

Semesterarbeit SS 05

# Drehzahlbestimmung

Studenten:

Markus Gaugler, Christoph Frehner

Betreuer:

Prof. Dr. Guido M. Schuster

HSR

Hochschule für Technik Rapperswil

Abteilung Elektrotechnik

Rapperswil, Juli 05

## Inhaltsverzeichnis:

1.	Aufgabenstellung .....	5
1.1.	Rahmenbedingungen: .....	5
1.2.	Kurzbeschreibung .....	5
1.3.	Aufgaben .....	6
1.4.	Erwartete Ergebnisse .....	6
1.5.	Arbeitsweise .....	7
2.	Abstract .....	8
3.	Projektplan .....	10
4.	Theorie Lichtmaschine .....	11
4.1.	Aufbau .....	12
4.2.	Freilauflichtmaschine .....	14
5.	Messdatenerfassung .....	16
5.1.	Hardware .....	16
6.1.	Gesamtaufbau .....	18
6.2.	TMS320C6711 DSK .....	19
6.3.	DSP Global ProtoPlus Karte .....	20
6.4.	PCM3003 Audio Tochterkarte .....	20
6.5.	EA eDIP240-7 .....	21
6.6.	Spannungsregler .....	21

7.	Theorie Signalaufarbeitung.....	23
7.1.	Allgemeines.....	23
7.2.	Einlesen des Signals.....	23
7.3.	Korrelation.....	25
7.4.	Transformation.....	26
7.5.	Filterung.....	27
8.	Viterbi-Algorithmus.....	29
8.1.	Allgemeines.....	29
8.2.	Hidden Markow Modell.....	29
8.3.	Veranschaulichung.....	30
8.4.	Funktionsweise.....	30
8.5.	Implementation des Viterbi-Algorithmus.....	32
9.	Implementation unter MATLAB.....	35
10.	Implementation auf DSP-Plattform.....	36
11.	Display.....	38
11.1.	Display Programmierung.....	38
11.2.	SPI Schnittstelle.....	39
11.3.	Initialisierung McBsp als SPI Schnittstelle.....	42
11.4.	Datenübertragungsprotokoll.....	43
11.5.	Makros für die Drehzahlmessung.....	44
12.	Tauglichkeit dieses Messverfahrens.....	46

13.	Summary.....	48
14.	Verbesserungen.....	49
15.	Literaturverzeichnis .....	50
16.	Anhang.....	51

# 1. Aufgabenstellung

## 1.1. Rahmenbedingungen:

Thema: U/min Meter

Studenten: M. Gaugler, C. Frehner

Betreuer: Prof. Dr. Guido M. Schuster

Partner: KKE Keller Konstruktion & Entwicklung

Reiner Keller

Hanfacker 12

CH-8260 Stein am Rhein

tel. +41 052 741 36 25

mobile: +41 076 510 81 00

kke.keller@bluewin.ch

## 1.2. Kurzbeschreibung

Ziel dieser Semesterarbeit ist es, die Drehzahl eines Motors aus der Restwelligkeit der Lichtmaschine des Motors zu bestimmen. Die ermittelte Drehzahl soll dann als Spannung und/oder als Frequenz proportional zur Drehzahl ausgegeben werden. Die Messung sollte alle 100ms aktualisiert werden.

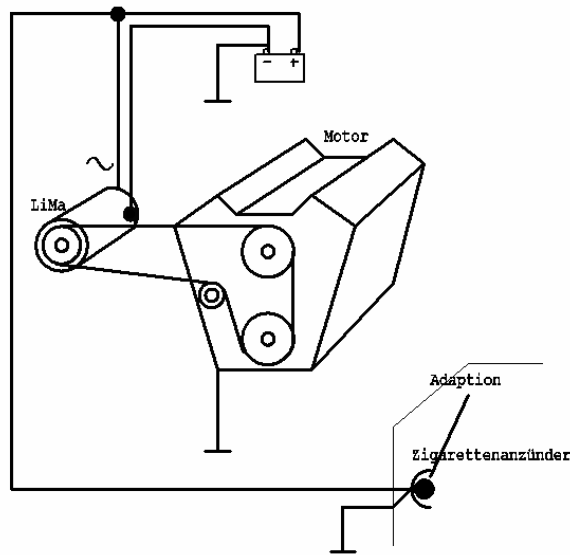


Abbildung 1: Funktionsprinzip

### 1.3. Aufgaben

- Einarbeitung in die Theorie der Lichtmaschinen
- Erfassen von Messdaten
- Algorithmenentwicklung und Selektierung mit Matlab/Simulink
- Echtzeitimplementierung des selektierten Verfahrens auf DSP

### 1.4. Erwartete Ergebnisse

- Dokumentation der Theorie, Algorithmenentwicklung und der Implementierung
- Matlab Programm und C/C++ Programm auf DSP
- Je ein Laborbuch

## **1.5. Arbeitsweise**

- Sie führen ein persönliches Laborbuch, wo Sie aufschreiben wann Sie was für wie lange machen und was die Ergebnisse sind
- Sie schicken vor jedem Treffen eine Agenda und nach dem Treffen ein Protokoll.

## 2. Abstract

Ziel dieser Arbeit ist es einen leicht adaptierbaren Drehzahlmesser zu entwickeln, welcher einfach an die 12 VDC Buchse des Fahrzeuges angeschlossen werden kann und so die aktuelle Drehzahl ermittelt. Dazu sollen an verschiedenen Fahrzeugen die Spannungssignale an den Zigarettenadaptern gemessen und ausgewertet werden. Danach soll mit Hilfe der digitalen Signalverarbeitung ein Programm auf Matlab erstellt werden, mit welchem man die Tauglichkeit dieses Verfahrens in der Praxis testen kann. Am Ende soll ein Prototyp vorliegen, bei welchem die Signalverarbeitung auf einem DSP oder MSP430 erfolgt und welcher ein Ausgangssignal liefert, welches proportional zur Drehzahl ist.

In der heutigen Zeit gibt es verschiedene Möglichkeiten die Drehzahl eines Fahrzeuges, sei es ein Auto oder ein Boot, zu bestimmen. Die einzelnen Verfahren weisen jedoch je nach Einsatzgebiet verschiedenste Vor- und Nachteile auf. So stellt die kompakte Bauweise von Motoren und Motorräumen das Problem, dass sie nur noch schwierig zugänglich sind und einen Einbau von Sensoren für die Drehzahlmessung verunmöglichen. Ein weiterer Punkt stellen die unterschiedlichen Motorenarten, Benzin oder Diesel, dar. So gibt es Verfahren, welche zwar gute Resultate für Fahrzeuge mit Benzinmotoren liefern, ihre Funktionsweise lässt jedoch einen Einsatz bei Dieselmotoren nicht zu.

In dieser Arbeit wurde nun die Möglichkeit untersucht, ob die Drehzahl mit Hilfe der Lichtmaschine im Fahrzeug bestimmt werden kann. Dieses Messverfahren hätte den Vorteil, dass es sich sowohl für Diesel, wie auch Benzin Fahrzeuge eignen würde und dass es ohne zusätzliche Installationen im Maschinenraum direkt in der Fahrgastzelle betrieben werden könnte. Es stellte sich jedoch schnell heraus, dass auch bei diesem Verfahren gewisse Probleme im Wege stehen. So wird bei den modernen Fahrzeugen das Signal von der Lichtmaschine derart gut geregelt, dass das Nutzsignal wesentlich kleiner wird als die Störsignale, welche durch andere elektronischen Verbraucher induziert werden. Dieses Problem setzt ein sehr leistungsstarken Algorithmus voraus, welcher trotzdem noch detektieren kann ob es sich um das gesuchte Signal mit der Information der Drehzahl handelt oder nicht. Ein weiteres Problem stellen die

Freilauflichtmaschinen dar, welche die Information über die Drehzahl des Motors durch ihre Funktionsweise zeitweise vollständig eliminieren. (Siehe Kap. 4 Lichtmaschine).

Für die Realisierung dieses Drehzahlmesssystems wird mittels FFT eine Frequenzanalyse des Eingangssignals durchgeführt. Aus den daraus gewonnenen Informationen wird danach mit Hilfe des Viterbi-Algorithmus die Frequenz ermittelt, welche die Drehzahl der Lichtmaschine enthält. Diese Frequenz wird dann unter Einbezug von der Polpaarzahl des Generators und dem Übersetzungsverhältnis zwischen dem Motorenpulley und dem Lichtmaschinenpulley zur aktuellen Drehzahl umgerechnet. Die Drehzahl wird dann zum einen auf einem Display dargestellt und zum anderen als Spannung ausgegeben, so dass man zusammen mit der Beschleunigung eine Aussage über die Leistung des Motors machen kann.

### 3. Projektplan

Woche		12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
<b>Tätigkeit</b>																
theoretische Grundlagen aufarbeiten	Soll															
	Ist	█														
Hardware Beschaffung/Installation	Soll															
	Ist	█	█													
Einarbeitung Matlab	Soll															
	Ist	█	█													
Messprogramm erstellen	Soll															
	Ist		█	█												
Messdatenerfassung und sortieren	Soll															
	Ist		█	█												
Datenauswertung	Soll															
	Ist		█	█	█											
Erstellen des Detektionsalgorithmus	Soll															
	Ist			█	█	█	█	█	█	█	█	█	█	█	█	█
Austesten des Algorithmus	Soll															
	Ist										█	█	█	█	█	█
Anforderungen an die Plattform erstellen	Soll															
	Ist										█	█	█	█	█	█
Plattform auswählen	Soll															
	Ist										█	█	█	█	█	█
Implementation auf neue Plattform	Soll															
	Ist															
Austesten des Systems	Soll															
	Ist															
Verbesserungen des Systems	Soll															
	Ist															
Erstellen der Dokumentation	Soll															
	Ist															

Erstellung des Projektplans

Abgabetermin: 05.07.05

## 4. Theorie Lichtmaschine

In einem Fahrzeug bezeichnet man den Stromerzeuger als Lichtmaschine oder auch als Alternator. Die Lichtmaschine hat die Aufgabe, die elektronischen Komponenten an Bord eines Fahrzeuges mit Strom zu versorgen und die Autobatterie aufzuladen. Dabei handelt es sich um einen Gleich- oder Drehstromgenerator. Die Lichtmaschine wird mit einem Riementrieb, wie ein Keil- oder Flachriemen, vom Motor angetrieben. Dabei kann nur bei laufendem Motor ein Strom an die Batterie und die angeschlossenen Verbraucher bereitgestellt werden. Die zur Verfügung gestellte Leistung ist gleich der abgegebenen mechanischen Leistung des Motors und wirkt als bremsendes Drehmoment auf den Motor.

Bei Leerlauf des Motors ist die abgebbare elektrische Leistung begrenzt, d.h. kleiner als die theoretisch mögliche Leistung. Wird mehr elektrische Leistung gefordert, so muss die Autobatterie den Differenzstrom bereitstellen.

In den frühen Jahren des Fahrzeugbaus wurden Gleichstromgeneratoren als Lichtmaschinen eingesetzt. Der Nachteil bei diesen Generatoren ist, dass sie erst bei höheren Drehzahlen einen nennenswerten elektrischen Strom produzieren. Dies führte vor allem in der kälteren Jahreszeit bei Fahrten in der Stadt mit niedrigen Drehzahlen, eingeschalteter Heizung und Beleuchtung zu entladenen Starterakkus. Zudem wurden die elektrischen Verbraucher in den Fahrzeugen immer vielfältiger und zahlreicher, was auch zu einem höheren Stromverbrauch führte. Deswegen werden heutzutage nur noch Drehstromgeneratoren eingebaut, da diese bereits bei Leerlaufdrehzahl des Motors einen nennenswerten elektrischen Strom liefern und wesentlich kompakter gebaut werden können. Der 3-Phasen-Drehstrom wird danach einfach durch Hochleistungs-Halbleiterdioden gleichgerichtet.

Die mögliche Abgabeleistung der Lichtmaschine von durchschnittlichen Fahrzeugen liegt bei 2 bis 3 kW. Bei 14 Volt Bordspannung fließt somit ein Strom von bis zu 215 Ampère.

$$I = \frac{P}{U} = \frac{3000W}{14V} \approx \underline{\underline{215A}} \quad (4.1)$$

## 4.1. Aufbau

Ein einfacher Generator besteht aus einem Rotor und einem Stator. Der Rotor besteht aus einer rotierenden Drahtwicklung innerhalb einem Magnetfeld, welches durch ein Polpaar bestehend aus Nord- und Südpol erzeugt wird. Dieses Polpaar bildet auch gleich den Stator. Die Ausgangsspannung ist ein Sinus mit einer Periode pro Umdrehung. Mittels einer Diodenschaltung wird das Wechselnungssignal in ein Gleichnungssignal transformiert. Nach der Gleichrichtung hat das neue Signal 2 Perioden pro Umdrehung. (Abbildung 2).

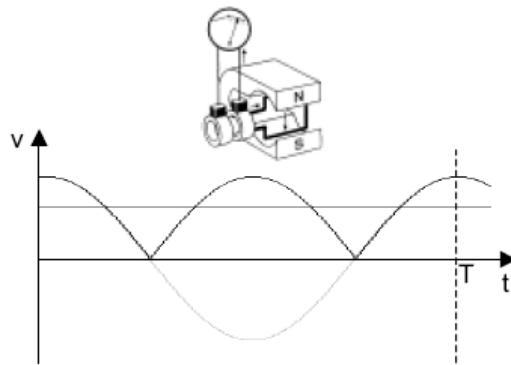


Abbildung 2:

In einer Lichtmaschine, wie sie im Fahrzeugbau eingesetzt wird, besteht der Rotor nicht nur aus einer Wicklung, sondern aus drei unabhängigen um  $120^\circ$  phasenverschobenen Wicklungen. Nach einer 3-Weg Gleichrichtung mittels 6 Dioden, besteht das Ausgangssignal aus einer periodischen Schwingung, welche 6 Perioden pro Umdrehung aufweist. Dadurch wird auch die durchschnittliche Spannung gegenüber einem Generator mit einer Wicklung erhöht. (Abbildung 3).

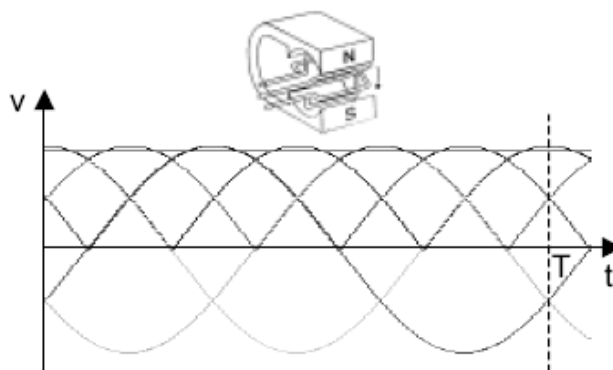


Abbildung 3:

Zusätzlich ist in einer Lichtmaschine auch die Anzahl der Polpaare gegenüber einem einfachen Generator erhöht. Typische Werte sind 12 oder 16 Polpaare, wobei in den meisten Fahrzeugen Lichtmaschinen mit 12 Polpaaren eingesetzt werden. Für jedes Polpaar erhält man nun 6 Perioden pro Umdrehung. Wenn nun also 12 Polpaare vorhanden sind, so steigt auch die Periode entsprechend und man erhält 36 Perioden pro Umdrehung des Rotors. Dies lässt auch die durchschnittliche Spannung wieder ansteigen. Des Weiteren wird in einer Lichtmaschine das Magnetfeld nicht durch einen Permanentmagneten, sondern durch einen Erregerstrom in einem Elektromagneten erzeugt. Zudem sind die Rollen von Stator und Rotor vertauscht, d.h. der Magnet befindet sich auf dem Rotor und die Wicklungen, in welche die Spannung induziert wird, sitzen aussen auf dem Stator. (Abbildung 4).

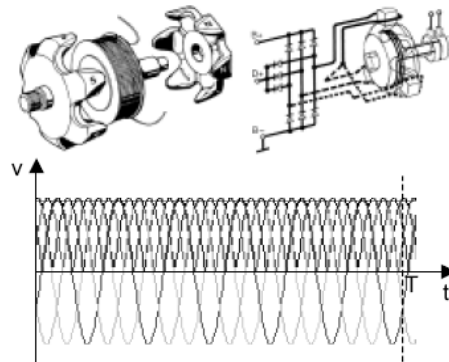


Abbildung 4:

Die Spannung eines Generators hängt stark mit der Drehzahl und der zugeschalteten Last zusammen. Dies könnte zu unzureichender oder zu starker Ladung der Batterie führen oder zu Helligkeitsschwankungen der Scheinwerfer. Darum ist es notwendig, dass diese Schwankungen mit einem so genannten Generatorregler ausgeglichen werden.

Dazu wird bei Drehstrom-Lichtmaschinen das elektrisch erzeugte Magnetfeld im Rotor durch einen integrierten Schaltkreis, den Laderegler, beeinflusst. Durch diesen Regler wird der maximal mögliche Ladestrom und die Maximalspannung limitiert. Der Erregerstrom, welcher das Magnetfeld erzeugt, wird dem Rotor-Elektromagnet zugeführt. Das sich drehende Magnetfeld induziert in den aussenliegenden Statorwicklungen einen 3-Phasen Drehstrom. Die gleichgerichtete Ist-Spannung wird mit einer internen stabilen

Referenzspannung verglichen und entsprechend wird die Stärke des Magnetfeldes durch mehr oder weniger Stromfluss nachgeregelt.

## 4.2. Freilauflichtmaschine

Ein spezieller Typ von Lichtmaschine ist die Freilauflichtmaschine. Die Zeiten, wo ein einfacher Keilriemen den Generator von der Kurbelwelle aus antrieb sind vorbei. Heute versorgen Poly-V-Riemen neben dem Generator und der Wasserpumpe auch noch eine Lenkhilfepumpe, einen Klimakompressor, eventuell noch einen Lüfter und vielleicht sogar die Ölpumpe mit Antriebsleistung. Zudem führen die Vielzahl von elektrischen Komponenten im Fahrzeug zu immer grösseren Massen der Generatoren. Dadurch wird eine grössere Lebensdauer des Keilriemens unabdingbar. Da aber, durch die grössere Masse des Rotors der Lichtmaschine, bei schnellen Beschleunigungswechseln ein Schlupf zwischen Antriebsriemen und Lichtmaschinenpulley entsteht, ist die Abnutzung grösser und die Lebensdauer des Riemens wird kleiner. Deshalb wurde das System des Freilaufes eingesetzt. Freilauf bedeutet, dass wenn der Motor schnell an Drehzahl verliert, dass sich dann die Freilaufscheibe am Lichtmaschinenpulley abkoppelt und der Generator durch seine eigene Schwungenergie alleine weiter dreht. Wenn sich dann die Drehzahl der Lichtmaschine auf die des Motors verringert hat, oder der Motor wieder auf die gleiche Drehzahl beschleunigt hat, dann koppelt sich die Freilaufscheibe wieder ein und der Generator wird wieder direkt vom Motor angetrieben.

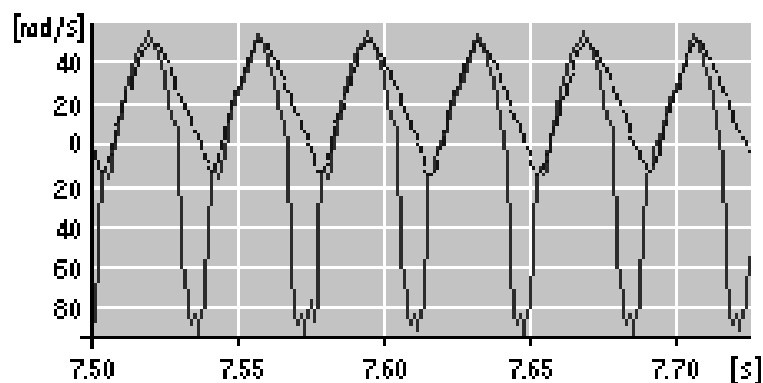


Abbildung 5:

Dieses System ist zu Vergleichen mit dem Antriebssystem an einem Fahrrad. Wenn der Fahrer auf hört zu treten, dann drehen auch die Pedalen nicht mehr weiter und die Kraftübertragung auf das Rad ist Unterbrochen, bis der Fahrer wieder mit der gleichen Umdrehung in die Pedalen tritt, wie das Rad dreht. Dieses Prinzip verhindert jetzt jedoch, dass die Drehzahlmessung über die Bordspannung auf allen Fahrzeugen funktionieren kann. Da während des Freilaufs die Umdrehung des Generators gemessen wird und nicht jene des Motors. (Abbildung 5).

## 5. Messdatenerfassung

Um möglichst viele Informationen über die Bordspannung von Fahrzeugen zu erhalten, wurden zuerst verschiedenste Messdaten erfasst. Dazu mussten wir mehrere Messungen an möglichst vielen verschiedenen Fahrzeugen, unterschiedlicher Fabrikation, Grösse, Alter und vor allem auch Motor durchführen. Besonders wichtig war, dass sowohl Benzin-, wie auch Dieseltreibene Autos verwendet wurden.

### 5.1. Hardware

Um Daten mobil zu erfassen, bedienten wir uns der Mathematik-Software MATLAB in Kombination mit einer DAQ-Karte von National Instruments. Da die Spannung alleine nicht sehr aussagekräftig ist, schossen wir gleichzeitig zu den Messungen Fotos mit einer Webcam. Um möglichst viel über das Verhalten der Spannung heraus zu finden, wurden Messungen bei diversen Drehzahlen aufgezeichnet, von Standgas bis fast maximale Drehzahl.

Für die Erfassung erster Daten wurde ein Matlab-File `data_record.m` erstellt, welches auf Knopfdruck 5000 Samples erfasst, und zusammen mit der Zeit und einem aktuellen Bild des Drehzahlmessers abspeichert. Dies ermöglicht genau in den interessanten Momenten die Spannung mitsamt der entsprechenden Drehzahl als Referenz aufzunehmen.

Um dynamische Erkenntnisse zu erhalten, kam das Matlab-File `echtzeit_aufnahme.m` zum Einsatz. Dieses Programm nahm jeweils mehrere Samplegruppen mit Drehzahlmesserbild hintereinander auf. Später konnten diese Daten filmähnlich wieder angesehen werden. Dies ermöglichte eine sehr gute Eruierung, wann und wie fest welche Störungen auftreten.

Die Signale sind für ein normales Benzin betriebenes Fahrzeug relativ schön und sauber. Ein gutes Beispiel zeigt Abbildung 6. Diese Daten wurden bei einem Ford Focus erfasst. Bei Diesel-Fahrzeugen sieht dies schon etwas anders aus. Bei einem getesteten Opel Personenbus der Hochschule Rapperswil war eine Spannung wie in Abbildung 7 zu messen. Hier sind deutlich mehr Störungen vorhanden. Wie aus den dynamischen

Messungen hervor ging, sind diese abhängig von der Beschleunigung. Will heißen, bei voller Beschleunigung treten extreme Störungen auf. Bei normaler Fahrt sind die Störungen relativ gering. Was die Gründe dafür sind konnte nicht näher erforscht werden.

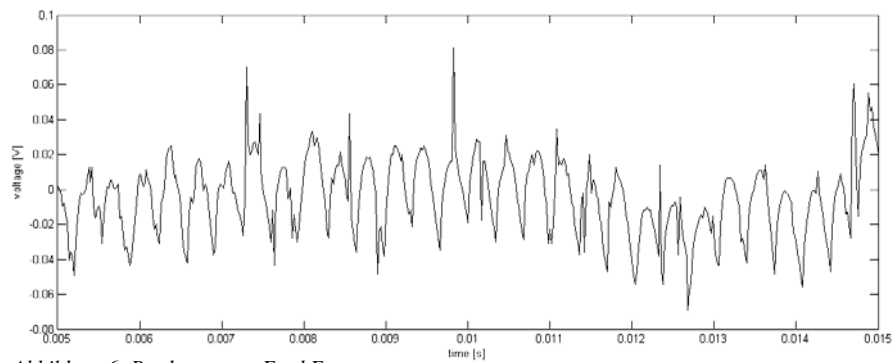


Abbildung 6: Bordspannung Ford Focus

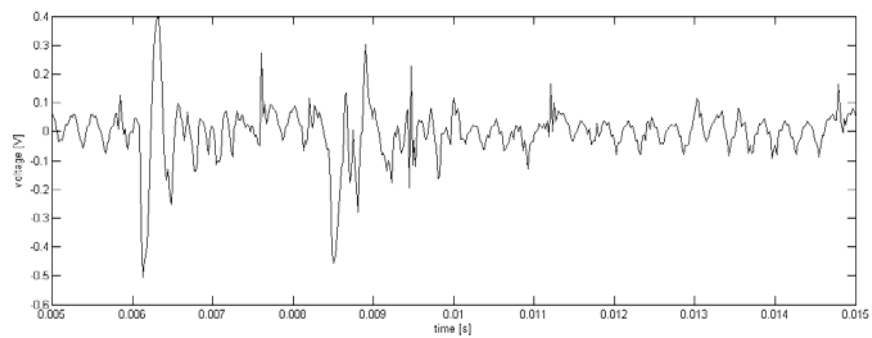


Abbildung 7: Bordspannung Opel Bus

# 6. Hardware Aufbau

## 6.1. Gesamtaufbau

Als Grundlage für den Drehzahlmesser dient das DSP Starter Kit TMS320C6711 DSK Evaluation Board von Texas Instruments. Für die Ausgabe der frequenzabhängigen Spannung und zum Einlesen des Spannungssignals vom Bordnetz wird die Audio Tochterkarte PCM3003 verwendet. Diese erlaubt höhere Abtastraten und benutzt einen Stereo AD- und DA-Wandler. Die Tochterkarte kann einfach auf das Evaluation Board aufgesteckt werden. Die Kommunikation von Tochterkarte und TMS320C6711 funktioniert über den Multichannel Buffered Serial Port 1 (McBsp1) des DSP's. Für das Userinterface wird das Touchpanel Display EA eDIP240-7 von Electronic Assembly eingesetzt. Dieses wird über eine SPI-Schnittstelle mit dem McBsp0 verbunden. Damit dies einfach realisierbar ist, wird zwischen Evaluation Board und Audio Tochterkarte noch ein ProtoPlus Prototyping Development Board eingesteckt, auf welchem man die Leitungen für die SPI-Schnittstelle abgreifen kann. An das Fahrzeug angeschlossen wird das gesamte System über einen handelsüblichen Adapter für Zigarettenbuchsen im Fahrzeug. Von diesem Adapter aus wird zum einen das Spannungssignal auf den AD-Wandler geführt und zum anderen wird die Spannung über einen Spannungsregler auf 5V geregelt und als Speisespannung für den Drehzahlmesser verwendet.

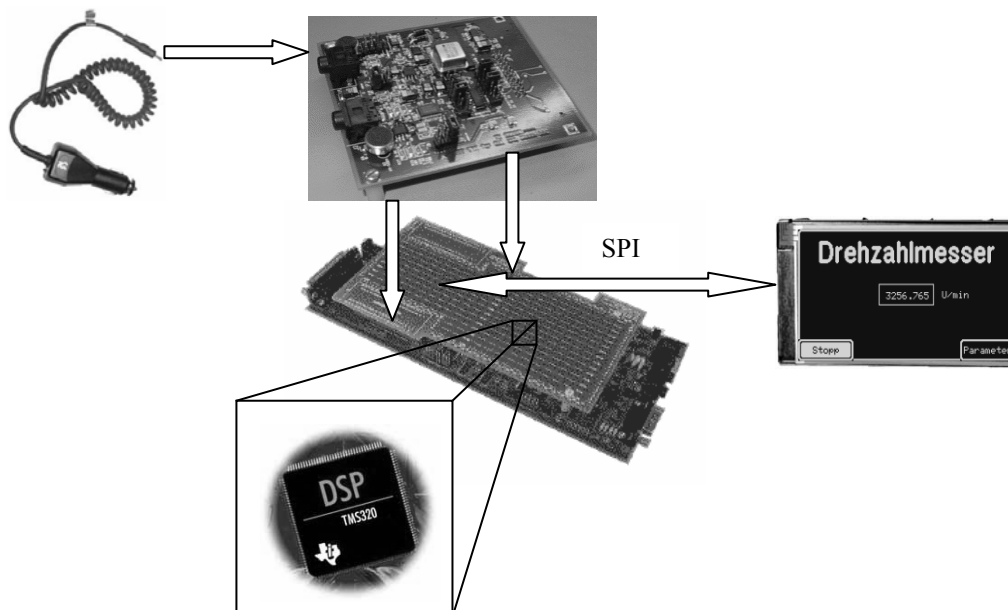


Abbildung 8: Hardware Aufbau

## 6.2. TMS320C6711 DSK

Für die Signalverarbeitung in dieser Applikation wird die Entwicklungsplattform DSP Starter Kit TMS320C6711 DSK mit dem DSP TMS320C6711 als Herzstück eingesetzt. Beim DSP handelt es sich um einen 32 bit Floating-Point DSP von Texas Instruments. Einige Eckdaten des DSK-Boards sind:

- 32 bit Flieskomma DSP
- 150 MHz Systemtakt, 1200 MIPS, 600 MFLOPS
- 16 MB SDRAM auf dem DSK-Board
- Enhanced Direct Memory Access (EDMA), 16 Kanäle
- 2x McBSp (Multichannel Buffered Serial Port) für die serielle Kommunikation mit der Peripherie

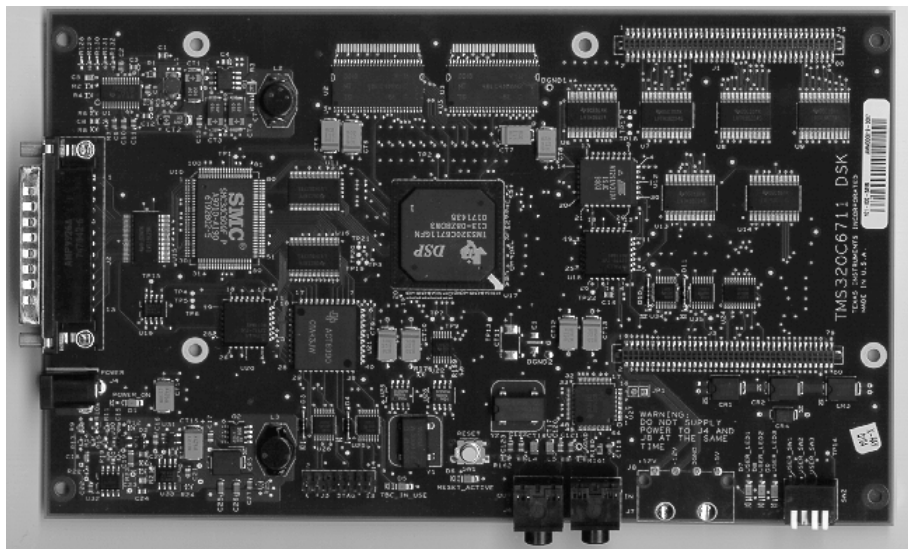


Abbildung 9: DSP-Board

## 6.3. DSP Global ProtoPlus Karte

Die ProtoPlus Karte ist eine 6-layer Tochterkarte, welche auf das Evaluation Board aufgesteckt werden kann. Auf dieses Board können weitere Karten aufgesteckt werden und auch selbst Abgriffe angelötet werden. Dazu stehen neben Anschlüssen zu den Pins des DSP auch noch etliche 3.3V, 5V sowie GND Anschlüsse zur Verfügung.

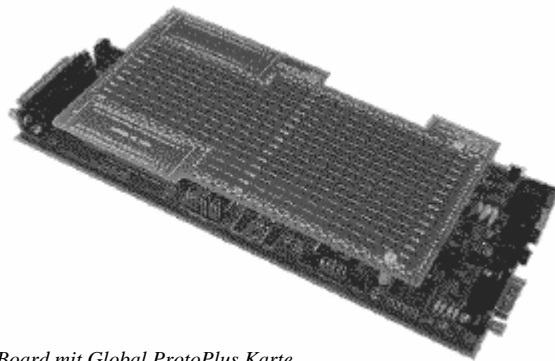


Abbildung 10: DSP-Board mit Global ProtoPlus Karte

## 6.4. PCM3003 Audio Tochterkarte

Die Audio Tochterkarte ist eine Erweiterungskarte für TMS320 DSK's von Texas Instruments. Diese Karte erlaubt Ein- und Ausgabe von Mono- und Stereo-Signalen mit Taktfrequenzen bis 48kHz. Die wichtigsten Eigenschaften der Audiokarte sind:

- Line-in/out stereo mini audio Stecker
- Variable Sample-Rate. Entweder vom „onboard“ Oscillator der Karte, oder über Timer-Output Pin des DSP-Prozessors.

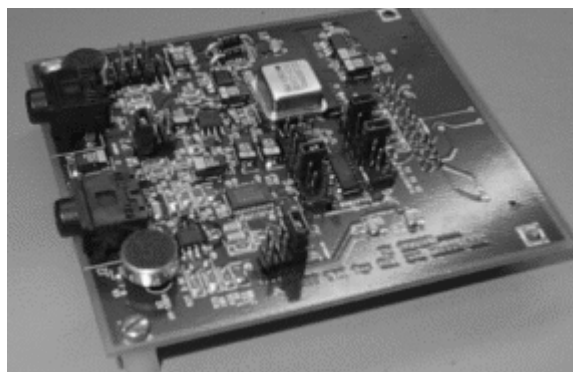


Abbildung 11: Audio Tochterkarte

## 6.5. EA eDIP240-7

Das EA eDIP240-7 ist ein 240x128 Pixel LCD-Display mit Touch-Control und integrierter Intelligenz. Es besitzt ein 32kB EEPROM, auf welchem die Bilder und Makros gespeichert werden können. Einige Eigenschaften des Displays sind:

- 32kB EEPROM
- 3 verschiedene Interface (RS-232, I<sup>2</sup>C-Bus oder SPI-Bus)
- 240x128 Pixel
- Pixelgenaue Positionierung
- bis zu 256 Bilder und Makros speicherbar
- Analoges Touchpanel mit variablem Raster

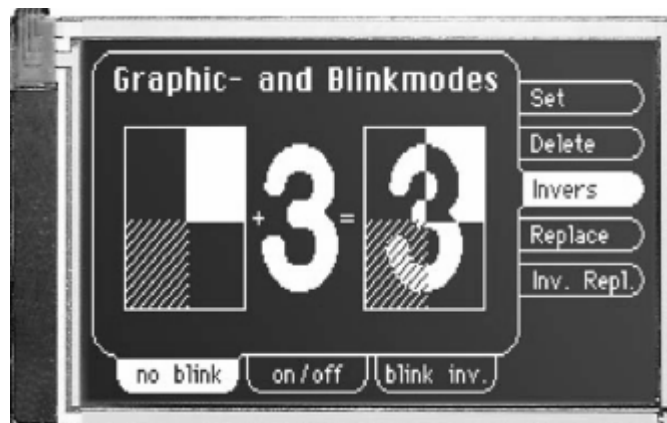


Abbildung 12: Display mit Touchpanel

## 6.6. Spannungsregler

Um den DSP portabel zu machen, wird eine Spannungsquelle benötigt. Da die Drehzahlmessung sowieso am Bordspannungsnetz angeschlossen wird, ist es naheliegend die Speisung auch gleich darüber zu beziehen. Da das DSP-Board einen relativ grossen Strom (bis ca. 5A) beim Starten benötigt muss der Spannungsregler ziemlich

leistungsfähig sein. Einen solchen Strom vom Bordnetz zu verlangen sollte eigentlich kein Problem darstellen, ausser der Zigarettenanzünder ist auf einen tieferen Wert abgesichert.

Zum Einsatz kommt der Spannungsregler LM2940 (Datenblatt siehe Anhang 2) von National Semiconductors. Dieser kann ca. 1A liefern. Darum ist es nötig einen Transistor parallel zu schliessen welcher den Mehrstrom liefern kann. Diese Beschaltung wurde aus den Application Notes des Bausteins LM340 (Anhang 1) abgeschaut, funktioniert jedoch problemlos auch mit dem LM2940.

Der Kondensator C3 und der Widerstand R2 bilden den RC-Hochpass. Die Spannung  $V_s$  wird dann zum DSP geführt, um daraus die Frequenz zu detektieren. Der Spannungsregler LM2940 hat seine Eingangsspannung über den Widerstand R1, daher muss dieser relativ viel Leistung vertragen. Theoretisch sind dies:

$$P_{R1} = \frac{V_{R1}^2}{R_1}; V_{R1} = V_{EB}; P_{R1} = \frac{0.7V^2}{3.4\Omega} = \underline{\underline{144mW}} \quad (6.1)$$

Über dem PNP-Transistor A1232 (Datenblatt siehe Anhang 3) fällt etwas mehr Leistung ab. Theoretisch bei einem maximalen Strom von ca. 5A:

$$P_{TR_{max}} = V_{EC} \cdot I_C = 7V \cdot 5A = \underline{\underline{35W}} \quad (6.2)$$

Diese Leistung wird in der Realität nie vorhanden sein, da der Spannungsregler auch noch einen Strom zum Ausgang beisteuert. Aus Vorsichtsmassnahmen weist der Transistor trotzdem eine Verlustleistung von 100W auf. Aufgrund dieser Leistungen werden die beiden Halbleiterelemente an einen Kühlkörper montiert.

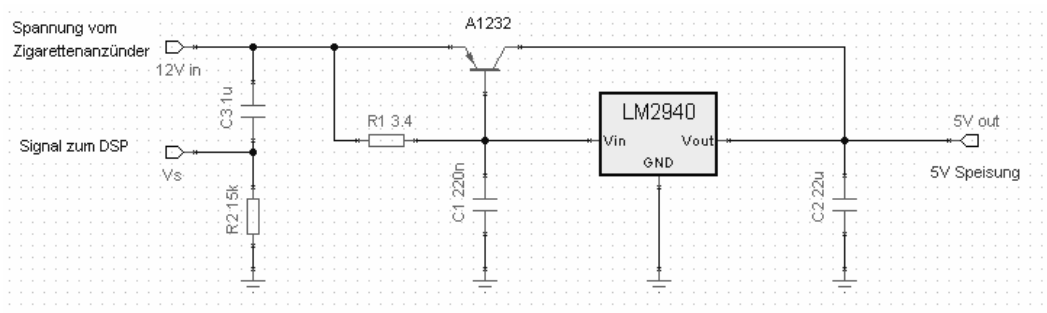


Abbildung 13: Schema des Spannungsreglers

# 7. Theorie Signalaufarbeitung

## 7.1. Allgemeines

Bevor mit dem Viterbi-Algorithmus (siehe entsprechendes Kapitel) der relevante Frequenzpeak detektiert werden kann, muss das Eingangssignal aufgearbeitet werden. Im Wesentlichen besteht dies aus dem Einlesen des Signals, der Korrelation, der Fourier-Transformation und der Filterung. Die einzelnen Teile werden im Folgenden genauer beschrieben. Das Signal liegt wie in Abbildung 14 aufgezeigt am Eingang an.

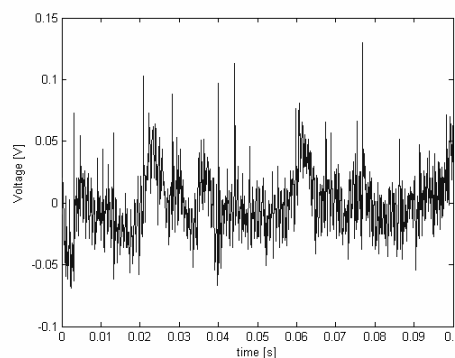


Abbildung 14: Eingangssignal

## 7.2. Einlesen des Signals

Zuerst wird das Signal hardwareseitig Hochpass gefiltert. Dieser Hochpass in Form eines simplen RC-Gliedes weist eine sehr tiefe Grenzfrequenz, ca. 10Hz, auf und dient nur der Gleichspannungs-Unterdrückung.

Das Eingangssignal wird bei der DAQ-Karte mit einem gleichförmigen 12-Bit ADC abgetastet, auf der DSP-Plattform kommt hier ein 16-Bit ADC zum Einsatz. Die Abtastfrequenz unter Matlab beträgt 50kHz und auf dem DSP 24kHz. Diese reicht völlig aus, da unter normalen Bedingungen eine Frequenz von ca. 8.5kHz auftreten kann.

Nach dem Abtasttheorem, welches aussagt, dass die Abtastfrequenz mindestens

$$f = \frac{60 \cdot n}{3 \cdot p \cdot v}$$

n: Drehzahl in U/min  
p: Polpaarzahl, typisch 12 oder 16  
v: Übersetzungsverhältnis, typisch zwischen 2 und 3

$$\frac{60 \cdot 10'000}{3 \cdot 12 \cdot 2} = \underline{\underline{8333 Hz}} \quad (7.1)$$

doppelt so hoch sein muss wie die höchste zumessende Frequenz, müsste die Abtastfrequenz also mindestens 17kHz sein.

Es werden jeweils 256 Samples in einem Frame zusammengenommen. Die Wahl traf diese Länge, damit nicht zulange auf ein neues Frame gewartet werden muss, und damit die Rechenzeiten bei der Signalverarbeitung nicht zu hoch werden. 256 Samples bei einer Abtastfrequenz von 50kHz entsprechen einer Zeitdauer von 5.12ms, bzw. 10.66ms bei einer solchen von 24kHz, über die das Signal abgetastet wird. Abbildung 15 zeigt eine Samplegruppe von 256 Samples. Darin sind ca. drei Schwingungen enthalten, was zurück auf eine dreifach zu hohe Abtastfrequenz schliessen lässt.

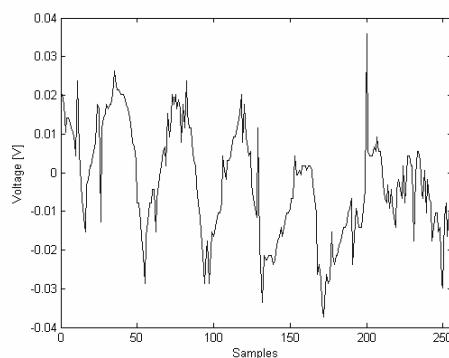


Abbildung 15: 256 Samples

Das Signal wird dann mit einem Hanning-Fenster ausgeschnitten. Dies wird so gemacht, damit das Frequenzspektrum deutlicher den relevanten Peak zeigt, als wenn das Signal mit einem Rechteck-Fenster ausgeschnitten würde. Das Hanning-Fenster bedeutet, dass die Nebenfrequenzen, welche durch das Ausschneiden erzeugt werden, relativ stark unterdrückt werden, im Gegensatz zum Rechteck-Fenster. Dafür ist der Peak etwas breiter, was zur Folge hat, dass Frequenzspitzen in nächster Nähe in der Main Lobe „versinken“. In dieser Anwendung ist dies jedoch nicht von Bedeutung, da ja nur der

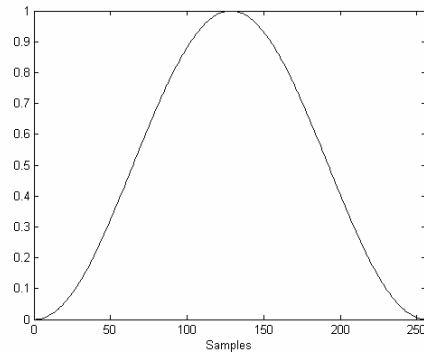


Abbildung 16: Hanning-Fenster

Hauptpeak von Relevanz ist. Abbildung 17 zeigt das Frame aus Abbildung 15 mit dem Hanning-Fenster (Abbildung 16) multipliziert. Die Formel für die Berechnung des Hanning-Fensters lautet:

$$w(s) = \sin^2\left(\frac{\pi \cdot s}{256}\right) \quad (7.2)$$

Für den vorliegenden Fall, wo die Länge des Frames 256 Samples misst.

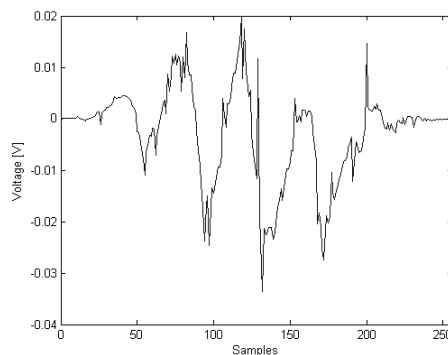


Abbildung 17: Signal in Hanning-Fenster

### 7.3. Korrelation

Als nächster Schritt folgt die Korrelation. Diese soll zufällige unkorrelierte Störsignale herausfiltern. In der Matlab-Version kann hier sogar auf die Kreuzkorrelation zurückgegriffen werden. Hier werden zwei Frames, ein früheres und ein späteres miteinander korreliert. Da bei der Berechnung auf der DSP-Plattform nicht genügend Zeit

zur Verfügung steht um mehrere Samplegruppen abzuwarten, wird hier die Autokorrelation, also die Korrelation zwischen ein und demselben Signal, eingesetzt. Bei der Kreuzkorrelation können zusätzlich periodische Signale die nur in einer der beiden Frames vorhanden sind gefiltert werden. Das Resultat dieses Verarbeitungsschritts ist in Abbildung 18 zu sehen. Gut erkennbar ist nun nur noch eine harmonische Schwingung vorhanden, welches die gesuchte Frequenz aufweist. Die Umhüllende schafft eine ungewollte tiefe Frequenz die später wieder herausgefiltert werden muss. Die Formel für die Kreuzkorrelation lautet:

$$R_{XY}(s) = \sum_{t=-\infty}^{\infty} x(t) \cdot y(t+s) \quad (7.3)$$

Für die Autokorrelation wird einfach zweimal das gleiche Signal eingesetzt. Im Wesentlichen stellt die Korrelation eine Faltung der beiden Eingangssignale dar.

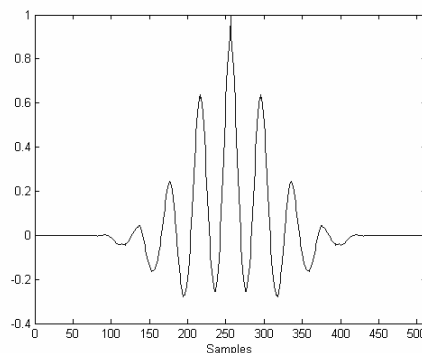


Abbildung 18: Autokorrelation

## 7.4. Transformation

Dieses nun generierte Signal wird jetzt Fouriertransformiert. Das Signal wird damit vom Zeitbereich in den Frequenzbereich transformiert. Da die ganze Signalverarbeitung digital abläuft, kommt hier natürlich die schnelle Fouriertransformation zum Einsatz. Auf dem DSP wird das Radix 2-Prinzip angewendet. Bei diesem Prinzip wird die gesamte FFT in kleinere Teil-FFT's aufgeteilt. Und man macht sich die Eigenschaft zu nutze, dass der Drehoperator (Twiddle-Faktoren) periodisch und symmetrisch ist. Dadurch muss

nicht jeder Koeffizient neu berechnet werden, was einen entscheidenden Geschwindigkeitsvorteil bringt. Die verwendete Funktion arbeitet nach dem „decimation in time“-Verfahren. Das bedeutet, dass die Ausgangswerte nicht in der richtigen Reihenfolge sind, sondern bit-reversed. Ergo muss dies rückgängig gemacht werden. Dies geschieht in dem von jedem Wert die binäre Position einfach rückwärts gelesen wird. Um die Twiddle-Konstanten zu berechnen wird folgende Formel angewendet:

$$W_N^K = e^{-jK\frac{2\pi}{N}} \quad (7.4)$$

Die Formel für eine FFT lautet:

$$Y(k) = \sum_{n=0}^{N-1} X(n)W_N^{kn} \quad (7.5)$$

In Abbildung 19 ist das Frequenzspektrum welches aus dem Signal aus Abbildung 18 hervor geht. Sehr gut zu sehen sind die zwei grossen relevanten Frequenzpeaks. In der Mitte ist die niederfrequente Schwingung vorhanden, welche in Abbildung 18 die Umhüllende darstellt. Die ersten Harmonischen sind ebenfalls noch schwach vorhanden.

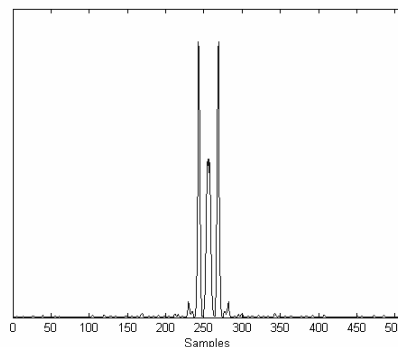


Abbildung 19: Fouriertransformiert

## 7.5. Filterung

Das nun erhaltene Frequenzspektrum enthält je nachdem ziemlich viele tieffrequente Störungen. Häufig sind diese unerwünschten Signale sogar grösser als der

relevante Frequenzpeak. Um diese Signale zu eliminieren, wird ein FIR-Filter benutzt. Das Filter weist eine Grenzfrequenz von 700Hz auf. Somit kommen auch die Frequenzen bei der tiefsten Drehzahl noch ungehindert durch. Alle Signale mit tieferer Frequenz werden gefiltert. Bei den Testfahrten stellte sich heraus, dass diese Grenzfrequenz dem Fahrzeug noch grob angepasst werden soll. Dieselbetriebene Autos haben aufgrund ihrer höheren Übersetzung beim Standgas eine etwas höhere Frequenz als Benziner. Deswegen könnte es sinnvoll sein die Grenzfrequenz des Filters der

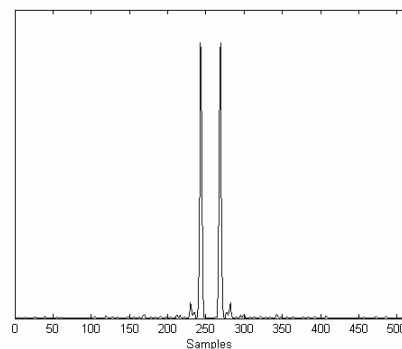


Abbildung 20: Hochpass gefiltert

Übersetzung anzupassen. Bei der Implementierung auf dem DSP wird nun zwischen einer Grenzfrequenz von 700Hz und 900Hz entschieden. In Abbildung 20 ist nun das Frequenzspektrum nach der Filterung zu sehen. Im Vergleich zur Abbildung 19 fehlt hier der tieffrequente Frequenzpeak. Das Signal kann so nun mit dem Viterbi-Algorithmus bearbeitet werden. Für dieses Filter wurde jeweils ein FIR-Filter verwendet. Für die Anwendung auf dem DSP, wurden die Filterkoeffizienten mit Matlab berechnet, und im entsprechenden c-File eingefügt.

# 8. Viterbi-Algorithmus

## 8.1. Allgemeines

Der Viterbi-Algorithmus wurde 1967 von Dr. Andrew J. Viterbi zur Decodierung von Faltungscodes entworfen. G.D. Forney leitete daraus 1972 den Optimalenempfänger für verzerrte und gestörte Kanäle her.

Der Viterbi-Algorithmus findet seine Anwendung heutzutage zum Beispiel in Handys oder Wireless-LANs zur Entzerrung oder Fehlerkorrektur der Funkübertragung. In der Nachrichtentechnik und Informatik ist dieser Algorithmus weit verbreitet. Die Informationstheorie, Spracherkennung, Bioinformatik und Computerlinguistik verwenden gerne den Viterbi-Algorithmus. Die Spracherkennung ohne diese Methode wäre schwierig zu realisieren.

Der Viterbi-Algorithmus ist eng verwandt mit dem Faltungscode zur Vorwärtsfehlerkorrektur von Datenströmen.

## 8.2. Hidden Markow Modell

Da das Hidden Markow Modell (HMM) die Basis für den Viterbi-Algorithmus legt, wird hier in kurzen Zügen das Prinzip dieses Modells beschrieben.

Hidden Markow Modelle sind stochastische Modelle, welche sich durch zwei Zufallsprozesse beschreiben lassen. Der erste Zufallsprozess entspricht dabei einer Markow-Kette, welche durch Zustände und Übergangswahrscheinlichkeiten gekennzeichnet ist. Die Zustände der Kette sind jedoch von aussen nicht direkt sichtbar, daher der Name Hidden. Ein zweiter Zufallsprozess erzeugt stattdessen zu jedem Zeitpunkt beobachtbare Ausgangssymbole gemäss einer zustandsabhängigen Wahrscheinlichkeitsverteilung. In dieser Problemstellung soll nun für ein gegebenes verborgenes Markow-Modell die wahrscheinlichste Sequenz der versteckten Zustände bestimmt werden, welche eine vorgegebene Ausgabesequenz erzeugt hat. Zur Lösung dieses Problems wird der Viterbi-Algorithmus zur Hilfe gezogen.

### 8.3. Veranschaulichung

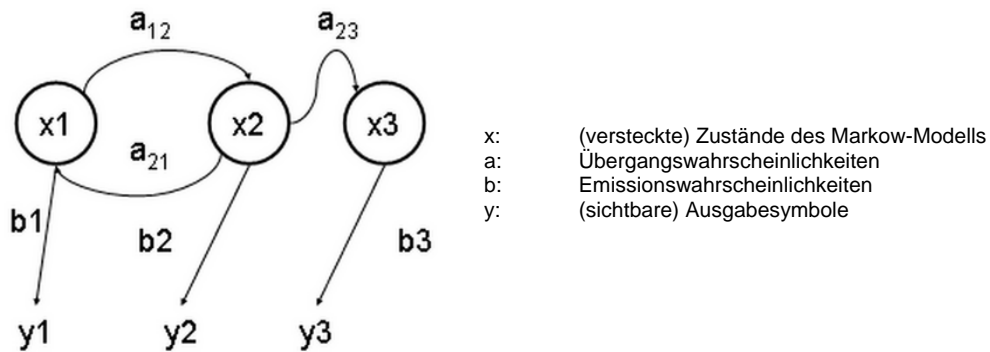


Abbildung 21. Markow-Modell

### 8.4. Funktionsweise

Der Viterbi-Algorithmus ist eine Methode aus dem Gebiet der Dynamischen Programmierung, welche die wahrscheinlichste Abfolge von versteckten Zuständen findet, die zu einer Abfolge von beobachtbaren Ereignissen führt. Die Abfolge von versteckten Zuständen wird Viterbi Pfad genannt. Dieser entspricht der Markow-Kette im gleichnamigen Modell.

Diesen Algorithmus kann man also zur Erkennung von Mustern verwenden. Dies ist ein weites Feld, da Lebewesen ständig Sinnesreize interpretieren müssen und aus dem bereits gelernten diese Signale einordnen. Der Viterbi-Algorithmus tut genau dies auch und ist somit ein wichtiger Baustein der künstlichen Intelligenz. Dies heisst jedoch nicht, dass der Algorithmus immer richtig liegt.

Das bei Viterbi zu lösende Problem gleicht der Suche des kürzesten Weges zwischen zwei Orten in einem sehr regelmäßigen Straßennetz; man bezeichnet dieses als Trellis (Spaliergitter).

Der Viterbi-Algorithmus findet zu jeder Beobachtung  $Y$  die wahrscheinlichste Zustandsfolge  $X$ , die ein HMM bei der Ausgabe von  $Y$  genommen hat. Dazu definiert man ein Trellis.

$$\varphi_j(t) = \max_{X_1 \dots X_{t-1}} P(X_1 \dots X_{t-1}, o_1 \dots o_{t-1}, X_t = j | \mu) \quad (8.1)$$

$\varphi_j(t)$  enthält an jeder Stelle die Wahrscheinlichkeit des wahrscheinlichsten Weges, der unter Emission von  $o_1 \dots o_{t-1}$  zum Zustand  $j$  zur Zeit  $t$  führt. Zusätzlich braucht man noch die Information, welcher Weg das war. Diese Information wird durch das unten definierte zweite Trellis  $\psi$  geliefert. Die Initialisierung ist:  $\varphi_i(t) = \delta_{ij}$  mit dem Startzustand  $j$ . Dabei arbeitet man sich nach vorne durch,

$$\varphi_j(t+1) = \max_{i=1, \dots, N} \varphi_i(t) \delta(i, j) \lambda(i, j, o_t) \quad (8.2)$$

und merkt sich, woher man gekommen ist.

$$\psi_j(t+1) = \arg \max_{i=1, \dots, N} \varphi_i(t) \delta(i, j) \lambda(i, j, o_t) \quad (8.3)$$

Der Operator  $\arg\max$  bestimmt das Argument des Maximalwerts, wobei das Argument jenes ist, worüber die Maximierungsoperation läuft. Im Fall oben ist dies der Index  $i$ , so dass man in  $\psi$  gerade speichert, woher das  $\varphi$  gekommen ist. Man braucht dies später, um sich wieder zurückzuangeln.

Wenn beide Trellises gefüllt sind, sucht man von hinten startend die wahrscheinlichste Sequenz  $X$ .

$$\hat{X}_{T+1} = \arg \max_{i=1, \dots, N} \varphi_i(T+1) \quad (8.4)$$

$$\hat{X}_t = \psi_{\hat{X}_{t+1}}(t+1) \quad (8.5)$$

$$P(\hat{X}) = \varphi(\hat{X}_{T+1}) \quad (8.6)$$

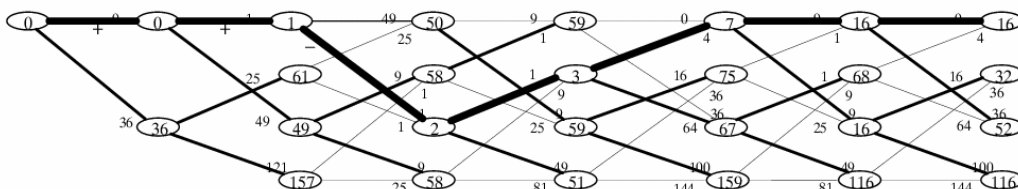


Abbildung 22: Trellis

## 8.5. Implementation des Viterbi-Algorithmus

Da für die Drehzahlbestimmung der Bordspannung die überlagerte Frequenz der Lichtmaschine nötig ist, wird im folgenden Abschnitt hauptsächlich mit den Frequenzen des Generators gearbeitet.

Für die Anwendung des Viterbi-Algorithmus auf die Drehzahlmessung wird folgendermassen vorgegangen. Von jeder Messung werden aus dem Leistungsdichtespektrum die 50 dominantesten Frequenzen herausgesucht. Dabei werden der Amplitudenwert und die Frequenz gemeinsam als Paar abgespeichert. Dies geschieht im c-Code mittels den Funktionen `dspf_sp_maxval()` und `dspf_sp_maxidx()` aus der DSP Library.

Danach wird für jeden Punkt errechnet, wie gross der kleinste Aufwand (Kosten) ist, um von der letzten Messung zu diesem Punkt zu gelangen. Dazu werden die Übergangskosten von jedem Frequenzpunkt aus der letzten Messung und die Kosten welche der Punkt durch seine eigene Amplitude verursacht zusammen gerechnet. Die Übergangskosten bestehen dabei aus dem Aufwand des vorangegangenen Punktes und dem Wegaufwand von der alten Frequenz zur neuen. Dies lässt sich folgendermassen formulieren. Von jedem Punkt aus der letzten Messung wird der Aufwand genommen, welcher für diesen entstanden ist. Als Wegaufwand wird dann für diesen Punkt die Differenz zwischen seiner Frequenz und der Frequenz, für welche die Kosten gesucht werden, ermittelt und mit dem Faktor  $\alpha$  gewichtet.  $\alpha$  bestimmt dabei, wie Flexibel das System auf Frequenzsprünge reagieren soll. Diese beiden Teilkosten werden dann als Übergangskosten zusammen gezählt. Addiert mit den Kosten, welche durch die Amplitude entstanden sind, ergibt dies dann 50 verschiedenen Kosten für einen einzigen Frequenzwert aus der aktuellen Messung. Aus diesem Feld sucht man den kleinsten Aufwand heraus und ordnet diesen zusammen mit dem Frequenzpunkt, welche dabei die kleinsten Übergangskosten verursacht hat, dem aktuellen Punkt zu. So erhält jeder Frequenzpunkt vier Werte, welche seine Eigenschaft und Wahrscheinlichkeit als Drehzahlrepräsentant beschreiben. Formell sieht dies so aus.

$$w_f(n) = \alpha * \text{abs}(f(n) - f_q(n-1)) + c_q(n-1) \quad \begin{array}{l} q \\ f: \\ c: \\ a: \\ w: \end{array} \quad \begin{array}{l} = 1 \dots 50 \\ \text{aktueller Frequenzpunkt} \\ \text{Aufwand (Kosten)} \\ \text{Amplitude} \\ \text{Übergangskosten} \end{array} \quad (8.7)$$

$$c_f(n) = \min\{-a_f(n) + w_f(n)\} \quad (8.8)$$

Für jeden der 50 Werte einer Messung werden diese Schritte durchgeführt und danach wird eine neue Messung eingelesen, bis ein Trellis entstanden ist, welches 10 Messungen umfasst. Als Ausgabefrequenz für die Drehzahlberechnung wird danach der Weg zurück durch das Trellis, bis zur fünften Messung gesucht. Die Ausgabe aus der Mitte des Trellis wurde gewählt, da so die Frequenz, welche für die Berechnung verwendet wird, sowohl von seinen Vorgängern, wie auch von dem aktuellen Wert abhängt. Beim Weg suchen ermittelt man aus der letzten Messung, welche in diesem Fall in der zehnten Spalte des Trellis steht, jenen Punkt, welcher den kleinsten Aufwand aufweist. Bei diesem Frequenzpunkt ist dann seine Herkunft abgespeichert. Beim abgespeicherten Punkt kann man dann wieder nachschauen, welches sein Vorgänger war, bis man bei der fünften Spalte angekommen ist. Danach hat man die Frequenz, welche in die Drehzahl umgerechnet werden kann.

Die Wegsuche erfolgt mit folgender Code Sequenz:

```

if(check==10)
{
  for(i=anzMeas-1;i>=5;i--)
  {
    if(i==anzMeas-1)
    {
      minCost=10000;
      for(j=0;j<anzMax;j++)
      {
        if(minCost>trellis[i][j].cost)
        {
          minCost=trellis[i][j].cost;
          way=trellis[i][j].fromTrellisPos;
        }
      }
    }
    else
    {
      frequenz=freqVec[trellis[i][way].freqPos];
      way=trellis[i][way].fromTrellisPos;
    }
  }
}
else
{
  frequenz = 0;
  check++;
}

```

Nach dem Auslesen der Frequenz muss man das Trellis für die nächste Messung vorbereiten. Dafür wird von dem Punkt, welchen man zuvor erhalten hat noch weiter zurückgegangen, bis man in der ersten Spalte angelangt ist. Dort wird dessen Aufwand ausgelesen und danach wird dieser Wert von allen Aufwänden in der letzten Spalte abgezählt. So wird verhindert, dass der Aufwand gegen einen immer kleineren Wert strebt und die Frequenzänderung ihren Einfluss verliert. Danach wird das ganze Trellis um eine Position geschoben, so dass in der letzten Spalte wieder Platz entsteht, für eine neue Messung.

Für die neue Messung, welche nachher eingelesen werden kann, wird dann wieder für jeden Punkt sein kleinster Aufwand berechnet.

Mit diesem Algorithmus erreicht man, dass der DSP zwischen einer kurzzeitigen Störfrequenz und der Drehzahlinformation unterscheiden kann.

## 9. Implementation unter MATLAB

Unter Matlab wurde eine Version ohne graphische Benutzeroberfläche „viterbi\_realtime\_without\_save.m“, und eine Version mit Benutzeroberfläche „viterbi\_realtime2\_gui.m“ erstellt. Für die Version ohne Benutzeroberfläche wurde noch ein weiteres Programm entwickelt „viterbi\_realtime\_with\_save.m“, welches die Daten aufzeichnet und eine nachträgliche Analyse im Labor erlaubt. Der Ablauf der Signalverarbeitung, sowie der Frequenzpeak-Suchvorgang ist im entsprechenden Unterkapitel ausführlich erklärt. Die Idee der graphischen Anwendung war ein einfach zu bedienendes Programm zu schaffen, mit dem anschaulich gezeigt werden kann wie die Frequenzdetektion funktioniert. Ein Screenshot dieser Anwendung ist in Abbildung 23 zu sehen.

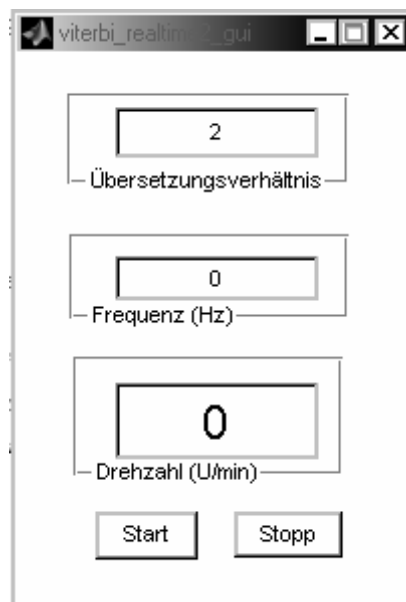


Abbildung 23: graphische Benutzeroberfläche

## 10. Implementation auf DSP-Plattform

Für komplexere Berechnungen auf dem DSP konnten wir auf vorprogrammierte Funktionen von Texas Instruments zurückgreifen. Aus der TMS320C67x DSP Library werden die Funktionen für die Autokorrelation, sowie für die schnelle Fouriertransformation eingesetzt. Die Funktionen für die Erzeugung der Twiddle-Konstanten, welche für die FFT benötigt werden, sowie die Funktion um diese in die bitreversed-order zu bringen sind aus dem Codefile (Anhang 5), ebenfalls von Texas Instruments. Die Funktionen, sowie wie sie in dieser Anwendung zum Einsatz kommen, werden im Folgenden kurz beschrieben. Für genauere Informationen kann in der Dokumentation [1] dieser Bibliothek nachgesehen werden.

```
void DSPF_sp_autocor (float *restrict r, const float *restrict x, int nx, int nr)
```

Diese Funktion führt die Autokorrelation des Eingangssignals mit der Länge  $nx$  durch. Der Eingangsvektor  $x$  muss  $nr + nx$  lang sein. Wobei  $nr$  die Anzahl Nullen ist, welche im ersten Teil des Eingangsvektors vorzufinden sind. Für diese Anwendung hat der Eingangsvektor eine Länge von 256 Samples, dazu werden noch 256 Nullen vorne angehängt. Dieses Anfügen von Nullen ist nötig, damit beim Falten keine Werte in einen ungültigen Bereich geschoben werden. Das Resultat dieser Berechnung wird in den Vektor  $r$  geschrieben. Das Ergebnis einer Autokorrelation ist in jedem Fall eine gerade Funktion mit der doppelten Länge des Eingangssignals. Dies bedeutet, dass nur die eine Hälfte im Ausgangsvektor zurückgegeben wird. Dieser ist demzufolge 256 Samples lang.

```
void DSPF_sp_cfft2_dit (float * x, float * w, short n)
```

Diese Funktion berechnet die schnelle Fouriertransformation. Die Eingangsdaten liegen im Vektor  $x$ . Dieser Vektor enthält jeweils die real Anteile gefolgt von den imaginär Anteilen. Demzufolge ist der Vektor zweimal so lang wie die Anzahl der komplexen Koeffizienten. Im vorliegenden Fall ist der Vektor 1024 Samples lang, enthält ergo 512 komplexe Eingangswerte. Die Twiddle-Faktoren sind im Vektor  $w$  gespeichert. Dabei gilt es zu Beachten, dass diese in bitreversed order vorliegen und ebenfalls komplex sind. Die Länge, über welche die FFT gerechnet wird, ist in der Variable  $n$  abgelegt, hier also 512. Das Resultat wird wiederum im Vektor  $x$  abgelegt, ebenfalls komplex. Dieser Ausgangsvektor muss nun jedoch noch in die normale Ordnung gebracht werden, da er bitreversed zurückgegeben wird.

```
void gen_twiddle(float *w, int n)
```

Diese Funktion ist zuständig für die Generierung der Twiddle-Koeffizienten. Diese werden im Vektor w zurückgegeben. Die Variable n gibt die Anzahl der Koeffizienten an.

```
void bit_rev(float* x, int n)
```

Mit dieser Funktion können Vektoren die in der bitreversed Ordnung vorliegen in normale Ordnung gebracht werden. Das Vorgehen dabei ist, dass die Position als binäre Zahl geschrieben wird. Die neue Position ist die binäre Zahl rückwärts gelesen. Daraus folgt, dass die Funktion in beide Richtungen benutzt werden kann.

# 11. Display

Für die Benutzerschnittstelle wird das LCD Touchpanel EA eDIP240-7 verwendet. Die Menüstrukturen des Displays werden als Makros in ein internes 32kB EEPROM abgespeichert und auch aus diesem ausgeführt. Dem DSP-Board werden nur noch die aktuell aktiven Makros, sowie die für die Drehzahlberechnung relevanten Parameter übergeben. Der Datenaustausch erfolgt dabei über eine SPI-Schnittstelle.

## 11.1. Display Programmierung

Die Programmierung des Displays erfolgt mit einer Software des Herstellers, welche von deren Homepage heruntergeladen werden kann. Sofern ein Programmer EA 9777-1USB angeschlossen ist, kann das erstellte Programm über die USB-Schnittstelle in das EEPROM gebrannt werden. Die Übertragung des Programms ins EEPROM kann auch über einen MAX232 erfolgen. Dabei ist das Display wie unten dargestellt anzuschliessen.

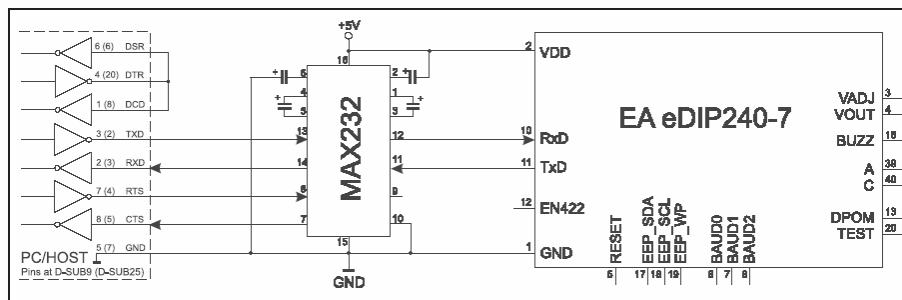


Abbildung 24: Display-Anschlusschema

Auf dem Display können verschiedene Makros programmiert werden. Makros beinhalten einzelne oder mehrere Befehlsfolgen. Dabei kann zwischen normalen Makros und Touchmakros unterschieden werden. Die normalen Makros beinhalten hauptsächlich konstante Text- und Grafikausgaben und können als Hintergrund für mehrere Touchmakros verwendet werden. Die Touchmakros definieren die Touchbereiche, mit welchen Eingaben, sowie Ausgaben gemacht werden können. Die Makros können sowohl von andern Makros, oder vom DSP mittels den Befehlen Makro ausführen gestartet

werden. Wurde ein Makro ausgeführt, kann vom Display ein Rückgabewert an den DSP gesendet werden. Dieser Rückgabewert kann ausgewertet werden und der DSP weiss, welche Befehle vom User getätigt wurden. Eine komplette Übersicht der Programmierbefehle zur Erstellung von Makros findet man im Datenblatt (Anhang 4). Der allgemeine Aufbau eines Befehls wird hier anhand des Befehls zur Ausgabe eines Textes visualisiert.

```
ESC ZL x1, y1, 'Hello World'
```

Dies erzeugt eine Zeichenkette mit dem Text „Hello World“, welche Linksbündig an der Startposition x1, y1 ausgegeben wird.

## 11.2. SPI Schnittstelle

Das Display wird über die SPI-Schnittstelle an den DSP angeschlossen. Wird das Display wie nachfolgend gezeigt beschaltet, so ist der SPI-Mode aktiviert. Die Datenübertragung erfolgt dann über die serielle synchrone SPI-Schnittstelle. Eine Datenübertragung ist bis zu 100kHz möglich. Wird jedoch zwischen den einzelnen Bytes während der Übertragung Pausen von jeweils min. 100µs eingehalten, kann ein Byte mit bis zu 3MHz übertragen werden.

Pinout eDIP240-7						
SPI mode						
Pin	Symbol	In/Out	Function	Pin	Symbol	Function
1	GND	-	Ground Potential for logic (0V)	21	N.C.	not connected
2	VDD	-	Power supply for logic (+5V)	22	N.C.	not connected
3	VADJ	In	Operating voltage for LC driving (input)	23	N.C.	not connected
4	VOUT	Out	Output voltage for LC driving	24	N.C.	not connected
5	RESET	-	L: Reset	25	N.C.	not connected
6	SS	In	Slave Select	26	N.C.	not connected
7	MOSI	In	Serial In	27	N.C.	not connected
8	MISO	Out	Serial Out	28	N.C.	not connected
9	CLK	In	Shift Clock	29	N.C.	not connected
10	DORD	In	Data Order (0=MSB first; 1=LSB first)	30	N.C.	not connected
11	SPIMODE	In	connect to GND for SPI interface	31	N.C.	not connected
12	N.C.		do not connect, reserved	32	N.C.	not connected
13	DPOM	In	L: disable Power-On-Macro do not connect for normal operation	33	N.C.	not connected
14	CPOL	In	Clock Polarity (0=LO 1=HI when idle)	34	N.C.	not connected
15	CPHA	In	Clock Phase (sampled on 0=1st 1=2nd edge)	35	N.C.	not connected
16	BUZZ	Out	Buzzer output	36	N.C.	not connected
17	EEP_SDA	Bidir.	Serial Data Line for int. EEPROM	37	N.C.	not connected
18	EEP_SCL	Out	Serial Clock Line for int. EEPROM	38	N.C.	not connected
19	EEP_WP	In	H: Write Protect for int. EEPROM	39	A	LED backlight+ / internal connection
20	TEST SBUF	IN Out	open-drain with internal pullup 20..50k IN (Power-On) L: Testmode OUT L: data in sendbuffer	40	C	LED backlight- / internal connection

Abbildung 25: Display-Beschaltung

Mit den Eingängen DORD, CPOL und CPHA werden die Hardwarebedingungen an den Master angepasst.

DORD (**D**ata**O**RDer): = 0: MSB (Bit7) wird zuerst gesendet.

= 1: LSB (Bit0) wird zuerst gesendet.

CPOL (**C**lock**P**OLariy): = 0: Ruhezustand von SCK LOW.

= 1: Ruhezustand von SCK HIGH.

CPHA (**C**lock**P**Hase): = 0: Übernahme bei 1. Flanke

= 1: Übernahme bei 2. Flanke

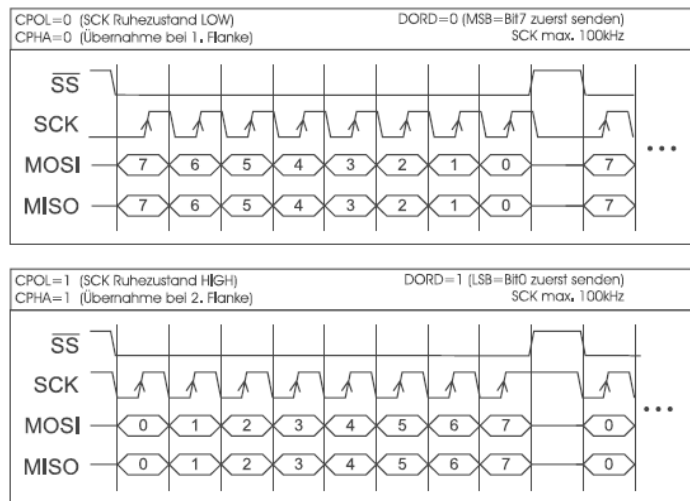


Abbildung 26: SPI Datenübertragung

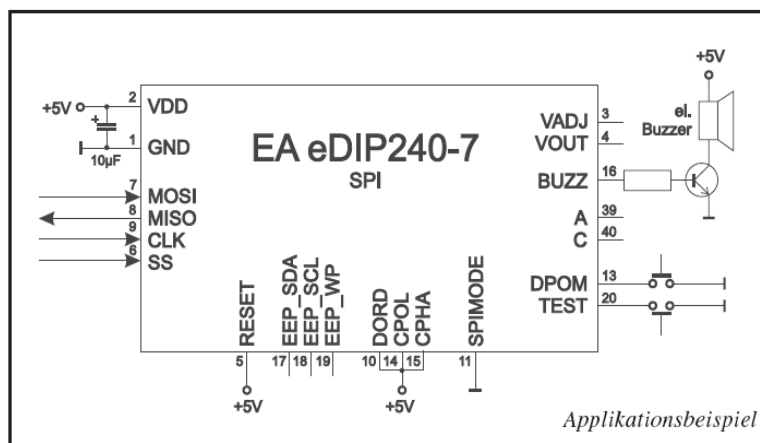


Abbildung 27: Displaybeschaltung für SPI

Am Pin 20 (SBUF) zeigt das Display mit einem low-Pegel an, dass in internen Sendebuffer Daten zur Abholung bereit stehen. Diese Leitung kann an einen der vier externen Interrupts des DSP's angeschlossen werden.

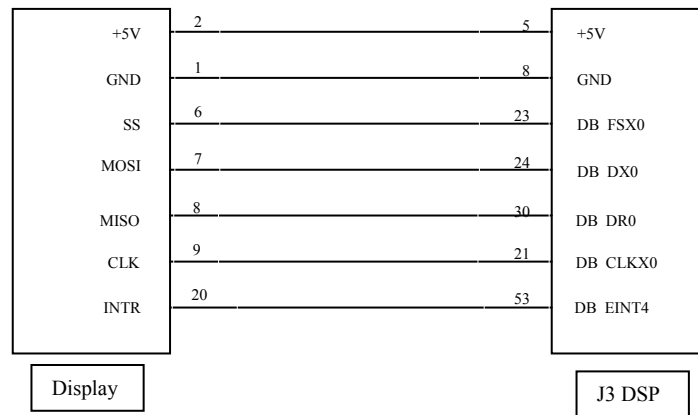


Abbildung 28: Verbindungen SPI-Schnittstelle

Damit beim Display die SPI-Schnittstelle genutzt werden kann, müssen die Verbindungen gemäss Abbildung 28 gemacht werden. Es ist zu beachten, dass noch der Jumper J1 auf dem Board gesetzt werden muss!

## 11.3. Initialisierung McBsp als SPI Schnittstelle

Als SPI Schnittstelle wird der McBsp0 Bus benutzt. Dies geschieht, indem man verschiedene Register richtig initialisiert. Am einfachsten verwendet man die CSL, um den McBsp-Bus zu initialisieren. Dazu muss die Library `cs16711.lib` eingebunden werden, welche von Texas Instruments zur Verfügung gestellt wird.

Zur Initialisierung der SPI Schnittstelle kann die Funktion `SPI_Init()` verwendet werden. Diese benötigt jedoch noch die Funktion `ConfigMcbsp()`. Diese setzt die Register des McBsp's so, dass er danach als SPI Schnittstelle verwendet werden kann. Es kann angegeben werden, ob die Übernahme bei der ersten oder zweiten Flanke geschieht. Die Angaben müssen mit jenen des Displays übereinstimmen. Eine genaue Beschreibung der verschiedenen Register findet man in der Hilfe des Code Composer Studios. Für die Drehzahlmessung wurden folgende Einstellungen gewählt:

DORD (DataORder): MSB wird zuerst gesendet.

`MCBSP_XCR_XWDREVRSDISABLE` auf 0 setzen

CPOL (ClockPOLarity): Ruhezustand von SCK low.

`MCBSP_PCR_CLKXP_RISING` auf 0 setzen

`MCBSP_SPCR_CLKSTP_NODELAY` auf 01 setzen

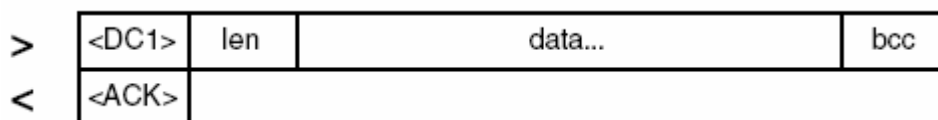
CPHA (ClockPHase): Übernahme bei 1. Flanke

`MCBSP_RCR_RCOMPAND_MSB` auf 00 setzen

Dies bedeutet, dass die drei Anschlüsse des Displays (10, 14, 15) auf GND gesetzt werden müssen.

## 11.4. Datenübertragungsprotokoll

Die Datenübertragung ist jeweils eingebettet in einen festen Rahmen mit Prüfsumme (Protokollpaket). Das Display quittiert dieses Paket mit dem Zeichen <ACK> bei erfolgreichem Empfang oder <NAK> bei fehlerhafter Prüfsumme oder Empfangspufferüberlauf. In jedem Fall wird bei <NAK> das komplette Paket verworfen und muss noch mal gesendet werden. Wird keine Quittierung gesendet, so ist mind. ein Byte verloren gegangen. Wenn nicht innerhalb des eingestellten Timeouts von 2 Sekunden die noch fehlenden Bytes empfangen werden, wird das komplette Paket verworfen und muss nochmals übertragen werden. Die Anzahl der Rohdaten pro Paket ist auf max. 64 Byte begrenzt (len ≤ 64) Befehle die grösser als 64 Byte sind (z.B. Bild laden ESC UL ...) müssen auf mehrere Pakete aufgeteilt werden. Alle Daten in den Paketen werden nach korrektem Empfang vom Display wieder zusammengefügt.



len = Anzahl der Nutzdaten in Byte ohne Prüfsumme, ohne <DC1>, max 64 Byte  
 bcc = 1 Byte = Summe aus allen Bytes inkl. <DC1> und len, Modulo 256  
 <DC1> = 17 (dez) = \$11  
 <ACK> = 6 (dez) = \$06

Abbildung 29: Befehle/Daten zum Display senden

Eingerahmt von <DC1>, der Anzahl Daten „len“ und der der Prüfsumme „bcc“ werden die jeweiligen Nutzdaten übertragen. Als Antwort sendet das Display <ACK> zurück.

Die Befehlsfolge <DC2>, 1, S, bcc entleert den Sendebuffer des Displays. Das Display antwortet zuerst mit der Quittierung <ACK> und beginnt dann alle gesammelten Daten



len = Anzahl der Nutzdaten in Byte ohne Prüfsumme, ohne <DC1>  
 bcc = 1 Byte = Summe aus allen Bytes inkl. <DC1> und len, Modulo 256  
 <DC2> = 18 (dez) = \$12  
 <ACK> = 6 (dez) = \$06

Abbildung 30: Inhalt des Senderbuffers anfordern

wie z.B. Touchastendrücker zu senden. Das „S“ muss dabei als Grossbuchstabe übertragen werden, sonst geht es nicht.

Weitere Paketvarianten für Paketwiederholung, Informationen anfordern und Protokolleinstellungen können im Datenblatt (Anhang 4) nachgeschlagen werden.

## 11.5. Makros für die Drehzahlmessung



Abbildung 31: Dialog für Parametereingabe

Beim Einschalten des Drehzahlmesssystems erscheint ein Parameterfenster, in welchem als Standard die Übersetzung auf 3 und die Polpaarzahl auf 12 eingestellt sind. Hier kann entweder mit den Tasten „set“ in eines der zwei Untermenüs zur Änderung der Parameter gewechselt werden, oder mit „U/min“ direkt weiter zur Drehzahlmessung gegangen werden.



Abbildung 32: Dialog für Eingabe der Übersetzung

Dieses Fenster erscheint, wenn im Parameterfenster bei der Übersetzung die Taste „set“ angewählt wurde. Hier kann über die Tastaturfelder die Übersetzung zwischen dem Motor- und dem Lichtmaschinenpulley eingegeben werden und mittels der Taste „Enter“ an den DSP weitergegeben werden. Gleichzeitig wird diese Taste auch zur Rückkehr ins Parameterfenster verwendet.



Abbildung 33: Auswahl der Polpaarzahl

Wird die Taste „set“ bei der Polpaarzahl gedrückt, erscheint eine Auswahl von verschiedenen Polpaarzahlen. Diese können nicht freigewählt werden, da sie in der Fahrzeugindustrie genormt sind. Die übliche Polpaarzahl beträgt 12 und ist deshalb als Standard gesetzt. Wird die Taste „set“ ein weiteres Mal angewählt, so kehrt man ohne Änderung ins Parameterfenster zurück.

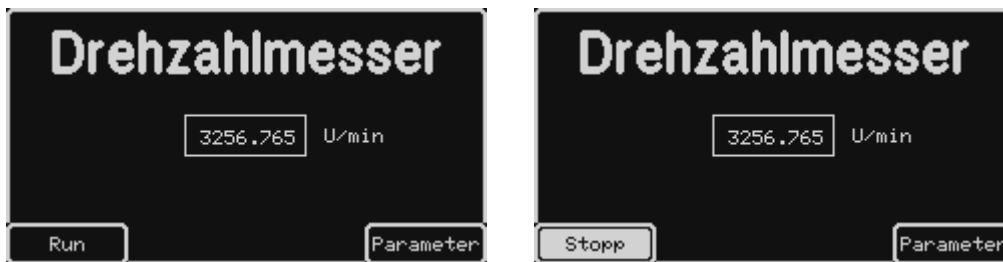


Abbildung 34: Hauptfenster

Vom Parameterfenster aus kann man ins Menü der Drehzahlmessung wechseln. Hier wird eine Messsequenz mit der Taste „Run“ / „Stopp“ gestartet, bzw. gestoppt. Während der Messung wird die Drehzahl im Fenster fortlaufen aktualisiert. Im gestoppten Zustand kann auch hier wieder über die „Parameter“ Taste ins gleichnamige Fenster gewechselt werden.

## 12. Tauglichkeit dieses Messverfahrens

Es hat sich gezeigt, dass dieses Verfahren die Drehzahl zu bestimmen nicht sehr zuverlässig ist, aber auch diverse Vorteile mit sich bringt.

Von grossem Vorteil ist, dass dieses Messverfahren sehr schnell startbereit ist. Im Grunde genommen muss nur der Zigarettenanzünd-Adapter eingesteckt werden, und die Drehzahlmessung kann beginnen. Ebenfalls ist das System sehr flexibel. Es kann bei Benzinmotoren genauso wie bei Dieselmotoren eingesetzt werden. Theoretisch wäre die Messung auch bei Booten möglich. Im Umfang dieses Projektes konnte diese jedoch nie getestet werden.

Ein grosser Nachteil ist ganz klar, dass das Übersetzungsverhältnis bekannt sein muss, um die Drehzahl genauer zu bestimmen. Dieses Verhältnis ist für jedes Fahrzeugmodell unterschiedlich. Diese Zahl wissen leider meistens nicht einmal die Auto-Händler selbst. In den technischen Daten zum Fahrzeug sind diese im Normalfall auch nicht enthalten. Eine Möglichkeit dieses zu bestimmen wäre, eine bestimmte Drehzahl zu fahren, und dann das Gerät zu kalibrieren. Diese Idee scheitert jedoch daran, dass nicht jedes Fahrzeug über einen Drehzahlmesser verfügt, bzw. die vorhandenen Drehzahlmesser sehr ungenau sind. Zudem ist es relativ schwierig konstant eine exakte Drehzahl zu fahren. Also führt die einzige Methode dieses Verhältnis exakt zu bestimmen, über das Ausmessen des Umfangs der Pulleys. Dies kann je nach Fahrzeug nur mit sehr grossem Aufwand gemacht werden.

Ein weiterer Nachteil ist, dass moderne Autos zum Teil über sehr stabile Bordnetze verfügen, auf welchen fast keine Frequenzrippel von der Lichtmaschine mehr sichtbar sind. Eine solche Ausführung konnten wir bei einem Mercedes-Benz 270CDI ausmessen. Bei solchen Fahrzeugen wird natürlich auch dieses Messverfahren verunmöglicht. In Zukunft werden diese wahrscheinlich immer häufiger anzutreffen sein.

Einzelne Fahrzeuge verfügen auch über so genannte Freilauflichtmaschinen. Diese weisen einen freilaufenden Rotor auf, was dazu führt, dass wenn der Motor abgebremst wird die Lichtmaschine weiterdrehen kann. Dies ist für die Messung nicht von so grosser Bedeutung, da die Leistung beim Abbremsen des Fahrzeugs nicht von sehr grossem

Interesse sein wird. Genauere Informationen zum Funktionsprinzip der Freilauflichtmaschine sind im Abschnitt über die Lichtmaschinen zu finden.

## 13. Summary

Ziel dieser Arbeit war es, ein Verfahren für ein Drehzahlmesssystem zu entwickeln, welches als Referenz nur die Spannung des Bordnetzes im Fahrzeug benötigt. Zudem sollte auch gleich ein Prototyp gebaut werden, der für erste Testversuche eingesetzt werden kann.

Die Zielvorgaben konnten weitgehend erfüllt werden. Es wurde erhofft, dass man mit diesem Messprinzip auf allen Fahrzeugen ein gutes Resultat erzielen kann. Bei den ersten Studien über das Messverfahren, wie es in diesem Projekt angewendet wird, mussten wir jedoch feststellen, dass dies leider nicht der Fall ist. Für weitere Details verweisen wir auf das Kapitel 12 in diesem Dokument. In den Fahrzeugen, welche wir zu Testzwecken zur Verfügung hatten, konnte dieses System jedoch mit erfreulichen Resultaten eingesetzt werden. Für die Ermittlung der Drehzahl werden verschiedenste Methoden aus der digitalen Signalverarbeitung angewandt. Zusätzlich wird für die Optimierung der Drehzahlbestimmung ein Algorithmus aus der dynamischen Programmierung eingesetzt, welcher seinen Einsatz auch in Mobiltelefonen findet. Der Prototyp kann nun an Fahrzeuge angeschlossen werden und deren Drehzahl kann auf dem integrierten Display oder auf einer externen Auswertungsstation abgelesen werden.

Dieses Projekt war sehr interessant, fordernd, spassig bis nervenaufreibend. Die Messdatenerfassung bereitete uns viel Spass, da wir verschiedene Fahrzeuge testen konnten. Die Signalverarbeitung war sehr interessant und half uns die gelernte Theorie aus der Vorlesung in einem praktischen Umfeld anzuwenden. Dadurch konnten wir dieses Wissen sehr gut vertiefen. Die Theorie über den Viterbi, war ebenso interessant, als aber auch fordernd, da wir bis anhin keine grossen Kenntnisse über diesen Algorithmus verfügten. Die Implementierung erforderte dabei sehr viel Konzentration. Die Projektstudien in der Matlab-Umgebung waren lehrreich, da wir so eine sehr leistungsstarke Entwicklungsplattform kennen lernen konnten, welche wir auch später in unserer Karriere einsetzen können. Viel Nerven dagegen kostete uns die Implementierung des Programmes auf der DSP-Plattform, sowie die Ansteuerung des Displays, da uns teilweise kleine, gut versteckte Fehler das Leben unnötig schwer machten. Der Projektplan konnte leider nicht ganz so eingehalten werden wie im Voraus geplant. Dies ist hauptsächlich darauf zurück zu führen, dass das Erstellen des

Detektions-Algorithmus zum Zeitpunkt als der Projektplan erstellt wurde, als beinahe fertig erachtet wurde. Dies war jedoch noch nicht der Fall, wie wir später herausfinden mussten. Die Anwendung des Viterbi-Algorithmus war danach noch ziemlich Zeit aufwändig, da in der Literatur zum Teil schwer Verständliche Notationen verwendet wurden. Dadurch verschob sich die gesamte Projektplanung nach hinten. Jedoch konnten wir das Projekt dennoch auf den vereinbarten Zeitpunkt fertig stellen.

## 14. Verbesserungen

Das Projekt ist noch keineswegs Perfekt, es kann in unterschiedlichen Punkten weiter verbessert werden. Ein schönes Feature wäre beispielsweise eine Möglichkeit für eine automatische Kalibrierung einzubauen. Dies hätte einen grossen praktischen Nutzen für Anwender welche keine Kenntnisse von dem Übersetzungsverhältnis haben. Das Messresultat wird dadurch jedoch keineswegs genauer. Eine Möglichkeit für dieses Feature wäre, dass der Benutzer auf eine bestimmte Drehzahl beschleunigt und dann per Tastendruck diese bestätigt. Aus diesen Daten könnte die Übersetzung zurück gerechnet werden. Um eine höhere Auflösung in der Frequenz bzw. Drehzahl zu erhalten, bieten sich verschiedene Möglichkeiten an. Der naheliegendste Ansatz wäre, einfach mehr Samples abzuwarten, und den ganzen Algorithmus mit diesem längeren Vektor durch zurechnen. Dies hätte jedoch einen erheblich grösseren Zeitaufwand zur Folge. In eine ähnliche Richtung würde die Zero-Padding Version führen. Hierfür müsste der Datenvektor vor der Fouriertransformation mit Nullen gestreckt werden. Die Kosten dafür sind jedoch wiederum der höhere Rechenaufwand. Der eleganteste Verbesserungsansatz wäre, die Frequenzauflösung an der Stelle, wo der relevante Frequenzpeak auftritt, nachträglich zu erhöhen. Diese Lösung ist etwas aufwändig, jedoch nicht mit einem so erheblichem Zeitaufwand verbunden, wie die vorgängigen Ansätze.

## 15. Literaturverzeichnis

- [ 1 ] TMS320C67x DSP Library Programmer's Reference Guide (SPRU657.pdf)
- [ 2 ] Prof. Dr. G. M. Schuster. Viterbi-Algorithmus. A video compression scheme with optimal bit allocation among segmentation, motion and residual error, S. 68-78, Juni 1996 ([www.medialab.ch/guido](http://www.medialab.ch/guido))
- [ 3 ] Wikipedia. Viterbi-Algorithmus / HMM.  
(<http://de.wikipedia.org/wiki/Viterbi-Algorithmus>)
- [ 4 ] M Demleitner. Viterbi\_Algorithmus. Der Viterbi-Algorithmus.  
([www.cl.uni-heidelberg.de/kurs/ws04/stat/html/page030.html](http://www.cl.uni-heidelberg.de/kurs/ws04/stat/html/page030.html))
- [ 5 ] Dr.-Ing. P. Solfrank, Dipl.-Ing. P. Kelm. Freilauflichtmaschine. Dynamiksimulation von PKW-Nebenaggregatantrieben, Juni 1999.  
([www.ina.de/inaupdate/news/publics/present/fach\\_pkw\\_nebenaggregat/fa\\_nebenaggregat\\_de.asp](http://www.ina.de/inaupdate/news/publics/present/fach_pkw_nebenaggregat/fa_nebenaggregat_de.asp))
- [ 6 ] Wikipedia. Lichtmaschine / Generator.  
(<http://de.wikipedia.org/wiki/Lichtmaschine>)
- [ 7 ] Kraus Messtechnik GmbH. Lichtmaschine / Generator. Datenblatt RPM-8000.  
([www.kmt-gmbh.com](http://www.kmt-gmbh.com))
- [ 8 ] U. Tuor. Display EA eDIP240-7. EA eDIP240.doc, Mai 2005.  
(<http://www.medialab.ch/ds/software/EA-KIT>)
- [ 9 ] Electronic Assembly. Display EA eDIP240-7. Datenblatt edip240-7, Mai 2005  
(<http://www.lcd-module.de/deu/dbl/dbl.htm>)
- [ 10 ] Dr. A. Schüeli. Digitale Signalverarbeitung. Skript zu Vertiefungsfach „Digitale Signalverarbeitung“, Oktober 2002

## 16. Anhang

- [ 1 ] Typical Applications LM340 (LM340.pdf)
- [ 2 ] Data sheet LM2940/LM2940C 1A Low Dropout Regulator (LM2940.pdf)
- [ 3 ] Data sheet PNP Transistor (sb1232413-00366-0-2SB1232.pdf)
- [ 4 ] Datenblatt Display EA eDIP240-7 (edip240-7.pdf)
- [ 5 ] Code Composer File für Twiddle-Konstanten (tw\_r2fft.c)
- [ 6 ] Code Composer File Main-Funktion (uMin\_v6\_25.c)
- [ 7 ] Code Composer File Viterbi (viterbi2.c)
- [ 8 ] Agenden
- [ 9 ] Sitzungsprotokolle