Weiterentwicklung des eingebetteten Systems "Autonomes Motorrad"

Institut für Technische Informatik und Kommunikationsnetze
Eidgenössische Technische Hochschule Zürich

Titelseite: Bild des fahrbereiten Motorradmodells

Betreuung:
Prof. Dr. L. Thiele/TIK
Prof. Dr. A. H. Glattfelder/IfA
Matthias Gries/TIK

Autor:
Simon Stürm, Juli-1998

Aufgabenstellung

Weiterentwicklung des eingebetteten Systems “Autonomes Motorrad”

Aufgabenstellung für die Diplomarbeit von Simon Stürm, IIIB SS98

In dieser Arbeit soll das bereits in mehreren Semesterarbeiten [SA98] entworfene Modell eines autonom fahrenden Motorrades weiterentwickelt werden, so dass es im Rahmen einer Vorlesung zum Entwurf eingebetteter Systeme vorgestellt werden kann. Unter eingebetteten Systemen versteht man Rechensysteme, die auf einen Anwendungsbereich zugeschnitten (z.B. industrielle Rechneranwendungen, Informationsverarbeitung, Steuerung) und in einen technischen Kontext eingebunden sind. Eingebettete Systeme sind hybride Schaltungen aus der Sensorik und Elektrotechnik, die nach bestimmten Kriterien wie Kosten, Energieverbrauch und Platzbedarf optimiert gegebenenfalls autonom Steuerungs- und Regelungsaufgaben übernehmen. Die Arbeit fächert sich daher in unterschiedliche Arbeitsgebiete auf, die Themen aus der Mechanik, der Regelungstechnik und dem Entwurf von Hard- und Software umfassen. Aufgrund der interdisziplinären Ausrichtung wird die Arbeit vom Institut für Automatik (IfA) und vom Institut für Technische Informatik und Kommunikationsnetze (TIK) betreut.

Folgende Teilaufgaben sind für die Inbetriebnahme des Modells zu bearbeiten:

- Qualitative und quantitative Bestimmung von Kenngrössen der Sensor- und Stellsignale wie z.B. Varianz, Erwartungstreue und typischer Wertebereich von pulsweitenmodulierten Signalen.
- Qualitative und quantitative Bestimmung von mechanischen Eigenschaften des verwendeten Motorradmodells und der Servomotoren.
- Vervollständigung der Serviceroutinen für die Verarbeitung und Generierung der Mess- und Stellsignale, wobei ggf. auch Anpassungen an der Hardware erforderlich sind.
- Inbetriebnahme des Modells mit Hilfe eines minimalen Regelkonzepts.

Folgende Teilaufgaben sind für die Optimierung und Präsentation des Modells zu bearbeiten:

- Das Regelkonzept ist unter Berücksichtigung der statistischen und mechanischen Kenngrössen zu verfeinern, so dass das Modell langsamer und dabei stabiler fahren kann, als es vor den Änderungen der Fall war. Ggf. sind weitere Sensorsignale zu verwenden, so dass die Demonstration des Modells auf kleinem Raum, z.B. in einem Hörsaal, möglich wird.

1

- Das Motorradmodell ist in einer für Laien verständlichen Form zu modellieren und zu dokumentieren, so dass der Bezug und die Schnittstellen zwischen Hardware- und Softwarestrukturen und dem mechanischen Modell deutlich werden.
- Die Dokumentation dieser Teilaufgabe sollte insbesondere die Schwierigkeiten der Regelung herausstellen und die Teilschritte der Verfeinerung der Regelung begründen. Schliesslich sollte sie eine einfache Weiterverwendung und Demonstration des Modells z.B. im Rahmen einer Lehrveranstaltung erlauben.

Hilfsmittel:

- Verschiedene Simulations- und Modellierungsumgebungen wie z.B. Statemate, Matlab Simulink o.ä.
- HiWare-Entwicklungsumgebung mit Assembler/C-Crosscompiler und Debugger für die Mikrokontroller- Softwareerstellung auf dem Motorola Mikrocontroller MC68332 [Mot90].
- Die Einrichtungen der Laboratorien der Institute IfA und TIK sowie der Mechanikwerkstatt.

Betreuung:

- M. Gries, Institut TIK, ETZ G63, Tel.: 01 / 632 70 61
  email: gries@tik.ee.ethz.ch
- Prof. Dr. A. H. Glattfelder, Institut IfA, ETL K22.3
  email: glatt@aut.ee.ethz.ch

Professoren:

- Prof. Dr. A. H. Glattfelder, Institut IfA, ETL K22.3
  email: glatt@aut.ee.ethz.ch
- Prof. Dr. L. Thiele, Institut TIK, ETZ G87
  email: thiele@tik.ee.ethz.ch

Literatur

[Mot90] *"M68332EVK Evaluation Kit User's Manual"*, Motorola Semiconductor Inc., 1990

[SA98] M. Neff, R. Odermatt, A. Stucki, *"Modellierung, Konstruktion und Implementation eines autonom fahrenden Zweiradmodells"*, Institute TIK und IfA der ETH Zürich, Semesterarbeiten WS 1997/98

KAPITEL 1 Einleitung

Eingebettete Systeme bestehen aus einer Recheneinheit, welche über Sensoren und Aktoren mit ihrer Umwelt und gegebenenfalls über ein Mensch-Maschinen-Interface mit ihrem Bediener zusammenarbeitet. Um in einer Vorlesung über den Entwurf eingebetteter Systeme ein anschauliches Beispiel zeigen zu können, entstand die Idee, ein rechnergesteuertes Motorrad zu bauen. Es soll sich mit Hilfe von Sensoren und Aktoren selbst stabilisieren und über die Fernsteuerung gelenkt werden. Dabei spielt nicht nur die Qualität der Stabilisierung, sondern auch die Übersichtlichkeit des Systems eine Rolle. Es soll mit möglichst wenigen Sensoren und einer relativ einfachen Regelung auskommen und klar dokumentiert werden. Die Einfachheit soll es ermöglichen, das Motorrad ohne grossen Zeitaufwand, zum Beispiel im Rahmen einer Vorlesung, in Betrieb zu nehmen. Die erste Arbeit an diesem Projekt [SA98] brachte wichtige Grundlagen, auf welche diese Arbeit aufbaut. Vor allem durch die mathematische Modellierung und die Auswahl des Motorrades in jener Arbeit wurde das Projekt in eine klare Richtung gebracht.

Beim vorliegenden Motorradmodell handelt es sich um ein mechanisches System, das von sich aus instabil ist und sich ohne Unterstützung einer Regelung nicht fernsteuern lässt. Ein Motorradfahrer kann insbesondere durch Gewichtsverlagerung Einfluss auf das Fahrverhalten seines Motorrades nehmen. Diese Möglichkeit fehlt beim Modell. Die Stabilisierung kann daher nur durch aktives Lenken erfolgen. Zusätzlich müssen die Komponenten, welche für die Elektronik und Sensorik ausgewählt werden, Kriterien bezüglich Gewicht, Grösse und Stromverbrauch genügen. Die Elektronik muss resistent gegen Erschütterungen und elektromagnetische Störungen sein. Die mechanische Konstruktion muss verbessert werden, so dass die Lenkung präziser und der Lenkwinkel grösser wird. Durch die Erweiterung des Lenkwinkels kann der Kurvenradius verkleinert werden, was ein Fahren auf kleinerem Raum ermöglicht. Die anschliessende Auswahl der Regelungsmethode und das Bestimmen der Regelparameter ist ein zentraler Punkt dieser Arbeit. Die Regelung muss dabei in der Software implementiert und Serviceroutinen für die Verarbeitung der Sensorsignale (analog und digital) angeboten werden. Die Arbeit ist daher interdisziplinär ausgelegt, da sie Arbeitsgebiete aus der Mechanik, Regelungstechnik, Sensorik und Elektronik umfasst, und behandelt Forschungsgebiete zweier Institute, des Instituts für technische Informatik (TIK) und des Instituts für Automatik (IfA).

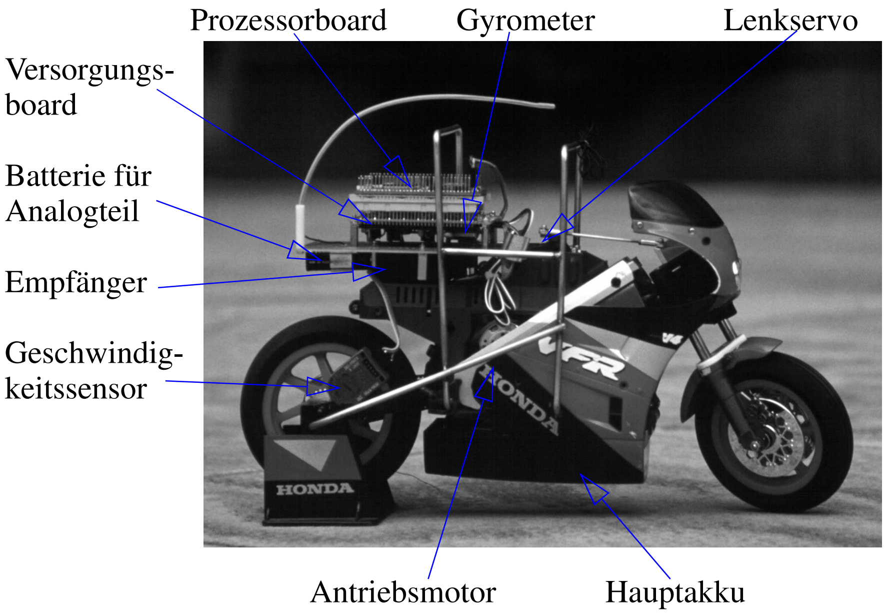


Abbildung 1. **Seitenansicht des Motorrades**

Das folgende Kapitel soll einen Überblick über die Arbeit geben. Die Auswahl der Sensoren und Aktoren wird in den Kapiteln 3 und 4 beschrieben. Kapitel 5 und 6 erklären die mathematische Modellierung und die Auslegung der Regelung. In Kapitel 7 findet man die Schaltpläne der verbesserten Elektronik und in Kapitel 8 die Beschreibung der Softwarestruktur. Die Arbeit schliesst mit einer Zusammenfassung und einem Ausblick auf Möglichkeiten der weiteren Verbesserung ab. Der Anhang beginnt mit den Softwarelistings der C-Programme für die Regelungs- und Serviceroutinen und des Matlab M-Files für die Simulation. Im Anhang B folgen die Messresultate der dynamischen Eigenschaften von Servo und Gyrometer, die Messkurven von Lenkwinkel, Winkelgeschwindigkeit, Neigung und Geschwindigkeit, welche in freier Fahrt des geregelten Motorrades aufgezeichnet wurden, und die dazugehörigen Simulationsergebnisse. Dann folgt die Bedienungsanleitung, welche man vor der Inbetriebnahme studieren sollte. Zum Schluss folgt noch eine Liste der Variablen, welche zusätzlich im Verlauf des Berichts beschrieben sind.

KAPITEL 2

Überblick über das System

2.1 Verwendetes Modellmotorrad

Das verwendete Modellmotorrad ist ein im Massstab 1:5 nachgebildetes Modell der Firma Graupner. Es ist 40 Zentimeter lang und 2.3 Kilogramm schwer. Das gekaufte Modell wurde durch Gewichtsverlagerung gesteuert, das Vorderrad war frei beweglich und konnte auch zur Seite ausgeschwenkt werden (vgl.[SA98]). Durch Federn wurde das Vorderrad in die Mitte gedrückt und erhielt dadurch recht gute Fahreigenschaften. Es brauchte jedoch eine recht hohe Mindestgeschwindigkeit, und der Kurvenradius war relativ gross.

2.2 Mechanischer Umbau

Die Lenkung wurde bereits in der Vorgängerarbeit durch eine feste Vordergabel ersetzt, welche durch einen Lenkservo gesteuert wird. Dadurch konnte man das System mathematisch erfassen und durch einen Prozessor regeln. Das Fahren von Hand, das heisst, gesteuert durch die Fernbedienung, ist jetzt ohne Unterstützung des Prozessors nicht mehr möglich. Die Anforderung an die Genauigkeit und die Geschwindigkeit der Lenkbewegungen ist zu hoch.

2.3 Unterbringung der Elektronik

Anstelle der Fahrerpuppe wurden drei Platinen angebracht, erstens eine Befestigungsplatine, zweitens eine Versorgungsplatine und drittens die Prozessorplatine. Auf der Versorgungsplatine wurde ein Gyrometer, ein Sensor welcher die Winkelgeschwindigkeit misst, untergebracht, und am Hinterrad wurde ein Sensor zum Messen der Geschwindigkeit befestigt. Als Prozessor kommt ein Motorola MC68332 zum Einsatz, ein 32 Bit Microcontroller, dessen Subsysteme wichtige Timing- und Schnittstellenaufgaben übernehmen können.

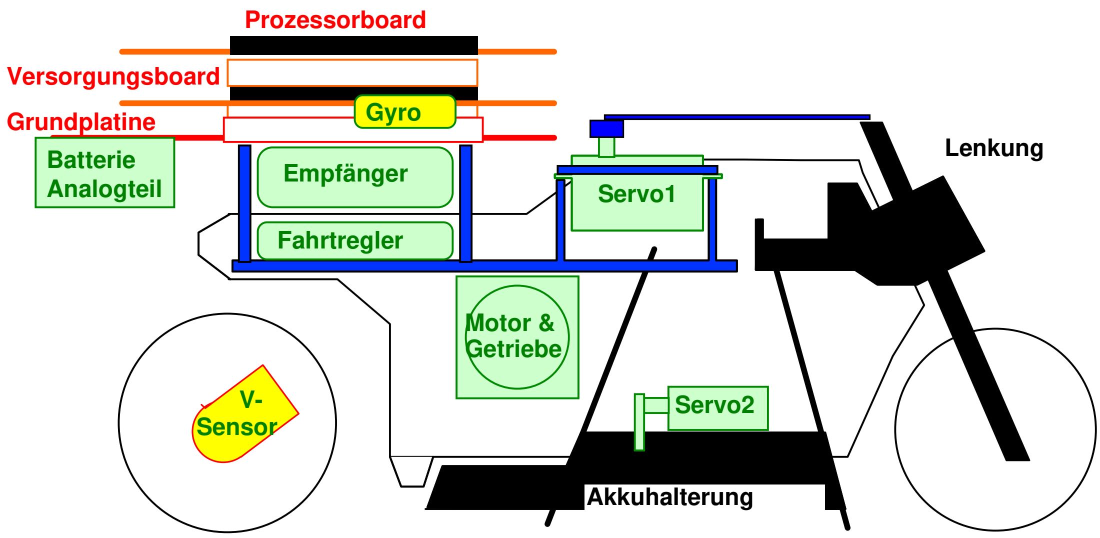


Abbildung 1. Schema des autonomen Motorrades

2.4 Regelung in Software

Die eigentliche Regelungsfunktion, das heisst, die Berechnung neuer Stellbefehle aufgrund von Eingängen der Sensoren, ist in Software implementiert. Die Programme werden in der Programmiersprache C geschrieben, mit einem Crosscompiler in Assembler übersetzt und dann auf den Prozessor heruntergeladen. Dies ermöglicht das einfache Verändern der Regelungsfunktion oder ihrer Parameter und damit das Ausprobieren verschiedener Regelkonzepte.

KAPITEL 3

Sensoren des Systems

Die Zusammenstellung von Robotersensoren [BEF96] bringt einen guten Überblick über den heutigen Stand der Technik und die Preislage von Sensoren. Im folgenden Abschnitt werden jene Sensortypen diskutiert, welche für dieses Projekt evaluiert wurden.

3.1 Allgemeine Evaluation von möglichen Sensoren

3.1.1 Inklinometer

In der Arbeit [SA98] wird ein Inklinometer zum Bestimmen der Neigung des Motorrades vorgeschlagen. Aus verschiedenen Gründen ist dieses Messteil ungeeignet für diese Aufgabe. Ein Inklinometer, egal ob ein Pendel, ein Quecksilberbehälter oder eine Wasserwaage, misst die Abweichung β der Resultierenden von Zentrifugalkraft *Ff* und Schwerkraft *Fg* zur Achse des Motorrads mit Neigung α.

$$\beta = \alpha - \operatorname{atan}\left(\frac{F_f}{F_g}\right)$$

Sofern man die Grösse der Fliehkraft genau bestimmen kann, lässt sich aus dem Messresultat eines Inklinometers die Neigung ausrechnen. Der Winkel, welcher das Inklinometer misst, kann folgendermassen angenähert werden:

$$\beta = \alpha + \frac{h_{ink}v}{gl}\dot{\alpha} - \frac{v^2}{gl}\gamma$$

Das heisst, der Messwinkel des Inklinometers ist abhängig von der Geschwindigkeit *v*, dem Lenkwinkel γ, der Winkelgeschwindigkeit $d\alpha/dt$ und der Neigung α des Motorrades. Aus diesem Zusammenhang die Neigung zu berechnen ist sehr schwierig. Erschwerend kommt hinzu, dass die Dämpfung des Inklinometers in Oel eine Verzögerung der Messung bewirkt. Die Berechnung müsste also zusätzlich ein Modell der zeitlichen Verzögerung enthalten. Ausserdem ist die Einschwingzeit des Inklinometers von 50 bis 75 ms, bei einer Zeitkonstante des Motorrades von 125 ms, zu gross (vgl Messungen im Anhang B.7).

3.1.2 Gyrometer

Gyrometer nutzen die Corioliskraft, welche bei einer Dralländerung, das heisst, bei der Richtungsänderung der Drehachse eines Kreisels entsteht, um die Winkelgeschwindigkeit dieser Änderung zu bestimmen. Dabei unterscheidet man drei Arten von Gyros:

Mechanische Gyros

Mechanische Gyros enthalten einen mechanischen Kreisel. Dieses Prinzip wurde Anfang des 19. Jahrhunderts zum ersten Mal entwickelt und ist in vielen Flugzeugen im Einsatz. Diese hochpräzisen Messgeräte kosten zwischen 10000 und 100000 Dollar. Es gibt auch ein paar Low-Cost Gyros, deren Genauigkeit und Drift jedoch schlechter ist. Hier ein Beispiel für eine Low-Cost Variante, der Futaba FG 154, welcher für Modellbauhelikopter verwendet wird:

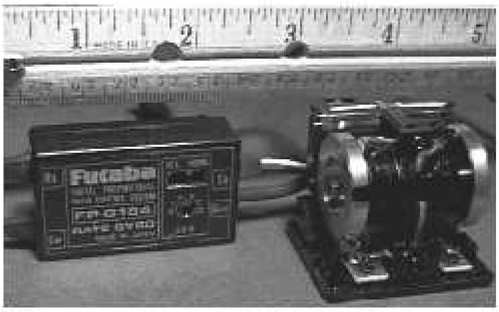


Abbildung 2. **Futaba Modellbaugyro**

Dieser Modellbaugyro wiegt ca. 100 Gramm und kostet etwa 150 Dollar.

Optische Gyros

Optische Gyros wurden erst durch die Entwicklung der Lasertechnologie möglich. Zwei gegenläufige Lichtwellen in einem ringförmigen, verlustfreien Spiegel bilden eine stationäre, stehende Welle. Dreht man nun die Umgebung dieser Welle, so kann man durch Zählen der Nullstellen den Drehwinkel herausfinden. Dies ist das Grundprinzip eines optischen Gyros. Da jedoch ein verlustfreier Spiegel nicht realisierbar ist, sehen die implementierten Geräte etwas anders aus. Genauere Angaben über die verschiedenen Arten von optischen Gyros erhält man in [BEF96].

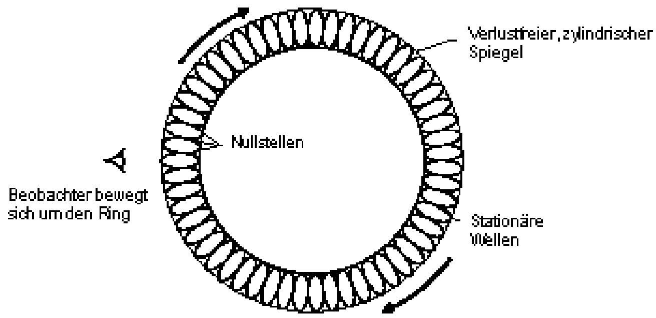


Abbildung 3. Prinzip des idealisierten,optischen Gyros

Ein Beispiel für einen möglichen optischen Gyro ist der Andrew Autogyro. Er wiegt ca. 250 Gramm, hat eine Drift von 18 Grad pro Stunde und der Preis liegt im Bereich zwischen 1000 und 1500 Dollar.

Abbildung 4. Beispiel für einen optischen Gyro

Piezoelektrische Gyros

Piezoelektrische Gyros sind mechanische Gyros, welche jedoch keine rotierenden Teile, sondern nur schwingende Teile haben. Es gibt auf dem Markt Sensoren mit einer schwingenden Gabel oder mit einem schwingenden Dreieck (vgl. [IBM96]).

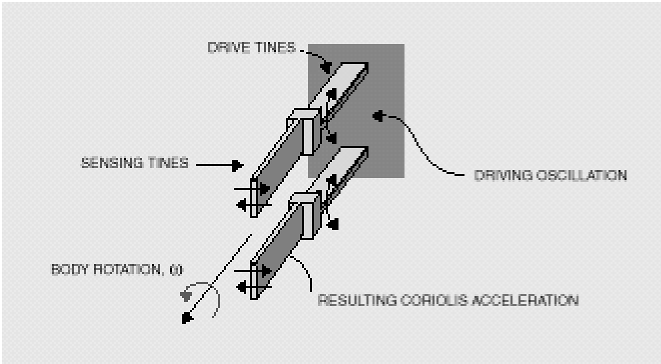


Abbildung 5. **Gyrosystem mit schwingender Gabel**

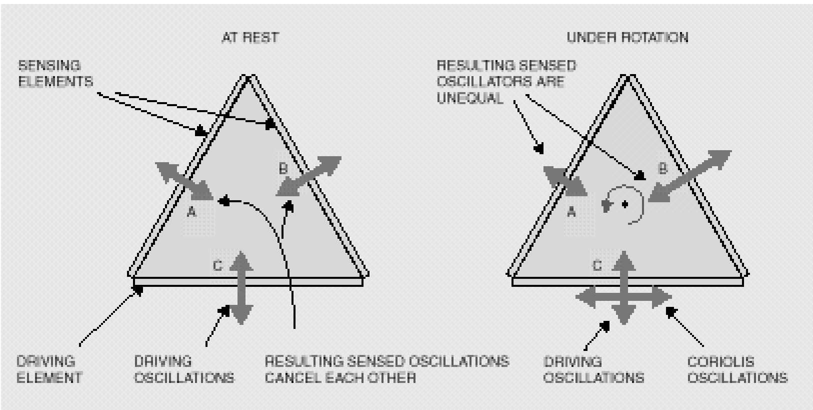


Abbildung 6. **System mit schwingendem Dreieck**

Ein Beispiel für piezoelektrische Gyros sind die Gyrostars von Mutrata, welche das System des schwingenden Dreiecks verwenden. Diese Sensoren sind nur wenige Gramm schwer und können deshalb gut auf einer Platine untergebracht werden. Natürlich ist bei dieser günstigen und kleinen Variante, der Preis liegt bei 45 Franken, die Genauigkeit nicht besonders gut. Die Linearität liegt im Bereich von +- 5 Prozent und die Drift beträgt bis zu 30 Grad pro Minute.

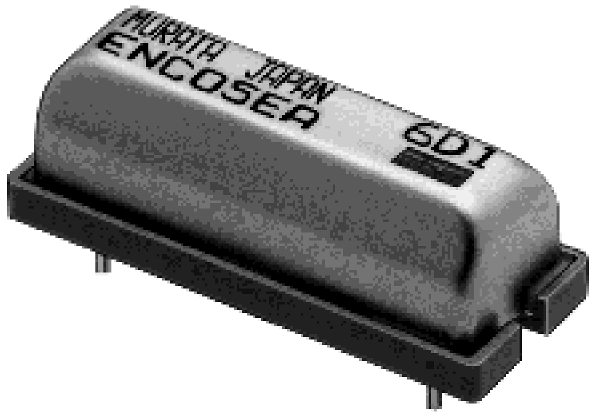


Abbildung 7. Murata ENC-05 piezoelektrischer Gyro

Zum Abschluss der Sammlung von Informationen über Gyros folgt nun eine Übersicht über die erhältlichen Modelle:

Tabelle 1. Übersicht über Gyrometer

Modell	Typ	Linearität (% F.S)	Drift (grad/s)	Gewicht	Preis
Futaba FG 154	mechanisch	?	?	100g	150$
Andrew Autogyro	optisch	0.0005	0.005	250g	1250$
Hitachi OFG-3	optisch	0.0005	0.0028	480g	1500$
Murata ENV-05	piezoel.	0.5	0.45	45g	80-300$
Murata ENC-05	piezoel.	0.5	?	2.7g	30$

Die Auswahl des Gyrometers wird in “3.2 Gyrometer” und jene des Geschwindigkeitssensors in “3.3 Geschwindigkeitssensor” beschrieben.

3.1.3 Distanzsensoren

Um den realen Neigungswinkel des Motorrades unabhängig von Fliehkräften zu messen, kam die Idee einer seitlichen Distanzmessung zum Boden. Diese Distanzmessung muss sehr genau sein, denn ein Fehler von einem Millimeter bewirkt eine Abweichung der Neigung um ein Grad. Zusätzlich ist eine Störungsanfälligkeit auf eine Neigung der Zielfläche, eine Änderung des Untergrundes und auf die Bewegung der Zielfläche relativ zum Messgerät zu beachten.

Ultraschallsensoren

Eine Messung mit Ultraschall ist anfällig auf eine Neigung der Zielfläche. Bei einer Neigung von mehr als 10 Grad, welche bei unserem Motorrad bei weitem überschritten wird, wird der Strahl vom Empfänger weg reflektiert. Die Messung ist ebenfalls anfällig auf Luftverwirbelungen, wie sie durch den Fahrtwind entstehen, und ist deshalb für ein Motorrad ungeeignet.

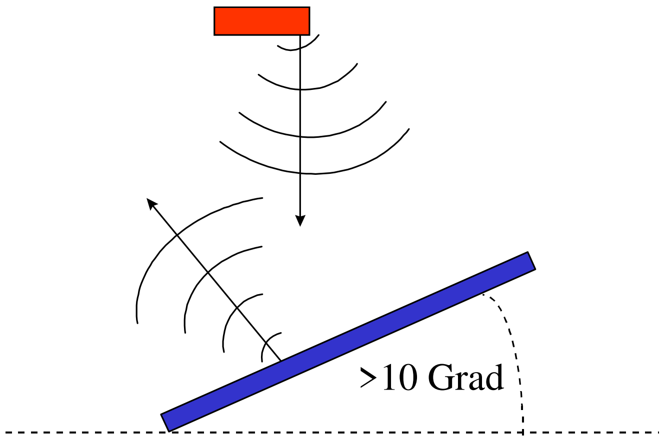


Abbildung 8. Ultraschallmessung bei geneigter Zielfläche

Laserdistanzmessung (LDS)

Bei der Laserdistanzmessung wird ein Laserstrahl auf die zu messende Fläche gerichtet. Die reflektierte Strahlung wird von einem PSD (Position Sensitive Device) gemessen und daraus die Entfernung bestimmt [Neu96]. Die Messung bringt auch bei einer Neigung der Fläche nur geringe Fehler und ist nicht anfällig auf Störungen durch die Bewegung des Objektes. Problematisch sind starke Farbschwankungen des Untergrundes ebenso wie Unebenheiten. Der Messbereich ist abhängig von der Montagehöhe *hs* und dem seitlichen Abstand *ls* von der vertikalen Achse des Motorrades. Der Neigungswinkel berechnet sich folgendermassen, wobei *hm* der Messwert ist:

$$\alpha = \operatorname{atan}\left(\frac{h_s - h_m}{l_s}\right)$$

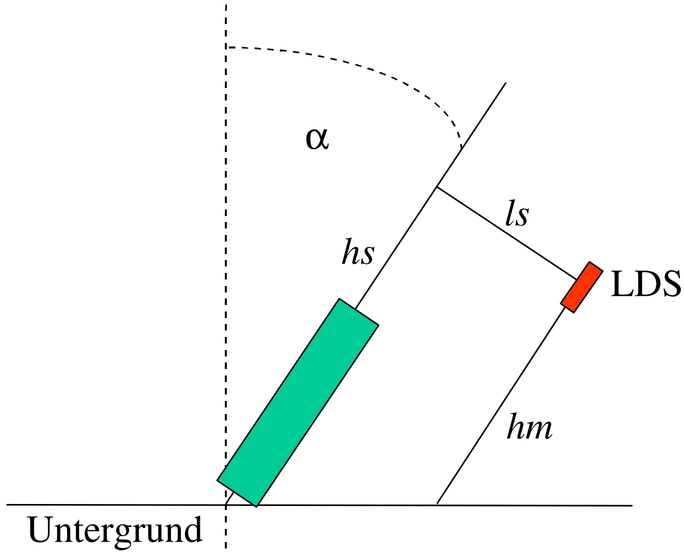


Abbildung 9. **Skizze zur Berechnung der Neigung mit LDS (Motorrad von hinten gesehen)**

Für eine seitliche Abweichung von 5cm ergibt sich ein Messbereich von +-50 mm, wofür Sensoren auf dem Markt erhältlich sind.

Tabelle 2. **Übersicht über LDS-Sensoren**

Produkt	**Gewicht**	**Speisung**	**Ausgang**	**Preis (Fr.)**
Matsushita LM10	480g	12-24 VDC	+-5V / 4...20 mA	ca. 1300.-
Raytec LDS1-100	250g (nur Sensor)	220 VAC	+-10V /4..20 mA	3880.-
NFM Inc. M5/100	?	21-28 VDC	+-10V /4..20 mA	2800.-

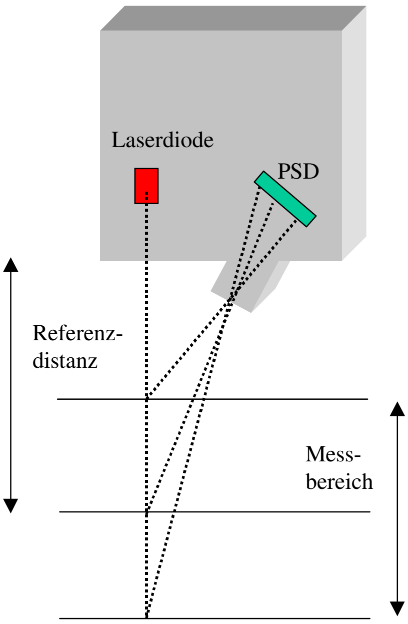


Abbildung 10. **Prinzip der Laserdistanzmessung**

Der Einsatz eines LDS-Sensors wäre die einzige berührungsfreie Methode, welche bei der Bestimmung des effektiven Neigungswinkels Erfolg verspricht. Gegen die Anschaffung eines solchen Sensors spricht das Gewicht, der Preis und die Erhöhung der Komplexität des Systems.

3.1.4 Geschwindigkeitssensoren

Um die Geschwindigkeit des Motorrades zu bestimmen, kamen folgende Sensortypen in Frage:

Ultraschall Dopplereffektsensor

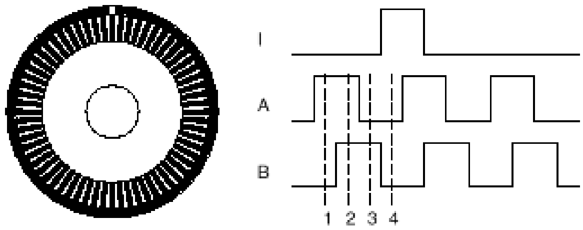
Eine berührungslose Methode zum Bestimmen der Geschwindigkeit sind Ultraschallsensoren. Durch Messen der Differenz zwischen der Frequenz des gesendeten Signals und jener der reflektierten Welle kann die Geschwindigkeit bestimmt werden. Die Sensoren sind relativ schwer und teuer und wurden deshalb für dieses Projekt nicht eingesetzt.

Messung der Hinterraddrehzahl mit einer Lichtschranke

Dies wäre die einfachste und billigste Messmethode. Ihr Nachteil besteht darin, dass die Auflösung relativ schlecht ist, da sich das Hinterrad bei den kleinsten Geschwindigkeiten nur 2 bis 3 Mal pro Sekunde dreht. Zudem ist bei Lichtschranken die Gefahr sehr gross, dass sich Sensor und Rad berühren und es deshalb zu Beschädigungen kommt.

Inkrementelle, optische Encoder

In einem Gehäuse misst eine Lichtschranke die Umlaufgeschwindigkeit einer Codescheibe. Von aussen wird die Achse dieser Codescheibe angetrieben, und als Ausgang dieses Sensors erhält man ein Rechtecksignal, dessen Periode der Drehgeschwindigkeit entspricht. Je nach Auflösung der Codescheibe eignet sich die Messung für kleinere oder grössere Geschwindigkeiten.



State	Ch A	Ch B
S_1	High	Low
S_2	High	High
S_3	Low	High
S_4	Low	Low

Abbildung 11. **Codescheibe und Ausgangsignal eines optischen Encoders**

Der ausgewählte Geschwindigkeitssensor wird in “3.3 Geschwindigkeitssensor” beschrieben.

3.2 Gyrometer

Modell

Da mit dem piezoelektrischen Gyro ENC-05 von Murata am Institut für Automatik gute Erfahrungen gemacht wurden, wurde entschieden, dieses Bauteil auch hier einzusetzen. Mit einem Stückpreis von 45.- Franken gehört es zum günstigen Preissegment. Wichtig war auch die Versorgungsspannung von 5 Volt, welche auf dem Versorgungsboard des Modells zur Verfügung steht.

Montage

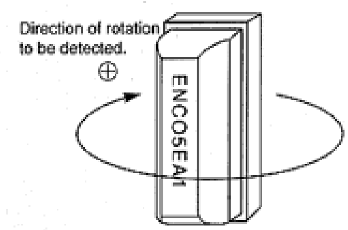


Abbildung 12. **Montagerichtung des Gyros**

Das Gyrometer konnte einfach auf die Versorgungsplatine montiert werden. Da es eine Winkelgeschwindigkeit misst, spielt es keine Rolle, ob die Platine auf dem Kopf steht oder nicht.

Messresultate

Um Herstellerspezifikationen zu überprüfen, welche die Genauigkeit betreffen, wurde versucht, eine Messanordnung aufzubauen. Es wäre sehr aufwändig, die Genauigkeit des Gyros exakt zu bestimmen. Mit einer einfachen Anordnung wurde versucht abzuschätzen, ob das Gyrometer vernünftige Werte liefert. Die Prozessorplatine wurde als Pendel aufgehängt, und mit dem Gyro wurde die Winkelgeschwindigkeit der Pendelbewegung gemessen. In Abhängigkeit der Anfangsauslenkung kann die maximale Winkelgeschwindigkeit der Bewegung berechnet werden.

$$\omega_{max} = \sqrt{\frac{2g(1-\cos\theta_0)}{L}}$$

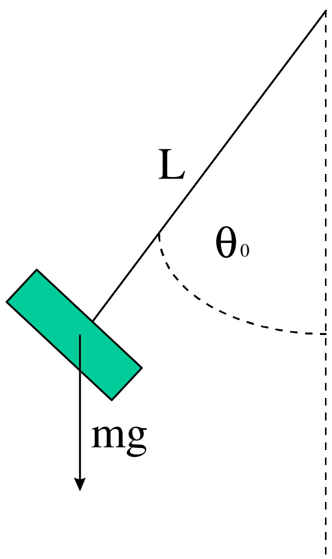


Abbildung 13. **Messanordnung als Pendel**

Zudem sollte die Messkurve ein Sinussignal sein.

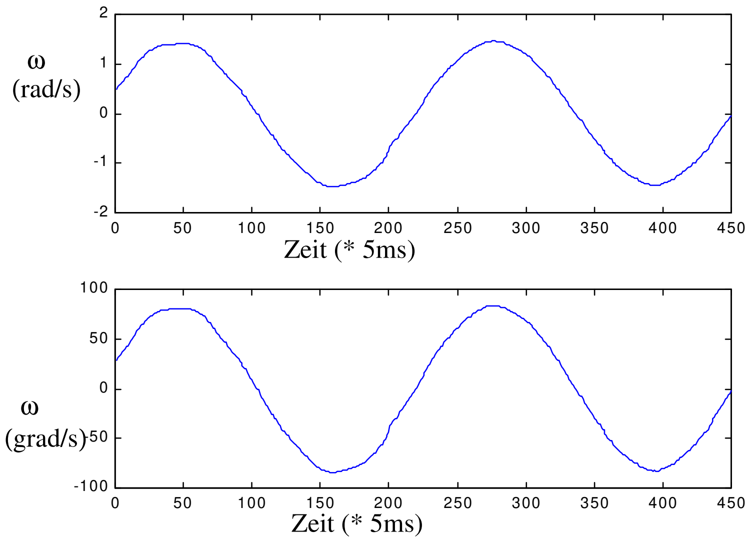


Abbildung 14. Messkurve für 15 Grad Anfangsauslenkung (oben: rad, unten: Grad)

Tabelle 3. Vergleich von Soll- und Messwerten

Anfangsauslenkung(Grad)	gerechneter Wert (rad/s)	Messwert(rad/s)
5	0.471	0.46
10	0.942	0.93
15	1.400	1.46

Der Fehler dieser Messwerte, 4 Prozent, liegt im Bereich der Genauigkeit der Messanordnung. Die Messungen wurden nicht wiederholt, da der Zeitaufwand für eine Messung sehr gross ist.

3.3 Geschwindigkeitssensor

Zur Messung der Geschwindigkeit wurde ein optischer Encoder HEDS-5700 von Hewlett-Packard angeschafft. Der Sensor generiert 512 Pulse pro Umdrehung der Achse und wird mit Hilfe eines Zahnrades an das Hinterrad angehängt.

Abbildung 15. Montage des optischen Encoders von HP

Die Auswertung des Sensorsignals erfolgt über den Kanal 3 der Time Processing Unit (TPU, vgl. Kapitel 7.1 und 8.1.3). Der Anschluss konnte ohne jegliche Zwischenelektronik erfolgen.

3.4 Messung des Lenkwinkels

Der Vollständigkeit halber ist noch auf den letzten Sensor hinzuweisen, das Drehpotentiometer des Lenkservos. Dies ist ein Sensor, dessen Ausgangssignal dem Prozessor nicht zur Verfügung steht. Vom Prozessor kommt ein Stellsignal an den Servo, welcher den Servobügel auf einen bestimmten Winkel regelt. Trotzdem ist das Vorhandensein dieses Sensors wichtig für die Regelung des Systems. Durch diesen Sensor kann man den Kurvenradius des Motorrades bestimmen. Aus der Geschwindigkeit und dem Kurvenradius kann der Neigungswinkel der stabilen Lage berechnet werden (vgl. Kapitel 6).

KAPITEL 4

Aktoren des Systems

4.1 Anforderungen an den Steuerservo

4.1.1 Anforderungen

Die Simulation am mathematischen Modell ergibt, dass die Qualität des Steuerservos ein sehr kritischer Punkt ist (vgl. Abbildung 83 und 84 im Anhang B). Für die Totzeit, das heisst, die Zeit die zwischen dem Befehl zum Drehen und der Bewegung vergeht, wurde ein Maximum von 20 ms bestimmt. Es ergab sich durch eine Betrachtung des Frequenzganges (Bodediagramm) der Regelung, wobei für eine ausreichende Phasen- und Amplitudenreserve eine Polfestlegung bei Omega[1] gleich 8 nötig ist (vgl. Anhang B.6). Dies ist vor allem bei hohen Geschwindigkeiten der Fall und stellt an die Totzeit folgende Bedingung:

$$T_t < \frac{\frac{\pi}{2}}{10 \cdot \Omega} = 20ms$$

Ausserdem darf das gesamte Spiel von Servo und Vorderrad nicht mehr als 1.5 bis 2 Grad betragen. Ist das Spiel grösser, so beginnt das Motorrad zu schwingen.

4.1.2 Funktionsweise eines Servos

Ein Servo besteht aus einem Antriebsmotor, einem Getriebe, einem Potentiometer zur Winkelmessung und einer Elektronik zur Regelung. Damit ist ein Servo eine Kombination zwischen einem Aktor und einem Sensor. Die wesentlichen Qualitätsunterschiede bestehen beim Getriebe (Metall oder Kunststoff), bei der Lagerung der Achse (mit Kugellager oder nicht) und bei der Stärke des Motors. Ein guter Servo hat wenig Spiel, eine feste Drehachse, ein hohes Drehmoment und eine schnelle Zustellzeit. Die Ansteuerung eines Servos erfolgt mit

1. Ω ist die Regelgeschwindigkeit (vgl. Polfestlegung in Kapitel 6)

einem pulsweitenmodulierten Signal, welches normalerweise mit 50 Herz getaktet ist. Die Mittelstellung der Servos liegt bei einem Signal mit etwa 1.5 ms Pulsweite, die Anschläge sind von Modell zu Modell unterschiedlich, liegen aber zwischen 1 und 2 ms.

4.1.3 Übersicht über erhältliche Modelle

TABELLE 4. Übersicht über Servomodelle

Typ	Geschwindigkeit (40 Grad)	Drehmoment (Ncm)	Getriebe	Preis (Fr)
Graupner C512	0.15s	27.5 (4.8V)	plastic	29.-
Volz FS Speed	0.06s	21.5 (4.8V)	plastic	50.-
Volz Mini Star 3	0.10s	29.5 (4.8V)	metall	80.-
Futaba S9404	0.075s (6V)	57 (6V)	plastic/metall	152.10
Futabe S9602	0.06s (6V)	27 (6V)	plastic/metall	125.10
Futaba S9203	0.07s	55 (4.8V)	plastic/metall	159.-
Futaba S9402	0.075s (6V)	80 (6V)	plastic/metall	159.-

4.2 Messungen des Graupner Servos C512

Der Graupner Servo C512 wurde in der Vorgängerarbeit [SA98] für die Lenkung vorgesehen. Es musste gemessen werden, ob seine Qualität den Ansprüchen gerecht wird. Als Servobügel diente der Standardkunstoffbügel.

4.2.1 Messanordnung

Mit Hilfe einer gewachsten Schnur wurde die Drehbewegung des Servos auf ein Drehpotentiometer übertragen. Die Spannungen, die so gemessen wurden, sind linear vom Drehwinkel des Potentiometers (und damit auch des Servos) abhängig.

Abbildung 16. Messanordnung zur Servomessung

Mit der Umschaltung eines Pins wird vom Prozessor gegen aussen gezeigt, wann im Programm der Stellbefehl für den Servo gegeben wurde. Die maximale Verzögerung, welche sich zwischen dem Stellbefehl und der Schaltung des Pins ergibt, ist 270 Clockzyklen (=16 us, vgl. [Mot90]) und ist im Verhältnis zu den Messzeiten vernachlässigbar. Mit einem Speicher-KO wurden die Messkurven aufgezeichnet und danach ausgewertet.

4.2.2 Totzeit und Bewegungsdauer

Die Messkurven befinden sich im Anhang B.1. Die folgende Tabelle gibt eine Übersicht über die Resultate.

TABELLE 5. Messresultate zum Servo C512

Bewegungsgrösse (Grad)	Totzeit (ms)	Stellgeschwindigkeit (Grad/s)
36	50	277
10	55	133
4	50	66
2	100	22

Die Totzeit des Servos beträgt zirka 50ms. Der Zusammenhang zwischen Stellweite und Geschwindigkeit ist in etwa linear.

4.2.3 Spiel

Zur Messung des Spiels (entspricht der Hysterese) musste der Servo zuerst ein Stück gegen Links, dann in die Mitte, ein Stück gegen Rechts und wieder in die Mitte schalten. Durch das Spiel ergibt sich ein Unterschied der Mittelstellung, je nach dem, ob der Servo gerade von Links oder von Rechts kommt.

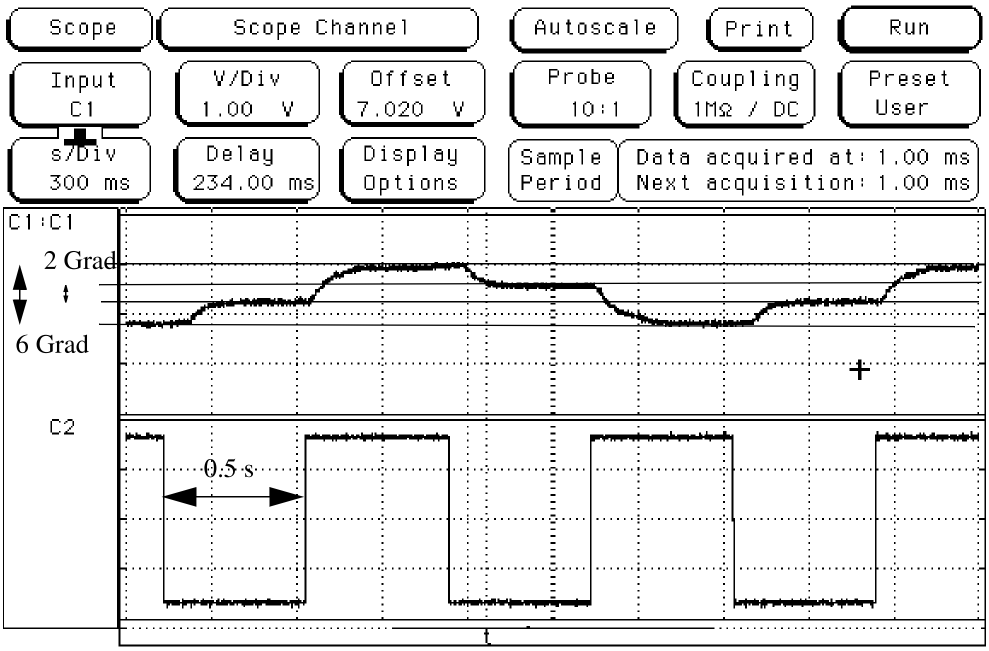


Abbildung 17. Servobewegung zur Spielmessung

Es ergibt sich ein Spiel von zirka 2 Grad. Dies ist das Spiel des Servos. Hinzu kommt noch das Spiel, welches sich durch die Übertragung auf die Vordergabel und das Vorderrad selbst ergibt.

4.3 Versuche zur Verbesserung der Servoleistung

4.3.1 Erhöhung der Taktung des PWM-Signals

Das Pulsweitenmodulierte Signal kann von der Standard 50 Hz Taktung bis auf etwa 150 Hz erhöht werden. Bei höheren Taktungen beginnt der Servo zu schwingen, was zu einer Beschädigung führen könnte. Bei 50 Hz beträgt die maximale Verzögerung, welche einen Teil der Totzeit ausmacht, 20 Millisekunden, bei 150 Hz noch 6.66 ms. Die gemessene Totzeit konnte so auf etwa 40 ms reduziert werden.

4.3.2 Reduzierung des Spiels mit einer Feder

Eine Möglichkeit, das Spiel zu reduzieren, ist das Anbringen einer Feder, welche den Servobügel immer auf eine Seite zieht. Es stellte sich heraus, das der Servo schon bei Federn mit wenig Zug zu schwach war. Deshalb wurden keine weiteren Messungen gemacht.

4.3.3 Überlagerung eines Dithers

Durch die Überlagerung einer Schwingung über das Stellsignal wird der Servo immer zwischen dem einen Ende des Spiels und dem anderen hin und her geschaltet. Dieser Servo ist jedoch zu langsam, um eine genug hohe Frequenz überlagern zu können.

4.4 Messungen des Futaba Servos S9203

4.4.1 Totzeit und Stellgeschwindigkeit

Aufgrund der genannten Messungen und Simulationen wurde ein Futaba Servo besorgt. Er besitzt eine deutlich höhere Stellgeschwindigkeit und ein doppelt so grosses Drehmoment (vgl. Tabelle 4). Da das Getriebe zum Teil aus Metall ist, ein Zahnrad besteht aus Titan, eines aus Aluminium und zwei aus Kunststoff, ist das Spiel ebenfalls kleiner.

TABELLE 6. Messungen des Futaba S9203

Bewegungsgrösse (Grad)	**Totzeit (ms)**	**Stellgeschwindigkeit (Grad/s)**
36	17	514
10	10	166
4	10	80
2	20	16

Der neue Servo ist bei grossen Bewegungen schneller als der Alte, wichtig ist aber vor allem die kürzere Totzeit. Das pulsweitenmodulierte Stellsignal wurde bei diesen Messungen bei 100 Hz festgelegt.

4.4.2 Linearität und Auflösung

Dem Servo wurde ein Stellsignal gegeben, welches ihn schrittweise vom einen Anschlag an den anderen brachte. Die Sprünge des PWM-Signals waren dabei konstant, die Veränderung betrug jeweils 7.604 us, was einem Sprung von zirka einem Grad entspricht.

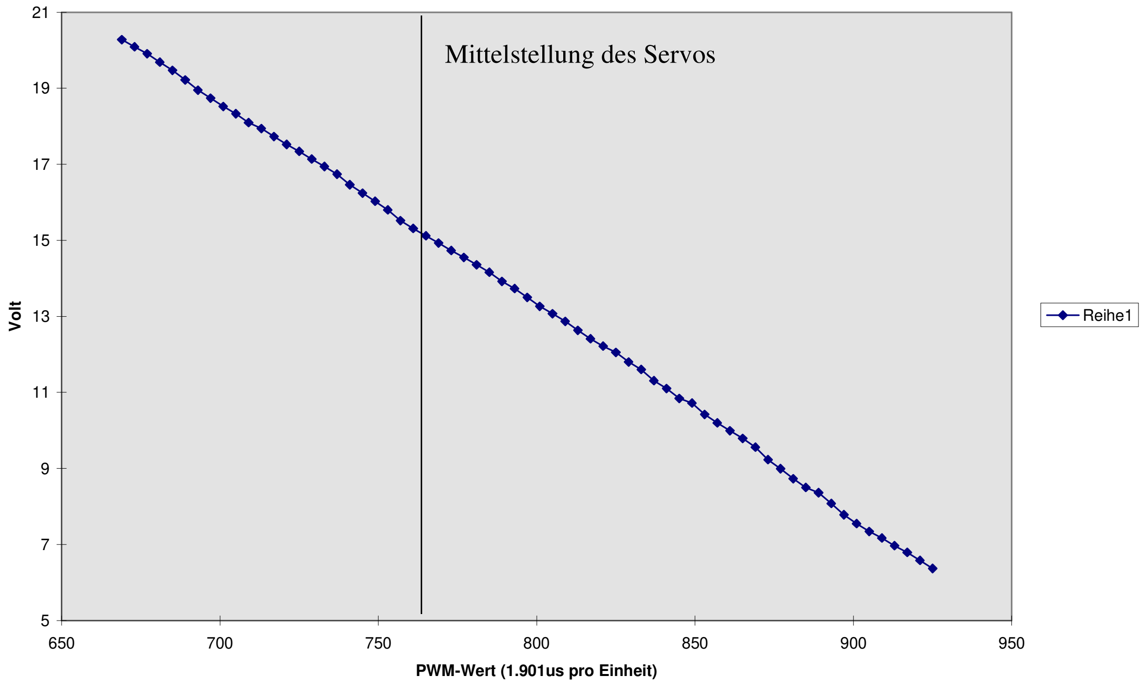


Abbildung 18. Messkurve zur Linearität des Servos

Für die Standardabweichung von der Geraden, welche in diese Messwerte hineingelegt wurde, ergeben sich 0.0912 Volt. Die maximale Abweichung von der Geraden beträgt 0.1905 Volt. Bei einem Verhältnis von 0.2318 Volt pro Grad entspricht das einer Standardabweichung von 0.3934 Grad und einem maximalen Fehler von 0.822 Grad. Man beachte, dass die Absolutgrösse der Messwerte abhängig ist von der Grösse der Spannung, welche an das Potentiometer angelegt ist. Man kann deshalb von den Messwerten nicht auf einen absoluten Winkel schliessen, sondern nur relative Sprünge betrachten.

Zum Bestimmen der Auflösung wurde die gleiche Messung mit den kleinstmöglichen Sprüngen des Stellsignals gemacht. Das PWM-Signal wurde jeweils um 1.901 us geändert, was der Timerbasis der Time Processing Unit (TPU) entspricht. Diese Sprünge entsprechen einer Drehung des Servohebels um 0.25 Grad. Sie Messugen ergaben, dass der Servo diese minimalen Stellsprünge auch umsetzt. Die maximale Abweichung vom gewünschten Wert betrug 0.2092 Grad und die Standardabweichung 0.09 Grad.

Zusammengefasst heisst das, dass der Servo den Anforderungen bezüglich Geschwindigkeit und Genauigkeit gerecht wird. Die Auflösung des Servos rechtfertigt es, die Stellgrössen wirklich genau auszurechnen, da der Servo dies auch umsetzen kann.

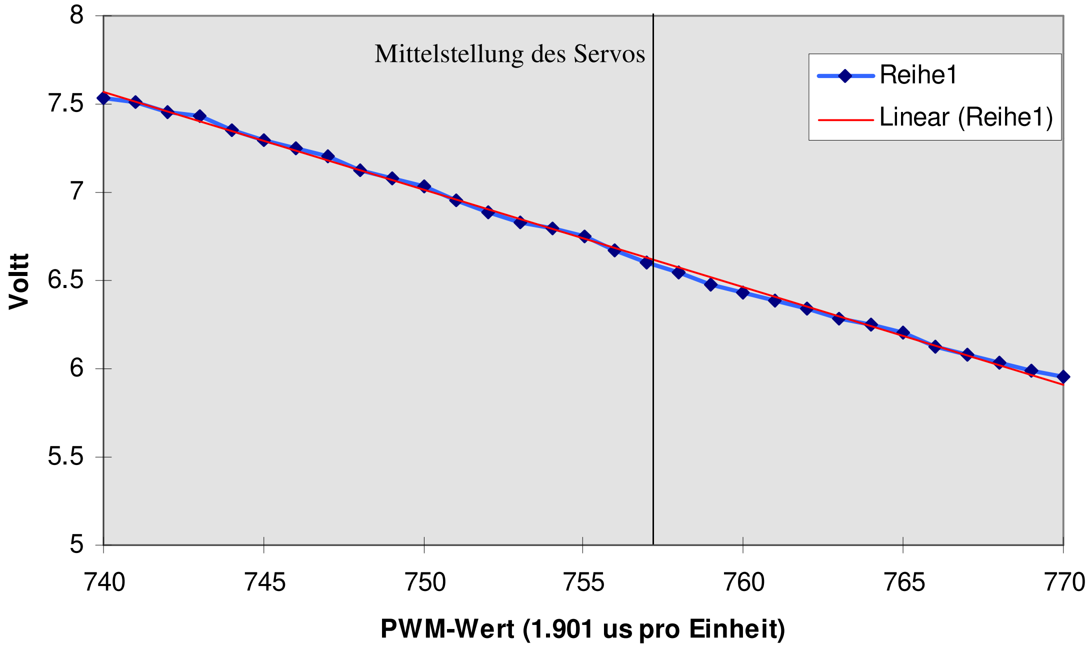


Abbildung 19. Messung zur Auflösung des Servos

4.4.3 Spiel

Um das Spiel zu messen, wurde die gleiche Messanordnung wie für den Graupner Servo verwendet. Die Messkurven finden sich im Anhang "B.1.2 Futaba Servo S9203". Das Spiel beträgt ca. 0.5 Grad.

4.5 Übertragung der Servobewegung auf das Vorderrad

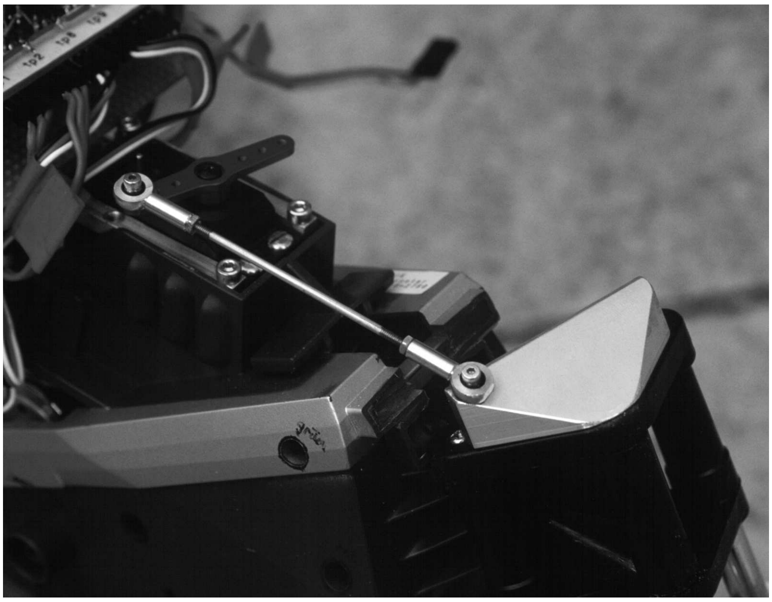


Abbildung 20. **Konstruktion zur Übertragung der Bewegung auf das Vorderrad**

Geometrie

Durch ein Gestänge wird die Lenkbewegung des Servos auf die Vordergabel übertragen. Der Drehpunkt der Vorderachse liegt dabei leider nicht auf der Achse, auf welcher die Servostange befestigt ist. Deshalb dreht sich die Vorderachse nicht um den gleichen Winkel, um den der Servobügel gedreht wird, wie dies in der Vorgängerarbeit [SA98] angenähert wurde. In Abbildung 21 ist der Servo grün und die Drehachsen des Servos und der Vordergabel sind blau dargestellt.

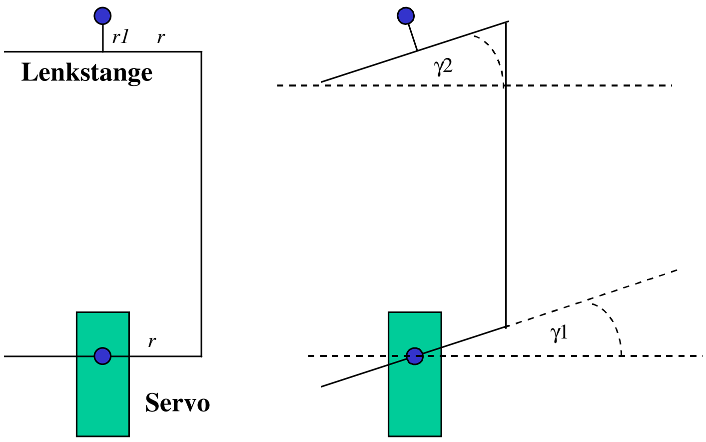


Abbildung 21. **Übertragung der Drehung des Servobügels auf das Vorderrad**

Der Zusammenhang zwischen dem Drehwinkel $\gamma 1$ des Servos und $\gamma 2$ des Vorderrades ist folgender:

$$r \cdot \sin\gamma_1 = r \cdot \sin\gamma_2 + r_1 \cdot \sin\gamma_2 \cdot \tan\gamma_2$$

r: Länge des Servobügels = 18 mm

r1: Abweichung von der Drehachse = 15 mm

Linearisiert ergibt sich folgender Zusammenhang:

$$r \cdot \gamma_1 = r \cdot \gamma_2 + r_1 \cdot \gamma_2^2$$

TABELLE 7. Vergleich von Servo- und Lenkwinkel

Servowinkel (Grad)	Lenkwinkel (Grad)	Verhältnis
-15	-22	1.466
-10	-12.15	1.215
-5	-5.43	1.086
-1	-1.015	1.0115
1	0.986	0.986
5	4.68	0.936
10	8.86	0.886
15	12.66	0.844
20	16.18	0.809
25	19.48	0.7792
30	22.58	0.753

Um den gewünschten Lenkwinkel $\gamma 2$ zu erhalten, muss der Servowinkel $\gamma 1$ folgendermassen angepasst werden (genähert):

$$\gamma_1 = \gamma_2 + \frac{(\gamma_2)^2}{3}$$

Mechanik

Um das Spiel, welches sich durch die Übertragung vom Servo auf das Vorderrad ergibt, möglichst klein zu halten, wurden zwei Änderungen an der Mechanik gemacht. Der Servobügel, zuvor aus Kunststoff, wurde durch ein Aluminiumteil ersetzt. Bei den Kräften, die während der Fahrt auftreten können, sollte sich dieser Bügel nicht biegen und deshalb nicht zum Spiel beitragen. Ebenfalls verbessert wurde die Befestigung der Servostange am Bügel. Es wurde ein Kugelgelenklager angeschraubt, welches ebenfalls weitgehend spielfrei ist.

4.6 Erweiterung des Lenkwinkels

Zu Beginn der Arbeit war der Lenkwinkel, bedingt durch die Mechanik des Motorrades, auf 18 Grad beschränkt. Bei einer Kurvenfahrt, gerade bei langsamen Geschwindigkeiten, muss eine Reserve am Lenkwinkel zur Stabilisation übrigbleiben. Nimmt man 5 Grad Reserve an, so ergibt sich ein minimaler Kurvenradius von 1.25 Meter. Den Zusammenhang zwischen Lenkwinkel und Kurvenradius entnehme man Kapitel 5 “Modellierung”. Da ein Ziel das Fahren auf engstem Raum in einem Hörsaal war, musste der Kurvenradius noch verkleinert werden. Durch mechanische Anpassungen am Motorrad konnte der maximale Lenkwinkel auf 32 Grad erweitert werden, was bei gleicher Reserve einen Kurvenradius von etwa 60 cm zulässt.

KAPITEL 5

Modellierung

Für die Berechnung und Simulation der Motorraddynamik, musste ein mathematisches Modell, welches das mechanische Verhalten des Motorades möglichst gut beschreibt, aufgestellt werden. Die Grundlagen dieser Modellierung wurden bereits in früheren Arbeiten [SA98] [GET94] zusammengetragen. In dieser Arbeit ging es darum, den richtigen Mittelweg zwischen Genauigkeit und Einfachheit des Modells zu finden. In den späteren Testfahrten hat sich gezeigt, dass das Modell recht gut mit der Wirklichkeit übereinstimmt. Es folgt als erstes eine Liste der Konstanten und Variablen, die im weiteren benutzt werden.

- α: Neigungswinkel (positiv gegen Rechts)
- γ: Lenkwinkel (positiv gegen Rechts)
- h: Höhe des Schwerpunktes
- l: Abstand der Auflagepunkte der Räder
- m: Masse des Motorrades
- g: Normwert der Fallbeschleunigung
- Lv, Lh: Drall des Vorder- und Hinterrades

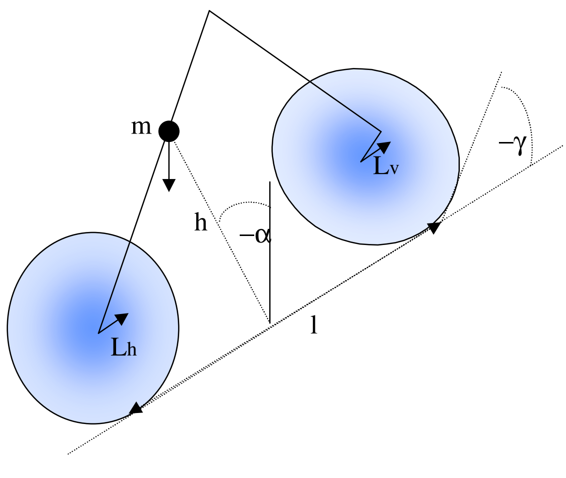


Abbildung 22. **Parameter der mathematischen Modellierung**

- *L*: Gesamtdrall
- *Jv,Jh*: Trägheitsmoment des Vorder- und Hinterrades
- *Rk*: Radius der Kreisfahrt
- *v*: Geschwindigkeit
- *J*: Trägheitsmoment des Motorrades um Achse durch Auflagepunkte
- *Mg*: Drehmoment der Gewichtskraft
- *Mz*: Drehmoment der Fliehkraft
- *Mk*: Drehmoment durch Kurvenfahrt (Dralländerung)
- *Mv*: Drehmoment durch Lenkbewegung (Dralländerung)

5.1 Mathematisches Modell des Motorrades

Wie schon in der Arbeit [SA98] vorgeschlagen, wird das Motorrad als ein bewegtes, invertiertes Pendel modelliert. Dieses kann sich, wie das Motorrad auch, nur in einer Ebene seitlich neigen. Um die Differentialgleichung, welche die Dynamik dieses Pendels beschreibt, herzuleiten, werden die Drehmomente, welche auf das Pendel wirken, berechnet.

5.1.1 Gewichtskraft

Das Pendel wird als Massepunkt angenommen, auf den die Gewichtskraft *m*g* wirkt. Das Drehmoment, welches eine Drehbewegung um die Achse durch die Auflagepunkte der Räder bewirkt, ist:

$$M_g = mgh \cdot \sin\alpha$$

5.1.2 Zentrifugalkraft

Die Stärke der Fliehkräfte, welche auf das Motorrad wirken, hängt vom Kurvenradius der Bewegung des Schwerpunktes ab. Die Kurvenkrümmung der Trajektorien, auf denen sich die Räder bewegen, ist aus Abbildung 23 ersichtlich.

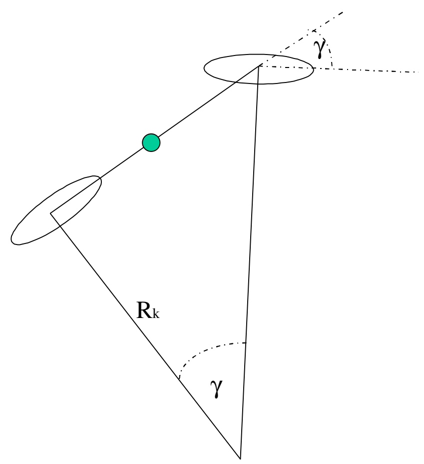


Abbildung 23. Herleitung des Kurvenradius

Zum Berechnen der Kurvenradien betrachtet man das Dreieck, welches von den Auflagepunkten der Räder und vom Zentrum der Kurvenfahrt gebildet wird. Der Kurvenradius, auf dem das Hinterrad fährt, beträgt:

$$r_k = \frac{l}{\tan\gamma}$$

Das Hinterrad fährt einen leicht engeren Radius als das Vorderrad, dessen Kurvenradius sich so berechnet:

$$r_k = \frac{l}{\sin\gamma}$$

Für die Trajektorie des Schwerpunktes, welcher zwischen Vorder- und Hinterrad liegt, ergibt sich durch seine Kippgeschwindigkeit eine Abweichung von der Kurvenkrümmung der Radtrajektorie. Das heisst, der Schwerpunkt fliegt aus der Kurve hinaus oder in die Kurve hinein.

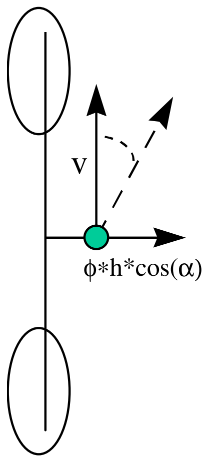


Abbildung 24. **Abweichung der Bewegungsrichtung des Schwerpunktes**

Die Bewegungsrichtung des Schwerpunktes weicht damit von jener der Räder ab. Für die Krümmung der Trajektorie ergibt sich damit folgende Gleichung:

$$\frac{\sin\left(\gamma + \frac{\dot{\alpha} \cdot h}{v}\cos\alpha\right)}{l} < k < \frac{\tan\left(\gamma + \frac{\dot{\alpha} \cdot h}{v}\cos\alpha\right)}{l}$$

Der Schwerpunkt liegt zudem noch abhängig von der Neigung neben der Achse durch die Auflagepunkte der Räder. Da sich das Motorrad in die Kurve hineinlegt, ist der Kurvenradius der Bewegung des Schwerpunktes kleiner als jener der Räder.

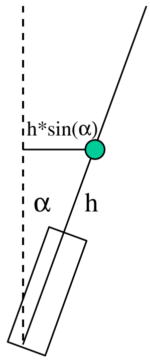


Abbildung 25. Einfluss der Neigung auf den Kurvenradius

Für den Kurvenradius der Bewegung des Schwerpunktes ergibt sich damit folgende Formel:

$$\left[\frac{l}{\tan\left(\gamma + \frac{\dot{\alpha} \cdot h}{v} \cos\alpha\right)} - h \cdot \sin\alpha\right] < r_k < \left[\frac{l}{\sin\left(\gamma + \frac{\dot{\alpha} \cdot h}{v} \cos\alpha\right)} - h \cdot \sin\alpha\right]$$

Da der Schwerpunkt näher beim Hinterrad als beim Vorderrad liegt, wurde im weiteren angenommen, dass er direkt über dem Hinterrad liegt. Was nun interessiert, ist die Zentrifugalkraft, welche auf das Motorrad wirkt:

$$F_z = -\frac{mv^2}{r_k} = -mv^2\left(\frac{\tan\left(\gamma + \dot{a}\frac{h}{v} \cdot \cos\alpha\right)}{l - h\tan\left(\gamma + \dot{a}\frac{h}{v} \cdot \cos\alpha\right)\sin\alpha}\right)$$

Das zugehörige Drehmoment um die Achse in Fahrtrichtung ergibt sich zu:

$$M_z = F_z \cdot h \cdot \cos\alpha = \frac{mv^2}{r_k} h \cdot \cos\alpha$$

5.1.3 Drallvektoren (Corioliskräfte)

Wird die Achse eines Rades oder eines Kreisels gedreht, entsteht ein Drehmoment, welches proportional zur Dralländerung ist. In unserem Fall werden bei einer Kurvenfahrt die Achsen beider Räder mit der Winkelgeschwindigkeit dieser Kurve gedreht. Das Vorderrad wird zudem bei einer Änderung des Lenkwinkels Gamma gedreht, was ebenfalls ein Drehmoment hervorruft.

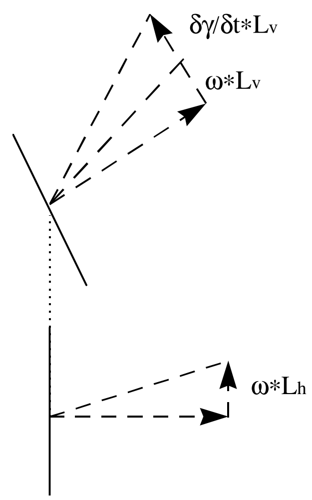


Abbildung 26. Drallvektoren

Das Drehmoment, welches sich durch die konstante Kurvenfahrt ergibt, ist:

$$M_k = |\underline{\dot{L}}| = |\omega_k \times \underline{L}| = \frac{v}{r_k}(L_h + L_v \cos\gamma)\cos\alpha$$

Jenes Drehmoment, welches sich durch die Änderung des Lenkwinkels ergibt, ist:

$$M_v = \dot{\gamma} L_v \cos(\alpha)\cos(\gamma)$$

Der Drall der Räder berechnet sich aus dem Trägheitsmoment und der Winkelgeschwindigkeit:

$$L_v = J_v \frac{v}{r_v}$$

$$L_h = J_h \frac{v}{r_h}$$

Der Gesamtdrall L der Räder ergibt sich zu:

$$L = L_v \cdot \cos\gamma + L_h$$

5.1.4 Differentialgleichung

Die Differentialgleichung setzt sich zusammen aus den Drehmomenten, die in den vorherigen Abschnitten berechnet wurden. J ist dabei das Trägheitsmoment des Motorrades.

$$J\ddot{\alpha} = M_g + M_z + M_k + M_v$$

$$J\ddot{\alpha} = mgh \cdot \sin\alpha - \frac{mv^2 h\cos\alpha}{r_k} + \frac{v(L_h + L_v\cos\gamma)\cos\alpha}{r_k} + L_v\dot{\gamma}\cos(\alpha)\cos(\gamma)$$

Der stationäre Betriebszustand ist in senkrechter Lage und bei kleinen Lenkwinkeln. Die Linearisierung um $\alpha=0$, $\gamma=0$ ergibt folgende, für unser Modell brauchbare Differetialgleichung:

$$J\ddot{\alpha} = mgh\alpha - \left(\frac{mvh^2}{l} - \frac{hL}{l}\right) \cdot \dot{\alpha} - \left(\frac{mv^2h}{l} - \frac{vL}{l}\right)\gamma + L_v\dot{\gamma}$$

Aufgeteilt in zwei Differentialgleichungen mit $x1=\alpha$ und $x2=d\alpha/dt$ ergibt sich folgendes Differentialgleichungssystem:

$$T \cdot \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & -d_1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \cdot u$$

Die Parameter setzen sich wie folgt zusammen:

$$T = \sqrt{\frac{J}{mgh}}$$

$$b_1 = \frac{1}{T} \cdot \frac{L_v}{mgh}$$

$$b_2 = -\frac{v}{gl}\left(v - \frac{L}{mh}\right)$$

$$d_1 = \frac{1}{T}\left(\frac{vh}{gl} - \frac{L}{mgl}\right)$$

5.1.5 Übertragungsfunktion und Pole

Ebenfalls mit den Parametern aus Kapitel 5.1.4 wurde die Übertragungsfunktion der Strecke dargestellt. Diese entspricht der laplacetransfomierten Differentialgleichung des Systems.

$$G(s) = \frac{b_1 s + b_2}{T^2 s^2 + T d_1 s - 1}$$

Die Strecke besitzt zwei Pole, einen in der stabilen und einen in der instabilen Hälfte der Gaussebene.

$$p_{1,2} = \frac{-d_1 \pm \sqrt{d_1^2 + 4}}{2T}$$

Das heisst, das System ist selbst nicht stabil und fällt ohne einen Regler um. Für steigende Geschwindigkeit geht der Pol in der rechten Halbebene gegen 0. Das Motorrad wird also bei grossen Geschwindigkeiten stabiler, aber es stabilisiert sich nicht von selbst.

5.1.6 Erweiterung des Modells um den Nachlauf des Vorderrades

Der Modellbautöff hat eine besondere Konstruktion der Vordergabel, den Nachlauf. Das heisst, dass der Auflagepunkt des Vorderrades nicht mit der Drehachse der Gabel übereinstimmt. Dies bewirkt, dass das Motorrad bei einer Lenkbewegung seitlich verschoben wird.

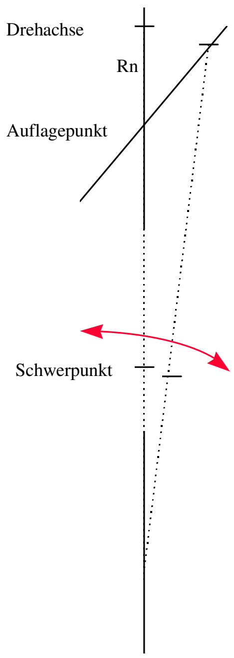


Abbildung 27. **Auswirkung des Nachlaufes bei einer Lenkbewegung**

Für die seitliche Verschiebung der Drehachse ergibt sich folgende Gleichung:

$$s_n = r_n \cdot \sin\gamma \approx r_n \cdot \gamma$$

Der Schwerpunkt liegt bei etwa einem Drittel der Länge (vom Hinterad gesehen), und deshalb kann für seine Verschiebung etwa ein Drittel der obigen Berechnung angenommen werden. Die seitliche Verschiebung hat auch einen Einfluss auf den Neigungswinkel des Motorrades, wie die folgende Skizze illustriert.

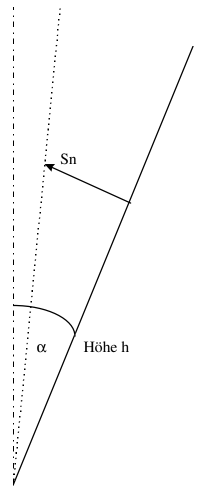


Abbildung 28. Einfluss der Verschiebung auf die Neigung

Der Nachlauf des Vorderades hat auf die Winkelgeschwindigkeit und damit auf die Messwerte des Gyrometers einen Einfluss.

$$\dot{\alpha} = \frac{d}{dt}\frac{r_n \cdot \sin\gamma}{3 \cdot h} \approx \frac{r_n \cdot \dot{\gamma}}{3 \cdot h}$$

Im Simulinkmodell (vgl. “5.2 Simulinkmodell des Motorrades”) wird dieser Einfluss mit *b3* bezeichnet, *Rn* ist der Radius des Nachlaufs.

$$b_3 = \frac{r_n}{3 \cdot h}$$

5.2 Simulinkmodell des Motorrades

Das Simulinkmodell ist eine schematische Darstellung der Differentialgleichungen in Kapitel 5.1.4, erweitert um die Modellierung des Nachlaufs.

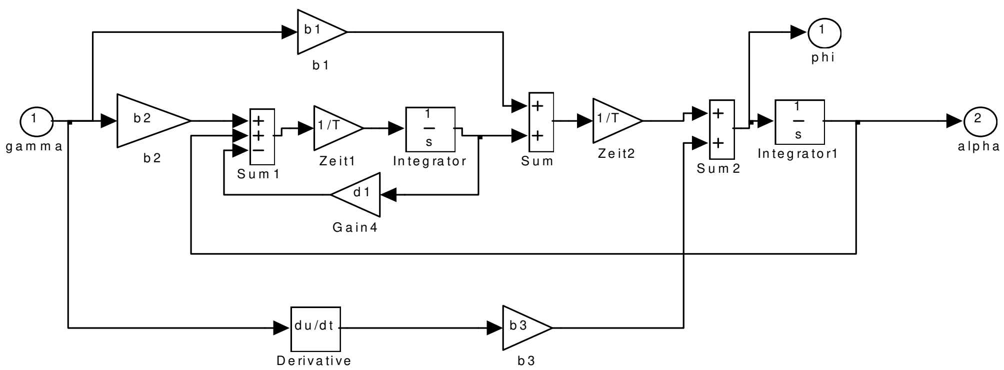


Abbildung 29. Simulinkmodell des Motorrades

Eingangsgrösse des Modells ist der Lenkwinkel γ und ausgegeben wird die Rollgeschwindigkeit ϕ und der Neigungswinkel α.

5.3 Modell des Servos

Für die Modellierung des Servos müssen folgende Eigenschaften aus Kapitel 4 einbezogen werden:

- Totzeit
- Anschläge
- Stellgeschwindigkeit
- Spiel

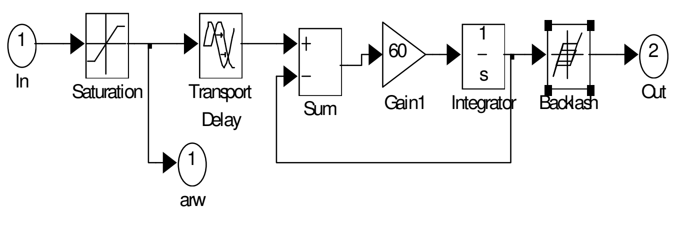


Abbildung 30. Simulinkmodell des Servos

Die Anschläge werden durch eine Sättigung modelliert, die Totzeit durch eine Transportverzögerung, die Stellgeschwindigkeit durch ein Glied erster Ordnung und das Spiel durch eine Hysterese.

5.4 Modell des Gyrometers

Das Gyrometer liefert eine Spannung, welche Proportional zur Winkelgeschwindigkeit des Motorrades ist. Einzig wenn die Winkelgeschwindigkeiten zu hoch werden, das heisst grösser als 90 Grad pro Sekunde, kommt es an seine Anschläge. Das Gyro ist deshalb als eine Kombination von Verstärkungs- und Sättigungsteil modelliert.

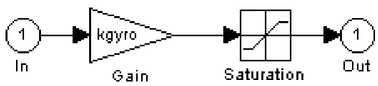


Abbildung 31. Simulinkmodell des Gyros

KAPITEL 6

Regelung

Im folgenden Kapitel werden, sofern nicht anders angegeben, alle Winkel in Radian und alle Geschwindigkeiten in Meter pro Sekunde gemessen. Nun folgt die Liste der verwendeten Parameter:

- *kp0*: Verstärkungsfaktor der inneren PI-Regelung
- *Ti0*: Nachstellzeit der inneren PI-Regelung
- *Ts*: Abtastzeit der diskreten Regelung
- *kp1*: Verstärkungsfaktor der äusseren PI-Regelung
- *Ti1*: Nachstellzeit der äusseren PI-Regelung
- *vf*: Parameter zur Berechnung der Neigung aus dem Lenkwinkel
- Ω: Regelgeschwindigkeit
- *D*: Dämpfung

6.1 Minimales Regelkonzept

Das Motorrad sollte zuerst mit einem minimalen Regelungskonzept in Betrieb genommen werden. Eine PI-Regelung auf die Rollgeschwindigkeit, das heisst die Winkelgeschwindigkeit des seitlichen Umfallens, ist eine minimale Regelung. Es wird nur ein Sensor, das Gyrometer, benötigt. Die Regelung besitzt einen relativ geringen Rechenaufwand.

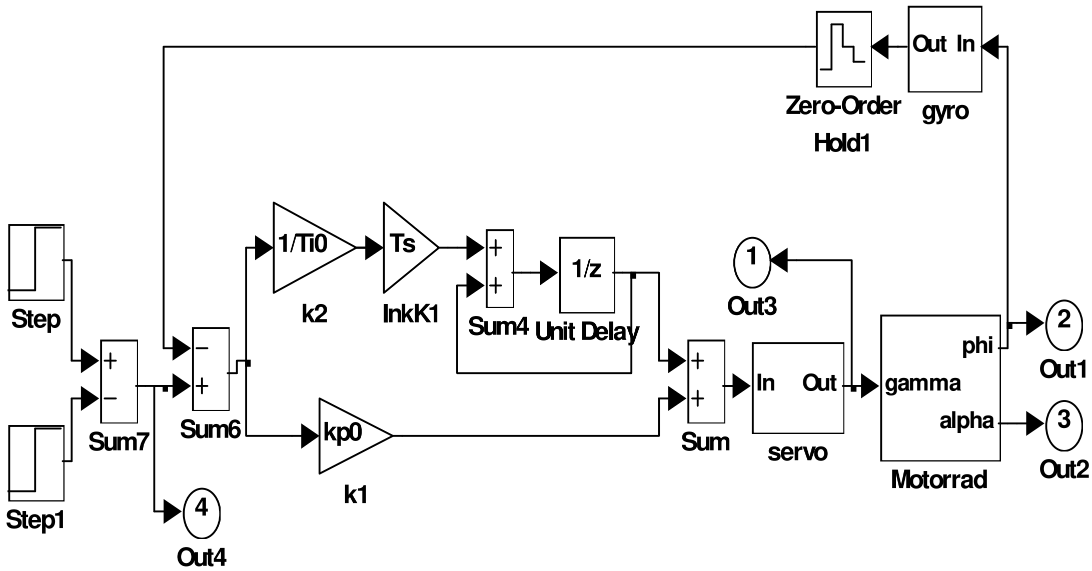


Abbildung 32. PI-Regelung auf die Rollgeschwindigkeit

6.1.1 Auslegung und Simulation

Die Differentialgleichung für die Winkelgeschwindigkeit wurde im vorherigen Kapitel berechnet. Für kleine Winkel kann der Einfluss der Neigung vernachlässigt werden, und man erhält die Übertragungsfunktion vom Lenkwinkel auf die Rollgeschwindigkeit:

$$G_u(s) = \frac{b_2}{sT - d_1}$$

Mit der PI-Regelung ergibt sich das Charakteristische Polynom der Strecke:

$$chp = s^2 + s \cdot \frac{d_1 + b_2 k_p}{T} + \frac{b_2}{T \cdot T_I}$$

Die Polfestlegung brachte folgende Regelungsparameter:

$$T_{I0} = \frac{b_2}{T\Omega^2}$$

$$k_{p0} = \frac{2\Omega DT - d_1}{b2}$$

Die Variablen b2, d1 und T sind in Kapitel 5 zu finden und sind abhängig von der Geschwindigkeit des Motorrades. Am Anfang wurde die Regelung nur für eine Geschwindigkeit von 3 m/s ausgelegt, das heisst, die Regelparameter wurden konstant gelassen. Mit dem Entwurfsparameter Omega gleich 5.5 ergaben sich folgende Werte für den Proportional- und den Integralanteil (vgl. Abbildung 32):

- Ti0= -0.9625
- kp0= -0.1156

Die folgenden Simulationsergebnisse wurden mit Matlab/Simulink für die obigen Regelungsparameter erreicht:

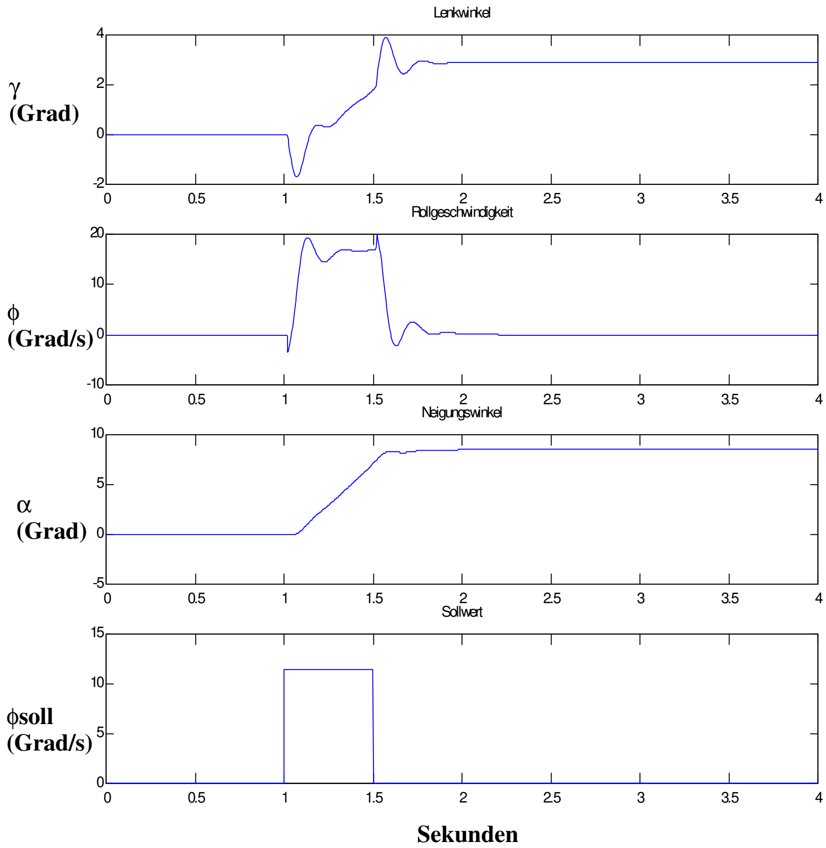


Abbildung 33. Simulation mit PI-Regelung (Winkel in Grad)

Von der Fernsteuerung kommt dabei der Sollwert für die Rollgeschwindigkeit, das heisst, einer konstanten Stellung des Knüpels an der Fernsteuerung entspricht eine konstante Winkelgeschwindigkeit auf die Seite. Bei der Simulation fällt auf, dass eine Rollgeschwindigkeit auf eine Seite durch eine Lenkbewegung auf die andere Seite ausgelöst wird.

6.1.2 Ergebnisse

Die Implementation dieser Regelung brachte das Motorrad zum ersten Mal zum laufen. Damit konnte gezeigt werden, dass man mit dem mathematischen Modell nicht weit von der Realität entfernt war, denn die berechneten Regelgrössen funktionierten. Es musste jedoch ein enger Geschwindigkeitsbereich eingehalten werden. Wurde das Motorrad zu schnell, so begann es zu schwingen, war es zu langsam, so fiel es um. In der Kurve bestand die Tendenz, weiter in diese Kurve hineinzufallen.

6.2 Erweiterung durch Gain Scheduling

Als Gain Scheduling bezeichnet man die Auswahl verschiedener Regelungsalgorithmen aufgrund von Arbeitsbedingungen. In diesem Projekt entsprechen diesen unterschiedlichen Bedingungen die verschiedenen Geschwindigkeiten.

6.2.1 Auslegung und Simulation

Die Regelungsparameter wurden für verschiedene feste Geschwindigkeiten *v* berechnet. Das heisst, man teilt das Geschwindigkeitsspektrum in verschiedene Arbeitsbereiche auf, und für jeden Bereich wird ein fester Regler verwendet. Dabei ist darauf zu achten, dass die Übergänge zwischen den Bereichen möglichst störungsfrei erfolgen. Die so erhaltenen Werte wurden durch eine Funktion in v angenähert, was einen kontinuierlichen Übergang zwischen den Zonen ermöglicht (Tabelle 8).

TABELLE 8. Polfestlegung für v>2.3 m/s

v	omega	D	kp0	Ti0	kp0=-1/v^2	Ti0=-0.3*v
2	4	1	-0.2980	-0.6684	-0.2500	-0.6
2.5	4.5	1	-0.1786	-0.8252	-0.1600	-0.75
3	5	1	-0.1156	-0.9625	-0.1111	-0.9
3.5	5.5	1	-0.0787	-1.0827	-0.0816	-1.05
4	6	1	-0.0555	-1.1883	-0.0625	-1.2
4.5	6.5	1	-0.0401	-1.2814	-0.0494	-1.35
5	7	1	-0.0294	-1.3641	-0.0400	-1.5

In einem späteren Schritt wurde das Gain Scheduling für kleine Geschwindigkeiten noch verfeinert (Tabelle 9 und Tabelle 10). Der Übergang, zwischen der Funktion für grössere und jener für kleinere Geschwindigkeiten erfolgt durch Fallunterscheidung, wobei darauf geachtet wurde, dass die Sprünge bei diesen Übergängen nicht zu gross wurden.

TABELLE 9. Polfestlegung für v>1.4m/s

v	omega	D	kp0	Ti0	kp0=-0.6253/v^2	Ti0=-0.1671*v^2
1.4	4	0.6	-0.2731	-0.3275	-0.3190	-0.3275
1.6	4	0.7	-0.2557	-0.4278	-0.2443	-0.4277
1.8	4	0.75	-0.1926	-0.5414	-0.1930	-0.5414
2	4	0.8	-0.1484	-0.6684	-0.1563	-0.6684
2.2	4	0.85	-0.1164	-0.8088	-0.1292	-0.8090

TABELLE 10. Polfestlegung für v>1.1m/s

v	omega	D	kp0	Ti0	kp0=-0.4757/v^2	Ti0=-0.1671*v^2
1.1	4	0.49	-0.3935	-0.2022	-0.3931	-0.2022
1.2	4	0.52	-0.3305	-0.2406	-0.3303	-0.2406
1.3	4	0.55	-0.2815	-0.2824	-0.2815	-0.2824
1.4	4	0.6	-0.2731	-0.3275	-0.2427	-0.3275

6.2.2 Ergebnisse

Durch das Gain Scheduling wurde das Motorrad in einem grösseren Geschwindigkeitsbereich fahrbar. Die Verfeinerung für kleine Geschwindigkeiten wurde nötig, weil dort das Motorrad zu schütteln begann. Dieser Effekt entsteht durch den Nachlauf des Vorderrades (vgl. Kapitel 5), welcher zur Folge hat, dass durch die Lenkbewegungen das Motorrad seitlich verschoben wird. Ebenfalls störend wirkt dort das Messrauschen, weil die Verstärkungen bei kleinen Geschwindigkeiten immer grösser werden.

6.3 Kaskadenregelung

Um die Regelung zu verbessern, kam die Idee einer Kaskadenregelung. Das heisst, um die Regelung auf die Rollgeschwindigkeit wurde noch eine Reglung auf den Neigungswinkel gelegt. Wenn das Motorrad in einer stabilen Lage ist, wie sie bei der obigen PI-Regelung eingenommen wird, so zeigt die Resultierende von Schwer- und Fliehkraft vom Schwerpunkt durch die Auflagepunkte der Räder.

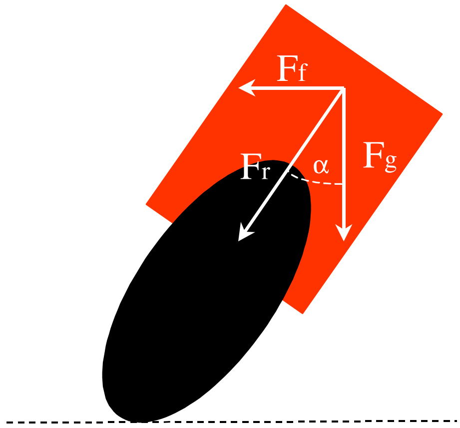


Abbildung 34. Kräfte bei stabiler Kurvenfahrt

Aus dem Lenkwinkel und der Geschwindigkeit kann man aber die Fliehkraft berechnen, und die Grösse der Schwerkraft ist bekannt, so dass man den Neigungswinkel des Motorrades berechnen kann. Das heisst, man benutzt den Geschwindigkeitssensor und das Potentiometer des Servos als Sensoren des äusseren Regelkreises und kann deshalb auf einen weiteren Sensor verzichten, was der Einfachheit des Systems dient. Für den Neigungswinkel gilt (vgl. Abbildung 34):

$$\alpha = \mathrm{atan}\left(\frac{F_f}{F_g}\right) \approx \frac{v^2 \gamma}{gl}$$

Diese Berechnung funktioniert nur für die stabile Lage. Die Rollgeschwindigkeit muss daher ausreichend klein sein. Das bedeutet, dass die Regelgeschwindigkeit des äusseren Regelkreises deutlich kleiner sein muss als jene des inneren. Der Lenkwinkel, der zur Berechnung des Neigungswinkels benutzt wird, wird ausserdem gefiltert, indem man den Mittelwert der letzten 20 Lenkwinkel nimmt.

6.3.1 Auslegung

Für die Auslegung der Regelung wurde der obige Regelkreis als abgeschlossenes System betrachtet. Es wirkt wie ein Integrator mit der Winkelgeschwindigkeit am Eingang und dem Winkel am Ausgang. Die Polfestlegung bringt:

$$k_{p1} = 2 \cdot \Omega$$

$$T_{I1} = \frac{1}{\Omega^2}$$

Der Parameter *vf* dient zur Berechnung des Neigungswinkels wie oben:

$$v_f = \frac{v^2}{gl}$$

Das Schema der Regelung sieht so aus:

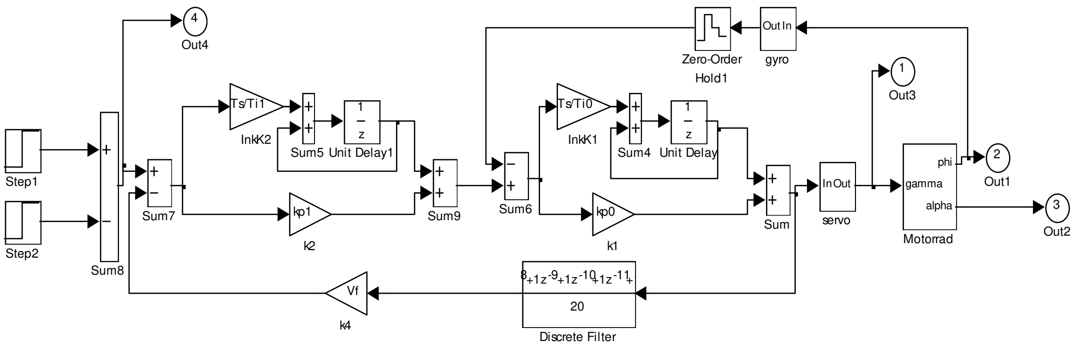


Abbildung 35. Simulinkmodell der Kaskadenregelung

6.3.2 Simulation

Der Eingangswert der Regelung ist ein Sollwert für die Neigung. Eine bestimmte Neigung entspricht bei einer gewissen Geschwindigkeit auch immer einem Kurvenradius *rk*:

$$r_k = \frac{v^2}{g \cdot \alpha}$$

Die Neigung als Sollwert der Regelung zu nehmen ist deshalb ein sinnvoller Ansatz.

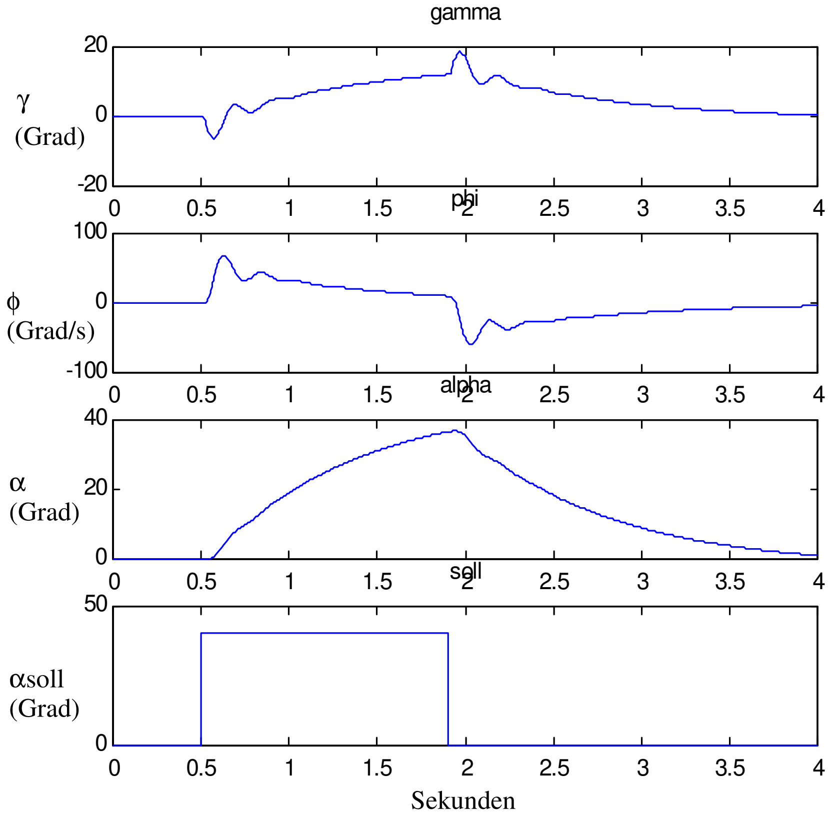


Abbildung 36. Simulation einer Kurvenfahrt (Winkel in Grad)

Die folgende Abbildung zeigt eine 3D-Simulation der gleichen Kurvenfahrt wie oben. Um sich in die Kurve zu legen, wird das Vorderrad zuerst auf die Gegenseite gelenkt. Um sich nach der Kurvenfahrt wieder aufzurichten, wird noch mehr in die Kurve hineingelenkt.

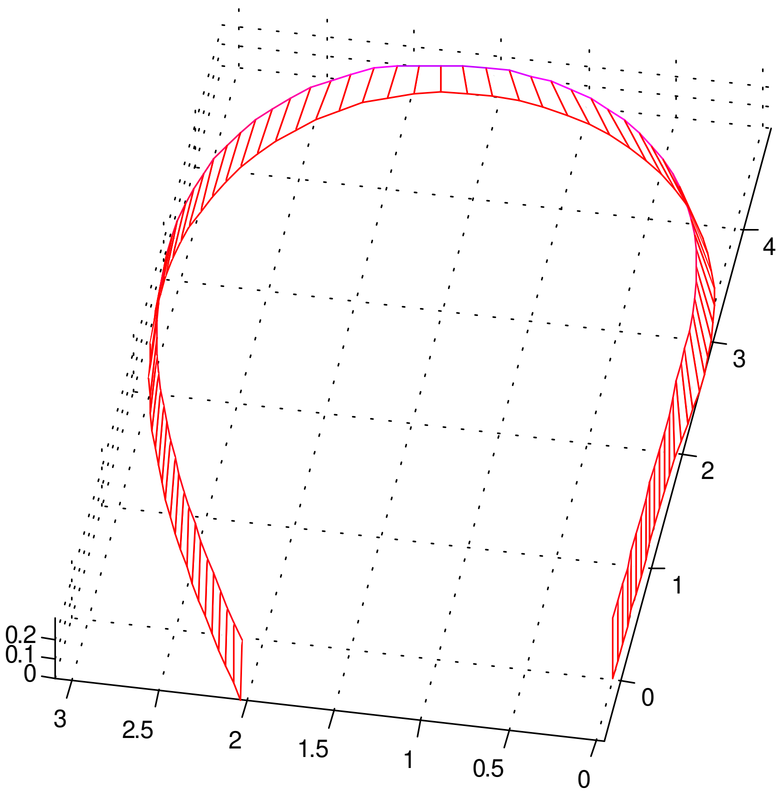


Abbildung 37. 3D-Simulation der Kurvenfahrt

Weitere Simulationen findet man im Anhang B.

6.3.3 Ergebnisse

Dieser Regler brachte sehr gute Fahreigenschaften, und das Motorrad lässt sich gut lenken. Es fährt für Geschwindigkeiten über 1.4 m/s stabil, darunter beginnt es zu schwingen und fällt bei noch kleinerer Geschwindigkeit um. Bei langsamen Fahrten beträgt der Kurvenradius ca. 1 Meter, bei schnelleren wird die Haftung auf dem Boden zum Problem. Die Neigung wurde für schnelle Fahrten auf 40 Grad beschränkt. Lässt man den Steuerknüppel in einer konstanten Stellung, so fährt das Motorrad immer den gleichen Kurvenradius.

Wichtiges Instrument zum Analysieren und Verbessern der Regelung ist die Möglichkeit zum Abspeichern und Hochladen der Daten auf den PC. Die Messkurven, welche auf diese Art entstanden sind, finden sich im Anhang B, und die Methode zum Aufzeichnen solcher Messkurven wird im Softwareteil in Kapitel 8 beschrieben.

6.4 Weitere Regelkonzepte

6.4.1 Zustandsregelung

Für eine Zustandsregelung bräuchte es einen zusätzlichen Sensor, welcher die absolute Neigung misst. Das Inklinometer ist für diese Messung ungeeignet, eine Laserdistanzmessung kommt in Frage.

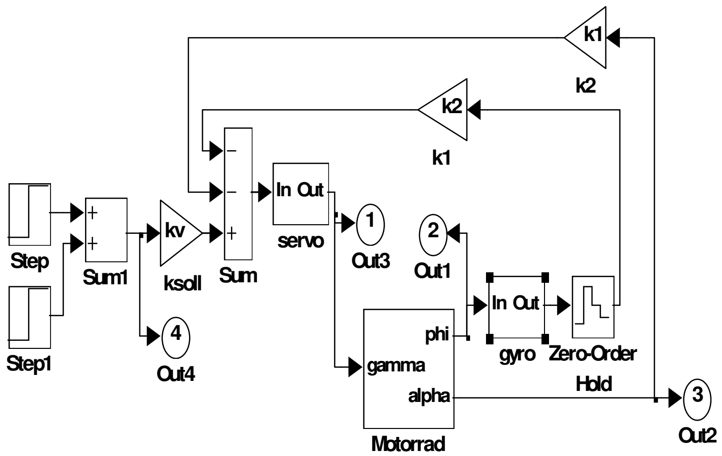


Abbildung 38. **Schema der Zustandsrückführung**

Das charakteristische Polynom des Systems sieht so aus:

$$s^2 + s\left(\frac{d_1 + b_2 k_2 + b_1 k_1}{T}\right) + \frac{d_1 b_1 k_1 + b_2 k_1 + b_1 k_2 - 1}{T^2} = 0$$

Die Polfestlegung bringt folgende Werte für k1 und k2:

$$k1 = \frac{\Omega^2 T^2 - 2\Omega T \frac{b_1}{b_2} + d_1 \frac{b_1}{b_2} + 1}{d_1 b_1 + b_2 - \frac{b_1^2}{b_2}}$$

$$k_2 = \frac{2\Omega T}{b_2} + \frac{d_1}{b_2} - \frac{\Omega^2 T^2 b_1 - 2\Omega T \frac{b_1^2}{b_2} + d_1 \frac{b_1^2}{b_2} + b_1}{d_1 b_1 b_2 + b_1^2 + b_2^2}$$

Die Zustandsregelung brächte ein stabiles System und wäre auch bei relativ kleinen Geschwindigkeiten noch brauchbar. Theoretisch sind damit Geschwindigkeiten von leicht unter 1 m/s möglich, die Verstärkung wird dabei jedoch sehr gross, und das Messrauschen würde zum Problem.

6.5 Regelung der Geschwindigkeit

Um das Fahren bei einer konstanten Geschwindigkeit zu erleichtern, wurde eine Regelung der Geschwindigkeit eingebaut. Hat man beim ursprünglichen Modellmotorrad an der Fernsteuerung noch die Stärke des Antriebs eingestellt, so wird jetzt eine gewünschte Geschwindigkeit eingestellt. Es wird folgende Regelung verwendet:

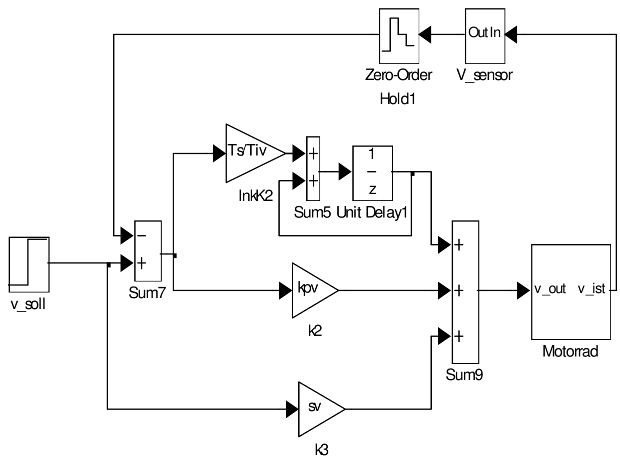


Abbildung 39. Regelung der Geschwindigkeit

Es handelt sich dabei um eine Kombination aus einer PI-Regelung und einer direkten Steuerung. Für die Auslegung der Regelung müsste man viel über den Fahrtregler und den Antrieb wissen. Da Modellbaukomponenten jedoch schlecht dokumentiert sind, wurde auf eine genaue Auslegung verzichtet. Die Regelungsparameter wurden im wesentlichen durch Probieren festgelegt, und eine genauere Bestimmung in einer späteren Arbeit könnte Sinn machen. Die Regelung funktioniert bei ebenem Gelände jedoch schon recht gut.

KAPITEL 7

Hardware

Die Hardware besteht aus zwei Platinen, der Prozessorplatine einerseits und der Versorgungs- und Analogschaltungsplatine andererseits. Die Pozessorplatine ist ein Evaluationboard M68332BCC von Motorola [Mot90]. Es enthält den Microcontoller, 128 Kilobytes EPROM und 64 Kilobytes RAM. Die zweite Platine enthält die Spannungsversorgung und die Analog-interfaceelektronik. Die Verbindung zwischen den beiden Platinen erfolgt über 4 Pinreihen, die folgendermassen belegt sind:

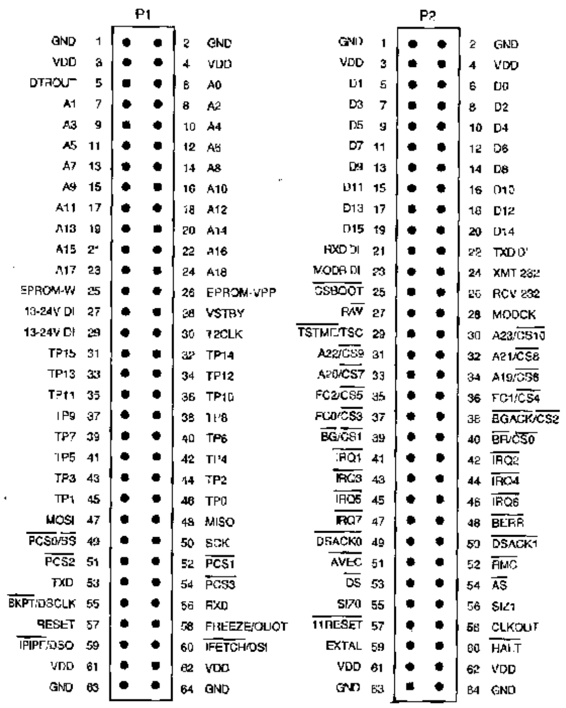


Abbildung 40. Pinbelegung der Prozessorplatine (Ansicht vom hinteren Ende des Motorrades)

7.1 Signale und Blockschaltbild

7.1.1 Übersicht über Signale

Die folgenden Tabellen geben eine Übersicht über die Signale, welche zum Prozessor kommen, und jene, welche von ihm weggehen. Die Analogsignale arbeiten mit Spannungen zwischen 0 und 5 Volt. Die pulsweitenmodulierten Signale (PWM) sind Signale von und zu Modellbaukomponenten.

Tabelle 11. **Übersicht über ankommende Signale**

Signal	Kodierung	Bindung
ϕ: Winkelgeschwindigkeit	analog	AD-Wandler, AIN1
v_soll: Geschwindigkeitssollwert	PWM	TPU, Kanal 0
α: Neigungssollwert	PWM	TPU, Kanal 1
choice: Auswahl	PWM	TPU, Kanal 2
v: Geschwindigkeit	Periodenlänge	TPU, Kanal 3

Tabelle 12. **Übersicht über abgehende Signale**

Signal	Kodierung	Bindung
v_out: Fahrtregler	PWM	TPU, Kanal 8
γ: Lenkwinkel	PWM	TPU, Kanal 9
Gewichtsverlagerung	PWM	TPU, Kanal 10

Die Pulsweite der PWM Signale variiert zwischen 1.4 und 1.8 ms, und die Frequenz beträgt standardmässig 50 Hz, die Servos verkraften aber auch höhere Frequenzen bis ca. 150 Hz. Das Blockschaltbild (Abbildung 41) gibt eine Übersicht über die zu verarbeitenden Signale.

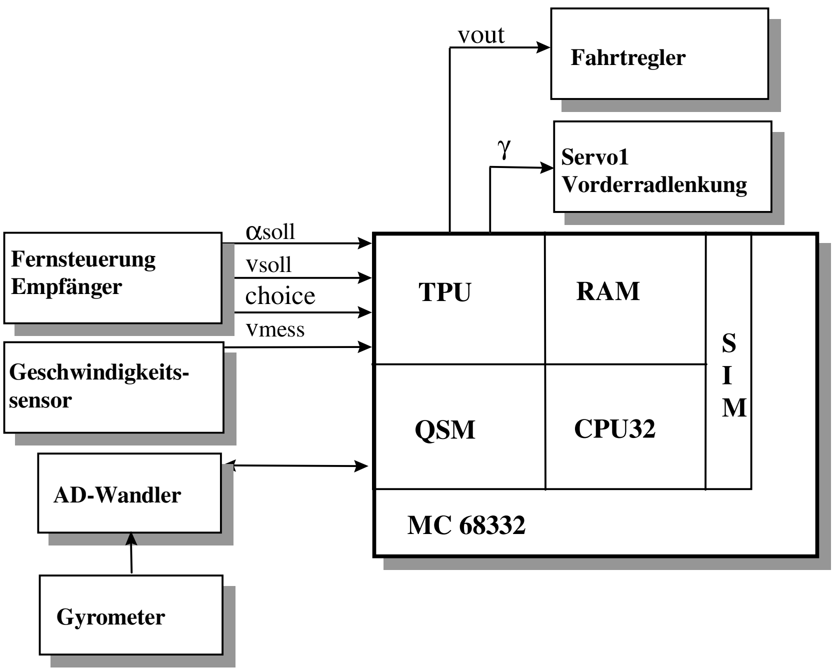


Abbildung 41. **Blockschaltbild der Hardware**

7.1.2 Signale der Time Processing Unit (TPU)

Es werden zwei Betriebsmodi der TPU verwendet. Für die Modellbauteile wird die Pulsbreitenmodulation (PWM) und für den Geschwindigkeitssensor die Perioden Messung verwendet. Für die Funktionswahl muss das "Kanal Funktionswahl Register" gesetzt werden (vgl. [SA98] in Kapitel 8.3.1). Für die Periodenmessung muss das Register auf den Wert 0xB und für PWM-Funktion auf 0xF gesetzt werden (siehe auch [Mot90]).

7.2 Spannungsversorgung

Zur elektrischen Versorgung werden zwei vollständig getrennte Speisungen verwendet. Aufgeteilt wird dabei in einen Analog- und einen Digitalteil. Der Digitalteil, das heisst die Prozessorplatine, wird vom Hauptakku versorgt. Dieser speist ebenfalls den Fahrtregler und damit den Antriebsmotor, die Empfängerplatine und die Servos. Die Nennspannung dieses Akkus ist 7.2 Volt, doch bei voller Last wird diese Spannung zusammenfallen, denn der Akku hat nach Herstellerangaben 4 Ohm Innenwiderstand. Für den Prozessor ist dieser Spannungsabfall bis etwa 3 Volt hinunter verkraftbar, fällt die Spannung darunter, so stoppt das Programm. Dies geschieht jedoch nur, wenn der Akku fast leer und damit auch für den Antriebsmotor nicht mehr genug Spannung vorhanden ist. Für den Analogteil hätte ein solcher Spannungsabfall jedoch schwerwiegende Folgen. Sinkt dort die Versorgungsspannung zu stark, so verzerren sich die Messwerte, welche die Sensoren liefern. Die Ausgangsspannung des Referenzspannungsteils REF-02 sinkt bereits bei einer Eingangspannung von knapp unter 7.2 Volt. Deshalb wurde entschieden, für den Analogteil eine 9 Volt Blockbatterie anzubringen, um genug Spannungsreserve zu haben.

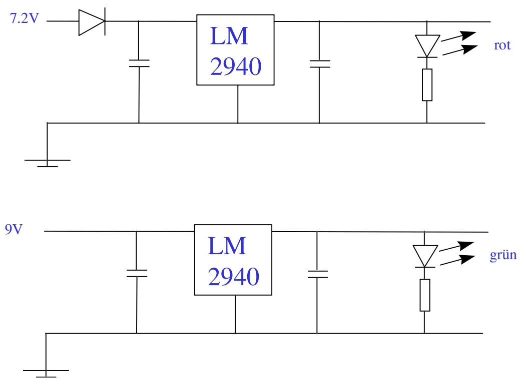


Abbildung 42. Schaltplan der zwei Spannungsversorgungen

Die beiden Servos und die Empfängerplatine werden direkt vom Fahrtregler mit 5V gespiesen.

7.3 Verstärkung und Filterung des Gyrosignals

Das Gyrometer liefert ein Analogsignal von 1.11 mV pro Grad pro Sekunde. Dieses Signal muss verstärkt und gefiltert werden, um es zu verarbeiten. Die Schaltung ist abgeleitet von einem Applikationsbeispiel im Datenblatt des Gyrometers. Das Signal wird um den Faktor 10 verstärkt und mit einem Tiefpass mit Eckfrequenz bei 1 kHz gefiltert. Dazu kommt ein Hochpass, um den Gleichspannungsanteil zu beseitigen. Die Eckfrequenz dieser Filterung musste tiefer als im Datenblatt angesetzt werden, weil sonst zuviel Nutzsignal abgeschnitten wurde, und liegt jetzt bei ca. 0.02 Hz.

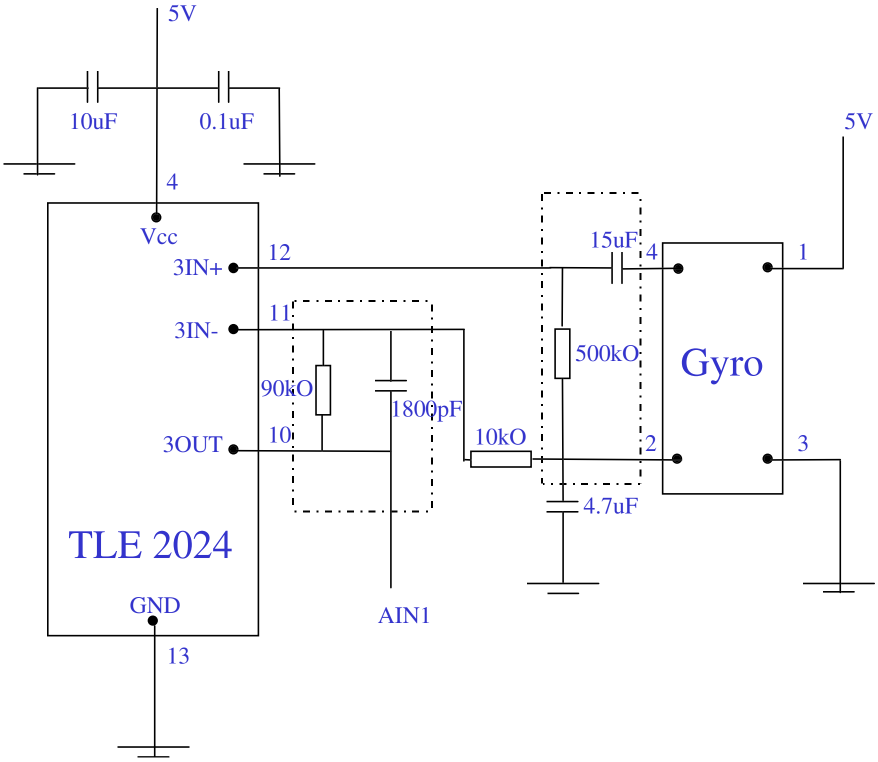


Abbildung 43. Beschaltung des Gyros

Durch die Verstärkung auf 11.1 mV/Grad/s ergibt sich nun folgender Signalbereich:

Tabelle 13. Signalbereich des Gyrometers

Winkelgeschwindigkeit (Grad/s)	Spannung	Digitalwert (12bit, 0-5V)
-90	1.4334	1174
0	2.4334	1993
90	3.4334	2812

Das Gyrometer nutzt also nicht den ganzen Bereich des AD-Wandlers aus. Ein Bit der Wandlung entspricht zirka 1/10 Grad pro Sekunde, was für die Regelung jedoch genügend genau ist.

7.4 Der AD-Wandler

Nach mehreren Versuchen, den AD-Wandler AD 7858 aus der Vorgängerarbeit richtig aufzusetzen, wurde entschieden, auf ein anderes Bauteil, den TLC 2543 von TI, zu wechseln. Es handelt sich dabei um einen 12 Bit Wandler mit 11 Analogkanälen. Die Ansteuerung dieses Wandlers ist einfacher als beim alten. Es muss nur ein Register zur Auswahl des Kanals gesetzt werden, und die Wandlung beginnt automatisch nach dem Schreiben dieses Registers. Der Anschluss an den Microcontroller MC68332 erfolgt über das Queued Serial Modul (QSM), wobei MOSI und MISO als Datenleitungen, SCLK als Transferclock und PCS1 als Chipselect verwendet werden. Die Ansteuerung des AD-Wandlers wird in Kapitel 8 (Software) beschrieben.

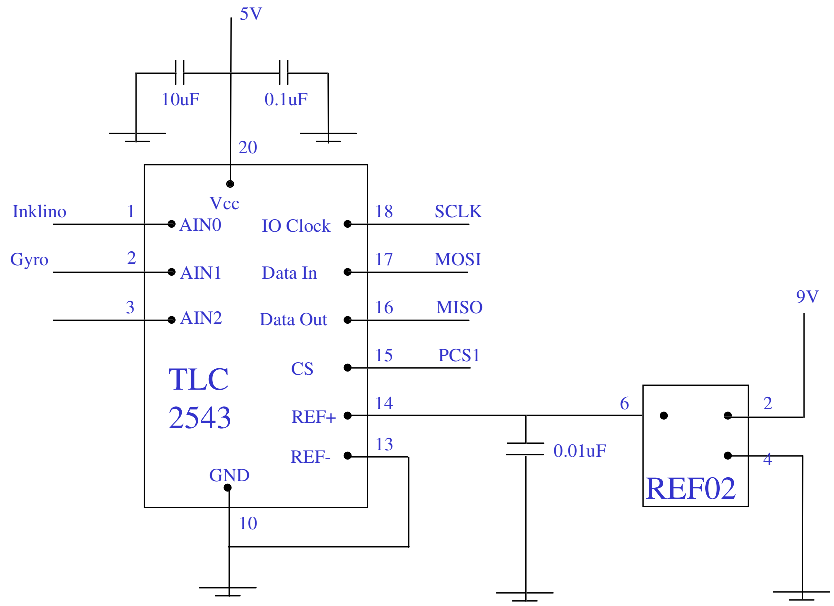


Abbildung 44. **Beschaltung des AD-Wandlers**

Zum Bereitstellen einer stabilen Referenzspannung wird ein Spannungsstabilisator REF-02 von Analog Devices verwendet. An den AD-Wandler wurde eine Rampe angelegt. Die Abweichungen von der Rampe betragen ca. 20 mV oder 0.5 Prozent (Abbildung 45).

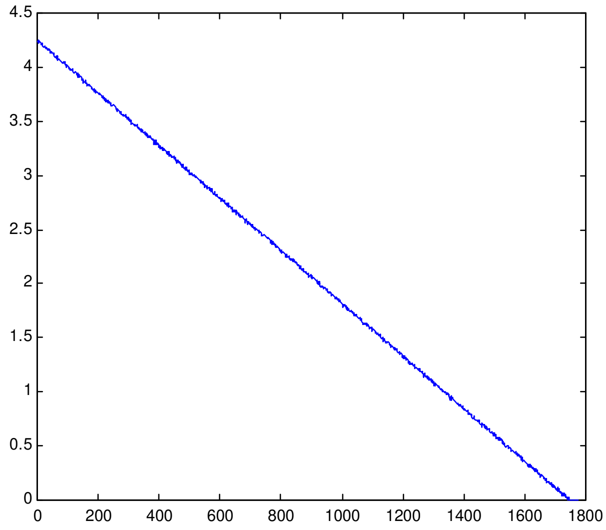


Abbildung 45. **Rampenmessung mit AD-Wandler**

7.5 BDI-Interface

Über das Background Debug Mode Interface (BDI) erfolgt die Anbindung des Microcontrollers an den PC. Mit der Crosscompiler Software HiCross können die Programme in der Programmiersprache C geschrieben und hinuntergeladen werden. Mit dem Debugger können Variablen des Programmes oder Register gelesen und geschrieben werden. Mit Hilfe der Terminalfunktion können Daten von der Tastatur an den Prozessor oder vom Prozessor an den Bildschirm bzw. in ein Textfile übertragen werden. Wird nun das BDI-Kabel einfach abgehängt, so entsteht ein Interrupt, und das Programm kommt zum stehen. Dies kann man umgehen, indem man den Debugger beendet, bevor man das Kabel abhängt, was aber den Nachteil hat, dass man es später nicht wieder an den PC anhängen kann. Für das Aufzeichnen von Daten ist das Wiederanhängen des Interfaces aber unumgänglich.

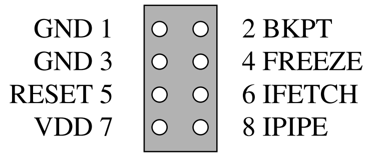


Abbildung 46. **Pinbelegung des BDI-Interface**

Mit einem Schalter, welcher den BKPT/DSCLK Pin auf High zieht, kann der Interrupt beim Abhängen umgangen werden. Dieser Tritt auf, wenn der Pin während mehr als zwei Zyklen der Übertragung auf Low bleibt. Dieser Schalter muss während dem Abhängen des Kabels betätigt werden.

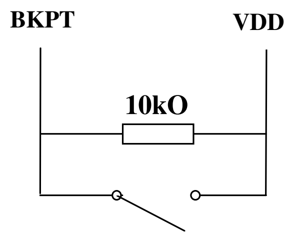


Abbildung 47. Schalter zum Abhängen des BDI-Interfaces

7.6 Signalqualität der Fernsteuerung

Die Signale, welche von der Fernsteuerung kommen, sind pulsweitenmoduliert. Die Qualität dieser Signale ist entscheidend für die Regelung, da von der Fernsteuerung die Sollwerte für die Geschwindigkeit und die Lenkung kommen. Eine genaue Regelung macht nur einen Sinn, wenn die Sollwerte genau eingestellt werden können.

7.6.1 Wertebereich

Die pulsweitenmodulierten Signale der Fernsteuerung variieren zwischen 912 und 1984 Mikrosekunden. Die Mittelstellung der Fernbedienung entspricht einem Signal mit 1454 us Pulsweite. Die Timerbasis der Time Processing Unit (TPU) ist auf 1.901 us eingestellt. Die Pulsweiten, welche als Vielfache der Timerbasis angegeben werden, variieren zwischen 480 und 1044, und die Mittelstellung liegt bei 765.

7.6.2 Stabilität des Signals

Der Steuerknüppel an der Fernsteuerung wurde in einer festen Stellung gelassen und das Signal, welches auf dem Motorrad ankommt, betrachtet. Das Signal wird vom Empfänger (Modellbauteil) aufgenommen und an den Prozessor weitergegeben. Die Aufzeichnung erfolgte auf dem Prozessor und ist als vielfaches der Timerbasis (1.901 us) angegeben.

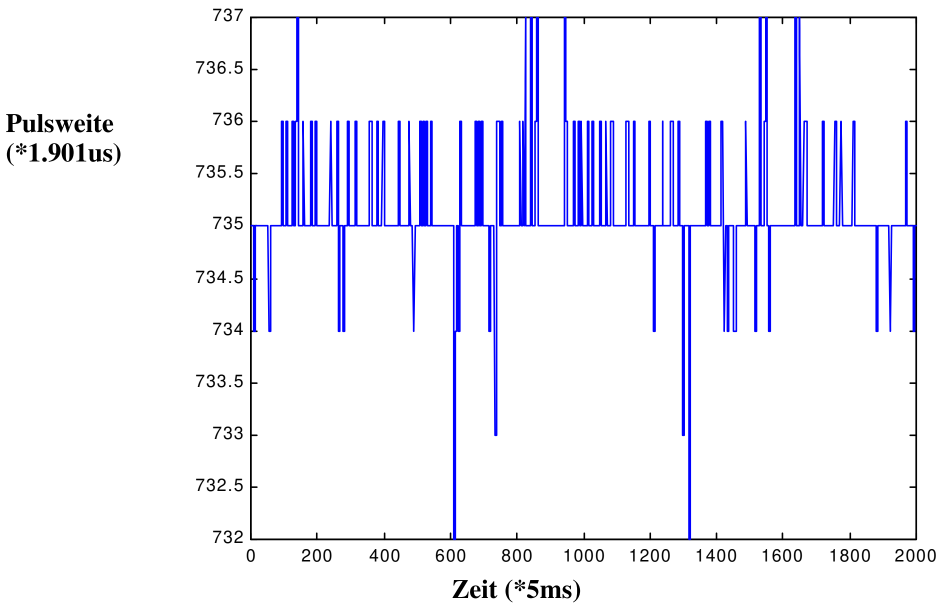


Abbildung 48. **Messkurve eines PWM Kanals**

Tabelle 14. Auswertung der PWM Stabilität (Pulsweite, *1.901us)

Mittelwert	Maximum	Minimum	STD	Var
735.1135	737	732	0.5402	0.2918
786.8425	789	783	0.7120	0.5069

Der Unterschied zwischen Maximal- und Minimalwert beträgt 5-6 mal die Timerbasis. Das entspricht einem Prozent des Wertebereichs. Das heisst, ein eingestellter Wert wird auf 1 Prozent konstant gehalten. Die Fehler bei der Übertragung von der Fernbedienung auf die Elektronik des Motorrades sind also relativ klein.

KAPITEL 8

Software

8.1 Aufbau

8.1.1 Modul “main”

Im Modul “main” werden die Initialisierungroutinen aufgerufen, und danach wird in eine Endlosschleife gegangen. Während dieser Endlosschleife treten periodische Interrupts auf, nach denen die Regelungsfunktion ausgeführt wird.

8.1.2 Modul “reg”

Das Modul “reg” enthält die Regelungsprozedur

void ISR_Regler(void),

eine Interrupt Service Routine welche nach jedem periodischen Interrupt abgearbeitet wird. Zu Beginn dieser Routine werden die Eingangswerte eingelesen. Die Geschwindigkeitsregelung

*void V_Regler (float *v_ist,float *v_soll,float *v_out)*
*/*Regelung der Geschwindigkeit v_ist= Geschw. (m/s)*/*
*/*v_soll= Geschw.Soll (m/s) v_out= PWM-Ausgang für Fahrtregler*/*

wird vom *ISR_Regler* aufgerufen und berechnet aufgrund des Soll- und des Istwertes einen neuen Ausgangswert für den Fahrtregler. Die Prozedur

*void Lenk_Regler (float *v_ist,float *r_soll,float *phi,float *out0)*
*/*Berechnung des neuen Lenkwinkels v_ist= Geschw. (m/s)*/*
*/*r_soll= Neigungssollwert (rad) phi= Rollgeschw. (rad/s) oout0= Lenkwinkel (rad)*/*

berechnet den neuen Ausgangswert für den Lenkservo. Zum Schluss werden die neu berechneten Werte für die Aktoren ausgegeben.

8.1.3 Modul “tpu”

Das Modul “tpu” enthält die Funktionen zum Initialisieren und Ansteuern der Kanäle der Time Processing Unit (TPU). Der Kanal 0 wird zum Einlesen des Geschwindigkeitssollwertes benutzt:

void Init_Vsoll(void);

ushort Read_Vsoll(void);

Dabei wird die Periodenlänge als ein Vielfaches der Timer Basis TCR1 von 1,901 us zurückgegeben. Das gleiche gilt für den Kanal 1, den Sollwert der Lenkung.

void Init_Lenksoll(void);

ushort Read_Lenksoll(void);

Ebenfalls gleich verhält es sich für den Kanal 2, der Auswahl zwischen Regelung und Steuerung, welche durch den Schalter in der Fernbedienung gemacht wird.

void Init_Choice(void);

ushort Read_Choice(void);

Der Kanal 3 wird zum Messen der Geschwindigkeit benutzt, und es wird gleich der skallierte Wert in Metern pro Sekunde zurückgegeben.

void Init_Vmess(void);

float Read_Vmess(void);

Die nächsten Kanäle werden zum Ausgeben von PWM Signalen an die Servos und an den Fahrtregler benutzt. Kanal 8 wird zur Ansteuerung des Fahrtreglers benutzt.

void Init_Motor(void);

void Update_Motor(ushort Pulsbreite);

Die Pulsbreite wird wieder als Vielfaches der Timer Basis TCR1 der TPU angegeben. Der Minimalwert für diese Pulsbreite ist 566, da das Motorrad nur vorwärts fährt. Kanal 9 dient zur Ansteuerung des Lenkservos, und seine Ausgabe wird durch die Anschläge des Servos beschränkt. Die Anschläge befinden sich bei den Pulsbreiten 665/925, und die Frequenz des PWM-Signals ist auf 125 Hz festgelegt

void Init_Servo1(void);

void Update_Servo1(ushort Pulsbreite);

Kanal 10 steuert den Servo für die Gewichtsverlagerung an.

void Init_Servo2(void);

void Update_Servo2(ushort Pulsbreite);

Der Kanal 15 wurde als diskreter Input/Output Kanal konfiguriert und dient als Hilfskanal für Messungen.

void Init_DIO(void);

void Force_High(void);

void Force_Low(void);

8.1.4 Modul “qsm”

Die Prozeduren des Queued Serial Modul (QSM) dienen zur Ansteuerung des AD-Wandlers. Die Initialisierung

void Init_ADC(void)

bringt das QSM in den Wraparound Mode. Das heisst, es werden 16 Messungen gemacht und dann gleich wieder bei der Ersten begonnen. Bei 15 Messungen wird der AD-Wandler angsteuert und der Kanal AIN1 des Wandlers ausgelesen. Bei der 16. Messung wird der Chipselect des Wandlers auf High gesetzt, um die Kommunikation zu reseten. Die Clockfrequenz der Datenübertragung wird auf 1.05 MHz festgelegt, die Zeit vom Setzen des Chipselect auf 1.49 us und die Zeit zwischen zwei Transfers auf 20.9 us. Diese Zeiten garantieren die Einhaltung der Timingbedürfnisse des AD-Wandlers, könnten aber auch noch knapper kalkuliert werden. Die Dauer einer Messung berechnet sich so:

$$t = 1.49\mu s + 12 \cdot 0.95\mu s + 20.9\mu s = 33.8\mu s$$

Dies entspricht einer Abtastrate von 30 kHz und könnte noch bis ca. 48 kHz gesteigert werden. Werden alle Kanäle ausgenutzt, so ist pro Kanal eine Abtastrate von 4.3 kHz möglich. Nach der Initialisierung muss die Wandlung mit der Prozedur

void Start_ADC(void);

gestartet werden. Das Auslesen der Werte erfolgt mit der Prozedur

*void Read_ADC(ushort *gyro);*

Die Übertragung zwischen Prozessor und AD-Wandler wird dabei gestoppt und aus den letzten 15 abgespeicherten Werten ein Mittelwert “gyro” gebildet. Danach werden die Wandlungen wieder fortgesetzt.

8.1.5 Modul “sim”

Das System Integration Modul (SIM) stellt die Möglichkeit für einen periodischen Interrupt zur Verfügung.

void Init_PeriodicTimer(uint CountValue, func_ptr func);

Die Berechnung der Frequenz der Interrupts erfolgt so:

CountValue=0x00YY

$$f = \frac{EXTALfreq}{4 \cdot 512 \cdot YY}$$

CountValue=0x01YY

$$f = \frac{EXTALfreq}{4 \cdot YY}$$

Zur Zeit wird mit einem Wert von 0x0029 eine Periode von 5ms gewählt.

8.2 Programmablauf

Nach dem Start des Programmes werden zuerst alle Initialisierungsprozeduren aufgerufen. Als letztes wird der periodische Interrupt initialisiert. Danach wird im Abstand von 5 ms der Interrupt ausglöst und die Reglerprozedur abgearbeitet. Diese liest die Werte der Sensoren und der Fernsteuerung ein, berechnet die neuen Werte der Regelung und gibt diese an die Aktoren aus. Abbildung 49 zeigt den Ablauf des Programmes schematisch.

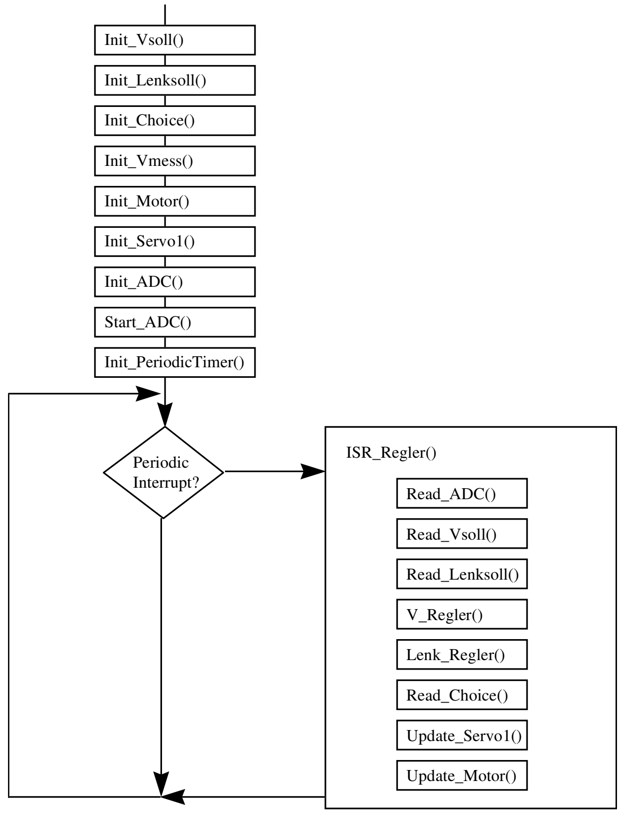


Abbildung 49. Schema des Programmablaufs

8.3 Dauer der Prozesse

Die folgende Tabelle gibt einen Überblick über die Dauer der einzelnen Prozesse. Das Bestimmen dieser Werte erfolgte durch das Setzen eines TPU-Pins des Prozessors und die Aufzeichnung des Verlaufes auf dem Speicher-KO.

TABELLE 15. Dauer der einzelnen Prozesse

Prozess	**Dauer (ms)**
AD-Wanler auslesen	0.1
Geschwindigkeitssensor lesen	0.3

TABELLE 15. Dauer der einzelnen Prozesse

Prozess	**Dauer (ms)**
Geschwindigkeitssoll lesen und skallieren	0.3
Lenksoll lesen	0.3
Lenksoll skallieren	0.5
Geschwindigkeit regeln	0.25
innere Regelparameter bestimmen	0.1
äussere Regelparameter bestimmen	0.9
Regelung aussen	0.25
Regelung innen	0.25
Auswahlwert einlesen	0.2
Ausgabe	0.1

Die gesamte Regelung dauert ca. 3.5 ms, das heisst, die Regelung ist abgearbeitet, bis der nächste Interrupt auftritt.

8.4 Modellierung als Petri Netz

Der Dokumentation von Software kommt in der heutigen Zeit eine immer grössere Bedeutung zu, weil man es sich nicht leisten kann, bei der Weitergabe der Arbeit an einen Nachfolger zu viel Zeit zu verlieren. Deshalb sollte der Code in einer formalen Art beschrieben werden. Als Formalismus kommen dabei Petri Netze zur Anwendung [Mur89], wobei diese mit dem “Codesign”-Tool [Ess96] gezeichnet wurden. Stellen sind als Kreise und Transitionen als Quadrate dargestellt. Es handelt sich um Timed-Petri Netze und die Dauer der einzelnen Transitionen wurden jeweils neben diese geschrieben. In Abbildung 50 sieht man die diskrete Regelung mit einer Abtastzeit von 5 ms einerseits und die Sensoren und Fernbedienungskanäle mit ihrer asynchronen oder unregelmässigen Aktualisierung andererseits.

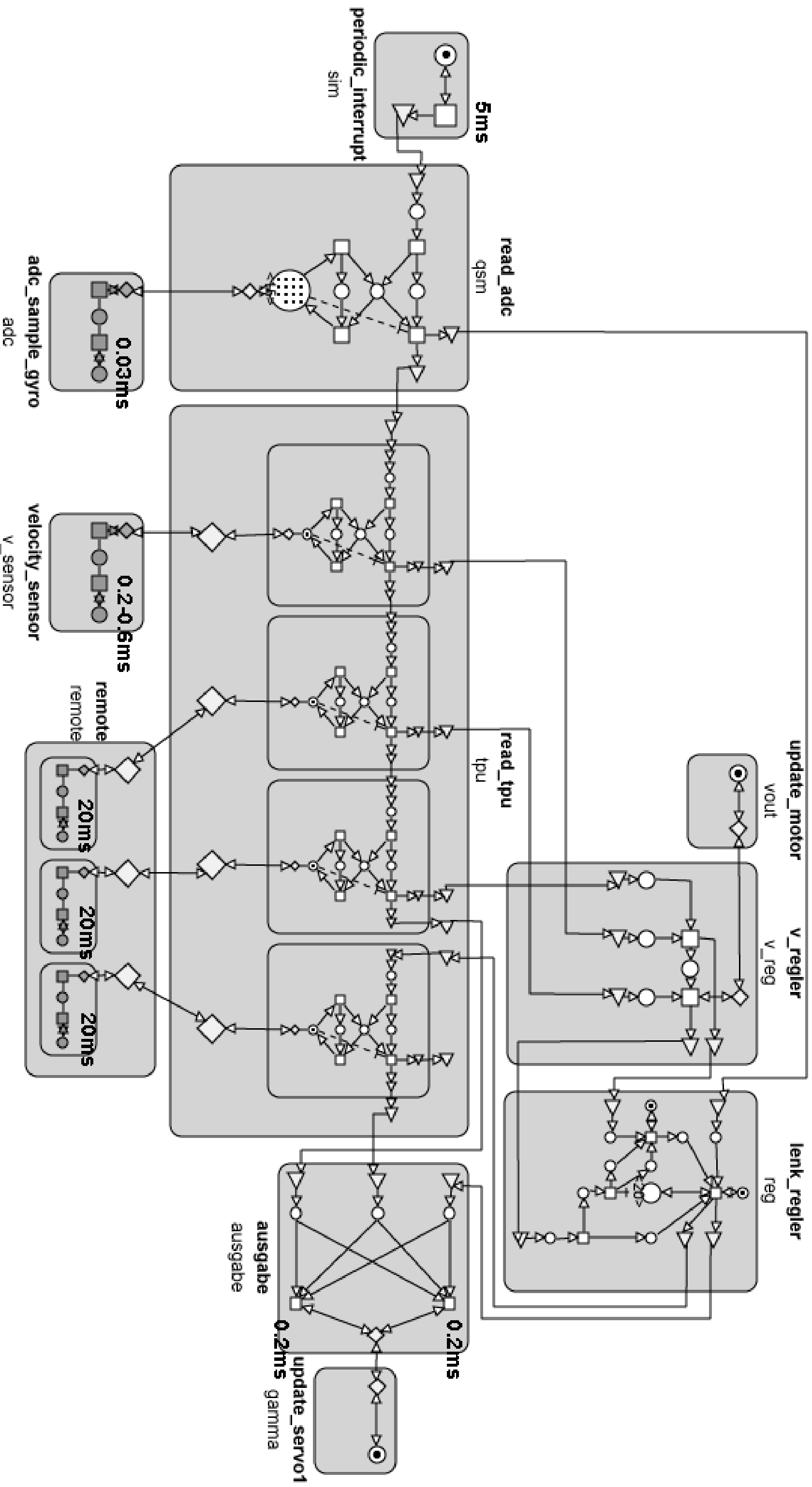


Abbildung 50. **Ablauf der Regelungsroutine (vgl. Kap. 8.2/8.3)**

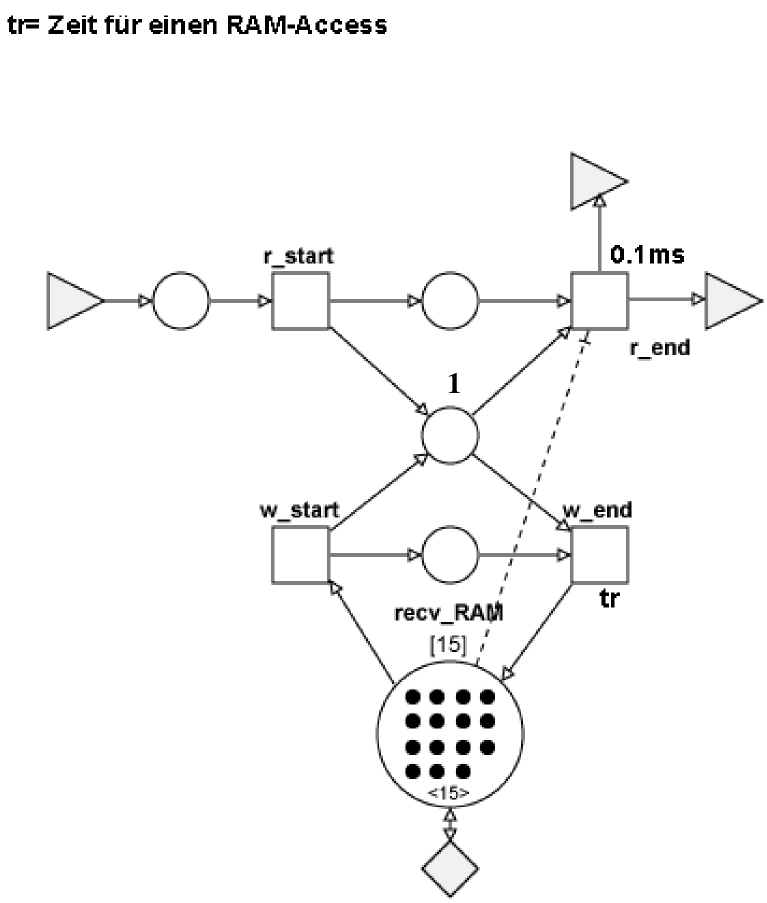


Abbildung 51. **Auslesen des AD-Wandlers (bildet Kap. 8.1.4 nach)**

In Abbildung 51 sieht man ein Semaphorkonstrukt. Dies wird verwendet, um zu beschreiben, dass ein Speicherstück nicht gleichzeitig beschrieben und ausgelesen werden kann. Bei der AD-Wandlung geschieht dies softwaremässig. Die Übertragung der Werte wird angehalten, während die schon abgespeicherten Werte ausgelesen werden.

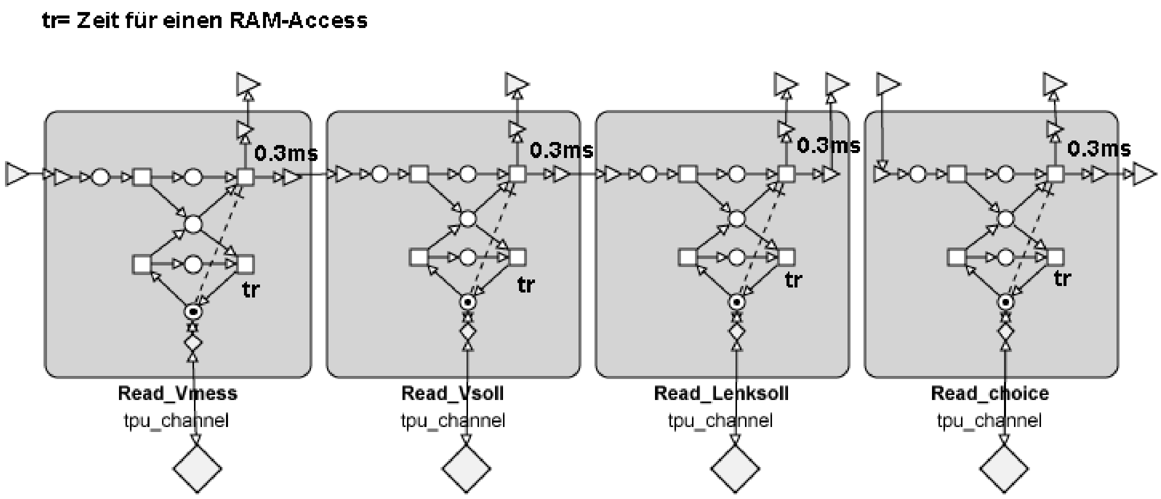


Abbildung 52. **Auslesen der TPU-Kanäle (Kap. 8.1.3)**

In Abbildung 52 sieht man die gleichen Semaphore wie schon in der vorherigen Abbildung. Die Verhinderung von Kollisionen beim Schreiben und Lesen erfolgt hier jedoch durch die Busarbitrierung des Prozessors.

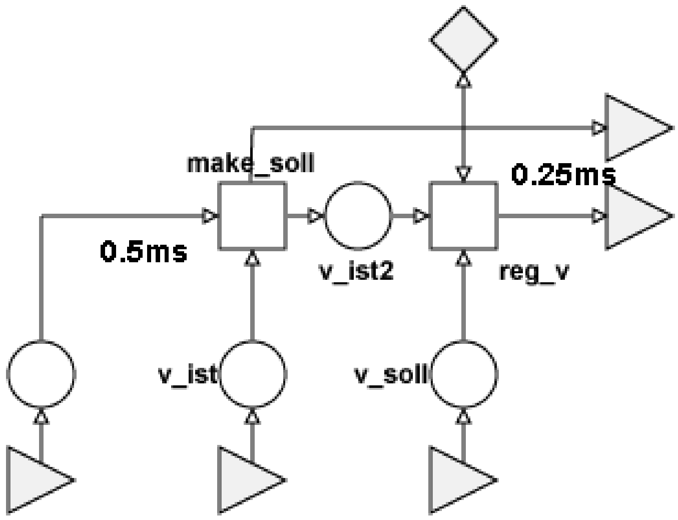


Abbildung 53. **Regelung der Geschwindigkeit (Kap. 8.1.2)**

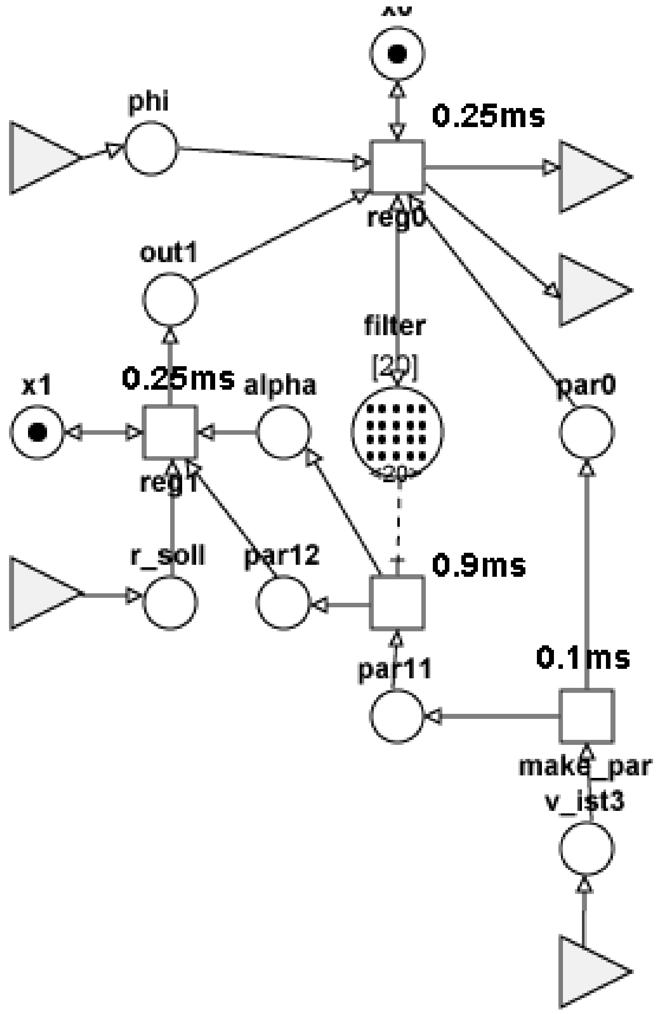


Abbildung 54. Regelung der Lenkung (Kap. 8.1.2)

8.5 Möglichkeit zum Tracen

Die Prozedur

*void Give_Trace(uint *uno,uint *due,uint *tre,uint *coun)*

im Modul “reg.c” ermöglicht die Übergabe von Werten von der Regelungsprozedur an einen verarbeitenden Prozess im Mainmodul. Dieser nutzt die Pausen zwischen den Aufrufen der Regelungsprozedur und speichert die Werte ab:

```
while(count<4000)
    {Give_Trace(&awert,&bwert,&cwert,&count);
    if (count>old_count)
      {
      atrace[count-2000]=awert;
      btrace[count-2000]=bwert;
      ctrace[count-2000]=cwert;
      old_count=count;
      }
    }
```

Später werden sie über die Terminalfunktion an den PC übermittelt:

```
for (count=0;count<2000;count++)
 {WriteInt(atrace[count]);WriteString(" ");
 WriteInt(btrace[count]);WriteString(" ");
 WriteInt(ctrace[count]);
 WriteLn();
 }
```

Die Werte erscheinen nun im Terminalwindow und können durch eine Menueinstellung auch abgespeichert werden.

KAPITEL 9

Zusammenfassung und Ausblick

9.1 Zusammenfassung

Ziel dieses Projekt war es, ein Beispiel für ein eingebettetes System zu schaffen. Ein Modellmotorrad sollte unter Hilfe von Sensoren, Aktoren und einem Rechner stabilisiert werden.

Da relativ schnell klar wurde, dass ein Inklinometer, wie es in der Vorgängerarbeit ausgewählt wurde, für die dynamischen Eigenschaften des Motorrades ungeeignet ist, mussten zuerst geeignete Sensoren evaluiert werden. Die Entscheidung fiel dabei auf ein Gyrometer, welches die Winkelgeschwindigkeit des Motorrades messen kann und diese in ein Analogsignal umwandelt.

Ein AD-Wandler bringt dieses Signal in eine für den Prozessor brauchbare Form. Um diesen Teil der Elektronik vor Störungen und Spannungsabfällen zu schützen, wurde eine separate Speisung mit einer 9 V Batterie eingebaut.

Es ist unerlässlich für die Regelung, die genaue Geschwindigkeit des Motorades zu kennen, da die Regelungsparameter mit dem Quadrat der Geschwindigkeit variieren. Ein Geschwindigkeitssensor von HP wurde montiert und an den Prozessor angeschlossen.

Die mathematische Modellierung des Motorades, welche für die Auslegung und Simulation der Regelung nötig ist, konnte zu einem grossen Teil aus der Vorgängerarbeit übernommen werden. Trotzdem mussten einige Fehler korrigiert und die Modellierung verfeinert werden.

Für die ersten Fahrversuche wurde eine PI-Regelung auf die Winkelgeschwindigkeit implementiert. Diese wurde durch Gain Scheduling erweitert, um die Regelung den verschiedenen Geschwindigkeiten anzupassen. Danach wurde diese Regelung zu einer Kaskadenregelung ausgebaut. Um Messwerte, welche in freier Fahrt gemacht werden, auswerten zu können, wurde eine Möglichkeit geschaffen, Daten vom Prozessorboard des Motorrades auf den PC hochzuladen. Die Daten werden dabei im RAM abgespeichert und dann über die Terminalfunktion des Debuggers auf den Bildschirm und in ein Textfile geladen.

Der Servo wurde durch ein präziseres und schnelleres Bauteil ersetzt und die Übertragung seiner Bewegung auf das Vorderrad verbessert. Die Übertragung hat jetzt weniger Spiel und der Lenkwinkel konnte von 18 auf 32 Grad erweitert werden. Dies bringt eine stabilere Fahrt und vor allem engere Kurvenradien. Die Minimalgeschwindigkeit ist 1.4 m/s und der minimale Kurvenradius misst etwa einen Meter. Das gekaufte Motorradmodell brauchte demgegenüber eine etwa doppelt so hohe Geschwindigkeit und einen Kurvenradius von 2 bis 3 Metern.

Das Ziel ein brauchbares Beispiel für ein eingebettetes System zu schaffen, konnte erreicht werden. Schliesslich wurde ein Video gedreht und geschnitten, welches die guten Fahreigenschaften des Motorrades demonstriert und für eine Vorlesung oder eine Demonstration gebraucht werden kann. Die Dokumentation wurde schliesslich mit Hilfe einer Checkliste, einer Bedienungsanleitung und einer Petri-Netzmodellierung der Softwarestruktur abgerundet.

9.2 Ausblick

Möchte man das Verhalten des Motorrades noch wesentlich verbessern, so wird man mit einem völlig neuen Konzept arbeiten müssen. Man wird von dem relativ einfachen System wegkommen und wahrscheinlich einen zusätzlichen Sensor zum Bestimmen der absoluten Neigung besorgen müssen. Bei der Modellierung wird der Übergang zu einem vereinfachten Mehrkörpermodell der nächste Schritt sein, denn bei kleinen Geschwindigkeiten werden die Lenkbewegungen des Vorderrades so gross, dass das Einkörpermodell an seine Grenzen stösst. Simulationen zeigen, dass bei Geschwindigkeiten unter 1 m/s die Fliehkräfte so klein werden, dass selbst eine perfekte Zustandsregelung nicht mehr stabilisierend wirkt. Deshalb ist der Einbau einer Gewichtsverlagerung zu prüfen. Die Verlagerung des Akkus, wozu jetzt die Möglichkeit besteht, hat zu wenig Einfluss auf das ganze System, da er zu leicht und zudem nicht genügend hoch eingebaut ist.

KAPITEL 10

Literatur

[BEF96]	J.Borenstein, H.R.Everett, L.Feng, ”*Where am I? Sensors and Methods for Mobile Robot Positionin*g”, Univerity of Michigan, 1996
[DigC90]	Franklin, Powell, Workman, “*Digital Control of Dynamic Systems*”, Addison Wesley, 1990
[Ess96]	R.Esser, ”*CodeSign Concepts and Tutorial*”, Institut TIK der ETH Zürich, 1996
[Get94]	Neil H. Getz, “*Conrol of Balance for a Nonlinear Nonholonomic Nonminimum Phase Model of a Bicycle*” American Control Conference, 1994
[Gyr97]	“*Gyrostar, Specification and Data Sheet ENC-05*”,Murata Inc.,1997[a]
[Hp96]	“*HEDS-5700, Technical Data*”, Hewlett Packard Inc.,1996[b]
[IBM96]	C. Verplaetse, ”*Self-motion-sensing Toys and Tools*”, IBM SYSTEMS JOURNAL, Vol 35, 1996
[Lin95]	Prof.Dr.A.H.Glattfelder, Prof.Dr.W.Schaufelberger, ”*Lineare Regelsysteme”*, Vorlesungsskript IfA, ETH Zürich, 1995
[Mot90]	“*M68332EVK Evaluation Kit User’s Manual*”, Motorola Semiconductor Inc., 1990
[Mur89]	T. Murata, ”*Petri Nets: Properties, Analysis and Applications*”, Proceedings of the IEEE, Vol. 77, No. 4, April 1989
[Neu96]	Dr.P.A.Neukomm, ”*Sensor- und Aktorsysteme*”, Vorlesungsskript EEK, ETH Zürich, 1996
[Pai97]	B.Paillard,”Guidance Control and Trajectory Stabilization of a Bicycle”, University of Sherbrooke1997 [c]

[SA98]	M.Neff, R.Odermatt, A.Stucki, “*Modellierung, Konstruktion und Implementation eines autonom fahrenden Zweiradmodells*”, Institute TIK und IfA der ETH Zürich, WS 97/98
[TLC97]	“*TLC 2543C, Technical Data*”, Texas Instruments Inc.,1997[d]
[TLE97]	“*TLE 2024, Technical Data*”, Texas Instruments Inc.,1997
[UBa97]	U. Baumann, “*Stabilisierung eines Invertierten Pendels”,* Institut IfA der ETH Zürich, 1997
[Wis96]	Dirk von Wissel, ”*DAE Control of Dynamical Systems: Example of a Riderless Bicycle*”, Ecole des Mines de Paris, 1996

a. http://www.iijnet.or.jp/murata/index.html

b. http://www.hp.com/go/components

c. http://www.gel.usherb.ca/mecatronique/educ/velo_an.html

d. http://www.ti.com

ANHANG A Softwarelistings

A.1 “main”-Modul

```
/****************************************************************************/
/*                              main.c                                      */
/****************************************************************************/
#include "tpu.h"
#include "qsm.h"
#include "sim.h"
#include "reg.h"

void main(void)
{
     Init_Vsoll();                      /*Kanal0 */
     Init_Lenksoll();                   /*Kanal1 */
     Init_Choice();                     /*Kanal2 */
     Init_Vmess();                      /*Kanal3 */
     Init_Motor();                      /*Kanal8 */
     Init_Servo1();                     /*Kanal9 */
     Init_ADC();                        /*Serielle Schnittstelle initialisieren */
     Start_ADC();
     Init_PeriodicTimer(0x0029, ISR_Regler);  /*Periode auf 5ms setzen */
     while (1)
        {
        }
}
```

A.2 “reg”-Modul

```
/****************************************************************************/
/*                              reg.h                                       */
/****************************************************************************/
#include "reg_def.h"
#ifndef REG_FUNC_H
#define REG_FUNC_H
/*Regleralgorithmus des Toeffs*/
void ISR_Regler(void);
void Give_Trace(uint *uno,uint *due,uint *tre,uint *coun);
#endif
```

```
/****************************************************************************/
/*                              reg.c                                       */
/****************************************************************************/
#include "reg.h"
#include "reg_def.h"
#include "tpu.h"
#include "qsm.h"

ushort gamma,gyro=0,lenk=786,mcount;
float out0=0,phi,x0=0,x1=0,xv=0;
float v_ist,v_soll,v_out=560,v_alt;
float r_soll,a_max;
float filter[20];

/*Tracing Funktion*/
void Give_Trace(uint *uno,uint *due,uint *tre,uint *coun)
{*uno=Read_Vmess();
 *due=gyro;
 if (out0>0.5) {*tre=867.5;}
 else if (out0<-0.5) {*tre=638.5;}
 else {*tre=753+229.183*out0;}
 *coun=count;
}
/*Regelung der Geschwindigkeit v_ist= Geschw. (m/s)*/
/*v_soll= Geschw.Soll (m/s) v_out= PWM-Ausgang für Fahrtregler*/
void V_Regler (float *v_ist,float *v_soll,float *v_out)
{
  if ((*v_soll-v_alt)>0.3){v_alt+=0.3;}
  else if ((*v_soll-v_alt)<-0.3){v_alt-=0.3;}
  else {v_alt=*v_soll;}
  xv=xv+(v_alt-*v_ist)/10;
  if (v_alt==0) {xv=0;}
  *v_out=560+50*v_alt+30*(v_alt-*v_ist)+xv;
  if (*v_out<560) {*v_out=560;}
  if (*v_out>950) {*v_out=950;}
}
/*Berechnung des neuen Lenkwinkels v_ist= Geschw. (m/s)*/
/*r_soll= Neigungssollwert (rad) phi= Rollgeschw. (rad/s)  oout0= Lenkwinkel (rad)*/
void Lenk_Regler (float *v_ist,float *r_soll,float *phi,float *out0)
{
  float kp0,ti0,kp1,ti1,dummy,out1,alpha;
  ushort i=0;

  dummy=0;
  for (i=0;i<20;i++){dummy+=filter[i];}
  dummy=dummy/20;
  alpha=dummy*(*v_ist)*(*v_ist)/2.8449;
  if ((*v_ist)>2.3)
   {kp1=1.25-*v_ist/6.66;
    ti1=5;
    kp0=-1/((*v_ist)*(*v_ist));
    ti0=-0.3*(*v_ist);
   }
  else if ((*v_ist)>1.4)
   {kp1=1;
    ti1=4;
    kp0=-0.6253/((*v_ist)*(*v_ist));
    ti0=-0.1671*(*v_ist)*(*v_ist);
   }
  else if ((*v_ist)>1.1)
   {kp1=1;
    ti1=4;
    kp0=-0.4757/((*v_ist)*(*v_ist));
    ti0=-0.1671*(*v_ist)*(*v_ist);
   }
  else
```

```
   {kp1=1;
    ti1=4;
    kp0=-0.3688;
    ti0=-0.2022;
   }
  x1=x1+0.005*(*r_soll-alpha)/ti1;
  out1=x1+kp1*(*r_soll-alpha);
  x0=x0+0.005*(out1-*phi)/ti0;
  *out0=x0+kp0*(out1-*phi);
  filter[mcount]=*out0;
  mcount++;
  if (mcount==20) {mcount=0;}
}

#pragma TRAP_PROC
void ISR_Regler(void)
{
  float dummy;

  /*Einlesen von Werten gyro,v_ist,r_soll*/
  /******************************************************/
  Read_ADC(&gyro);
  v_ist=Read_Vmess();                    /*m/s*/
  v_soll=Read_Vsoll();
  lenk=Read_Lenksoll();
  /*Skalieren der Eingangswerte                         */
  /******************************************************/
  v_soll=v_soll-520;
  v_soll=v_soll*0.01627907;              /*m/s*/
  if (v_soll<0) {v_soll=0;}
  if (v_soll>6) {v_soll=6;}
  if (v_soll==0) {x0=0;x1=0;}

  dummy=gyro*0.001220703125;             /*Volt*/
  dummy=dummy-2.4334;                    /*offset*/
  phi=dummy/0.6359831525952;             /*rad/s*/

  if (v_ist>2.6) {a_max=0.698;}          /*max. 40 Grad Neigung*/
  else {a_max=(v_ist*v_ist/9.81);}       /*min 1 Meter Radius*/

  dummy=lenk;
  dummy=(dummy-786)/200*a_max;
  if ((dummy>=-0.7)&&(dummy<=0.7)){r_soll=dummy;}
  /*Steuerung+Regelung von v                            */
  /******************************************************/
  V_Regler(&v_ist,&v_soll,&v_out);
  /*Regelung der Lenkung                                */
  /******************************************************/
  Lenk_Regler (&v_ist,&r_soll,&phi,&out0);
  /*Ausgabe                                             */
  /******************************************************/
  dummy=(1+out0/3)*out0;
  gamma=758+229.183*dummy;               /*PWM-wert (24Grad=100)*/
  if (Read_Choice()>600)
    {Update_Servo1(lenk);
     Update_Motor(v_out);
    }
  else
    {Update_Servo1(gamma);
     Update_Motor(v_out);
    }
}
```

A.3 “qsm”-Modul

```
/*************************************************************************/
/*                              qsm_func.h                              */
/*************************************************************************/
#include "qsm_def.h"
#ifndef QSM_FUNC_H
#define QSM_FUNC_H
/*Funktionen zur Ansteuerung der AD-Wandlung*/
/*Initialisiert das QSM für AD- Wandlung*/
void Init_ADC(void);
/*Auslesen der gewandelten Werte gyro= 12-Bit Wert*/
void Read_ADC(ushort *gyro);
/*Startet die AD-Wandlungen*/
void Start_ADC(void);
/*Stoppt die AD-Wandlungen*/
void Halt_ADC(void);
#endif

/*************************************************************************/
/*                                 qsm.c                                */
/*************************************************************************/
#include "qsm.h"
#include "qsm_def.h"
/*********************************************************************
Funktionen zur Handhabung des AD Wandlers an der QSPI Schnittstelle
*********************************************************************/
/*Setzt die Word zum Senden auf Null*/
void Reset_TransRam(void)
{
  int i;
  for (i=0;i<16;i++) *(WORD*)(0xfffd20+i*2) = 0x0000;
}
/*Setzt das Word an der Stelle pos auf data
Falls pos= -1 wird das ganze TransRam mit data beschrieben*/
void Set_TransRam(ushort data,int pos)
{
  if (pos==-1)
  {
      int i;
      for(i=0;i<16;i++) *(WORD*)(0xfffd20+i*2)= data;
  }
  else *(WORD*)(0xfffd20+2*pos)= data;
}
/*Setzt das Ram für den Empfang von Daten auf Null*/
void Reset_ReceiveRam(void)
{
  int i;
  for(i=0;i<16;i++) *(WORD*)(0xfffd00+i*2) = 0x0000;
}
/*Liest das empfangene Word an der Stelle pos im ReceiveRam*/
ushort Read_ReceiveRam(int pos)
{
   return *(WORD*)(0xfffd00+2*pos);
}
/*Setzt das Control Ram auf Null*/
void Reset_ContrRam(void)
{
  int i;
  for (i=0;i<16;i++) *(BYTE*)(0xfffd40+i) = 0x0000;
}
/*Setzt das Byte an der Stelle pos auf data
Falls pos= -1 wird das ganze ContrRam mit data beschrieben*/
void Set_ContrRam(ushort data,int pos)
{
 if (pos==-1)
```

```
 {
    int i;
    for(i=0;i<16;i++) *(BYTE*)(0xfffd40+i)=data;
 }
 else *(BYTE*)(0xfffd40+pos) = data;
}
/*Initialisiert das QSM für AD- Wandlung*/
void Init_ADC(void)
{

        /*Transfer zum AD*/

        Set_TransRam(0x0200,-1);            /*Kanal 2:Gyro*/

        Set_ContrRam(0xFC,-1);              /*Control Bytes setzen*/
        Set_ContrRam(0xFF,0);               /*Control Byte setzen */

        *QMCR |=0x4003;
        *QPDR=0x00;
        /*Pin Assignment Register*/
        *QPAR= 0x13;                        /*PCS1,MOSI,MISO für QSPI gebraucht*/
       *QDDR= 0x16;                      /*Output:PCS1,SCK,MOSI  Input:MISO*/
        /*QSPI Submodul Register*/
        *SPCR0= 0xB008;
        /*QSPI Master, 12 Bit, SCK= 1.05 MHz @ 16.8MHz, CPOL=0,CPHA=0*/
        *SPCR1= 0x190B;
        /*QSPI disabled, DSCKL= 1.49us>1.425us, DTL=20.9 us>12.2us*/
        *SPCR2= 0x4F00;                     /*Wraparound Mode, 16 Messungen*/


}
/*Startet die AD-Wandlungen*/
void Start_ADC(void)
{
*SPCR1|=_SPE;
while(!(*SPSR & _SPIF));
}
/*Stoppt die AD-Wandlungen*/
void Stop_ADC(void)
{
*SPCR1&=~(_SPE);                            /*Bit15 setzen*/
}
/*Auslesen der gewandelten Werte gyro= 12-Bit Wert*/
void Read_ADC(ushort *gyro)
{
ushort i;
uint dummy;
*SPCR3 |= _HALT;                            /* HALT */
while (!(*SPSR & _HALTA)) {};               /* Wartet bis Übertragung fertig ist*/
for (i=0;i<15;i++)
 {
 dummy+= *(WORD*)(0xfffd00+2+i*2);
 }
*gyro=dummy/15;
*SPCR1|=(_SPE);                             /*SPE in jedem Fall wieder gesetzt*/
*SPCR3 &= ~(_HALT);                         /*Übertragung wird fortgesetzt    */
}
```

A.4 “sim”-Modul

```
/****************************************************************************/
/*                              sim.h                                       */
/****************************************************************************/
#include "sim_def.h"
#ifndef SIM_FUNC_H
```

```
#define SIM_FUNC_H
typedef void (*func_ptr) (void);            /*Funktionspointer*/
/*Initialisierung des periodischen Interrupts*/
/*CountValue= Periode  func= Interrupt Service Routine*/
void Init_PeriodicTimer(uint CountValue, func_ptr func);
#endif

/******************************************************************************/
/*                              sim.c                                    */
/******************************************************************************/
#include "sim.h"
#include "sim_def.h"
void SetVBR(void)
{
   asm
   {
    MOVE.L     #VBR_ADR,D0                /*VectorBaseRegister auf VBR_ADR setzen*/
    MOVEC      D0,VBR
   }
}
/*VecNum= Vektornummer func= Interrupt Service Routine*/
void SetISRAdr ( int VecNum, func_ptr func) /*Eintragen der Interrupt Service
                                                      Routine Adresse*/
{                                           /*in der Exception Vector Table */
   asm
   {   MOVEC    VBR,A0;                     /*Vector Base Register in A0*/
       MOVE.W   VecNum,D0;                  /*Vektor Nummer in D0*/
       ANDI.L   #0x000000FF,D0;             /*Vektor Nummer im Bereich von 0..255
                                                          ausmaskieren*/
       ASL      #2,D0;                      /*Vektor Nummer *4 */
       ADDA.L   D0,A0;                      /*VBR und Vektor Offset addieren*/
       MOVE.L   func,A1;                    /*Adresse der ISR in A1 legen */
       MOVE.L   A1,(A0);                    /*Adresse der ISR in der Tabelle
                                                      abspeichern (A0)*/
   }

}
/*CountValue= Periode   func= Interrupt Service Routine*/
void Init_PeriodicTimer(uint CountValue, func_ptr func)
{
   SetVBR();                                /*VBR auf 0x000000 setzen */
   SetISRAdr(0x40,func);                   /*Interrupt Service Routine (ISR) in der
                                                 Exception Tabelle eintragen */
                                            /*Vektor Nummer ist 64 (=$40)
                                              (Erster User Defined Vector) */
   pitr &= 0x0000;

   pitr |= CountValue;                      /*Periode 5ms */

   picr &= 0x0000;
   picr |= 0x0640;                          /*Interrupt Level 6, Vector $40*/

   asm
   {   ANDI.W #0xf0ff,SR;                   /*Interrupt Levels unter 6 ausblenden*/
       ORI.W  #0x0500,SR;
   }
   mcr|= 0x0004;                          /*InterruptArbitrationField auf 4 setzen*/
}
```

A.5 *“tpu”-Modul*

```
/******************************************************************************/
/*                              tpu.h                                       */
/******************************************************************************/
```

```
#include "tpu_def.h"
/* Inputs  */
/*Kanal 0: Sollwert für Geschwindigkeit  */
void Init_Vsoll(void);
ushort Read_Vsoll(void);
/*Kanal 1: Sollwert für Lenkung */
void Init_Lenksoll(void);
ushort Read_Lenksoll(void);
/*Kanal 2: Auswahl Regelung/Steuerung*/
void Init_Choice(void);
ushort Read_Choice(void);
/*Kanal3: Geschwindigkeitssensor */
void Init_Vmess(void);
float Read_Vmess(void);
/* Outputs  */
/*Kanal 8: Ansteuerung Antriebsmotor   */
void Init_Motor(void);
/*Pulsbreite*1.901us=Pulsweite*/
void Update_Motor(ushort Pulsbreite);
/*Kanal 9: Ansteuerung Lenkservo  */
void Init_Servo1(void);
/*Pulsbreite*1.901us=Pulsweite*/
void Update_Servo1(ushort Pulsbreite);
/*Kanal 10: Ansteuerung Gewichtsverlagerungsservo  */
void Init_Servo2(void);
/*Pulsbreite*1.901us=Pulsweite*/
void Update_Servo2(ushort Pulsbreite);
/* Hilfsfunktionen */
/*Kanal 15  */
void Init_DIO(void);
void Force_High(void);
void Force_Low(void);

/*****************************************************************************/
/*                              tpu.c                                        */
/*****************************************************************************/
#include "tpu.h"
#include "tpu_def.h"

/* Eingaenge */

 /*********************************************************************
Funktionen zur Handhabung des Kanal 0
Messung der Pulsbreite des PWM-Signals der Fernsteuerung (PPWA)
(Input = Geschwindikgeitssollwert)
*********************************************************************/
void Init_Vsoll(void)
{
        pram->chan0.p0= 0x0007;
        /*TCR1 als Time Basis gesetzt, Steigende Flanke erkennen*/
        pram->chan0.p1= 0x0100; /* Anzahl Pulsweiten die aufsummiert werden  */
       pram->chan0.p4= 0xFF00; /* Accum_Rate; alle 0.5ms wird Accum aufgefrischt  */

        /*Kanalfunktion auf $F(PPWA) setzen */
        cfsr3=cfsr3|0x000F;

        /*Modus 2 24-Bit Pulsbreiten aufsummieren   */
        hsqr1=hsqr1|0x0002;

        /*Host Service Request absetzen zur Initialisierung  */
        hsrr1=hsrr1|0x0002;

        /*Kanalprioritaet auf High setzen */
        cpr1=cpr1|0x0003;
}
ushort Read_Vsoll(void)
{
```

```
        return pram->chan0.p5;
}
/*********************************************************************
Funktionen zur Handhabung des Kanal 1
Messung der Pulsbreite des PWM-Signals der Fernsteuerung (PPWA)
(Input = Lenksollwert)
*********************************************************************/
void Init_Lenksoll(void)
{
        pram->chan1.p0= 0x0007; /*TCR1 als Time Basis gesetzt,
                                Steigende Flanke erkennen */
        pram->chan1.p1= 0x0100; /* Anzahl Pulsweiten die aufsummiert werden */
        pram->chan1.p4= 0xFF00; /* Accum_Rate; alle 0.5ms wird Accum aufgefrischt */

        /*Kanalfunktion auf $F(PPWA) setzen    */
        cfsr3=cfsr3|0x00F0;

        /*Modus 2 24-Bit Pulsbreiten aufsummieren */
        hsqr1=hsqr1|0x0008;

        /*Host Service Request absetzen zur Initialisierung  */
        hsrr1=hsrr1|0x0008;

        /*Kanalprioritaet auf High setzen  */
        cpr1=cpr1|0x000C;
}
ushort Read_Lenksoll(void)
{
        ushort dummy;
        dummy=1572-(pram->chan1.p5);
        if (dummy>986) {return 986;}
        if (dummy<586) {return 586;}
        else {return dummy;}
}
/*********************************************************************
Funktionen zur Handhabung des Kanal 2
Messung der Pulsbreite des PWM-Signals der Fernsteuerung (PPWA)
(Input = Auswahl Regelung/Steuerung)
*********************************************************************/
void Init_Choice(void)
{
        pram->chan2.p0= 0x0007; /*TCR1 als Time Basis gesetzt, Steigende Flanke
                                erkennen  */
        pram->chan2.p1= 0x0100; /* Anzahl Pulsweiten die aufsummiert werden     */
        pram->chan2.p4= 0xFF00; /* Accum_Rate;alle 0.5ms wird Accum aufgefrischt   */

        /*Kanalfunktion auf $F(PPWA) setzen */
        cfsr3=cfsr3|0x0F00;

        /*Modus 2 24-Bit Pulsbreiten aufsummieren  */
        hsqr1=hsqr1|0x0020;

        /*Host Service Request absetzen zur Initialisierung */
        hsrr1=hsrr1|0x0020;

        /*Kanalprioritaet auf High setzen */
        cpr1=cpr1|0x0030;
}
ushort Read_Choice(void)
{
        return pram->chan2.p5;
}
/*********************************************************************
Funktionen zur Handhabung des Kanal 3
Messung der Zeit zwischen 2 Input Transitions (PMA)     HP-Sensor
(Input = Geschwindigkeit)
```

```
*******************************************************************/
void Init_Vmess(void)
{
        pram->chan3.p0=0x0006; /*Steigende Flanke erkennen*/
        pram->chan3.p1=0x00FF; /*Num of Theeth=max*/
        pram->chan3.p3=0x00FF; /*TCR2 max value=max*/

        /*Kanalfunktion auf $B(PMA) setzen */
        cfsr3=cfsr3|0xB000;

        /*Modus 1:PMA Count Mode*/
        hsqr1=hsqr1|0x0040;

        /*Host Service Request absetzen zur Initialisierung*/
        hsrr1=hsrr1|0x0040;

        /*Kanalprioritaet auf High setzen*/
        cpr1=cpr1|0x00C0;
}
float Read_Vmess(void)
{
       if (((pram->chan3.p2)&0x00ff)>0x20) {return 0;}
       else if ((pram->chan3.p5)==0) {return 1;}
       else {return (413.3421/(pram->chan3.p5));}
}
//Ausgaenge
/*******************************************************************
Funktionen zur Handhabung des Kanal 8
Generierung eines PWM Signals zur Ansteuerung des Fahrtreglers
*******************************************************************/
/*Initialisierung
Timer1 Basis ist auf 1.901us (16.8333MHz/32) gesetzt.
Periodendauer = 20ms   = 10521*1.901us
High Time     = 0ms
*/
void Init_Motor(void)
{
        pram->chan8.p0 = 0x0092; /* TCR1 als Time Basis gesetzt, Force Pin Low*/
        pram->chan8.p2 = 0x0000; /* High Time 0ms setzen*/
        pram->chan8.p3 = 0x2919; /* Periodendauer auf 20ms setzen*/

        /*Kanalfunktion auf $9(PWM) setzen*/
        cfsr1=cfsr1|0x0009;
        /*Host Service Request absetzen zur Initialisierung*/
        hsrr0=hsrr0|0x0002;

        /*Kanalprioritaet auf High setzen*/
        cpr0=cpr0|0x0003;
}
/*Pulsbreite*1.901us=Pulsweite*/
void Update_Motor(ushort Pulsbreite)
{
        if (Pulsbreite < 566) Pulsbreite=566;
        pram->chan8.p2 = Pulsbreite;     /*High Time setzen*/
}
/*******************************************************************
Funktionen zur Handhabung des Kanal 9
Generierung eines PWM Signals zur Ansteuerung der Lenkung
*******************************************************************/
/*Initialisierung
Timer1 Basis ist auf 1.901us (16.8333MHz/32) gesetzt.
Periodendauer = 20ms   = 10521*1.901us
High Time     = 0ms
*/
void Init_Servo1(void)
{
        pram->chan9.p0 = 0x0092; /*TCR1 als Time Basis gesetzt, Force Pin Low */
```

```
        pram->chan9.p2 = 0x0000; /* High Time 0ms setzen   */
        pram->chan9.p3 = 0x1070; /* Periodendauer auf 8ms setzen (125Hz)   */

        /*Kanalfusnktion auf $9(PWM) setzen mit bitweiser Oder-Verknüpfung  */
        cfsr1=cfsr1|0x0090;
        /*Host Service Request absetzen zur Initialisierung  */
        hsrr0=hsrr0|0x0008;

        /*Kanalprioritaet auf High setzen  */
        cpr0=cpr0|0x000C;
}
/*Übergabe einer neuen Pulsbreite für Fahrtregler an Kanal8
High Time
   0% (kein Ausschlag)= 1.44ms = 760*1.901us
 100% (+31° Ausschlag)= 1.89ms = 925*1.901us
-100% (-31° Ausschlag)= 1.26ms = 665*1.901us
*/
/*Pulsbreite*1.901us=Pulsweite*/
void Update_Servo1(ushort Pulsbreite)
{
        /*Begrenzung*/
        if (Pulsbreite>925) Pulsbreite=925;
        if (Pulsbreite<665) Pulsbreite=665;

        pram->chan9.p2 = Pulsbreite;     /*High Time setzen*/
}
/************************************************************************
Funktionen zur Handhabung des Kanal 10
Generierung eines PWM Signals zur Ansteuerung der Gewichtsverlagerung
************************************************************************/
/*Initialisierung
Timer1 Basis ist auf 1.901us (16.8333MHz/32) gesetzt.
Periodendauer = 20ms   = 10521*1.901us
High Time     = 0ms
*/
void Init_Servo2(void)
{
        pram->chan10.p0 = 0x0092; /* TCR1 als Time Basis gesetzt, Force Pin Low*/
        pram->chan10.p2 = 0x0000; /* High Time 0ms setzen*/
        pram->chan10.p3 = 0x2919; /* Periodendauer auf 20ms setzen*/

        /*Kanalfunktion auf $9(PWM) setzen mit bitweiser Oder-Verknüpfung*/
        cfsr1=cfsr1|0x0900;
        /*Host Service Request absetzen zur Initialisierung*/
        hsrr0=hsrr0|0x0020;

        /*Kanalprioritaet auf High setzen*/
        cpr0=cpr0|0x0030;
}
/*Pulsbreite*1.901us=Pulsweite*/
void Update_Servo2(ushort Pulsbreite)
{
        /*Begrenzung  */
        if (Pulsbreite > 978) Pulsbreite=978;
        if (Pulsbreite < 560) Pulsbreite=560;
        pram->chan10.p2 = Pulsbreite;     /*High Time setzen*/
}
/************************************************************************
Funktionen zur Handhabung des Kanal 15
discrete IO
************************************************************************/
void Init_DIO(void)
{
        cfsr0|=0x8000;
        cpr0|=0xC000;
        hsrr0|=0xC000;
```

```
}
void Force_High(void)
{
        hsrr0|=0x4000;
}
void Force_Low(void)
{
        hsrr0|=0x8000;
}
```

A.6 Matlab File

```
% Simulink-Modell des Motorrades
% ******************************

% Parameter des Modells
% ******************************************************

% statische Parameter
% -------------------
% Konstanten
g=9.8067;                          %Erdbeschleunigung
% Geometrie
l=0.29;                          %Radstand (aus Katalog)
h=0.112;                         %Hoehe des Schwerpunktes ueber Boden
                                 %(gemessen 9.4.98)
rh=0.064;                        %Radius des Hinterrades (aus Katalog)
rv=0.059;                        %Radius des Vorderrades (aus Katalog)
rhf=0.044;                       %Radius der Felge des Hinterrades
                                 %(gemessen 19.01.98)
rvf=0.04;                        %Radius der Felge des Vorderrades
                                 %(gemessen 19.01.98)
% Masse
m=2.308;                         %Gesamtgewicht (gewogen 8.4.98)
mh=0.205;                        %Masse des Hinterrades (gewogen 16.1.98)
mv=0.160;                        %Masse des Vorderrades (gewogen 16.1.98)
% Messung des Traegheitsmoments der Raeder
rg=0.008/2;                      %halbe Dicke des Gewichts, welches ans Rad geklebt
                                 %wurde
mg=0.05;                         %Masse des Gewichts
% Vorderrad
pv=5;                            %Anzahl Perioden der Schwingung am Vorderrad
rgv=rv+rg;                       %Abstand des Mittelpunkts des Gewichts vom
                                 %Drehpunkt des Vorderrrades
Tv1=4.25;Tv2=4.28;Tv3=4.3;Tv4=4.3;%Messwerte Vorderrad
Tv=mean([Tv1;Tv2;Tv3;Tv4]);      %Mittelwert dieser Messwerte
Jv=mg*g*rgv/(pv*2*pi/Tv)^2-mg*rgv^2;% Traegheitsmoment des Vorderrades
Jvmin=1/2*mv*rv^2;               %untere Beschraenkung des Traegheitsmoments des
                                 %Vorderrades
Jvmax=1/2*mv*(rv^2+rvf^2);       %obere Beschraenkung des Traegheitsmoments des
                                 %Vorderrades
% Hinterrad
ph=3;                            %Anzahl Perioden der Schwingung am Hinterrad
rgh=rh+rg;                       %Abstand des Mittelpunkts des Gewichts vom Drehpunkt
                                 %des Hinterrades
Th1=2.78;Th2=2.80;Th3=2.72;Th4=2.72;% Messwerte Hinterrad Zeitverbrauch von ph
Th=mean([Th1; Th2; Th3; Th4]);   %Mittelwert dieser Messwerte
Jh=mg*g*rgh/(ph*2*pi/Th)^2-mg*rgh^2;% Traegheitsmoment des Hinterrades
Jhmin=1/2*mh*rh^2;               %untere Beschraenkung des Traegheitsmoments des
                                 %Hinterrades
Jhmax=1/2*mh*(rh^2+rhf^2);       %obere Beschraenkung des Traegheitsmoments des
                                 %Hinterrades
%Messung des Traegheitsmoments des Motorrades um die Kippachse
rf=0.12;                         %Abstand des Angriffspunktes der Federwaage von der
```

```
                                     %Drehachse
af=4*pi;                             %Auslenkung Motorrad aus der Nulllage [rad]
Ff=210/102;                          %Kraft auf die Federwaage
cf=Ff*rf/af;                         %Federkonstante der Gummizuege
% Motorrad
pm=3;                                %Anzahl Perioden der Schwingung
Tm1=13.6;Tm2=13.66;Tm3=13.66;Tm4=13.68;% Messwerte Motorrad fuer den
                                       % Zeitverbrauch
Tm=mean([Tm1;Tm2;Tm3;Tm4]);          % Mittelwert dieser Messwerte
Js=cf/(pm*2*pi/Tm)^2;                %Traegheitsmoment des Rahmens
Jsmin=1/2*m*(h/2)^2;                 %untere Beschraenkung des Traegheitsmoments des
                                     %Rahmens
Jsmax=1/2*m*h^2;                     %obere Beschraenkung des Traegheitsmoments des
                                     %Rahmens
J=m*h^2+Js;                          %Traegheitsmoment des Motorades um die
                                     %Kippachse durch die Auflagepunkte
%Nachlauf
rn=0.015;
b3=-rn/(3*h);

% dynamische Parameter
% -------------------
% Motorrad
v=3;                                 %Geschwindigkeit des Motorrades
Lh=Jh*v/rh;                          %Drall des Hinterrades
Lv=Jv*v/rv;                          %Drall des Vorderrades
L=Lh+Lv;                             %Gesamtdrall der Raeder
% Servo
kservo=1;                            %Verstaerkung des Servos
aservo=32*pi/180;                    %Anschlag des Servos bzw. der Lenkung
%gyro
kgyro=1;
agyro=pi/2;

% Modellierung des Motorrades
% ***************************************************
% Motorrad
T=sqrt(J/(m*g*h));                   %Zeitkonstante
d1=1/T*((v*h)/(g*l)-L/(m*g*l));      %Daempfung
b1=Lv/(m*g*h*T);
b2=-v/(g*l)*(v-L/(m*h));
Am=1/T*[0 1;1 -d1];
Bm=[b1/T;b2/T];
% Maximale Neigung
% ****************************************************
if v>2.3
   a_max=0.7;
else
   a_max=v*v/9.81;
end
% Rückführparameter
% ****************************************************
wo=6.5;
D=1;
Ti0=b2/(T*wo*wo);
kp0=(2*wo*T*D-d1)/b2;
ome=0.4;
Ti1=1/(ome^2);
kp1=2*ome;
Vf=(v^2)/(g*l);
% Regelung
% ****************************************************
Ts=0.005;
% Simulationsparameter
% ****************************************************
opts = nan*ones(6,1);
```

```
opts(1) = 1e-2;                         %relative error
opts(2) = 1e-6;                         %minimum step size
opts(3) = 1e-2;                      %maximum step size
opts(4) = nan;                       %not used in rk45
opts(5) = 3;                         %display warning prameter
opts(6) = 2;                         %plot parameter
TF = 4;                              %Simulationsdauer
%Simulation
% *******************************************************
[tt xx yy] = rk45('sst9',TF,[],opts);
theta=v/l*Ts*cumsum(yy(:,1));
ko1=v*Ts*cumsum(cos(theta));
ko2=v*Ts*cumsum(sin(theta));
alpha=yy(:,3);
i=length(alpha);
theta=theta(1:10:i);
alpha=alpha(1:10:i);
ko1=ko1(1:10:i);
ko2=ko2(1:10:i);
ko3=[ko1                ko1-0.3*sin(theta).*sin(alpha)];
ko4=[ko2                ko2+0.3*cos(theta).*sin(alpha)];
ko5=[zeros(size(ko2))   0.3*cos(alpha)                ];
yy=yy*180/pi;
% 2D-Plot
subplot(414), plot(tt,yy(:,4)),title('soll')
subplot(413), plot(tt,yy(:,3)),title('alpha')
subplot(412), plot(tt,yy(:,2)),title('phi')
subplot(411), plot(tt,yy(:,1)),title('gamma')
pause;
% 3D-Plot
subplot(111),mesh(ko3,ko4,ko5),axis equal,view(-80,30),colormap(hsv)
```

ANHANG B

Messresultate und Simulationskurven

B.1 Servomessungen

B.1.1 Graupner Servo C512

Messungen von Totzeit und Stellgeschwindigkeit

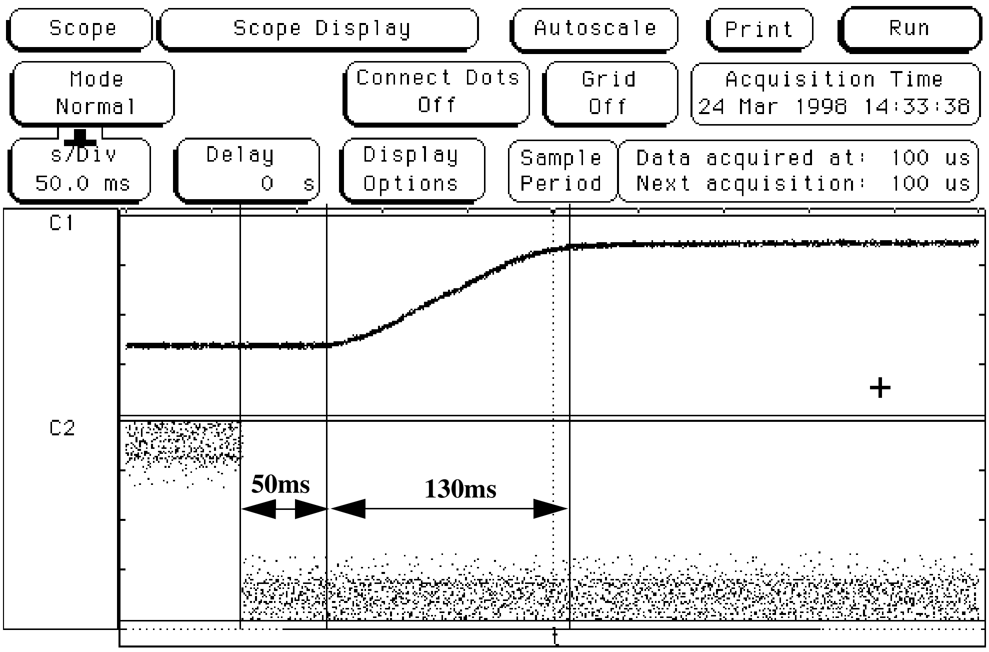


Abbildung 55. **Messung der Bewegung von -18 zu +18 Grad**

- Totzeit: ~50ms
- Bewegungszeit: ~130ms

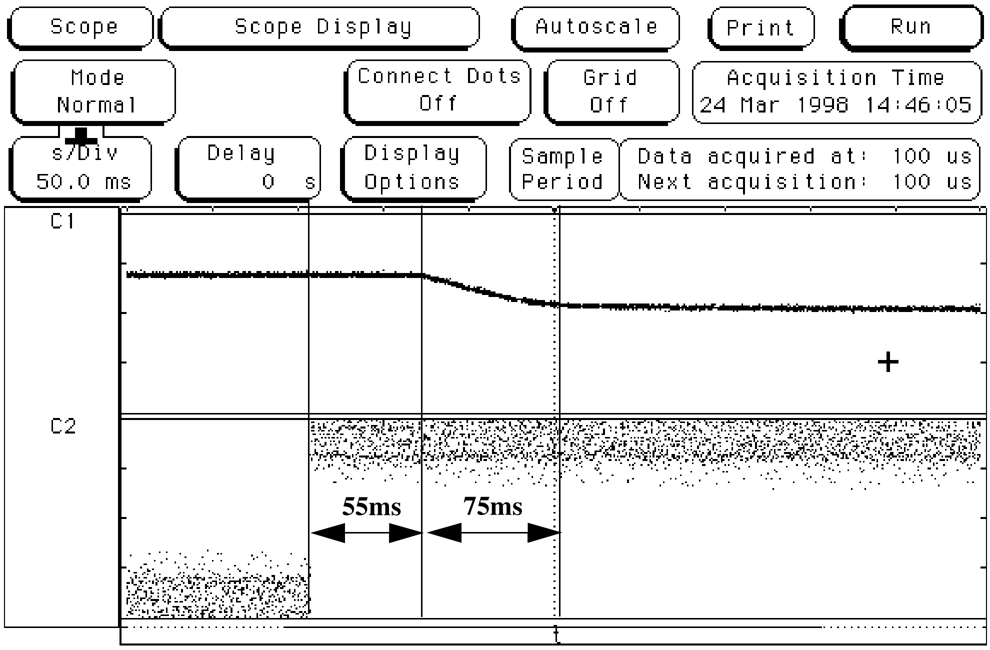


Abbildung 56. Messung der Bewegung von +5 zu -5 Grad

- Totzeit: ~55ms
- Bewegungszeit: ~75ms

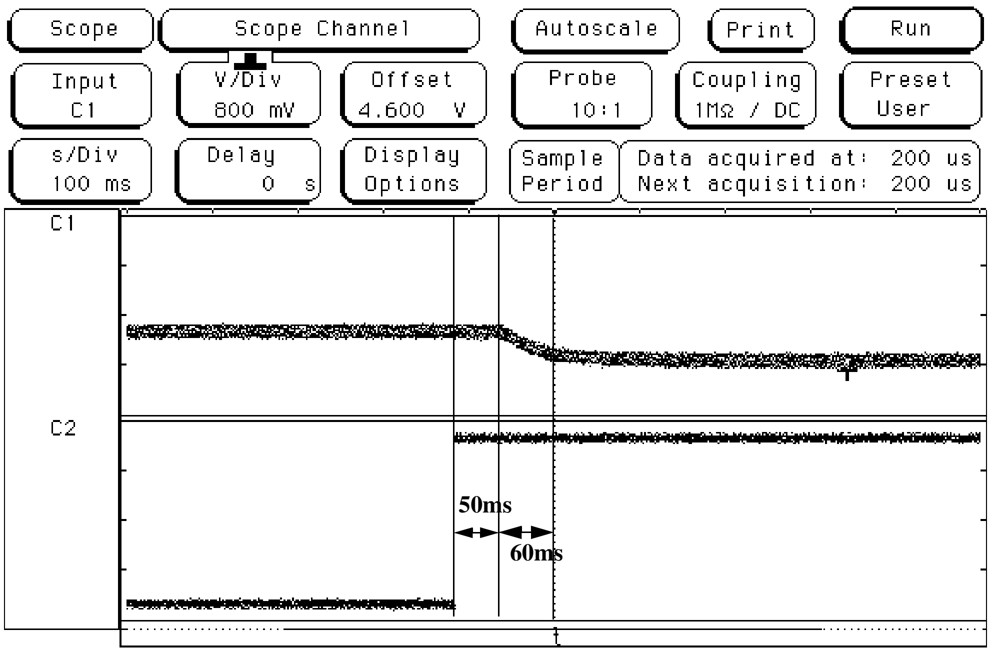


Abbildung 57. Messung der Bewegung von +2 zu -2 Grad

- Totzeit: ~50ms

- Bewegungszeit: ~60ms

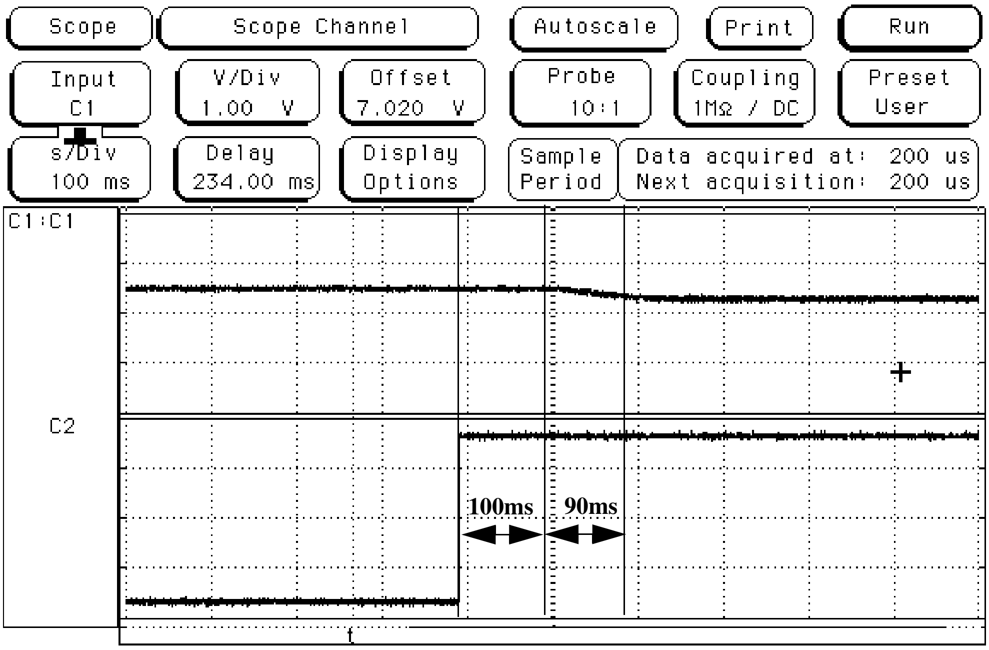


Abbildung 58. **Messung der Bewegung von +1 zu -1 Grad**

- Totzeit: ~100ms
- Bewegungszeit: ~90ms

Messungen des Spiels

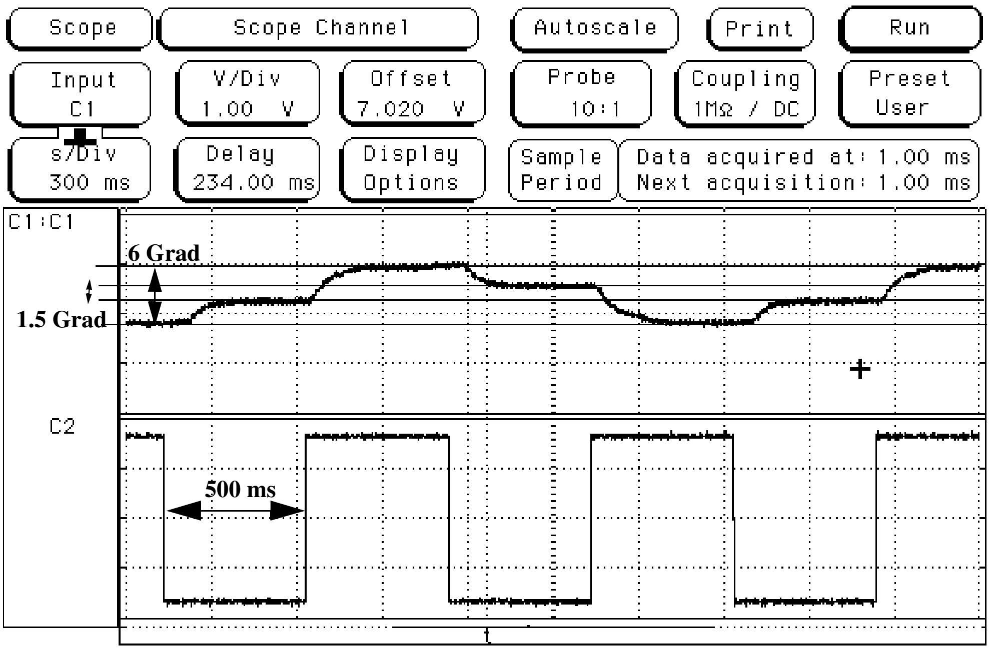


Abbildung 59. **Messung bei einer Bewegung von +-3Grad**

- ca. 1.5 Grad Spiel

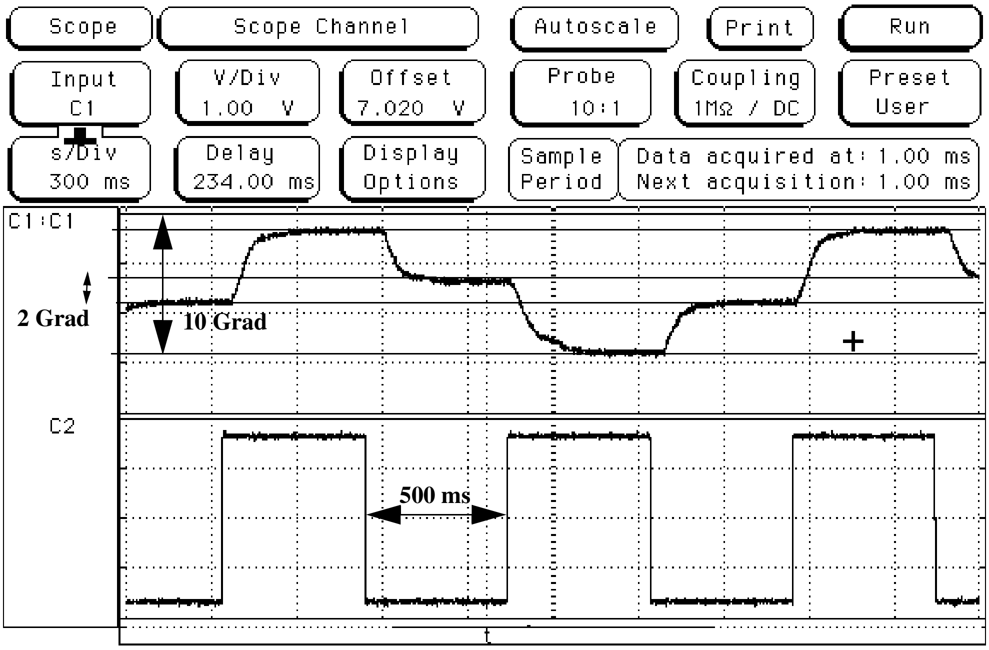


Abbildung 60. **Messung der Bewegung von +-5 Grad**

- ca. 2 Grad Spiel

B.1.2 Futaba Servo S9203

Messungen zu Totzeit und Stellgeschwindigkeit

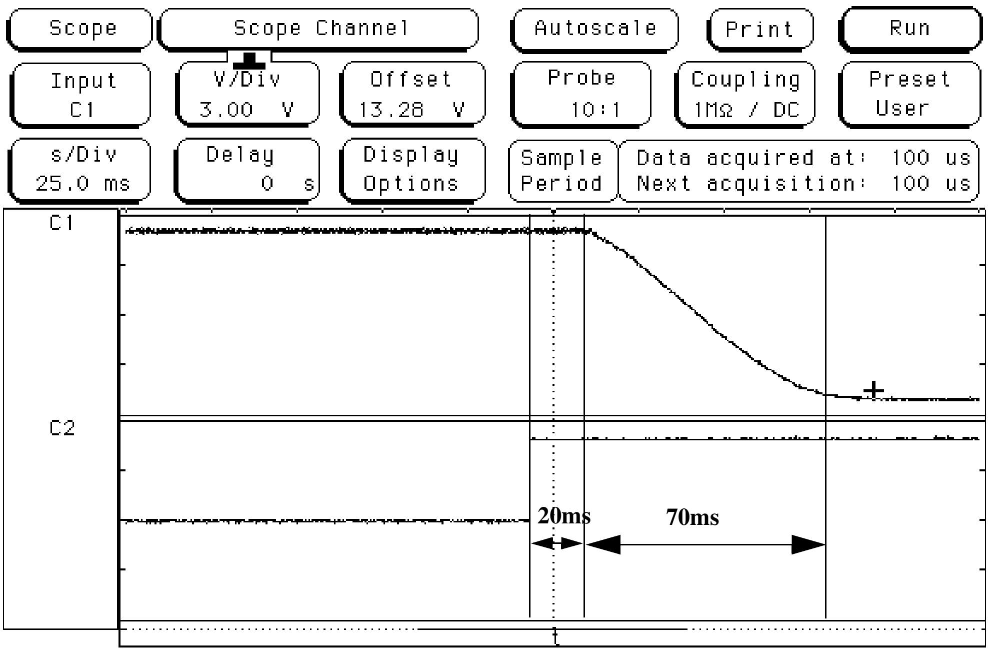


Abbildung 61. **Messung der Bewegung von -18 zu +18 Grad**

- Totzeit:20ms
- Stellzeit:70ms

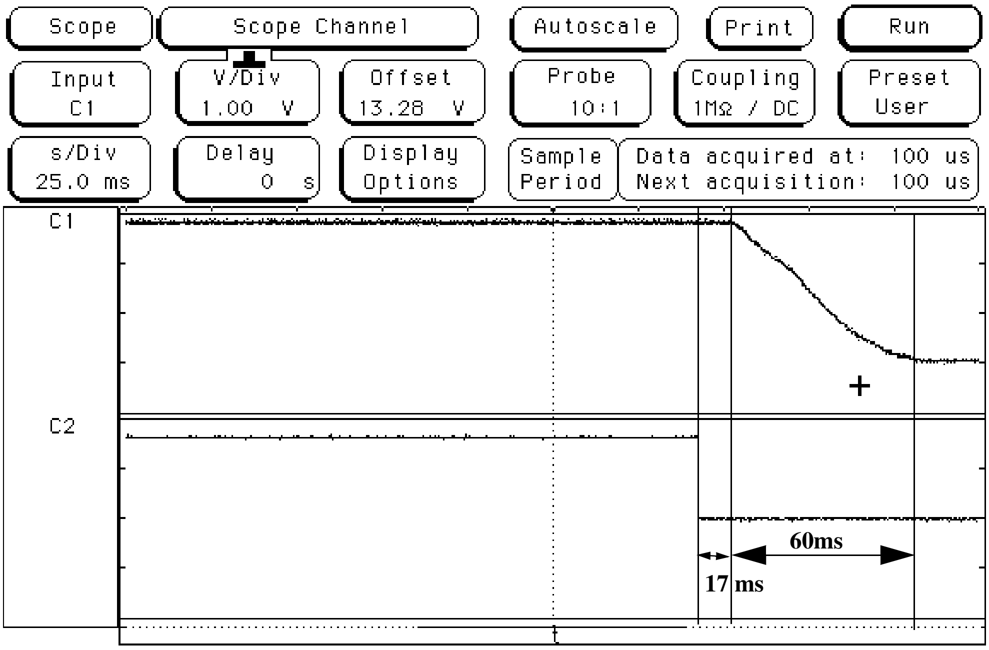


Abbildung 62. **Messung der Bewegung von +5 zu -5 Grad**

- Totzeit:17ms
- Stellzeit:60 ms

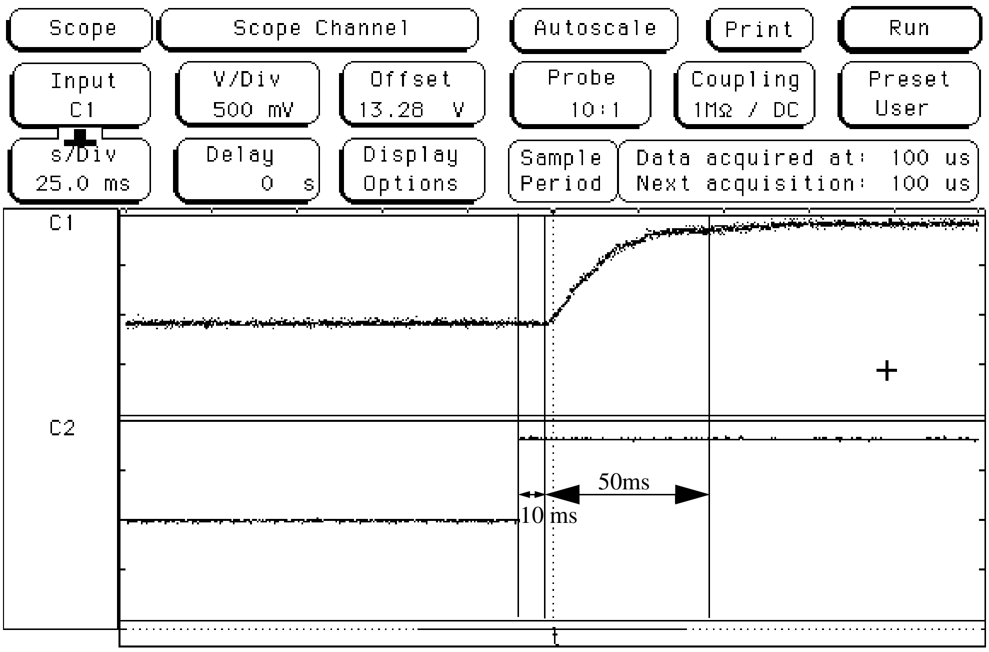


Abbildung 63. **Messung der Bewegung von +2 zu -2 Grad**

- Totzeit: 10ms

- Stellzeit:50 ms

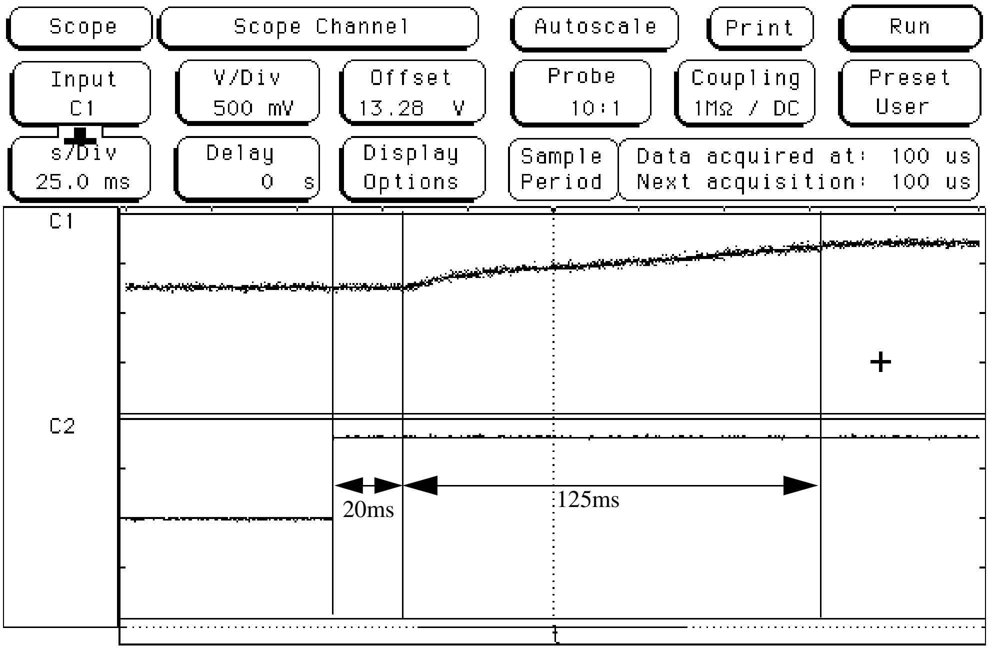


Abbildung 64. **Messung der Bewegung von +1 zu -1 Grad**

- Totzeit: 20ms
- Stellzeit:125 ms

Messungen des Spiels

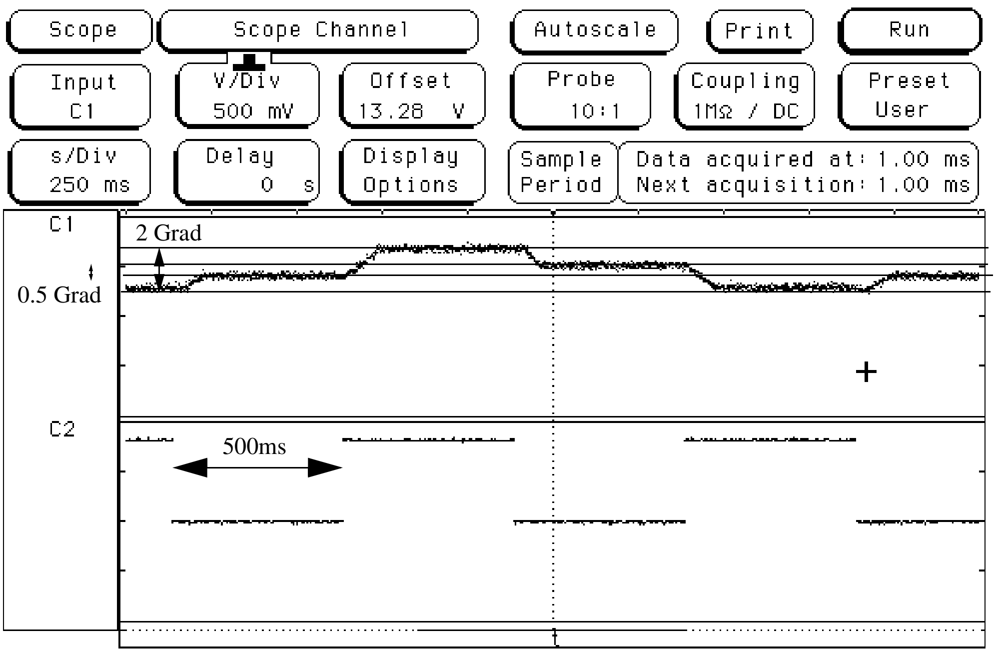


Abbildung 65. **Messung bei einer Bewegung von +-1Grad**

- ca. 0.5 Grad Spiel

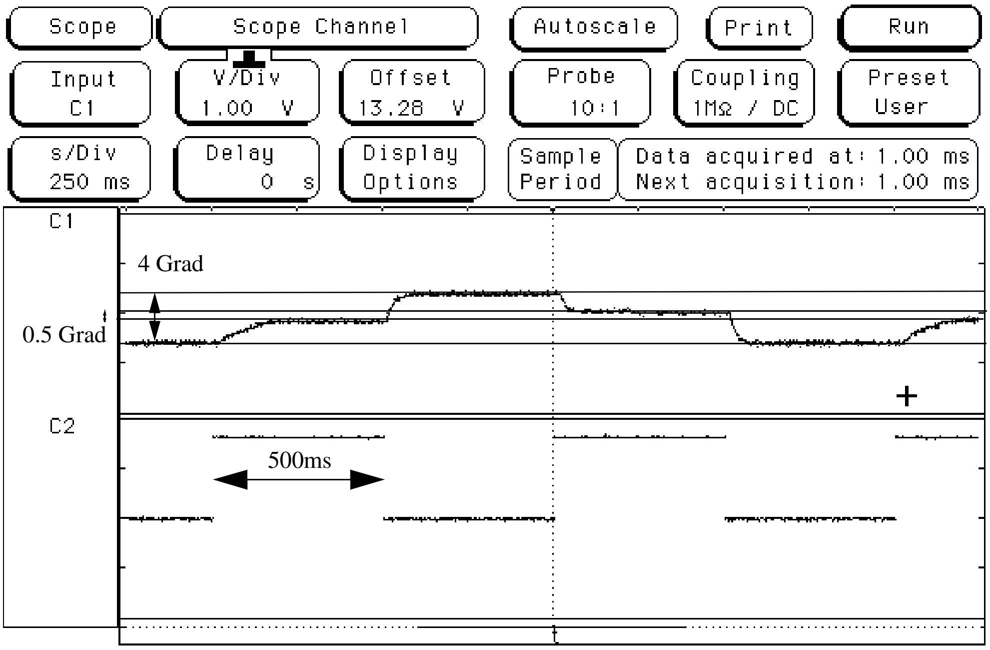


Abbildung 66. **Messung bei einer Bewegung von +-2Grad**

- ca. 0.5 Grad Spiel

B.2 Messungen des AD-Wandlers

Mit einem Funktionsgenerator wurde eine Rampe auf einen Eingang des AD-Wandlers gegeben. Die gewandelten Werte wurden abgespeichert und am Schluss auf den PC hochgeladen.

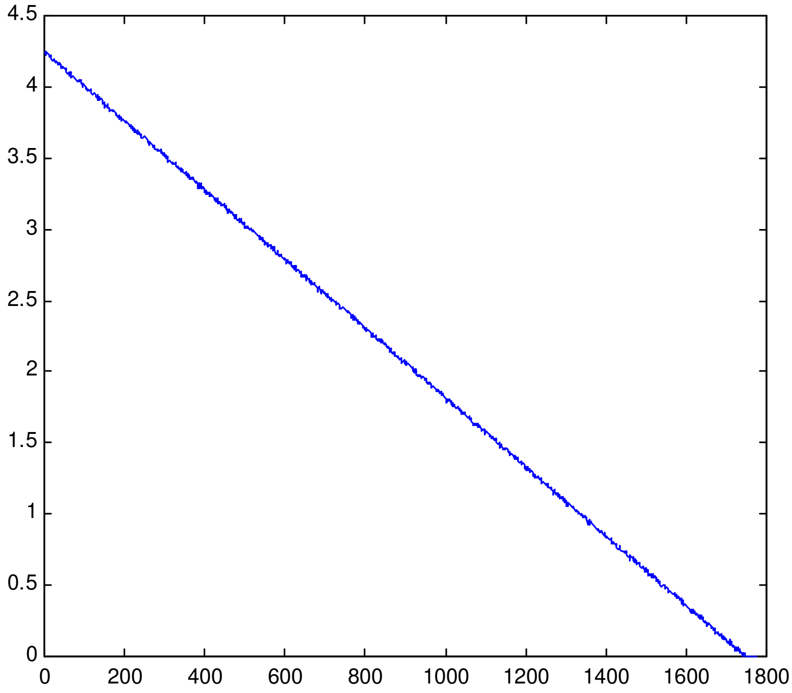


Abbildung 67. Rampenmessung mit AD-Wandler

Verarbeitet und in eine Kurve gezeichnet wurden die Messresultate mit Matlab. Das Rauschen hat eine Amplitude von ca. 20 mV (0.5 Prozent). Wichtig für die Wandlung ist eine stabile Referenzspannung. In Tabelle 16 ist Vref die Ausgangsspannung des Referenzspannungsteils REF-02 und V5 ist der Ausgang des Spannungsstabilisators LM2940.

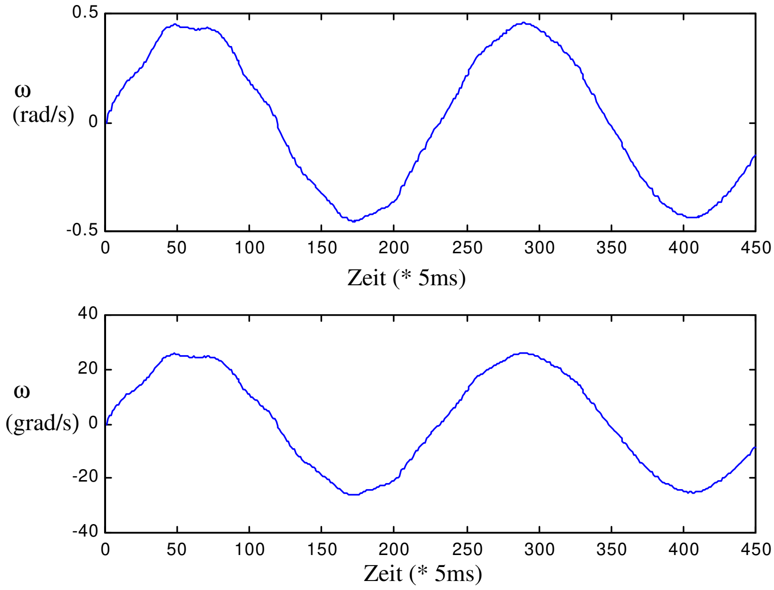
TABELLE 16. Verlauf der regulierten Spannung bei Abfall der Batteriespannung (gemessen)

VDC	V5	Vref
10	5.03	5.04
8	5.03	5.05
7.2	5.05	5.06
7	5.04	4.83
6.8	5.03	4.62
6.6	5.03	4.41
6.4	5.01	4.24
6.2	5.03	4.06
6	4.99	3.88
5.8	4.79	3.66

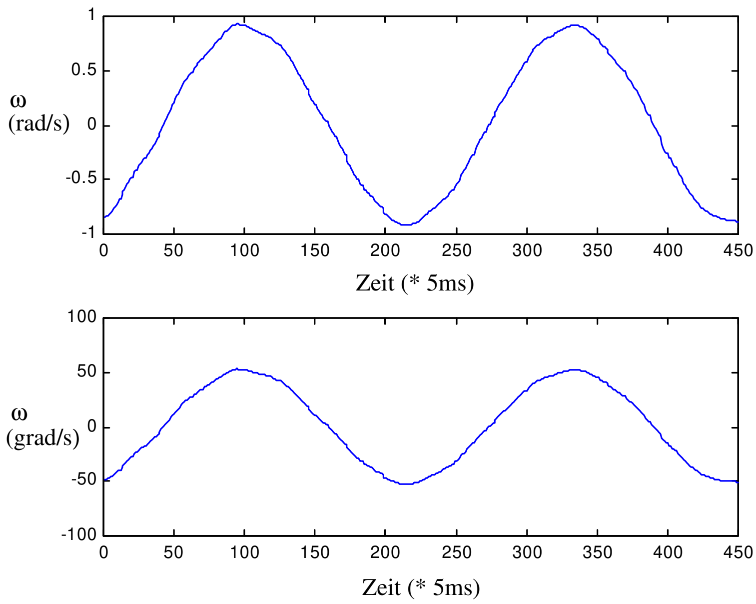
B.3 Messungen des Gyros

Die Messanordnung ist in Kapitel 3 beschrieben, und es wurden folgende Kurven gemessen:

5 Grad Anfangsauslenkung:



10 Grad Anfangsauslenkung:



15 Grad Anfangsauslenkung:

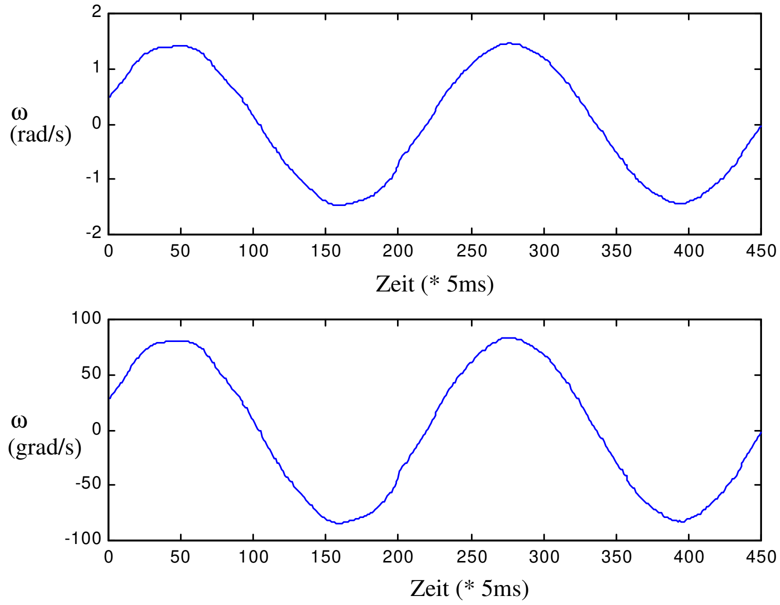


TABELLE 17. Vergleich von Soll- und Messwerten

Anfangsauslenk-ung(Grad)	**Omega (soll,rad/s)**	**Messwert (rad/s)**
5	0.471	0.46
10	0.942	0.93
15	1.400	1.46

Die Genauigkeit dieser Messwerte,ca. 4 Prozent, liegt im Bereich der Genauigkeit der Messanordnung. Die Abweichungen von der Sinusform entstehen zum Teil durch überlagerte Schwingungen.

B.4 Traces in freier Fahrt

Die folgenden Aufzeichnungen wurden durch ein Hochladen der Daten vom Prozessorboard auf den PC aufgezeichnet. Es sind Traces, welche in freier Fahrt mit Stabilisation durch den Kaskadenregler gemacht wurden. Die erste Kurve zeigt den Messwert des Gyrometers, die zweite den Lenkwinkel, die dritte den errechneten Neigungswinkel und die vierte die Geschwindigkeit. Die Kurven erstrecken sich jeweils über 10 Sekunden.

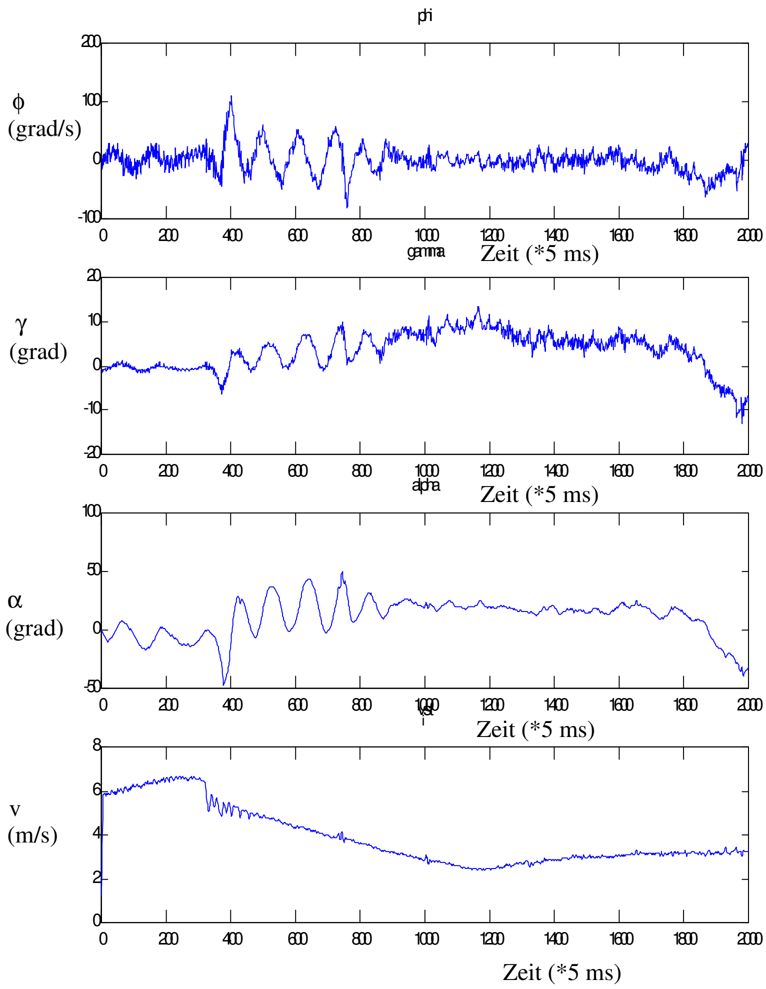


Abbildung 68. Geschwindigkeit von 6.5 bis 3 m/s sinkend

Die kleinen Schwingungen (Abbildung 68) auf der Winkelgeschwindigkeit entstehen durch das Messrauschen und die Vibrationen während der Fahrt. 6.5 m/s ist die Maximalgeschwindigkeit des Motorrades. Beim starken Abbremsen von der Maximalgeschwindigkeit auf ca. 4 m/s begann das Motorrad leicht zu schwanken und beruhigte sich dann wieder.

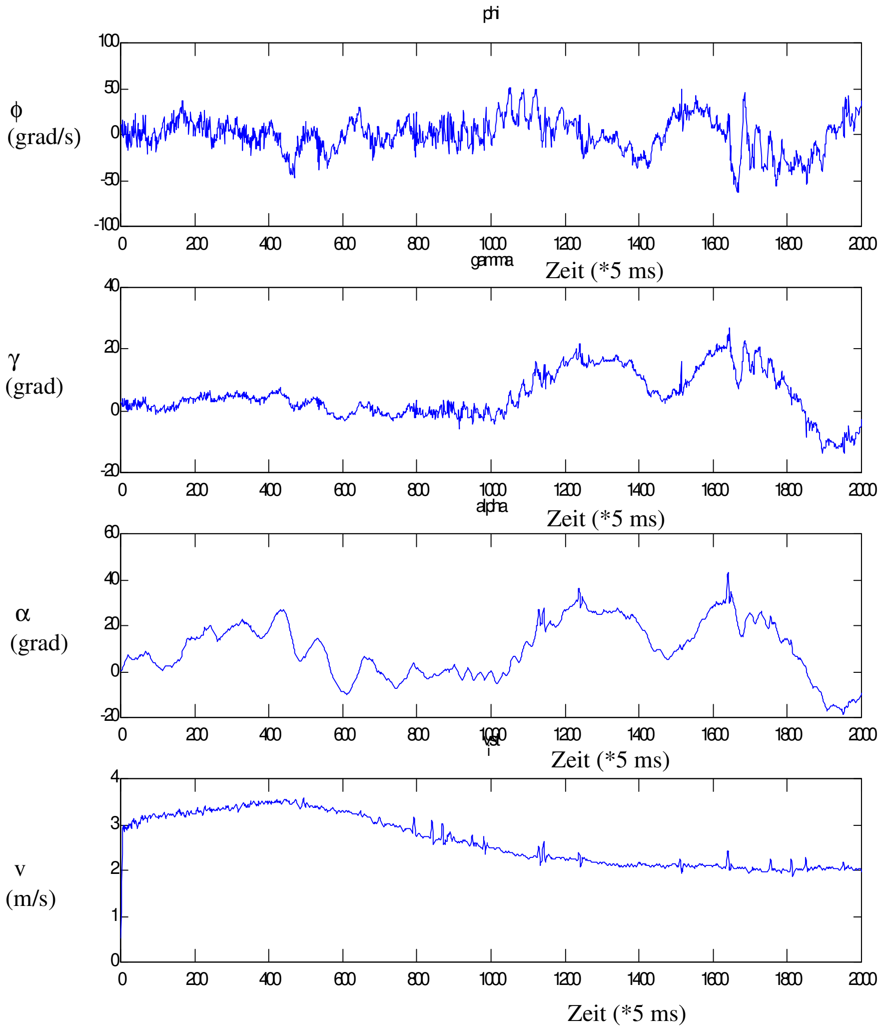


Abbildung 69. Geschwindigkeit von 3.5 bis 2 m/s sinkend

Dies (Abbildung 69) ist ein sehr stabiler Teil des Geschwindigkeitsspektrums. Die Kurven zeigen eine langgezogene, schnelle Rechtskurve und dann ein gerades Stück mit einer Verkleinerung der Geschwindigkeit. Dann folgen zwei langsamere Rechtskurven und zum Schluss noch eine Linkskurve.

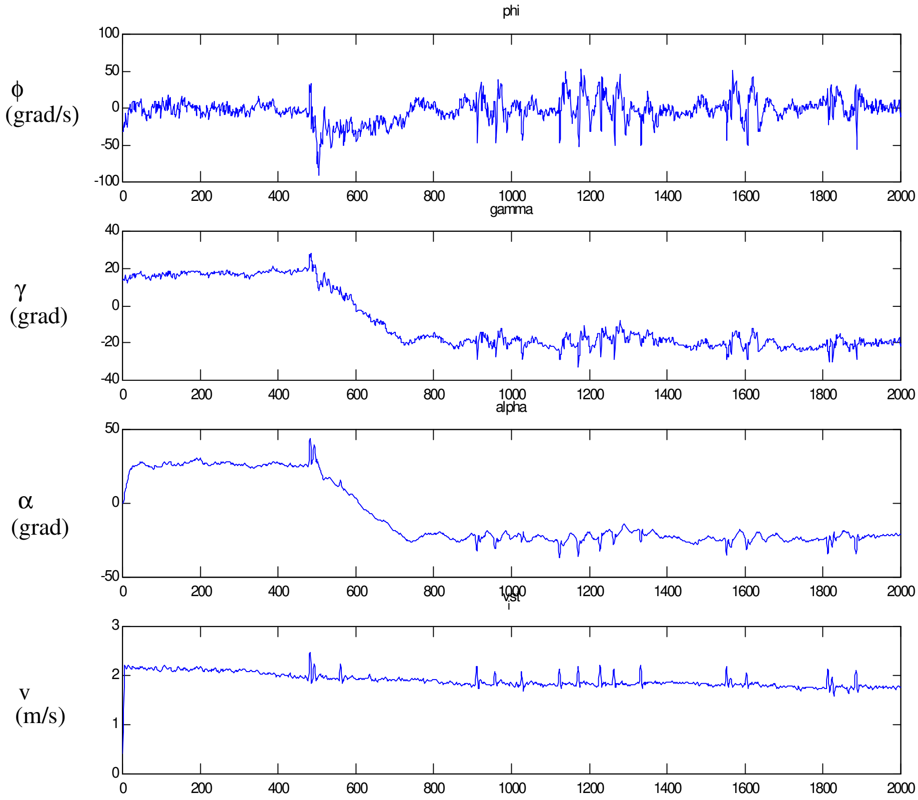


Abbildung 70. Geschwindigkeit von 2.2 bis 1.8 m/s sinkend

Abbildung 70 zeigt eine sehr stabile Fahrt. Der Übergang von einer Rechtskurve auf eine Kreisfahrt gegen Links ist gut zu sehen.

Bei den folgenden Kurven wurde zur genaueren Bestimmung der Geschwindigkeit eine Kurve eingefügt, welche lediglich die Nachkommastelle für Geschwindigkeiten bis 1.6 m/s zeigt. Auf die Alpha-Kurve wurde verzichtet.

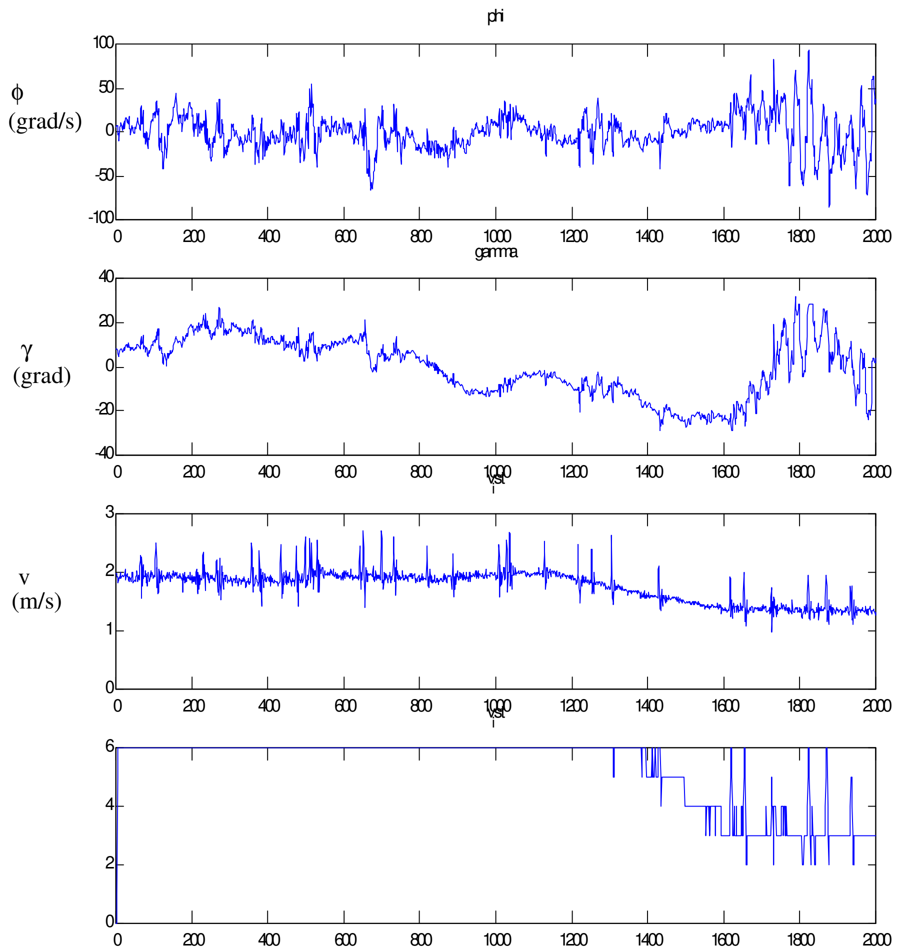


Abbildung 71. **Geschwindigkeit von 1.8 bis 1.3 m/s sinkend**

Gegen Ende der Kurven in Abbildung 71 kommt das Motorrad an die Grenzen der Stabilität. Bei 1.3 bis 1.4 m/s beginnt es sich aufzuschwingen. Es fällt allerdings nicht um, sondern schüttelt lediglich.

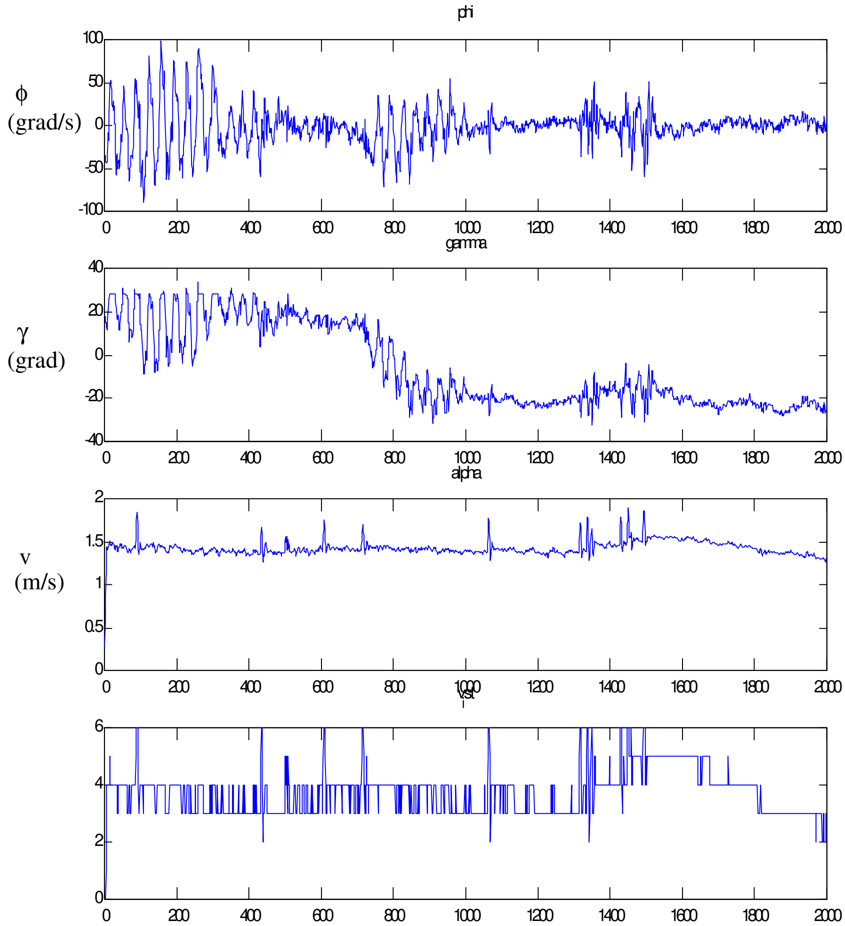


Abbildung 72. **Geschwindigkeit von 1.4 bis 1.2 m/s sinkend**

In Abbildung 72 sieht man, wie sich das Motorrad aufschwingt und auch wieder beruhigt. Die ganze Fahrt ist an der Minimalgeschwindigkeit.

B.5 Simulationen

B.5.1 Kaskadenregler

Für verschiedene Geschwindigkeiten wurden Simulationen gemacht. Zusätzlich zu Lenkwinkel, Winkelgeschwindigkeit und Neigungskurven wurden 3D-Simulationen angefügt.

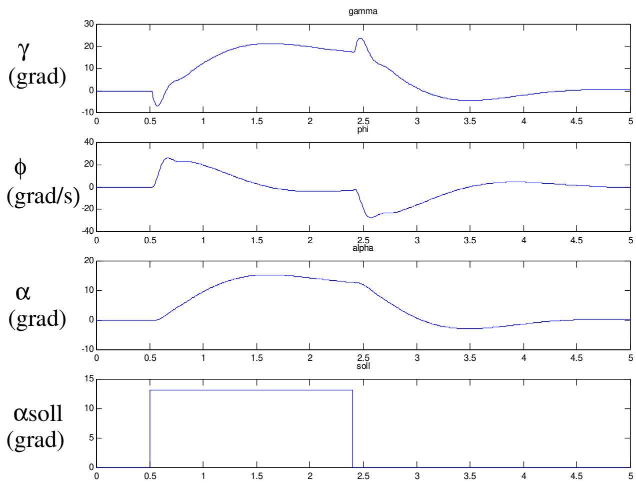


Abbildung 73. **Simulation bei v=1.5 m/s**

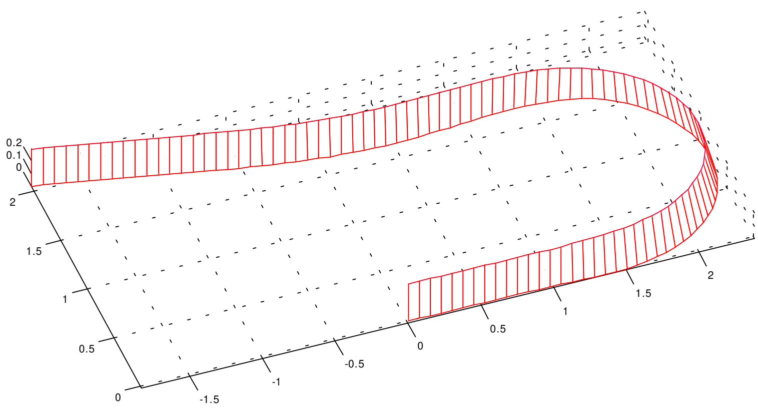


Abbildung 74. **3D-Simulation bei v=1.5 m/s**

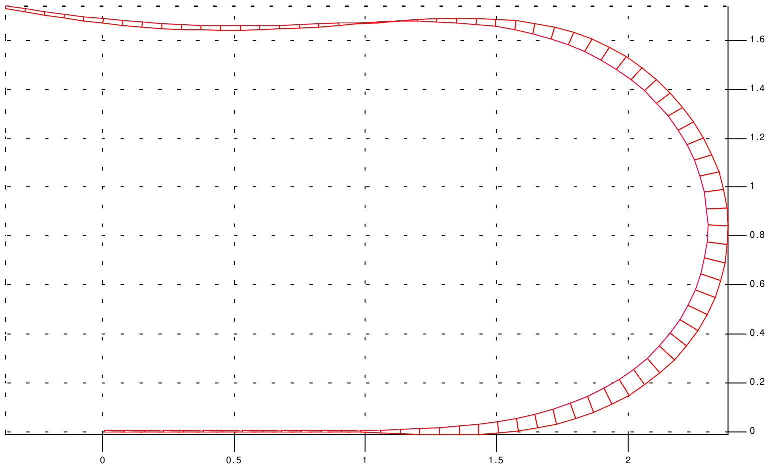


Abbildung 75. 3D-Simulation bei v=1.5 m/s, Sicht von oben

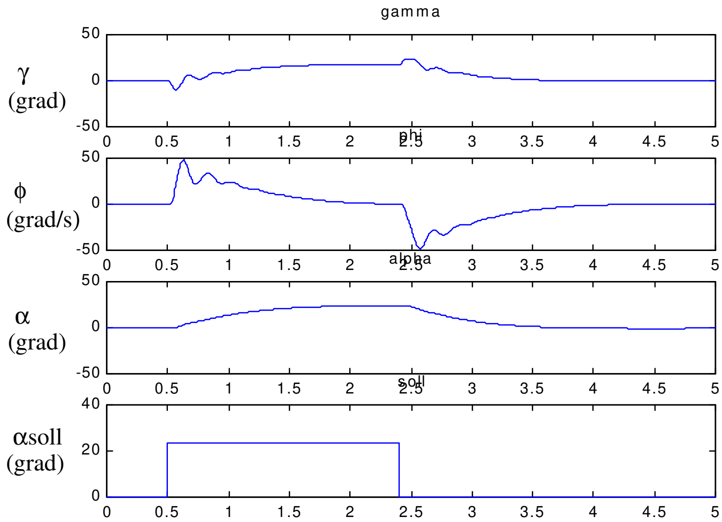


Abbildung 76. Simulation bei v=2 m/s

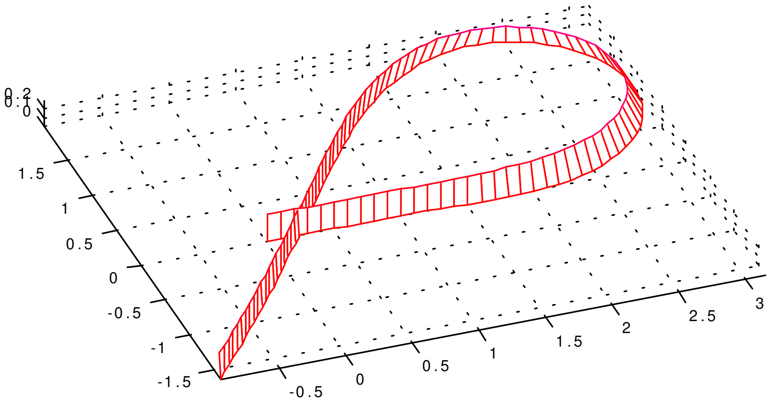


Abbildung 77. **3D-Simulation bei v=2 m/s**

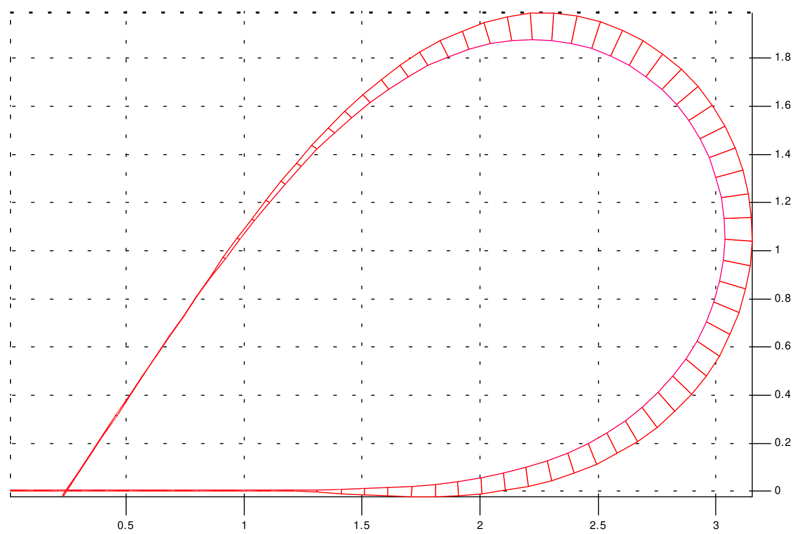


Abbildung 78. **3D-Simulation bei v=2 m/s, Sicht von oben**

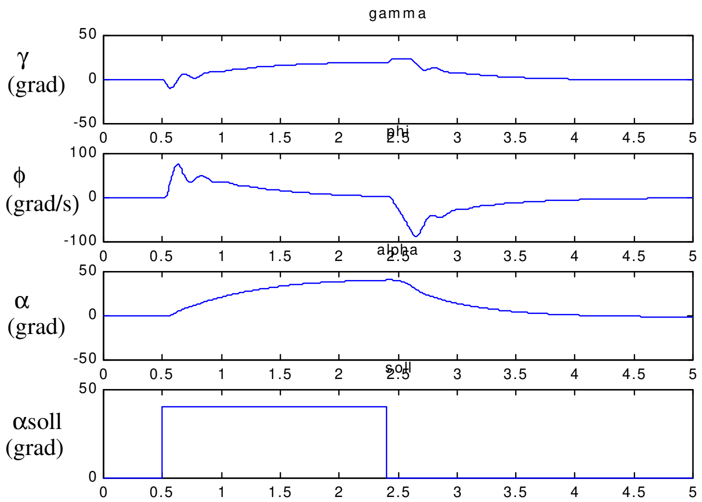


Abbildung 79. Simulation bei v=2.5 m/s

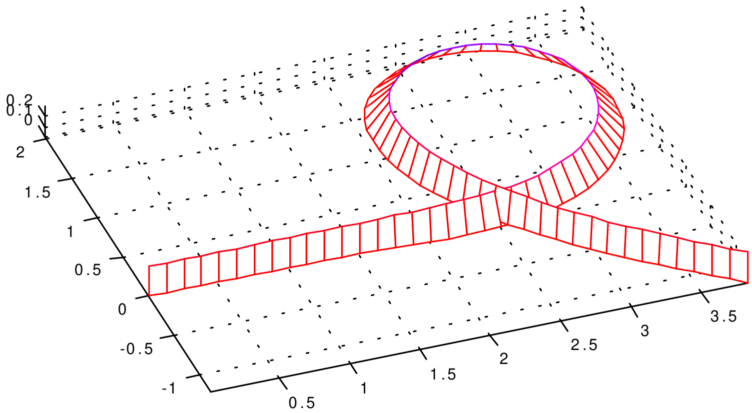


Abbildung 80. 3D-Simulation bei v=2.5 m/s

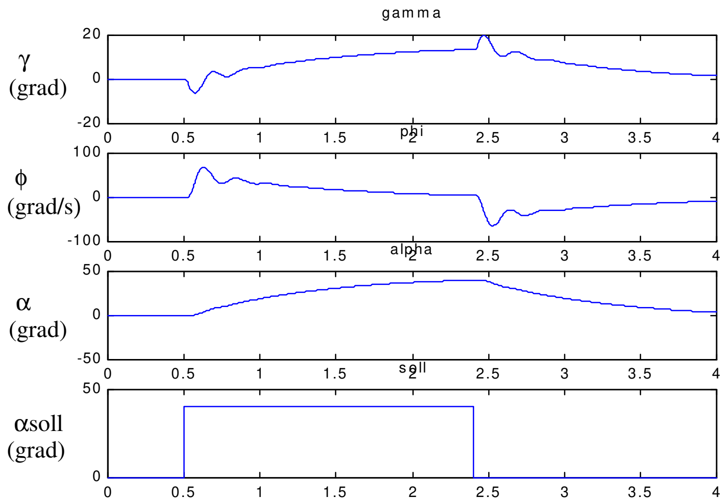


Abbildung 81. Simulation bei v=3 m/s

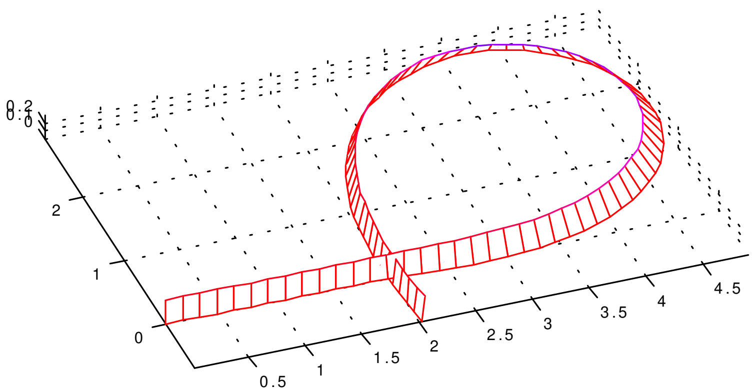


Abbildung 82. 3D-Simulation bei v=3 m/s

B.5.2 PI-Regelung für verschiedene Servos

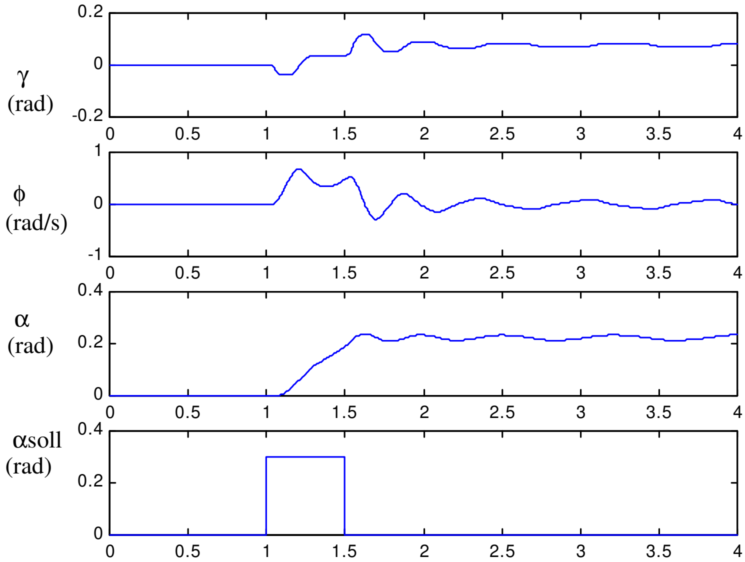


Abbildung 83. **Sprung für einen Servo mit 1 Grad Spiel**

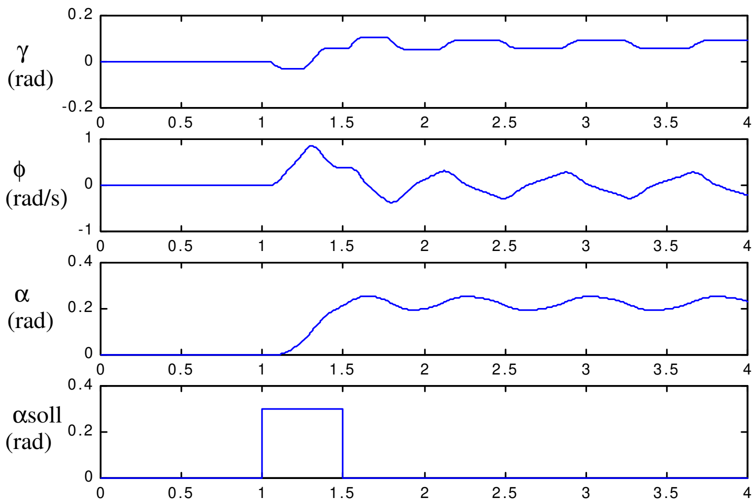


Abbildung 84. **Sprung für einen Servo mit 1 Grad Spiel**

Es ist klar ersichtlich, dass ein grosses Spiel am Servo zu einem schwingen des Motorrades führt. Die Regelung beginnt, zwischen den beiden Grenzen des Spiels hin und her zu schalten.

B.6 Analyse der PI-Regelung im Frequenzbereich

Für verschiedene Geschwindigkeiten wurden die Frequenzgänge der PI-Regelung aufgezeichnet.

B.6.1 Frequenzgang mit Polfestlegung bei Omega gleich 4

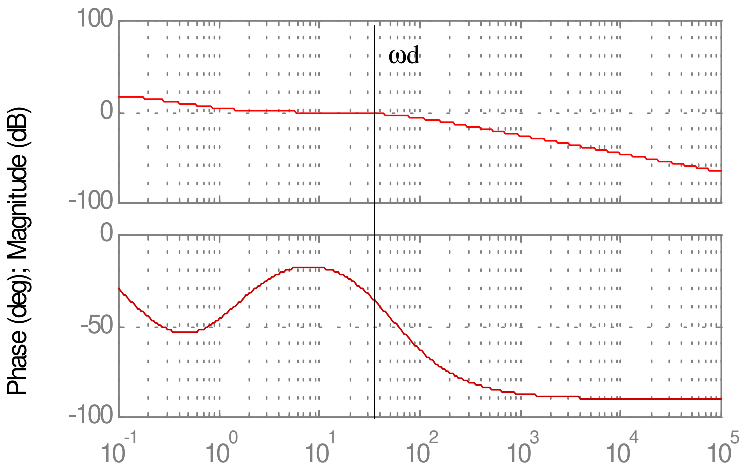


Abbildung 85. Frequenzgang bei v=0.5 m/s

- Phasenreserve ~140 Grad

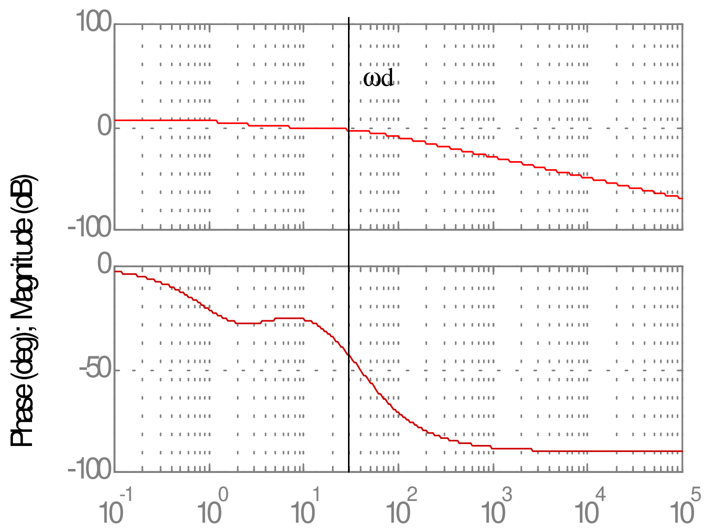


Abbildung 86. Frequenzgang bei v=1.5 m/s

- Phasenreserve ~130 Grad

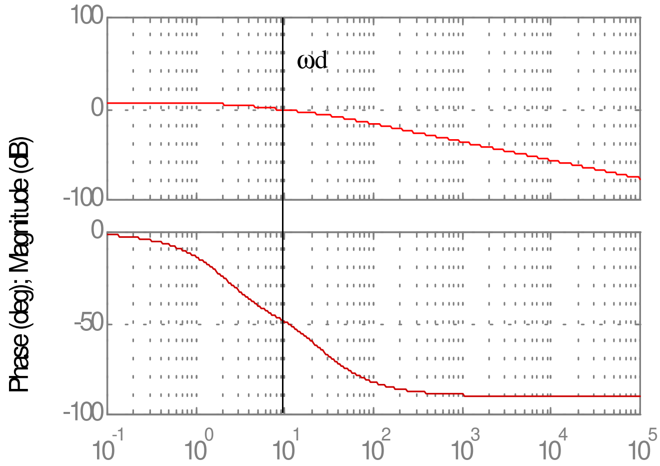


Abbildung 87. Frequenzgang bei v=2.5 m/s

- Phasenreserve ~130 Grad

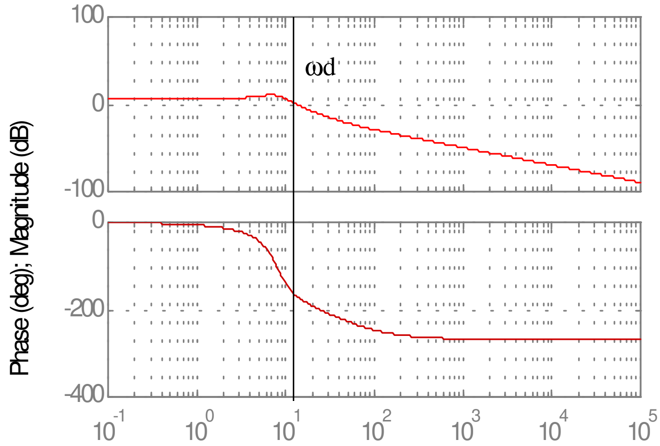


Abbildung 88. Frequenzgang bei v=3.5 m/s

- Phasenreserve ~0 Grad

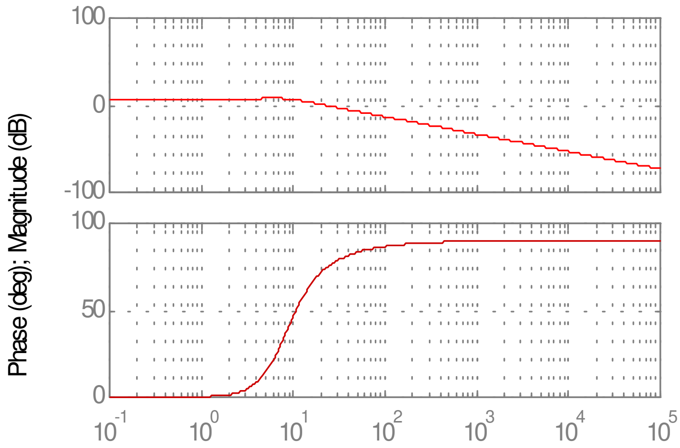


Abbildung 89. **Frequenzgang bei v=4.5 m/s**

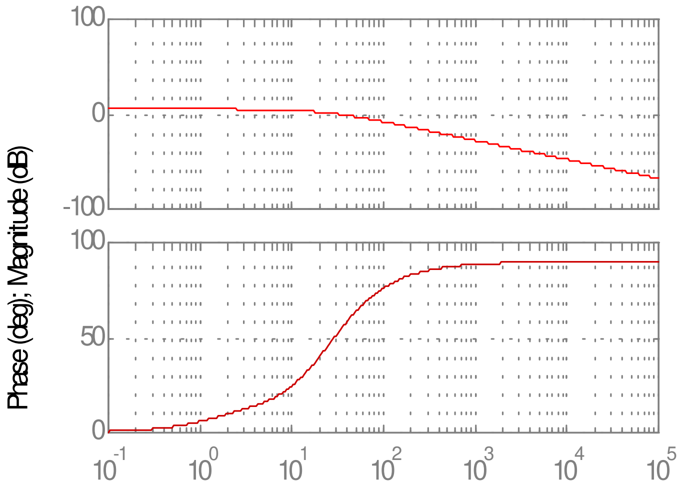


Abbildung 90. **Frequenzgang bei v=5.5 m/s**

Bei der Polfestlegung mit Omega gleich 4 wird das System ab einer Geschwindigkeit von 3.5 m/s instabil. Deshalb wurde der Frequenzgang auch für ein Omega von 8 berechnet.

B.6.2 Frequenzgang bei Omega gleich 8

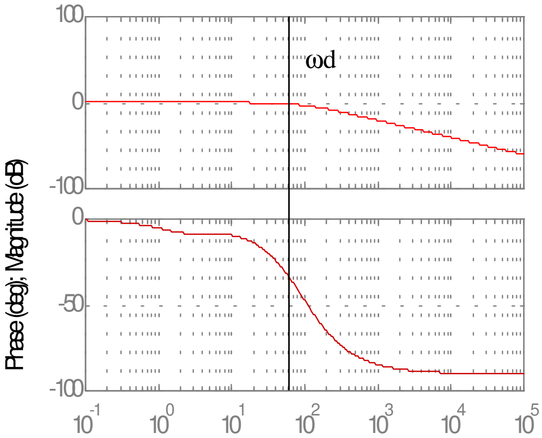


Abbildung 91. Frequenzgang bei v=0.5 m/s

- Phasenreserve ~150 Grad

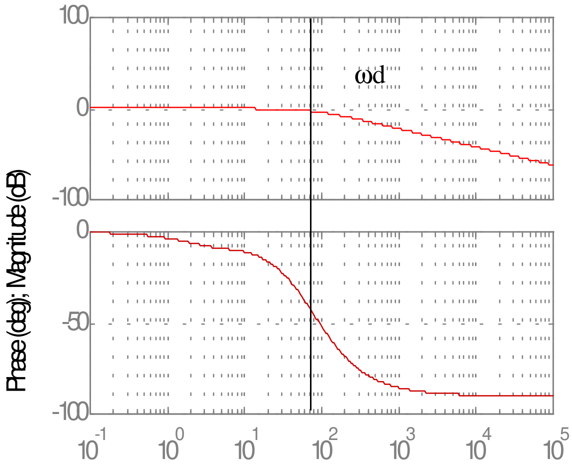


Abbildung 92. Frequenzgang bei v=1.5 m/s

- Phasenreserve ~140 Grad

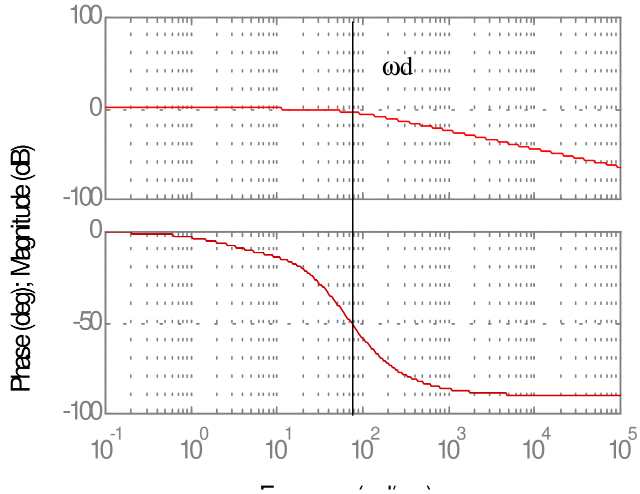


Abbildung 93. **Frequenzgang bei v=2.5 m/s**

- Phasenreserve ~130 Grad

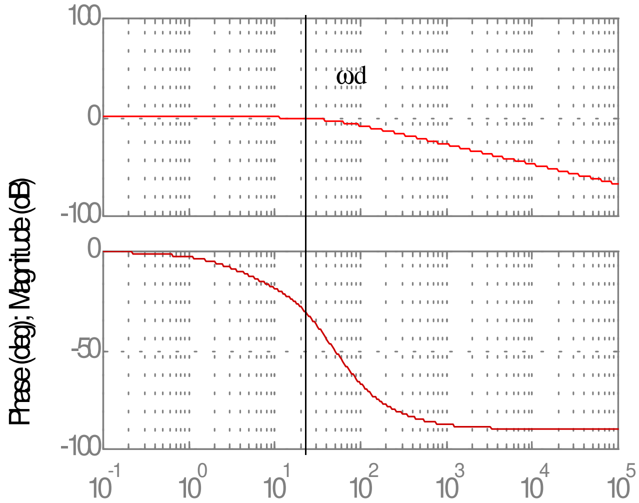


Abbildung 94. **Frequenzgang bei v=3.5 m/s**

- Phasenreserve ~150 Grad

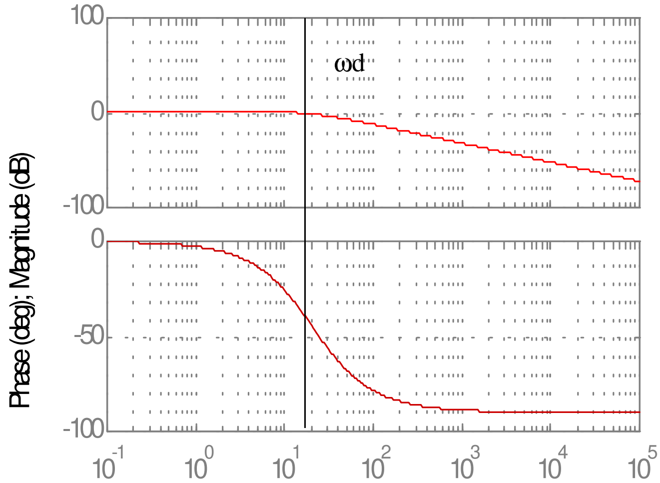


Abbildung 95. Frequenzgang bei v=4.5 m/s

- Phasenreserve ~140 Grad

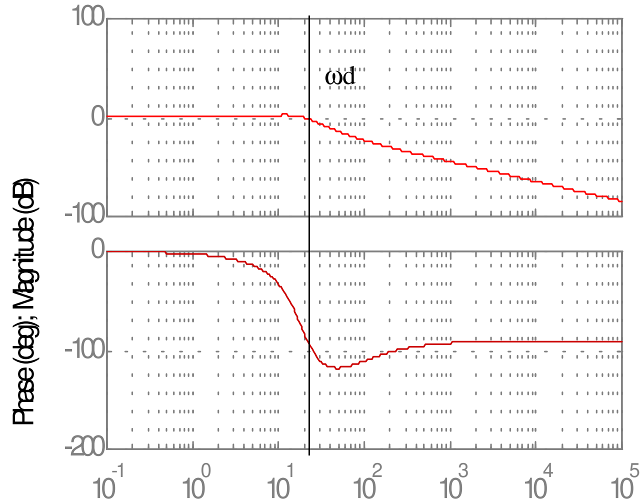


Abbildung 96. Frequenzgang bei v=5.5 m/s

- Phasenreserve ~90 Grad

Mit der Polfestlegung bei Omega gleich 8 wird das System im ganzen Geschwindigkeitsbereich stabil. Die Phasenreserve beträgt immer mindestens 90 Grad.

B.7 Messungen des Inklinometers

B.7.1 Messungen am Institut für Automatik (IfA)

Zum Bestimmen der dynamischen Eigenschaften des Inklinometers wurden Messungen in Zusammenarbeit mit dem Projekt invertiertes Pendel [UBa97] am Institut für Automatik gemacht. Es handelt sich dabei um ein Metallgestell, welches auf Rädern mit Antriebsmotoren balanciert wird.

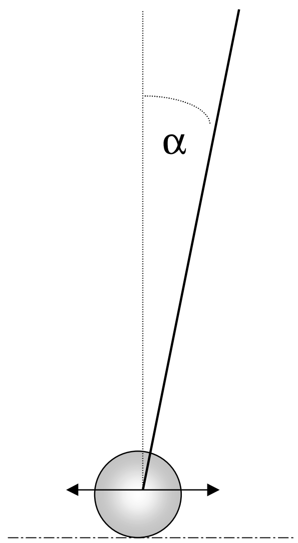


Abbildung 97. Skizze des invertierten Pendels

Die Eigenschaften dieses Pendels sind ähnlich wie jene des Motorrades. Es steht ein absoluter Neigungssensor zur Verfügung, mit welchem das Pendel stabilisiert wird. Abbildung 98 zeigt den Vergleich des Ausgangswertes des Inklinometers mit dem Ausgang des absoluten Neigungssensors. Die Messwerte sind in Radian angegeben. Man sieht deutlich, dass das Inklinometer weit ausschlägt, obwohl die wirkliche Neigung sehr klein ist.

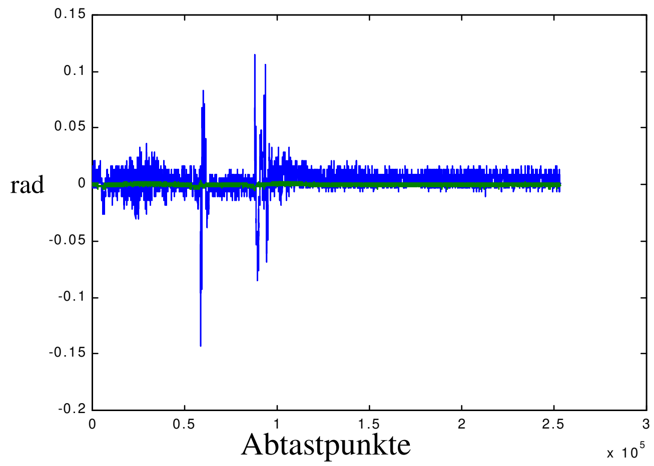


Abbildung 98. Inklinometer und wirkliche Neigung

Die Ausschläge des Inklinometers stammen von der seitlichen Beschleunigung der Räder des Pendels, welche zur Stabilisierung eingesetzt wird. Abbildung 99 zeigt die Winkelbeschleunigung der Räder des Pendels.

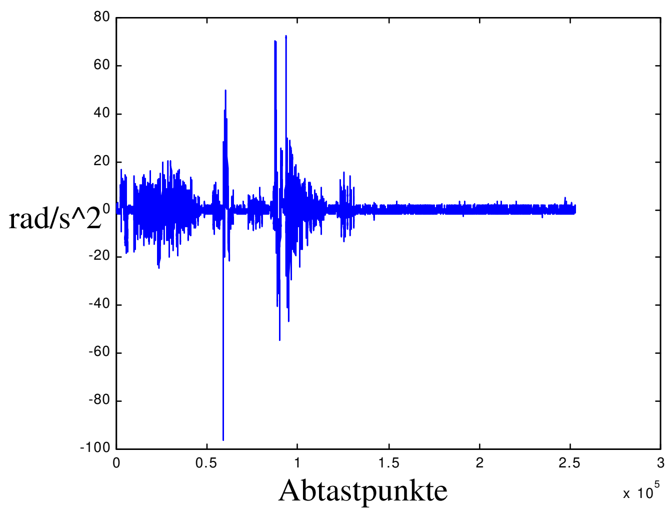


Abbildung 99. Winkelbeschleunigung der Räder

Mit den Formeln aus Kapitel 3.1.1 wurde versucht, die Neigung aus dem Inklinometersignal zu berechnen (Abbildung 100). Dies Misslang jedoch, weil die Störungen beim Inklinometer zeitlich zur Ursache der Ausschläge versetzt sind.

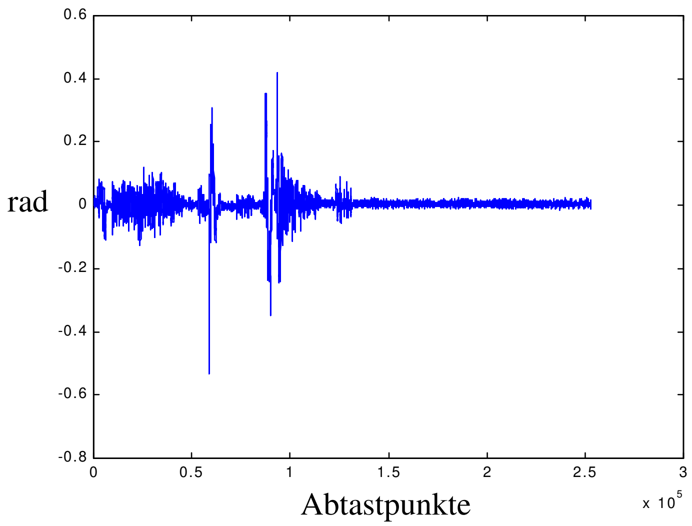


Abbildung 100. Korrigiertes Inklinometersignal

Die gleiche Messung wurde wiederholt und die Resultate sind in den drei folgenden Abbildungen aufgezeichnet.

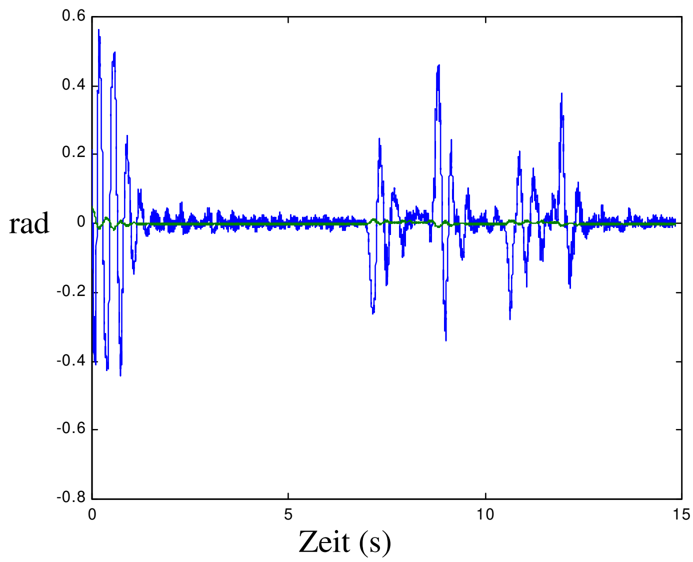


Abbildung 101. Inklino und realer Winkel

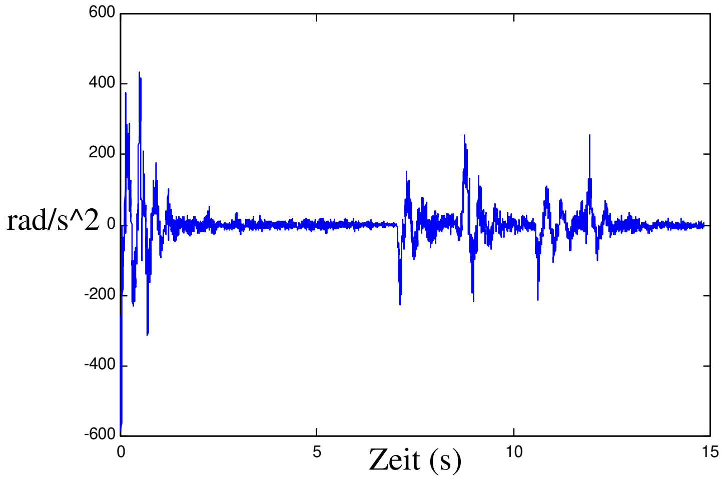


Abbildung 102. Winkelbeschleunigung

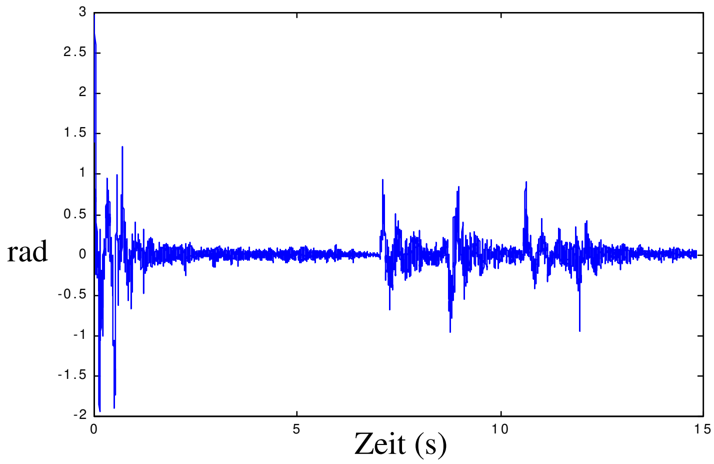


Abbildung 103. Korrigiertes Inklinometersignal

ANHANG C

Bedienungsanleitung

C.1 Checkliste

Vor Inbetriebnahme:

- Sind alle Stecker am richtigen Ort eingesteckt? (vgl. Abbildung 108)
- Haben beide Batterien genug Spannung? (rote (Hauptakku) und grüne LED (für Analogteil), Spannungsmessung: min. 7.2 V)
- Sind die Muttern an der Achse des Vorderrades und die Kreuzschrauben an der Vordergabel angezogen?
- Sind alle Kabel so verlegt, dass sie bei einem Sturz nicht abgerissen werden?
- Ist das Servogetriebe in Ordnung? (Kein Knacken beim Drehen des Servohebels.)

Vor Download des Programmes:

- Ist die Fernsteuerung eingeschaltet und hat sie genug Energie? (Level Meter in oberer Hälfte der Skala)
- Ist der Geschwindigkeitshebel auf “Stop” und der Umschalter auf “Steuern”?

C.2 Bedienung

C.2.1 Einschalten

Klappen sie den Hauptschalter (siehe Abbildung 107) gegen hinten, beide Spannungsversorgungen werden miteinander aktiviert. Nach dem Einschalten dauert es einen Augenblick, bis die Elektronik den Betriebszustand erreicht, vor allem die Interfaceelektronik des Gyrometers braucht ihre Zeit.

C.2.2 Download

Das Downloadkabel wird so an den Background Mode Connector angeschlossen, dass die rote Seite des Kabels auf die rechte Seite des Motorrades kommt. Die zwei Konnektoren, welche auf der roten Seite des Kabels sind, bleiben unverbunden. Nach dem Starten des Debuggers wird mit dem Menüpunkt *Target->Load* das Programm auf den Prozessor geladen. Durch drücken der "*R*"- Taste wird das Programm gestartet, eventuell muss noch ein Datenfenster angeklickt werden, damit der Befehl wirklich ausgeführt wird. Es besteht die Möglichkeit, entweder den Signalpfad "normal" oder "trace" auszuwählen. In beiden Varianten heisst das Programm "main.abs"

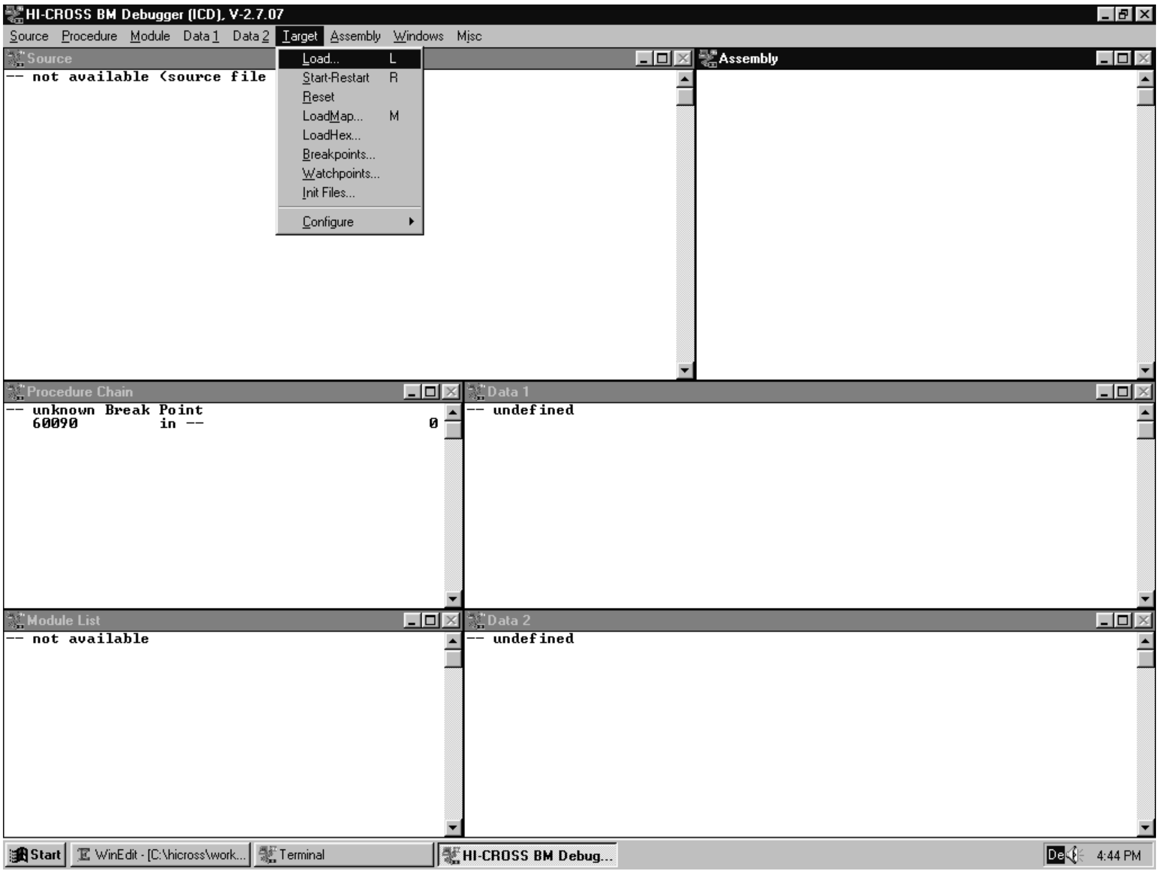


Abbildung 104. Laden des Programmes im Debugger

Achten sie darauf, dass vor dem Starten des Programmes die Fernsteuerung eingeschaltet ist und alle Hebel in der gewünschten Stellung sind.

Möchte man die Tracingfunktion benützen, so muss man unter dem Menüpunkt *Misc->Terminal* des Debuggers "*File+Window*" wählen (Abbildung 105). So erscheinen Daten, die vom Microcontroller an den PC geschickt werden, nicht nur im Terminalfenster, sondern werden auch abgespeichert. Es muss das Programm, welches für das Tracen ausgelegt ist, geladen werden.

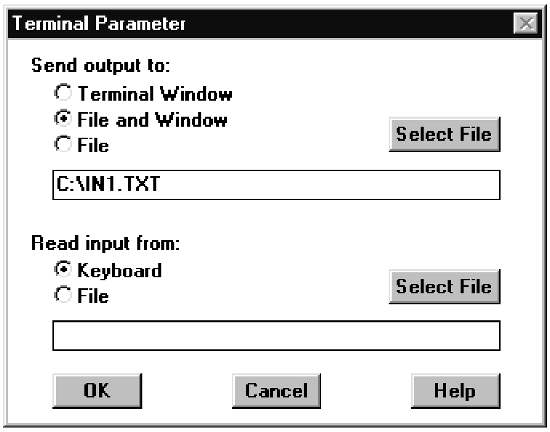


Abbildung 105. **Auswahl von "File and Window"**

C.2.3 Abhängen des Kabels

Um das Downloadkabel abzuhängen, muss der Releaseschalter gegen hinten gezogen werden. Während man diesen Schalter betätigt, kann das Kabel abgezogen werden.

C.2.4 Anfahren

Das Anfahren ist am einfachsten, wenn man mit der einen Hand das Motorrad hält und mit der anderen die Fernsteuerung. Man schaltet den Umschalter an der Fernsteuerung auf "Regeln" und schiebt, am besten mit dem Kinn, den Geschwindigkeitshebel (siehe Abbildung 106) nach oben. Beschleunigt das Motorrad, so kann man es loslassen.

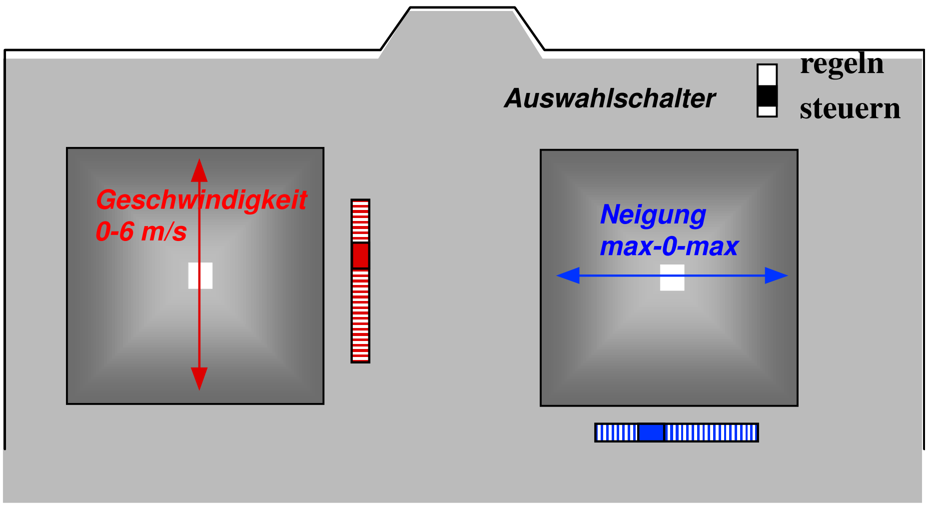


Abbildung 106. **Ansicht der Fernsteuerung**

C.2.5 Upload von Messresultaten

Hat man mit der Tracingfunktion Werte abgespeichert, so stoppt das Motorrad am Ende der Aufzeichnungen. Dabei wird der Motor abgestellt und das Motorrad fällt um. Dann kann das Motorrad an den laufenden Debugger wieder angeschlossen werden. Der Releaseschalter wird wieder betätigt und währenddessen der BDI-Stecker angeschlossen. Die Fehlermeldung, welche beim Abhängen des Interfaces auf dem Bildschirm erschien, sollte immer noch da sein. Durch drücken der “Enter”-Taste kann der Debugger wieder in seine normale Funktion übergehen. Sollte ein Interrupt aufgetreten sein, so kann man mit der “C”-Taste (Continue) versuchen das Programm fortzusetzen. Im Terminalwindow kann man nun die Übertragung der Werte verfolgen.

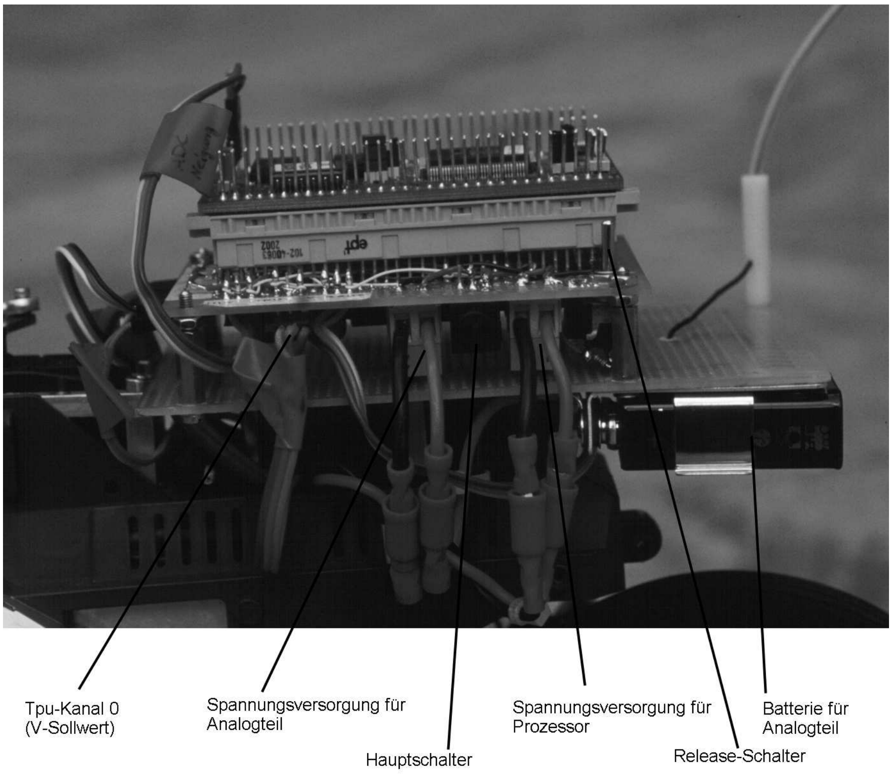


Abbildung 107. Elektronik von der Seite

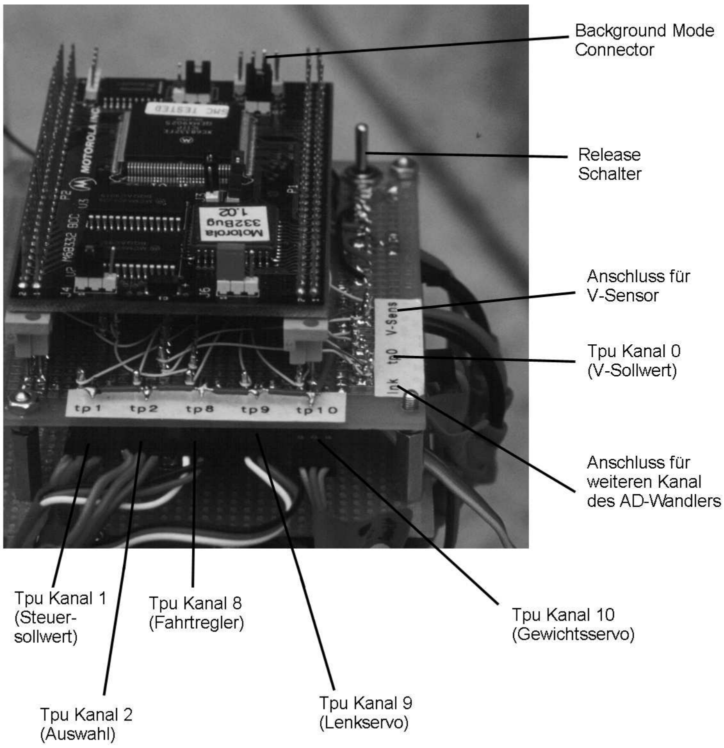


Abbildung 108. **Elektronik von Vorne**

ANHANG D

Variablen und Abkürzungen

Modellierung des Motorrades

- *a*: Neigungswinkel (positiv gegen Rechts)
- *g*: Lenkwinkel (positiv gegen Rechts)
- *h*: Höhe des Schwerpunktes
- *l*: Abstand der Auflagepunkte der Räder
- *m*: Masse des Motorrades
- *g*: Normwert der Fallbeschleunigung
- *Lv,Lh*: Drall des Vorder- und Hinterrades
- *L*: Gesamtdrall
- *Jv,Jh*: Trägheitsmoment des Vorder- und Hinterrades
- *rk*: Radius der Kreisfahrt
- *v*: Geschwindigkeit
- *J*: Trägheitsmoment des Motorrades um Achse durch Auflagepunkte
- *Mg*: Drehmoment der Gewichtskraft
- *Mz*: Drehmoment der Fliehkraft
- *Mk*: Drehmoment durch Kurvenfahrt
- *Mv*: Drehmoment durch Lenkbewegung
- *hink*: Höhe des Inklinometers

Messung des Gyrometers

- *L*: Länge des Pendels
- $\theta 0$: Anfangsauslenkung des Pendels
- ωmax: Maximale Winkelgeschwindigkeit

Regelung

- *kp0*: Verstärkungsfaktor der inneren PI-Regelung
- *Ti0*: Nachstellzeit der inneren PI-Regelung
- *Ts*: Abtastzeit der diskreten Regelung

- *kp1*: Verstärkungsfaktor der äusseren PI-Regelung
- *Ti1*: Nachstellzeit der äusseren PI-Regelung
- *vf*: Parameter zur Berechnung der Neigung aus dem Lenkwinkel
- Ω: Regelgeschwindigkeit
- *D*: Dämpfung

Prozessor

- *CPU*: Central Processing Unit
- *TPU*: Time Processing Unit
- *QSM*: Queued Serial Module
- *SIM*: System Integration Module

Kodierung

- *PWM*: pulsweitenmoduliert