz, indem eine Kraft (Lorenz-Kraft) auf einen stromdurchflossenen Leiter im Magnetfeld wirkt.

Der Gleichstrommotor besteht aus zwei grundlegenden Bauteilen: Erstens aus einem feststehenden Komponenten Ständerjoch, aus dem Polschenkel und den Polschuhen, sowie der sogenannten Erregerwicklung. Zweitens aus dem drehbaren Rotor (Anker) und der Ankerwicklung. Die Bürsten, die aus gesintertem Graphit angefertigt werden, binden Ständer und Anker.

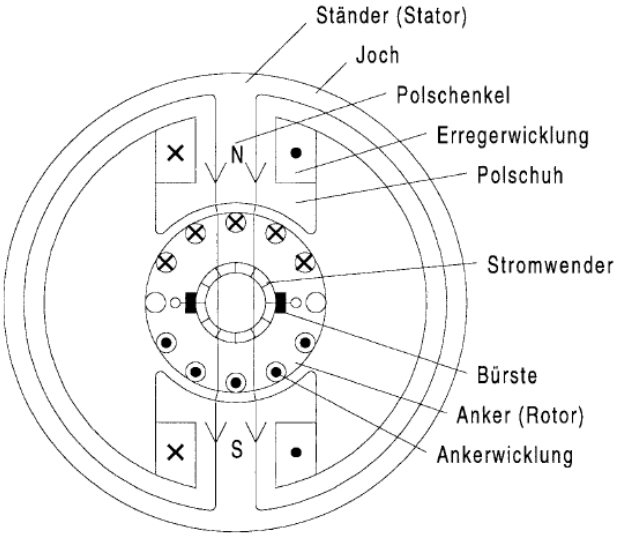


Abbildung 11: Der Trommelanker[3]

3.1.1 Ständerwicklung

Am Ständerjoch, das das Gehäuse der Maschine bildet, werden die Polschenkel angebracht. Über dem Polschenkel liegt die Erregerwicklung. Der Gleichstrom fließt in diesen Wicklungen, sodass der Erregerfluss Φ entsteht, der durch die Polschuhe zum Anker geleitet wird. Der gesamte Ständer (Ständerjoch, Polschenkel und Polschuhe) besteht aus ferromagnetischem Material.

3.1.2 Ringanker

Die Funktion des Gleichstrommotors lässt sich aus dem Läufer nach Paccinotti erkennen und die Läuferform wird als Ringanker definiert.

Dieser besteht aus einem Eisenring, der aus Einzelblechen zylindrisch aufgeschichtet ist, dadurch verringern sich die Eisenverluste bei der Rotation im Ständerfeld. Der Ringanker trägt die Ankerwicklung. In der folgenden Abbildung hat die Ankerwicklung 12 Windungen.

[3] Vgl. Prof. Dr. Jens Ginzel: Lehrskript Elektrische Antrieb, S.16

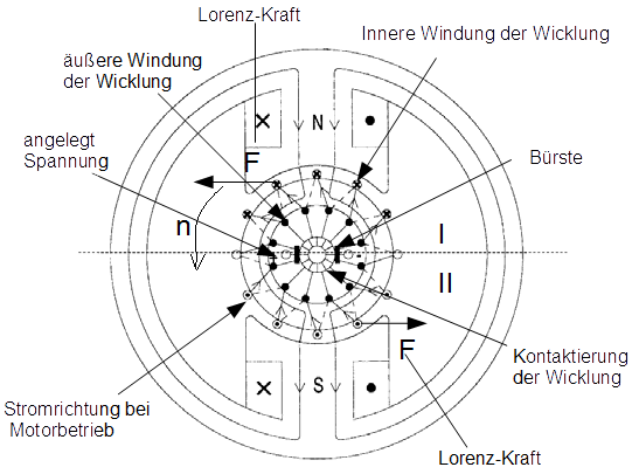


Abbildung 12: Der Ringanker nach Paccinotti[4]

Die Bürste teilt sich vom Anker aus in zwei parallele Zweige auf. Eine Spannung wird an die Bürste angelegt und der Strom wird über die Bürste der Ankerwicklung zugeführt. Mit dem Ständermagnetfeld und der stromdurchflossenen Ankerwicklung wird die Kraft (Lorenz-Kraft) erzeugt. Aufgrund dieser Kraftwirkung in der jeweils entgegengesetzten Richtung am unteren und oberen Ende des Ringankers, entsteht ein Drehmoment, welches wiederum eine Rotationsbewegung des Ringankers verursacht.

Wegen der äußeren Wicklung mit den verschiedenen Winkelstellungen ist das Drehmoment unterschiedlich. Das Drehmoment hängt von der Windungszahl ab. Je größer die Anzahl, desto gleichmäßiger ist das erzeugte Drehmoment.

3.1.3 Kommutator

In der Abbildung des dargestellten prinzipiellen Aufbaus schleifen die Bürsten direkt auf der Ankerwicklung. Eine Gleichspannung wird an den Bürsten des Motors angelegt, so dass eine mechanische Gleichrichtung aus den in den einzelnen Leitern entstehenden Wechselspannungen erfolgt. Bei der Drehung des Ringankers wird je eine Windung seiner Wicklung unter jeder Bürste kurzgeschlossen. Nun muss die Stromrichtung sich in diesen Windungen umkehren, bevor diese zur jeweils anderen Halbebene gehören. Diesen Vorgang nennt man Stromwendung oder Kommutierung.

3.2 Art der GM

Es gibt verschiedene Arten von Gleichstrommotoren, die sich in der Form der Erregung unterscheiden. Werden die Permanentmagnete zur Erzeugung des Hauptflusses verwendet, so spricht man von einem permanenterregten Gleichstrommotor. Wenn die

[4] Vgl. Prof. Dr. Jens Ginzel: Lehrskript Elektrische Antrieb, S.17

Erregerwicklung an einer gesonderten Stromversorgung angeschlossen ist, so handelt es sich um einen fremderregten Gleichstrommotor.

Die Anker- und Erregerwicklung kann entweder parallel- oder reihengeschaltet werden. Man spricht dann von Nebenschluss- bzw. Reihenschlussmaschinen. In den folgenden Abbildungen werden die Ersatzschaltungen des Nebenschluss- bzw. Reihenschlussmotors angezeigt.

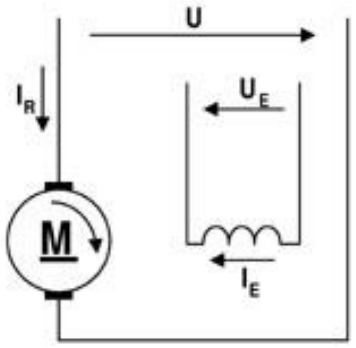


Abbildung 13: Nebenschlussmaschine

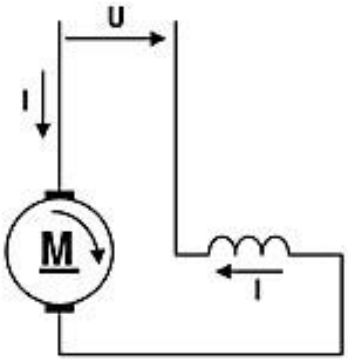


Abbildung 14: Reihenschlussmaschine

3.3 Mathematisches Modell der GM

Um den Gleichstrommotor zu simulieren, muss das mathematische Modell des Gleichstrommotors erstellt werden. Danach kann mit der Software Matlab/Simulink das Blockschaltbild aufgebaut werden.

3.3.1 Grundgleichungen

Die Gleichungen der GM basieren im elektrischen Teil auf der Lorenzkraft und das Induktionsgesetz. Der mechanische Teil besteht einfach aus den Bewegungsgleichungen für ein mechanisches System.

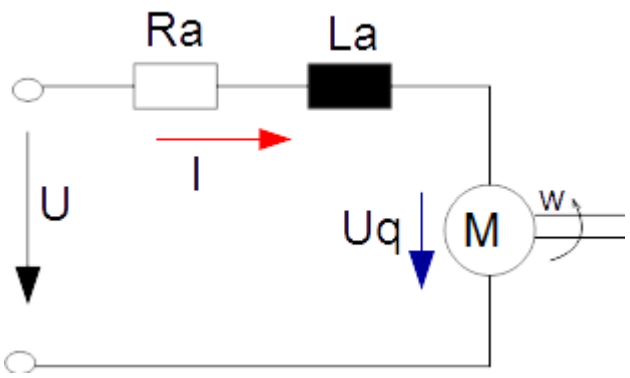


Abbildung 15: elektrisches Ersatzschaltbild der GM

Man legt eine Klemmenspannung Ua an. Ra und La bezeichnen den Ankerwiderstand bzw. die Ankerinduktivität. Durch die Drehung des Ankers entsteht eine induzierte Spannung im Anker. Im Ersatzschaltbild wird diese Spannung Uq bezeichnet.

Kirchhoffsches Gesetz:

$$Ua = IRa + La\frac{dI}{dt} + Uq \qquad \text{(Gl. 3.1)}$$

Die induzierte Spannung hängt von der Maschinenkonstante K ab:

$$Uq = Kw \qquad \text{(Gl. 3.2)}$$

Ersetzt die Gleichung (3.2) in (3.1)

$$Ua = IRa + La\frac{di}{dt} + Kw \qquad \text{(Gl. 3.3)}$$

Das elektrische Drehmoment des GM:

$$Ma = KI \qquad \text{(Gl. 3.4)}$$

Die Bewegungsgleichung des GM:

$$J\frac{dw}{dt} = Ma - Mr \qquad \text{(Gl. 3.5)}$$

$$J\frac{dw}{dt} = KI - Bw \qquad \text{(Gl. 3.6)}$$

mit den Parametern

Ua: Ankerspannung in V

Uq: induzierte Spannung in V

Ra: Ankerwiderstand in Ω

La: Ankerinduktivität in H

Mr = Bw: viskoses Reibmoment

J: Trägheitsmoment in kgm^2

K: Maschinenkonstante

In Simulink realisiert man die Differenzengleichung (DGL) am besten, indem man sie nach der höchsten vorkommenden Ableitung auflöst.

Die Gleichung (3.1) wird umgestellt:

$$\frac{dI}{dt} = \frac{Ua}{La} - \frac{Ra}{La} I - \frac{K}{La} w \qquad \text{(Gl. 3.7)}$$

Mit der DGL (3.6) und (3.7) kann man das Blockschaltbild des GM aufbauen. Mit der Software Matlab/Simulink wird der Gleichstrommotor simuliert.

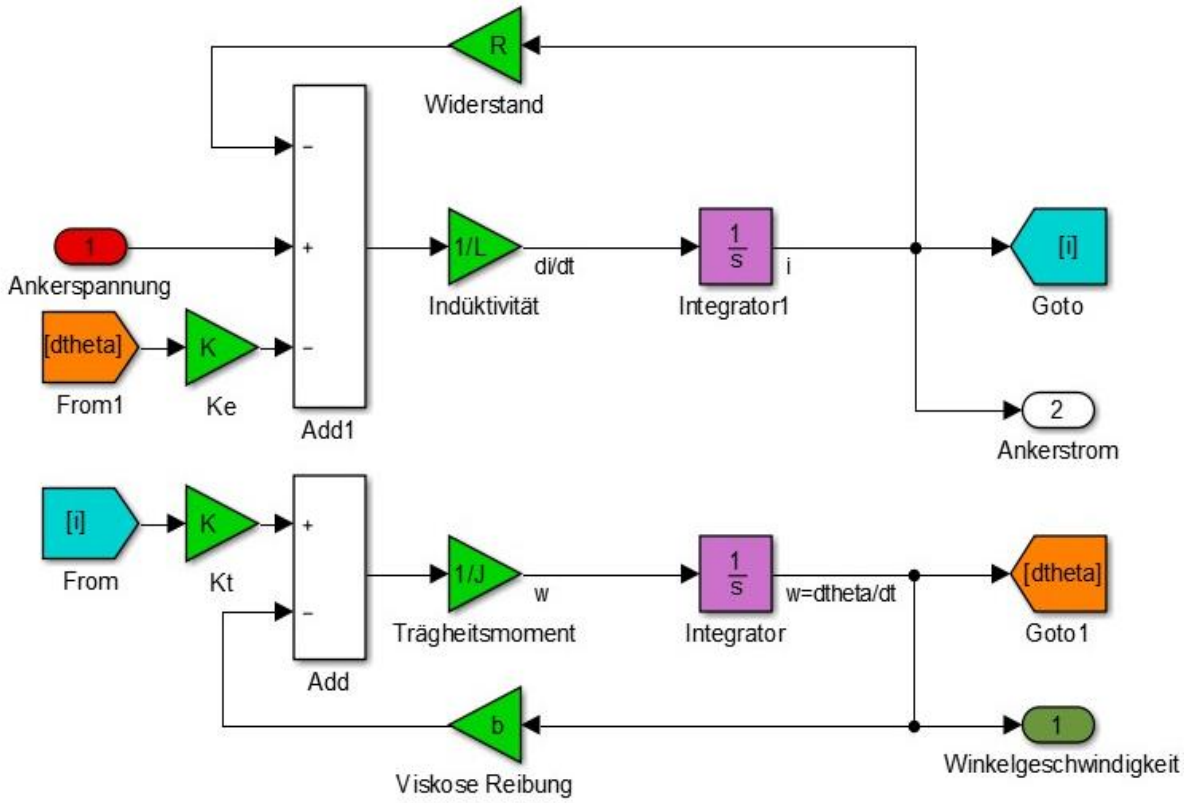


Abbildung 16: Blockschaltbild der GM mit Matlab/Simulink

Dieses Blockschaltbild wird in dem nächsten Kapitel verwendet, um die Parameter der GM zu identifizieren.

4 Parameteridentifikation der GM

Im vorherigen Kapitel wurden die Grundlagen und die Funktion der GM erklärt und das mathematische Modell wurde aufgebaut. Dieses Kapitel handelt von der Parameteridentifikation.

Um ein System zu simulieren, werden die Modellgleichungen und Modellparameter zuerst bestimmt. Häufig gibt es das Problem, dass die Parameter der Maschinen nicht vorhanden sind oder es gibt kaum Möglichkeiten, die Parameter zu messen. In dieser Arbeit müssen alle Parameter des Getriebemotors bestimmt werden. Die folgende Abbildung zeigt das Experiment der Systemidentifikation für den Gleichstrommotor.

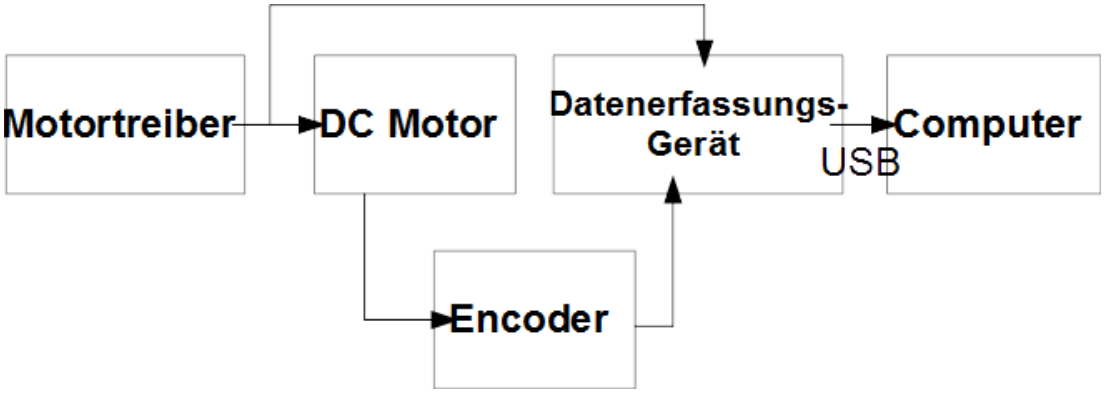


Abbildung 17: Modell der Identifikation der Getriebemotorparameter

Im Industriebereich nutzt man dafür ein Datenerfassungsgerät, damit die Daten der Messung von der zeitabhängigen Drehzahl mit der bestimmten Eingangsspannung aufgenommen werden können. Aber das Datenerfassungsgerät ist teuer (z.B. das NI USB-6509 vom Hersteller National Instruments kostet 495€) und steht im JBW nicht zur Verfügung.

Durch die spezifische Simulink-Software (Matlab/Simulink) kann man das Modell des dynamischen Systems von diesen Messdaten aufbauen. In dieser Arbeit wird ein vorhandenes Arduino-Board als Datenerfassungsgerät genutzt und die „theoretische Methode“ der Softwaresimulation wird zur Parameteridentifikation durch den Einsatz der Software Simulink/Matlab oder LabVIEW angewendet. Dazu wird die Methode der kleinsten Quadrate verwendet und die Parameter der GM werden aus einer Vielzahl von Messungen bestimmt.

Die folgende Abbildung zeigt das Verfahren der Parameteridentifikation

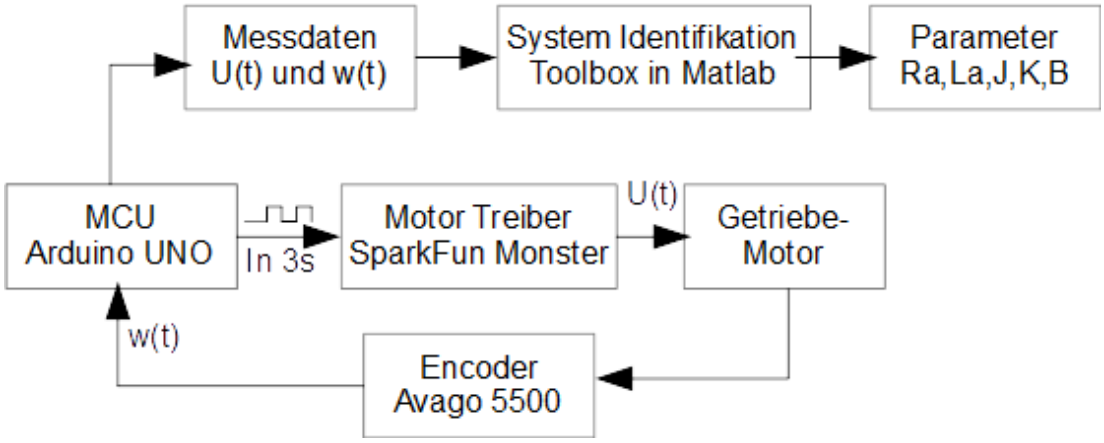


Abbildung 18: Konzept der Parameteridentifizierung der GM

4.1 Modellierungsprozess

Bevor die Messung der GM durchgeführt werden kann, muss man die Beziehung zwischen der Ankerspannung Ua und der Winkelgeschwindigkeit w bestimmen. Die Ankerspannung U(t) ist eine zeitabhängige Eingangsfunktion des Messsystems und die Ausgangsfunktion ist die Winkelgeschwindigkeit w(t).

Die Übertragungsfunktion beträgt:

$$G(t) = \frac{w(t)}{U(t)} \qquad \text{(Gl. 3.8)}$$

Um das Übertragungsglied zwischen U(t) und w(t) zu bekommen, muss der Strom I aus der Gleichung (3.6) eliminiert werden:

$$I = \frac{J}{K} * \frac{dw(t)}{dt} + \frac{B}{K} * w(t) \qquad \text{(Gl. 3.9)}$$

Die Gleichung (3.8) muss integriert werden:

$$\frac{dI}{dt} = \frac{J}{K} * \frac{dw(t)}{dt^2} + \frac{B}{K} * w(t) \qquad \text{(Gl. 3.10)}$$

Ersetzt (3.9) und (3.10) in der Gleichung (3.3):

$$U(t) = \frac{LJ}{K} * \frac{dw(t)}{dt^2} + \frac{RJ+LB}{K} * \frac{dw(t)}{dt} + \left(K + \frac{RB}{K}\right) * w(t) \qquad \text{(Gl. 3.11)}$$

Die Koeffizienten werden zusammengefasst

$$U(t) = a * \frac{dw(t)}{dt^2} + b * \frac{dw(t)}{dt} + c * w(t) \qquad \text{(Gl.3.12)}$$

Mit $a = \frac{LJ}{K}$; $b = \frac{RJ+LB}{K}$; $c = (K + \frac{RB}{K})$

Die obere Differentialgleichung wird in den Laplace-Bereich transformiert:

$$U(s) = W(s) * (as^2 + bs + c) \qquad \text{(Gl. 3.13)}$$

Die Übertragungsfunktion ist:

$$G(s) = \frac{W(s)}{U(s)} = \frac{1}{(as^2+bs+c)} \qquad \text{(Gl. 3.14)}$$

Damit ist die Übertragungsfunktion eines Gleichstrommotors bestimmt. Im nächsten Schritt wird die mathematische Methode der kleinsten Quadrate aufgestellt.

4.2 Methode der kleinsten Quadrate

4.2.1 Motivation und Grundlagen

Die Gleichung (1.11) beschreibt die Beziehung zwischen der Ankerspannung U(t) und der Winkelgeschwindigkeit w(t) in Abhängigkeit von der Zeit t. Nach der Laplace Transformation bekommt man die quadratische (polynomische mit Grad 2) Beziehung zwischen der Ankerspannung U(t), der Winkelgeschwindigkeit w(t) und der Zeit t. In diesem Kapitel wird das einfache Messsystem aufgebaut, damit die Ausgangswerte (Winkelgeschwindigkeit) mit dem rechteckförmigen Eingangssignal (Ankerspannung) in der Zeit von 3 Sekunden aufgenommen werden. Weil das Eingangssignal konstant (12V) ist, ist nur die Geschwindigkeit in der Zeit von 10 Millisekunden (Abtastzeit) von Interesse. Die Auswahl der Abtastzeit wird im Weiteren erklärt.

Die Grafik der Datenpunkte beschreibt die Messwertpaare (t, w). Wenn alle Datenpunkte auf einer Geraden oder einem Polynom lägen, dann hätte die Messung keine Messfehler und der Zusammenhang zwischen den Wertepaaren wäre linear oder quadratisch. Aber im allgemeinen Fall gibt es Messfehler und das offizielle Ziel dieser Methode ist die Bestimmung der Kurve zu den Datenpunkten. Diese Kurve verläuft möglichst nahe an den Datenpunkten. Von der folgenden Abbildung kann man den Unterschied zwischen dem mathematischen Modell und dem gemessenen System in Echtzeit erkennen.

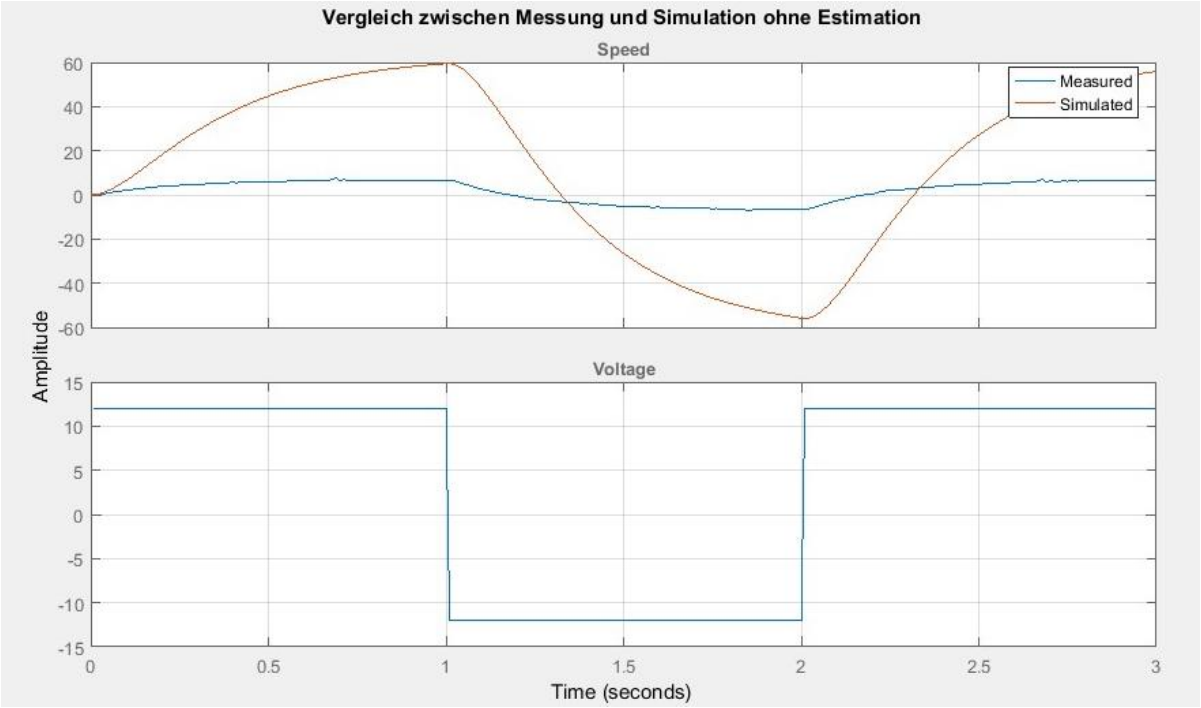


Abbildung 19: Vergleich zwischen Messung und Simulation ohne Estimation

Die rote Grafik beschreibt das mathematische Modell der GM und die blaue Grafik zeigt die gemessenen Messdatenpunkte. Man nutzt die Methode der kleinsten Quadrate, um die rote Grafik möglichst nahe an die blaue Grafik zu verschieben.

In der Stochastik wird die Methode der kleinsten Quadrate meistens als Schätzmethode in der Regressionsanalyse genutzt. Mit dieser Methode kann man die Parameter Ra, La, J, K und B der GM ermitteln.

Nachfolgend sollen drei Formen der Regressionsanalyse dargestellt werden und zwar

- die lineare,
- die quadratische (polynomische mit Grad 2) und
- die exponentielle Regression.

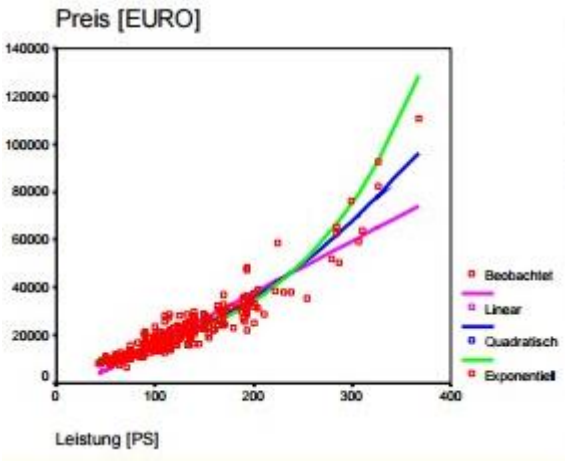


Abbildung 20: Beispiel für die Abhängigkeit zwischen Preis und Leistung[5]

Die allgemeine Funktion der Regression im eindimensionalen Fall lautet

y = f(x) + e

mit y: die abhängige Variable

x: die unabhängige Variable

f(x): die gesuchte Funktion

e: der Fehler bzw. das Residuum des Modells

Quadratische Regression

Aus der Gleichung (3.13) bekommt man die quadratische Funktion:

$y = f(x) = ax^2 + bx + c$

Man ersetzt die Variable x mit der Variable t und y mit w

Die Messpunkte sind die Wertepaare (t1,w1), (t2,w2), (t3,w3)…, so trifft die passende Polynomgleichung $w = f(x) = at^2 + bt + c$ in der Regel nicht genau auf die Punkte, sondern es gibt bei jedem Wertepaar einen großer Fehler e

$w1 = at1^2 + bt1 + c + e1$

$w2 = at2^2 + bt2 + c + e2$

$w3 = at3^2 + bt3 + c + e3$

….

$wn = atn^2 + btn + c + en$

[5]Vgl. Prof. Küch/ Dr. Ricabal Delgado: Lehrstuhl Statistik http://www.wiwi.uni-rostock.de/fileadmin/Institute/VWL/LS_Statistik/vorl_gs/Regression_II.pdf Abruf: 2016-11-15

Nun muss man diesen Fehler e bestimmen.

$$en = wn - (atn^2 + btn + c) \qquad \text{(Gl. 3.15)}$$

Aus der Gleichung (3.5) kann man erkennen, dass der Fehlerwert entweder positiv oder negativ ist. Wenn der Wert negativ ist, können die Endergebnisse der Fehler nicht genau berechnet werden. Dies bedeutet, dass dieser Wert unbedingt positiv sein muss.

Aus der Gleichung (3.15) quadrieren beide Seiten

$$\sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (w_i - at_i^2 - b\,t_i - c)^2$$

d.h. die Straffunktion ist

$$F(a,b,c) = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} (w_i - at_i^2 - b\,t_i - c)^2$$

Um das Ziel der Methode der kleinsten Quadrate zu erreichen, muss die Summe der quadrierten Fehler minimiert werden, sodass $e_1^2 + e_2^2 + e_3^2 + \cdots e_n^2$ möglichst klein wird.

Die Straffunktion ist die polynomische Gleichung und die Voraussetzung der Minimierung ist die erste Ableitung gleich 0.

Diese Funktion hat drei unbekannte Variablen a, b, c. Man muss die partielle Ableitung der Straffunktion mit den Variablen a, b, c berechnen.

d.h. $\frac{\partial F}{\partial a} = \frac{\partial F}{\partial b} = \frac{\partial F}{\partial c} = 0$

$$\frac{\partial}{\partial a} F(a,b,c) = -2 \sum_{i=1}^{n} (w_i - at_i^2 - b\,t_i - c)(t_i^2) = 0$$

$$\frac{\partial}{\partial b} F(a,b,c) = -2 \sum_{i=1}^{n} (w_i - at_i^2 - b\,t_i - c)(t_i) = 0$$

$$\frac{\partial}{\partial c} F(a,b,c) = -2 \sum_{i=1}^{n} (w_i - at_i^2 - b\,t_i - c) = 0$$

Man schreibt alle Gleichungen in ein lineares Gleichungssystem:

$$\begin{pmatrix} \sum_{i=1}^{n} t_i^4 & \sum_{i=1}^{n} t_i^3 & \sum_{i=1}^{n} t_i^2 \\ \sum_{i=1}^{n} t_i^3 & \sum_{i=1}^{n} t_i^2 & \sum_{i=1}^{n} t_i \\ \sum_{i=1}^{n} t_i^2 & \sum_{i=1}^{n} t_i & n \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} t_i^2 w_i \\ \sum_{i=1}^{n} t_i \;\; w_i \\ \sum_{i=1}^{n} w_i \end{pmatrix} \qquad \text{(Gl. 3.16)}$$

Man ersetzt alle Messwertpaare (t, w) in der Gleichung (3.16) und nutzt den Gauß - Algorithmus, dann können die drei Variablen a, b, c errechnet werden.

In der praktischen Phase kann man auf die obige statische Theorie basieren, damit das mathemische Modell des dynamischen Systems mit Hilfe der „System Identifikation Toolbox[6]“ in Matlab/Simulink von gemessenen Eingangs- und Ausgangswerten dargestellt werden kann. In der Arbeit werden die Drehzahlwerte als

[6] Vgl. DC Servo Motor Parameter Estimation http://de.mathworks.com/help/sldo/examples/dc-servo-motor-parameter-estimation.html Abruf 2016-11-15

Ausgangswerte mit der bestimmten Eingangsspannung in Abhängigkeit der Zeit gemessen. Diese Daten werden durch die System Identifikation Toolbox analysiert und verarbeitet und anschließend kann durch das mathematische Blockschaltbild das Modell mit den bestimmten Parametern (Ra, La, J, K und B) dargestellt werden.

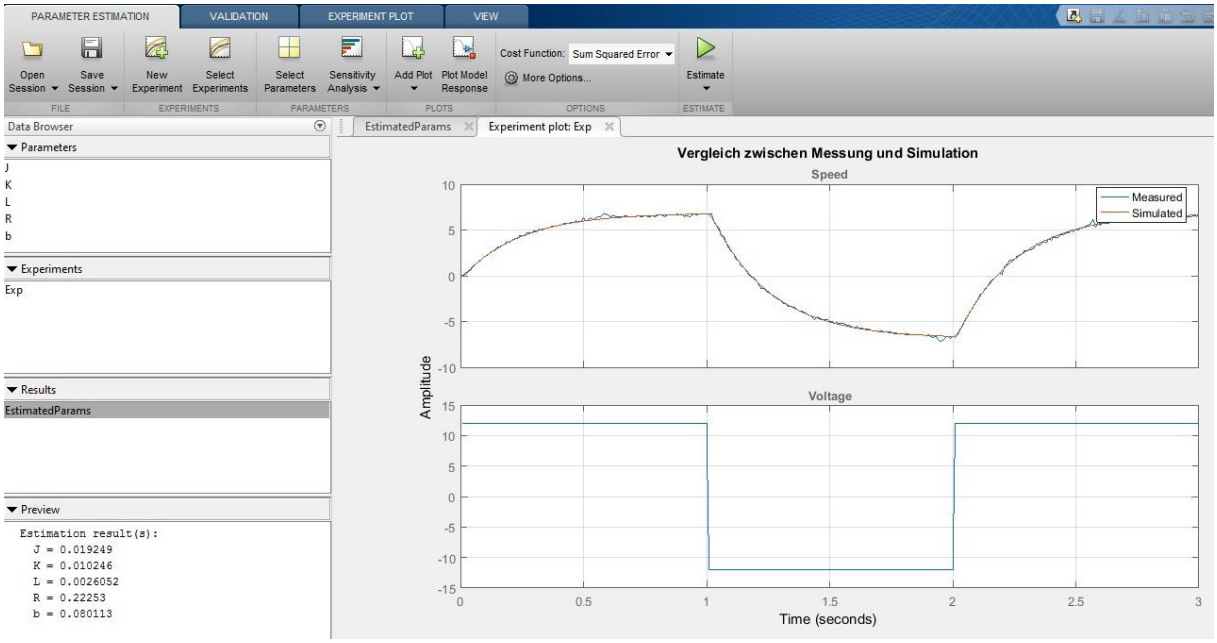


Abbildung 21: System Identifikation Toolbox mit Matlab/Simulink

Im nächsten Schritt wird die Hardware zur Messung der GM aufgebaut und die Experimente werden realisiert. Dann werden die Messdaten aufgenommen. Dies ist die Vorbereitung für die Parameteridentifikation.

4.3 Arduino Uno Boards

In diesem kleinen Messsystem benutzt man eine günstige Mikrocontroller-Plattform, ein Arduino-Board. Das Arduino-Board besteht aus einem einfachen E/A-Board mit einem Mikrocontroller und analogen und digitalen Ein- und Ausgängen. Die Hardware eines typischen Arduinoboards basiert auf einem AVR-Mikrocontroller (US-amerikanische Hersteller Atmel). Um die Programmierung des Arduinoboards zu vereinfachen, entwickelte der Hersteller eine eigene integrierte Entwicklungsumgebung (IDE) mit den Programmiersprachen C und C++. Diese IDE ist kostenlos verfügbar und für die verschiedenen Plattformen wie Windows, Linux und MacOS geeignet. Die IDE basiert auf Processing, einer objektorientierten und typisierten Programmiersprache, die für die Einsatzbereiche Grafik, Simulation und Animation spezialisiert ist. In folgender Tabelle wird das Arduino-Board mit anderen Mikrocontrollersystemen (Raspberry oder Mikrocontroller von den verschiedenen Herstellern z.B. Mikrochips) verglichen:

Tabelle 4: Vergleich zwischen Arduino und anderen Mikrocontrollersystemen

Arduino-Boards	andere Mikrocontrollersysteme
• Kompaktes System	• Das System muss aufgebaut werden
• Einfache Programmierung mit großer Bibliothek und Programmiersprache C und C++	• Komplizierte Programmierung und Programmiersprache mit C, C++ und Assemblersprache.
• Entwicklungsumgebung ist Open-Source und funktioniert mit verschiedenen Plattformen	• Ungünstiger Preis und mit eingeschränkten Plattformen
• Funktion nur mit vorhandenem Mikrocontroller, Ersatz nicht möglich	• Unkomplizierter Ersatz vom Mikrocontroller

Heutzutage wird das Arduinoboards in verschiedenen Bereichen verwendet:

- Elektronik
- Robotik
- Steuerung
- Sensordatenerfassung

In dieser Arbeit wird das Arduinoboards zur Steuerung des Fahrsimulators sowie die Sensordatenerfassung für die Parameteridentifikation der GM verwendet. Das Arduinoboards gibt es in drei Versionen mit einer unterschiedlichen Anzahl von digitalen Ein- und Ausgängen und verschiedenen Mikrocontrollern.

Tabelle 5: verschiedene Versionen von Arduino[7]

Eigenschaften			
	Arduino UNO	Arduino NANO	Arduino MEGA
Mikrocontroller	ATmega328	ATmega328	ATmega2560
Betriebsspannung	5V	5V	5V
Digitale I/O Pins	14	14	54
Analoge Eingänge	6	8	16
Auflösung Analogwerte	10 bit -1024	10 bit - 1024	10 bit - 1024
PWM Channel	6	6	14
Flash Memory	32 KB	32 KB	256 KB
EEPROM	1 KB	1 KB	4 KB
Taktfrequenz	16 MHz	12 MHz	16 MHz
Serielle Pin	2 (1,2)	2 (1,2)	8 (1,2,14-19)
Externe Interrupts	2	2	6

In dem Fahrsimulator soll das Arduinoboards UNO für die Steuerung des Systems verwendet werden. Für die Steuerung der GM braucht man den Vierquadrantensteller, der aus einer elektronischen H-Brückenschaltung besteht. In dem nächsten Kapitel wird die Funktion der H-Brückenschaltung diskutiert.

[7] Homepage: http://www.arduino.cc

4.4 Grundsätzliche H-Brückenschaltung

Die H-Brückenschaltung besteht aus vier Transistoren. Die Hauptaufgabe dieser Schaltung ist die Ansteuerung der GM in beiden Drehrichtungen. Nun interpretiert man die grundsätzliche Funktion der H-Brückenschaltung.

Um die Erklärung der Funktion zu erleichtern, werden vier Transistoren mit vier einfachen Schaltern S1, S2, S3 und S4 ersetzt. In folgender Abbildung ist der einfache Schaltplan der H-Brückenschaltung zu sehen.

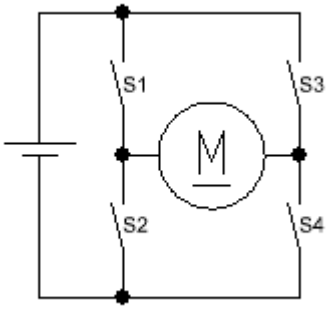


Abbildung 22: einfache-Brückenschaltung mit Schalter

Vier Schalter werden durch eine Gleichspannung versorgt. Wenn man einen GM in beide Richtungen betreiben will, muss man dafür sorgen, dass die Stromrichtung durch den GM geändert werden kann.

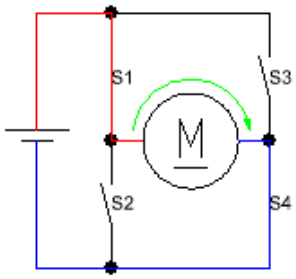


Abbildung 23: Motor im Rechtlauf

Wenn beide Schalter S1 und S4 eingeschaltet sind, fließt der Strom über S1 zum Motor und über S4 wieder zurück zur Spannungsquelle. Dies bedeutet, dass der Motor im Rechtslauf dreht.

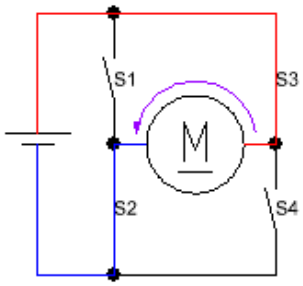


Abbildung 24: Motor im Linkslauf

Wenn beide Schalter S3 und S2 eingeschaltet sind, fließt der Strom über S3 zum Motor und über S2 wieder zurück zur Spannungsquelle. Dies bedeutet, dass der Motor im Linkslauf dreht.

Um den Motor per Mikrocontroller anzusteuern, werden die Schalter mit verschiedenen Bauelementen ersetzt:

- mit dem Relais
- mit dem BJT (Bipolar Junction Transistor)
- mit dem MOSFET (Metal Oxide Semiconductor Field- Effect Transistor)

Die Verwendung dieser Arten ist abhängig von der angeforderten Leistung des Systems.

4.4.1 Mit dem Relais

In der folgenden Abbildung werden zwei Relais als zwei Schalter verwendet, damit der Motor in beide Drehrichtungen angesteuert werden kann.

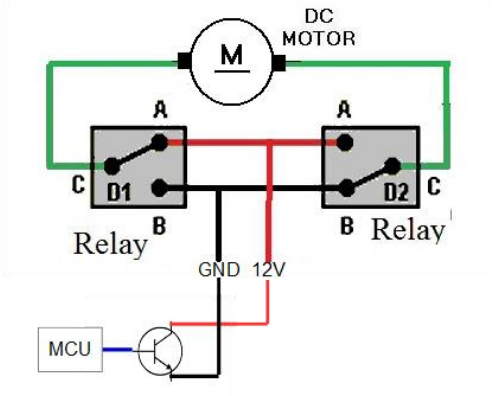


Abbildung 25: Motorsteuerungsrelais

Ein Steuersignal wird von dem Mikrocontroller über einen Transistor gesendet. Dieses Signal steuert das Umschalten von beiden Relais an. In der vorherigen Abbildung wird

das linke Relais mit der Spannung 12V verbunden und das rechte Relais mit dem GND. Der Motor dreht im Rechtslauf. Durch den Einsatz von Relais können Motoren mit großer Leistung geschaltet werden. Das Relais ist der „mechanische Kontakt “ d.h. nur einschalten und ausschalten. Deshalb ist das Relais nicht geeignet für das PWM (engl. Pulse Width Modulation) -Signal, was bedeutet, dass die Geschwindigkeit vom Motor nicht geregelt werden kann. Dies ist ein großer Nachteil des Relais.

Um die Geschwindigkeit mit dem PWM Signal zu regeln, muss man den „mechanischen“ Kontakt (Relais) durch den „elektrischen“ Kontakt (BJT oder MOSFET) ersetzen.

4.4.2 Mit dem BJT Transistor

Folgend werden die grundsätzliche Funktion des BJTs (bipolar Junction Transistor) und dann die Funktion der H-Brückenschaltung mit dem BJT beschrieben.

Der bipolare Transistor wird durch einen elektrischen Strom angesteuert. Auf Grund des Aufbaus unterscheidet man zwei Arten von Bipolar Transistoren:

Die Funktionen von beiden Arten sind identisch, im Folgenden wird die Funktion vom npn-Transistor erklärt:

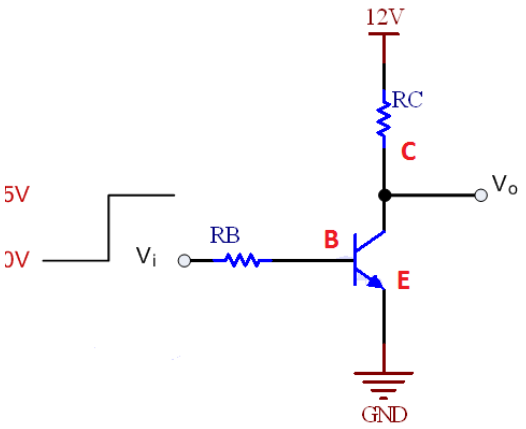


Abbildung 26: Motorsteuerung mit npn Transistor

Wenn die Eingangsspannung per Mikrocontroller mit einem rechteckigen Signal (0 oder 5V) versorgt wird, hat das Signal von der Ausgangsspannung auch eine rechteckige Form (12V oder 0V). Zum Beispiel ist die versorgte Eingangsspannung 0V, dann fließt der Strom über den Widerstand RB nicht und der Strom über den Widerstand RC ist sehr klein. Die Ausgangsspannung Vo ist 12V.

Ist aber die Eingangsspannung 5V und der Strom über den Widerstand RC erreicht den maximalen Wert und die Kollektor-Emitter-Spannung UCE ist gleich 0, dann bedeutet dies, dass die Ausgangsspannung Vo 0V ist. Deshalb ist das Signal von der Ausgangsspannung auch ein rechteckförmiges Signal bzw. PWM-Signal in Abhängigkeit von der Zeit. Wie aus der Gleichung (1.13) ersichtlich, ist die Geschwindigkeit der GM abhängig von der Spannung. Die Geschwindigkeit wird durch diese Spannung angesteuert. Die folgende Abbildung zeigt ein Beispiel des PWM-Signals:

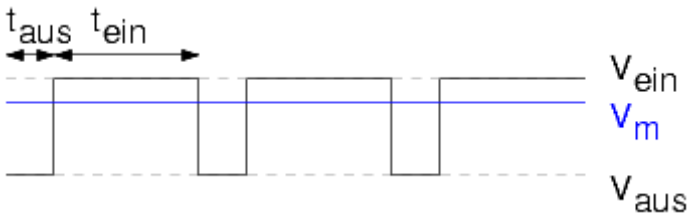


Abbildung 27: Das Beispiel des PWM-Signals

Die Beziehung zwischen der Ein- und Ausgangsspannung von dem PWM-Signal wird durch die folgende Formel beschrieben:

$$V_m = V_{cc} * \frac{t_{ein}}{t_{ein}+t_{aus}} = p * V_{cc}$$

Oder Motorspannung = Betriebsspannung *Tastverhältnis

Mit p: Tastverhältnis (engl. Duty Cycle)

In der Praxis nutzt man den Mikrocontroller mit Hilfe des Timers bzw. Counters, um das PWM-Signal zu erzeugen. Die Geschwindigkeit kann durch die einstellbare Impulsbreite angesteuert werden.

Die folgende Abbildung zeigt die Schaltung mit dem BJT-Transistor

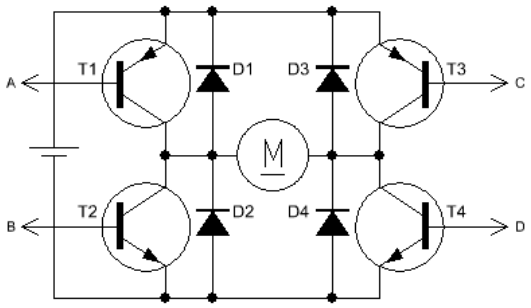


Abbildung 28: Steuerschaltung mit dem BJT-Transistor

Das Umschalten von den Eingängen T1, T2, T3 und T4 wird mit dem Mikrocontroller geregelt. Man verwendet 4 Dioden (Schottky-Diode), damit die Transistoren und der Mikrocontroller vor der induzierten Spannung geschützt werden. Für die obere Schaltung werden zwei pnp-Transistoren benutzt und für die untere Schaltung zwei npn-Transistoren.

Die H-Brückenschaltung mit dem BJT-Transistor hat folgende Vorteile:

- Hohe Strombelastbarkeit
- Sehr geringer Spannungsabfall
- Geringe Kosten

Nachteil: Die Transistoren müssen mit teuren Schutzdioden betrieben werden. Deshalb verwendet man die H-Brückenschaltung mit einem MOSFET. Bei modernen MOFETs kann auf Schutzdioden verzichtet werden.

4.4.3 Mit dem MOSFET

Mit dem Vorteil der hohen Strombelastbarkeit (d.h. hohes Drehmoment) wird die H-Brückenschaltung mit einem MOSFET in dem Fahrsimulator verwendet.

Die Grundlagen des MOSFETs sind ähnlich wie bei einem BJT-Transistor. Im Unterschied zu den bipolaren Transistoren sind die Anschlüsse mit Gate (G), Drain (D) und Source (S) bezeichnet. Das Gate entspricht der Basis eines Transistors, Drain dem Kollektor und Source dem Emitter. Wie bei den BJT-Transistoren gibt es bei den MOSFETs N- und P-Typen.

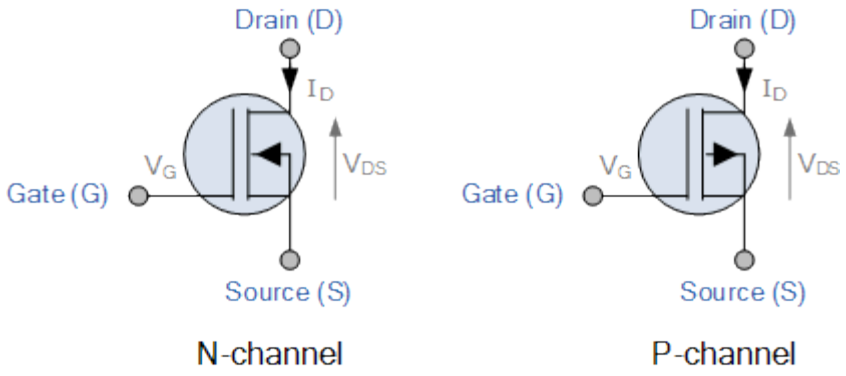


Abbildung 29: Schaltzeichen von MOSFET N- und P-Channel

Wenn eine Gleichspannung bei dem G(Gate)-Anschluss angelegt wird, ist der MOSFET gesperrt oder geleitet. Die Umschaltung des MOSFETs ist abhängig von der Gate-Spannung. Bei einer Gate-Spannung (U_GS) von 0V sperrt der N-MOSFET, weil der Widerstand zwischen Gate- und Source-Anschluss sehr groß ist. Im Gegenteil leitet der P-MOSFET bei dieser Gate-Spannung. Wenn die Betriebsspannung (allgemein von 5V) bei dem Anschluss Gate angelegt wird, verhält es sich genau umgekehrt. Der N-MOSFET leitet und der P-MOSFET sperrt.

Heutzutage wird der MOSFET in der H-Brückenschaltung verwendet, weil die Strombelastbarkeit des MOSFETs höher ist als die des BJT-Transistors. Eine H-Brückenschaltung besteht aus vier MOSFET (zweimal N-MOSFET und zweimal P-MOSFET). Die Anordnung der MOSFET wird in dem nächsten Schritt erklärt.

Nun wird die Schaltung zwischen N-MOSFETs und dem Motor betrachtet. Die folgende Abbildung zeigt diese Schaltung

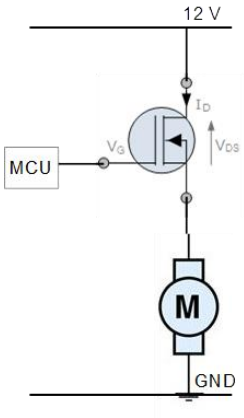


Abbildung 30: Steuerschaltung von N-MOSFET mit Mikrocontroller (MCU)

Bei der Gate-Spannung von 0V sperrt der N-MOSFET und der Widerstand zwischen D und S ist somit sehr groß. Die Spannung bei dem Source-Anschluss ist gleich 0V und der Motor dreht sich nicht.

Wenn man die Gleichspannung an den Gate-Anschluss legt, ist der Widerstand zwischen D und S sehr klein und die Spannung bei dem Source-Anschluss ist gleich der Betriebsspannung (12V). Der MOSFET wird geleitet und der Motor dreht sich. Um den MOSFET einzuschalten, ist die Spannung bei dem Gate-Anschluss größer als die Spannung bei Source (d.h. UG>12V, weil die Spannungsversorgung für den Motor 12V beträgt). Die Gate-Anschluss wird mit einem Pin des Mikrocontrollers verbunden und durch das Steuerprogramm kann man den logischen Ausgangswert (High oder Low, bzw. 1 und 0) von diesem Pin ansteuern. Bei dem Wert 1 hat die Gate-Spannung (UG) den gleichen Wert wie die Ausgangsspannung des Pins des Mikrocontrollers. Weil aber die Ausgangsspannung des Mikrocontrollers auf einen maximalen Wert von 5V begrenzt ist, ist eine direkte Verbindung zwischen dem Mikrocontroller und dem Gate-Anschluss nicht möglich. Um dieses Problem zu lösen, wird der N-MOSFET durch einen P-MOSFET ersetzt.

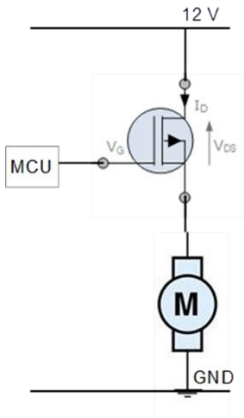


Abbildung 31: Steuerschaltung von P-MOSFET mit Mikrocontroller (MCU)

Wenn die Ausgangsspannung (5V) des Mikrocontrollers mit dem Gate-Anschluss verbunden ist, ist die Gate-Spannung (UG) kleiner als die Source-Spannung (US=12V). Dies bedeutet, dass der MOSFET geleitet hat d. h. der Motor wird mit der Spannung 12 V versorgt und er dreht sich um.

Auf diesem Grund muss man zwei P-MOSFET für obere H-Brückenschaltung verwenden und zwei N-MOSFET für die untere Schaltung. In der folgenden Abbildung wird die komplette H-Brückenschaltung gezeigt

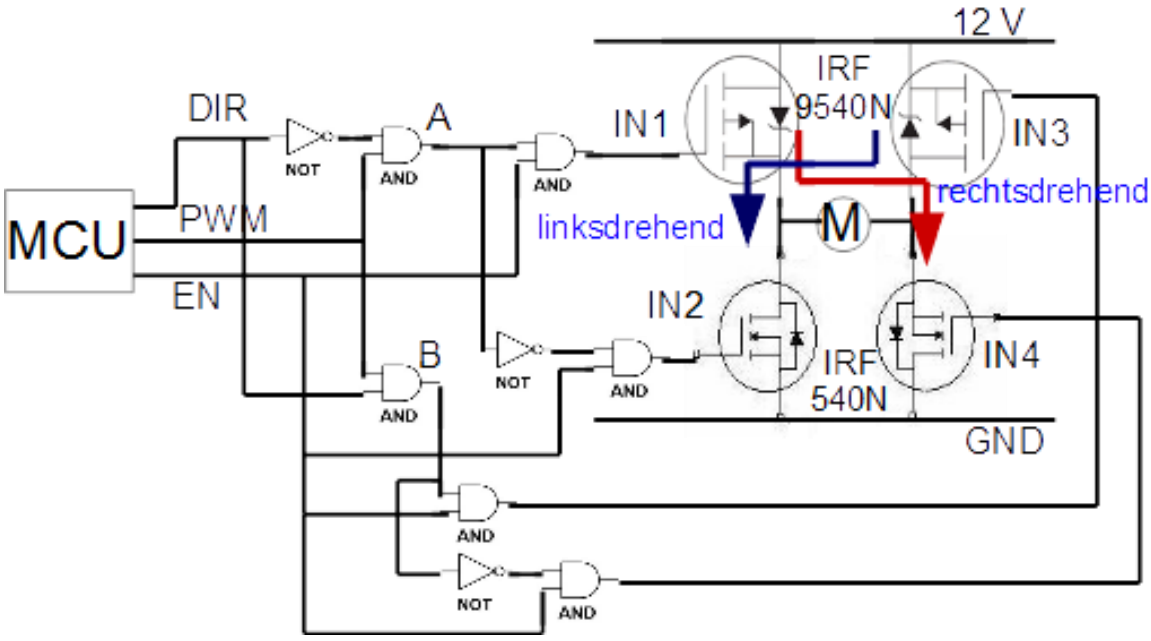


Abbildung 32: Die komplette H-Brückenschaltung mit den MOSFETs

Um das Ein- und Ausschalten der H-Brückenschaltung anzusteuern, werden die vier Eingänge IN1, IN2, IN3 und IN4 mit den Ausgängen vom Mikrocontroller (MCU) verbunden. Dies bedeutet, dass man vier Ausgänge des Mikrocontrollers braucht. Um die Anzahl von den Ausgängen zu reduzieren, kann man die Logik-Bausteine (AND- und NOT-Verknüpfung) verwenden. Außerdem kann die H-Brückenschaltung mit den Logik-Bausteinen vor einem Kurzschluss geschützt werden. Wenn beide MOSFET

einer Seite auf leitend geschaltet werden, ist das ein Kurzschluss. Mit der logischen Ansteuerung durch den Mikrocontroller wird die Umschaltung von vier MOSFET gesichert. Die 1. Diagonale ist leitend (oder sperrend) und die 2. Diagonale ist sperrend (oder leitend). Das ist der Hauptgrund für die Verwendung von logischen Verknüpfungen.

Nun betrachtet man die Funktion der kompletten H-Brückenschaltung. Die Geschwindigkeit und Drehrichtung der GM werden von drei Ausgängen des Mikrocontrollers durch die H-Brückenschaltung angesteuert. Ein DIR-Signal steuert die Drehrichtung, das PWM-Signal ist für die Geschwindigkeit und das EN-Signal steuert das Aktivieren der H-Brückenschaltung.

Die folgende Tabelle zeigt die Zustände der GM:

Tabelle 6: Zustände der GM

PWM	DIR	A	B	IN1	IN2	IN3	IN4	MOTOR
0	0	0	0	0	0	0	0	stoppend
0	1	0	0	0	0	0	0	stoppend
1	0	1	0	1	0	0	1	rechtsdrehend
1	1	0	1	0	1	1	0	linksdrehend

Von der logischen Tabelle wird die logische Ansteuerung durch den Mikrocontroller (Arduino UNO) durchgeführt. Hier ist ein Beispiel-Code für die Ansteuerung der H-Brückenschaltung mit dem Motor Treiber „Monster“ der Firma SparkFun:

```
void motorGo(uint8_t motor, uint8_t direct, uint8_t pwm)
{
  if (motor <= 1)
  {
    if (direct <=4)
    {
      // Set inA[motor]
      if (direct <=1)
        digitalWrite(inApin, HIGH);
      else
        digitalWrite(inApin, LOW);
```

```
      // Set inB[motor]
      if ((direct==0)||(direct==2))
        digitalWrite(inBpin, HIGH);
      else
        digitalWrite(inBpin, LOW);

      analogWrite(pwmpin, pwm);
    }
  }
}
```

4.4.4 Motor Treiber Monster

Heutzutage produziert der Hersteller die komplette H-Brückenschaltung als integrierte Schaltung (engl. Integrated circuit, kurz IC) mit inklusiven Logik-Bausteinen. Man nennt dies Motortreiber (engl. Motor Drivers). Im dem Praktikum wurde der Fahrsimulator mit dem Motortreiber Monster aufgebaut. Das System hat mit Einschränkung funktioniert, d.h. das Gewicht des Spielers konnte max. 50 kg betragen und die Spielzeit dauerte nur 15 bis 20 Minuten, weil zwei Motoren mit ungenügenden Leistung benutzt wurden und der Motortreiber Monster schnell überhitzte. In dieser Arbeit werden zwei Motoren mit großen Leistung durch die statische Berechnung ausgewählt und die Anforderungen FST-04 wird erreicht. Deshalb muss man einen stärkeren Motortreiber (60A) auswählen. Die Auswahl des Motortreibers wird im Kapitel 6.2 erklärt. In diesem Kapitel geht es um die Drehzahlmessung des Getriebemotors. In diesem Experiment wird die Drehzahl des Getriebemotors im Leerlaufs-Zustand gemessen. Der Leerlaufstrom des neuen Getriebemotors wurde mit dem Wert 1,67A gemessen. Mit dem Motortreiber Monster beträgt der Motorstrom 14A. Außerdem ist die Programmierung für Monster nicht kompliziert. Aus diesem Grund wird der vorhandene Motortreiber Monster mit IC VNH3SP30-E von dem Hersteller SparkFun zur Drehzahlmessung benutzt.

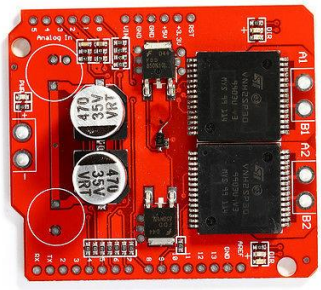


Abbildung 33: SparkFun Monster Moto Shield[8]

Die technische Daten des Motortreibers Monster:

Die maximale Betriebsspannung: 16V

Der maximale Motorstrom: 30A

Der typische Motorstrom: 14A

Der angesteuerte Motor: 2

Der Widerstand des MOSFETS: 19 mOhm

Maximale PWM-Frequenz: 20 kHz

Automatisches Abschalten bei Unter- und Überspannung sowie bei Überhitzung und messbarem Motorstrom

[8] Homepage: http://www.sparkfun.com

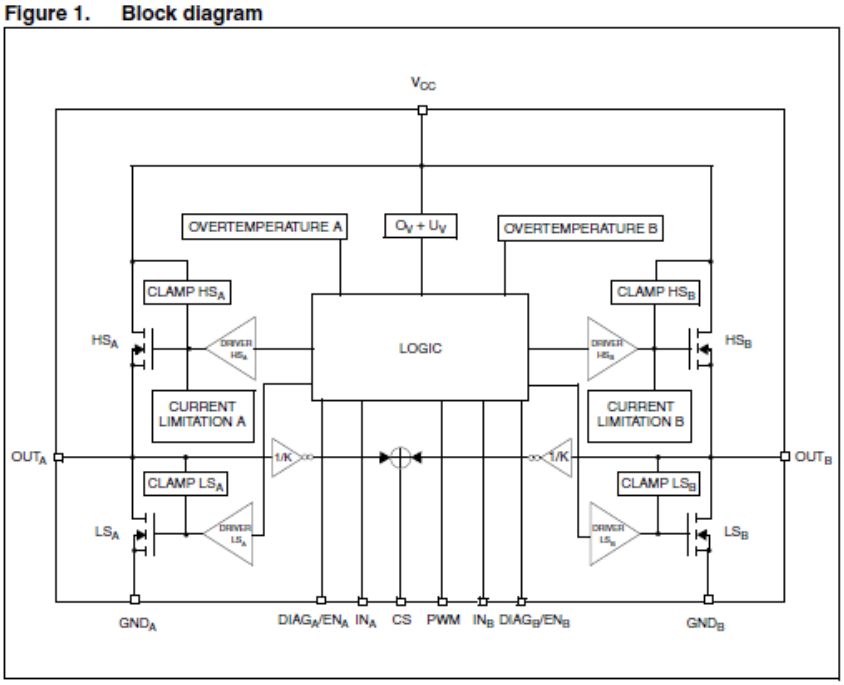


Abbildung 34: Blockdiagramm von Monster Moto Shield[9]

Im vorherigen Text ging es um die Funktion des Mikrocontrollers und des Motortreibers. Um die Parameter der GM zu identifizieren, muss eine Drehzahlmessung der GM durchgeführt werden. Danach kann man mit Matlab-Simulink die Datenerfassung dieser Messung zur Parameteridentifikation verwenden. Im Fahrsimulator wird dafür ein Inkrementalgeber vom Hersteller Avago Technologies benutzt.

4.5 Inkrementalgeber HEDM-5500

Ein Inkrementalgeber (auch Drehimpulsgeber und Drehgeber genannt) dient der Erfassung von Winkeländerung bei Achsen und Wellen. Ein Inkrementalgeber liefert eine bestimmte Anzahl an Impulsen pro Umdrehung, deshalb wird er häufig als Drehzahlmesser oder zur Positionsbestimmung in der Automatisierungstechnik verwendet. Inkrementalgeber können mit Schleifkontakt, optisch oder magnetisch arbeiten. Im gesamten System verwendet man einen Inkrementalgeber mit Optik für die Parameteridentifikation und einen Inkrementalgeber mit Schleifkontakt (Potenziometer genannt) für den Fahrsimulator.

In folgender Tabelle werden die Vorteile von optischen und „mechanischen“ Inkrementalgebers dargestellt.

[9]Vgl. Datenblatt für VNH2SP30-E http://cdn.sparkfun.com/datasheets/Dev/Arduino/Shields/10832.pdf Abruf 2016-11-15

Tabelle 7: Vergleich zwischen optischen und mechanischen Inkrementalgebers

Eigenschaften	**Optischer Inkrementalgeber**	**Potenziometer**
Bauform		
Drehrichtung und Drehzahl bestimmen	Möglich	Möglich
Ausgangssignal	digital	analog
begrenzter Drehwinkel	Ohne	360^0
Genauigkeit	sehr genau (10 bit)	niedrig
mit Arduino programmieren	Sehr kompliziert	Sehr einfach
Störung	Ja, mit fremden Licht	wenig
Preis	Nicht günstig (ca. 40€)	Günstig (5€)

Wegen der hohen angeforderten Genauigkeit bei der Drehzahlmessung muss man einen optischen Inkrementalgeber einsetzen. Um den Kostenaufwand des Fahrsimulators gering zu halten, kann man für den Betrieb des Fahrsimulators ein Potenziometer mit geringer Genauigkeit verwenden.

Nun betrachtet man die Grundlagen und die Funktion eines optischen Inkrementalgebers.

4.5.1 Funktion

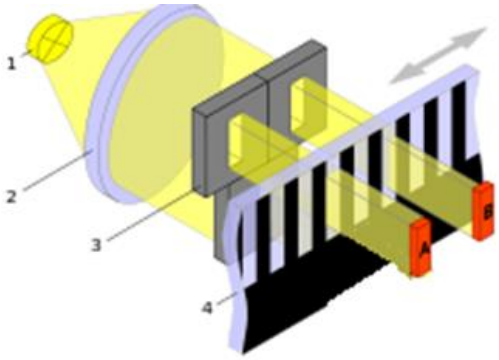


Abbildung 35: Funktion eines Inkrementalgebers[10]

Der grundsätzliche Inkrementalgeber besteht aus einer Infrarot-Leuchtdiode als Lichtquelle, zwei fotooptischen Bauelementen A und B (meist den Fototransistoren) und einer Abtastplatte (Glasscheibe). Die Glasscheibe trägt ein Gitter mit einer bestimmten Teilungsanzahl. Die Leuchtdiode erzeugt den Lichtstrahl, der über eine Abtastplatte zu zwei Fototransistoren geleitet wird. Die zwei Fototransistoren werden so angeordnet, dass der Fototransistor A gegenüber B um 90° phasenverschoben ist. Zwei Ausgangssignale von A und B werden mit dem Mikrocontroller verbunden, damit man die Drehrichtung und Drehzahl der GM bestimmen kann.

4.5.2 Signalauswertung

Wegen der Anordnung der zwei Fototransistoren erhält man zwei um 90° elektrisch phasenverschobene Signale (A und B) bei der Drehung der GM.

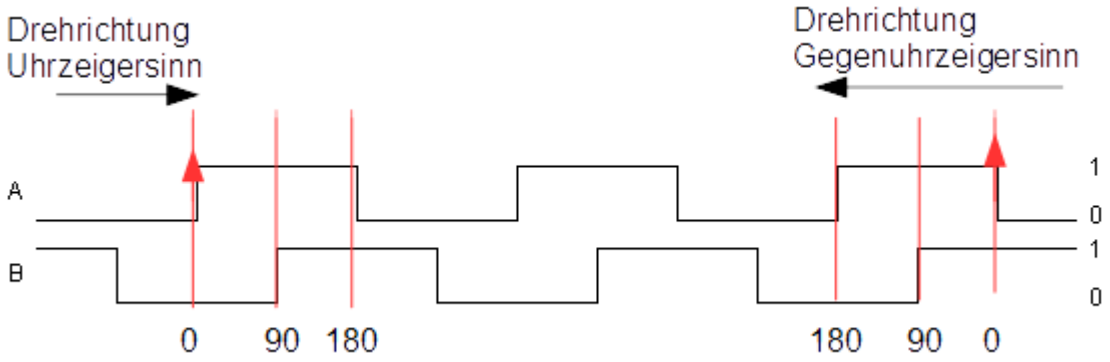


Abbildung 36: Signalauswertung der Drehrichtung im Uhrzeigersinn und Gegenuhrzeigersinn

In einer Drehrichtung ist das Signal A gegenüber dem Signal B um 90° voreilend und die umgekehrte Richtung ist um 90° nacheilend. In der oberen Abbildung sind die

[10] Vgl. Prof. Dr. Rasmus Rettig: Lehrskript Sensorik, S.48

Signale von links nach rechts bei der Drehrichtung im Uhrzeigersinn. Die Flanke des Signals A wechselt von 0 auf 1 und das Signal B bleibt 0. Bei der Drehrichtung im Gegenuhrzeigersinn wechselt die Flanke des Signals A auch von 0 auf 1, aber das Signal B bleibt 1. Wegen der Anordnung der zwei Fototransistoren kann man die Drehrichtung bestimmen. Außerdem werden die Impulse beider Signale A und B vom Mikrocontroller gezählt, damit die Drehgeschwindigkeit festgelegt werden kann. Die Genauigkeit der Drehzahlmessung hängt von der Teilung der Glasscheibe ab.

4.5.3 Art der Programmierung mit Encoder

Aus der Positionsregelung oder Drehzahlregelung und der Auflösung des Inkrementalgebers gibt es mehrere Varianten der Zählung des Impulses mit dem Mikrocontroller (Arduino UNO) über die Timer.

- **Timer mit Funktion Input Capture**

Der Arduino Uno mit dem Mikrocontroller ATmega328 hat zwei 8-Bit Timer (Timer 0 und Timer 2) und einen 16-Bit (Timer 1). Ein Timer ist einfach ein bestimmtes Register im Mikrocontroller. Der Timer wird auf- oder abwärtsgezählt, wenn ein Ereignis geschehen ist. Das Signal A oder B wird mit dem Input Capture Pin (ICP, Arduino Uno – Pin 8) verbunden. Bei einem Flankenwechsel von Signal A oder B an einen ICP Pin speichert der Timer seinen aktuellen Wert im ICR (Input Capture Register). Durch die Differenz aus zwei aufeinanderfolgenden Werten im ICR kann man die Frequenz vom Signal A oder B bestimmen, um die Drehzahl zu berechnen. Außerdem kann man einen zusätzlichen Timer zur Drehrichtungsbestimmung verwenden.

Diese Methode misst die Zeit mit hoher Genauigkeit. Ein Nachteil ist aber die komplizierte Programmierung des Timers.

- **Counter (Zähler)**

Der Arduino Uno hat auch drei Zähler und zwar zwei 8-Bit Zähler (Zähler 0 und 2) und einen 16-Bit Zähler (Zähler 1). Zwei Ausgangssignale A und B werden mit zwei Zählern vom Arduino Uno verbunden. Dadurch kann man die Impulse von beiden Signalen A und B in der bestimmten Zeit zählen und die Geschwindigkeit der GM wird dann berechnet. Diese Methode dient nur für die Bestimmung der Drehzahl, die Drehrichtung kann nicht bestimmt werden.

- **Externate Interrupt**

Diese Methode kombiniert die zwei Methoden mit Timer und Zähler. Sowohl die Drehzahl als auch die Drehrichtung der GM kann damit bestimmt werden. Außerdem ist die Programmierung vom Arduino damit vereinfacht und gleichzeitig die Messgenauigkeit gestiegen.

Bei bestimmten Ereignissen in Prozessoren wird ein sogenanntes Interrupt ausgelöst. Deshalb werden zwei Ausgangssignale A und B von dem Inkrementalgeber an zwei Interrupt-Pins vom Arduino angeschlossen. Der Arduino Uno hat zwei Interrupt-Pins und zwar Pin 2 (INT0) und Pin 3(INT1). Die Interrupts sind bei einer fallenden (high nach low) oder steigenden Flanke (low nach high) aktiviert. Danach wird eine Funktion, die die Impulse von dem Signal A oder B in der bestimmten Zeit zählt, aufgerufen. Der

Zähler wird inkrementiert oder dekrementiert, wenn der Motor sich im Uhrzeigersinn oder Gegenuhrzeigersinn dreht. In dieser Methode muss man die Taktfrequenz vom Arduino beachten. Die maximale Drehzahl der GM beträgt 100 RPM (oder 10 rad/s) im Leerlauf und die Impulse pro Umdrehung des Inkrementalgeber betragen 1000 CPR. Die maximale Frequenz des Signals A oder B wird berechnet:

$$f = 100\,\frac{rad}{s} * 1000\,\frac{Impulse}{rad} = 10000\,Hz = 10\,kHz$$

Die Taktfrequenz des Arduino Uno muss unbedingt größer sein als der Wert f=10kHz. Laut Datenblatt des Arduino beträgt seine Taktfrequenz 16MHz und das bedeutet, dass man die Methode mit Encoder (1000 CPR) zur Drehzahlmessung verwenden kann.

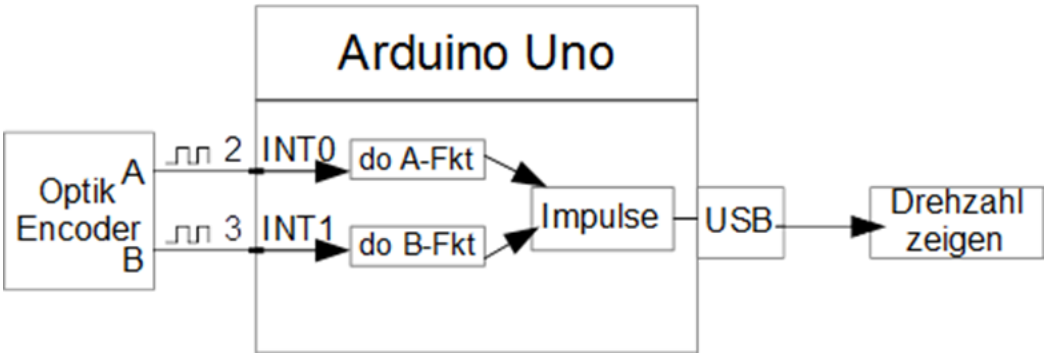


Abbildung 37: Modell der Drehzahlmessung

In dem folgenden Aktivitätsdiagramm wird ein Algorithmus der Drehzahlmessung gezeigt:

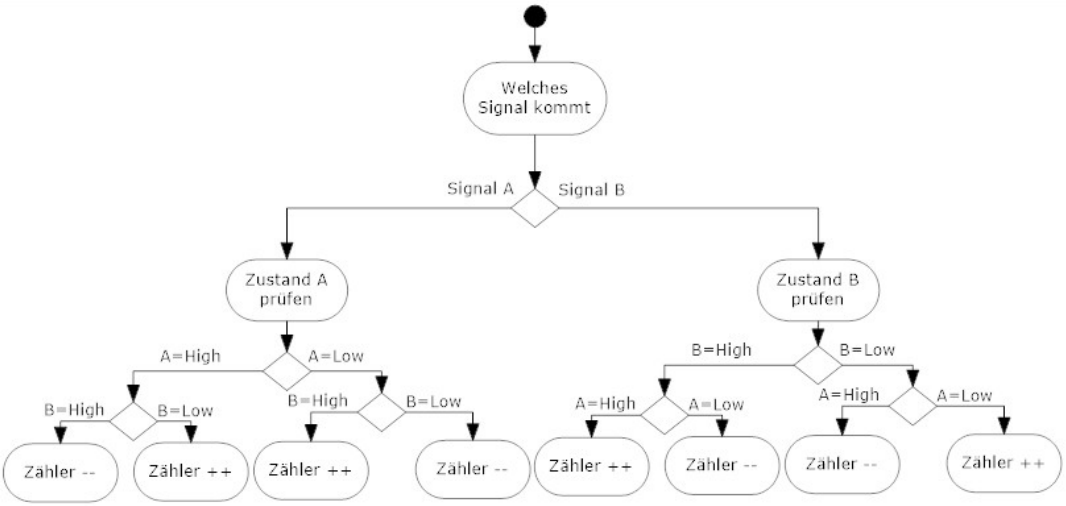


Abbildung 38: Algorithmus der Drehzahlmessung

In dieser Arbeit wird ein Inkrementalgeber mit 1000 CPR (Impulse pro Umdrehung, engl. Count per Revolution) genutzt und es werden insgesamt 4 Zustände der Flanke von beiden Signalen A und B genutzt, um die Impulse zu zählen. Deshalb wird der Drehwinkel mit folgender Formel berechnet:

$$Drehwinkel = \frac{2\pi * Anzahl\ der\ Impulse}{1000\ CPR * 4}$$

4.5.4 Ergebnis der Messung

Das Experiment wurde 3 Sekunden lang durchgeführt. Dies bedeutet, dass der Getriebemotor sich 3 Sekunden lang dreht, und nach jeder Sekunde die Drehrichtung verändert. Je 10 Millisekunden werden die Impulse der Signale A oder B gezählt und die Drehzahl per Arduino berechnet. Diese aufgenommenen Ergebnisse werden verwendet, um die Parameter des Getriebemotors mit Matlab/Simulink zu identifizieren.

Damit die Genauigkeit der Drehzahlmessung erhöht wird, wurde das Experiment fünfmal durchgeführt. Danach wurden in Matlab/Simulink die Ergebnisse der Drehzahlmessung zur Parameteridentifikation der GM genutzt.

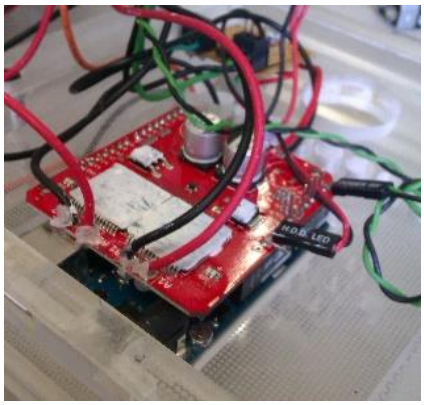


Abbildung 39: Verbindung zwischen Arduino Uno und Monster

Die folgende Abbildung zeigt den Vergleich zwischen dem Simulations-Modell und der Messung in Echtzeit:

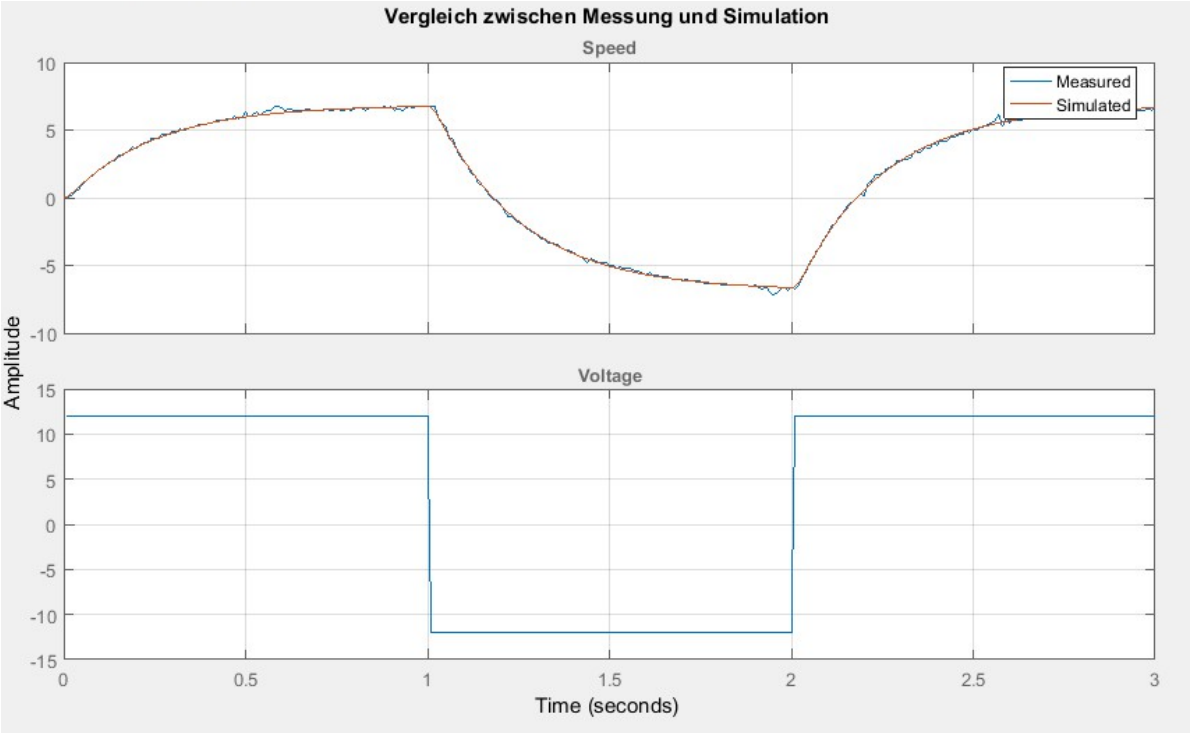


Abbildung 40: Vergleich zwischen Messung und Simulation mit Hilfe der „System Identifikation Toolbox“

Nach der Parameter-Abschätzung in Matlab/Simulink sind beide Grafiken von der Simulation und der Messung fast identisch. Jetzt kann man alle Parameter der GM bestimmen.

Tabelle 8: Messdatenerfassung der Drehzahlmessung

Parameter	Einheit	1. Versuch	2. Versuch	3. Versuch	4. Versuch	5. Versuch	Mittel-wert
Ankerwiderstand (RA)	Ω	0,22253	0,21855	0,21740	0,22205	0,21846	**0,21980**
Ankerinduktivität (LA)	H	0,00261	0,00276	0,00278	0,00236	0,00261	**0,00262**
Maschinenkonstante (K)		0,01025	0,01053	0,01015	0,01026	0,01012	**0,01026**
Trägheitsmoment (J)	$kg.m^2$	0,01925	0,01958	0,01882	0,01943	0,01921	**0,01926**
Viskose Reibung (B)		0,08011	0,08410	0,08131	0,08001	0,08028	**0,08116**

Die berechneten Parameter des benutzten Getriebemotors:

Parameter	Wert
Ankerwiderstand (RA)	**0,219798**
Ankerinduktivität (LA)	**0,002622**
Maschinenkonstante (K)	**0,010262**
Trägheitsmoment (J)	**0,019259**
Viskose Reibung (B)	**0,081164**

Die bestimmten Parameter werden im nächsten Kapitel verwendet, um die Parameter vom PID-Regler zu identifizieren.

5 Positionsregelung der GM

5.1 Grundlagen der Regelungstechnik

Ein vorhandenes Fahrsimulator-System des JBWs wurde während meines Praktikums ohne professionelle Regelungstechnik verwendet. Obwohl dies System funktioniert hat, war aber die Reaktion von beiden Motoren langsam und es gab wenig Genauigkeit. In meiner Arbeit entsteht ein Fahrsimulator mit einer kompetenten Regelungstechnik, indem ein PID Regler eingesetzt wird. Damit kann das System eine hohe Genauigkeit und schnelle Reaktion erreichen. Nun betrachtet man die Grundlagen eines Regelkreises.

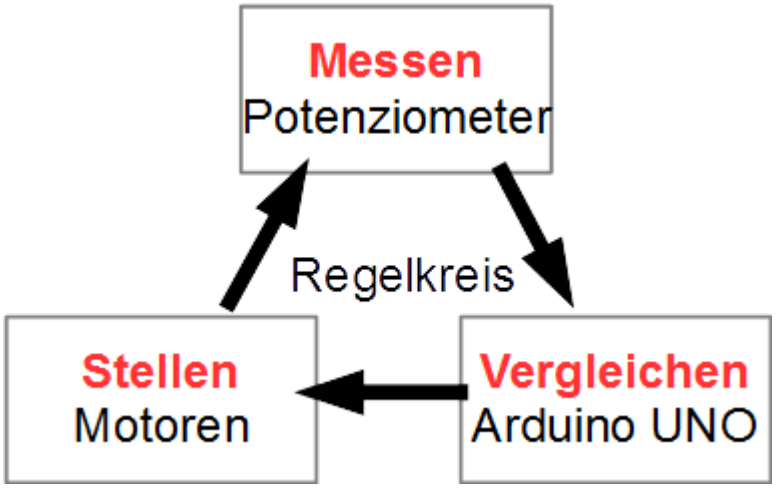


Abbildung 41: Diagramm eines Regelkreises[11]

Prinzip der Regelung

- Messen

Die Regelgröße (aktuelle Position) wird mittels Potenziometer gemessen.

- Vergleichen

Der Wert der Regelgröße wird mit dem Sollwert (eingestellte Position aus dem Game) durch den Arduino UNO verglichen. Die Differenz wird Regelabweichung genannt.

- Stellen

Nach der Berechnung vom Arduino UNO wird die Stellgröße (gewünschte Position) für zwei Motoren bestimmt.

[11] Vgl. Unger Jochem: Einführung in die Regelungstechnik, S.15

In der nächsten Abbildung wird das Blockschaltbild eines einfachen Standardregelkreises gezeigt. Es besteht aus zwei Hauptteilen Regler und Regelstrecke.

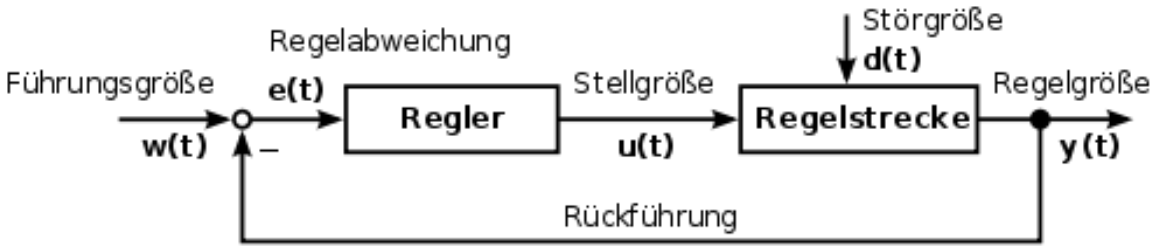


Abbildung 42: Blockdiagramm des Regelkreises

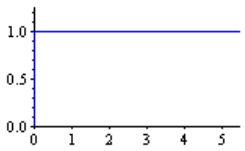
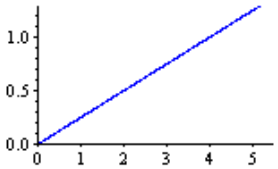
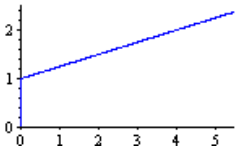
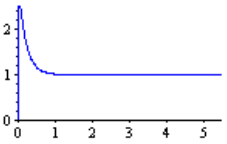
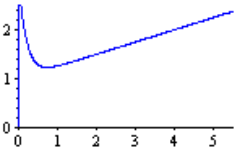
Die Aufgabe des Reglers ist die Minimierung der Differenz zwischen Soll- und Istwert der Regelgröße. Der Regler stellt einen Ausgleich her. Allgemein wird der Regler nach stetigem und unstetigem Verhalten unterschieden:

Tabelle 9: Vergleich zwischen stetigen und unstetigen Regler

Stetige Regler	Unstetige Regler
- Generiert aus der Regeldifferenz einen kontinuierlichen Stellwert	- Generiert aus der Regeldifferenz einen diskreten Stellwert.
Beispiel: P-, I-, PD-, PI-, PID-Regler	- Beispiel: Zwei-, Drei-, Mehrpunktregler

Im Fahrsimulator wird die aktuelle Position der GM durch ein analoges Potenziometer ermittelt. Der Mikrocontroller Arduino Uno nutzt diese Werte, um durch einen Regler-Algorithmus die eingestellte Position festzulegen. Daher wird ein stetiger Regler verwendet. In folgender Tabelle handelt es sich um die verschiedenen Arten des stetigen Reglers mit dem Eingangswert e(t) und dem Ausgangswert u(t):

Tabelle 10: lineare Regler

Typ	Verhalten	Formel	Symbol
P	proportional	$u(t) = K_p * e(t)$	
I	integral	$u(t) = K_I \int_0^t e(\tau)d\tau$	
PI	proportional und integral	$u(t) = K_p * e(t) + K_I \int_0^t e(\tau)d\tau$	
PD	proportional und differential	$u(t) = K_p * e(t) + K_D \frac{de(t)}{dt}$	
PID	proportional, integral und differential	$u(t) = K_p * e(t) + K_I \int_0^t e(\tau)d\tau + K_D \frac{de(t)}{dt}$	

Jede Art des Reglers hat sowohl Vorteile als auch Nachteile und die Auswahl des Reglers ist abhängig von den Anforderungen des Systems. In folgender Abbildung werden die Eigenschaften von allen Reglern mit PT2-Glied (Verzögerungsglied mit 2. Ordnung) als Regelstrecke verglichen.

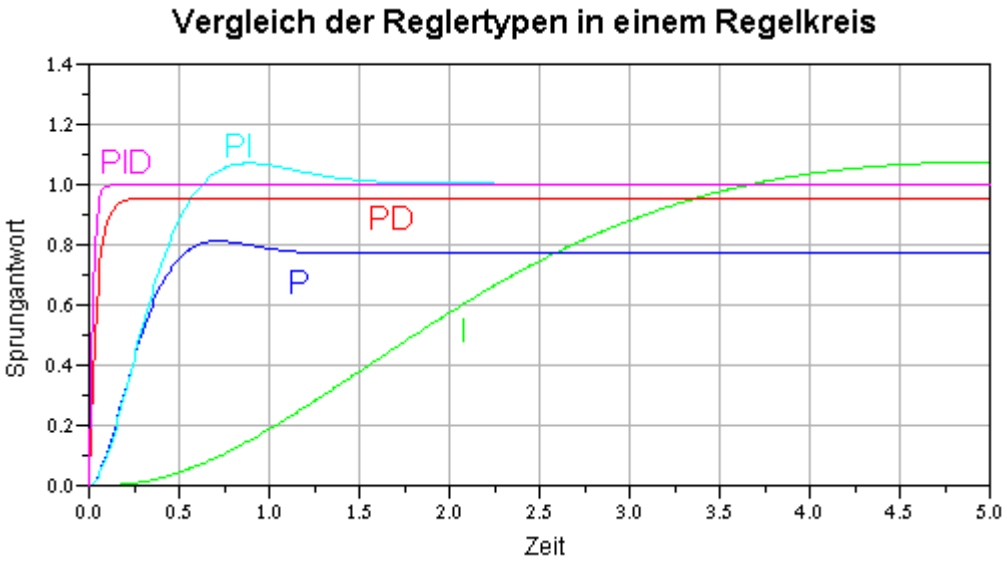


Abbildung 43: Vergleich der Reglertypen in einem Regelkreis[12]

In der oberen Abbildung kann man deutlich erkennen, dass die Regler mit I-Anteil (PI und PID) den Endwert 1 oder kleinere Regelabweichungen erreichen können. Die Regler mit D-Anteil (PID und PD) reagieren am schnellsten und die Regler ohne D-Anteil besonders mit P-Anteil (PI und P) sind mittelschnell. Durch die Anforderungen des Systems müssen die Getriebemotoren schnell ausgeregelt werden und es gibt keine bleibende Regelabweichung, deshalb wird ein PID-Regler ausgewählt. Nun diskutiert man die grundsätzliche Funktion und die Bestimmung der Parameter eines PID-Reglers.

5.2 Entwurf eines PID-Reglers

Um einen gut funktionierenden Regelkreis aufzubauen, muss das dynamische Verhalten der Regelstrecke ermittelt werden. Es heißt das Verhalten des Getriebemotors.
Aus der Gleichung (3.11) im Kapitel 3 kann man die Übertragungsfunktion der GM im Laplace-Bereich bestimmen.

$$G(s) = \frac{W(s)}{Ua(s)} = \frac{K}{(Js+B)(L_A s+R_A)+K^2} \qquad \text{(Gl 5.1)}$$

Wegen der Positionsregelung interessiert man sich für das Verhältnis des Drehwinkels θ und der Ankerspannung Ua. Deshalb muss die obige Übertragungsfunktion weitergerechnet werden.

[12] Vgl. Regelungstechnik http://rn-wissen.de/wiki/index.php/Regelungstechnik Abruf: 2016-11-15

Es gibt die Beziehung zwischen dem Drehwinkel θ und der Winkelgeschwindigkeit W:
$W = \frac{d\theta}{dt}\ oder\ \int dW * dt = \theta$

Die Gleichung (5.1) wird integriert und nochmal im Laplace-Bereich transformatiert. Nun wir die Übertragungsfunktion zwischen dem Drehwinkel θ und der Ankerspannung Ua ermittelt.

$$G(s) = \frac{\theta(s)}{Ua(s)} = \frac{K}{(Js+B)(L_A s+R_A)+K^2} * \frac{1}{s} \quad \left[\frac{rad}{V}\right]$$

Alle Werte von den Parametern der GM werden in die Übertragungsfunktion eingesetzt:

$$G(s) = \frac{\theta(s)}{Ua(s)} = \frac{0{,}01026}{5{,}05*10^{-5}*s^3+4{,}45*10^{-3}*s^2+0{,}0179*s}$$

Durch die obige Übertragungsfunktion kann man einfach erkennen, dass die Regelstrecke ein PT3 – Glied (Verzögerungsglied mit 3. Ordnung) besitzt. Jetzt wird ein Blockdiagramm der Positionsregelung der GM erstellt:

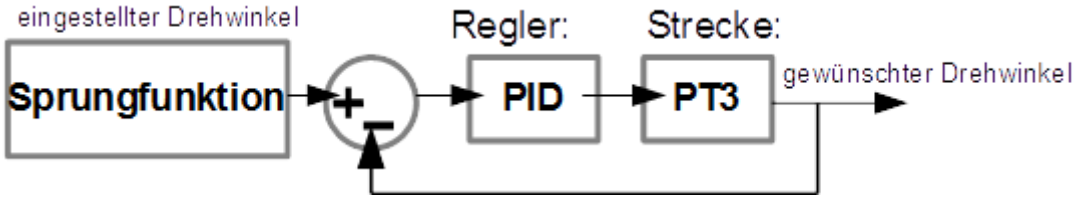


Abbildung 44: Das Blockdiagramm des geschlossenen Regelkreises

Um die PID – Reglerparameter zu dimensionieren, wurden viele Methoden entwickelt. In dieser Arbeit betrachtet man nur zwei wichtige Methoden, die empirische Dimensionierung und die Dimensionierung nach Einstellregeln. Die erste Methode wird ohne Kenntnis der Regelstreckenparameter verwendet und die PID – Parameter werden einfach nach praktischen Erfahrungswerten vorgewählt und variiert. Deshalb ist diese Methode geeignet, um einfache System zu dimensionieren. Im Gegensatz zu der Dimensionierung nach Einstellregeln müssen die Art der Regelstrecke und ihre Streckenparameter bekannt sein. Im vorherigen Abschnitt wurden die Art der Strecke als ein PT3 - Glied und alle Parameter schon bestimmt. Deshalb nutzt man diese Methode, um das System zu dimensionieren.

Um die PID – Reglerparameter zu bestimmen, gibt es zwei bekannten Einstellregeln: Die von Ziegler/Nichols und von Chien/Hrones/Reswick. Die zweite Methode beruht auf der Aufnahme der Sprungantwort der Regelstrecke, dies bedeutet, dass man die Sprungantwort der Stecke bestimmen muss. Durch diese Sprungantwort werden die Verzugszeit Tu und die Ausgleichszeit Tg ermittelt. Bei der ersten Methode braucht man die Sprungantwort nicht, aus diesem Grund werden die Einstellregeln nach Ziegler/Nichols in dieser Arbeit verwendet. Im folgenden Abschnitt wird das Verfahren dieser Methode erklärt.

5.2.1 Einstellung nach der Ziegler/Nichols – Methode

Bei der Methode nach Ziegler/Nichols werden die Reglerparameter verstellt, damit das System die Stabilitätsgrenze erreichen kann und der Regelkreis fängt an zu schwingen. Damit wird die Periode der Schwingung ermittelt und mit der „Ziegler/Nichols Tabelle “ kann man alle Parameter bestimmen.

Die Vorgehensweise von dieser Methode ist folgende:

1. Der Regler wird zuerst als P-Regler betrieben, d.h. ki = kd = 0 und Anfangswert von P-Regler ist gleich (Kp = 0)
2. Kp wird langsam erhöht, bis der geschlossene Regelkreis die Stabilitätsgrenze erreicht und eine Dauerschwingung auftritt.
3. Die Verstärkung Kp wird zurückgenommen und bezeichnet als K_{Pkrit}
4. Die Periodendauer T_{Ktit} der Schwingung wird gemessen
5. Die Reglereinstellwerte können dann aus „Ziegler/Nichols Tabelle “ entnommen werden:

Tabelle 11: Die Bestimmung von Kp, Tn und Tv nach Ziegler/Nichols

Regler Typ	K_P	T_n	T_v
P	$0{,}5 * K_{Pkrit}$	-	-
PI	$0{,}45 * K_{Pkrit}$	$0{,}83 * T_{Krit}$	-
PID	$0{,}6 * K_{Pkrit}$	$0{,}5 * T_{Krit}$	$0{,}125 * T_{Krit}$

Mit der Nachstellzeit $T_N = \frac{K_P}{K_I}$ und der Vorhaltezeit $T_V = \frac{K_D}{K_P}$

Nun betrachtet man den Regelkreis mit dem Führungsverhalten. Das Führungsverhalten beschreibt die Auswirkung von Führungsgröße w(t) auf die Regelgröße x(t).

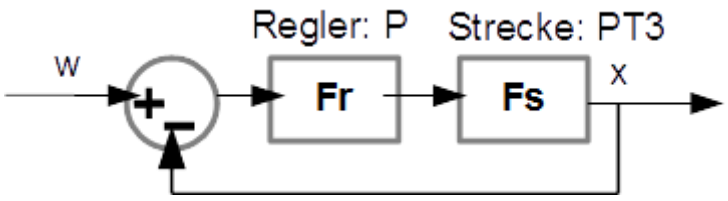


Abbildung 45: PT3-Regelstrecke mit P-Regler

Für diesen geschlossenen Regelkreis wird die Führungsübertragungsfunktion ermittelt:

$$F_w = \frac{F_R * F_S}{1+(F_R * F_S)}$$

Die Stabilität des geschlossenen Regelkreises ist abhängig von der genauen Lage der Pole der Übertragungsfunktion. Wenn alle Pole links der imaginären Achse liegen, ist das System stabil. Aus diesem Grund betrachtet man nur die Nullstellen des Nenners, d.h. $1 + (F_R * F_S) = 0$. Der P – Regler beträgt eine Verstärkung Kp und diese Verstärkung muss bestimmt werden. Nun werden die Verstärkung Kp und die Übertragungsfunktion der Strecke (Gleichung …) in der oberen Gleichung eingesetzt.

$$1 + \left(\frac{K_P * K}{((Js+B)(L_A s+R_A)+K^2)*s}\right) = 0$$

Und das Nennerpolynom wird bestimmt:

$$L_A J s^3 + (L_A B + R_A J)s^2 + (R_A B + K^2)s + KK_P = 0$$

Die Bestimmung der Verstärkung Kp oder die Untersuchung des Regelkreises beruht auf dem Hurwitz – Kriterium.

Das reelle Polynom für den 3. Grad nach dem Hurwitz – Kriterium beträgt:

$$a(s) = a_3 s^3 + a_2 s^2 + a_1 s + a_0$$

Dann wird die Hurwitzdeterminante gebildet:

$$H = \begin{vmatrix} a_2 & a_0 & 0 \\ a_3 & a_1 & 0 \\ 0 & a_2 & a_0 \end{vmatrix} = \begin{vmatrix} (L_A B + R_A J) & KK_P & 0 \\ L_A J & (R_A B + K^2) & 0 \\ 0 & (L_A B + R_A J) & KK_P \end{vmatrix}$$

Der geschlossene Regelkreis ist stabil, wenn alle Unterdeterminanten größer als Null sind, d.h.

$$H_1 = |a_2| = |(L_A B + R_A J)| \geq 0$$

$$H_2 = \begin{vmatrix} a_2 & a_0 \\ a_3 & a_1 \end{vmatrix} = (L_A B + R_A J) * (R_A B + K^2) - (L_A J * KK_P) \geq 0$$

$$H_3 = a_3 * H_2 = KK_P * [(L_A B + R_A J) * (R_A B + K^2) - (L_A J * KK_P)] \geq 0$$

Die Motorparameter betragen:

$$R_A = 0{,}219788\ Ohm$$

$$L_A = 0{,}002622\ H$$

$$K = 0{,}010262$$

$J = 0{,}019259\ \ kgm^2$

$B = 0{,}081164$

Und die Berechnung aller Unterdeterminanten:

$H_1 = 0{,}0044 > 0$

$H_2 = 7{,}9782 * 10^{-5} - 5{,}182 * 10^{-7} * K_P \geq 0 \rightarrow K_P \leq 153{,}9589$

$H_3 = 0{,}010262 * K_P * [7{,}9782 * 10^{-5} - 5{,}182 * 10^{-7} * K_P] \geq 0 \rightarrow K_P \geq 0$

Durch die Bedingungen von H_2 und H_3 wird die Verstärkung K_P ermittelt:

$\mathbf{0 \leq K_P \leq 154}$

Wenn die Verstärkung K_P den Maximalwert 154 erreicht, liegt das System an der Stabilitätsgrenze. Es bedeutet, dass die Verstärkung K_P mit dem Wert 154 genommen wird. Der geschlossene Regelkreis führt die Dauerschwingung aus. Nun wird das System mit Matlab/Simulink simuliert.

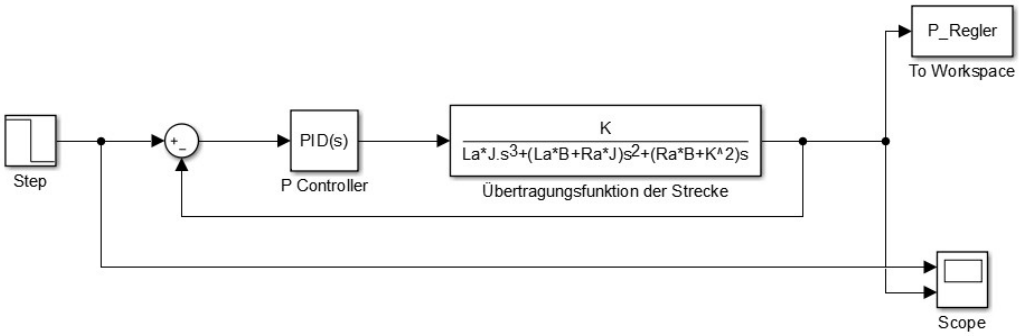


Abbildung 46: Das Blockdiagramm des Systems in Matlab/Simulink

Nach dem Simulieren bekommt man die Grafik des Systems:

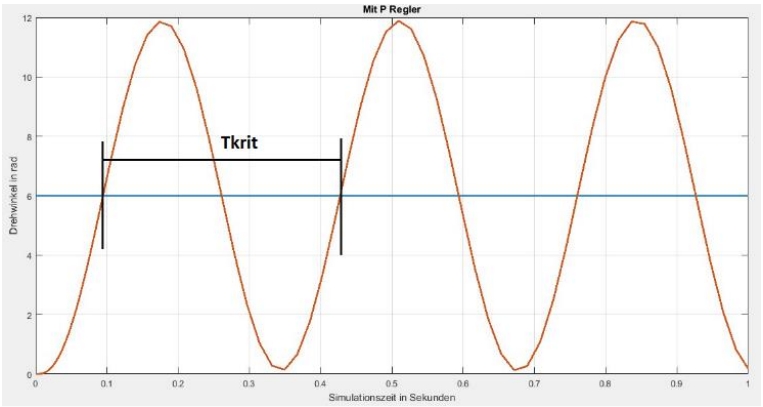


Abbildung 47: Die Sprungantwort mit P-Regler (Kpkrit=154)

Die Periodendauer wird gemessen: $T_{Krit} = 0{,}334\ s$

Durch die „Ziegler/Nichols Tabelle“ werden die Vorhaltezeit und Nachstellzeit berechnet:

$K_P = 0{,}6 * K_{Pkrit} = 0{,}6 * 154 = 92$

$T_N = 0{,}5 * T_{Krit} = 0{,}5 * 0{,}334s = 0{,}1673\ s$

$T_V = 0{,}125 * T_{Krit} = 0{,}125 * 0{,}334s = 0{,}0418\ s$

Nun kann man alle PID – Reglerparameter bestimmen:

$K_P = 0{,}6 * K_{Pkrit} = 0{,}6 * 154 = 92$

$K_I = \frac{K_P}{T_N} = \frac{92}{0{,}1673} = 552$

$K_D = K_P * T_V = 92 * 0{,}0418 = 4$

Alle PID-Reglerparameter werden eingesetzt und das System wird nochmal simuliert.

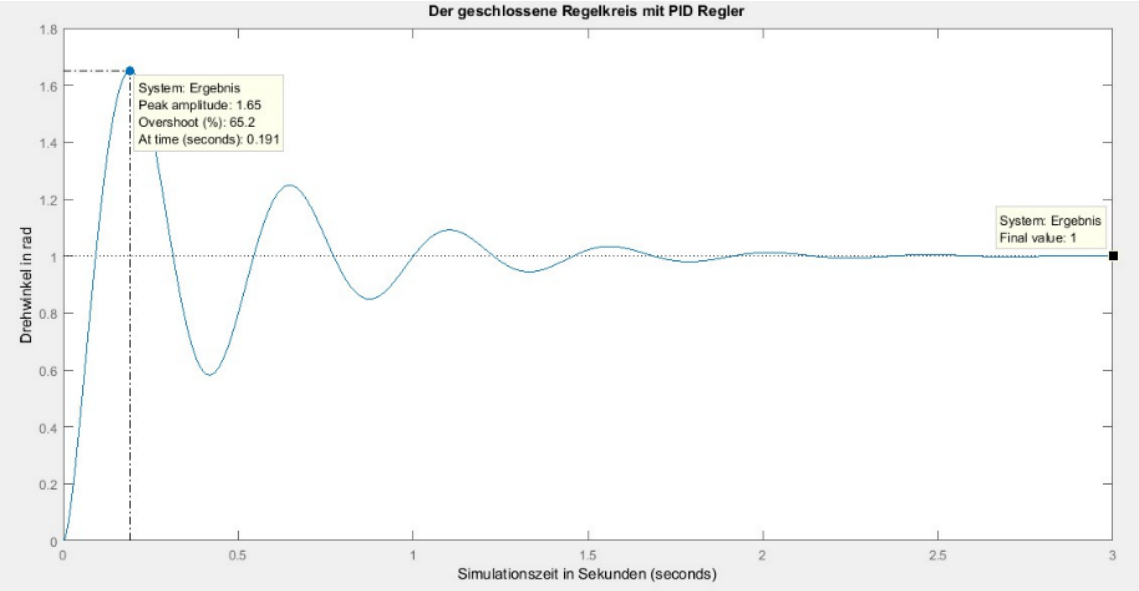


Abbildung 48: Die Sprungantwort des geschlossenen Regelkreises mit hoher Überschwingungsweite

Durch die obere Grafik der Sprungantwort von dem geschlossenen Regelkreis kann man erkennen, dass der Endwert den stationären Wert 1 erreicht. Aber die Überschwingungsweite (engl. Overshoot), die die größte vorübergehende Sollwertabweichung nach einem Sollwertsprung (z.B. 1 rad) ist, hat einen hohen Wert, d.h. 65,2 %. Im Industriebereich versucht man, dass dieser Wert auf weniger als 10 Prozent reduziert wird. Um dieses Ziel zu erreichen, wird das automatische Tunen von PID-Reglern mit Matlab/Simulink durchgeführt.

5.2.2 PID Einstellung in Matlab/Simulink

Durch die Verwendung eines automatischen Werkzeugs PID-Tuner in Matlab/Simulink kann man die optimalen Werte von drei Regelgliedern (Proportionalglied, Integralglied und Differenzialglied) des PID-Reglers ermitteln. In folgender Abbildung wird das Werkzeug PID – Tuner[13] angezeigt

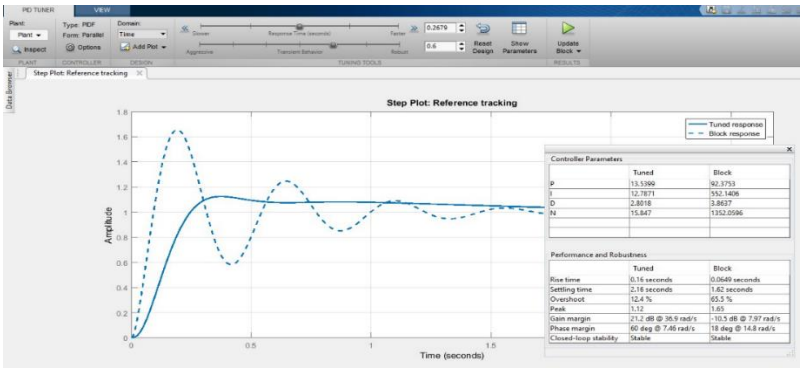


Abbildung 49: Die Sprungantwort nach der Verwendung des Werkzeugs PID-Tuner in Matlab/Simulink

Durch die Veränderung von zwei Parametern „Response Time“ und „Transient Bahavior“ kann man den PID-Regler mit den richtigen Eigenschaften einstellen.

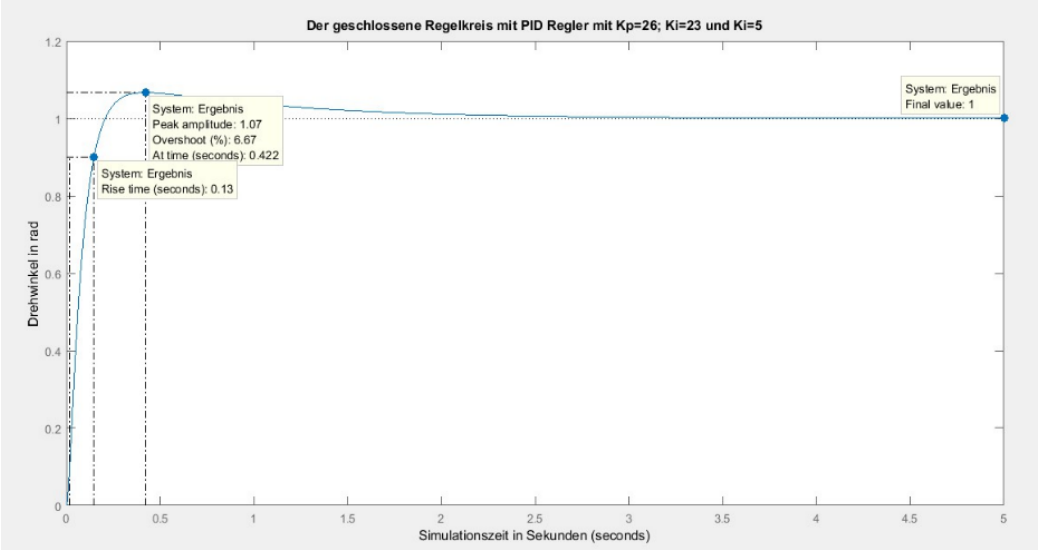


Abbildung 50: Die Sprungantwort des geschlossenen Regelkreises mit Kp=26, Ki=23 und Kd=5

[13]Vgl. PID Controller Tuning in Simulink https://de.mathworks.com/help/slcontrol/gs/automated-tuning-of-simulink-pid-controller-block.html Abruf: 2016-11-15

Mit dem eingestellten PID-Regler (**Kp=26, Ki=23 und Kd=5**) besitzt der geschlossene Regelkreis die richtigen Eigenschaften:

- Überschwingungsweite (engl. Overshoot) = 6,67 %
- Endwert (engl. Final value) = 1 (d.h. Ohne Abweichung zwischen Sollwert und Istwert)
- Anstiegszeit (engl. rise time) = 0,13 s (d.h. nach 0,13 Sekunden erreicht das System 90 % des Sollwertes).

In folgendem Kapitel nutzt der Mikrocontroller Arduino Uno diesen PID-Regler mit den Werten **Kp=26, Ki=23 und Kd=5,** um durch einen Regler-Algorithmus die eingestellte Position festzulegen.

6 Fahrsimulator

6.1 Beschreibung des Systems

Zuerst wird ein Systemkontext von dem gesamten System „Fahrsimulator“ erstellt. Dadurch kann man alle Komponenten des Systems festlegen und die Beziehungen zwischen diesen Komponenten herstellen.

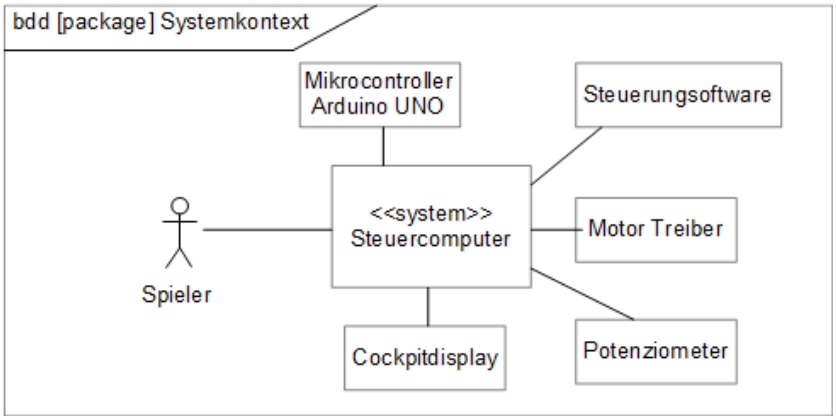


Abbildung 51: Systemkontext des Fahrsimulators

Der Steuercomputer spielt eine wichtige Rolle bei dem Fahrsimulator. Dieser Steuercomputer und die Steuerungssoftware (Motion Simulator Software) erzeugen eine Verbindung zwischen den Rennspielen und dem Steuergerät. Zuerst werden alle Werte der Rennspiele (z.B. Drehzahl, Geschwindigkeit, Bewegungsrichtung, usw.) aufgenommen, und dann werden diese Werte durch eine serielle Schnittstelle zu dem Steuergerät gesendet. Das Steuergerät erzeugt die Bewegung für den beweglichen Stuhl.

6.2 Motor Treiber Sabertooth

Um die Anforderung FST 04 (Spieler mit einem Gewicht von 80 kg) erreichen zu können, werden zwei Getriebemotoren mit großer Leistung verwendet. Deshalb braucht der Motor großen Motorstrom und einen Motor-Treiber mit großer Stromversorgung. In dieser Arbeit wird ein Motor-Treiber Sabertooth 2x60 vom Hersteller Dimension Engineering genutzt. Im Vergleich zu dem im Kapitel 4.4.4 vorgestellten Motor-Treiber Monster vom Hersteller SparkFun hat dieser Motor-Treiber Sabertooth mehrere Vorteile:

Tabelle 12: Vergleich zwischen Sabertooth und Monster

Sabertooth 2x60	• SparkFun Monster
	
• Kontinuierlich fließender Strom: 60 A pro Channel. • Versorgungsspannung: 6-30V • Überstrom- und Überhitzungsschutz • Mehrere Eingangsmodule bearbeiten, z.B. Analog, simplified serial und R/C • Hoher Preis: 160€	• Kontinuierlich fließender Strom: 14A pro Channel • Versorgungsspannung: 6-16V • Überhitzungsschutz • Nur Analogsignal bearbeiten • Günstiger Preis: 70€

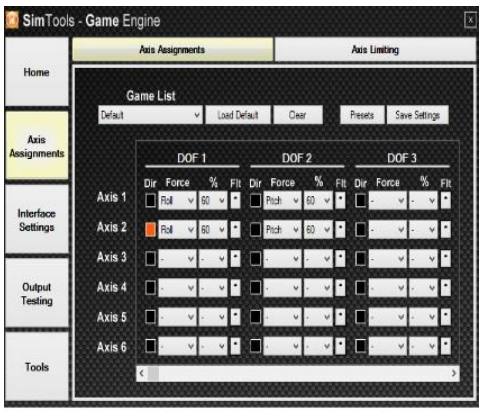
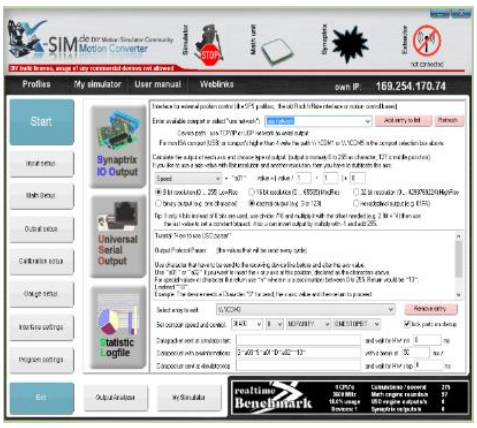
Durch die vorherige Tabelle kann man erkennen, dass der Motortreiber von Sabertooth den Motor mit 60A versorgt, und der von Hersteller Monster nur mit 14A. Deshalb wird man den Motortreiber von Sabertooth 2x60 verwenden, um zwei Getriebemotoren WG88BD88-1 anzusteuern.

6.3 Motion Simulator Software

Um die Bewegung des Fahrsimulators durchzuführen, kann man zwei unterschiedliche Motion Simulator Programme von zwei Entwicklern X-Sim [14]und SimTool[15] nutzen. Jede Software hat Vor- und Nachteile und ist abhängig vom Anwender. In diesem Projekt wurde SimTools für die Ansteuerung der Motoren (Bewegung) und für die Ansteuerung des Cockpits benutzt.

[14] Homepage http://www.x-sim.de
[15] Homepage http://www.xsimulator.net

Tabelle 13: Motion Simulator Software SimTool und X-Sim

SimTools	X-Sim
	
• kostenfreie Software • unterstützt mehrere Rennspiele (ca. 45 Rennspiele) • Parameter von Gamedaten nicht ablesbar • mit wenigen Hardware-Steuerungen (Arduino, Sabertooth, Monster) kompatibel • maximal 6 parallele Schnittstellen gleichzeitig zu nutzen	• Zwei Versionen: Kostenfreie Version (mit Internet vorhanden) und offline Version (60 €) • Unterstützt ca. 50 Rennspiele • Alle Parameter auszulesen und Diagramm erstellbar • Mit viel verschiedener Hardware (Arduino, Sabertooth, Monster, Hydraulische System...) kompatibel • Max. 2 Schnittstellen

Im nächsten Abschnitt geht es um die Kommunikation zwischen den einzelnen Komponenten des Fahrsimulators.

6.4 Sequenzdiagramm

Mit dem Sequenzdiagramm wird beschrieben, welche Komponenten beteiligt sind, und in welcher zeitlichen Reihenfolge die Informationsaustausche stattfinden. In folgender Abbildung wird das Sequenzdiagramm des Fahrsimulators gezeigt:

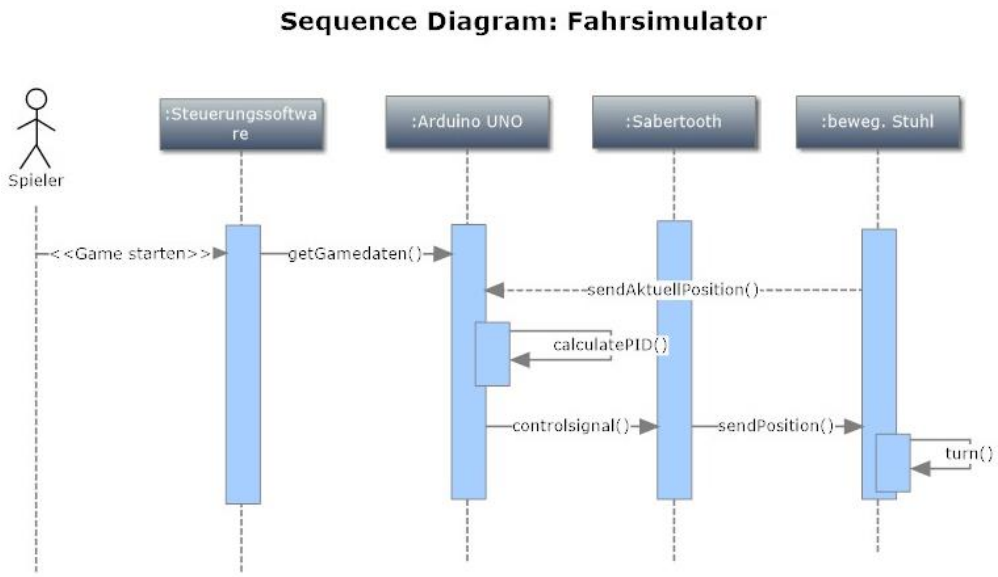


Abbildung 52: Sequenzdiagramm des Fahrsimulators

Wenn der Spieler ein Rennspiel startet, werden alle Gamedaten (z.B. Geschwindigkeit, Drehrichtung, Drehzahl, usw.) durch die Steuerungssoftware (SimTool oder X-Sim) ausgelesen. Danach werden diese Daten zu dem Mikrocontroller Arduino UNO gesendet. Der Mikrocontroller wartet auf das Signal (aktuelle Position des Getriebemotors) des Potenziometers und nach dem Empfang des Signals wird eine PID – Funktion durch den Arduino UNO implementiert. Diese Funktion macht einen Vergleich zwischen der aktuellen Position des Potenziometers mit der gewünschten Position im Datengame. Nach der PID-Berechnung sendet Arduino UNO die genaue Position an den Motor Treiber Sabertooth. Der Sabertooth steuert zwei Getriebemotoren zu dieser gerechneten Position.

6.5 Cockpit

Um die Drehzahl (RPM) und die Geschwindigkeit anzuzeigen, wird ein analoger Tachometer E36 vom Hersteller BMW verwendet, der mit einem Arduino-Mega 2560 angesteuert wird. Die folgende Abbildung zeigt das Blockdiagramm des Cockpits:

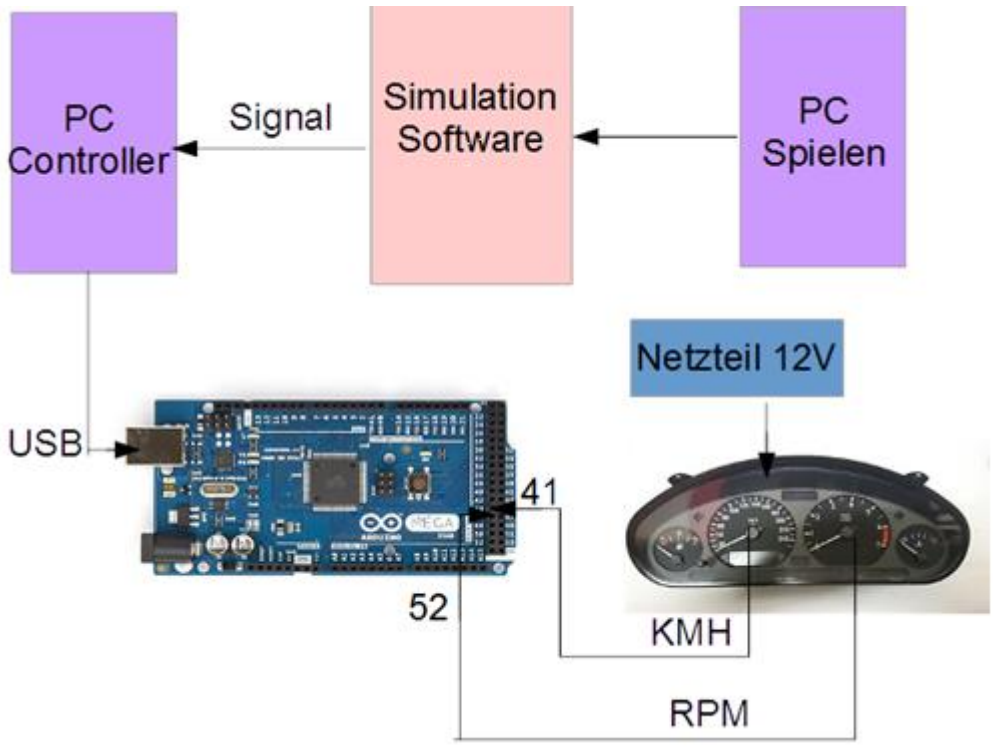


Abbildung 53: Schaltplan des Cockpits

Durch das Plugin Game Dash [16] der Simulation Software SimTool empfängt der Arduino Mega die Gamedaten und steuert damit den analogen Tachometer an.

6.6 Netzteil 12VDC

In dem Fahrsimulator verwendet man zwei Getriebemotoren 12VDC mit großer Leistung. Aus diesem Grund braucht das System ein gutes Netzteil mit hohem Versorgungsstrom und einer konstanten Spannung 12VDC. Um den Versorgungsstrom des Netzteils zu erhöhen und die Störsignale zu vermeiden, wurden zwei gleiche Netzteile 12VDC parallelgeschaltet.

Daten eines Netzteils:

- Max. Strom: 48A
- Spannung: 12VDC
- Max. Leistung: 650 W

[16] Game Dash 1.1, URL: https://www.xsimulator.net/community/threads/game-dash.4978 . Abruf: 2016-11-15

7 Sicherheitsmaßnahmen

7.1 Hardware

7.1.1 Notschalter

Um das Gesamtsystem zu schützen, verwendet man einen Notschalter. Wenn das Fahrsimulatorssystem oder der Fahrer/Spieler ein Problem hat, kann man das System schnell ausschalten.

Abbildung 54: Fahrsimulator mit Notschalter

7.1.2 Überhitzungsschutz

Um den Motortreiber Sabertooth vor Überhitzung zu schützen, wird ein günstiges Temperaturmodul genutzt, damit wird die Sicherheit des Systems erhöht. Das Temperaturmodul besteht aus einem Temperatursensor, einer Led-Anzeige und einem kleinen aktiven Lautsprecher. Der Temperatursensor misst die Wärme vom MOSFET des Motor Treibers. Wenn diese Wärme über 40 Grad Celsius ist, schaltet sich der Minilautsprecher automatisch ein und alarmiert den Spieler. Außerdem wird der Motortreiber mittels des Aufbaus von zwei Lüftern abgekühlt.

Abbildung 55: Gehäuse für Steuermodul mit Temperatur-Anzeiger

7.1.3 Motor Schutz

Während der Funktion des Getriebemotors birgt das Anfassen des Motors eine Verletzungsgefahr für den Spieler. Um dieses zu vermeiden, wird der Getriebemotor in ein Gehäuse eingebaut.

7.2 Software

Mit dem Mikrocontroller Arduino UNO wird die Geschwindigkeit des Getriebemotors durch die Änderung vom Tastverhältnis des PWM-Signals gesteuert. Man kann zwei Motoren steuern, um sich schneller oder langsamer zu bewegen. Mit dem Motor Treiber Sabertooth nutzt man eine Funktion „ST.motor()“ in der Programm- Bibliothek. Die Geschwindigkeit ist abhängig von der Eingabe des Wertes in dieser Funktion. Im folgenden Text wird die Funktion erklärt.

Die Funktion „ST.motor()“ arbeitet mit zwei Parametern:

ST. motor (Motor, Power)

Motor: welcher Motor (1 oder 2) wird angesteuert

Power: $-127 \leq Wert \leq 127$. Wenn der Wert gleich 0 ist, bleibt der Motor stehen. Und wenn der Wert gleich 127 oder -127 ist, dreht der Getriebemotor sich in die Vorwärts- und Rückwärtsrichtung mit der maximalen Geschwindigkeit. Werte zwischen 0 und + oder -127 ergeben langsamere Bewegungen.

8 Zusammenfassung/Ausblick

Im letzten Kapitel wird eine zusammengefasste Darstellung der Arbeit beschrieben. Dies beinhaltet, wie der Fahrsimulator nach den Anforderungen des Kunden aufgebaut wurde. Im nachfolgenden Unterpunkt “Ausblick“ werden Ansatzpunkte beschrieben, die im Anschluss an diese Bachelorarbeit weiterbearbeitet werden können

8.1 Zusammenfassung

In der Arbeit wurde ein Fahrsimulatorssystem auf Grundlage der vorhandenen Anforderungen entwickelt. Es wurden die Parameter des Getriebemotors mittels der mathematischen Methode der kleinsten Quadrate identifiziert, ohne dass ein besonderes Datenerfassungsgerät für die Drehzahlmessung zur Verfügung stand. Durch die Simulation in Matlab/Simulink konnten unkompliziert alle Parameter des PID-Reglers bestimmt werden. Es können zwei unterschiedliche Motion Simulation Programme genutzt werden, um damit die Gamedaten zu dem Mikrocontroller zu senden. Das System ist offen für alle Rennspiele, dessen Daten von den Motion Simulation Programmen ausgelesen werden können. Das System des Fahrsimulators bewegt sich realistisch, ohne Verzögerungen und parallel zu den Bewegungen des Fahrzeugs im Spiel und der Spieler fühlt sich, als würde er in einem realen Auto sitzen. Ohne die Arbeitszeit zu berechnen, wurden für die in dieser Arbeit verwendeten Bauteile knapp 300,00 Euro investiert. Mit den Kosten für die vorhandene Sitzkonstruktion wurde das geforderte Budget von 700,- Euro eingehalten. Auf Löt- und Schweißarbeiten wurde weitestgehend verzichtet, um die Wartung zu vereinfachen. Für die Vereinfachung der Bedienung für den Spieler wurde der Startvorgang des Fahrsimulators automatisiert. Durch den modularen Aufbau kann der Fahrsimulator einfach demontiert und transportiert werden.

Insbesondere wurden alle Probleme des Fahrsimulators während des Praktikum gelöst. Durch die statische Berechnung konnte der geeignete Getriebemotor ausgewählt werden. Die aktuelle Version des Fahrsimulators kann mit einem Spieler mit großem Gewicht (80kg) ohne Probleme genutzt werden. Das Steuermodul wird mit einem professionellen Gehäuse mit Kühlung betrieben, womit ein langer Spielbetrieb möglich ist.

8.2 Ausblick

In der nächsten Version des Fahrsimulators könnte man die mechanische Konstruktion des Sitzplatzes mit hoher Stabilität und Sicherheit optimieren. Der Fahrsimulator sollte mit drei Motoren arbeiten. Damit könnte der Fahrsimulator sich mit den maximalen Freiheitsgraden von 6DOF bewegen, um noch mehr Realismus zu erreichen.

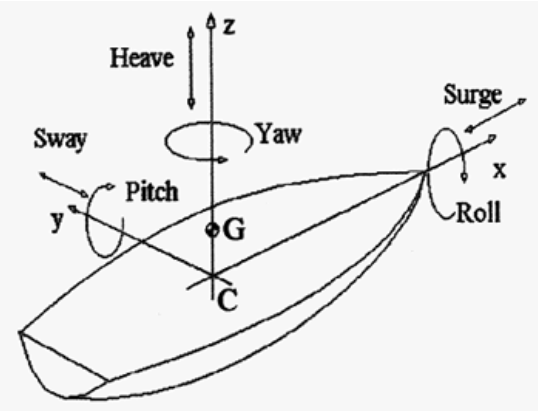


Abbildung 56: Das Bewegungssystem mit 6 DOF[17]

8.3 Hinweis zum Anhang

Der Anhang zur Arbeit befindet sich auf CD und ist bei den Prüfern Professor Haase und Professor Maaß einzusehen. Darin enthalten sind die Datenerfassungen der Drehzahlmessung, das Matlab-Skript und der Programmcode, der nur in digitaler Form zur Verfügung steht.

[17] Bewegungssystem https://www.xsimulator.net/co-ordinate-system Abruf: 2016-11-15

Literaturverzeichnis

[1] SCHRÖDER, DIERK: *Elektrischer Antriebe – Grundlagen*. 4., erw. Aufl. München: Springer-Vegl., 2009. – ISBN 978-3-642-02989-9

[2] SCHRÖDER, DIERK: *Elektrische Antriebe – Regelung von Antriebssystemen*. 4., erw. Aufl. München: Springer-Verl., 2015. – ISBN 978-3-642-30096-7

[3] GINZEL, JENS: *Elektrische Antriebstechnik*. Vorlesungsskript, 2013

[4] RETTIG, RAMUS: *Sensorik*. Vorlesungsskript, 2015

[5] DIETMAR, GROSS; WERNER, HAUGER (Mitarb.); JÖRG, SCHRÖDER (Mitarb.); WOLFGANG, WALL (Mitarb.): *Technische Mechanik 3*. 13., überarb. Aufl. Berlin: Springer Vieweg, 2015. – ISBN 978-3-642-53954-1

[6] DIETMAR, GROSS; WERNER, HAUGER (Mitarb.); JÖRG, SCHRÖDER (Mitarb.); WOLFGANG, WALL (Mitarb.): *Technische Mechanik 1*. 12., überarb. Aufl. Heidelberg: Springer Vieweg, 2016. – ISBN 978-3-662-52715-3

[7] FISCHER, ROBERT; JÜRGENS, GUNTER (Mitarb.); NAJORK, ROLF (Mitarb.): *Das Getriebebuch*, Aufl. Wien: Springer-Verl., 2012. – ISBN 978-3-7091-0876-5

[8] JOHANNES, VOLMER: *Getriebetechnik: Lehrbuch*. 5., durchges. Aufl. Berlin: Technik-Verl., 1987. – ISBN 3-341-00270-7

[9] FRICKE, ANDREAS; GÜNZEL, DETLEF (Mitarb.); SCHAEFFER, THOMAS (Mitarb.): *Bewegungstechnik*, Aufl. München: Carl Hanser Verl., 2015. – ISBN 978-3-446-444 10-2

[10] DELGADO, RICABAL: *Regressionsanalyse*. Vorlesungsskript, URL http://www.wiwi.uni-rostock.de/fileadmin/Institute/VWL/LS_Statistik/vorl_gs/Regression_II.pdf . – Abruf: 2016-11-15

[11] MathWorks Documentation: *DC Servo Motor Parameter Estimation*, URL https://de.mathworks.com/help/sldo/examples/dc-servo-motor-parameter-estimation.html . – Abruf: 2016-11-15

[12] JOCHEM, UNGER: *Einführung in die Regelungstechnik*. 3., überarb. und erg. Aufl. Stuttgart: Teubner-Verl., 2004. – ISBN 3-519-20140-2

[13] MEINERS, ULFERT: *Regelungstechnik 2*. Vorlesungsskript, 2014

[14] FÖLLINGER, OTTO; DÖRRSCHEIDT, ULRICH KONIGORSKI (Bearb.); BORIS LOHMANN (Bearb.): *Regelungstechnik: Einführung in die Methode und ihre*

Anwendungen. 12., überarb. Aufl. Offenbach: VDE-Verl., 2016. – ISBN 978-3-8007-4166-3

[15] PLÖTZENEDER, FRIEDRICH; PLÖTZENDER, ANDREAS: *Powerprojekte mit Arduino und C*, Franzis-Verl., 2013. – ISBN 978-3-6456-5131-8

[16] CHRISTOPH, KECHER; ALEXANDER, SALVANOS: *UML 2.5: Das umfassende Handbuch*. 5., erw. Aufl. Bonn: Rheinwerk Computing-Verl., 2015. – ISBN 978-3-8362-2977-7

[17] SPARKFUN MONSTER MOTO SHIELD: *Programmbibliothek*. Online unter: http://cdn.sparkfun.com/datasheets/Dev/Arduino/Shields/MonsterMoto_Shield_Example.pde Abruf: 2016-11-15

[18] DIMENSIONENGINEERING: *Programmbibliothek für Arduino*. Online unter: https://www.dimensionengineering.com/info/arduino Abruf: 2016-11-15

[19] *Sabertooth dual 60A motor driver*: Datenblatt. Online unter: https://www.dimensionengineering.com/datasheets/Sabertooth2x60.pdf Abruf: 2016-11-15

[20] *Optical incremental encoder HEDM 5500*: Datenblatt. Online unter: http://docs.avagotech.com/docs/AV02-1046EN Abruf: 2016-11-15

[21] BMW e36 Tacho zu Arduino. Online unter: http://www.sim-pc.de/bmw-e36-tachosteckerbelegung Abruf: 2016-11-15

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, den 25 November 2016 _____
Ort, Datum *Unterschrift*