Inhaltsverzeichnis

[1 Aufgabenstellung 2](#_Toc368748854)

[1.1 Problemstellung 2](#_Toc368748855)

[1.2 Aktuelle Situation 2](#_Toc368748856)

[1.3 Ziel der Bachelorarbeit 2](#_Toc368748857)

[2 Das Model 3](#_Toc368748858)

[2.1 Mechanische Komponenten 3](#_Toc368748859)

[2.2 Elektronische Komponenten 4](#_Toc368748860)

[3 Entwicklung des Algorithmus 5](#_Toc368748861)

[3.1 Kompensationsansatz 5](#_Toc368748862)

[3.3 Entwicklung des Regleralgorithmus 8](#_Toc368748863)

[3.4 Bestimmen der Parameter 9](#_Toc368748864)

[4 Programmierung 10](#_Toc368748865)

[4.1 „Main“- Funktion 10](#_Toc368748866)

[4.2 Funktion: Dateneinlesen 11](#_Toc368748867)

[4.3 Funktion: LCDansteuern 11](#_Toc368748868)

[4.3 Funktion: Ausgangansteuern 11](#_Toc368748869)

[4.4 Sonstige Funktionen 12](#_Toc368748870)

[5 Zusammenfassung 13](#_Toc368748871)

[5.1 Stand am Ende des Bearbeitungszeitraumes 13](#_Toc368748872)

[5.2 Verbesserungsmöglichkeiten 13](#_Toc368748873)

[5.3 Schlussfolgerung 13](#_Toc368748874)

[6 Literaturverzeichnis 14](#_Toc368748875)

[7 Anhänge 15](#_Toc368748876)

[7.1 Programmcode 15](#_Toc368748877)

[7.2 readme.txt 20](#_Toc368748878)

[7.3 Schaltplan 21](#_Toc368748879)

[7.4 Testprotokoll 22](#_Toc368748880)

[7.5 Kopien der Anmeldung 23](#_Toc368748881)

# 1 Aufgabenstellung

## 1.1 Problemstellung

Bei der Installation von Offshorewindanlagen ist es wichtig, dass eine an einem Kran hängende Last gleichmäßig in der gleichen Höhe über dem Meeresgrund gehalten werden kann. Die Position der Last ändert sich jedoch durch den Seegang konstant. Hierdurch wird eine Kompensation des Seegangs nötig.

Es gibt mehrere Möglichkeiten, den Seegang zu kompensieren:

* Eigenantrieb
* Rumpfform
* Stabilisatoren
* Position des Krans auf dem Schiff möglichst mittig
* regelungstechnische Kompensation

Alle Kompensationsmöglichkeiten haben die Gemeinsamkeit, dass sie den Seegang nicht zu 100% kompensieren können.

## 1.2 Aktuelle Situation

Aktuell werden zur Seegangskompensation die pneumatischen Systeme der Firma Bosch Rexroth verwendet. Hier ist mir über die regelungstechnische Umsetzung nichts bekannt.

Des Weiteren gibt es einen Ansatz mittels Prognosealgorithmus, siehe: Sebastian Küchler

„Aktive Seegangskompensation mit beobachtergestützter Prognose der vertikalen Schiffsbewegung“.

## 1.3 Ziel der Bachelorarbeit

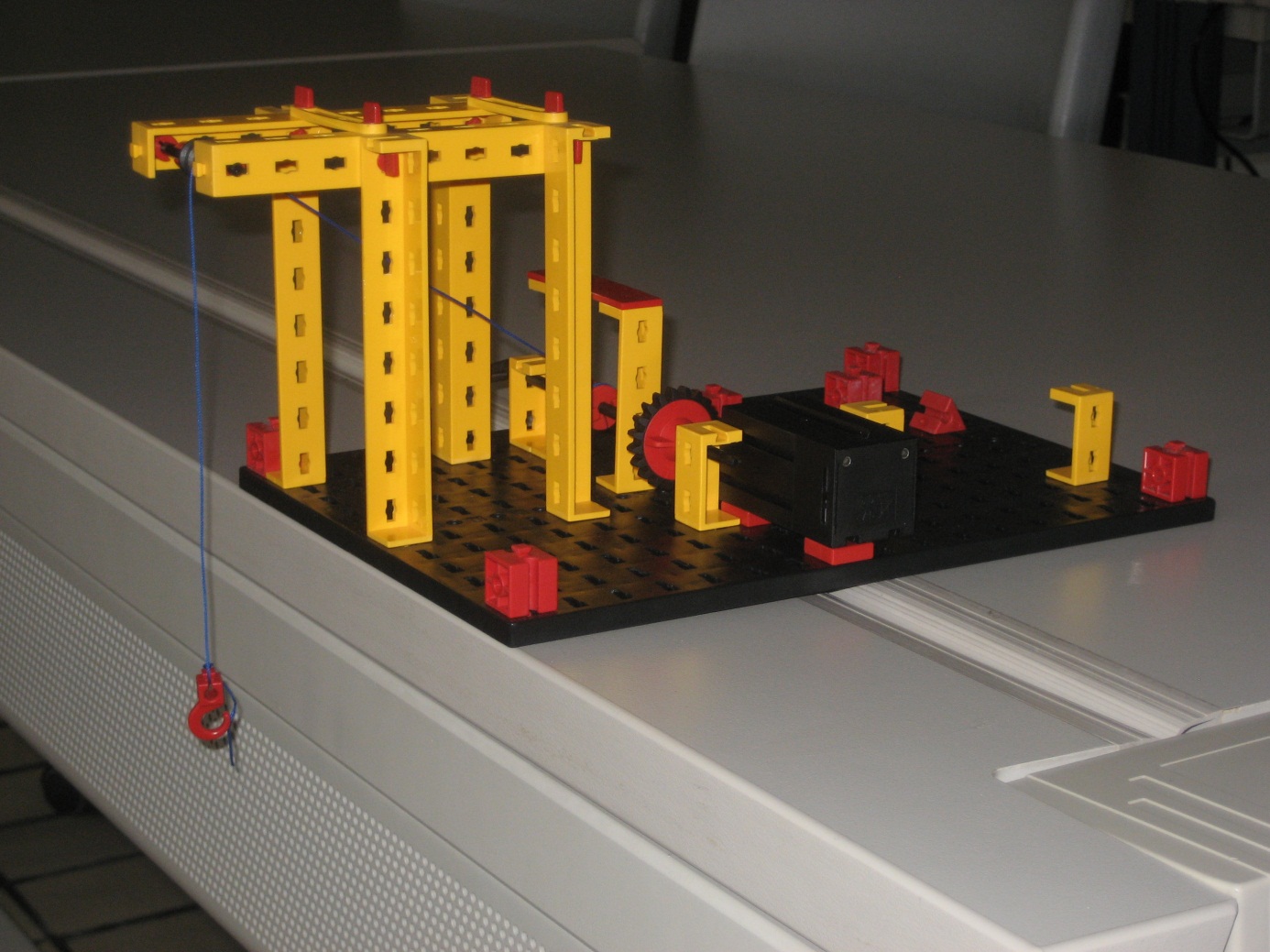
Da im Rahmen dieser Bachelorarbeit nur eine regelungstechnische Möglichkeit zur Seegangskompensation gefunden werden soll, wird davon ausgegangen, dass durch alle anderen Kompensationsmethoden das Schiff auf einer Position gehalten wird und es sich nur noch in der Horizontalen bewegen kann.

Ziel der Bachelorarbeit ist es also, die durch Seegang verursachte Höhenänderung zu kompensieren. Dies soll ohne pneumatische Komponenten und ohne komplizierte Prognosealgorithmen geschehen. Die Hauptstörgröße in Form der Höhenänderung und der Abstand zum Meeresgrund werden mit einem Sensor erfasst. Es soll nun ein Programm entwickelt werden, welches den Motor so ansteuert, dass die Last möglichst auf einer Position gehalten wird.

# 2 Das Model

Da der Fachhochschule kein Schiff mit einem Kran zur Verfügung steht, musste ein Model zu Simulationszwecken gebaut werden.

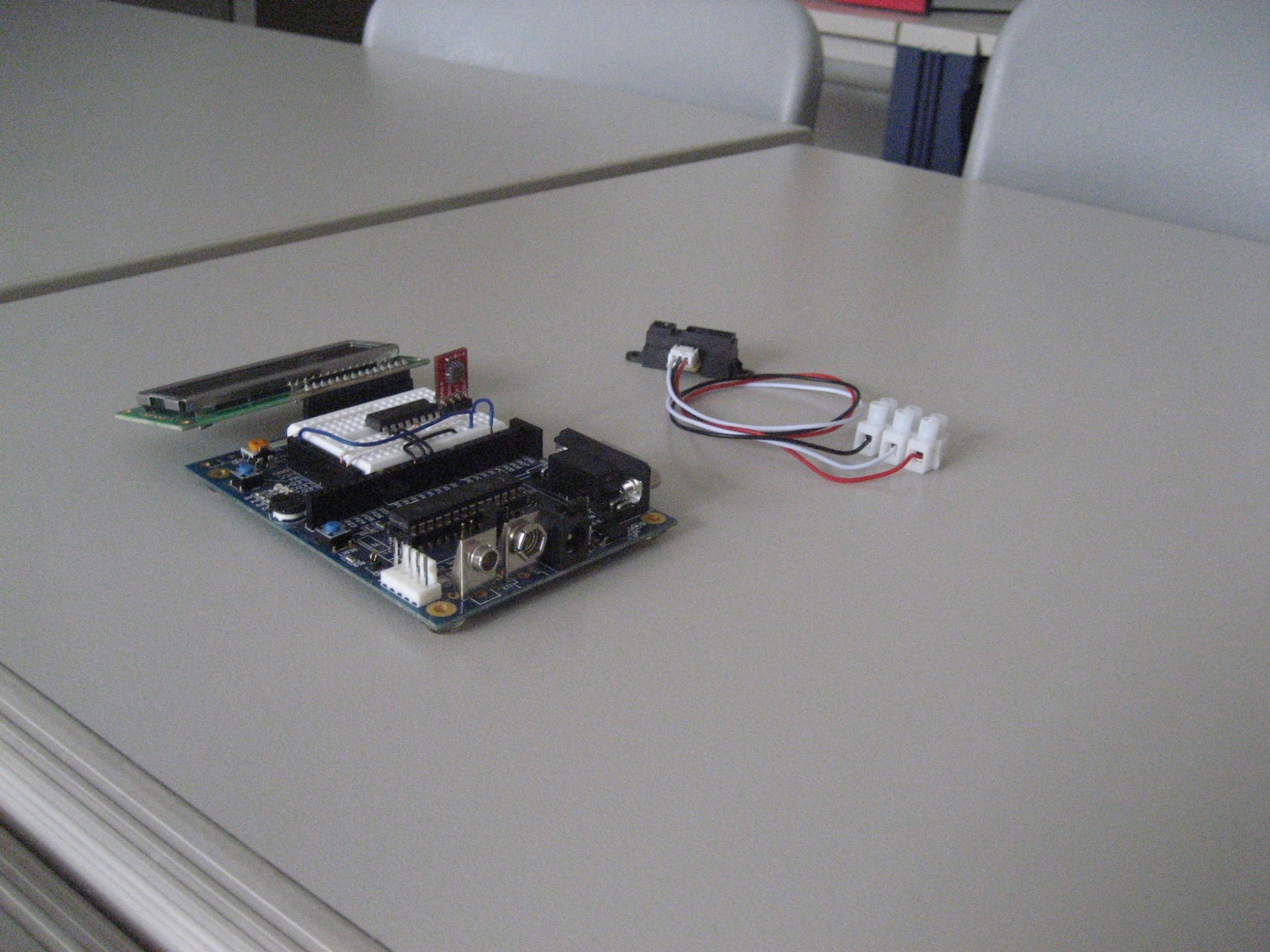
## 2.1 Mechanische Komponenten



Das Model wurde mithilfe eines Baukastens der Firma Fischertechnik gebaut. Die Kraftübertragung vom Motor zur Seilwinde erfolgt über zwei Zahnräder. Motor, Controller und Sensoren sind in das Model integriert. Um den Seegang zu simulieren, wird das Model mittels Seilen und einer Umlenkrolle an der Decke befestigt.

Eine Handkurbel ermöglicht es, den Haken per Handbetrieb zu bewegen.

## 2.2 Elektronische Komponenten



Zur Regelung des Systems wird ein PSoC1 Evaluation Kit mit einem Prozessor vom Typ CY8C27446-24PXI verwendet. Auf dem mitgelieferten LCD-Display wird die Entfernung der Last zum Entfernungssensor angezeigt. Das System wird durch das Board mit einer Versorgungsspannung von fünf Volt versorgt.

Zur Ansteuerung des Motors wird ein Treiberbaustein vom Typ L293 D verwendet. Dieser enthält zwei H-Brücken, von denen eine zur Motorsteuerung verwendet wird. Durch eine Pulsweitenmodulation am enable-Eingang wird die Drehzahl gesteuert.

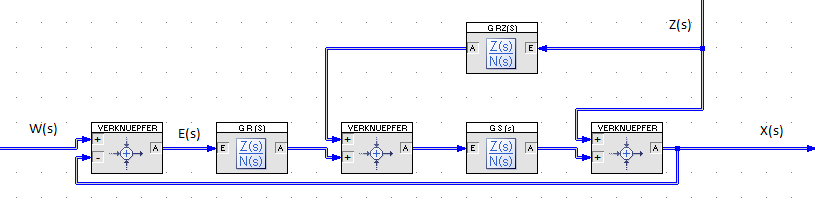
Der Gleichstrommotor der Firma Fischertechnik benötigt eine Versorgungsspannung von neun Volt, daher muss zusätzlich zu der fünf Volt Spannungsversorgung des PSoCs eine neun Volt Spannungsquelle am Motortreiber angeschlossen werden.

Entfernungs- und Beschleunigungssensor geben beide eine analoge Spannung aus. Der Schaltplan ist im Anhang 7.3 beigefügt. Das obige Bild zeigt den Entfernungssensor (rechts) und das PSoC1 Evaluation Board mit Treiberbaustein und Beschleunigungssensor (links).

# 3 Entwicklung des Algorithmus

## 3.1 Kompensationsansatz

Um die durch den Seegang entstehende Höhenänderung zu kompensieren, muss die Last um die gleiche Distanz und in entgegengesetzter Richtung bewegt werden. Die Höhenänderung kann über den Beschleunigungsmesser direkt gemessen werden. Ein Vorteil der Störgrößenaufschaltung ist, dass sie die Stabilitätseigenschaften des Regelkreises nicht verändert (Tröster, 2005). Die Störgröße wird am Ende der Strecke zu deren Ausgang addiert. Dadurch ergibt sich der im folgenden Blockschaltbild dargestellte Aufbau:



Wählt man die Übertragungsfunktion GRZ(s) so, dass sie der Umkehrfunktion der Strecke entspricht, verhält sich das System derart, als hätte man den Additionspunkt hinter die Strecke verschoben. Subtrahiert man nun das Ergebnis von GRZ(s) vom Reglerausgang, anstatt es zu addieren, entspricht das einer Subtraktion der Störgröße am Ende der Regelstrecke. Das heißt, die Störgröße wird vollständig kompensiert.

Da die Erfassung und Verarbeitung der Daten des Beschleunigungssensors eine gewisse Zeit dauert, kann der Seegang nicht zu 100% kompensiert werden. Es kommt zu einer Zeitverschiebung zwischen der Störgröße und dem von der Störgrößenaufschaltung generiertem Signal.

Die Auswirkung der Zeitverschiebung muss nun untersucht werden. Zur einfachen Modelbildung geht man erst einmal davon aus, dass die durch die Wellen hervorgerufene Höhenänderung sinusförmig verläuft. Da die Stabilität des Regelkreises nicht durch die Störgrößenaufschaltung beeinflusst wird, kann der Regler hierbei ignoriert werden. Subtrahiert man nun zu einer sinusförmigen Störgröße ein weiteres Sinussignal mit gleicher Periodendauer und Amplitude, heben diese sich auf. Durch die Zeitverzögerung T kommt es nun aber zu einer Phasenverschiebung zwischen dem zweiten Signal und der Störgröße. Es gilt die Gleichung:

Wendet man nun das Additionstheorem der Trigonometrischen Funktionen (I.N. Bronstein, 2012) an, so erhält man:

Im besten Fall ist die Zeitverschiebung T = 0 und man erhält:

Die Störgröße wird also vollständig kompensiert. Je näher die Verzögerungszeit dem Punkt T = π kommt, desto größer wird die durch die Phasenverschiebung hervorgerufene Regeldifferenz.

Im schlimmsten Fall T = π verdoppelt sich die Amplitude, das heißt die Störgröße wird verstärkt:

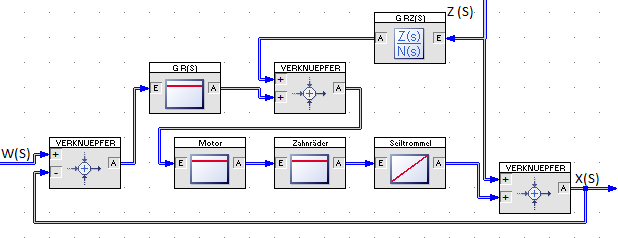
Nähert sich die Verzögerungszeit nun dem Punkt T = 2π an, sinkt die Regeldifferenz und beträgt am Punkt T = 2π null. Um eine Verstärkung der Störgröße zu vermeiden, sollte die Verzögerungszeit möglichst klein gehalten werden.

Da die Periodendauer bei Seegang wesentlich größer als die Zeit ist, die Microcontroller und Sensor zum Erfassen der Daten benötigen, kann das oben genannte Phänomen bei einer Seegangskompensation vernachlässigt werden.

## 3.2 Entwicklung mit Blockschaltbildern

Betrachtet man das Blockschaltbild aus dem letzten Kapitel, muss dieses noch weiter der Realität angepasst werden, da sonst ein Reglerentwurf und eine Simulation unmöglich sind. Es wird davon ausgegangen, dass der Microcontroller schnell genug arbeitet, um ihn wie ein analoges Gerät zu behandeln. Zunächst wird die Übertragungsfunktion der Strecke GS(s) genauer betrachtet. Sie besteht aus zwei P-Gliedern, welche den Motor und die Zahnräder darstellen, und einem I-Glied für die Seiltrommel.

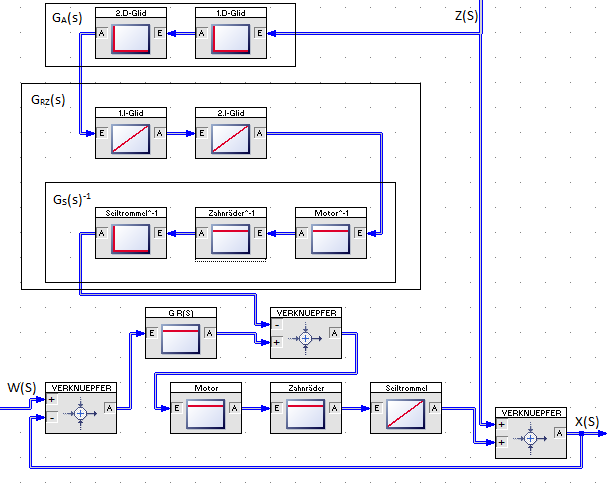
Soll der Regler ein möglichst gutes Führungsverhalten aufweisen, wird ein reiner P-Regler gewählt (Hückelheim, Tabelle Reglerauswahl), der Reglerblock wird also durch einen reinen P-Regler ersetzt.



Die Übertragungsfunktion GRZ(s) setzt sich aus zwei I-Gliedern und der Umkehrfunktion der Strecke GS(s)-1 zusammen. Zusätzlich muss die Übertragungsfunktion des Beschleunigungssensors GA(s) vor GRZ(s) hinzugefügt werden, um aus der Störgröße die Beschleunigung zu bestimmen. Es gilt die in der Mechanik bekannte Gleichung:

Das heißt, die Beschleunigung ist die zweifache zeitliche Ableitung der zurückgelegten Distanz. Die Übertragungsfunktion des Beschleunigungssensors besteht also aus zwei D-Gliedern.

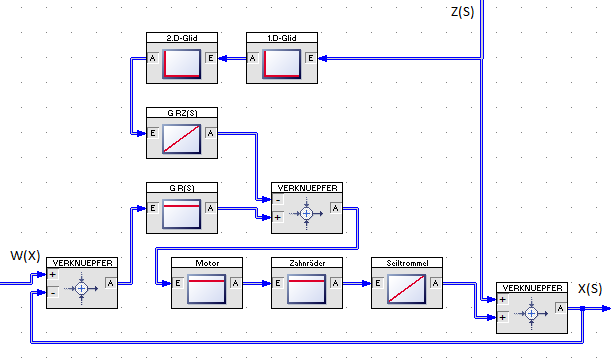
Die zwei I-Glieder werden benötigt, um aus der Beschleunigung wieder die zurückgelegte Distanz zu berechnen.



Nun können die beiden I-Glieder mit dem D-Glied und dem P-Glied der Unterfunktion zusammengefasst werden. Die beiden I-Glieder haben einen Verstärkungsfaktor von eins.

Dadurch erhält man ein I-Glied mit folgenden Parametern:

mit:



## 3.3 Entwicklung des Regleralgorithmus

Aus dem Blockschaltbild lässt sich folgende Gleichung herleiten:

beziehungsweise:

Diese muss zum Einsatz im Programm noch angepasst werden. Da die Regeldifferenz noch nicht bekannt ist, muss diese zunächst noch im Microcontroller mit folgender Gleichung berechnet werden:

Der Microcontroller arbeitet mit zeitdiskreten Signalen. Dies macht eine entsprechende Anpassung der Gleichung erforderlich. Das Integral kann nun durch eine Summe ersetzt werden:

Bei der Entfernungsmessung besteht kein linearer Zusammenhang zwischen gemessener Entfernung und ausgegebener Spannung, somit muss dies im Programm berücksichtigt werden. Der relevante Teil der Kurve zwischen 20 und 60 cm entspricht annähernd der Funktion dritter Ordnung, deren Parameter mit Hilfe des Programms „skidavis“ bestimmt werden können. Die Funktion hat folgende Funktionsgleichung:

Allerdings ist die obenstehende Funktion relativ komplex. Stattdessen kann man auch die Gerade mit der kleinstmöglichen Abweichung von der Funktion ermitteln, wodurch sich folgende, für den Microcontroller einfacher zu berechnende Funktion ergibt:

Setzt man diese Gleichung in die Funktion des Algorithmus ein, so erhält man:

Im Programm wird die Gleichung dann in zwei Schritten berechnet. Zuerst wird die Summe bestimmt:

Anschließend wird die Ausgangsgröße bestimmt:

## 3.4 Bestimmen der Parameter

Zunächst müssen die Streckenparameter bestimmt werden. Der Motor hat bei einer Spannung von neun Volt eine Drehzahl von 360 U/min , das heißt, er verhält sich wie ein P-Glied mit den Parametern:

Die Zahnräder haben eine Zahnzahl von zehn am Antrieb und 21 am Abtrieb, daraus ergibt sich ein P-Glied mit einer Übersetzung von:

Die Seiltrommel hat einen Durchmesser von 0,77 mm. Pro Umdrehung wird einmal der Durchmesser abgewickelt. Daraus ergibt sich ein I-Glied mit:

Fasst man nun die einzelnen Streckenparameter zusammen, so erhält man eine I-Strecke:

Damit können die Parameter der Übertragungsfunktion GRZ(s) bestimmt werden:

Die Reglerparameter können dann bei funktionierender Anlage mittels der Auswertung der Sprungantwort nach den Einstellregeln nach Ziegler/Nichols erfolgen.

# 4 Programmierung

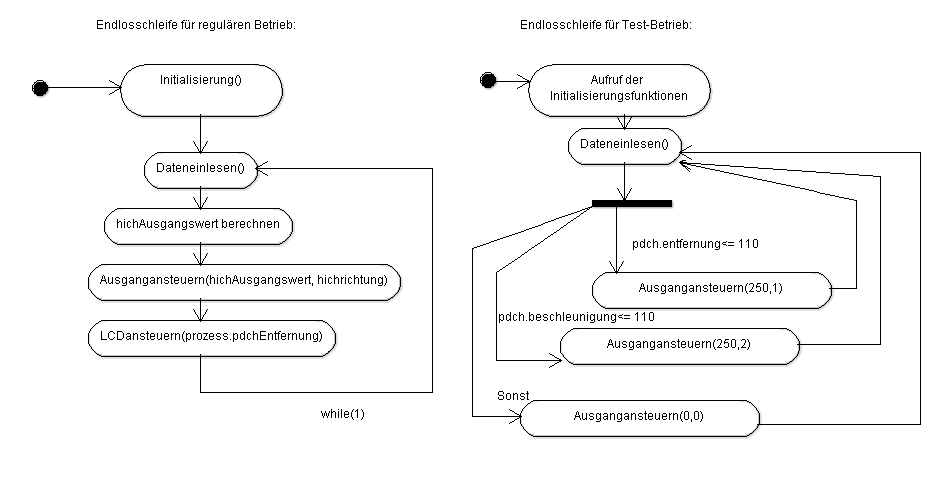
Am Anfang der Programmierung wurde das Chip-Level-Design des PSoC und die Ein- und Ausgänge konfiguriert. Danach konnte mit dem Schreiben des Programmes in C begonnen werden.

Zum Versionsmanagement wird die Internetplattform GitHub benutzt. Dadurch ist jede Programmänderung nachvollziehbar und es kann jederzeit an einen früheren Stand zurückgekehrt werden. Weiterhin existiert immer mindestens eine Sicherheitskopie. Das Projekt ist zu finden unter: <https://github.com/lesky/Batchlorarbeit--Seegangskompensation-bei-Krananlagen>.

Aufgrund der geringen Größe des Projektes wurde in der Readme.txt des Projektes eine Fehlerliste geführt. Auf die Benutzung von Bugtrackingsoftware wie „Mantis“ oder „Bugzilla“ wurde verzichtet.

Zur Parameterübergabe zwischen den einzelnen Funktionen sind sämtliche relevanten Variablen in einer globalen Struktur namens „prozess“ abgelegt.

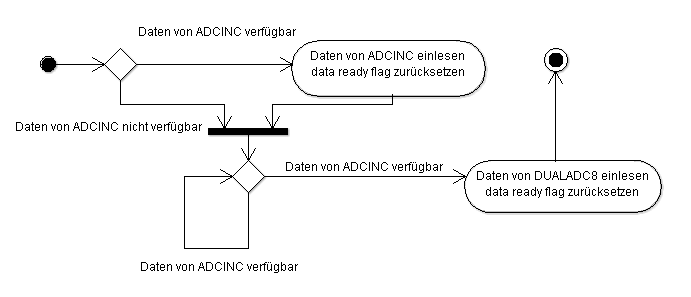
## 4.1 „Main“- Funktion



Zu Beginn der „Main“- Funktion werden sämtliche im Chip-Level-Design festgelegten User Modules initialisiert. Anschließend werden in einer Endlosschleife die Daten eingelesen, verarbeitet und ausgegeben.

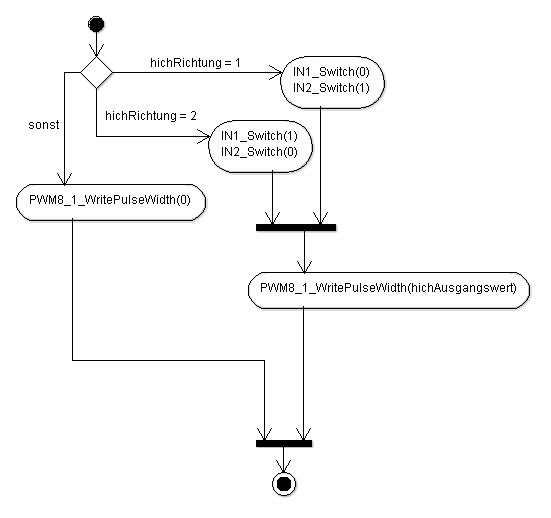
Zu Testzwecken wurde in der „Main“- Funktion eine weitere Endlosschleife programmiert. Diese wird anstelle der oben genannten Schleife kompiliert, wenn der Präprozessor TEST definiert ist. In ihr wurden sämtliche Tests programmiert. Das obige Diagramm zeigt bereits die in Kapitel 5.1 beschriebene Auswertung der von den Tastern generierten Signale.

## 4.2 Funktion: Dateneinlesen



Das Einlesen der Daten erfolgt in zwei Schritten. Zuerst wird überprüft, ob die Daten des für den Sollwert zuständigen A/D-Wandlers verfügbar sind. Sind diese verfügbar, werden sie eingelesen. Anschließend wird das „data ready flag“ des A/D-Wandlers zurückgesetzt. Der Sollwert ist nicht zeitkritisch, wodurch hier nicht auf die Daten gewartet wird. Als nächstes werden die Beschleunigung und die Entfernung abgefragt. Sind diese Daten nicht verfügbar, wird so lange gewartet, bis die Daten erfasst werden können. Anschließend wird auch hier das „data ready flag“ zurückgesetzt und die Funktion verlassen.

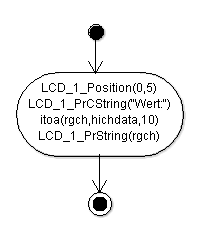
## 4.3 Funktion: Ausgangansteuern



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| hichRichtung | IN1 | IN2 | Enable | Motor |
| 1 | 1 | 0 | PWM | Rechtslauf |
| 2 | 0 | 1 | PWM | Linkslauf |
| sonst | egal | egal | 0 | Leerlauf |

Die Funktion steuert den Motor an. Ihr werden zwei Parameter vom Datentyp character übergeben. Die Variable hichAusgangswert entspricht hierbei der Pulsweite und die Variable hichRichtung der Drehrichtung. Wird der Funktion ein Wert von eins oder zwei übergeben, dreht sich der Motor in die entsprechende Richtung (siehe Tabelle rechts). Bei jedem anderen Wert wird der „enable“- Eingang des Motortreibers auf null gesetzt und der Motor läuft im Leerlauf.

## 4.4 Sonstige Funktionen

Die Funktion LCDansteuern gibt eine Zahl vom Datentyp character zusammen mit dem Text „Wert:“ auf dem LCD-Display aus. Zuerst wird der Text auf dem Display angezeigt. Da es nicht möglich ist, die Zahl, wie bei C üblich, direkt mittels %c mit dem Text auszugeben, muss hierfür die Funktion „iota()“ aus der C Standard-Bibliothek „stdlib.h“ genutzt werden.

Um den Code übersichtlicher zu gestalten, sind sämtliche Initialisierungen in die Funktion Initalisierung ausgelagert. Diese wird zu Beginn der „Main“- Funktion einmal aufgerufen.

# 5 Zusammenfassung

## 5.1 Stand am Ende des Bearbeitungszeitraumes

Leider konnte die Seegangskompensation bis zum Ende des Bearbeitungszeitraumes nicht realisiert werden. Dies hat zwei Gründe: Zum einen wurde während der Tests kurz vor Abgabetermin festgestellt, dass der Beschleunigungssensor defekt ist und es konnte aufgrund von zu langen Lieferzeiten nicht rechtzeitig ein Ersatz beschafft werden. Zum anderen sind die Ergebnisse, die die A/D-Wandler dem Programm übergeben, nicht nachvollziehbar (siehe Testprotokoll Anhang 7.5). Hierfür konnte die Ursache nicht gefunden werden.

Das Ansteuern des Motors funktioniert dafür fehlerfrei.

Auch die Funktion des Algorithmus konnte mit Hilfe der Simulationssoftware BORIS nachgewiesen werden.

Nach Absprache mit Herrn Stief wurde das Programm nun so geändert, dass der Motor nun mit zwei Tastern so angesteuert wird, dass sich der Hacken vertikal bewegt.

Um den bereits programmierten Algorithmus nicht zu beeinträchtigen, wurde die Tasterabfrage in der Testschleife programmiert.

## 5.2 Verbesserungsmöglichkeiten

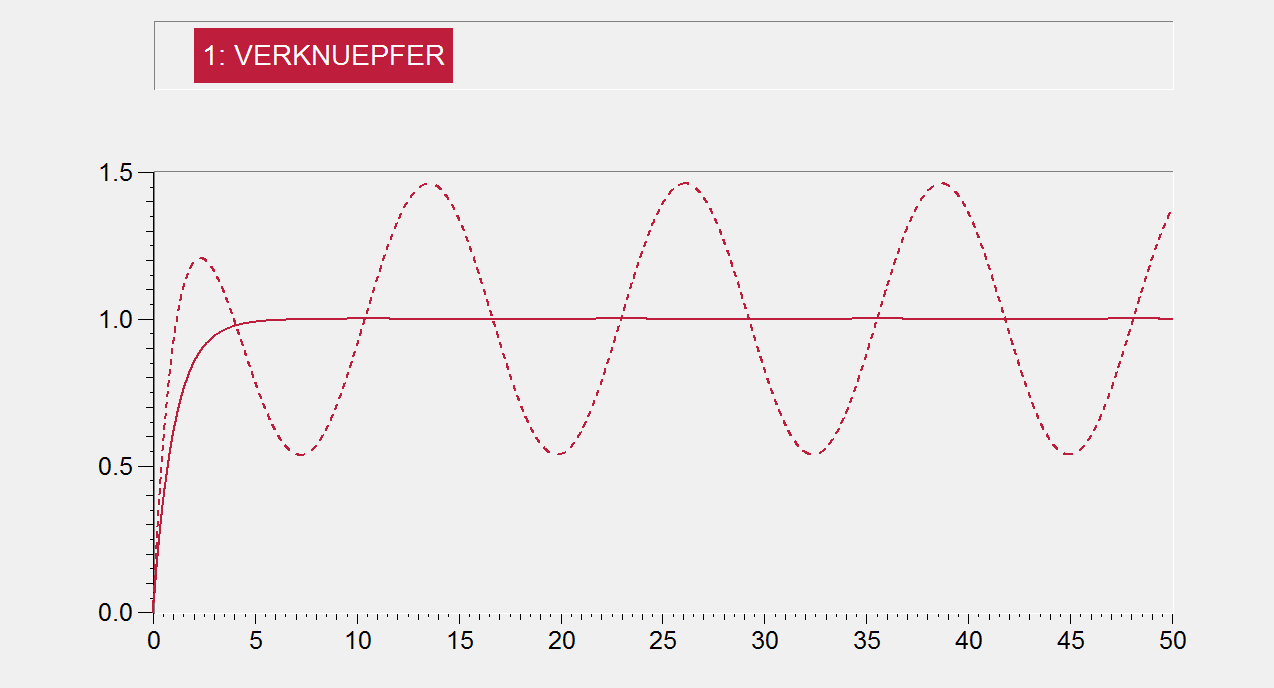
Die wichtigste Verbesserung wäre die Korrektur der oben genannten Fehler. Des Weiteren kann die Regelung noch an folgenden Stellen optimiert werden:

* Verwenden schnellerer Sensoren und eines schnelleren Prozessors;
* Ersetzen der analogen Sensoren, durch welche mit einer digitalen Datenübertragung zum Microcontroller, da die A/D-Wandler im direkten Vergleich relativ langsam sind.

Beide Änderungen sorgen dafür, dass die im Kapitel 3.1 genannte Verzögerungszeit kleiner wird und damit der Seegang besser kompensiert werden kann.

## 5.3 Schlussfolgerung

Die Aufgabenstellung wurde im Rahmen dieser Bachelorarbeit nicht vollständig erfüllt. Zwar ist die Entwicklung der Seegangskompensation, und damit der Hauptteil der Bachelorarbeit, abgeschlossen, aber aufgrund des defekten Beschleunigungssensors und der Fehler der A/D-Wandler konnte der Algorithmus nicht im Model getestet werden. Die Simulation im Computer zeigt jedoch, dass der Algorithmus prinzipiell funktioniert. Auch wurde die hardwaremäßige Umsetzung komplett geplant und größtenteils umgesetzt. Das nachfolgende Diagramm zeigt den Verlauf am Reglerausgang, einmal mit Störgrößenaufschaltung (durchgezogene Linie), sowie ohne (gestrichelte Linie).



# 6 Literaturverzeichnis

(kein Datum). Abgerufen am 22. 8 2013 von rn-wissen: http://www.rn-wissen.de/index.php/Getriebemotoren\_Ansteuerung

Bosch Rexroth AG. (3 2012). Sicher durch Wellen Sekundärgeregelte Antriebe Kompensieren den Seegang. *drive&control*, S. 2-3.

Chacon, S. (2009). *Pro Git (Expert's Voice in Software Development).* http://git-scm.com/book.

Hückelheim, K. (kein Datum). *Elementare Übertragungsglieder*. Abgerufen am 12. 09 2013 von https://elearning.fh-frankfurt.de

Hückelheim, K. (kein Datum). *Tabelle Reglerauswahl*. Abgerufen am 12. 09 2013 von https://elearning.fh-frankfurt.de

I.N. Bronstein, K. S. (2012). *Tsaschenbuch der Mathematik.* Frankfur am Main: Wissenschaftlicher Verlag Harri Deutsch GmbH.

Lehmann, H. L. (2008). *Taschenbuch der Elektrotechnik und Elektronik.* Münchem.

Papula, L. (2009). *Mathematiche Formelsammlung für Ingeniere und Naturwissenschaftler.*

Rabe, G. L. (kein Datum). *github*. Abgerufen am 28. 08 2013 von https://github.com/lesky/Weather-Station/

Tröster, F. (2005). *Steuerungs- und Regelungstechnik für Ingenieure.* München: Oldenburg Wissenschaftsverlag GmbH.

# 7 Anhänge

## 7.1 Programmcode

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* C main-Funktion \*

\* Programm: Seegangskompensation bei Krahnanlagen \*

\* Controler: CY8C27446-24PXI \*

\* \*

\* I/O-Konfiguration: \*

\* LCD -> Port 2 \*

\* Selbsttest -> 1.1 \*

\* IN1 -> 1.2 \*

\* IN2 -> 1.3 \*

\* PWM 1 -> 1.4 \*

\* Beschleunigung-> 0.7 \*

\* Entfernung -> 0.4 \*

\* Sollwert -> 0.5 \*

\* \*

\* Zusätzlich definierte Prüfixe zur Variablenbenennung \*

\* Mittels Ungaricher Notation nach Charles Simonyi: \*

\* pd -> Prozessbezogene Daten \*

\* ko -> Konstanten \*

\* hi -> Interne Hilfsgrößen \*

\* in -> Datentyp Integer \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <m8c.h>

#include "PSoCAPI.h"

#include <stdlib.h>

// Präprozessor:Für testzwecke

// #defein Test nicht auskomentieren:

#define TEST

// Funktionsprototypen:

void LCDansteuern(char);

void Dateneinlesen(void);

void Ausgangansteuern(char, char);

void Initalisierung(void);

// globale Structur zur Übergabe der Prozessdaten:

struct

{

char pdchBechleunigung, pdchEntfernung;

char pdchSollwert;

} prozess;

void main(void)

{

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Deffinitionen und Deklarationen \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

// Konstanten

char kochKR;

char kochKRZ;

// Variablen

char hichAusgangswert;

char hichBeschleunigungssumme;

// Initialisierung des Controllers

Initalisierung();

// Präprozessor: kompiliere whileschleife wenn kein test

#ifndef TEST

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Endlosschleife für Dauerbetrieb \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

while(1)

{

// Daten Einlesen

Dateneinlesen();

// Parameter Berechnen

hichBeschleunigungssumme = hichBeschleunigungssumme + prozess.pdchBechleunigung;

hichAusgangswert = ( prozess.pdchSollwert - prozess.pdchBechleunigung ) \* kochKR - kochKRZ \* hichBeschleunigungssumme;

// Daten Ausgeben:

Ausgangansteuern(hichAusgangswert);

LCDansteuern(prozess.pdchEntfernung);

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Achtung die Folgenden Zeilen dienen nur zu Testzwecken\*

\* und sind für das Fertige Programm nicht relevant. \*

\* Sie werden anstelle der Regulären while Schleife \*

\* Kompiliert, wenn die Bedingung #ifndef TEST nicht \*

\* erfüllt ist. \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#else

// whileschleife zu testzwecken

// Konstante zur verweildauer in der schleife

while (1)

{

Dateneinlesen();

//Ausgangansteuern(250, 2);

// LCDansteuern(1);

//if (PRT0DR & 0x01)

//{LCDansteuern(1);}

//else {LCDansteuern(PRT0DR & 0x01);}

//LCDansteuern(PRT0DR & 0x01);

// Erkennen von eingangssignalen über A/D-Wandler

if (prozess.pdchEntfernung <= 110)

{

LCDansteuern(1);

Ausgangansteuern(250,1);}

else if (prozess.pdchBechleunigung <= 110)

{ LCDansteuern(2);

Ausgangansteuern(250,2);}

else {

Ausgangansteuern(0,0);

}

};

// Präprozessor: Ende der Verzweifung

#endif

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Funktionsdeklarationen \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void LCDansteuern(char hichdata)

{

char rgch[5];

// Text auf LCD ausgeben

LCD\_1\_Position(0,5);

LCD\_1\_PrCString("Wert:");

// Zahl Auf LCD ausgeben

itoa(rgch,hichdata,10);

LCD\_1\_PrString(rgch);

}

void Dateneinlesen(void)

{

// Wenn Sollwertdaten bereit sind

if(ADCINC\_fIsDataAvailable() != 0)

// Einlesen des Sollwertes

// data ready flag zurüvksetzen

prozess.pdchSollwert = ADCINC\_cClearFlagGetData();

// Auf Entfernung und Position Warten

while(DUALADC8\_fIsDataAvailable == 0);

// Einlesen der Beschleunigung

prozess.pdchBechleunigung = DUALADC8\_cGetData1();

// Einlesen der Entfernung

// data ready flag zurüvksetzen

prozess.pdchEntfernung = DUALADC8\_cGetData2ClearFlag();

}

void Ausgangansteuern(char hichAusgangswert, char hichRichtung)

{

// Drehrichtung Auswählen

switch(hichRichtung) {

// Linkslauf

case 1:

// IN1 und IN2 Ansteuern

IN1\_Switch(0);

IN2\_Switch(1);

break;

// Rechtslauf

case 2:

// IN1 und IN2 Ansteuern

IN1\_Switch(1);

IN2\_Switch(0);

break;

// Ungültiger Wert

default:

// Pulsweite auf 0 setzen

PWM8\_1\_WritePulseWidth(0);

// Funktion verlassen

return ;

}

// Pulsweite auf hichAusgangswert setzen

PWM8\_1\_WritePulseWidth(hichAusgangswert);

}

void Initalisierung(void)

{

//globale Interrupts Freigeben

M8C\_EnableGInt;

// Initialisieren des LCD-Displays

LCD\_1\_Start();

// Initialisieren des PWM-Moduls

// PWM8\_1\_WritePeriod(kochPeriodendauer);

PWM8\_1\_Start();

PGA\_1\_Start(PGA\_1\_HIGHPOWER);

PGA\_2\_Start(PGA\_2\_HIGHPOWER);

PGA\_3\_Start(PGA\_3\_HIGHPOWER);

// Initialisieren des Dualen AD-Wandlers

// für Entfernung und Beschleunigung

DUALADC8\_Start(DUALADC8\_HIGHPOWER);

DUALADC8\_GetSamples();

// Initialisieren des AD-Wandlers

// für den Sollwert

ADCINC\_Start(ADCINC\_HIGHPOWER);

ADCINC\_GetSamples(0);

//Initialisieren der Digitalen Ausgänge

IN1\_Start();

IN2\_Start();

}

## 7.2 readmee.txt

Controler: CY8C27446-24PXI

Anutor: Benjamin Lesky

I/O-Konfiguration:

LCD -> Port 2

IN1 ->1.5

IN2->1.3

PWM 1-> Pin 1.4

Beschleunigung-> 0.7

Entfernung -> 0.4

Sollwert-> 0.5

Bugs:

-A/D Wandler geben merkwürdige

werte aus

-Drehrichtung Funktioniert nicht (Fixt 15.09.2013)

-P 0.1 Defekt?

-H-Bruecke braucht zu viel spannung (Fixd 05.09.2013)

-Ausgang 1.3 gibt keine Spannung aus (Fixd 16.09.2013)

-Ausgang 1.4 gibt keine Spannung aus (Fixd 05.09.2013)

-Ausgang 1.5 gibt keine Spannung aus (Fixd 16.09.2013)

-Merkwuerdige ausgabe von Zahlen am LCD (Fixd 16.09.2013)

-Eingaenge werden nicht eingelesen (Fixd 16.09.2013)

bzw gewandelte ergebnisse nicht am lcd ausgegeben

ToDO:

-Initialisierungen auslagern

-eigene präfixe in Komentar einfügen (Erledigt)

-Testroutine programmieren (Erledigt 28.08.13)

-Korekturfaktor Einfuegen (nicht mehr nötig 25.09.2013)

-Arraygroee anpassen (Erledigt 29.08.13)

-Selbsttest (nicht mehr nötig 25.09.2013)

-Pause bei Drehrichtungsumkehr (Erledigt 28.09.13)

-Eingaenge "verdraten" (Erledigt 14.08.13)

-Algorithmus Programmieren (Erledigt 28.07.13)

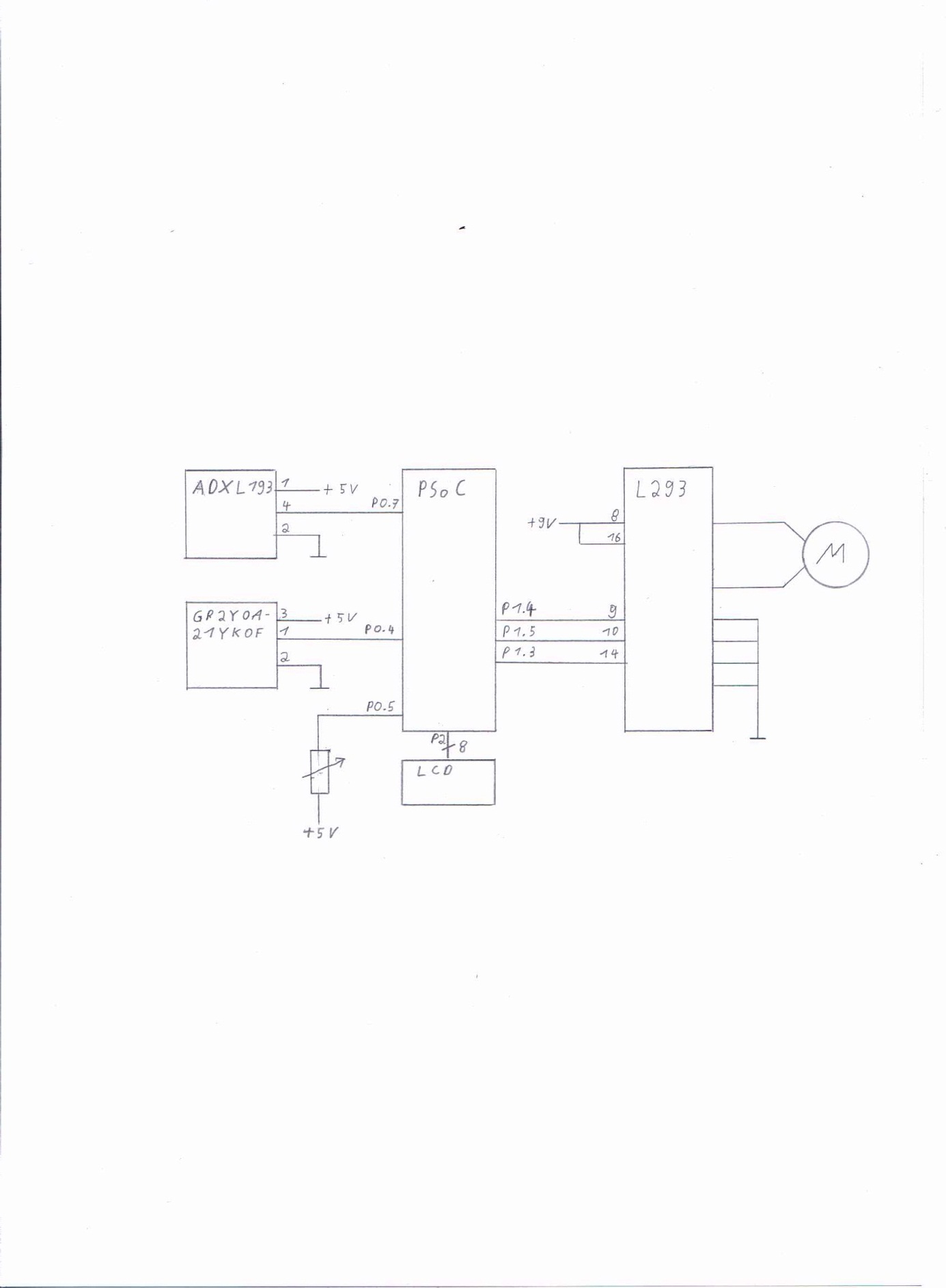
-Sollwert Programmieren (Erledigt 22.07.13)

-PwM Programmieren (Erledigt 24.07.13)

-Verstaerker Programmieren (Erledigt 15.08.13)

Copyright (c) 2013 Benjamin Lesky

## 7.3 Schaltplan



Da der PSoC Teil des Evaluation Kit ist, ist am Controller nur die Pinbelegung der Eingänge und Ausgänge eingezeichnet.

## 7.4 Testprotokoll

Prüfprotokoll A/D Wandler und Motoransteuerung

Motoransteuerung

In der Testschleife wurden der Funktion Ausgangansteuern Direkt die Testparameter übergeben.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Funktionsparameter | | IST | | | SOLL | | |
| hichAusgangswert | hichRichtung | IN1 | IN2 | Pulsweite | IN1 | IN2 | Pulsweite |
| 0 | 0 | 0 V | 4,9 V | 0 % | egal | egal | 0 % |
| 125 | 0 | 4,9 V | 4,9 V | 0 % | egal | egal | 0 % |
| 257 | 0 | 4,9 V | 4,9 V | 0 % | egal | egal | 0 % |
| 0 | 1 | 4,9 V | 0 V | 0 % | 5 V | 0 V | 0 % |
| 125 | 1 | 4,9 V | 0 V | 50 % | 5 V | 0 V | 50 % |
| 255 | 1 | 4,9 V | 0 V | 100 % | 5 V | 0 V | 100 % |
| 0 | 2 | 0 V | 4,9 V | 0 % | 0 V | 5 V | 0 % |
| 127 | 2 | 0 V | 4,9 V | 50 % | 0 V | 5 V | 50 % |
| 255 | 2 | 0 V | 4,9 V | 100 % | 0 V | 5 V | 100 % |

A/D Wandler

Die Ausgegebenen Werte wurden dierekt auf dem LCD-Display angezeigt

|  |  |  |  |
| --- | --- | --- | --- |
| U in V | Beschleunigung | Entfernung | Sollwert |
| 0 | 130 | 130 | 90 |
| 0,9 | 177 | 177 | 239 |
| 1,9 | 224 | 223 | 214 |
| 2,9 | 144 | 143 | 233 |
| 3,9 | 614 | 613 | 329 |
| 5 | 103 | 102 | 198 |

## 7.5 Kopien der Anmeldung