Inhaltsverzeichnis

[1 Aufgabenstellung 2](#_Toc368305129)

[1.1 Problemstellung 2](#_Toc368305130)

[1.2 Aktuelle Situation 2](#_Toc368305131)

[1.3 Ziel der Bachelorarbeit 2](#_Toc368305132)

[2 Das Modell 3](#_Toc368305133)

[2.1 Mechanische Komponenten 3](#_Toc368305134)

[2.2 Elektronische Komponenten 3](#_Toc368305135)

[3 Entwicklung des Algorithmus 5](#_Toc368305136)

[3.1 Kompensationsansatz 5](#_Toc368305137)

[3.2 Entwicklung mit Blockschaltbildern 6](#_Toc368305138)

[3.3 Entwicklung der Regleralgorithmus 8](#_Toc368305139)

[3.4 Bestimmen der Parameter und Simulation in BORIS 9](#_Toc368305140)

[4 Programmierung 10](#_Toc368305141)

[4.1 Main Function 10](#_Toc368305142)

[4.2 Funktion: Dateneinlesen 11](#_Toc368305143)

[4.3 Funktion: LCDansteuern 11](#_Toc368305144)

[4.4 Funktion: Ausgangansteuern 11](#_Toc368305145)

[4.4 Funktion: Initialisierungen 12](#_Toc368305146)

[4.5 Funktion: Test 12](#_Toc368305147)

[5 Zusammenfassung 14](#_Toc368305148)

[5.1 Stand am Ende des Bearbeitungszeitraumes 14](#_Toc368305149)

[5.2 Verbesserungsmöglichkeiten 14](#_Toc368305150)

[5.3 Schlussfolgerung 14](#_Toc368305151)

[6 Literaturverzeichnis 15](#_Toc368305152)

[7 Anhänge 16](#_Toc368305153)

[7.1 Programmcode 16](#_Toc368305154)

[7.2 readmee.txt 21](#_Toc368305155)

[7.3 Schaltplan 22](#_Toc368305156)

[7.4 Klemmenplahn 23](#_Toc368305157)

[7.5 Testprotokoll 24](#_Toc368305158)

# 1 Aufgabenstellung

## 1.1 Problemstellung

Bei der Installation von Offshorewindanlagen ist es wichtig, dass eine an einem Kran hängende Last konstant 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 Seegang zu kompensieren:

- Eigenantrieb

- Rumpfform

- Stabilisatoren

- Position des Krans auf dem Schiff möglichst weit mittschiffs

- regelungstechnische Kompensation

Allen 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 ?????? verwendet. Hier ist mir über die regelungstechnische Umsetzung nichts Bekannt.

Des weiteren gibt es einen Ansatz mittels Prognosealgorithmen, siehe: Sebastian Küchler

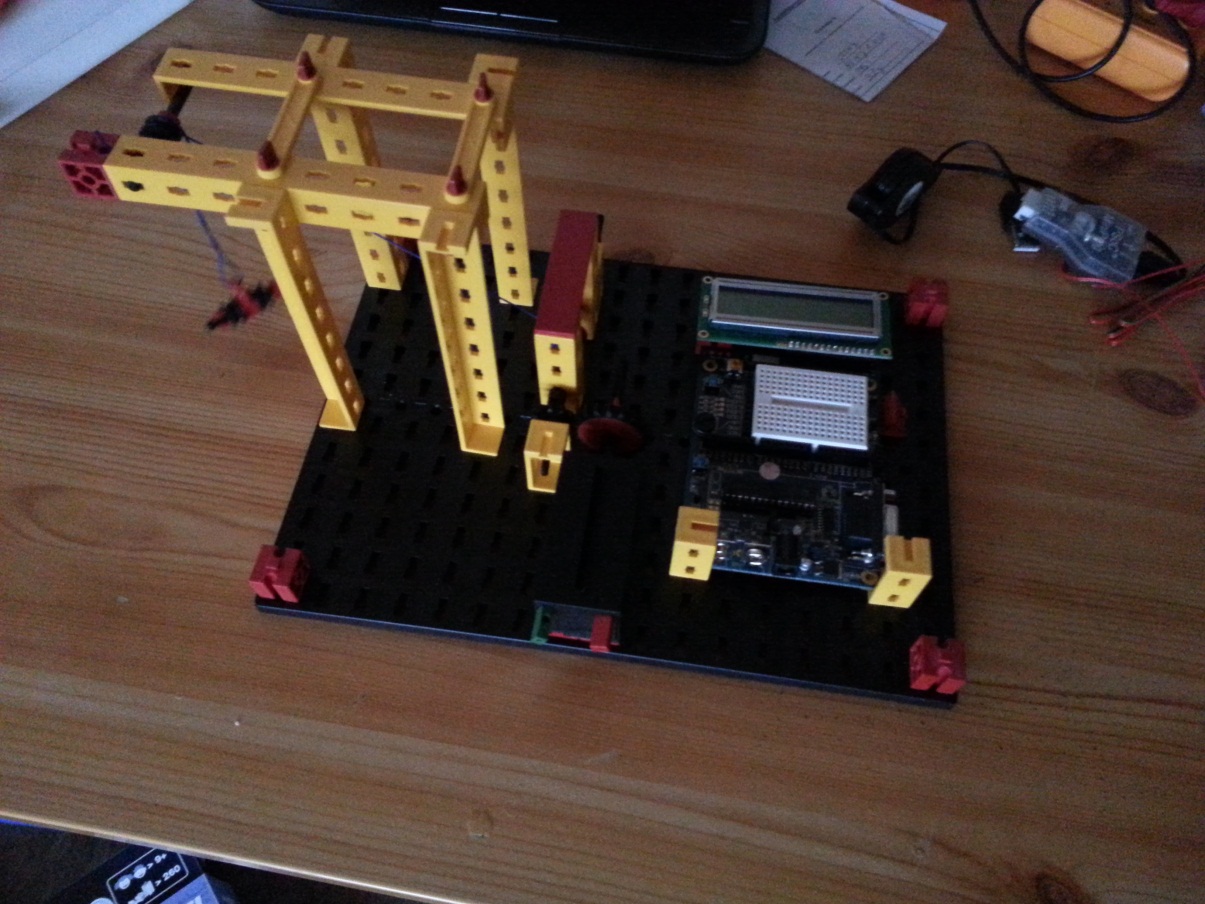
Aktive Seegangskompensation mit beobachtergestützter Prognose der vertikalen Schiffsbewegung

(Ausfürlicher Besser)

## 1.3 Ziel der Bachelorarbeit

Da im Rahmen dieser Bachelorarbeit nur eine reglungstechnische Möglichkeit zur Seegangskompensation gefunden werden soll, wird davon ausgegangen, dass alle durch andere Kompensationsmetoden das Schiff auf einer Position gehalten wird und sich nur noch in der Horizontahlen bewegt. Ziel der Bachelorarbeit ist es also, die durch Seegang verursachte Höhenänderung zu kompensieren. Dies soll ohne pneumatiche Komponenten und ohne komplizierte Prognosealgorithmen geschehen. (Formulierung).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 Modell

Da der Fachhochschule kein Schiff mit einem Kran zur Verfügung steht musste ein Modell zu simulationszwecken gebaut werden.

## 2.1 Mechanische Komponenten

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

## 2.2 Elektronische Komponenten

Zur Regelung des Systems wird ein PSoC1 Evaluation Kit mit einem Prozessor vom Typ CY8C27446-24PXI verwendet. Auf dem mitgelieferten LCH-Display wird die Entfernung der Last zum Entfernungssensor angezeigt. Das System wird durch das Bord mit einer Versorgungsspannung von 5 Volt versorgt.

Zur Ansteuerung des Motors wird ein Treiberbaustein vom Typ L293 D verwendet dieser enthält 2 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 9 Volt daher muss zusätzlich zu der 5 Volt Spannungsversorgung das PSoCs eine 9V Spannungsquelle am Motortreiber angeschlossen werden.

Infrarot und Beschleunigungssensor geben beide eine analoge Spannung aus. (Mehr Schreiben)

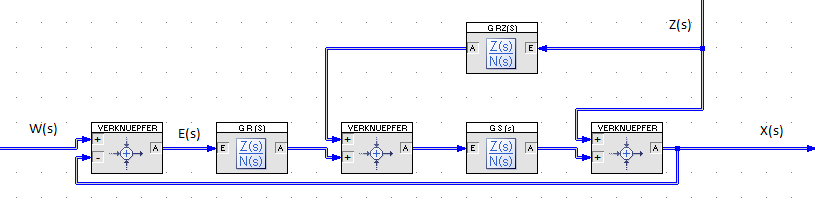
Der Schaltplan ist in Anhang ????? beigefügt.

-Positionierung

-Klemmleisten

# 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 Höhenäderung kann über den Beschleunigungsmesser direkt gemessen werden. Ein Vorteil der Störgrößenaufschaltung ist, dass sie die Stabielitätseigenschaften des Regelkreises nicht verändert (Tröster, 2005). Die Störgröße wird am Ende der Strecke zu deren Ausgang addiert. Dadurch ergibt sich der im folgenden Diagramm dargestellte Aufbau: 

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

Die Auswirkung der Zeitverschiebung muss nun untersucht werden. Zur einfachen Modellbildung wird erst einmal davon ausgegangen, 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), so erhält man:

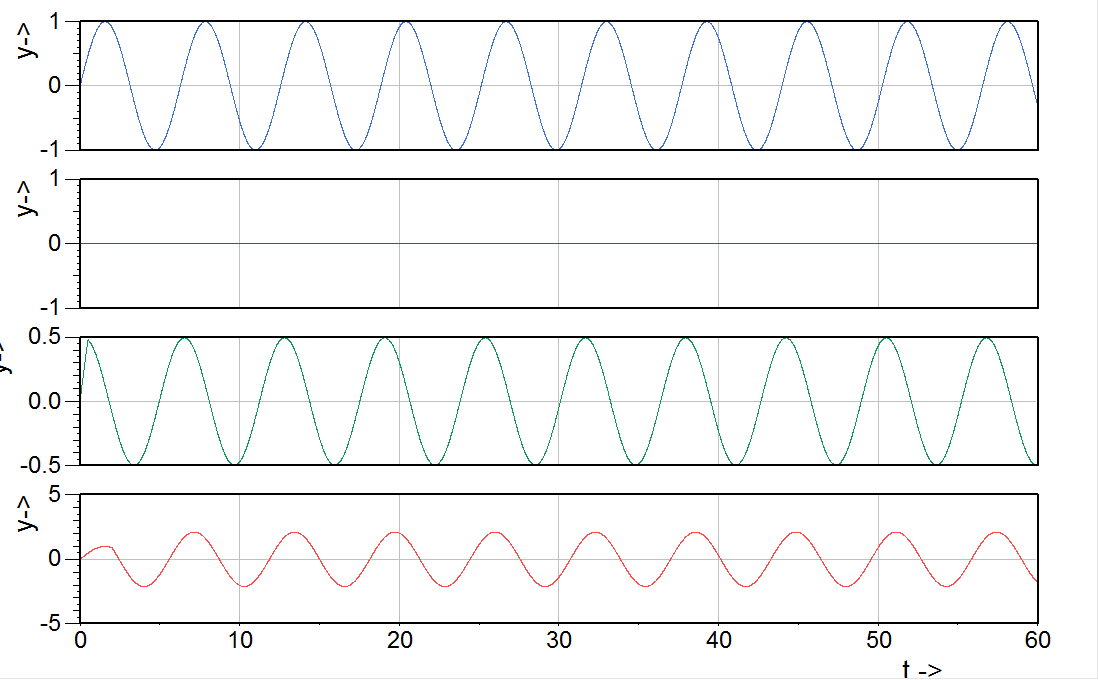
Im besten Fall ist die Zeitverschiebung T = 0und man erhält:

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 sinkt die Regeldifferenz wieder und beträgt am Punkt T = 2π wider null. Um eine Verstärkung der Störgröße zu vermeiden, sollte die Verzögerungszeit möglichst klein gehalten werden. Da die Periodendauer bei Wellen mit Werten von mehreren Sekunden im Vergleich zu der Zeit von wenigen Millisekunden die der Microcontroller und der Sensor zum Erfassen der Daten benötigen, relative groß ist, kann dieses Phänomen im Falle einer Segangskompensation allerdings vernachlässigt werden.



-Mathematichen Nachweis für addierung

-WOK Verfaren

-Nyquist

## 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 Microkontroller schnell genug arbeitet, um ihn wie ein Analoges gerät zu behandeln. Zunächst wird die Übertragungsfunktion der Strecken GS(s) genauer betrachtet. Sie besteht aus einem P-Glied, welches den Motor und die Zahnräder darstellt und ein I-Glied für die Seilwinde.

Der Regler soll ein möglichst gutes Führungsverhalten aufweisen soll wird ein reiner P-Regler gewählt (Hückelheim, Tabelle Reglerauswahl).(Besser Erklären)

-

Die Übertragungsfunktion GRZ(s) setzt sich aus der Übertragungsfunktion des Beschleunigungssensors, zwei I-Gliedern und der Umkehrfunktion der Strecke GS(s)-1 zusammen.

Es gilt die in der Mechanik folgende 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 Glieder werden benötigt, um aus der Beschleunigung wieder die zurückgelegte Distanz zu berechnen.

Nun können die Beiden I-Glider mit dem D-Glied und dem P-Glied Zusammengefast werden. Und man Erhält ein I-Glied mit Folgenden Parametern:

I und D Glider ohne verstärkung Beschleunigungssensor nicht berücksichtigt

Mit



## 3.3 Entwicklung der Regleralgorithmus

Aus dem Blockschaltbild lässt sich folgende Gleichung bestimmen:

beziehungsweise

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

Der Microcontroller arbeitet mit zeitdiskreten Signalen und die Gleichung muss entsprechend angepasst werden. Das Integral kann nun durch eine Summe ersetzt werden.

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

De diese Gleichung relativ komplex ist kann man stattdessen auch die Gerade mit der kleinstmöglichen Abweichung von der Funktion ermitteln. Man erhält dann folgende, einfacher zu berechnende Gleichung: (Formulierung erklären warum besser)

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

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

Anschließend wird die Ausgangs Größe bestimmt.

-Parameter Bestimmen:

-Dazu Drezahl = F(Pulsweite) bestimmen

## 3.4 Bestimmen der Parameter und Simulation in BORIS

- Zahnrad Ausmessen

- Windendurchmesser

- Sprungantwort Motor

- PWM

- Aus Streckenparametern Parameter für GRZ(S)

- Kr bestimmen Sprungantwort da Anlage nicht 100% Funktioniert nur in BORIS bzw Sprungantwort berechnen

- GGF nachoptimieren der Parameter durch Planmäßiges Probieren

# 4 Programmierung

Zu begin der Programmierung wurde das ChipLevelDesing des PSoC und die Ein und Ausgänge Konfiguriert. Danach konnte mit dem schreiben des C-Programmes begonnen werden.

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

-Bugtracking in Readme

Aufgrund der geringen Größe des Projektes erfolgte das Bugtracking erfolgte direkt in der Readmee.txt des Projektes. 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.

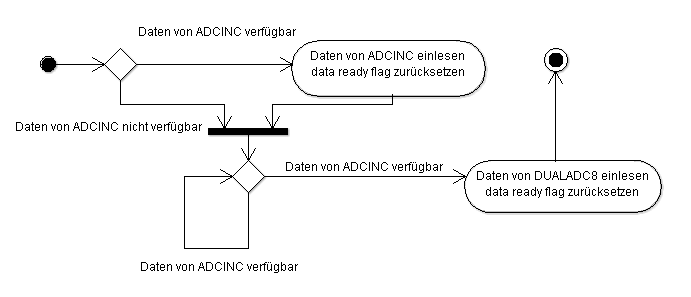
## 4.1 Main Function



Zu Beginn der Main Function werden sämtliche im Chipleveldeseing festgelegten User Modules initialisiert. Anschließend werden in einer Endlosschleife die Daten eingelesen, verarbeitet und ausgegeben.

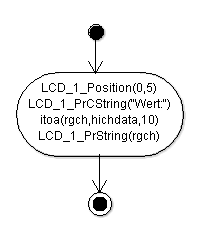
Zu Testzwecken wurde in der Main Function eine weitere Endlosschleife programmiert. Diese wird statt der oben genannten Schleife kompiliert, wenn der Präprozessor TEST definiert ist. In ihr wurden sämtliche Teste Programmiert

## 4.2 Funktion: Dateneinlesen

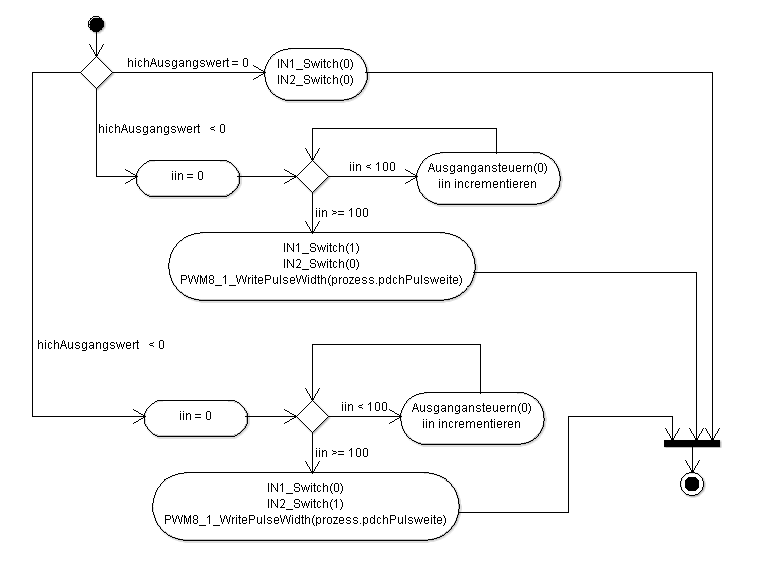


Das einlesen der Daten erfolgt in zwei Schritten. Zuerst wird überprüft ob die Daten des für den Sollwert zuständigen A/D-Wandlers verfügbar sind. Sind diese verfügbar werden sie eingelesen. Anschließend wird das data ready flag des A/D-Wandlers zurückgesetzt. Da der Sollwert nicht zeitkritisch ist,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

## 4.3 Funktion: LCDansteuern

Diese Funktion gibt eine Zahl vom Datentyp Charakter zusammen mit einem dem Text „Wert:“ auf dem LCD-Display aus. Zuerst wird der Text auf das Display geschrieben. 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. (Ausfürlicher)

## 4.4 Funktion: Ausgangansteuern



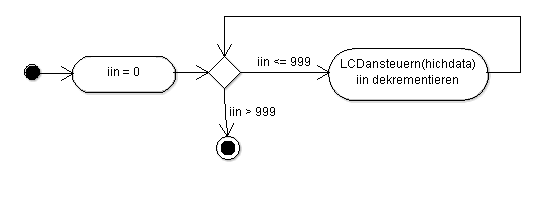
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| hichRichtung | IN1 | IN2 | Enable | Motor |
| 1 | 1 | 0 | PWM | Rechtslauf |
| 2 | 0 | 1 | PWM | Linkslauf |
| sonst | egal | egal | 0 | Leerlauf |

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 dir Funktion ein Wert von eins oder zwei übergeben dreht sich der Motor in die Entsprechende Richtung(siehe Tabelle rechts unten. Bei jedem anderen Wert wird der Enable Eingang der Motortreibers auf null gesetzt und der Motor läuft im Leerlauf. (Neues Diagramm und mehr Beschreibung)

## 4.4 Funktion: Initialisierungen

Um den Code Übersichtlicher zu gestalten sind sämtliche Initialisierungen in eine Funktion Ausgelagert, diese wird zu Beginn der Mainfunktion einmal Aufgerufen.

## 4.5 Funktion: Test



Die Funktion „Test“ wird nur dann kompiliert, wenn der Präprozessor TEST definiert ist. Sie gibt die ihr übergebenen Daten mit Hilfe der Funktion „LCDansteuern()“ direkt auf dem Bildschirm aus. Damit die Daten lange genug auf dem Display erscheinen, um sie lesen zu können, wird die Funktion „LCDansteuern()“ in einer Schleife 999 mal ausgeführt bevor die Funktion verlassen wird

# 5 Zusammenfassung

## 5.1 Stand am Ende des Bearbeitungszeitraumes

Leider konnte das die Seegangs Kompensation bis zum Ende des Bearbeitungszeitraumes nicht realisiert werden. Dies hat zwei Gründe. Zum einen wurde während der Tests kurz vor Abgabetermin festgestellt das der Beschleunigungssensor defekt ist ein Defekt auf und es konnte aufgrund von zu langen Lieferzeiten rechtzeitig kein Ersatz beschafft werden. Des Weiteren sind die Ergebnisse, die die A/D-Wandler dem Programm übergeben nicht nachvollziehbar (Siehe Testprotokoll Anhang7.5). Hierfür konnte die Ursache nicht gefunden werden.

Das ansteuern des Motors funktioniert allerdings Fehlerfrei (Formulierung).

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

Nach Absprache mit Herrn Stief wurde das Programm nun so geändert, das der Motor nun mit zwei Tastern Angesteuert wird. (Besser Ausfürlicher) Um den bereiz Programmierten Algorithmus nicht zu beeinträchtigen wurde die Taster abfrage in der Testschleife Programmiert.

## 5.2 Verbesserungsmöglichkeiten

Die wichtigste Verbesserung ist es die oben genannten Fehler zu Korrigieren. Des Weiteren kann die Regelung noch an folgenden Stellen Optimiert werden:

- Verwenden schnellerer Sensoren und eines Schnelleren Prozessors.

- Ersetzen der Analogen Sensoren durch solche mit einer Digitalen Datenübertragung zum Microcontroller, da die A/D-Wandler im direkten Vergleich relativ Langsam sind.

Beider änderungen sorgen für Dafür das die im Kapitel 3.1 Genannte Verzögerungszeit kleiner wird und damit der Seegamg besser kompensiert werden kann. (Vormulierung)

## 5.3 Schlussfolgerung

Die Aufgabenstellung wurde im Rahmen dieser Beachlerarbeit nicht vollständig erfüllt. Zwar ist die Entwicklung der Seegangskompensation, und damit der Hauptteil der Bechlorarbeit abgeschlossen.

Aufgrund des defekten Beschleunigungssensors und der Fehler bei den A/D-Wandlern konnte der Algorythmuss allerdings nicht im Modell getestet werden. Die Simulation im Computer Zeigt allerdings, dass der Algorithmus prinzipiell funktioniert. Auch wurde die Hardwarmäßige umsetzung komplett geplant und in großen Teilen auch umgesetzt.

Aus den Simulationsergebnissen lässt sich schließen, dass eine Seegangskompensation auch ohne komplexe Vorhersagealgorythmen möglich ist. (Mehr)

# 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);

// Präprozessor: kompiliere Funktion nur wenn Test

#ifdef TEST

void test(char);

#endif

// globale Structur zur Übergabe der Prozessdaten:

struct

{

char pdchBechleunigung, pdchEntfernung;

char pdchSollwert;

} prozess;

void main(void)

{

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

\* Deffinitionen und Deklarationen \*

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

// Konstanten

char kochKP;

char kochKS;

// Variablen

char hichAusgangswert;

char hichBeschleunigungssumme;

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

\* Initialisierung des Controllers \*

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

//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();

SELBSTTEST\_Start();

// 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 ) \* kochKP

- 1 / kochKS \* 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

// Selbsttest des Beshleunigungssensoers anschalten

SELBSTTEST\_Switch(1);

// whileschleife zu testzwecken

// Konstante zur verweildauer in der schleife

while (1)

{

Dateneinlesen();

Ausgangansteuern(0, 2);

// Testfunktionen Aufrufen:

// test(prozess.pdchBechleunigung);

// test(prozess.pdchEntfernung);

// test(prozess.pdchSollwert);

// 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(4);}

else if (prozess.pdchBechleunigung <= 110)

{LCDansteuern(5);}

else {

LCDansteuern(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)

{

// Variablendeklaration

int iin;

// 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);

}

// Präprozessor: kompiliere Funktion nur wenn Test

#ifdef TEST

// Ausgabe der Testdaten auf LCD-Display

void test(char hichdata)

{

// gibt 99999 mal LCD Aus

int iin;

for (iin = 0; iin <= 500; iin ++)

{

LCDansteuern(hichdata);

}

}

#endif

## 7.2 readmee.txt

Controler: CY8C27446-24PXI

Anutor: Benjamin Lesky

I/O-Konfiguration:

LCD -> Port 2

Selbsttest ->1.1

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

## 7.4 Klemmenplahn

## 7.5 Testprotokoll

Prüfprotokoll A/D Wandler und Motoransteuerung

Motoransteuerung

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |