Demonstrator: Simulation und Realisierung eines elektrischen Getriebes - Technikgruppe

Christian Benjamin Ries
Christian_Benjamin.Ries@fh-bielefeld.de

9. März 2010

Bis zum Ende des Wintersemesters 2009/2010 soll im Rahmen einer Projektveranstaltung (1) im Masterstudiengang **„Optimierung und Simulation“** ein Demonstrator zur **„Optimierung und Realisierung eines elektrischen Getriebes“** berechnet und realisiert werden. Zugrunde liegt dabei ein bereits vorhandener Versuchsaufbau, welcher um eine geeignete Horizontalführung – beispielsweise in Form einer Blattfeder – zu ergänzen ist. Aufgabenstellungen, wie die Synchronisation der zwei vorhandenen Elektromotoren über regelungstechnische Komponenten, sowie die Untersuchung des gesamten Systemverhaltens, sowohl messtechnisch als auch durch Simulation mittels geeigneter Software, sind neben der tatsächlichen Realisierung von besonderer Wichtigkeit.

Inhaltsverzeichnis

1 Einleitung

Die Technikgruppe ist für die Ist-Analyse (1, AP-Nr. 1.1), die Synchronisation der Motoren (1, AP-Nr. 1.2) und Definition der Aufnahmepunkte für die Horizontalführung (1, AP-Nr. 1.3.1) zuständig. Diese Ausarbeitung behandelt das Aufgabengebiet zur Realisierung des mechanischen Synchronlaufs mit einer grafischen Steuerungssoftware (LabVIEW 2009). Die Steuerung wird über eine benutzerfreundliche grafische Oberfläche erfolgen (vgl. (1, AP-Nr. 1.2.4). Die Abbildungen in dieser Arbeit werden mit der *Unified Modeling Language* (UML) in der Version 2.2 beschrieben (2).

1.1 Teilnehmer des Projektes

Mit Verlauf der Zeit sind einige Teilnehmer aus dem Projekt ausgestiegen. Folgende Personen befinden sich weiterhin in der Umsetzungsphase: Alex Reglowski; Anja Webel; Boumi K. Gaetan; Carolin Vollmer; Christian Benjamin Ries; Igor Friesen; Ilja Alkov; Kai-Fabian Henning; Kristof Kleiner; Lilija Nabiewa; Stefanie Stork; Sven Ludewig.

2 Grundlagen

Dieser Abschnitt beschreibt die Komponenten und Zusammenhänge alle Bestandsteile.

2.1 Entwicklungsumgebung

Die Entwicklungsumgebung besteht aus mehreren Hardware- und Softwarekomponenten, die Abhängigkeiten und Zusammenhänge werden in Abbildung 2 veranschaulicht. Tabelle 1 enthält eine Auflistung der Komponenten und beschreibt kurz die Verwendung. Der Abschnitt 2.3ff liefert weitere Details über diese Komponenten.

Identifikation	Name	Typ	Verwendung
1	Inkrementalgeber		Ermittlung der Winkelposition, Drehgeschwindigkeit/-richtung, Beschleunigung
2	Motor		Drehen der Achsen
3	Echtzeit-Kontroller	NI cRIO-9012	Steuerung der Inkrementalgeber und Motoren
4	Digitaler Signaleingang	NI 9423	Signaleingang für die Inkrementalgeber
5	Analoger Signalausgang	NI 9263	Signalausgang für die Motoren

Tabelle 1: Hardwarekomponenten in der Umsetzung

2.2 Inkrementalgeber

Inkrementalgeber (IGR) sind Sensoren die zur Erfassung von Lageänderungen Verwendung finden. Der Aufbau von IGR hängt vom Anwendungsfall ab. Es ist mit IGR möglich, die Drehrichtung einer Komponente zu ermitteln und wie schnell die Drehung in einem Moment ist. Durch die Zwischenspeicherung der Winkelwerte ist weiterhin die Erfassung von Beschleunigungen möglich.

2.2.1 Anwendungsbereiche

IGR werden zur Bestimmung von Positionen, Ermittlung der Geschwindigkeit und Beschleunigung eingesetzt.

2.2.2 Funktionsweise

Der funktionale Aufbau von IGR kann durch unterschiedliche Technologien erreicht werden, folgende technische Prinzipien sind bisher entwicket worden: Photoelektrische Abbildung, Abbildendes Messprinzip, Interferentielles Messprinzip, Magnetische Abtastung, Schleifkontakt.

Abbildendes Messprinzip Bei diesem Messprinzip wird ein Lichtstrahl durch eine Öffnung geschickt. Hinter der Öffnung befindet sich ein lichtempfindlicher Sensor, der ein Signal zur Verarbeitung weitergibt.

Interferentielles Messprinzip Wie bei dem abbildenden Messprinzip wird ein Lichtstrahl durch ein Gitter geschickt. Dieses Gitter sorgt dafür, dass drei Strahlanteile erzeugt werden. Diese haben zueinander einen Versatz um 120° und werden durch eine Folgeelektronik in ein verwertbares Signal umgewandelt.

Magnetische Abtastung Das Signal wird durch Hallelemente[1] erzeugt, die in Aufteilung auf einer Abtastplatte aufgebracht sind.

[1] http://www.elektroniknet.de/index.php?id=2113

Schleifkontakt Die Sensoren arbeiten mechanisch. Ein Scheifer liefert ein Signal wenn die Stellung des Schleifers mit einer Signalleitung in Kontakt kommt.

2.3 Architektur

Die Abbildung 1 zeigt die prinzipiellen Anwendungsfälle (*Use-Cases*) und Aktoren des elektrischen Getriebes. Es sind zwei Aktoren, „Anwender“ und „Automatik“, definiert. Der Aktor „Anwender“ steuert das elektrische Getriebe und kann Einstellungen vornehmen. Offensichtliche *Use-Cases* wurden hiervon ausgenommen, z.B. das Ein-/Ausschalten. Einstellungen betreffen die Drehzahl der Motoren und die Phasenverschiebung der zwei drehenden Achsen zueinander. Der Anwender kann sich den aktuellen Status des Systems anschauen, dies betrifft die Drehgeschwindigkeit und Winkelposition der Achsen relativ zur Winkelposition wenn der Z-Impuls schaltet. Der Anwender kann zur Zeit noch direkt die Motoren ansteuern, dies geschieht über die Regelung der Ausgangsspannung.

Der zweite Aktor ist „Automatik“. Automatik ist eine Regelung, die in die Steuerungseinheit (vgl. Abs. 2.3.1) implementiert ist. Einstellungen die vom Anwender getätigt wurden, werden von der Automatik nachgeregelt.

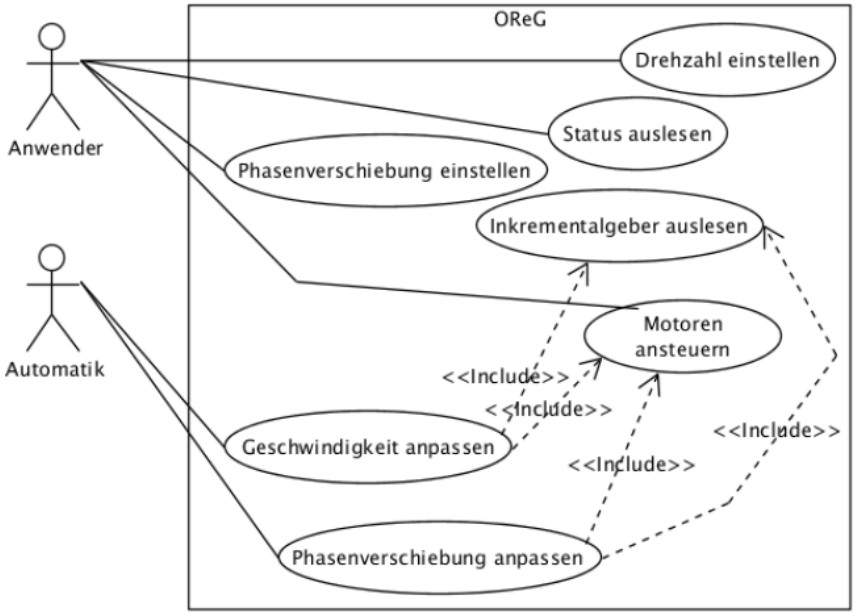

Abbildung 1: Anwendungsfalldiagramm für die Funktionen der Softwarekomponenten

Der *Use-Case* zur Anpassung der Phasenverschiebung (Phasenverschiebung anpassen) beinhaltet die *Use-Cases* „Motoren ansteuern“ und „Inkrementalgeber auslesen“, diese sind für die Berechnung der Phasenverschiebung nötig. Der *Use-Case* „Geschwindigkeit anpassen“ benötigt zur Ausführung die *Use-Cases* „Motoren ansteuern“ und „Inkrementalgeber auslesen“. „Inkrementalgeber auslesen“ wird von keinem der Aktoren direkt aufgerufen.

Die Abbildung 2 veranschaulicht die reale Situation des Demonstrators in einem Verteilungsdiagramm. Das Verteilungsdiagramm beinhaltet alle relevanten Komponenten der Entwicklungsumgebung. Drei Komponentenabstraktionen sind vorhanden: 1. Steuerungssoftware, 2. Echtzeitsteuerung, 3. Rüttelplatte. Die Steuerungssoftware ist mit LabVIEW realisiert und unter einer Windows XP Installation lauffähig. Die Echtzeitsteuerung enthält das Echtzeitsystem aus Abschnitt 2.3.1 mit den Steuerungsmodulen. Die Rüttelplatte besitzt zwei Achsen mit jeweils einem Motor und IGR. Die Umwucht ist ein statisches Element ohne Ansteuerungsmöglichkeiten. Die Steuerungssoftware ist mit der Echtzeitsteuerung durch eine Assoziation verbunden, die eine Netzwerkverbindung beschreibt. Dabei kann die Echtzeitsteuerung durch mehrere Steuerungen gesteuert werden. Die Analogausgangsmodule der Echtzeitsteuerung sind mit den Motoren verbunden, ebenso sind die Digitaleingangsmodule mit den IGRn verbunden.

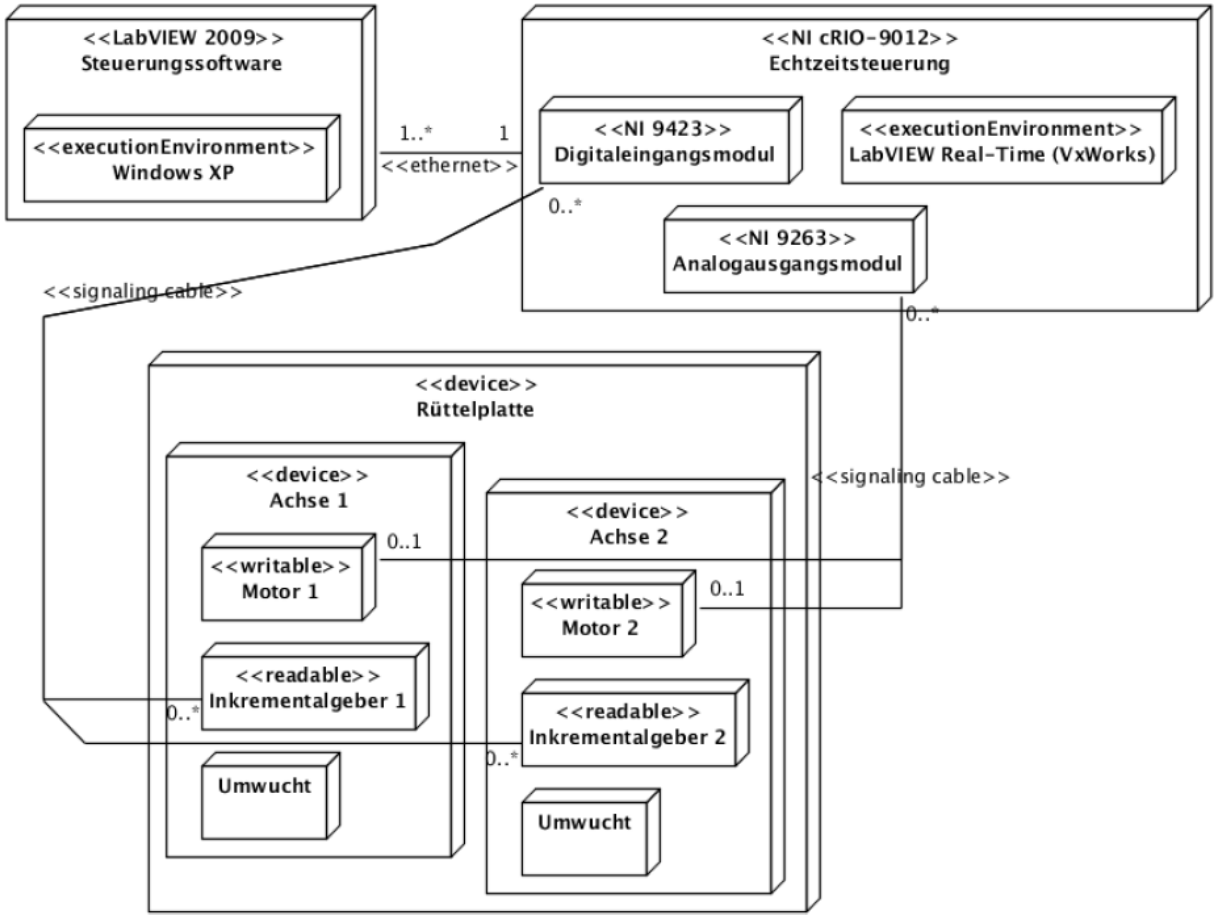

Abbildung 2: Aufbau der Hardwarearchitektur zur Steuerung/Messung der Motoren/Inkrementalgeber

2.3.1 NI cRIO-9012, NI 9423, NI 9263

Die Steuerung aller Hardwarekomponenten geschieht mit dem Echtzeitsystem (Real-Time) NI cRIO-9012[2] von National Instruments (NI). Für die Ansteuerung der Motoren wird das Modul NI 9263[3] verwendet, dieses dient zur Ausgabe einer Spannung zwischen ± 10 Volt. Das Auslesen der aktuellen Werte der IGR wird mit dem Modul NI 9423[4] durchgeführt. Das Modul NI 9423 besitzt acht digitale Eingänge, die jeweils eine Schaltzeit von $1\mu s$ besitzen.

Tabelle 2 beinhaltet die Anschlusskonfiguration der Architektur aus Abbildung 2. Die Kurzbezeichung DIx steht für *Digital Input* (Digitaleingang), das x für die jeweilige Belegungsnummer des Moduls. Ähnliches gilt für die Kurzbezeichnung AOx, dies steht für *Analog Output* (Analogausgang).

2.3.2 LabVIEW 2009

Für die Entwicklung aller Softwarekomponenten zur Ansteuerung der Motoren, zum Auslesen der IGR und Umsetzung der funktionalen Steuerelemente wird die Software LabVIEW 2009[5] vom Hersteller National Instruments eingesetzt.

3 Umsetzung

Die *Deadline* der Projektumsetzung ist auf den 26. Februar 2010 gelegt. Bis zum diesen Tag wurde von der Gruppe zur Umsetzung einer Regelungstechnik keine Regelung geliefert, so dass ein anderer Ansatz gewählt wurde. Die Aussteuerung der Drehachsen zueinander wurde iterativ durchgeführt. Das bedeutet,

[2] http://sine.ni.com/nips/cds/view/p/lang/en/nid/203347
[3] http://sine.ni.com/nips/cds/view/p/lang/en/nid/14170
[4] http://sine.ni.com/nips/cds/view/p/lang/en/nid/14173
[5] http://www.ni.com/labview

Bezeichnung	Beschreibung
Schnittstelle	NI 9423
Funktion	Empfängt die Signale der Inkrementalgeber
Konfiguration	DIx bezeichnet die Eingänge von NI 9423 Ax, Bx, Zx bezeichnet die Ausgänge der Inkrementalgeber Anschluss des 1. Inkrementalgeber: $\Rightarrow DIO = A1, DI1 = B1, DI2 = Z1$ Anschluss des 2. Inkrementalgeber: $\Rightarrow DI3 = A2, DI4 = B2, DI5 = Z2$
Schnittstelle	NI 9263
Funktion	Ausgangsspannung für die Motoren setzen.
Konfiguration	$AO0, AO1$ bezeichnet die Ausgänge von NI 9263 $AO0$ ist an Motor 1 angeschlossen $AO1$ ist an Motor 2 angeschlossen

Tabelle 2: Konfiguration der cRIO Schnittstellen

dass eine generell sehr rudimentäre Nachregelung zum Zuge kommt. Beide Drehachsen des elektrischen Getriebes besitzen jeweils einen Inkrementalgeber, welcher die Position der Drehung ermittelt. Nach einer ganzen Umdrehung geben die Inkrementalgeber jeweils einen weiteren Impuls (Z-Impuls) aus der diese komplette Umdrehung signalisiert.

Das Verhältnis dieser jeweiligen Werte, die Differenz zwischen den zwei Z-Impulsen für beide Drehachsen, liefert einen Wert für deren Versatz zueinander. Die Idee ist nun, dass dieser Wert kontinuierlich überprüft wird und entsprechend eine Nachregelung stattfindet. Bei einem Delta-Wert von 0 befinden sich die Drehachsen zueinander in einem synchronen Verhältnis, sprich beide Achsen drehen sich ohne Versatz zueinander. In Abbildung 4 wird dieses Prinzip als Aktivitätendiagramm verdeutlicht und veranschaulicht die bisherige Implementierung der Steuerungsanwendung. Der Abschnitt 3.3ff beinhaltet eine genauere Sichtweise dieser Umsetzung und beschreibt welche Problematiken damit verbunden sind.

Das Prinzip der Umsetzung ist weiterhin, dass eine Drehachse als *Master* und die zweite Achse als *Slave* fungiert. Dies soll heißen, dass nur eine Drehachse nachgeregelt wird und zwar im Verhältnis zu der zweiten Achse. Dieses Prinzip wird in zahlreichen technischen Anwendungen angewandt, u.a. in der *Bluetooth* Technologie[6]

3.1 Anforderungen

Zur Realisierung des mechanischen Synchronlaufs soll eine grafische Steuerungssoftware mit LabVIEW auf der Basis der zuvor optimierten Regelungskonzepte entwickelt werden (dies ist nicht möglich, vgl. Abs. 3). Die Steuerung soll über eine benutzerfreundliche Schnittstelle verfügen.

3.2 Programmablauf

Für die Steuerung der Motoren und Verarbeitung der Daten von den Inkrementalgebern werden mehrere Anwendungen implementiert. Die Architektur aus Abbildung 2 liefert die Vorgaben für die verschiedenen Zielplattformen um die Entwicklung der Anwendungen durchzuführen. Die Steuerung benötigt zwei Anwendungskomponenten, eine wird auf dem NI cRIO-9012 (Target) installiert und die zweite auf dem Computer zur Ansteuerung (Host) des Target.

In Abbildung 3 ist der Programmablauf des Target abgebildet. Zur Verdeutlichung sind drei Bereiche erstellt, die jeweils eine Aufgabe auf dem Targetsystem beschreiben. Der Bereich „Readout“ liest die Werte an den Eingängen (DI0-DI5) des Target ein, die mit den Ausgängen der Inkrementalgeber verbunden sind und berechnet den aktuellen Punkt in der Drehung um die Achse und hat die Einheit $\frac{Counts}{Interval}$. Im zweiten Bereich „Velocity“ werden die eingelesen Werte aus „Readout“ zur Berechnung der Geschwindigkeit (engl.

[6] Ein Standard zur Funkübertragung zwischen Geräten über kurze Distanz.

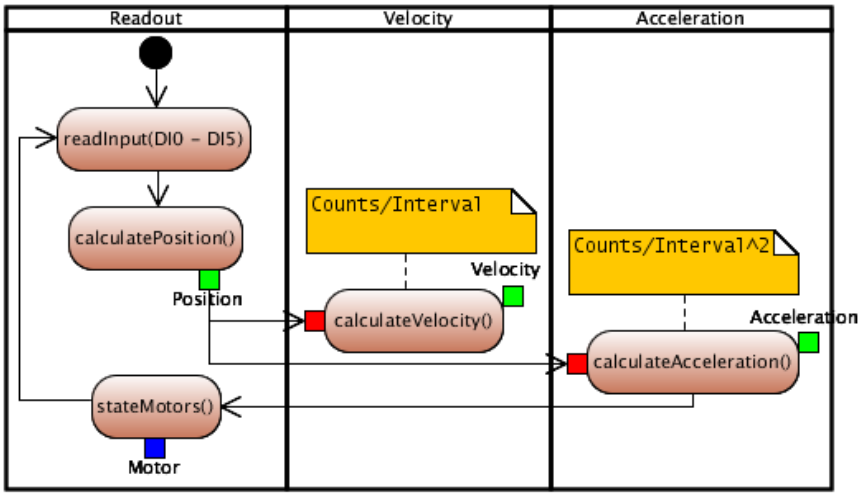

Abbildung 3: Programmablauf auf dem Echtzeitsystem NI cRIO-9012

velocity) verwendet. Die dritte Unterteilung „Acceleration“ errechnet die Beschleunigung (engl. acceleration). Die Berechnungen stellen zu jeder Zeit die Ergebnisse für die weitere Nutzung bereit. Diese Werte (die Ports: „Position“, „Velocity“ und „Acceleration“) können vom Host über eine Netzwerkverbindung (vgl. Abb. 2) gelesen werden. Zudem werden die aktuellen Spannungswerte der Motoren zugänglich gemacht und können zudem neu gesetzt werden.

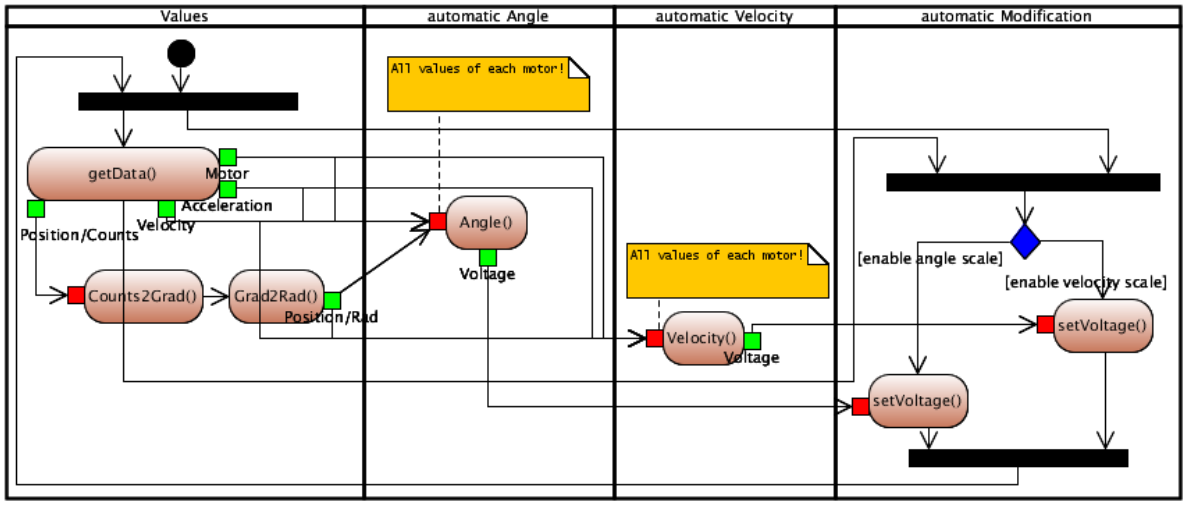

Abbildung 4: Programmablauf auf dem Host

Die Abbildung 4 verdeutlicht den Programmablauf auf dem Host zur Steuerung des elektrischen Getriebes. Es werden die Werte über die Netzwerkverbindung gelesen, symbolisch mit der Funktion *getData()* bezeichnet. Die Werte werden an die Funktionen *Angle()* und *Velocity()* weitergegeben um die Ausrichtung und die Werte zur Korrektur der Ausrichtung zu berechnen. Die Target Anwendung liefert den Positionswert in der Einheit $\frac{Counts}{Interval}$. Die Funktionen *Angle()* und *Velocity()* benötigen diesen Wert in Rad, so dass eine Umrechnung durch die Funktionen *Counts2Grad()* und *Grad2Rad()* erfolgt. *Counts2Grad()* berechnet einen Winkel in Grad und *Grad2Rad()* überführt diesen Wert in Rad. Im Bereich „automatic Modification“ kann der Anwender entscheiden, welche Entscheidungshilfe verwendet werden soll. Die Steuerung kann automatisch nach der Winkelstellung „automatic Angle“ oder bzgl. der Drehgeschwindigkeit „automatic Velocity“ geregelt werden. Die Auswahl wird durch den rechten Teil in der Abbildung 6 ermöglicht.

3.3 Systemverhalten

In Abbildung 5 sind folgende Definitionen verwendet. Die X-Achse veranschaulicht den zeitlichen Verlauf, dieser wird durch LabVIEW automatisch skaliert. Die Y-Achse beschreibt das Delta der Amplituden zwischen den Drehachsen in Grad. Die Beschreibung oben links in der Abbildung „delta(M1, M2) Angle [°] - Signalverlauf“ hat die zuvor genannte Bedeutung, *M1* steht für den ersten Motor, *M2* dementsprechend für den zweiten Motor.

Die Abbildung 5 veranschaulicht den Verlauf der in Abschnitt 3 erwähnten Idee, dass möglichst das Delta der Phasenverschiebung 0 entsprechen soll um einen Synchronlauf der Drehachsen zu erhalten. Die Basis dieses Verlaufs beruht auf dem Programmablauf aus Abbildung 4. Es wird durchgehend der Stand der Drehposition zueinander verglichen und gegebenenfalls nachgesteuert.

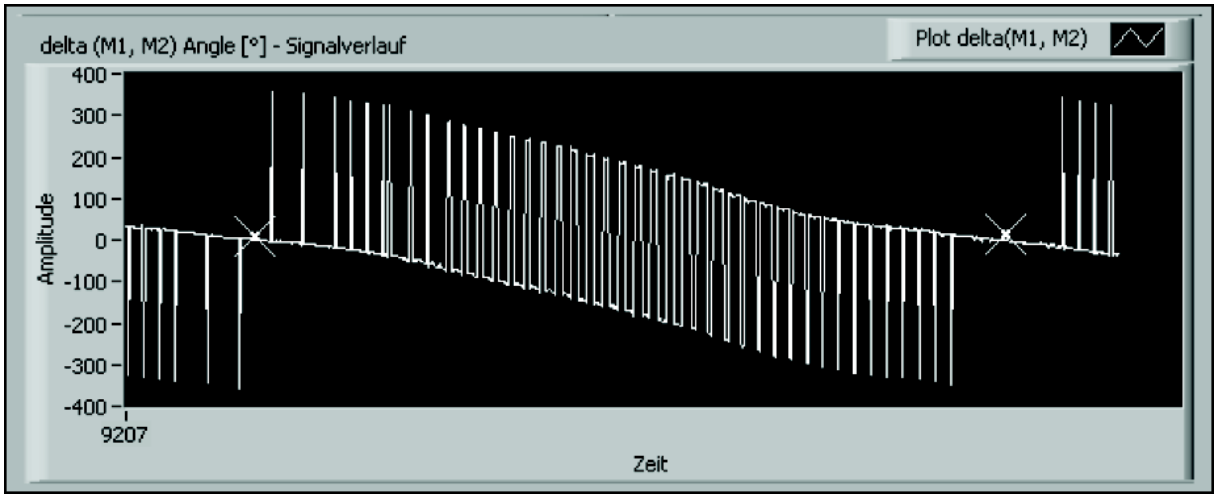

Abbildung 5: Phasenverschiebung der drehenden Achsen

An den Positionen mit dem Kreuz (s. Abb. 5) ist ein Synchronlauf vorhanden, es ist an dem Übergang der Ausschläge zu erkennen, dass die Phasenverschiebung kippt. Eine Kippung der Phasenverschiebung in diesem Zusammenhang ist gleichbedeutend mit einem Übergang von $\Delta Winkel = -10 \Rightarrow \Delta Winkel = 10$, sprich das Vorzeichen hat sich geändert. Dieses Verhalten ist weiterhin auch bei den Werten $\Delta Winkel = 180 \Rightarrow \Delta Winkel = 180$ zu beobachten. In der Mitte des Verlaufs kippen die Amplituden zueinander. Dieser in Abbildung 5 abgebildete Verlauf ist periodisch, d.h. die zweite Achse (M2, Slave) hängt immer hinterher. An dem Punkt, wenn das Δ zwischen den Achsen 0 ist muss eine weitere Regelung einspringen.

Zu diesem Zweck wurden zwei iterative Steuerungen implementiert. Wenn das elektrische Getriebe eingeschaltet wird, kann der Anwender nur den Master regeln. Sobald der Master eine Aussteuerung erhält, z.B. eine Spannung von 3 Volt, wird die Regelung angeworfen. Der Anwender kann entscheiden welche Regelung verwendet werden soll. Zu diesem Zweck wurde die Benutzerschnittstelle in Abbildung 6 so aufgebaut, dass der Anwender im rechten Bereich zwischen zwei Modus wählen kann: 1. *En-/Disable velocity scale*, 2. *En-/Disable angle scale*, es können auch beide angeschaltet werden. Der Schalter unten rechts in der Abbildung 6 lässt es zu, dass zwischen diesen Modus umgeschaltet werden kann. Die Tachoanzeige gibt den aktuellen Δ-Wert der beiden Drehachsen aus.

Das Prinzip ist nun folgendes: Im ersten Schritt stellt der Anwender eine Spannung am Master ein, der Motor läuft an und die Achse dreht sich. Nun wählt der Anwender den Modus zur automatischen Nachregelung durch Geschwindigkeitsanpassung „En-/Disable velocity scale“ aktivieren und den Schalter unten rechts auf „Velocity!“ stellen. Jetzt regelt die Steuerung aus Abbildung 4 die zweite Drehachse nach. Der Slave wird in den Drehzahlbereich des Master gesteuert und hält sich in diesem Bereich auf. Es ist nie ein exakter Synchronlauf möglich, da die Umwucht der jeweiligen Drehachsen kontinuierlich Störungen hervorruft, der Verlauf ist nichtlinear.

Die Regelung nach der Geschwindigkeit ist nicht für die Regelung der Phasenverschiebung zu gebrauchten. Für diesen Zweck ist eine zweite Regelung implementiert, die den Nullphasenwinkel der beiden Drehachsen miteinaner ins Verhältnis setzt. Der Anwender muss in der derzeitigen Implementierung manuell zwischen diesen zwei Regelungskonzepten umschalten. Zuerst werden beide Drehachsen auf eine nahezu gleiche Drehgeschwindigkeit gebracht, daraufhin solange gewartet bis der Δ-Wert zwischen diesen Drehachsen im Bereich von Null ist und dann auf den Regelungsmodus des Winkelangleichs (En-/Disable angle scale) umgeschaltet.

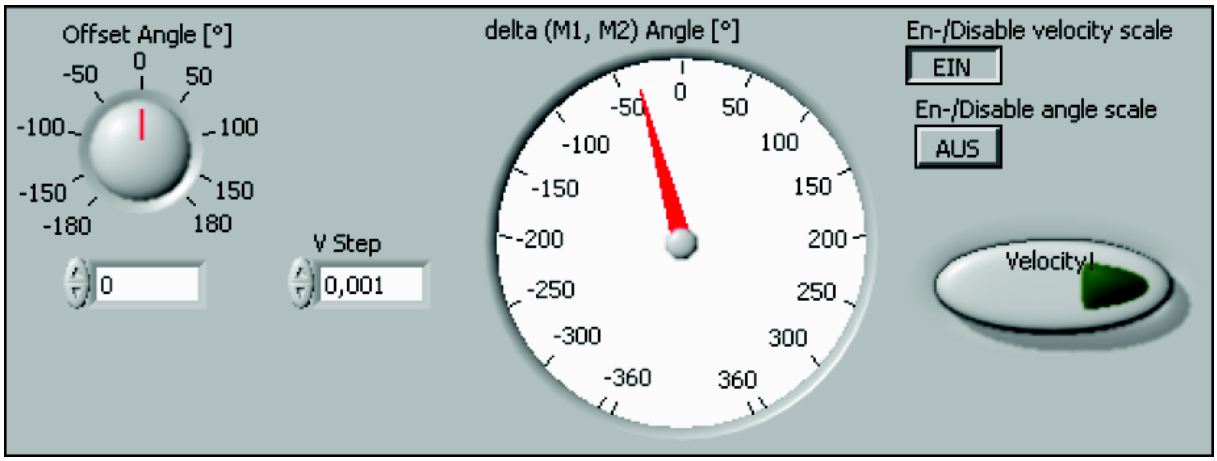

Abbildung 6: Steuereinheit für die Phasenverschiebung der drehenden Achsen

Der linke Bereich in Abbildung 6 ist für die Konfiguration des Nullphasenwinkel gedacht und ermöglicht im Modus der aktuellen Regelung die Möglichkeit zur Einstellung des Δ zwischen den Drehachsen. Das Feld „V Step“ in Abbildung 6 dient der Einstellung für die Toleranz des Δ-Werts zwischen den Drehachsen in beiden Modus.

3.4 Probleme

Es hat sich gezeigt, dass die Toleranz bei dem Vergleich der Δ-Werte zu hoch ist, so dass nie ein Synchronlauf möglich war. Der Slave regelt sich zum Master nach, allerdings bleibt der Slave nie im Bereich des Masters.

4 Fazit

Die Umsetzung einer iterativen Steuerung für den Synchronlauf von zwei umwuchtbehafteten Drehachsen ist nicht ohne eine, im mathematischen Sinne korrekte, Regelung möglich. Es muss bedacht werden, dass das Verhalten dieser Drehachsen ein sehr starkes nichtlineares Verhalten besitzt und eine reine Nachregelung nach dem Master-Slave-Prinzip, mit Geschwindigkeits- und Winkelanpassung, nicht möglich ist. Ein erweiterte Regelung, die dieses Verhalten durch ein Kräfte-Masse-Differentialgleichungssystem beschreibt, ist zu empfehlen.

Literatur

[1] Teilnehmer des Projektes: *Beschreibung der Arbeitspakete anhand des Projektstrukturplanes*, Fachhochschule Bielefeld, WS 2009/2010

[2] OMG Unified Modeling Language, Superstructure, Feb. 2009, url: http://www.omg.org/spec/UML/2.2/Superstructure