Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

Department Maschinenbau und Produktion

Entwicklung eines praktischen Mechatronik-Laborversuches

Peter Quade

20. Februar 2013

Hochschule für Angewandte Wissenschaften Hamburg
Department Maschinenbau und Produktion
Berliner Tor 21
20099 Hamburg

Verfasser: Peter Quade
Abgabedatum: 20.02.2013

1. Prüfer/in: Prof. Dr.-Ing. Stefan Wiesemann
2. Prüfer/in: Prof. Dr.-Ing. Wolfgang Schulz

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AD-Wandler	Analog-Digital-Wandler
CPU	Central Processing Unit
cRIO/CompactRIO	Compact Reconfigurable I/O-Technology
CAD	Computer-aided Design - Rechnerunterstütztes Konstruieren
DMS	Dehnungsmessstreifen
DC	Direct Current - Gleichstrom
FFT	Fast Fourier transform - Schnelle Fourier-Transformation
FPGA	Field Programmable Gate Array - Feld programmierbare (Logik-) Gatter-Anordnung
IR	Infrarot
I/O	Input/Output
IC	Integrated Circuit - Integrierter Schaltkreis
IEC	International Electrotechnical Commission
LabVIEW	Laboratory Virtual Instrumentation Engineering Workbench
LED	Light-emitting Diode
NI	National Instruments
PC	Personal Computer
PID-Regler	Proportional–integral–derivative controller -
PWM	Pulse-width Modulation - Pulsweitenmodulation
SPS	Speicherprogrammierbare Steuerung
TTL	Transistor-Transistor-Logik
VDI	Verein Deutscher Ingenieure
VI	Virtuelles Instrument

1 Einleitung

In der modernen Entwicklung von technischen Anlagen kommt der Verbindung von Mechanik und Elektrotechnik eine immer größere Bedeutung zu. Die interdisziplinäre Zusammenarbeit zwischen Mechanik, Elektrotechnik und Informationsverarbeitung wird kurz als Mechatronik bezeichnet. Der Begriff Mechatronik wurde zuerst von einem japanischen Unternehmen geprägt und 1971 als eingetragener Markenname für Industrieroboter benutzt. Heute wird damit ein ganzer Industriezweig benannt. Welche außergewöhnliche Bedeutung der Mechatronik zugesprochen wird zeigt sich schon an den Entwicklungen auf dem Ausbildungsmarkt. 1991 wurde in Deutschland der erste Lehrstuhl für Mechatronik geschaffen und seit 1998 ist der Ausbildungsberuf zum Mechatroniker anerkannt. Neben den Studiengängen zum Mechatroniker werden auch immer mehr interdisziplinäre Vertiefungsrichtungen für die Mechatronik angeboten. Die Stellenanzeigen für Mechatroniker füllen auf den einschlägigen Internetportalen mehrere Seiten.

In der Industrie geht es heute vornehmlich darum, immer komplexer werdende Produkte in immer komplexeren Produktions- und Fertigungsprozessen zu handhaben. Dabei müssen gleichzeitig die Kosten minimiert werden, um die Konkurrenzfähigkeit der eigenen Produkte zu gewährleisten. Dies kann und wird nur noch von einer intensiven Automatisierung der Prozesse erreicht. Hierfür müssen natürlich spezielle Anlagen entwickelt werden, welche selbst schon mechatronische Systeme sind. Ein Beispiel für die immer weiter steigende Komplexität von Produkten ist die von der Volkswagen AG im Jahr 1974 gestartete Produktreihe des VW Golf. Der Vergleich zwischen Golf I und Golf VII zeigt sehr anschaulich die immer weiter ansteigende Integration von Elektronik in die Mechanik eines Automobils. Dies reicht von der Umstellung von mechanischen auf digitale Kilometerzähler bis hin zu Einparkassistent-Systemen. Nicht nur die kürzeren Lebenszyklen stellen an die Fertigung immer größere Anforderungen, sondern auch der stetig steigende Kostendruck ausgehend von den osteuropäischen und asiatischen Ländern. Hier kann Aufgrund niedriger Lohnkosten sehr günstig produziert werden. Industrieländer, wie Deutschland, können dies nur durch einen hohen Automatisierungsgrad in der Produktion und Fertigung ausgleichen. Es werden mehr Produkte entwickelt, welche auf rechnergestützte Informationsverarbeitung und Automatisierung von Prozessparametern setzen, dass dies nicht immer reibungslos läuft ist aktuell bei SIEMENS mit dem neuen ICE-3 [11] zu sehen. Bei diesem kommt es Softwareseitig,

bei der Verarbeitung von Bremsbefehlen, zu einer Verzögerung von einer Sekunde zwischen Eingabe des Befehls und Ausgabe des Steuersignals an die Bremsanlage. Ein Anzeichen für einen nicht auf mechatronische Systeme optimierten Entwicklungsprozess. Anders ist es kaum vorstellbar, warum die Bearbeitungszeit eines solch fundamentalen Befehls nicht im Zusammenspiel mit dem Gesamtsystem schon früher getestet worden ist.

An der HAW-Hamburg wird der hohen Bedeutung von mechatronischen Systemen und der Automatisierung von Prozessen mit eigenen Vorlesungsmodulen wie Mechatronik, Automatisierungstechnik und Mess-, Steuer- und Regelungstechnik Genüge getan. Hier wird den Studierenden ein detaillierter Eindruck von mechatronischen Systemen gegeben und der Umgang mit den zahlreichen Komponenten eines solchen Systems vermittelt. Jedes dieser Module umfasst seminarischen Unterricht, in dem die theoretischen Grundlagen geschaffen werden und ein Labor in welchem das Erlernte an praktischen Beispielen vermittelt wird. Dieses Zusammenspiel von Theorie und Praxis ist das große Alleinstellungsmerkmal der HAW-Hamburg gegenüber anderen Lehreinrichtungen.

1.1 Ziele und Inhalte der Arbeit

Durch diese Arbeit sollen für das Mechatronik Modul an der HAW-Hamburg praktische Versuche entwickelt werden. Für die Versuche wurden keine genauen Rahmenbedingungen gegeben. Es ist also auch Bestandteil dieser Arbeit geeignete Komponenten aus dem Bereich der Mechatronik zu wählen, mit denen in den Versuchen gearbeitet werden soll, sowie geeignete Aufgaben mit diesen Komponenten zu entwickelt.

Ein mechatronisches System setzt sich aus Komponenten der Bereiche Elektrotechnik, Mechanik und Informationsverarbeitung zusammen. Für die Beschreibung eines mechatronischen Systems hat sich diese Einteilung entsprechend der Funktionen der Bauteile als günstig erwiesen. In den nachfolgenden Betrachtungen werden aus dem Bereich der Elektrotechnik speziell die Komponenten Sensorik und Aktorik betrachtet. Die Funktionen und Grundlagen zu den jeweiligen Komponenten werden in Kapitel 2 vorgestellt. Auf eine Vorstellung der Mechanik als Teil des machatronischen Systems wird ganz bewusst verzichtet, da dieses Gebiet bereits in ausreichender Form, im Rahmen von verschiedensten Vorlesungen, detailliert behandelt wird. Für die Vorstellung eines mechatronischen Systems wurde auf eine bewährte Methodik aus dem Bereich der Konstruktionslehre zurückgegriffen. Dort ist es üblich von Systemen und Subsystemen zu sprechen und technische Anlagen oder Produkte nach solchen zu unterteilen. Ein ähnliches Vorgehen wird auch in der VDI Richtlinie 2206 zum Entwicklungsprozess mechatronischer Systemen erläutert. Diese wird im Verlauf der Arbeit ebenfalls vorgestellt und der Unterschied zu einem regulären Entwicklungsprozess dargelegt.

Im Verlauf der Bearbeitung dieser Arbeit hat sich herausgestellt, dass die Vielfallt mechatronischer Systeme unmöglich in einem Versuch vermittelt werden kann. Deshalb werden mehrere Laborversuche zu den oben genannten Komponenten eines machatronischen Systems entwickelt. Durch die Aufteilung der Lehrinhalte auf mehrere Versuche kann den einzelnen Bereichen mehr Zeit zur Verfügung gestellt werden. Ebenfalls vorteilhalft ergibt sich daraus, dass der Umgang mit der Hard- und Software im Labor schrittweise erlernt werden kann. Als Ziel wurde festgelegt, für drei Laborveranstaltungen je drei Versuche zu entwickeln. Für die vierte Laborveranstaltung werden keine Inhalte erarbeitet. In dieser können dann entweder eine Einführung stattfinden oder weiterführende Laborversuche

durchgeführt werden. Eine genaue Beschreibung des Konzepts hinter den Laborversuchen findet sich im Kapitel 3.

Schließlich werden in den Kapiteln 4, 5 und 6 die Laborversuche aus den Bereichen Informationsverarbeitung, Sensorik und Aktorik vorgestellt. Für die Mechanik werden aus den genannten Gründen keine Laborversuche entwickelt. Wie diese Einteilung zustande gekommen ist, findet sich ebenfalls im Kapitel 3. Für jeden Versuch wird eine passende Musterlösung im Anschluss an die Versuchsbeschreibung gegeben. In den Musterlösungen werden die in den Versuchen geforderten Programme gezeigt, oder eigene Messergebnisse dargelegt. Abschließend werden die erarbeiteten Laborversuche im Kapitel 7 noch einmal kritisch betrachtet und in Kapitel 8 einige Ansatzpunkte für einen weiteren Ausbau des Mechatronik Labors gegeben.

2 Das mechatronische System

Unter einem mechatronischen System versteht man allgemein ein System, bei dem die Integration von Elektronik, Mechanik und Informationsverarbeitung einen besonders hohen Grad erreicht hat. Jedes technische Gebilde kann als System mit Eingangsgrößen und Ausgangsgrößen betrachtet werden, über welche es mit der Umgebung in Wechselwirkung steht. Vereinfacht kann jede technische Anlage als ein System mit Eingangs- und Ausgangsgrößen verstanden werden. In Abbildung 1 ist ein solches System dargestellt. Solch ein System kann beliebig in Subsysteme unterteilt werden, in welchen die Gesamtfunktion des Systems in Teilfunktionen aufgeteilt wird. So lässt sich jede Funktion, egal wie komplex, in sehr einfache Teilfunktionen zerlegen. Für die Eingangs- und Ausgangsgrößen haben sich der Energie-, Stoff- und Signalumsatz als Betrachtungsgrundlage durchgesetzt.

Abb. 1 Darstellung eines Systems mit Eingangs- und Ausgangsgrößen, nach [20].

Der Energieumsatz beschreibt z.B. die Umwandlung von elektrischer in mechanische und thermische Energie. Mit dem Stoffumsatz werden alle möglichen Veränderungen an und mit Stoffen beschrieben. Stoffe können einfach nur transportiert werden, beschichtet oder es können aus ihnen Halbzeuge sowie Fertigprodukte entstehen. In jeder Maschine fallen Informationen an, welche in Form von Signalen verarbeitet werden. Signale können aufgenommen, gewandelt, gespeichert, bearbeitet und weitergeleitet werden. Die Signalverarbeitung innerhalb eines technischen Gebildes wird mit Hilfe des Signalumsatzes dargestellt. In technischen Anlagen ist, entsprechend der Aufgabe, meist entweder der Energie-, Stoff- oder Signalumsatz von entscheidender Bedeutung. So ist bei einem Transportband für Eisenerz der Stoffffluss des Eisenerz von entscheidender Bedeutung, der Energie- und Signalfluss treten hier in den Hintergrund, sind aber trotzdem vorhanden.

Bei technischen Systemen im Bereich der Kommunikation oder Signalverarbeitung ist der Signalumsatz entscheidend, ohne einen Energieumsatz aber nicht möglich. Auch wenn die Energieversorgung kein Problem darstellt, oder der Fluss sehr klein ist, ist er vorhanden und muss in der Systemdarstellung berücksichtigt werden.

Aus den Anforderungen der Industrie, nach integrierten, effizienten und kostengünstigen Lösungen, hat sich immer mehr die Anforderung nach einer Verbindung der verschiedenen Subsysteme herauskristallisiert. So war es vor einigen Jahren kaum denkbar einen Sensor oder Aktor als strukturmechanisches Bauteil in die Gesamtkonstruktion zu integrieren. In den modernen mechatronischen Anlagen von heute, ist dies schon fast zum Alltag geworden.

Bei einem mechatronischen System besteht die Besonderheit im Wesentlichen darin, dass Subsysteme mit Funktionen aus gänzlich unterschiedlichen Bereichen auftreten und miteinander verbunden werden müssen. Hier sind vor allem die beiden Bereiche der Mechanik und der Elektrotechnik relevant.

In Abbildung 2 ist ein mechatronisches System im Sinne der VDI Richtlinie 2206 abgebildet. Hier erkennt man den direkten Zusammenhang zwischen den einzelnen Subsystemen sowie den Signalfluss. Ebenso ist der Regelkreis zwischen Grundsystem, Sensorik, Informationsverarbeitung und Aktorik gut zu erkennen. Dieser stellt das Grundgerüst jedes mechatronischen Systems dar. Zusätzlich zu diesem Regelkreis können noch weitere Subsysteme hinzugefügt werden, sollte deren Betrachtung von Bedeutung sein. So kann es für die Sensorik erforderlich sein Messwerte aus der Umgebung zu erfassen, oder für die Informationsverarbeitung Befehle vom Menschen entgegen zu nehmen.

Im klassischen Entwicklungsprozess werden die Subsysteme meist von unterschiedlichen Abteilungen bearbeitet. Ein nicht zu bearbeitendes System wird dann nur als Blackbox mit definierten Schnittstellen betrachtet. So kann es zwar zu guten Lösungen innerhalb eines Subsystems kommen, für das Gesamtsystem stellt sich in einigen Fällen eine nicht optimierte Lösung ein. Um diesem Prozess entgegen zu wirken sind in der Konstruktionsmethodik verschiedene Ansätze bekannt. An dieser Stelle wird weiter nur die VDI Richtlinie 2206 betrachtet und deren Lösungsverfahren dargelegt.

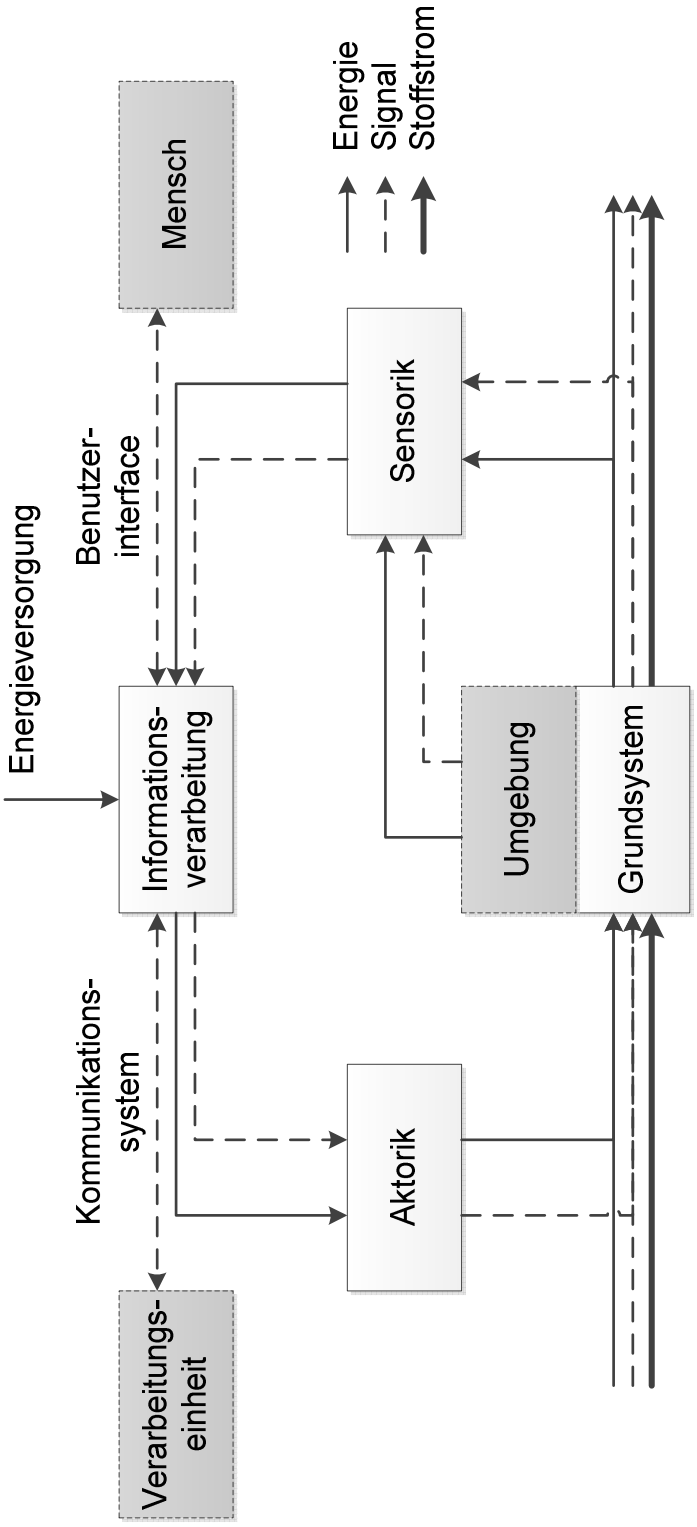

Abb. 2 Darstellung eines mechatronischen Systems mit den wichtigsten Komponenten, Signal- und Energieströmen und Anbindungen umgebener Systemen, modifiziert nach [16].

Entwicklungsprozesse werden bekannter weise in verschiedene Phasen unterteilt. In der VDI Richtlinie 2206 wird vorgeschlagen schon in frühen Phasen den Entwurf des Gesamtsystems in den Vordergrund zu rücken. Abbildung 3 verdeutlicht diesen Unterschied zu einem konventionellen Vorgehen, wie es in der VID Richtlinie 2221 vorgeschlagen wird.

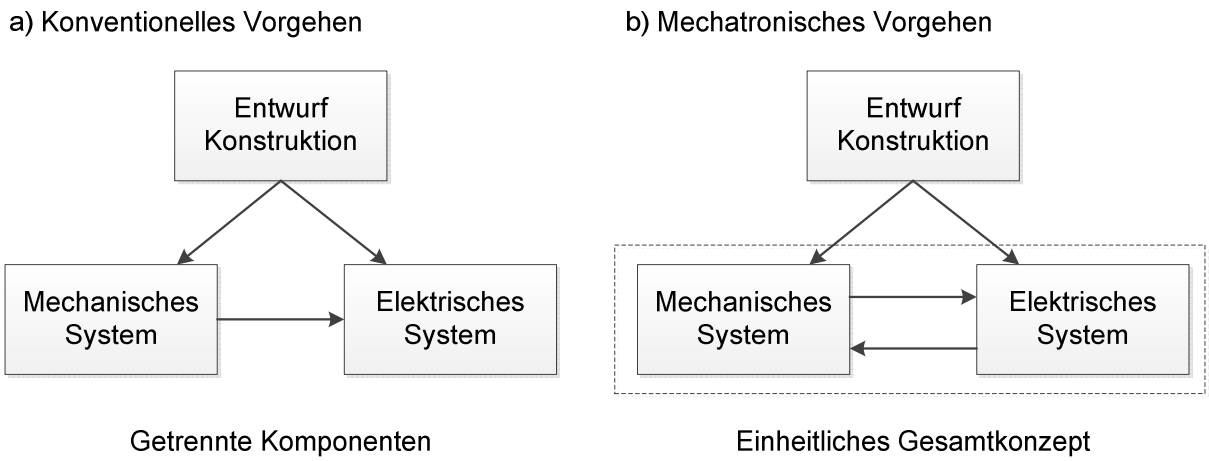

Abb. 3 Entwurf eines Systems nach konventionellem und mechatronischem Vorgehen [12].

Für die Bearbeitung des Gesamtsystems schlägt die VDI Richtlinie 2206 eine Kombination aus einem Mikro- und einem Makrozyklus vor. Der Mikrozyklus dient grundsätzlich der Lösungsfindung für speziell auftretende Probleme oder Anforderungen. Dieser wird jeweils so oft durchlaufen, bis ein adäquates Ergebnis vorliegt. Eingebettet in den Makrozyklus wird anschließend bewertet, in wieweit sich dieses Ergebnis mit den Anforderungen an das Gesamtsystem deckt. Für den Makrozyklus wird das V-Modell (siehe Abbildung 4, links) verwendet, da sich dieses als besonders geeignet erwiesen hat. Das V-Modell beschreibt lediglich den groben Rahmen eines Entwicklungsprozesses. Es gibt keine konkreten Lösungswege oder Lösungsfindungsverfahren vor. Je nach Entwicklungsaufgabe lassen sich die Phasen entsprechend der Anforderungen gestalten.

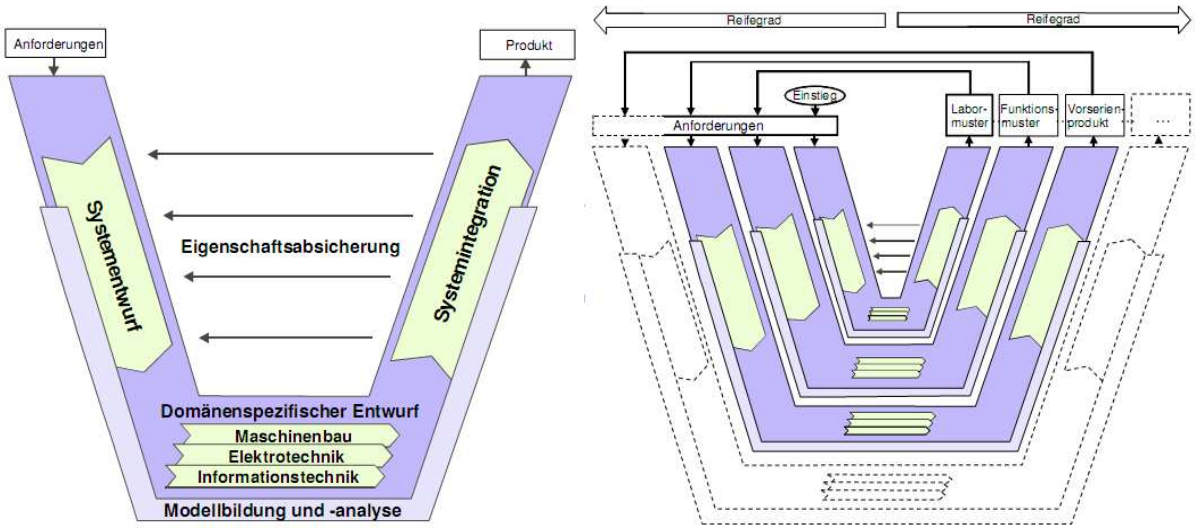

Abb. 4 Das V-Modell als Vorgehensmodell für einen Entwicklungsprozess, als einfacher und mehrfacher Durchlauf [16].

Ein Durchlauf des V-Modells entspricht einem Durchlauf im beschriebenen Makrozyklus. Beginnend mit dem *Systementwurf* und der dort stattfindenden Definition von Parametern für die Domäne spezifischen Entwicklungsbereiche. Anschließend werden die Mikrozyklen in den erforderlichen Entwicklungsbereichen Maschinenbau, Elektrotechnik und Informationstechnik durchlaufen. Abschließend werden die erarbeiteten Entwürfe wieder zu einem Gesamtsystem zusammengefügt. Während dieser *Systemintegration* werden die geforderten Eigenschaften aus dem *Systementwurf* validiert.

Wird das V-Modell mehrmals durchlaufen, kann mit jedem Durchlauf ein höherer Reifegrad des Produktes erreicht werden (siehe Abbildung 4, rechts). So können auch während des Entwicklungsprozesses noch Anforderungen überarbeitet und angepasst werden. Für die einzelnen Schritte gibt die VDI Richtlinie jeweils grobe Vorgehensweisen vor. Der Entwickler kann, je nach Entwicklungsstand, in die Richtlinie schauen und sich entsprechende Informationen holen, welche Aufgaben bzw. Arbeitsschritte als nächstes zu bearbeiten sind. Das Ziel der Richtlinie ist es jeden Entwickler ein möglichst modulares Grundgerüst an die Hand zu geben, mit dem er jede Entwicklungsaufgabe erfolgreich erfüllen kann. Auch für einen ganzen Entwicklungsprozess, ausgehend von einer Produktidee oder einer Anforderung an ein bestehendes System, soll die Richtlinie eine Struktur zur Verfügung stellen. Dies wird durch den Einsatz von einfachen Methoden für wiederkehrende Arbeitsschritte erreicht, wie dem wiederholten Durchlaufen des V-Modells.

Im weiteren Verlauf soll nicht weiter auf die VDI Richtlinie eingegangen werden, da sich diese wie schon beschrieben, vornehmlich mit dem Produktentwicklungsprozess

von mechatronischen Systemen beschäftigt. Weiter werden die einzelnen Elemente eines mechatronischen Systems betrachtet, sowie deren grundsätzliche Funktionen und Schnittstellen zu den anderen Subsystemen.

2.1 Aktoren

Mit Aktuatoren, auch Aktoren genannt, sind diejenigen Subsysteme benannt, welche für das Aufbringen von Kräften, Umsetzen von Bewegungen oder Einstellen von anderen physikalischen Größen, wie Druck oder Temperatur, verantwortlich sind. Aktoren sind in einer Regelstrecke hinter der Steuereinheit angeordnet und verändern entsprechend der angelegten Signale eine physikalische Größe innerhalb des zu regelnden Systems. Für die Umsetzung der Steuersignale benötigen sie immer eine Hilfsenergie. Das allgemeine Funktionsdiagramm eines Aktors ist in Abbildung 5 zu sehen.

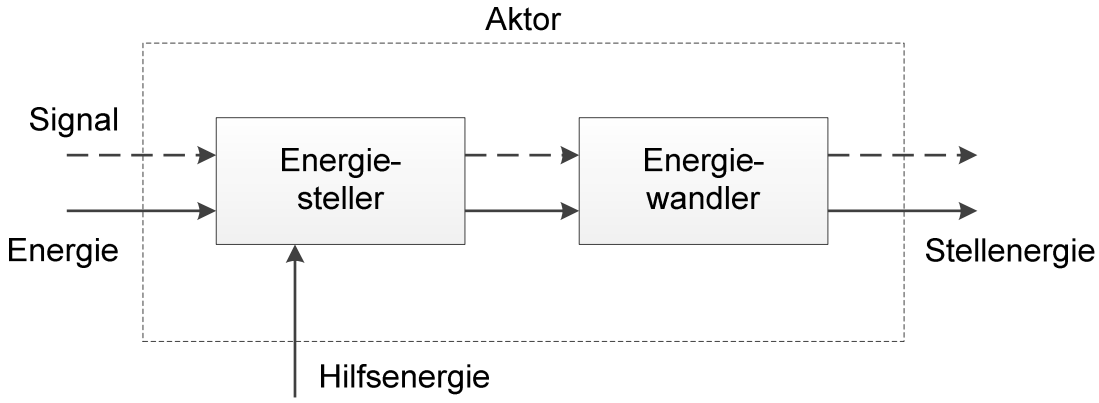

Abb. 5 Funktionsdiagramm eines Aktors, nach [12].

Die Steuersignale werden dem Aktor in der Regel über standardisierte Schnittstellen von einem Mikrokontroller zugeführt. Die Ausgangsgröße ist eine Energie oder Leistung, welche das nachgeschaltete mechanische System beeinflusst.
Aktoren werden nach ihrem Wirkprinzip oder ihrer Hauptaufgabe als Weg- oder Kraftaktoratoren eingeteilt. Bei einer Einteilung nach Wirkprinzipien ergeben sich die folgenden vier Gruppen von Aktoren:

- Elektromagnetische
- Fluidische
- Piezoelektrische
- und weitere Aktoren

Die wohl bekannteste Gruppe bilden die elektromagnetischen Aktoren. Hierunter sind die klassischen rotierenden elektrischen Maschinen eingeteilt, wie sie in der Antriebstechnik überall zu finden sind. Den größten Teil dieser Aktorengruppe machen die Synchron- und Asynchronmaschinen aus. Asynchronmaschinen sind die am weitesten verbreiteten Elektromotoren. Ihr Einsatzbereich reicht von kleinen Antrieben für Kompressoren bis zu Seilantrieben von Bergbahnen. Der größte Vorteil dieser Motoren ist das Fehlen von Kommutator und Bürsten. Was zu einem verringerten Wartungsaufwand führt. Synchronmaschinen werden hauptsächlich als Drehstromgeneratoren in Kraftwerken eingesetzt. Diese zeichnen sich durch ihren hohen Wirkungsgrad aus. Auch in der Robotik und Automatisierung kommen überwiegend Elektromotoren zum Einsatz. Dort werden Gleichstrom-Servomotoren und Schrittmotoren verwendet. Solche Motoren sind besonders gut regelbar. Die Bewegung wird durch die Ausnutzung elektromagnetischer Felder erzeugt. Auf einen stromdurchflossenen Leiter in einem Magnetfeld wirkt bekanntlich die Lorentz-Kraft. In elektrodynamischen Wandlern wird, unter der Benutzung der Lorenz-Kraft, entweder eine lineare Kraft oder eine Drehmoment erzeugt. In der Gruppe der elektrodynamischen Drehwandler finden sich alle bekannten Bauformen von Elektromotoren. In die Gruppe der elektrodynamischen Linearwandler fallen zum Beispiel Magnetlager, die Antriebseinheiten von Magnetschwebebahnen oder Linear-Magnetführungen.

Zu den fluidischen Aktoren gehören diejenigen Stelleinrichtungen, welche mit Hilfe von flüssigen oder gasförmigen Stoffen, Kräfte oder Bewegungen erzeugen. Darunter fallen zum Beispiel Pumpen, Ventile oder Hydromotoren.

Die Gruppe der Piezoaktoren beschreibt diejenigen Aktoren, welche auf Basis des Piezoeffekts arbeiten. Bei bestimmten Materialien führt das Anlegen einer Spannung zu einer Volumenänderung. Dieser Effekt wird zum Ausführen von sehr kleinen Bewegungen oder dem Aufbringen von Kräften genutzt. Piezoaktoren sind für Stellsignale mit besonders hohen Frequenzen geeignet. Neuartige Aktoren auf Grundlage des Piezoeffektes finden sich in vielen Produkten aus dem Bereich der Medizintechnik, dort vor allem für Geräte, welche innerhalb des Körpers eingesetzt werden.

Zu guter Letzt wurden alle weiteren Aktoren, welche nicht nach einen der vorherigen Grundlagen arbeiten, in eine Gruppe zusammengefasst. Dies sind in der Regel Aktoren für Spezialanwendungen, welche noch kaum Verbreitung finden oder auf

Wirkprinzipien beruhen, welche sich noch in der Grundlagenforschung befinden. Als Beispiel seien die elektrorheologen Aktoren genannt. Sie funktionieren mit Flüssigkeiten, welche ihre Viskosität beim Anlegen eines elektrischen Felds ändern. Besonders interessant erscheint der Effekt für Stoßdämpfer zu sein.

Auswahl von Aktoren für die Laborveranstaltung

Aus den vorgestellten vier Gruppen von Aktoren mussten nun einige für die Verwendung im Labor ausgewählt werden. Damit das Labor seine Nähe zur Praxis behält, sollte mindestens ein Vertreter aus der Gruppe der elektromagnetischen Aktoren verwendet werden. Für den ersten Versuch im Aktoriklabor (siehe Versuch 7, Kapitel 6.1) wurde deshalb ein DC-Motor ausgewählt. Dieser lässt sich einfach steuern und ist in der Industrie weit verbreitet. Aus derselben Gruppe wurden weiter ein DC-Ventilator und ein Lautsprecher ausgesucht. Diese beiden wurden gewählt, um den Studierenden die Ähnlichkeiten der Steuerungsmethodik, bei gänzlich unterschiedlichen Ausgangsgrößen, näher zu bringen.
Die so gewählten drei Aktoren gehören alle zur Gruppe der elektromagnetischen Aktoren. Nun musste sich noch, für einen weiteren Aktor, aus einer der anderen Gruppen entschieden werden. Fluidische Aktoren kamen für die Verwendung im Labor nicht in Frage. Denn diese erfordern in der Regel den Aufbau von Druckluftanlage (für pneumatische Aktoren) oder einer Anlage, welche Flüssigkeiten als Medium verwendet. Solche Aktoren werden zudem schon ausführlich in den Laboren der Vorlesung Fluidtechnik verwendet. Somit grenzte sich die Auswahl auf die beiden Gruppen der piezoelektrischen und der weiteren Aktoren ein.
Mit Hinblick auf eine weitere Verwendung in zukünftigen Laborversuchen wurde schließlich ein Peltier-Element aus der Gruppe der weiteren Aktoren ausgewählt. Dieses basiert auf Effekten rund um Halbleitermaterialien. Diese sollen im Verlauf der Arbeit noch eine wichtige Rolle für die Steuerung der Aktoren spielen, außerdem werden Sensoren auf Basis von Halbleitern verwendet. Im Fall des Peltier-Elements wird gezeigt, dass diese Halbleiter auch direkt als Aktoren fungieren können.
Für den letzten Versuch (siehe Versuch 9, Kapitel 6.5) wurde schließlich noch ein Schrittmotor ausgewählt. Schrittmotoren sind die am häufigsten verwendeten Motoren bei der computergestützten Steuerung von Anlagen und Geräten. Sie zeichnen sich vor allem durch ihre hohe Genauigkeit und sehr gute Steuerbarkeit aus. In letzter Zeit werden die Komponenten für die Steuerung solcher Motoren,

aufgrund ihrer zunehmenden Verbreitung immer günstiger, weshalb sich Schrittmotoren zunehmend auch in Konsumprodukten durchsetzten. Ein Beispiel dafür sind z.B. Drucker welche in den letzten Jahren stark im Preis gesunken sind. Weiteren Anwendungsgebiete sind Heizungsanlagen in Autos, Antriebe von CD, DVD oder Blu-ray Laufwerken sowie Roboter in Produktionsstraßen. Schrittmotoren lohnen sich jedoch nicht für den Einsatz als Leistungsmaschine wie z.B. einem Generator oder elektrischem Antrieb eines Kraftfahrzeugs.

2.2 Sensoren

Als Sensoren versteht man diejenigen Elemente in einem System, welche nicht elektrische Eingangsgrößen in elektrische Informationssignale umwandeln.

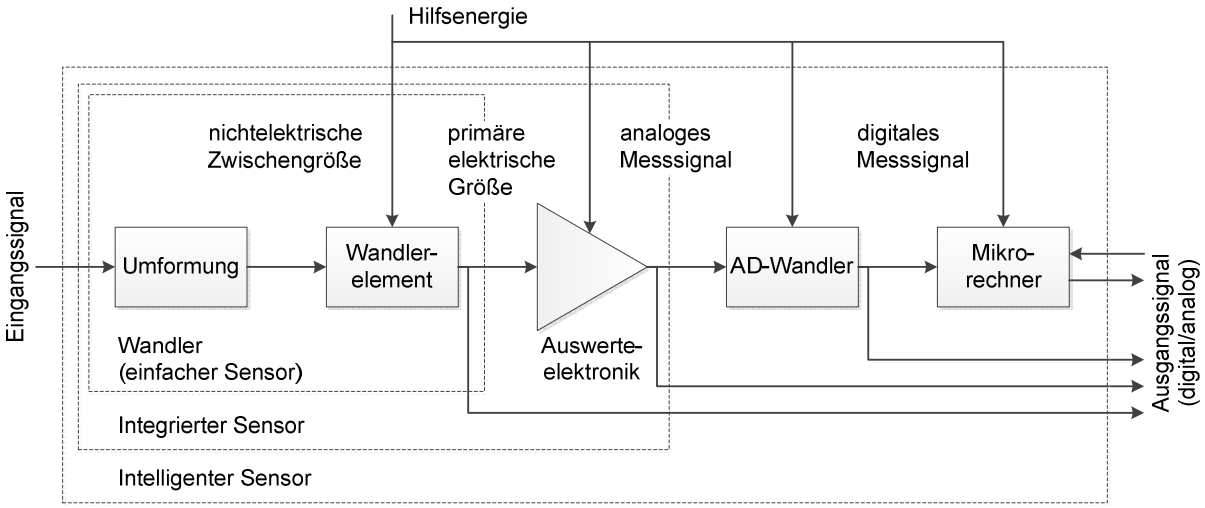

Abb. 6 Integrationsgrad von Sensoren, nach [12].

Für die Steuerung und Regelung eines Systems ist es notwendig, mit Hilfe von Sensoren, zustandsabhängige Größen zu erfassen. In Abbildung 6 ist der Unterschied zwischen Sensoren unterschiedlichen Integrationsgrades dargestellt. Die Eingangssignale werden je nach Integrationsgrad des Sensors in ein einfaches elektrischen Signal, analoges oder ein digitales Messsignal umgewandelt. Je nach Integrationsgrad spricht man von *Wandler*, *Integrierter* - oder *Intelligenter Sensor*. Ein *Wandler* oder auch *einfacher Sensor* liefert, nach der Umformung des Eingangssignals in eine nichtelektrische Zwischengröße und anschließenden Umwandlung, eine primäre elektrische Größe. Der Zwischenschritt der Umformung des Eingangssignals in die nichtelektrische Zwischengröße erfolgt nicht bei allen zu

messenden Signalen. Man unterscheidet hier die direkten und indirekten Messverfahren. Bei indirekten Messverfahren, wie der Kraftmessung, wird meist die Verformung als Folge der Krafteinwirkung gemessen, sprich die Kraft wird indirekt über die Verformung bestimmt. Als Beispiel für ein direktes Messverfahren kann das Drahtpotentiometer genannt werden. Dieses gibt direkt, ohne Zwischengröße, die Längenänderung in einer Widerstandsänderung aus.

Was alle Messverfahren gemeinsam haben ist das Wandler Element. In diesem wird aus dem Eingangssignal, oder der Zwischengröße, das eigentliche primäre elektrische Signal erzeugt. Hierfür werden die verschiedensten physikalischen Effekte ausgenutzt. Bei der Temperaturmessung mit Hilfe eines Widerstandsthermometers wird an dieser Stelle aus der Temperatur ein Widerstand, zum Beispiel mit Hilfe der physikalischen Eigenschaften von Halbleitern. Dieser Widerstand kann nun von einer geeigneten Elektronik gemessen werden und als Temperatur interpretiert werden. Die nächste Integrationsstufe wird als *Integrierte Sensoren* benannt bei denen diese Auswertung bereits innerhalb des Sensors stattfindet. Diese Sensoren liefern ein analoges Messsignal. Das analoge Messsignal eines Temperatursensors kann z.B. 0 bis 10 V über die Temperatur 0 bis 100 °C betragen. Eine weitere Integrationsstufe bildet der *intelligente Sensor*. In diesem kann mit Hilfe eines AD-Wandlers aus dem analogen ein digitales Signal erzeugt werden. Mit einem eingebauten Mikrorechner in den Sensor können dann weitere Anpassungen des Messwerts, verschiedene Auswertungen oder sogar die Erzeugung von Steuersignalen innerhalb des Sensors durchgeführt werden.

Die wichtigsten Kenngrößen von Sensoren sind der Messbereich, die Auflösung und die Genauigkeit. Der Messbereich beschreibt den Bereich der Eingangswerte, der auf den Bereich der Ausgangswerte abgebildet werden soll. Bei dem eben beschriebenen Temperatursensor wird der Messbereich 0 bis 100 °C linear proportional auf den Ausgangswertebereich von 0 bis 10 V abgebildet. Die Auflösung kennzeichnet zwei Eingangswerte, welche noch eindeutig zu zwei verschiedenen Ausgangswerten führen. Als Genauigkeit bezeichnet man die Summe aller möglichen statistischen Fehler, welche aus dem Sensor herrühren und das Ausgangssignal verfälschen.

Ähnlich wie bei Aktoren werden auch Sensoren nach ihren Wirkprinzipien gruppiert. Zusätzlich ist es bei Sensoren sinnvoll nach der Art der Messgröße zu unterscheiden.

Für die Messung kinematischer und dynamischer Größen wird eine Vielzahl von physikalischen Wirkprinzipien für die Erfassung der Messwerte verwendet. In der Kinematik geht es vorrangig um die Erfassung der Bewegung von Körpern. Messtechnisch sind also Beschleunigungen, Geschwindigkeiten und Wege zu erfassen. Wobei sich Beschleunigung und Geschwindigkeit als zeitliche Ableitungen des Weges bestimmen lassen. Dies gilt sowohl für translatorische (Weg s, Geschwindigkeit $v = \dot{s}$, Beschleunigung $a = \dot{v} = \ddot{s}$) als auch für rotatorische Bewegungen (Winkel φ, Winkelgeschwindigkeit $\omega = \dot{\varphi}$, Winkelbeschleunigung $\dot{\omega} = \ddot{\varphi}$). Unter den dynamischen Größen werden die Kraft F und das Drehmoment M verstanden. Tabelle 1 zeigt eine Übersicht von Wirkprinzipien, deren äquivalente Messgröße und die damit erfassbare kinematische oder dynamische Messgrößen.

		Kinematische/dynamische Größe			
Wirkprinzip	**Messgröße**	s, φ	v, ω	$a, \dot{\omega}$	F, M
potentiometrisch	Ohmscher Widerstand *R*	x			x
induktiv	Induktivität *L*	x			x
kapazitiv	Kapazität *C*	x			x
Ultraschall-Laufzeit	Zeit *t*	x			
magnetisch	Magnetische Flussdichte *B*	x	x		x
magnetostriktiv	*B, t*	x			
optisch	Intensität *I*	x	x		
piezoelektrisch	Ladung *Q*		x	x	
piezoresistiv	Widerstand *R*	x		x	x

Tabelle 1 Auflistung einiger physikalischer Wirkprinzipien für die Erfassung von kinematischen und dynamischen Messgrößen, modifiziert nach [12].

Für jedes in Tabelle 1 genannte Wirkprinzip sind noch unterschiedlichste Sensoraufbauten denkbar. So sind in der Gruppe der potentiometrischen Sensoren für die Weg und Winkel Messung z.B. Drahtpotentiometern oder für Kraft und Momenten Messung z.B. Dehnungsmessstreifen (DMS) zur Deformationsmessung zu finden. Aus diesen Messgrößen lassen sich dann weitere abgeleitete Größen berechnen. Der Druck auf eine Fläche lässt sich beispielsweise aus der Kraft und der Krafteinwirkungsfläche bestimmen.

Auswahl von Sensoren für die Laborveranstaltung

Für die Laborveranstaltung wurden Sensoren, entsprechend der Abbildung 6, mit verschiedenen Integrationsgraden gesucht. Es mussten *einfache*, *integrierte* und *intelligente Sensoren* ausgewählt werden.

Aus der Gruppe der *einfachen Sensoren* stand eine ganze Reihe von möglichen Sensoren zur Auswahl. Es wurde sich schließlich für einfache Schalter (eingebaut in einen Joystick), eine Fotodiode und ein Temperatursensor entschieden. Schalter sind die einfachste Art von Sensoren sie werden auch als binäre Sensoren bezeichnet. Schalter geben ein binäres Signal aus, welches die beiden Zustände wahr oder falsch (0 oder 1) einnehmen kann. Solche Sensoren werden z.B. als Endlagenschalter oder Bedienelemente eingesetzt. Die Fotodiode zählt ebenfalls zu den einfachen Sensoren. Die Funktion dieser basiert auf der Verwendung von Halbleitermaterialien, welche das eintreffende Licht in einen elektrischen Strom umwandeln (primäre elektrische Größe). Das primäre Ausgangssignal einer Fotodiode kann mit geeigneter Auswerteelektronik in ein analoges Signal umgewandelt werden. Im ersten Versuchen mit Sensoren (siehe Versuch 4, Kapitel 5.1) wird die Fotodiode für die Erstellung eines binären Signals verwendet. In der Praxis werden Fotodioden für die verschiedensten Aufgaben eingesetzt. Vor allem in Bereichen, wo berührungslos gemessen werden soll. So z.B. bei Lichtschranken, Inkrementalgebern, berührungslosen Drehzahlmessern oder bei Präsenzmeldern. Sie werden auch bei der Übertragung von Signalen eingesetzt. Jeder Fernseher verfügt über Fotodioden, welche die Steuersignale der Fernbedienung empfangen. In der Elektrotechnik werden Fotodioden zusammen mit Licht emittierenden Dioden (LED’s) in Optokopplern für die galvanische Trennung von Baugruppen und Schaltungen eingesetzt.

Der ausgewählte Temperatursensor (ebenfalls ein Halbleiterbauelement) wurde für den dritten Versuch im Sensoriklabor ausgewählt (siehe Versuch 6, Kapitel 5.5), weil er sich besonders für die Veranschaulichung der verschiedenen Integrationsstufen von Sensoren eignet. Zudem kann an diesem Sensor gut demonstriert werden, was sich hinter der *Auswerteelektronik* in einem *integrierten Sensor* verbirgt. Temperatursensoren finden sich in nahezu allen technischen Anlagen zur Überwachung der Betriebstemperatur. Ebenso in allen Anlagen, bei denen es um die Klimatisierung geht. Temperatursensoren können auf verschiedene physikalische Effekte beruhen. Der hier verwendete Sensor auf Basis von Halbleitern steht lediglich

für eine Vielzahl von möglichen Sensoren. Ein weiterer oft verwendeter physikalischer Effekt ist die Widerstandsänderung von Materialien wie z.B. Keramik, Silizium oder Platin bei Temperaturänderung.
Aus der Gruppe der integrierten Sensoren wurden für den zweiten Versuch (siehe Versuch 5, Kapitel 5.3) ein Neigungssensor und ein Beschleunigungssensor ausgewählt. Der Neigungssensor misst in zwei Achsen die Neigung und gibt zwei entsprechende Spannungen aus. Der Beschleunigungssensor misst die absolute Beschleunigung einer internen Masse. Auch dieser gibt entsprechend der gemessenen Beschleunigung eine Spannung aus. Mehr zu diesen beiden Sensoren folgt in der Versuchsbeschreibung zum Laborversuch 5. Neigungssensor werden z.B. in Baumaschinen für den Unfallschutz oder für die Überwachung von Robotern eingesetzt. Beschleunigungssensoren, wie der in diesem Fall verwendete, werden ebenfalls für die Überwachung von Anlagen eingesetzt. Mit diesen können z.B. Schwingungen von Wellen gemessen werden, um Beschädigungen an Lagern festzustellen.
Beide Sensoren eignen sich gut für den Einsatz im Labor, da sie einen hohen anschaulichen Wert aufweisen. Neigungen lassen sich leicht mit Hilfe eines Geodreieckt nachvollziehen und Beschleunigungen mit Hilfe von Bewegungen simulieren. Der Beschleunigungssensor kann aufgrund seiner Bauart auch für die Neigungsmessung eingesetzt werden. Weiteres dazu folgt in der Versuchsbeschreibung des Versuchs 5, Kapitel 5.3.
Aus der Gruppe der intelligenten Sensoren wurde kein Sensor gewählt, da die Auswertung der Signale dieser Sensoren mit einem größeren technischen Aufwand verbunden ist. Für solche Sensoren müsste dann, je nach Bauart, ein Bussystem aufgebaut werden, über welches die Signale gesendet und empfangen werden könnten. Für eine Laborveranstaltung ist die Verwendung von einfacheren Sensortypen weitaus anschaulicher, kostengünstiger und deshalb zu bevorzugen.

2.3 Informationsverarbeitung

Die Informationsverarbeitung, welche auch als Prozessdatenverarbeitung oder Steuer-, Regeleinheit bezeichnet wird, bildet das Verbindungselement zwischen den Sensoren und Aktoren. Sie dient der Auswertung eingehenden Signale und Ausgabe von Steuersignalen. Aktoren bringen im Allgemeinen eine Kraft, Bewegung oder

Energie in ein Grundsystem ein. Sensoren messen die relevanten Messgrößen, welche für die Funktion des Grundsystems von Bedeutung sind. Im Falle einer Regelung wertet die Informationsverarbeitung diese Signale aus und passt die Steuersignale entsprechend an. Bei einer Steuerung wird auf die Auswertung oder Rückführung der Messsignale verzichtet. Man spricht auch von offenen und geschlossenen Regelkreisen.

In der Automatisierungstechnik werden für die Informationsverarbeitung vor allem speicherprogrammierbare Steuerungen (SPS) eingesetzt. Historisch sind SPS aus den Relaisschaltungen hervorgegangen. Sie besitzen Eingänge, Ausgänge, ein Betriebssystem und eine Schnittstelle über die das Anwenderprogramm geladen werden kann. Das Programm legt fest wie die Ausgänge, in Abhängigkeit zu den Eingängen, geschaltet werden. Eine SPS zeichnet sich vor allem durch ihre hohe Flexibilität aus. Durch zunehmende Standardisierung und stätige Erweiterung der Funktionsmöglichkeiten haben SPS die klassischen festverdrahteten Steuerungen mit Relais fast vollständig verdrängt. Die neusten Bemühungen bei der SPS Entwicklung gehen in Richtung Mensch-Maschine-Schnittstelle, Alarmierung und Aufzeichnung von Betriebsdaten.

Die Arbeitsweise einer SPS wird als zyklische Bearbeitung bezeichnet. Ein Zyklus besteht aus drei Teilen. Zum Beginn wird ein aktuelles Prozessabbild erstellt. Die aktuellen Werte an den Eingängen werden hierfür in den Speicher geschrieben. Darauf folgt die Bearbeitung des Anwenderprogramms. Das Programm wird Zeile für Zeile ausgeführt und benutzt hierfür das Prozessabbild aus dem Speicher. Der Zyklus endet mit dem Schreiben der neuen Signale auf die Ausgänge. Die Durchlaufzeit eines Zyklus wird Zykluszeit genannt. Maßgeblich beeinflussend für die Länge der Zykluszeit ist der Umfang des Anwenderprogramms. Je nach Programmierweise und Hardwareausstattung der SPS sind auch Multitasking und unterbrechbare Tasks möglich.

Trotz der vielfältigen Einsatzmöglichkeiten konnten sich SPSen nicht in allen Bereichen durchsetzen. In einigen Fällen ist der Einsatz einfach nicht wirtschaftlich, oder die Möglichkeiten der SPS stoßen an ihre Grenzen. Typische Beispiele, bei denen der Einsatz eine SPS wirtschaftlich nicht sinnvoll ist, sind Steuerungen mit sehr wenigen Ein- und Ausgängen. Hierfür sind in der Regel elektronische Bausteine mit integrierten Schaltkreisen (*integrated circuit*, IC) vorhanden, und in der Anschaffung günstiger.

Neben der Steuerung über die reinen SPS, sind auch Systeme mit PC basierter Steuerung von Automatisierungsprozessen bekannt. Die Vorteile solcher Systeme liegen bei der Kosteneinsparung für Visualisierungssysteme, den unbegrenzten Speicherressourcen eines PC's und direkte Vernetzung im Intranet, mit Zugriff auf Leitrechner, Server und das Internet. Jedoch ergeben sich bei der Benutzung von PC-Lösungen die klassischen Nachteile, wie geringe Betriebssicherheit, kurzlebige Hardware (im Vergleich zur Anlagentechnik) und die eingeschränkte Echtzeitfähigkeit. Mehr zu Echtzeitsystemen folgt im Kapitel 3.2 bei der Vorstellung der Hardware für das zu entwickelnde Mechatroniklabor.

Aus den beiden Systemen SPS und PC mit SPS (Hard- oder Software) hat sich eine dritte Variante entwickelt. Dieses vereint die Vorteile beider Systeme, sprich die Langlebigkeit und Betriebssicherheit der SPS, mit dem Funktionsumfang von PC Lösungen. In der Praxis wird dies durch die Verwendung der FPGA-Technologie [07] erreicht. Mehr zum Thema FPGA und eine Erläuterung der Funktionsweise folgt ebenfalls in Kapitel 3.2.

Am 1. August 1994 wurden in der internationale Norm IEC 61131-3 fünf Programmiersprachen für die Programmierungen von SPSen definiert. In diesen Sprachen können alle SPSen der unterschiedlichen Hersteller programmiert werden. So ist es möglich Programme der einen SPS mit geringem Aufwand auf eine SPS eines anderen Herstellers zu übertragen. Die Standardisierung hat ebenfalls dazu geführt, das eine Vielzahl an Programmiertools auf dem Markt verfügbar sind. Darunter findet sich die Software Codesys [01], MULTIPROG [13] oder die an die IEC 61131-3 angelehnte Programmierumgebung STEP 7 [23] für SPSen von SIEMENS.

Bei der Verwendung von Mikrocontrollern für die Informationsverarbeitung wird das Angebot an Programmierumgebungen, entsprechend der Auswahl an Mikrokontrollern, noch größer. Dabei gibt es Softwarelösungen, welche verschiedene Programmiersprachen und Mikrocontroller unterstützen, aber auch Lösungen, welche für spezielle Controller vorgesehen sind. Einige open-source Umgebungen für Mikrocontroller sind z.B. ErikaEnterprise [08] oder Arduino [02]. ErikaEnterprise ist ein Betriebssystem für *embedded systems* und verwendet die Programmiersprache C. Arduino ist ein Open Source Projekt, welches neben der Software auch Hardware mit verschiedensten Schnittstellen liefert, auch diese Hardware wird mit C programmiert. Neben diesen Tools, welche mit textbasierten Programmiersprachen

arbeiten, können auch Systeme mit grafischen Umgebungen für die Programmierung von Mikrocontrollern verwendet werden. Mit der Software MATLAB [15] ist es möglich geschriebene Programme in C zu kompilieren und dann auf geeignete Mikrocontroller zu laden. Dies ist z.B. mit dem Arduino möglich [26].

Auswahl eines geeigneten Systems für die Laborveranstaltung

Um den aktuellen Entwicklungen in der Industrie, zu leistungsfähigen Echtzeitsystemen gerecht zu werden, stand schon früh fest, Hardware zu wählen welche unter Echtzeitbedingungen arbeiten kann. Für das Labor war es ebenso wichtig ein System zu wählen, welche leicht den gestellten Aufgaben entsprechend konfiguriert werden kann. Ein gewisses Maß an Benutzerfreundlichkeit und Robustheit sollten ebenfalls gewährleistet werden.
Aufgrund der Anforderung industriegerechte Hardware zu verwenden schied z.B. das günstige Arduino System von vornherein aus. Die Verwendung einer SPS für das Labor wäre zwar denkbar, aber Aufgrund der Tatsache, dass SPSen schon in den Laborveranstaltungen der Vorlesungen Mess-, Steuerungs- und Regelungstechnik sowie der Automatisierungstechnik verwendet werden und nur eingeschränkt echtzeitfähig sind, schied auch die Verwendung einer SPS aus.
Somit ist die Wahl schließlich auf Hard- und Software von National Instruments (NI) gefallen. Die verwendete Hardware ist echtzeitfähig und bildet zusammen mit der Software LabVIEW von NI ein geschlossenes System. Die Software zeichnet sich durch eine einfache Bedienung und Einsteigerfreundlichkeit aus. Zum Zeitpunkt der Erstellung dieser Arbeit wird LabVIEW und die passende Hardware von NI in keiner Vorlesung oder einem Labor verwendet. Dies soll nun, vor allem Aufgrund der stetig steigenden Bedeutung dieses Systems in der Industrie, im Machatroniklabor nachgeholt werden. Eine genauere Beschreibung der Hard- und Software, sowie Vor- und Nachteile dieser folgen im Kapitel 3.1 und 3.2.

3 Laborkonzept

An der HAW Hamburg versteht man unter einem Labor, einen praktischen Unterricht, bei dem die Studierenden aufgefordert sind selbstständig Aufgaben, entweder am PC, oder unter Benutzung verschiedenster Anlagen und Gerätschaften, zu erledigen. Die HAW verfolgt das Konzept, den Studierenden im Vergleich zu Universitäten eine größere Anzahl von solchen Laboren zu bieten. Die Ausbildung ist so praxisorientierter. Dies erleichtert den Studierenden den Berufseinstieg, da sie schon während des Studiums mit technischen Anlagen und Methoden arbeiten, welche auch in der Industrie eingesetzt werden.
Die Aufgaben in einem Labor können einfache Versuche sein, bei denen es um die Dokumentation der Betriebsparameter von Anlagen geht, oder komplexere Untersuchungen von Werkstoffen, chemische Versuche oder Gestaltung von Bauteilen mit Hilfe von CAD Systemen. Die Laborinhalte sind genauso vielseitig wie die vielen Facetten des Maschinenbaus. Deshalb ist es auch nicht einfach die genauen Inhalte eins Labors zu definieren. Im begleitenden Labor zur CAD-Vorlesung wird vorrangig am PC gearbeitet und der Umgang mit der Software CATIA V5 erklärt. Im Labor der Vorlesung Elektrotechnik werden hingegen praktische Untersuchungen an verschiedenen Versuchsaufbauten durchgeführt. Im Werkstoffkundelabor werden Materialproben unter Verwendung unterschiedlichster Verfahren untersucht. Im Mechatronik Labor soll diese Praxisnähe nun, unter Verwendung von echtzeitfähigen Controllern und Komponenten mechatronischen Systeme, umgesetzt werden.
Ein Labor soll aus den folgenden drei Grundbausteinen bestehen:

- Theorieteil
- praktisches Arbeiten
- Dokumentation und Auswertung von Messergebnissen

Der Theorieteil kann je nach Labor entweder in Form von begleitenden Unterlagen, als Vorarbeit in der Vorlesung, oder als Selbststudium stattfinden. Die Unterlagen sollen dann Hintergrundinformationen zu verwendeten Komponenten, Datenblätter oder nötige Formeln enthalten.

Für den praktischen Arbeitsanteil muss je nach vorhandener Laborausstattung entschieden werden, was die Studierenden selber herstellen, komplettieren, anschließen oder benutzen können.
Die Dokumentation und Auswertung der Ergebnisse ist obligatorisch. Eine Nacharbeitung in Form eines Laborprotokolls kann vorgesehen werden. Falls dies nicht vorgesehen ist, muss die Möglichkeit bestehen, während des Labors eine Dokumentation oder Auswertung zu erstellen, welche wiederum einfach und schnell zu kontrollieren sein soll.
In dieser Arbeit werden Versuche nach dem Baukastenprinzip entwickelt, welche dann nach Belieben zu einem Labor kombiniert werden können. Ziel ist es für drei Laborveranstaltungen mindestens je drei Versuche zu entwickeln. Die Inhalte der Labore richten sich nach den Grundbausteinen eines mechatronischen Systems. So sollen unter den Überschriften Informationsverarbeitung, Sensorik und Aktorik Versuche entstehen, welche sich möglichst flexibel zusammenstellen lassen. Weiterhin werden die Versuche so konzipiert, dass sie in gewissem Maße aufeinander aufbauen. Damit soll erreicht werden, dass Informationen aus einem vorherigen Versuch erneut abgerufen werden müssen. Für die nach Lehrplan vorgesehene vierte Laborveranstaltung werden in dieser Arbeit keine Versuche entwickelt. Dieses Labor kann z.B. für eine Einführung in die Software LabVIEW verwendet werden oder Versuche, welche auf den hier vorgestellten aufbauen. Einige Ideen für weiterführende Versuche werden im Kapitel 8 vorgestellt.
Mit dieser Arbeit sollten natürlich auch die passenden Musterlösungen für die Versuche erstellt werden. Diese sollen in erster Linie die geforderten Programme oder Hardwarekomponenten liefern und dokumentieren. Für die Aufgaben in den Versuchen gibt es sicher eine Vielzahl von Lösungen, welche nicht in den Musterlösungen vorgestellt werden. Damit die Studierenden jedoch möglichst auf einheitlichem Weg die gestellten Aufgaben bearbeiten, werden in den Versuchsbeschreibungen mögliche Lösungswege indirekt vorgegeben.
Im Anhang 10.1 werden die, für die Erstellung der Versuche benutzte Hardware, sowie die elektronischen Komponenten für die Herstellung der Schaltungen festgehalten.

3.1 Vorstellung der Software

Im Mechatronik Labor soll Hard- und Software von Nation Instruments (NI) zum Einsatz kommen. Von NI wurde 1986 die Entwicklungsumgebung LabVIEW (Laboratory Virtual Instrumentation Engineering Workbench) auf den Markt gebracht. Anfangs wurde sie für den Macintosh II geschrieben. Heute gibt es die Entwicklungsumgebung auch für Windows und Linux. In LabVIEW wird mit der *G* genannten grafischen Programmiersprache gearbeitet, diese basiert auf dem Datenfluss-Modell. Die ursprüngliche Verwendung von LabVIEW lag in der Datenerfassung und -verarbeitung. In den letzten Jahren hat sich LabVIEW allerdings immer weiter zu einer universellen Programmiersprache entwickelt, was nicht unwesentlich mit der stetig zunehmenden Verbreitung einhergeht.

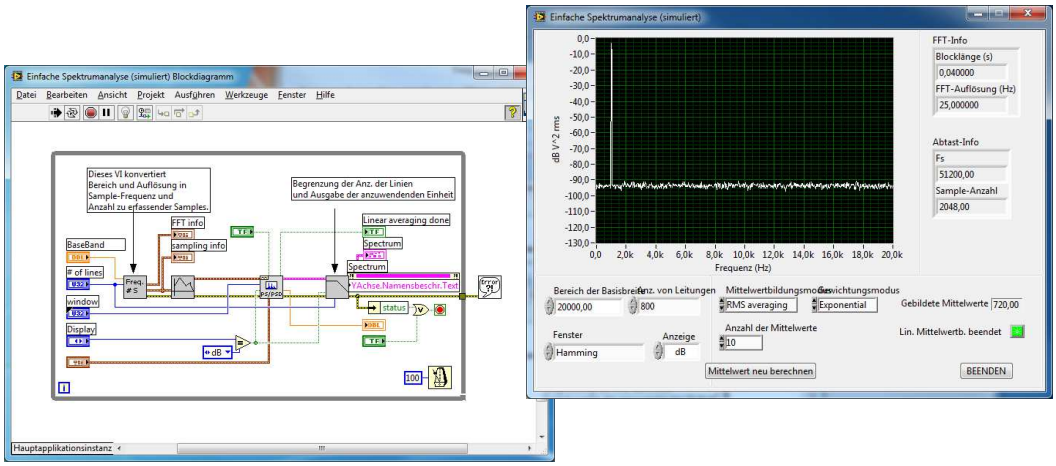

Abb. 7 Die beiden Ebenen eines LabVIEW Programms, oben rechts das Frontpanel, unten links das Blockdiagramm. Zu sehen ist das Beispielprogramm „Einfache Spektumanalyse (simuliert)“ aus der LabVIEW Hilfe.

Jedes LabVIEW Programm besteht aus zwei Komponenten. Der *Frontpanel* genannten Benutzeroberfläche (Abbildung 7, rechts) und dem *Blockdiagramm* genannten Bereich (Abbildung 7, links) für den grafischen Programmcode. Ein LabVIEW Programm wird als virtuelles Instrument (VI) bezeichnet. Die Verschachtelung von VIs ineinander ist ebenfalls möglich. Wenn ein Programmteil innerhalb eines VIs in ein separates VI ausgelagert wird, spricht man von einem Sub-VI. Ähnlich wie bei der Verwendung von Funktionen in C können diese Sub-VIs nun an den benötigten Stellen im Programm immer wieder eingebaut werden. VIs können sowohl Eingänge als auch Ausgänge besitzen, über welche Informationen aufgenommen oder ausgegeben werden. Wenn eine Sub-VI keine Eingänge besitzt,

wird sie nach dem Start des Hauptprogramms bzw. der Haupt VI sofort ausgeführt (es sei denn der Start wird durch andere Mechanismen explizit unterbunden).
Das Datenfluss-Modell ermöglicht es ungeübten Personen schnell in ein Verständnis für nicht selbst erzeugte Programme zu entwickeln. Jeder „Verbindungsstrich“ steht im Grunde für eine Variable, welche von einem Befehl zum anderen „weitergegeben“ wird. Der in LabVIEW generierte Programmcode wird immer von links nach rechts durchlaufen. Funktionsblöcke werden nur dann ausgeführt, wenn alle benötigten Variablen zur Verfügung stehen. Dies ähnelt stark einem Flussdiagramm, was für einen Maschinenbauingenieur i.d.R. leichter zu verstehen ist, als in Zeilen geschriebener Code. Aufgrund dieser leichten Verständlichkeit und den schnellen Zugang eignet sich LabVIEW besonders für den Einsatz im Rahmen eines Labors. Auf eine genaue Beschreibung von Funktionen, Schleifen und den Programmcode wird an dieser Stelle verzichtet. Genauere Erläuterungen folgen bei der Vorstellung der Laboraufgaben und dem vorgefertigten Code. Für eine gute Einführung in LabVIEW soll hier nur auf einige Quellen verwiesen werden. Das Buch *Einführung in LabVIEW* von Wolfgang Georgi und Ergun Metin [09] eignet sich sehr gut für den Einstieg. Es umfasst sowohl Grundlagenwissen, als auch erweiterte Programmierhinweise für fortgeschrittene Studierende. Das Buch ist in der Bibliothek der HAW verfügbar. Von NI wird ebenfalls ein Einführungskurs angeboten, dieser setzt jedoch die Verwendung bestimmter Hardware voraus [17]. Generell lässt sich LabVIEW auch ohne jegliche Hilfsmittel erlernen, da die angeschlossene Hilfe sehr umfangreich ist. Eine Aktivierung der *Kontexthilfe* (ein Klick auf das gelbe Fragezeichen im Frontpanel oder Blockdiagramm) wird in jedem Fall empfohlen.

3.2 Vorstellung der Hardware

Zu der eben vorgestellten Software LabVIEW wird von NI eine Vielzahl von Hardware zur Verfügung gestellt. Die Auswahl erstreckt sich von einfachen Geräten mit wenigen Schnittstellen, bis hin zu modularen Echtzeitsystemen. Für das Mechatroniklabor ist die Wahl schließlich auf eines der modularen Echtzeitsysteme gefallen. Die Gründe hierfür sind unterschiedlich. Zum einen wurde, bedingt durch die Anforderungen aus der Industrie, ein Echtzeitsystem gewählt, zum anderen wurde, auf Grund der Vielfältigkeit eines Labors, ein modulares System gefordert.

Die Forderung aus der Industrie erklärt sich durch eine einfache Betrachtung der aktuellen Anforderungen vieler technische Prozesse. Diese arbeiten unter harten Zeitbedingungen. Bei Prozessautomatisierungen mit hohen Taktraten, wie der Herstellung von Medikamenten, oder dem Bestücken von Platinen. Hier kommt es nicht nur auf Schnelligkeit, sondern auch auf eine hohe Präzision an. Besonders bei Robotern mit mehrachsigen Antrieben ist eine schnelle und genaue Steuerung notwendig. Solche Systeme werden dann von so genannten Echtzeitsystemen gesteuert oder geregelt. „Bei Echtzeitsystemen ist neben der Korrektheit der Ergebnisse genauso wichtig, dass Zeitbedingungen erfüllt werden.“ (H. Wörn, U. Brinkschulte, Echtzeitsysteme) [28] Der Begriff Echtzeitsystem ist in einigen Bereichen aber auch leicht irreführend. So kann auch ein Prozess, welcher erst nach fünf Minuten ein erneutes Steuersignal benötigt, durch eine als Echtzeitsystem bezeichnete Hardware gesteuert werden. Es ist lediglich erforderlich, dass diese Hardware das korrekte Signal vor Ablauf der fünf Minuten liefert. Im Labor soll das Verständnis für ein Echtzeitsystem deshalb unbedingt vermittelt werden. Da, wie in der Beschreibung zum Laborkonzept schon erwähnt, die Versuche möglichst Modular aufgebaut werden sollen, hat dies natürlich auch einen Einfluss auf die Auswahl der Hardware.

Abb. 8 Die für diese Arbeit verwendete Hardware von NI. Zu sehen sind der Controller (1), das Chassis (2) und die Module.

Den Grundstein der Hardwareauswahl bildet der Controller NI cRIO-9022 (siehe Abbildung 8, 1). Dieser arbeitet mit einem echtzeitfähigen Betriebssystem und wird auch in der Industrie zum Steuern von Anlagen und Prozessen verwendet. Zu dem Controller gehört das Chassis NI cRIO-9114 (Abbildung 8, 2), welches über einen hardwareprogrammierbaren Chip und acht Steckplätze für Erweiterungsmodule

verfügt. In Abbildung 8 sind ebenfalls die für diese Arbeit verwendeten Module zu sehen (von links nach rechts):

- NI-9421: digitales Eingangsmodul,
- NI-9472: digitales Ausgangsmodul,
- NI-9215: analoges Eingangsmodul,
- NI-9263: analoges Ausgangsmodul und
- NI-9403: digitales Ein-/Ausgangs-, TTL-Modul.

Nachfolgend werden der Controller und das Chassis gesondert vorgestellt. Dabei sollen vor allem die Funktionalität des hardwareprogrammierbaren Chips und die Unterschiede zum Controller erläutert werden. Denn dieser trägt maßgeblich für die Echtzeitfähigkeit des Systems bei.
Das CompactRIO System versteht sich als offene *Embedded-Architektur,* welche mit unterschiedlichsten Modulen bestückt werden kann, um für verschiedenste Zwecke eingesetzt werden zu können. Neben einem CPU und dem echtzeitfähigen Betriebssystem *VxWorks* besitzt das Chassis des CompactRIO auch einen *Field Programmable Gate Array*-Chip (FPGA). So ist es einerseits möglich Programme auf dem Controller laufen zu lassen oder direkt die FPGA zu konfigurieren und die Vorteile dieser zu nutzen.
Ein eingebettetes System (embedded system) bezeichnet einen elektronischen Rechner, der in einen technischen Kontext eingebunden ist. Der Rechner übernimmt dabei meist überwachende, steuernde oder regelnde Funktionen. Er kann ebenso weitere Funktionen, wie das Erfassen von Daten oder das Verarbeiten von Signalen erfüllen. Solche eingebetteten Systeme finden sich in fast jedem technischen Gerät, mal mit mehr und mal mit weniger komplexen Aufgaben. Die Steuerung einer Spülmaschine wird, genauso wie die Regelung der Kraftstoffzufuhr eines Verbrennungsmotors, von einem eingebetteten System übernommen. In komplexeren Gesamtsystemen werden viele einzelne, sonst autonom arbeitende eingebettete Systeme, vernetzt. Hierfür werden in der Industrie meist Bus-Systeme wie CAN- oder Profi-Bus verwendet. Auch das CompactRIO System kann als ein solches eingebettetes System verwendet und per Bus-Schnittstelle mit anderen Systemen vernetzt werden.

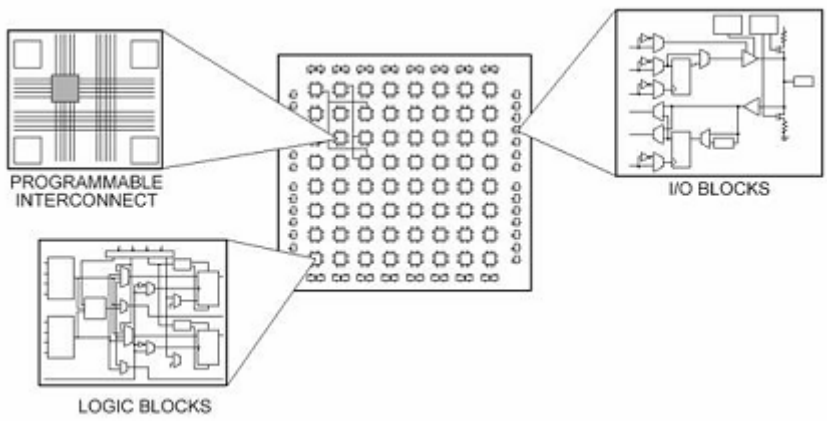

Abb. 9 Schematische Darstellung eines FPGA. Abgebildet sind die Logikblöcke, die Eingangs-/Ausgangs-Blöcke (I/O Blocks) und die programmierbaren Verbindungen zwischen den Logikblöcken [19].

Mit *Field Programmable Gate Array* (Anwendungs-, Feld-programmierbare Logik-Gatter-Anordnung) (FPGA) benennt man einen integrierten Schaltkreis (IC) auf dem logische Schaltungen programmiert werden können. Abbildung 9 zeigt schematisch ein FPGA, wie es auch in dem cRIO-Chassis verbaut ist. Für die FPGA können in LabVIEW spezielle Programme geschrieben werden. Diese werden dann mit Hilfe von Kompilierwerkzeugen auf die FPGA im cRIO-Chassis geschrieben. Wobei die Verbindungen, in dem geschriebenen Blockdiagramm des Programms, tatsächlich als Hardware-Verbindungen konfiguriert werden.

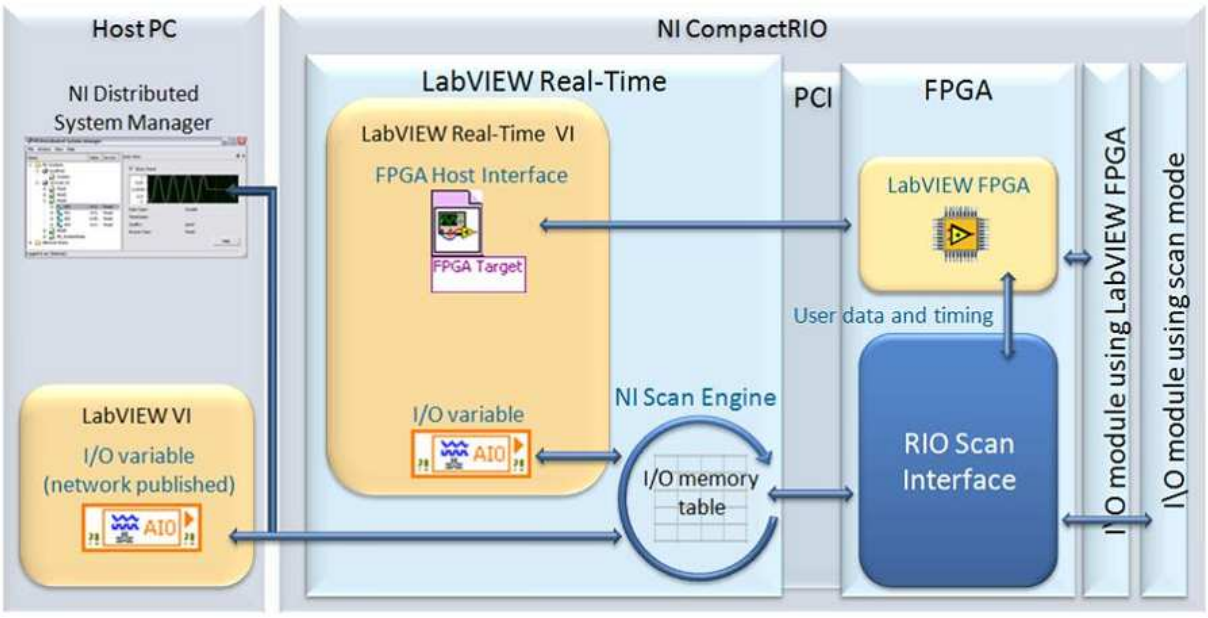

Abb. 10 Darstellung der Funktionsweise des CompactRIO Systems [18].

Abbildung 10 zeigt vereinfacht die Funktionsweise des CompactRIO Systems. Zu sehen ist der *Host PC* und das als *NI CompactRIO* benannte System, bestehend aus dem Controller (*LabVIEW Real-Time*) und der *FPGA*. Im Feld des Host PC ist der für

die Einrichtung des cRIO-Systems und zum Auslesen, sowie manuellem Einstellen von Ausgängen verwendete *NI Distributed System Manager* abgebildet. Darunter ist eine Netzwerkvariable zu sehen, mit welcher der Host PC mit der auf dem Controller laufenden *NI Scan-Engine* kommuniziert. Die Scan-Engine arbeitet ähnlich wie eine SPS mit einer eingestellten festen Periode (Scan-Periode). Zu Beginn jedes Durchlaufs wird ein Abbild der I/O Kanäle erstellt, mit welchem dann innerhalb des Programms auf dem Controller gearbeitet werden kann (siehe Abbildung 10, *LabVIEW Real-Time VI*). Mehr zur Scan-Engine und der Scan-Periode folgt in den Laborversuchen 1 und 2.
Der Controller ist über einen PCI-Bus mit der FPGA im Chassis des cRIO-Systems verbunden. Die *NI Scan-Engine* greift über die FPGA auf die Module im Chassis zu. Die FPGA ist auch dazu in der Lage selber auf die Anschlüsse der Module zuzugreifen.
Das CompactRIO System kann entweder Programme auf dem Controller ausführen oder per FPGA konfiguriert werden. Hier ergeben sich viele Vor und Nachteile für das jeweilige Verfahren. Deshalb muss, je nach Anwendung, entschieden werden, ob eine Konfigurierung der FPGA notwendig ist, oder ob die Möglichkeiten der CPU erforderlich sind. Einige dieser Unterschiede werden anschließend betrachtet und auf ihre Bedeutung hin bewertet.

Taktrate

Der Prozessor im Controller basiert auf der *PowerPC* Architektur und arbeitet mit einer Taktrate von 533 MHz (NI cRIO-9022). Dieser Prozessor zeichnet sich durch seine Echtzeitfähigkeit und die gute Leistungsfähigkeit bei niedrigem Stromverbrauch aus. Trotz der Taktrate von 533 MHz heißt dies nicht, dass auch Programme mit dieser Rate ausgeführt werden können. Softwareseitig sind Taktraten von bis zu 1 MHz einstellbar, allerdings konnte noch kein lauffähiges Programm, ohne Absturz, mit dieser Taktrate ausgeführt werden. Die effektive Taktrate mit der Programme auf dem Controller ausgeführt werden können beträgt ca. 2 kHz. Natürlich je nach Komplexität des Programms auch weniger. Eine einfache Möglichkeit die maximale Taktrate zu ermitteln wird später in Laborversuch 1 im Kapitel 4.1 vorgestellt.
Aus Abbildung 10 wird deutlich, dass bei der Verwendung des Controllers, immer über die FPGA auf die Anschlüsse der Module zugegriffen werden muss. Dies kann mit der direkten Programmierung der FPGA umgangen werden. Die FPGA arbeitet

mit einer festen Taktrate von 40 MHz. Für einzelne Schleifen sind auch Taktraten von bis zu 200 MHz einstellbar. Der größte Unterschied zum Controller besteht darin, dass diese Taktraten auch tatsächlich erreicht werden. In Hinblick auf die Taktrate würde man sich fragen, warum die Verwendung des Controllers überhaupt Sinn macht und nicht immer die FPGA benutzt werden soll. Dies erklärt sich bei der Betrachtung der möglichen Funktionen, welche auf dem Controller und der FPGA ausgeführt werden können.

Komplexität der Programme

Einer VI die auf dem Controller laufen wird, steht der volle Umfang der Funktionsblöcke von LabVIEW zur Verfügung. Diese VI wird dann, je nach getroffenen Einstellungen, ausgeführt und läuft unabhängig auf dem Controller weiter. So kann das zuvor erwähnte *Embedded*-Prinzip umgesetzt werden.
Für ein Programm, welches auf der FPGA konfiguriert werden soll, stehen nur sehr eingeschränkte Funktionen zur Verfügung. Da hier tatsächliche Hardware konfiguriert wird, müssen z.B. Vektoren immer eine festgelegte Größe haben, ein automatisches Anpassen ist nicht möglich. Diese Einschränkung soll nicht bedeuten, dass grundlegende Funktionen wie eine PID-Regelung oder eine FFT nicht verfügbar wären. Lediglich erweiterte Funktionen wie eine umfangreiche Datenverwaltung oder komplexe String-Verarbeitungen sind nicht verfügbar. Durch genaue Planung kann das Fehlen an Funktionen im FPGA Programm leicht umgangen werden. Für komplexe Funktionen kann ein Programmteil auf dem Controller laufen, während der Andere auf der FPGA ausgeführt wird (siehe Abbildung 10).
Während das Laden eines Programms auf den Controller in wenigen Sekunden abgeschlossen ist, muss ein Programm für die FPGA zuerst von einem Compiler übersetzt werden und anschließend konfiguriert werden. Dieser Vorgang kann je nach Größe des Programms mehrere Minuten in Anspruch nehmen. Genauso wie Programme auf dem Controller können auch Programme auf der FPGA unabhängig laufen.
Trotz dieser Nachteile erweist sich die Benutzung der FPGA, nach einer entsprechenden Anpassung der Programme, als außerordentlich sinnvoll. Denn in vielen technischen Prozessen müssen eingehende Signale schnell in neue Steuerbefehlt umgewandelt werden. Dabei handelt es sich meist um einfache Berechnungen, welche problemlos auf der FPGA durchgeführt werden können. Alle

weiteren Funktionen, wie die Prozessdatenspeicherung oder eine Aktualisierung der Sollwerte für Steuerungen, können auf dem Controller mit niedrigeren Taktraten durchgeführt werden.
Damit die Vorteile des Controllers (voller Umfang der Funktionspalette) und der FPGA (schnelle Taktrate) gemeinsam genutzt werden können stehen einige Methoden für die Kommunikation der beiden Systeme untereinander zur Verfügung. Im Laborversuch 3 in Kapitel 4.5 wird das Zusammenspiel von Controller, FPGA und einem Steuerungs-PC (Host-PC) genauer vorgestellt. Im Laborversuch 9 in Kapitel 6.5 wird die Ansteuerung eines Schrittmotors ebenfalls mit der FPGA realisiert.

4 Labor zur Informationsverarbeitung

Die Informationsverarbeitung ist in einem mechatronischen System für die Verarbeitung von Eingangssignalen, die Berechnung von Steuersignalen und deren Leitung zu den Aktoren verantwortlich.
In diesem Labor soll es vorrangig darum gehen, die in Kapitel 2.3 herausgearbeitete theoretischen Grundlagen mit Hilfe von praktischen Versuchen zu verdeutlichen. Für die Erstellung der Versuche wurde, wie zuvor schon beschrieben, Hard- und Software von NI verwendet. Die hier verwendete Hardware von NI wird, wie bereits im Kapitel 3.2 erwähnt, in der Industrie als eingebettetes System in Versuchsständen oder komplexen technischen Anlagen verwendet. In diesem Labor soll den Studierenden vermittelt werden, was die Programmierung eines solchen eingebetteten Systems ausmacht und wo die Leistungsgrenzen der verwendeten Hardware liegen.
In den Versuchen 1 und 2 werden vorbereitend für Versuch 3 Grundlagen der LabVIEW-Programmierung abverlangt. Diese Grundlagen werden auch in den späteren Versuchen benötigt. Außerdem sollen die Studierenden in diesen Versuchen ein Verständnis für die Module von NI bekommen und den richtigen Umgang mit diesen erlernen.
Im Versuch 3 wird die Hardware entsprechend industrieller Standards programmiert und die von NI vorgegebenen Schnittstellen für die Kommunikation zwischen den Programmteilen verwendet. Ein solches Vorgehen führt je nach Komplexität der gestellten Aufgabe, zu einem gewissen Zeitaufwand bei der Entwicklung und Kompilierung der Programmteile. Deshalb wird nur in Versuch 3 die komplette Kette der Programmierung des cRIO-Systems gezeigt und gefordert. In allen späteren Versuchen wird auf diesen zeitaufwändigen Schritt verzichtet. In den folgenden drei Laborversuchen soll ebenfalls der Umgang mit dem digitalen Oszilloskop (siehe Anhang 10.1) erlernet werden.

4.1 Versuch 1: Ermittlung der „Scan-Periode“ und deren Grenzen

Aufgabenstellung:

In diesem Versuch soll ermittelt werden, mit welcher Taktrate (Scan-Periode) die Scan-Engine auf dem cRIO-Controller arbeitet und ab welcher Taktrate die Bearbeitung nicht mehr fehlerfrei erfolgt.
Zunächst soll festgestellt werden, welche Periode aktuell eingestellt ist. Hierfür soll ein einfaches Programm entworfen werden, welches nach Ablauf jedes Takts einen der Ausgänge an dem digitalen Ausgangsmodul NI-9472 Ein-, bzw. Ausschaltet (true/false). Das so entstehende Signal soll mit Hilfe eines Oszilloskops betrachtet und die Periode abgelesen werden. Anschließend soll die Scan-Periode langsam gesteigert werden, bis die Bearbeitung des Programms nicht mehr durchgeführt werden kann.

1. Bis zu welcher Frequenz bearbeitet die Scan-Engine das Programm fehlerfrei?
2. Welche Komponenten spielen bei dieser Untersuchung noch eine wichtige Rolle?

Hinweise:
Falls der Controller nicht mehr reagieren sollte, kann auf dem Kontroller der RESET Knopf betätigt werden. Nachdem die Verbindung mit LabVIEW wieder hergestellt ist kann ganz normal weiter getestet werden.

Grundsätzliches zur Programmierung des cRIO-Systems

Für die Bearbeitung der Labore ist es sinnvoll, für jeden Versuch ein neues LabVIEW-Projekt anzulegen. In dieses Projekt muss stets das CompactRIO-System, als neues Gerät eingefügt werden. Die anschließende Auswahl, nach welcher Methode (Scan-Engine/FPGA) das cRIO-System programmiert werden soll, richtet sich nach der jeweiligen Aufgabenstellung.
Für die Durchführung der Aufgabe empfiehlt sich die Verwendung einer zeitgesteuerten Schleife (siehe Abbildung 11). Diese kann auf dem Controller unter verschiedenen Einstellungen ausgeführt werden.

Abb. 11 Eine zeitgesteuerte Schleife.

Um zur Zeitsteuerung der Schleife zu gelangen, kann entweder der Eingangsknoten (Abbildung 11, 1) doppelt angeklickt werden, oder die Option *Eingangskonten konfigurieren* mit einem Rechtsklick auf diesen ausgewählt werden. Weitere Informationen zu zeitgesteuerten Schleife sind in der LabVIEW Hilfe zu finden. In dieser Aufgabe soll nur die Zeitsteuerung der Schleife verwendet werden.
Die Scan-Periode soll in dieser Aufgabe in den Einstellungen des Controllers verändert werden.

4.2 Musterlösung Versuch 1

Die Aufgabe wurde mit Hilfe einer zeitgesteuerten Schleife mit einem Schieberegister erfüllt. Bei jedem Durchlauf der Schleife wird die boolesche Variable aus dem Schieberegister ausgelesen, negiert und in den digitalen Ausgang und das Schieberegister geschrieben (siehe Abbildung 12).

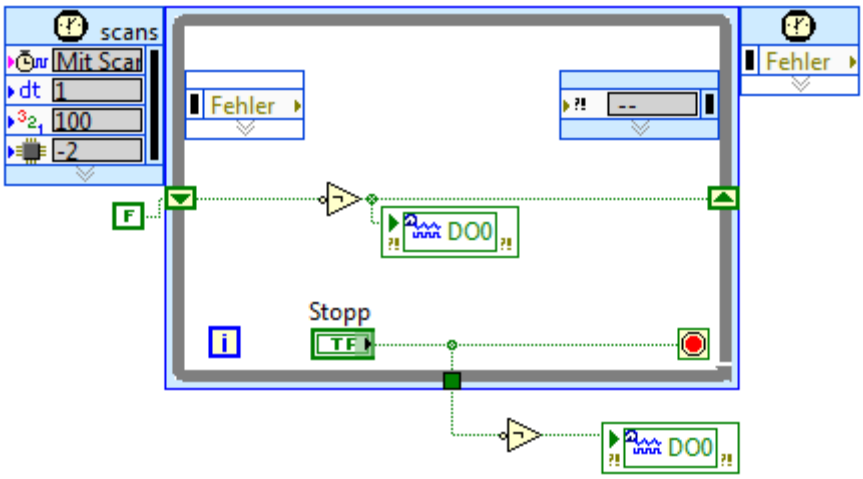

Abb. 12 Blockdiagramm des Programms für den Versuch 1.

Die zeitgesteuerte Schleife muss so eingestellt werden, dass sie synchron mit der Scan-Engine läuft. Zum Beenden der Schleife wird der Stopp-Schalter betätigt. Das erzeugte Signal wurde aus der Schleife negiert und in den digitalen Ausgang geschrieben. So ist sichergestellt, dass der Ausgang nach Abschluss des Programms ausgeschaltet ist. Nun kann zwischen dem digitalen Ausgang *DO0* und *COM* am NI-9472 Modul das erzeugte Signal gemessen werden. Auf dem Oszilloskop sollte sich ein ähnliches Bild wie in Abbildung 13, a zeigen. Damit sich ein Rechtecksignal ergibt wird eine Last benötigt. Dazu wird der Widerstand zwischen die Anschlüsse *DO0* und *COM* geschalten. Nun kann mit dem Oszilloskop genau gemessen werden (siehe Abbildung 13.b).

Für den zweiten Teil der Aufgabe wird die Scan-Periode manuell verstellt. Dies geschieht im Eigenschaftsfenster des Real-Time Controllers. Standardmäßig sollte eine Scan-Periode von 10 ms eingestellt sein. Was den Studierenden hier auffallen soll ist z.B. der Warnhinweis beim Umstellen auf den µs-Bereich. Auch wenn an vielen Stellen beworben wird, dass die Scan-Engine Programme mit 1 MHz ausführen kann, so deutet doch schon der Warnhinweis auf das Gegenteil hin.

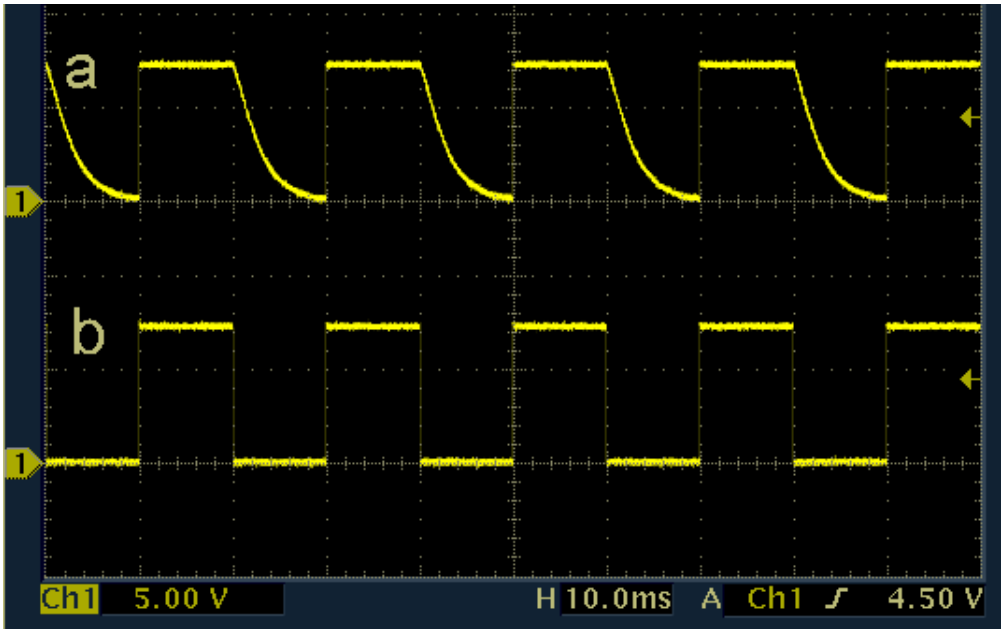

Abb. 13 Das erzeugte Signal am digitalen Ausgang: (a) ohne und (b) mit zwischengeschaltetem Widerstand.

In eigenen Versuchen konnte festgestellt werden, dass die Scan-Engine bis zu einer Periode von 300 µs, wenn auch mit Einschränkungen, noch arbeitet. Auf dem Oszilloskop war ein unregelmäßiges Flackern zu sehen, was auf ein nicht konstantes Signal hinweist.

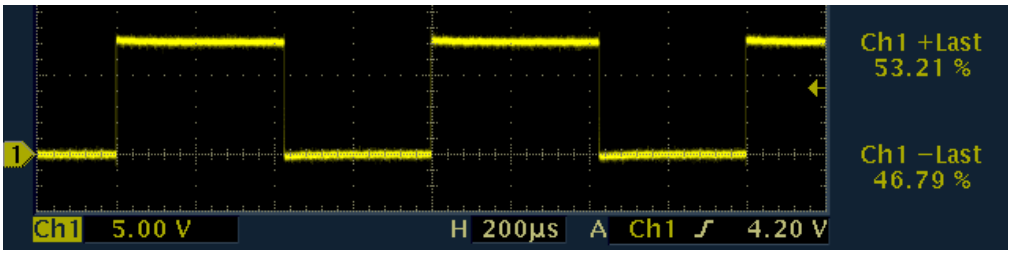

Abb. 14 Das erzeugte Signal bei einer Scan-Periode von 400 µs. Zu sehen ist ebenfalls eine Messung der positiven und negativen Signalanteile.

Was den Studierenden ebenfalls auffallen sollte, ist die ungleiche Länge der positiven und negativen Signale. Dieser Unterschied kann mit Hilfe des Oszilloskops gemessen werden. In der Abbildung 14 ist die durchgeführte Messung an der rechten Seite zu sehen. Bei einer eingestellten Scan-Periode von 400 µs ist zu erkennen, das der positive Anteil des Signals 53,21 % und der negative (in diesem Fall Null) Anteil 46,79 % von zwei Scan-Perioden ausmachen. Das positive Signal ist also um den Anteil von 3,21 % länger als erwartet. Für die Gesamtdauer von 800 µs ergibt sich eine Zeitdifferenz von:

$$\Delta t = 800\ \mu s\ \cdot 0{,}0321 = 25{,}68\ \mu s \qquad (4.1)$$

Die gemessene Zeitdifferenz zeigt sich nur beim Ausschalten des Signals. Aus dem Datenblatt des NI-9472 geht hervor das die interne Elektronik bei voller Last eine Schaltverzögerung von bis zu maximal 100 µs aufweisen kann.
Als Gründe hierfür können die verschiedenen Bauteile innerhalb des Moduls genannt werden. Für die Absicherung der Hardware wurden z.B. Optokoppler verbaut. Diese weisen eine maximale Schaltzeit von 18 µs auf. Dazu kommen weitere Zeitverzögerungen durch Transistoren und andere Bauteile.
Dieses Schaltverhalten des Moduls muss im Betrieb nicht unbedingt zu Beeinträchtigungen führen. Jedoch sollten solche Kenndaten bei der Auswahl von Modulen immer eine Rolle spielen und im Vorfeld abgesichert werden.

4.3 Versuch 2: Untersuchung eines externen Signals

Aufgabenstellung:

In diesem Versuch soll mit Hilfe des cRIO-Systems ein Signal aus einem Funktionsgenerator betrachtet werden. Dafür wird das analoge Eingangsmodul NI-9215 benutzt. Der Controller soll wieder mit der Scan-Engine benutzt werden.
Das Signal soll gleichzeitig mit einem Oszilloskop betrachtet werden, damit eine objektive Einschätzung über die Qualität des gemessenen Signals erstellt werden kann.
Damit der Einfluss der Scan-Periode deutlicher zu beurteilen ist, soll ein Programmteil geschrieben werden, mit dem diese während des Betriebes geändert werden kann. Die erforderlichen Funktionsblöcke werden vorgestellt. Das Programm soll auf dem Host-PC laufen.

Zusätzlicher Aufgabenteil:

Das eingelesene Signal soll über das cRIO-System an einen analogen Ausgang weitergeleitet werden. Hierfür wird das analoges Ausgangsmodul NI-9263 zum Erzeugen des Signals verwendet. Die beiden Signale (aus dem Funktionsgenerator und aus dem analogen Ausgangsmodul) sollen auf einem Oszilloskop betrachtet werden. Die Qualität sowie die Phasenverschiebung sind zu bestimmen.
Das Programm auf dem cRIO-Controller soll gleichzeitig mit dem Programm auf dem Host-PC beendet werden.

Programmierhinweise:

Für die Einstellung der Scan-Periode soll in diesem Versuch nicht, wie zuvor der Weg, über die Einstellungen des Controllers gegangen werden. LabVIEW stellt für das Konfigurieren der Scan-Engine spezielle Funktionsblöcke zur Verfügung. In der Funktionspalette *Mess-I/O* findet sich in der Kategorie *NI Scan Engine* eine Auswahl an Funktionsblöcken zum Auslesen und Einstellen von Parametern der Scan-Engine. Für diese Aufgabe sind die beiden Funktionsblöcke *Scan-Engine-Periode ermitteln* und *Scan-Engine-Periode festlegen* interessant. Die Kontexthilfe der beiden Blöcke (siehe Abbildung 15 und 16) gibt an, welche Informationen benötigt werden, damit die jeweilige Funktion ausgeführt werden kann.

Da das Programm für den Host-PC geschrieben werden soll, muss an den Anschlüssen *Zieladresse* die IP-Adresse des cRIO-Systems eingestellt werden.

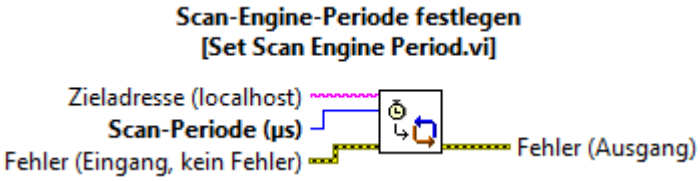

Abb. 15 Die Kontexthilfe für den Funktionsblock Scan-Engine-Periode festlegen.

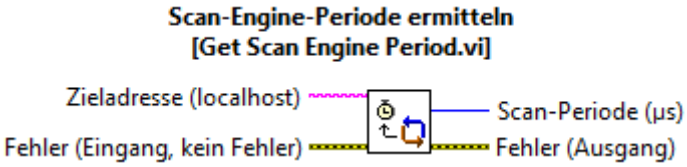

Abb. 16 Die Kontexthilfe für den Funktionsblock Scan-Engine-Periode ermitteln.

Zum gleichzeitigen Beenden des Host-Programms und des Controller-Programms sollte eine Netzwerkvariable verwendet werden. Wie beim Einbinden des cRIO-Systems in das Projekt kann eine Variable auf dem Controller erstellt werden. Für das Stoppen des Programms ist eine einfache boolesche Variable ausreichend.

4.4 Musterlösung Versuch 2

Für die Untersuchung des Signals aus dem Funktionsgenerator wurde ein analoges Eingangsmodul zur Verfügung gestellt. Die entsprechenden Anschlüsse können direkt aus dem Projektmanager in eine VI gezogen werden. So lässt sich das Programm zum Auslesen der Anschlüsse gemäß Abbildung 17 schnell erstellen.

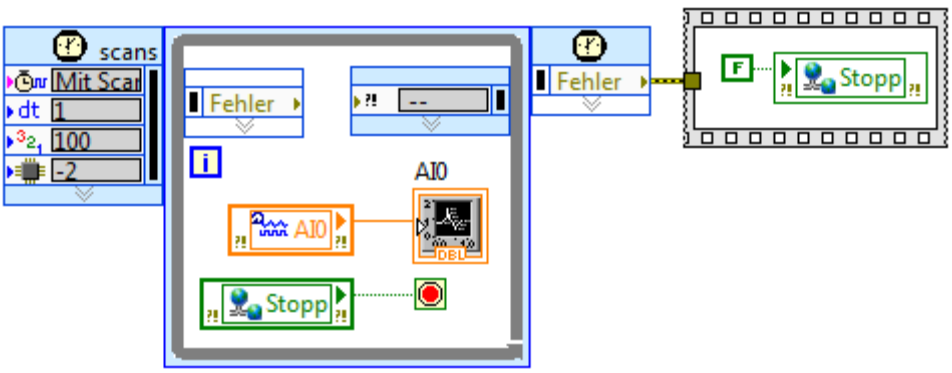

Abb. 17 Blockdiagramm des Programms auf dem Controller zum Darstellen des Signals.

Das Blockdiagramm des Programms ist in Abbildung 17 zu sehen. Der analoge Anschluss *AI0* wird innerhalb einer zeitgesteuerten Schleife ausgelesen und in einem Signalverlaufsdiagramm dargestellt. Die Schleife wird mit der Einstellung „mit Scan-Engine synchronisieren“ gesteuert. Zum Stoppen der Schleife wurde eine Netzwerkvariable Stopp verwendet. Diese muss nach beenden der Schleife wieder zurückgesetzt werden. Andernfalls würde das Programm beim nächsten Aufruf sofort wieder beendet werden.

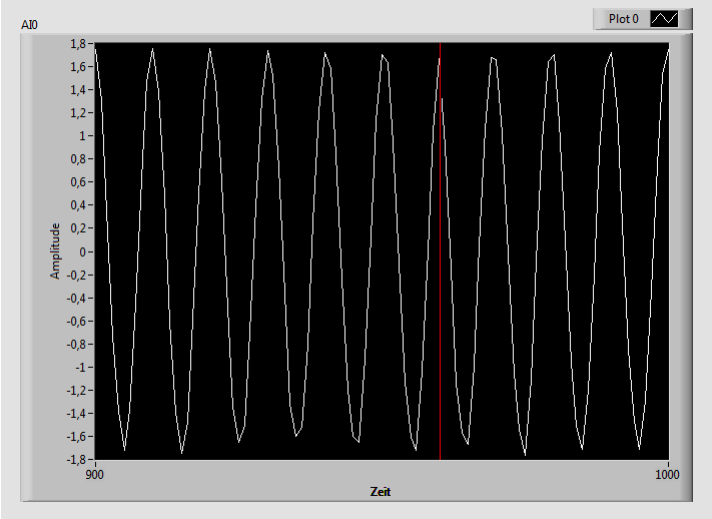

Abb. 18 Frontpanel des Programms auf dem Controller.

Auf dem Frontpanel des Programms (siehe Abbildung 18) findet sich das Signalverlaufsdiagramm. Hier sollten die Studierenden die verschiedenen Modi der Diagrammdarstellung ausprobieren und die geeignetste Einstellung für sich selber finden.

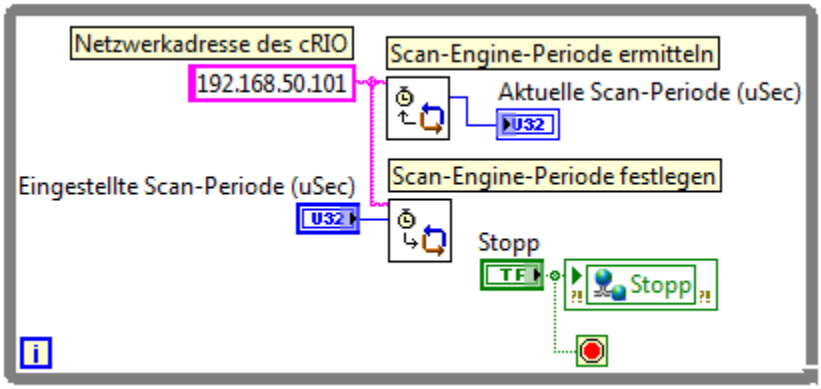

Abb. 19 Blockdiagramm des Host-Programms.

Für die Steuerung der Scan-Periode auf dem cRIO-Controller wird ein weiteres Programm geschrieben (siehe Abbildung 19). Dies läuft auf dem jeweiligen PC der

Studierenden (Host-PC). Es bietet sich wieder die Verwendung einer While-Schleife an. Zum Auslesen und Festlegen der Scan-Periode werden die beiden entsprechenden Funktionsblöcke verwendet. An dieser Stelle ist darauf zu achten, dass die Scan-Periode in µSec sowohl ausgelesen, als auch eingestellt wird. Auf dem Frontpanel (siehe Abbildung 20) kann nach dem Start des Host-Programms und dem Ausführen des Programms auf dem Controller die aktuelle Scan-Periode abgelesen und eine neue eingestellt werden.

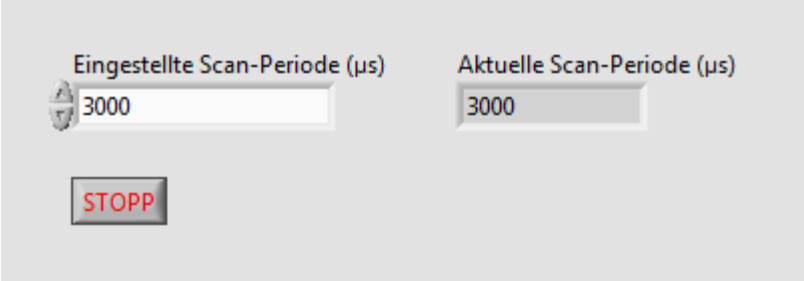

Abb. 20 Frontpanel des Host-Programms.

Für den zusätzlichen Aufgabenteil wurde das Programm auf dem Controller etwas angepasst. Das resultierende Blockdiagramm ist in Abbildung 21 zu sehen. Das Signalverlaufsdiagramm wurde entfernt. Stattdessen wird das eingelesene Signal *AI0* nun direkt an den analogen Ausgang *AO0* weitergegeben. Aus Sicherheitsgründen ist es sehr ratsam den analogen Ausgang nach beenden des Programms wieder auf den Wert Null zu setzten.

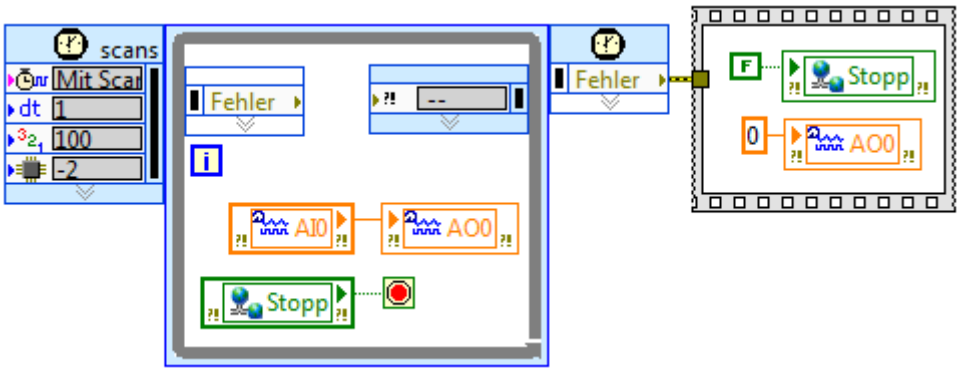

Abb. 21 Das angepasst Blockdiagramm des Controller-Programms.

Mit dem Oszilloskop kann nun das Signal aus dem Funktionsgenerator, mit dem über das cRIO-System weitergeleiteten, verglichen werden. Folgend werden einige interessant erscheinende Beispiele dargestellt.

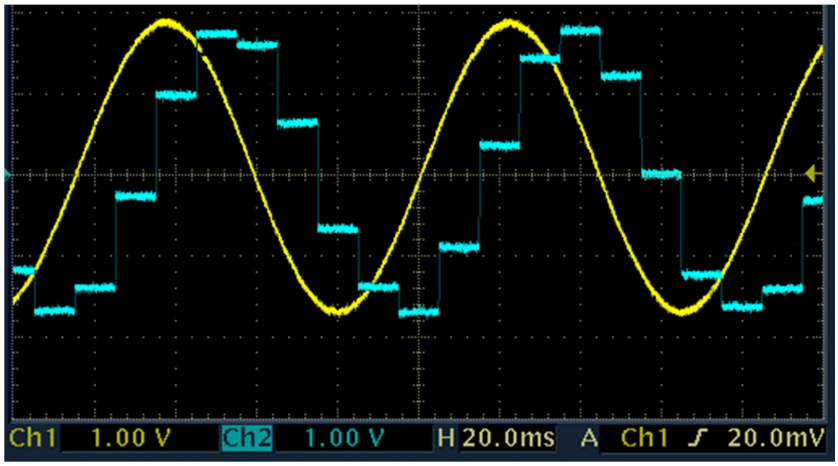

Abb. 22 Das Signal aus dem Funktionsgenerator (gelb) und das erzeugte Signal des cRIO-Systems (hellblau) bei einer Scan-Periode von 10 ms.

In Abbildung 22 ist ein im Funktionsgenerator erzeugtes Sinussignal (gelb) und das weitergeleitete Signal des cRIO-Systems (hellblau) zu sehen. Das weitergeleitete Signal wurde bei einer Scan-Perioden-Einstellung von 10 ms erstellt. Es ist eine deutliche Phasenverschiebung und der typische stufenförmige Signalverlauf zu sehen. Die Länge der Stufen entspricht in etwa der Scan-Periode.

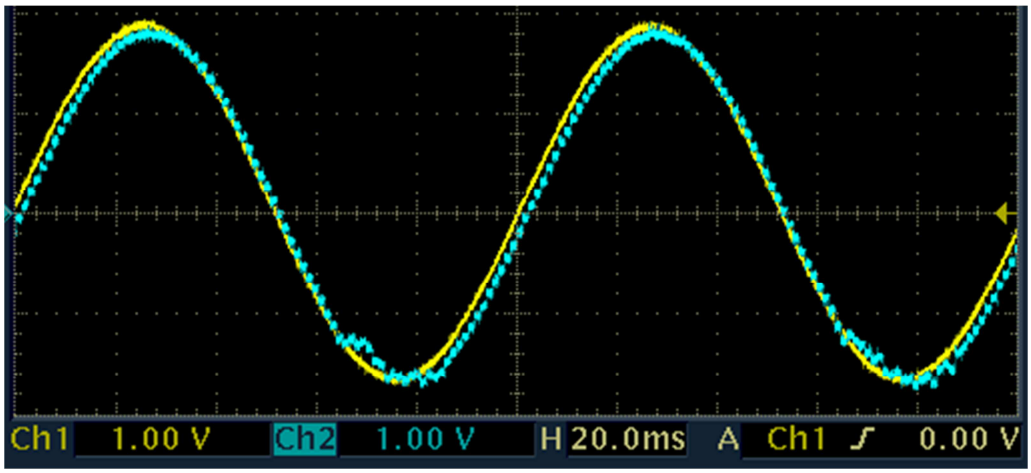

Abb. 23 Das Signal aus dem Funktionsgenerator (gelb) und das erzeugte Signal des cRIO-Systems (hellblau) bei einer Scan-Periode von 400µs.

Zum Vergleich ist in Abbildung 23 das gleiche Signal bei einer Scan-Periode von 400 µs abgebildet. Bei dieser Einstellung von 400 µs für die Scan-Periode würde man eine Phasenverschiebung um eben diese Zeit annehmen, das aber noch weitere Faktoren eine Rolle spielen zeigt sich in Abbildung 24. Am einfachsten lässt sich die Phasenverschiebung an einem Rechtecksignal abmessen (siehe Abbildung 24).

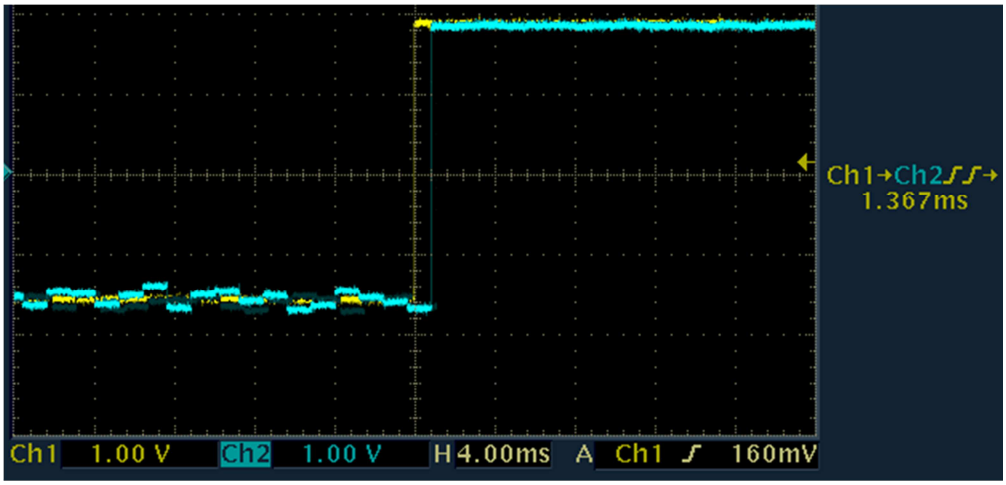

Abb. 24 Das Rechtecksignal aus dem Funktionsgenerator (gelb) und das erzeugte Signal des cRIO (hellblau) bei einer Scan-Periode von 400µs. Zusätzlich ist eine Messung von der ansteigenden Flanke (Signal 1) zur ansteigenden Flanke (Signal 2) zu sehen.

Hierfür wurde ein 10 Hz Signal mit Hilfe des Funktionsgenerators erzeugt (Abbildung 24, gelb). Bei einer Scan-Periode von 400 µs wurde ein entsprechendes Ausgangssignal am analogen Ausgang gemessen (Abbildung 24, hellblau). Mit dem Oszilloskop kann der Abstand zwischen den beiden ansteigenden Flanken entweder graphisch abgelesen, oder mit Hilfe der eingebauten Messmethoden ermittelt werden. In eigenen Versuchen ergab die Messung des Oszilloskops eine stark schwankende Phasenverschiebung von 0,9 bis 1,6 ms. In Abbildung 24 ist die durchgeführte Messung am rechten Rand zu erkennen.

Abgelesen werden kann eine Phasenverschiebung von rund 0,8 ms. Dies würde der Arbeitsweise der Scan-Engine und der maximal möglichen Verzögerung entsprechen. Wie in Kapitel 3.2 zur Scan-Engine beschrieben wurde, arbeitet diese ähnlich einer SPS mit Abbildern der Eingänge und schreibt, nach durchlaufen einer Periode, die neuen Werte an die Ausgänge. Sollte nun, kurz nach dem Ermitteln eines neuen Abbildes der Eingänge, sich diese sprunghaft ändern, wird die Änderung erst nach einer weiteren Periode festgestellt und nach noch einer weiteren Periode werden die Ausgänge entsprechend gesetzt. So könnte sich eine Verzögerung von zwei Perioden also 800 µs (0,8 ms) ergeben.

Der Unterschied zu den Messwerten des Oszilloskops lässt sich nur durch die Unregelmäßigkeit des Ausgangssignals des cRIO-Systems erklären. Es kommt hier offenbar zu einer fehlerhaften Erkennung der positiven Flanken.

4.5 Versuch 3: Konfiguration der FPGA und Kommunikation mit dem Host

Aufgabenstellung:

In diesem Versuch soll die FPGA des cRIO-Chassis konfiguriert werden. Hierfür werden die Funktionsweise der FPGA und mögliche Kommunikationswege zwischen dem Controller, der FPGA und dem Host-PC vorgestellt. Für eine einfache Aufgabe sollen die benötigten Programme für die FPGA, den Controller und den Host-PC geschrieben werden.
Die Aufgabe besteht wie in Versuch 1 darin, ein Programm zu schreiben, welches ein Rechtecksignal entsprechend einer eingestellten Frequenz ausgibt. Wie in Versuch 1 wird dazu ein digitaler Ausgang des NI-9472 Moduls verwendet, jedoch soll die Ausführung nun auf der FPGA des cRIO-Chassis stattfinden. Für die Steuerung der Frequenz soll ein Programm geschrieben werden, welches auf einem Host-PC ausgeführt wird. In diesem kann die Frequenz des Rechtecksignals eingestellt werden. Damit der eingestellte Wert an das cRIO-System übertragen wird, muss ein geeigneter Kommunikationskanal geöffnet werden.
Da es sinnvoll sein kann Berechnungen, Auswertungen von Messwerten oder Anpassungen von Stellwerten nicht auf der FPGA durchzuführen, sondern auf der CPU des Controllers, soll dies in dieser Aufgabe ebenfalls geschehen. Gründe weshalb eine solche Auslagerung von Berechnungen durchgeführt werden sollte, werden im Folgenden beschrieben. Auf dem Controller soll die Umrechnung der Frequenz stattfinden.
Das erzeugte Signal soll mit Hilfe eines Oszilloskops betrachtet werden und die maximale mögliche Umschaltzeit ermittelt werden.

Hintergrund zum Programmieren der FPGA:

In den vorherigen Versuchen war immer die Rede vom cRIO-System. Dabei handelt es sich um zwei Geräte. Einem Chassis NI cRIO-9114 und dem daran angeschlossenen Echtzeit-Controller NI cRIO-9022. Das NI cRIO-9114 ist ein rekonfigurierbares CompactRIO-Chassis mit 8 Steckplätzen für Module. Dieses Chassis beherbergt den so genannten FPGA. Eine genaue Beschreibung der Funktionsweise dieser FPGA ist dem Kapitel 3.2 zu entnehmen. Der Vorteil der Benutzung der FPGA ist die sehr schnelle Antwortzeit auf eintretende Signale von

nur rund 25 ns [19]. Der Nachteil ist die eingeschränkte Funktionalität, bestimmt durch den physikalischen Aufbau des Chips. In das Chassis werden die für die Aufgabe benötigten Module gesteckt.
Der Echtzeit-Controller NI cRIO-9022 besitzt einen 533 MHz CPU basierend auf der PowerPC Architektur. Zusammen mit dem Betriebssystems VxWorks ist dieser Controller in Echtzeitanwendungen einsetzbar.
NI stellt mit dem cRIO-System eine Kombination aus einer FPGA mit sehr schneller Reaktionszeit und einem Controller mit umfangreicher Funktionalität bereit. Damit dieses System optimal genutzt wird, muss schon vor der Programmierung sichergestellt werden, welche Funktionen auf welcher Plattform durchgeführt werden sollen. In diesem Versuch soll das cRIO-System eine recht einfache Aufgabe durchführen und die Überlegung welche Funktion auf welcher Plattform durchgeführt werden soll erübrigt sich, da die Funktionen vorgegeben wurden. In Aufgabenstellungen aus der Industrie muss diese Überlegung immer vor Beginn der Programmierung durchgeführt werden.

Das FPGA Programm:

Wie erwähnt soll die FPGA einen digitalen Ausgang am NI-9472 umschalten. Das Programm auf der FPGA wird ständig laufen, auch wenn der Host-PC abgeschaltet ist. Denn das cRIO-System ist für einen dauerhaften Einsatz in technischen Anlagen, oder für die Prozessüberwachung vorgesehen. Aus diesem Grund muss das Programm für die FPGA so gestaltet werden, dass beim Beenden des Host-Programms auch das Umschalten des Ausgangs beendet wird. Bei einem erneuten starten des Host-Programms soll die Funktionalität wieder hergestellt werden.
Als Grundlage empfiehlt sich das in Versuch 1 erstellte Programm. Durch den Einbau einer Case-Struktur (wahr/falsch) sollte kontrolliert werden können, ob der Ausgang gesetzt wird oder nicht.
Die Steuerung der Durchlaufzeit der Schleife soll hier mit der FPGA-exklusiven Funktion *Loop Timer* geschehen. Dieser soll in der Einstellung *Takte* benutzt werden.

Das Controller Programm:

Für den Controller soll ebenfalls ein Programm entwickelt werden, welches dauerhaft laufen soll. In diesem Programm wird die eingestellte Frequenz von Herz in Takte umgerechnet und an das FPGA Programm übergeben. Ebenso muss der FPGA

mitgeteilt werden, dass das Host-Programm ausgeführt wird und das Umschalten des Ausgangs wieder durchgeführt werden soll.

Für den Controller ist das in Abbildung 25 zu sehende Programm vorbereitet. Darin sind zwei While-Schleifen und einige vorbereitete Funktionen zu erkennen. Für die Kommunikation mit der FPGA wurde die Funktion *FPGA-VI-Referenz öffnen* (Abbildung 25, 1) eingebaut. Diese sorgt dafür, dass Werte in Bedienelemente des FPGA Programm geschrieben und aus Anzeigeelementen gelesen werden können. Hierzu muss noch die Funktion *Read/Write Control* aus der Funktionspalette *FPGA Interface* eingebaut werden.

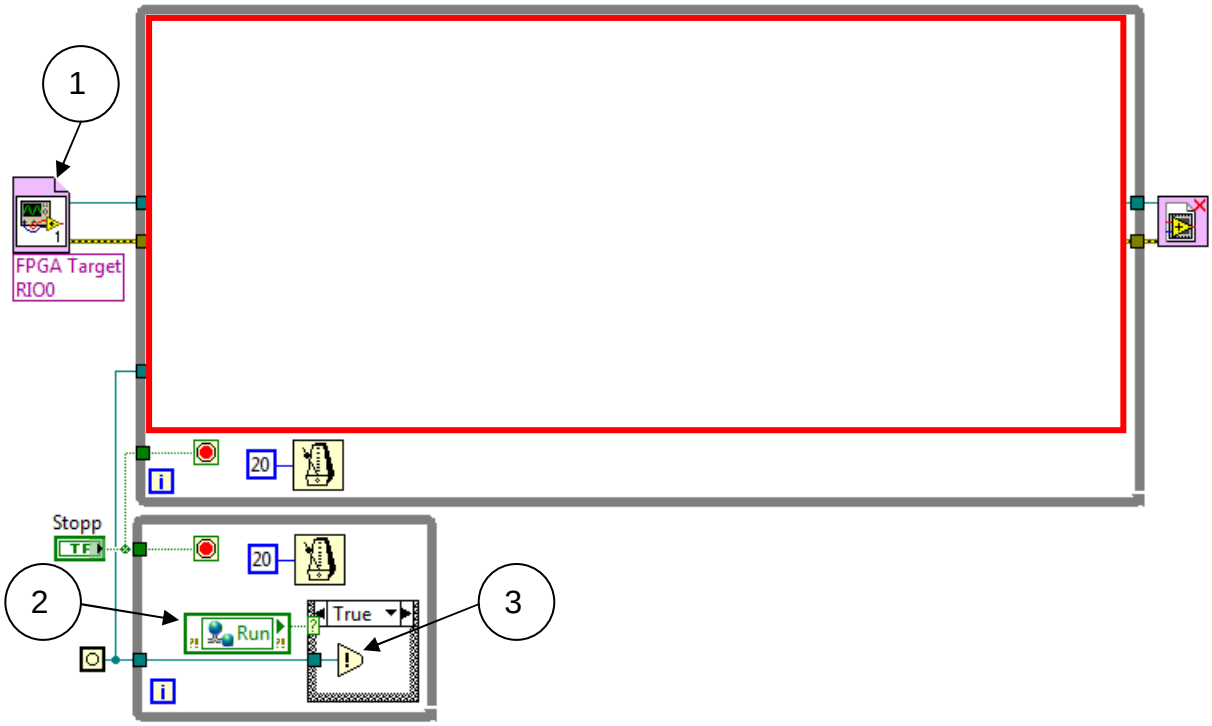

Abb. 25 Vorbereitetes Programm für den Controller.

Bei dem in Abbildung 25 mit einer 2 markierten Block handelt es sich um eine Netzwerkvariable. Diese liegt auf dem cRIO-Controller und kann vom Netzwerk aus mit Werten beschrieben oder ausgelesen werden. Die mit einer 3 markierte Funktion kann durch Verwendung von einer dazugehörigen Funktion, für die Steuerung der oberen Schleife genutzt werden. Genaue Informationen dazu sollen aus der LabVIEW Hilfe entnommen werden. Im dem roten Bereich in Abbildung 25 soll schließlich die Steuerung des FPGA-Programms realisiert werden. Hier noch einmal die geforderten Funktionen:

- Umrechnung der eingestellten Frequenz von Herz in Ticks

- Ein- und Ausschalten der Umschaltfunktion des FPGA-Programms
- Warten, sollte das Host-Programm nicht laufen

Das Host-Programm:

Auf dem Host-PC muss zuletzt noch ein Programm für die Eingabe der Frequenz geschrieben werden. Es ist ebenfalls zu überlegen wie dem Controller mittels der Netzwerkvariablen *Run* (siehe Abbildung 25, 2) mitgeteilt werden kann, dass das Host-Programm läuft bzw. beendet wurde.

4.6 Musterlösung Versuch 3

In diesem Versuch wurde gefordert, dass die FPGA des cRIO-Chassis konfiguriert werden soll. Hierfür ist es erforderlich, beim einbinden des cRIO-Systems in ein Projekt, darauf zu achten die entsprechende Option für die Betriebsart zu wählen.
Das Programm für die FPGA wurde gemäß den Vorgaben und mit derselben Funktionalität wie das Programm aus Versuch 1 geschrieben.

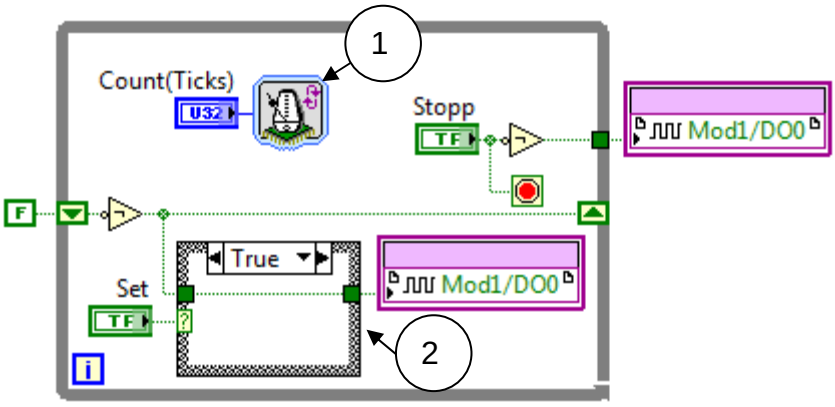

Abb. 26 Das Programm für das FPGA.

Der Aufbau des Programms ist in Abbildung 26 zu sehen. Es wird wie in Versuch 1 eine While-Schleife verwendet. Diesmal ist jedoch keine zeitgesteuerte While-Schleife benutzt worden. Für die Zeitsteuerung wurde, wie gefordert, die Funktion *Loop Timer* (Abbildung 26, 1) verwendet. Dieser wird über das Bedienelement Count(Ticks) gesteuert. Für das Ein- und Ausschalten der Umschaltfunktion wurde eine Case-Struktur (Abbildung 26, 2) eingebaut. Sollte das Bedienelement *Set*, *true* ausgeben, wird der digitale Ausgang bei jedem Durchlauf der Schleife umgeschaltet. Bei einer Ausgabe von *false* wird der Ausgang ebenfalls *false* gesetzt.

Für das Controller-Programm war ein Großteil vorgegeben. Dort waren auch wichtige Komponenten enthalten, welche für die Kommunikation zwischen den drei Programmen entscheidend sind. In Abbildung 27 ist das kompetierte Programm zu sehen. Die in der Aufgabenstellung erwähnte *Read/Write Control* ist mit einer 1 markiert. Dort können nun die Bedienelemente des FPGA-Programms ausgewählt und mit Werten beschrieben werden. Für die *Set* Variable, welche das Umschalten aktiviert, wurde die Netzwerkvariable *Run* verwendet. Eine weitere Netzwerkvariable Frequenz wurde für das Übertragen der eingestellten Frequenz hinzugefügt und die Umrechnung von Herz in Ticks programmiert. Da die FPGA mit 40 MHz taktet, wird die Anzahl an Takten pro Sekunde durch die Frequenz geteilt. Zwei Umschaltvorgänge bilden wiederrum die gesamte Periodendauer. Die Division hat eine Variable des Datentyps Double als Ergebnis, für die FPGA wird diese noch in ein vorzeichenloses Long (32-Bit Integer) umgewandelt.

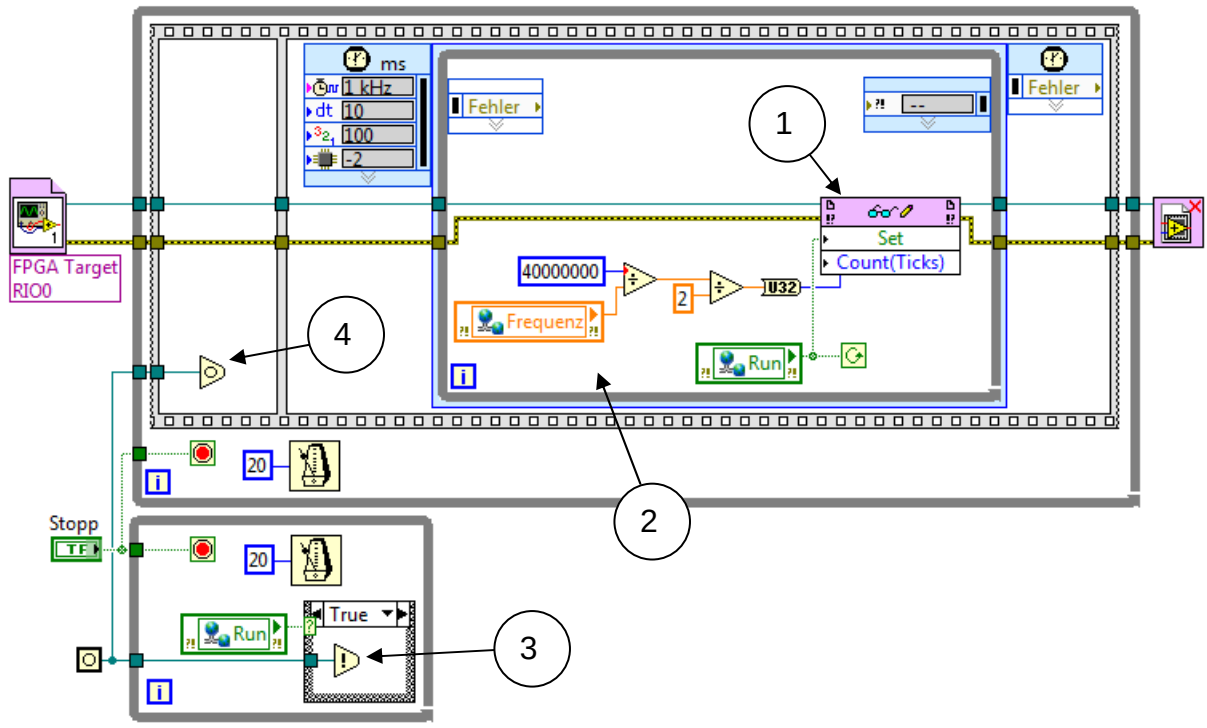

Abb. 27 Das Blockdiagramm des Controller-Programs.

Die *flache Sequenzstruktur,* sowie die Funktion *Auf Occurrence warten* (Abbildung 27, 4) sorgen dafür, dass das Programm beim Beenden des Host-Programms in einen Wartezustand wechselt.

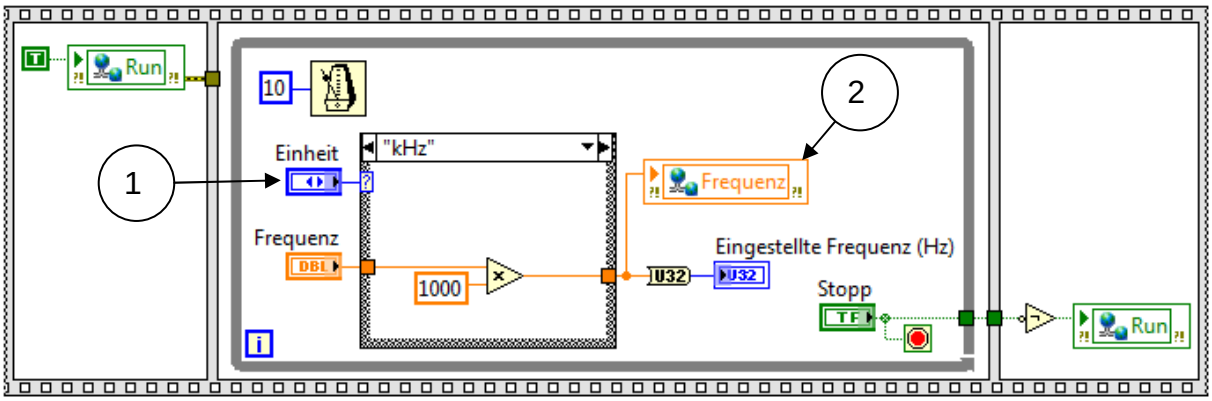

Abb. 28 Das Host-Programm zum Eingeben der Frequenz.

In Abbildung 28 ist das Programm des Host-PC zu sehen. Beim Start des Programms wird im ersten Rahmen einer flachen Sequenzstruktur die Netzwerkvariable *Run* auf *true* gesetzt, was im Controller-Programm zum Ausführen der *Occurrence festlegen* Funktion (Abbildung 27, 3) und dadurch zum Start der Schleife (siehe Abbildung 27, 2) führt. Im zweiten Rahmen der Sequenzstruktur befindet sich eine Schleife, in der die Frequenz eingestellt werden kann. Zusätzlich wurde eine Umschaltung von Hz in kHz und MHz in Form von angepassten Multiplikationen eingebaut (Abbildung 28, 1 und die Case-Struktur). Die so eingestellte Frequenz wird dann in die Netzwerkvariable *Frequenz* geschrieben (Abbildung 28, 2). In Abbildung 29 ist das zugehörige Frontpanel zu sehen. Die Auswahl des Multiplikators gescheit in dem Auswahlfeld 1.

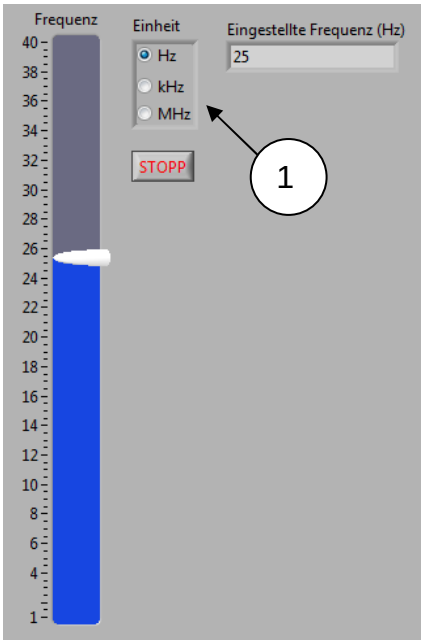

Abb. 29 Frontpanel des Host-Programms.

Bei der Auswertung des so erzeugten Signals mit einem Oszilloskop musste festgestellt werden, dass durch die Verwendung der FPGA keine schnellere Umschaltung des digitalen Ausgangs möglich ist. Dies hat den ganz einfachen Grund, dass das verwendete Modul kein schnelleres Umschalten zulässt. Die Gründe hierfür wurden im vorherigen Versuch bereits beschrieben. In Hinblick auf den in diesem Versuch deutlich gewordenen Unterschiede bei der Programmierung der FPGA und dem Controller ist es umso wichtiger im Vorfeld die Leistungsfähigkeit des gesamten Systems zu betrachten. Die Programmierung des Gesamtsystems nach industriellen Standards (Einsatz als embedded system) erfordert gegenüber der einfachen Programmierung des Controllers viel mehr Zeit und mehr Programmteile, sowie den Einsatz von Netzwerkvariablen und Kommunikationskanälen zwischen den Programmen.

Die FPGA weißt im Vergleich zum Controller zwar eine deutlich schnellere Taktrate auf, jedoch kann das verwendete Modul diese Schaltgeschwindigkeit nicht leisten.

In Versuch 8 wird unter Verwendung eines anderen Moduls gezeigt, wie gravierend die Unterschiede zwischen zwei Modulen sein können.

5 Labor zur Sensorik

In dieser Laborveranstaltung soll es vorrangig darum gehen, die Theorie aus Kapitel 2.2 praktisch zu vermitteln. Zum diesem Zweck werden in den folgenden Versuchen einige Sensoren unterschiedlichen Typs untersucht.
In den Versuchen sollen die Studierenden eigenständig benötigte Informationen für die richtige Interpretation der Ausgangssignale der Sensoren herausfinden. Alle Sensoren werden ihre Signale an das cRIO-System übermitteln. In geeigneten Programmen sollen diese Signale dann dargestellt oder ausgewertet werden.
Ziel dieser Laborversuche ist es, den Studierenden zu vermitteln, wie unterschiedlich die primären Ausgangssignale von Sensoren sein können und wie leicht oder auch aufwendiger es sein kann diese, für das cRIO-System, in einfach zu verarbeitende Signale umzuwandeln. Außerdem sollen die Studierenden in diesen Laborversuchen praktische Arbeiten in Form von Lötaufgaben durchführen. Im Versuch 4 wird der zu untersuchende Sensor aus elektronischen Bauteilen selber zusammengebaut und schließlich mit dem cRIO-System untersucht. Im Versuch 5 werden zwei integrierte Sensoren für die Neigungs- und Winkelmessungen benutzt. Abschließend wird im Versuch 6 ein Temperatursensor und die benötigte Hardware für die Auswertung seines primären Ausgangssignals vorgestellt. In diesem Versuch soll der zuvor beschriebene Aufwand für die Umwandlung eines primären in ein analoges Ausgangssignal (siehe Abbildung 6) verdeutlicht werden.
Die von den Studierenden im Versuch 4 hergestellte Schaltung, für die Auswertung der Lichtschrankensignale, wird im Laborversuch 7 für die Ansteuerung von Aktoren wieder verwendet. Eine weitere Verwendung der Lichtschranke und des Temperatursensors ist ebenfalls möglich, dies wird im Kapitel 8 gezeigt.

5.1 Versuch 4: Untersuchung von verschiedenen Sensoren

Aufgabenstellung:

In diesem Laborversuch sollen einfache Sensoren untersucht werden. Die ausgegebenen Werte der Sensoren sollen mit geeigneten Modulen mit dem cRIO-System aufgenommen werden. Zu untersuchen sind ein Joystick und der einfache Aufbau einer Lichtschranke.

Für die Untersuchung der beiden Sensoren soll das cRIO-System mit der Scan-Engine verwendet werden. Es sollen zwei Programme geschrieben werden, welche die Signale der jeweiligen Sensoren grafisch wiedergeben können. In diesem Fall kann dies z.B. mit booleschen Anzeigeelementen geschehen.

Der Joystick (siehe Abbildung 30) hat am Ende seines Anschlusskabels Anschlussklemmen an denen die Signale der Schalter abgenommen und mit dem digitale Ein-/Ausgangs-, TTL-Modul NI-9403 ausgewertet werden sollen. Die Verkabelung im Joystick kann durch einfaches Verfolgen der Leiterbahnen herausgefunden werden.

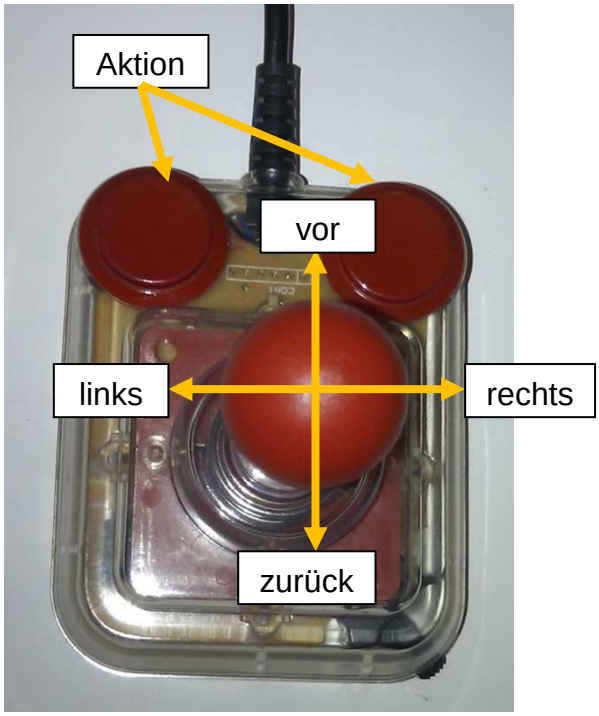

Abb. 30 Der verwendete Joystick und die möglichen Eingaben.

Für die Lichtschranke werden einzelnen elektronischen Komponenten bereitgestellt. Es handelt sich um eine IR-LED und einer Fotodiode. Welche der beiden Dioden die LED ist, sei zunächst unbekannt. Dies kann jedoch auf sehr einfache Weise herausgefunden werden. Nachdem die LED identifiziert worden ist, werden die beiden Dioden auf die bereitgestellte Leiterplatte gelötet. Die Polung der LED ist auf der Leiterplatte zu markieren. Da Fotodioden bei eintreffendem Licht einen Strom einstellen, kann dieses Signal nicht direkt an den bereitgestellten Modulen von NI gemessen werden. Für die Auswertung ist also weitere Elektronik notwendig. Es bietet sich an das Signal der Fotodiode mit Hilfe eines Transistors auszuwerten.
Die nötige Schaltung ist in Abbildung 31 zu sehen. Der Transistor und die weiteren Komponenten werden zur Verfügung gestellt und müssen auf einer Leiterplatte zusammengelötet werden.

Aufbau der Lichtschranke

Die Komponenten sind in Abbildung 31, links zu sehen. Im zusammengebauten Zustand sollte sich ein ähnliches Bild wie in Abbildung 31, rechts ergeben.

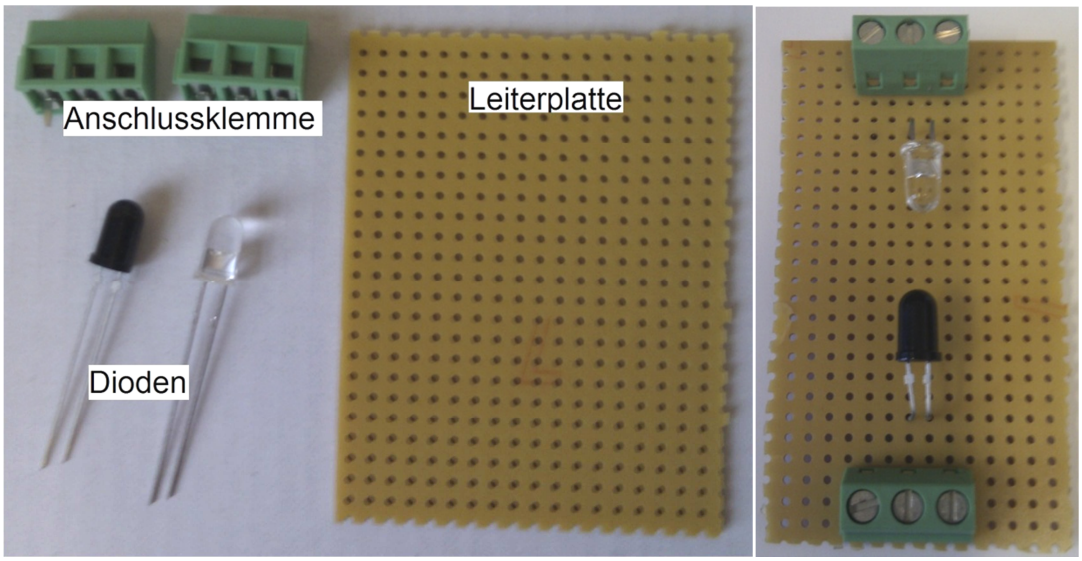

Abb. 31 Die einzelnen Komponenten (links) und die zusammengebaute Lichtschranke (rechts).

Um herauszufinden welche der beiden Dioden die LED ist, kann z.B. jede beliebige Handykamera benutzt werden. Hierfür muss die zu testende Diode nur mit 1,35 V Gleichspannung versorgt werden und dann mit der Handykamera aufgenommen werden. Nun sollte auf dem Bild ein violettes Leuchten zu sehen sein, sollte es sich um die LED und die richtige Polung handeln. Eine falsche Polung können die

Bauteile aushalten, solange drauf geachtet wird die Versorgungsspannung auf 1,35 V zu begrenzen.

Aufbau der Transistorschaltung

Der Schaltplan für die Transistorschaltung ist in Abbildung 32 zu sehen.

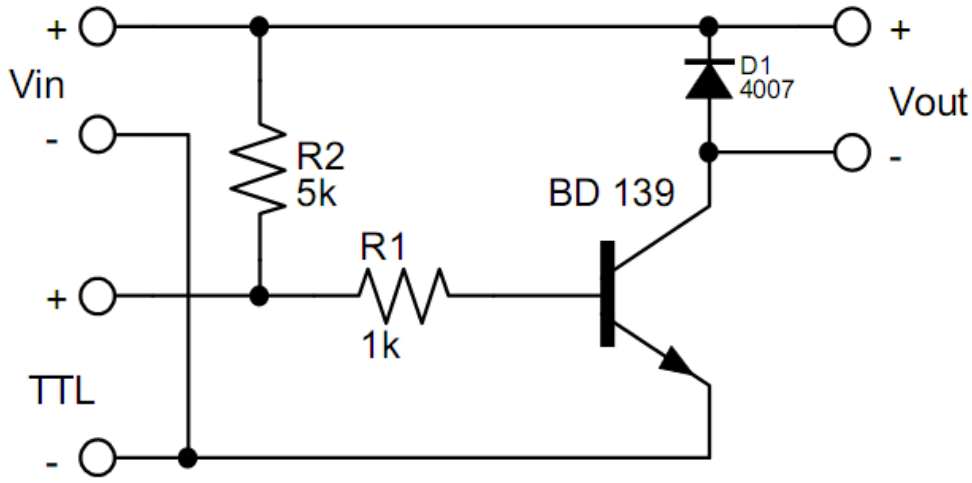

Abb. 32 Schaltplan der Transistorschaltung.

Alle nötigen Komponenten sind in Abbildung 33 dargestellt. In diesem Fall werden, aus Mangel an 5 kΩ Wiederständen, für *R2* zwei 10 kΩ Widerstände parallel geschaltet. Die Schaltrichtung der Diode lässt sich entweder an der durchgängigen Markierung an einem Ende des Gehäuses oder mit einem Multimeter bestimmen. Für die Pinbelegung des Transistors wird das passende Datenblatt bereitgestellt [31].

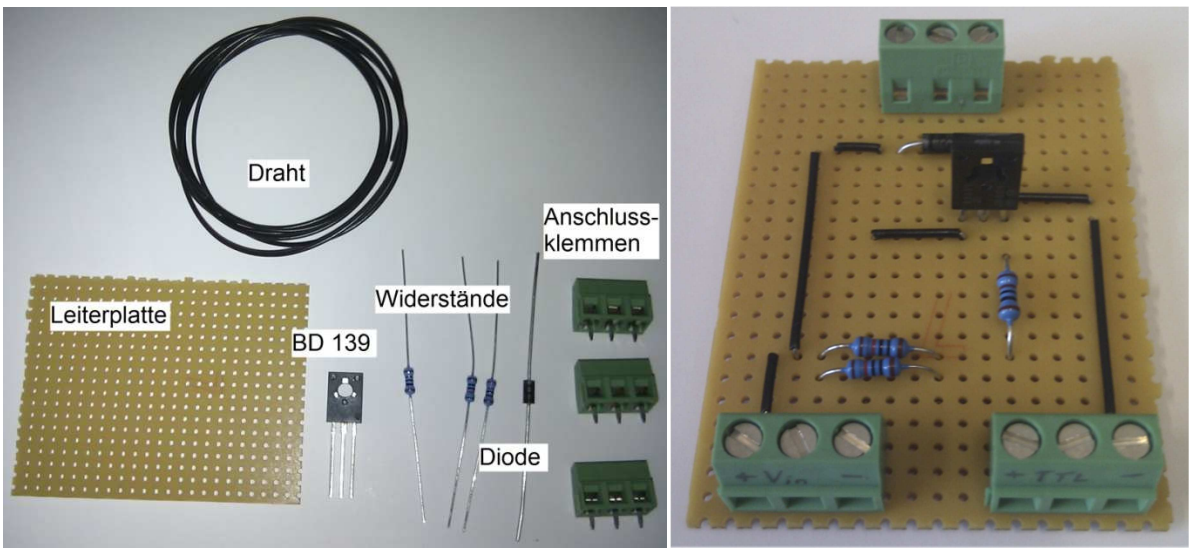

Abb. 33 Die einzelnen Komponenten (links) und die fertig montierte Schaltung (rechts).

Funktionsweise der Transistorschaltung:

Bei dem hier verwendeten Transistor handelt es sich um einen bipolaren NPN Transistor. Es gibt NPN und PNP Transistoren. Die Bezeichnung richtet sich nach dem Aufbau der Halbleiterschichten im Transistor. Vereinfacht können Transistoren mit zwei hintereinander geschalteten Dioden verglichen werden. In Abbildung 34 ist der Aufbau von Transistoren mit den dazugehörigen Diodenerstatzschaltbildern prinzipiell dargestellt. Zwischen den P- und N-Schichten bilden sich durch Diffusion Sperrschichten. Die Pole an einen Transistor werden immer mit Kollektor (C), Basis (B) und Emitter (E) bezeichnet. Die genaue Anschlussbelegung dieser drei Pole an den verschiedenen Bauformen kann dem jeweiligen Datenblatt entnommen werden.

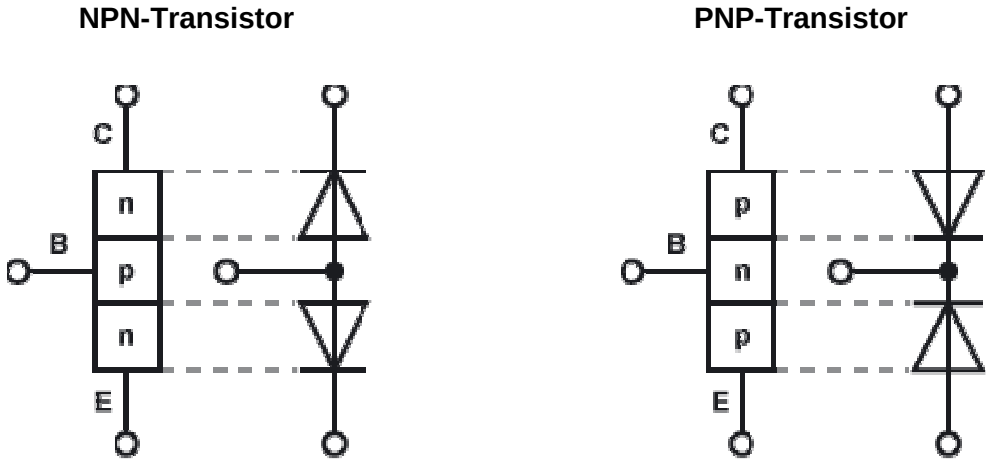

Abb. 34 Aufbau eines NPN-, und PNP-Transistors und eine Vergleichbare Schaltung mit Dioden, nach [10].

Damit die Funktion eines Transistors physikalisch erklärt werden kann, muss auch die physikalisch korrekte Stromrichtung (von Minus nach Plus), als Betrachtungsgrundlage genommen werden. In der mathematischen Berechnung wird die technische Stromrichtung (von Plus nach Minus) verwendet. In Abbildung 35 ist ein NPN-Transistor abgebildet. An diesen wird zwischen Basis und Emitter eine Spannung U_{BE} angelegt. Dadurch ist die untere Diode im Prinzip in Durchlassrichtung geschaltet. Die Elektronen wandern vom Minus-Pol der Spannungsquelle durch die N-Schicht (Emitter) in die P-Schicht (Basis) und schließlich zum Plus-Pol der Spanungsquelle. Beim Durchqueren der p-Schicht reichern sich Elektronen in der oberen Sperrschicht an und machen diese leitend. Jetzt kann ein Strom zwischen Kollektor und Emitter fließen (I_C).

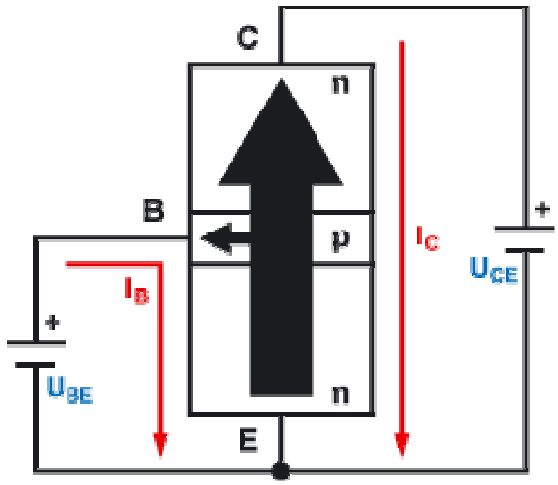

Abb. 35 Funktionsprinzip eines NPN-Transistors mit der physikalischen (schwarze Pfeile) und mathematischen Stromrichtung (rote Pfeile) [21].

In diesem Versuch werden die Transistoren vereinfacht als elektronische Schalter verwendet. Wenn die Fotodiode ein Signal feststellt wird sie leitend. Es fließt entsprechend der Schaltung in Abbildung 30 ein Strom von V_{in} + über den Widerstand *R2* nach *TTL* +, durch die Fotodiode und zurück nach *TTL* - zu V_{in} -. Sollte ein Hindernis zwischen die IR-LED und die Fotodiode kommen, wird diese nicht leitend. Der Strom fließt nun statt durch die Fotodiode durch den Transistor, was diesen wiederum für den Strom durch V_{OUT} leitfähig macht. Man erhält bei einem Signal der Fotodiode (nicht leitend) ein Ausgangssignal an V_{OUT} (leitend). Diese Negation des Signales kann erwünscht sein oder auch nicht. Bei diesem Versuch spielt sie keine Rolle. Mit der Verwendung eines äquivalenten PNP-Transistors in der gleichen Beschaltung kommt es zu keiner Negation.

Verwendung der Schaltungen:

Damit das cRIO-System die Signale der Lichtschranke auswerten kann, werden die beiden Schaltungen nun mit einander verbunden. Die Fotodiode wird entsprechend ihrer Polung mit den *TTL* Eingang der Transistorschaltung verbunden. Die LED wird mit 1,35 V versorgt und die Versorgungsspannung V_{in} an der Transistorschaltung eingestellt. Die Einstellung richtet sich nach dem digitalen Eingangsmodul NI-9421, welches an V_{out} angeschlossen werden soll. Aus den Begleitunterlagen zum Modul lässt sich die untere Spannungsschwelle für den „ON“-Status ermitteln.

5.2 Musterlösung Versuch 4

In diesem Versuch sollten die Studierenden die meiste Zeit damit beschäftigt sein, die Schaltungen zusammenzubauen. Die geforderten Programme in LabVIEW sind deshalb relativ einfach gehalten.
Die zusammengebauten Schaltungen sind bereits in den Abbildungen 31 und 33 zu sehen. In Abbildung 36 ist ein kompletter Testaufbau abgebildet.

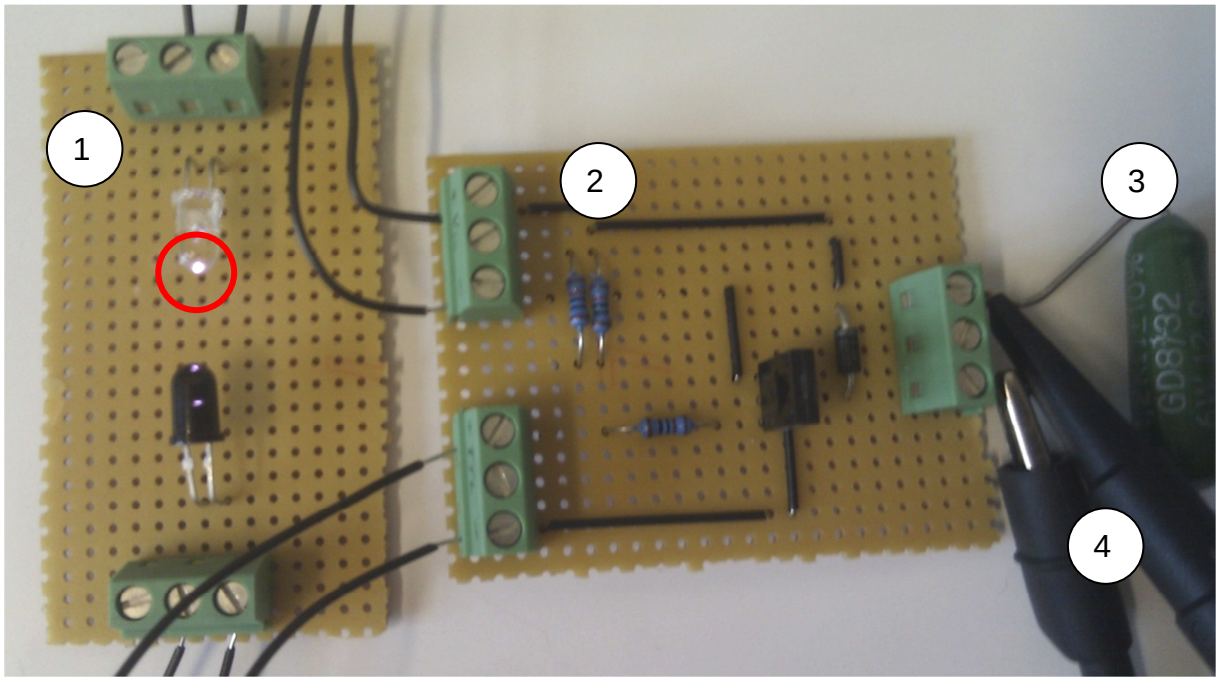

Abb. 36 Testaufbau der Lichtschranke mit Transistor-Schaltung, Widerstand und Messabnehmern des Oszilloskops.

Die wichtigsten Komponenten sind mit Nummern versehen. Abgebildet sind:

1. Die Lichtschranke. Zu sehen ist auch das zuvor beschriebene violette Leuchten der IR-LED (siehe Abbildung 36, im roten Kreis).
2. Die Transistorschaltung. Diese wird für die Verwendung mit dem digitalen Eingangsmodul mit einer Spannung von *12 V* betrieben.
3. Ein Widerstand, damit die Schaltung getestet werden kann. Stattdessen wird anschließend das Modul angeschlossen.
4. Die Messabnehmer des Oszilloskops.

Für die Auswertung des Joysticks kann das in Abbildung 37 abgebildete Programm verwendet werden.

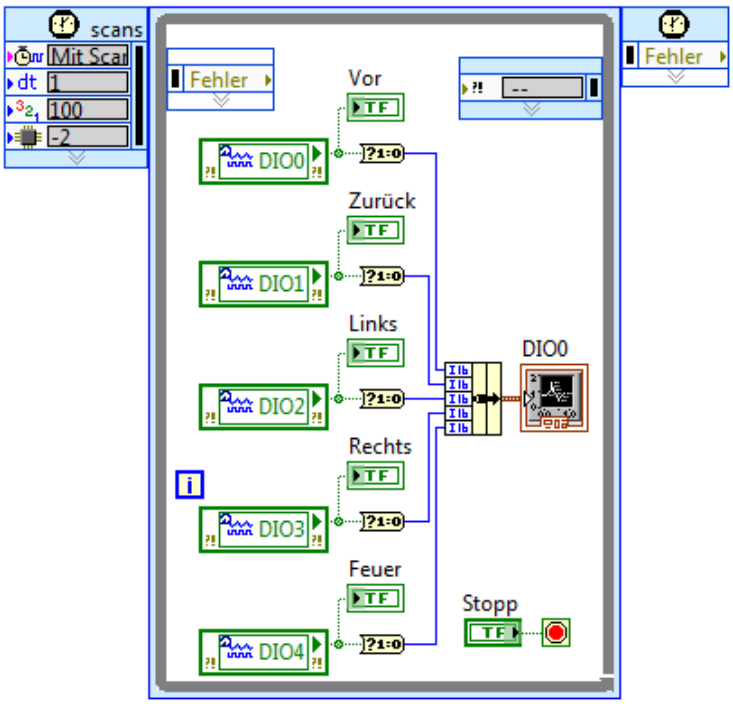

Abb. 37 Blockdiagramm des Programms zum Auswerten der Signale von Joystick.

In diesem Programm werden die Zustände der digitalen Eingänge einzeln mit booleschen Anzeigeelementen dargestellt (siehe Abbildung 38, 1). Eine weitere Visualisierungsvariante ist in Form eines Signalverlaufsdiagramms (Abbildung 38, 2) ebenfalls eingebaut. Damit die fünf booleschen Werte in einem Diagramm angezeigt werden können wurden sie zunächst in Nullen und Einsen umgewandelt und anschließend zu einem Cluster gebündelt. Das Signalverlaufsdiagramm interpretiert jedes Element des Clusters als eigenes Signal und stellt diese separat dar.

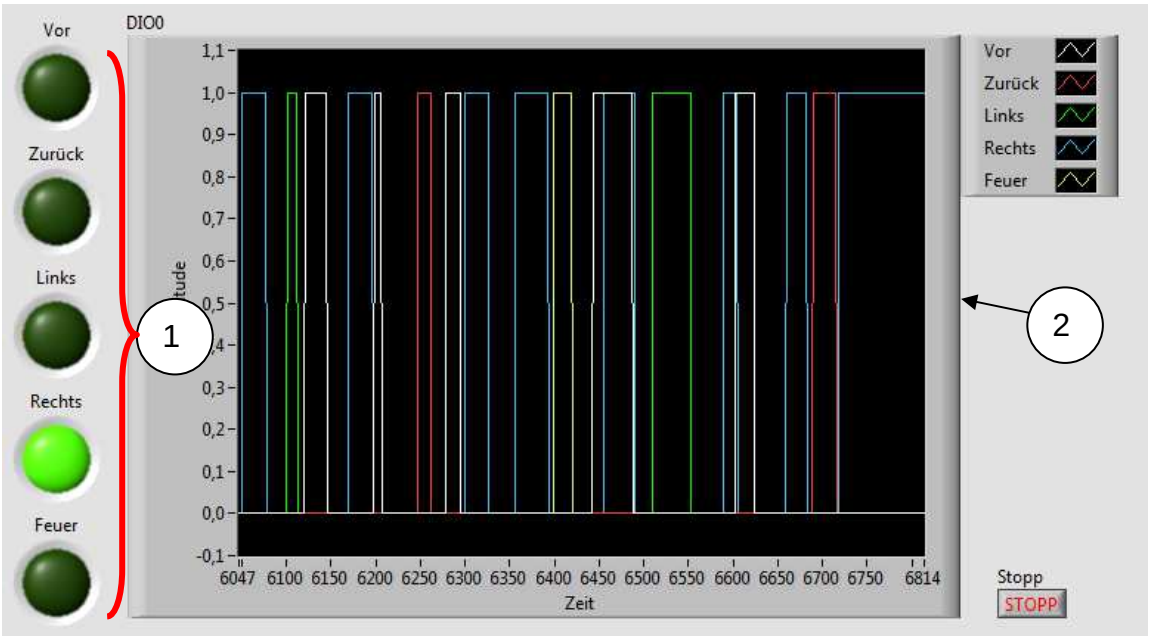

Abb. 38 Das Frontpanel des Programms zur Auswertung der Joystick Signale.

Das Programm für die Lichtschranke ist im Grunde dasselbe, wie jenes für den Joystick, nur das es lediglich einen digitalen Eingang auswertet.

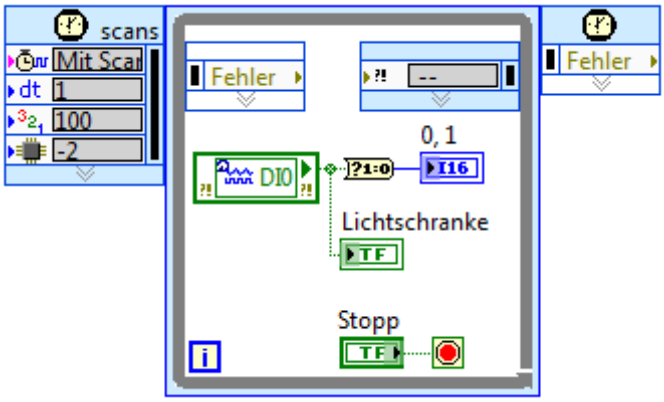

Abb. 39 Blockdiagramm des Lichtschrankenprogramms.

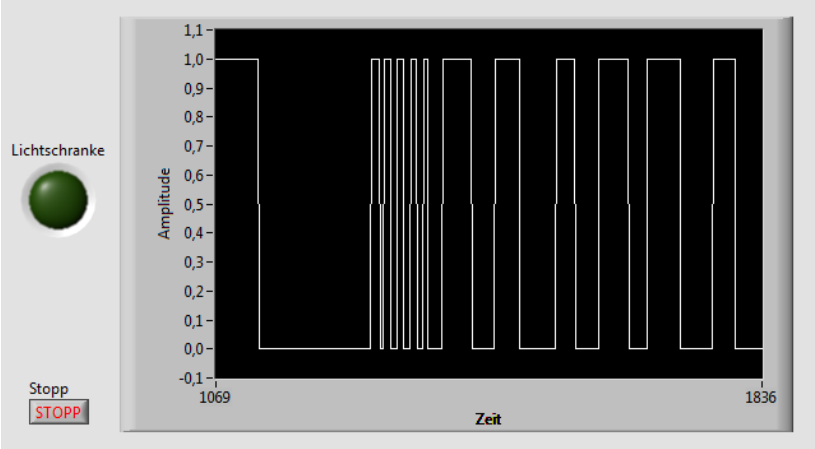

Abb. 40 Frontpanel des Lichtschrankenprogramms.

5.3 Versuch 5: Untersuchung eines Neigungs- und Beschleunigungssensors

In diesem Laborversuch sollen ein Neigungssensor und ein Beschleunigungssensor untersucht werden. Zunächst soll die Funktion der beiden Sensoren mit Hilfe des Oszilloskops getestet werden. Danach werden die Sensoren an das cRIO-System angeschlossen und Programme zum Auswerten der Signale geschrieben.

Der Neigungssensor misst die Neigung um zwei Achsen und gibt diese als Spannungen aus. Es soll mit Hilfe eines Geodreiecks herausgefunden werden, in welchem Winkelbereich der Sensor arbeitet und welche Spannungen er an den *Endpunkten* ausgibt. Die ermittelten Messwerte werden in Tabelle 2 dokumentiert. Anschließend sollen für vier vorgegebene Neigungen die Spannungen gemessen

werden. Für eine spätere einfache Auswertung in LabVIEW ist vor allem interessant, ob sich die Spannungen auch linear interpolieren lasen. Die errechneten Spannungen sollen zur Kontrolle ebenfalls in die Tabelle eingetragen werden.
Mit Hilfe der erfassten Daten kann nun ein LabVIEW Programm geschrieben werden, welches aus den gemessenen Spannungen die Neigung berechnet. Für den Beschleunigungssensor soll ebenfalls ein Programm geschrieben werden, welches aus den Messwerten die Neigung berechnet.

Winkel (°)		**Spannung (V)**	
x-Achse	y-Achse	U_x	U_y
0	**0**		
(max -)	**0**		
(max +)	**0**		
0	(max -)		
0	(max +)		
+10	**0**		
-10	**0**		
0	**+10**		
0	**-10**		

Tabelle 2 Dokumentation der Messwerte vom Neigungssensor.

5.4 Musterlösung Versuch 5

In eigenen Versuchen konnten für den Neigungssensor Messdaten ermittelt und dokumentiert werden (siehe Anhang 10.2). Die Überprüfung der linear interpolierten Werte für ±10° Neigung, um die x- und y-Achse, weisen eine hinreichende Genauigkeit auf.

Das Programm für die Berechnung der Neigungen wurde in zwei Teile zerlegt. In Abbildung 41 ist das Hauptprogramm zu sehen. In diesem werden die Messwerte vom analogen Eingangsmodul erfasst, die gemessenen Maximalwerte für die lineare Interpolation können eingegeben werden und schließlich werden die errechneten Neigungen graphisch ausgegeben.

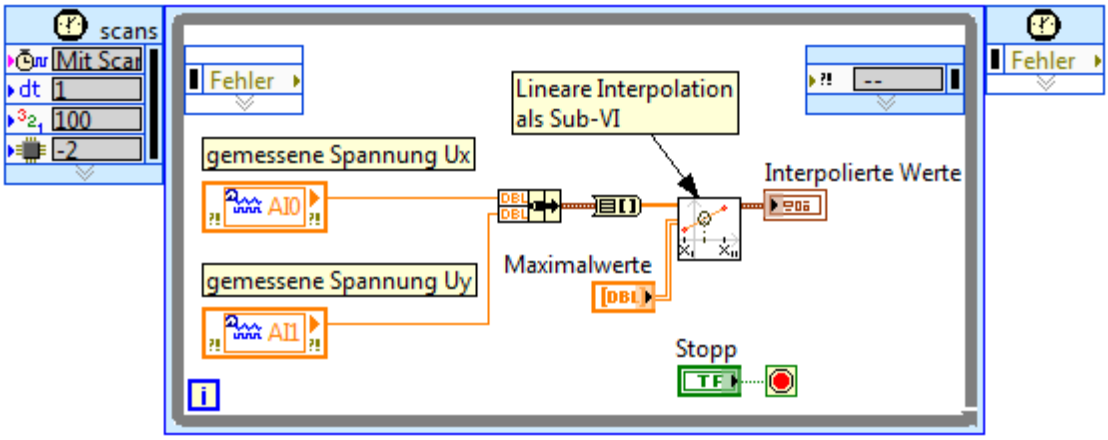

Abb. 41 Hauptprogramm für die Neigungsberechnung.

Die lineare Interpolation geschieht, wie in Abbildung 42 zu sehen, in der Sub-VI *lineare_Interpolation*. Als Eingänge benötigt die VI die aktuellen *Messwerte* in einem Array und ein Array mit den Werten, zwischen denen interpoliert werden soll (*Array (Maximalwerte)*). Ausgegeben werden die *interpolierten Werte* in einem Cluster mit zwei Elementen.

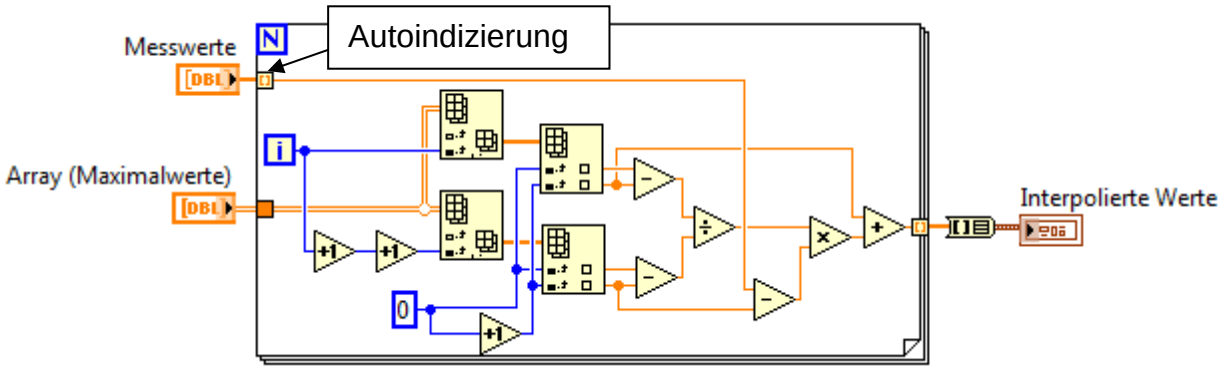

Abb. 42 Sub-VI *lineare_Interpolation*.

In dieser VI werden die beiden Messwerte von U_x und U_y in einer Schleife einzeln und nacheinander bearbeitet. Realisiert wurde dies mit einem autoindizierten Schleifentunnel. So werden der For-Schleife die Elemente aus dem Array *Messwerte* einzeln übergeben und die Schleife wird genauso oft ausgeführt, wie Elemente in dem Array vorhanden sind. Für diesen Fall, mit zwei Elementen im Array *Messwerte* (U_x und U_y), genau zwei mal. Mit Hilfe von einigen Arrayfunktionen wurden dann für die Interpolation die entsprechenden Werte aus dem *Array (Maximalwerte)* ausgelesen.
Die Interpolation wurde nach der Formel 5.1 durchgeführt.

$$f(x) = f(x_0) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_0) \tag{5.1}$$

Die Werte der Winkel $f(x_0)$, $f(x_1)$ sowie die zugehörigen Spannungen x_0 und x_1 wurden zuvor ermittelt. Die in eigenen Versuchen ermittelten Werte sind der Tabelle 12, Kapitel 10.3 zu entnehmen.
Damit die VI richtig arbeitet ist die Formatierung des *Array (Maximalwerte)* entscheidend. In der ersten Spalte stehen die maximalen Neigungswinkel um die x-Achse. In der zweiten Spalte diejenigen der y-Achse. Die dritte Spalte beinhaltet die zugehörigen Spannungen $U_x(\pm\alpha_{x,max})$ zu den maximalen Neigungen. In der vierten Spalte stehen die Spannungen zu den Neigungen um die y-Achse $U_y(\pm\alpha_{y,max})$. In Abbildung 43 ist das Frontpanel des Programms zu sehen. Im oberen Teil findet sich das Eingabefeld für das Array mit den Maximalwerten. Die Neigungen werden in einem Signalverlaufsdiagramm darunter dargestellt.

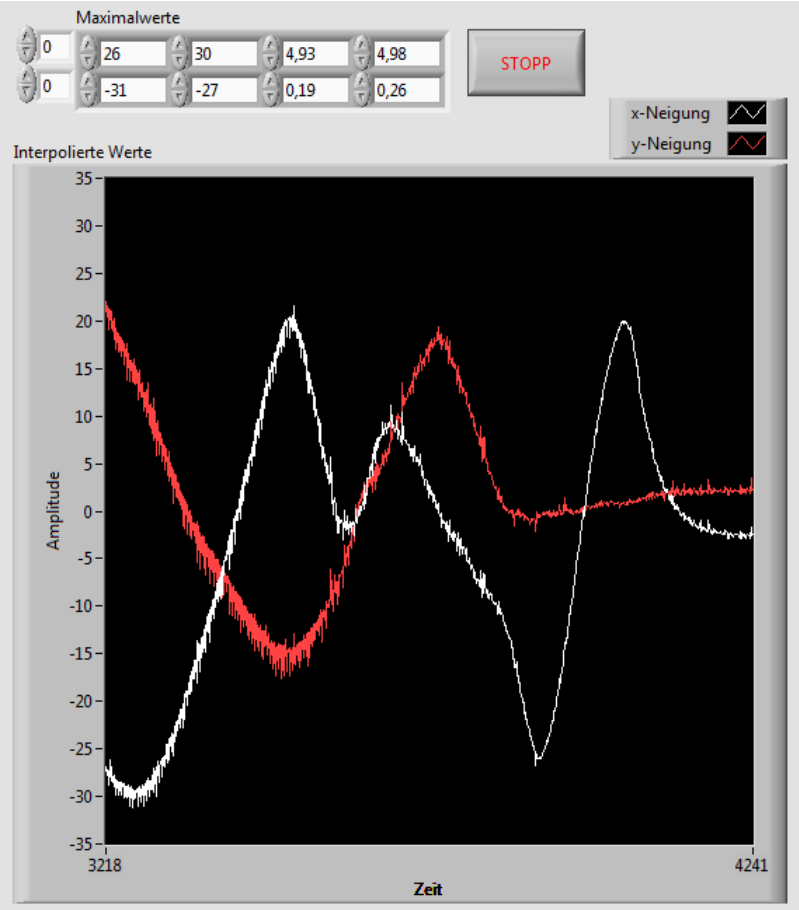

Abb. 43 Frontpanel des Neigungswinkelprogramms.

Für die Auswertung vom Beschleunigungssensor ergibt sich ein Programm wie in Abbildung 44 zu sehen. Aus der gemessenen Beschleunigung kann mit Hilfe einfacher Trigonometrie die Neigung berechnet werden:

$$\alpha(U) = \cos^{-1}\left(\frac{U}{i \cdot g}\right) \tag{5.2}$$

U steht für die gemessene Spannung in Volt, i für den Umrechnungsfaktor von der Spannung zur Beschleunigung (0,1019 V/(m/s²), entnommen dem Datenblatt des Sensors) und g für die Erdbeschleunigung in m/s².

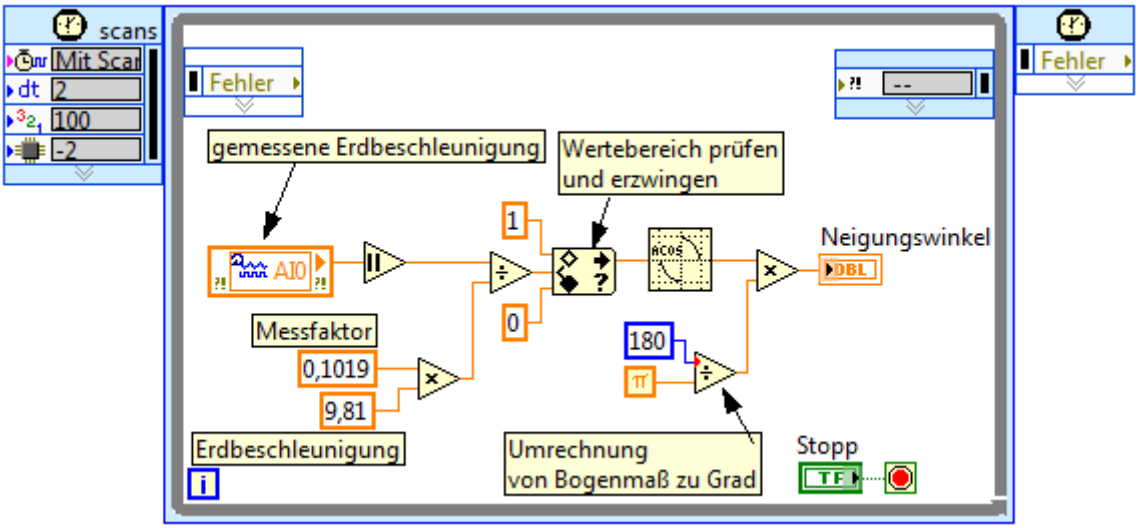

Abb. 44 Das Blockdiagramm mit der Berechnung des Neigungswinkels aus der gemessenen Erdbeschleunigung.

Neben der vorgestellten Formel für die Berechnung von $\alpha(U)$ wurde noch eine Funktion zum Prüfen und Erzwingen des Wertebereichs eingebaut. Da der Sensor auch eine Beschleunigung größer als die Erdbeschleunigung messen kann, würde dies sonst zu ungültigen Berechnungen des Arkuskosinus führen. Zudem war es notwendig eine Umrechnung von Bogenmaß zu Grad einzubauen.

Der Beschleunigungssensor ist prinzipiell dazu in der Lage auch „auf dem Kopf“ stehend Werte zu liefern. In dieser Position werden statt positive, negative Spannungen ausgegeben. In diesem Programm werden allerdings nur die absoluten Werte für die Berechnung verwendet.

Auf dem Frontpanel des Programms ist lediglich ein Signalverlaufsdiagramm zu sehen, welches den aktuellen Neigungswinkel ausgibt. Auf eine Abbildung wird an dieser Stelle verzichtet.

5.5 Versuch 6: Untersuchung eines Temperatursensors

In diesem Laborversuch soll ein Temperatursensor an das cRIO-System angeschlossen und die Temperatur gemessen werden. Dafür ist ein entsprechendes Programm zu schreiben.

Der Temperatursensor wird als Bauteil zur Verfügung gestellt. Die Aufgabe besteht darin zu ergründen, wie der Sensor angesteuert werden muss und wie eine mögliche Schaltung aussehen muss, um die Temperatur mit einem analogen Eingangsmodul am cRIO-System auszulesen.

Sensoren arbeiten für die Erfassung von Messwerten nach den verschiedensten physikalischen Prinzipien. Genauso unterschiedlich sind auch die primären Ausgangssignale, welche die Sensoren liefern. In diesem Versuch wird ein Temperatursensor benutzt, welcher mit Halbleitermaterialen arbeitet. Dieses verändert bei steigender Temperatur seine Leitfähigkeit.
In der Regel werden Sensoren so ausgeliefert, dass sie ein analoges Ausgangssignal liefern. Falls dies aber nicht der Fall sein sollte, muss entsprechende Elektronik beschafft werden, die diese Aufgabe erledigt. Für einfache Sensoren in Versuchsaufbauten kann es günstiger sein, solch eine Elektronik selber herzustellen. Zumal viele Schaltpläne direkt aus den Datenblättern der Bauteile entnommen werden können. In dem Datenblatt des Temperatursensors soll eine mögliche Schaltung für das Messen der Temperatur gefunden werden. Anschließend wird eine etwas angepasste Schaltung samt Schaltplan zur Verfügung gestellt. Die Schaltung soll nachvollzogen und anschließend kalibriert werden.

Aufbau der Schaltung für den Temperatursensor

In dem Datenblatt des Temperatursensors *AD 592* [30] kann auf Seite 8 eine Schaltung für ein Celsius/Fahrenheit Thermometer gefunden werden (siehe Abbildung 45).

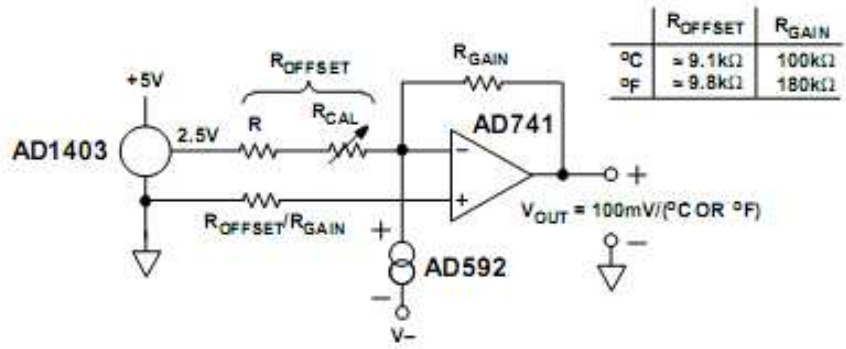

Abb. 45 Celsius/Fahrenheit Thermometer, nach [30].

In dieser Schaltung werden neben dem Sensor noch zwei weitere wichtige Bauteile verwendet. Der *AD1403* ist ein IC, welcher eine konstante Spannung von 2,5 V ausgibt, dies ist vor allem für eine genaue Temperaturmessung und Ausgabe der entsprechenden Spannung wichtig. Ebenso wurde hier ein Operationsverstärker *AD741* verwendet. Dieser stellt die Ausgangsspannung V_{OUT} entsprechend der gemessenen Stromstärke vom Temperatursensor ein. Auf eine genauere

Funktionsbeschreibung des Operationsverstärkers soll an dieser Stelle verzichtet werden, dafür stehen verschiedene Nachschlagewerke im Bereich der Elektrotechnik zur Verfügung.

Die Schaltung in Abbildung 45 wurde etwas abgeändert. Der dort verwendete Operationsverstärker wird heute kaum noch benutzt, da es genauere und stabilere Modelle gibt. Ein allgemein bewährter Operationsverstärker ist der *TL071* [34]. Für die Spannungsversorgung wurde anstelle des *AD1403* ein einstellbarer Spannungswandler verwendet. Der *LM317* [32] lässt sich mit zwei Widerständen genau auf die gewünschte Spannung einstellen. In dem überarbeiteten Schaltplan ist zudem die Spannungsversorgung für den Operationsverstärker eingezeichnet. Für die Versorgung der Bauteile mit -5V wurde ein DC-DC Konverter *AM15-0505SZ* benutzt. Abbildung 46 zeigt den überarbeiteten Schaltplan.

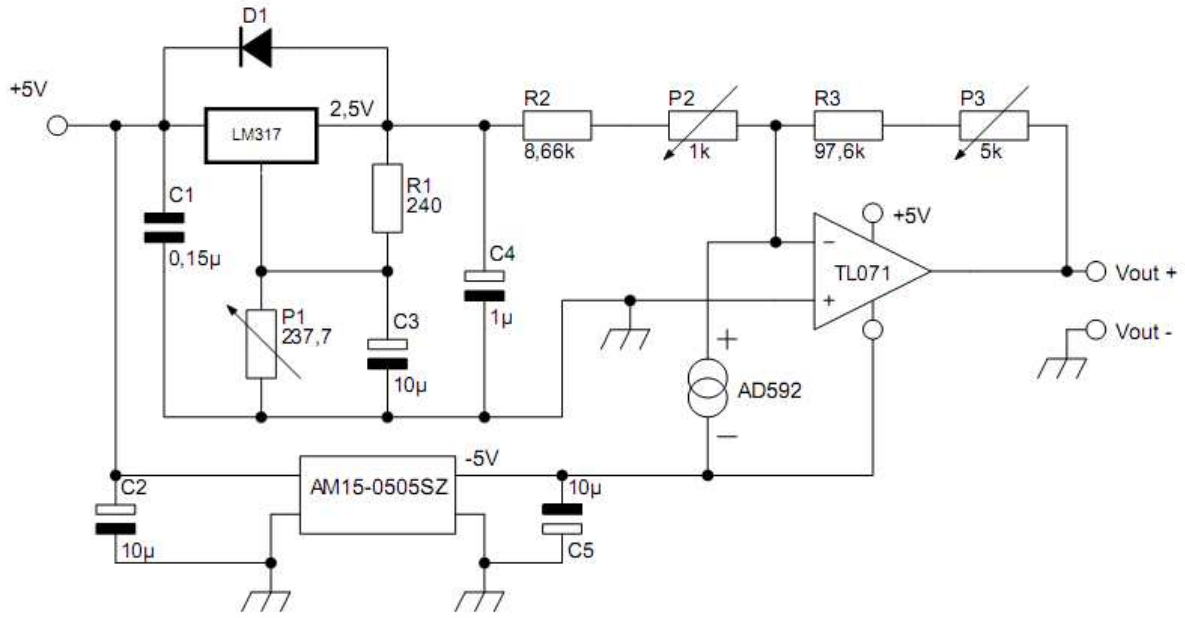

Abb. 46 Schaltplan des überarbeiteten Thermometers.

Wie die Schaltung aus Abbildung 45, gibt auch die überarbeitete Schaltung für je 1 °C gemessene Temperatur 0,1 V aus. Eine Temperatur von 21 °C sollte also zu einer Ausgangsspannung von 2,1 V führen.

Kalibrierung der Schaltung

Zum Kalibrieren der Schaltung stehen die drei Potentiometer *P1*, *P2* und *P3* zur Verfügung (siehe Abbildung 46 und 47). Eine Änderung des Widerstands *P1* bewirkt eine Änderung der ausgegebenen Spannung des *LM317.* Mit einem Multimeter kann

die Ausgangsspannung gemessen werden und vorsichtig auf 2,5 V eingestellt werden. Mit dem Widerstand *P2* kann der Nullpunkt justiert werden. Bei richtiger Einstellung gibt die Schaltung so 0 V bei 0 °C aus. Zum einfachen kalibrieren kann ein Thermometer verwendet werden und die ausgegebene Spannung mit Hilfe von *P2* so eingestellt werden, das sie der vom Thermometer gemessenen entspricht. Der Widerstand *P3* ist schließlich zum Ausgleichen der Nichtlinearität des Temperatursensors im hohen Temperaturbereich gedacht. Mit Hilfe eines Leistungswiderstands oder einer anderen Wärmequellen, kann der Temperatursensor zusammen mit dem Thermometer erwärmt werden und eine sich evtl. einstellende Differenz zwischen den gemessenen Temperaturen mit Hilfe von *P3* ausgeglichen werden.

In Abbildung 47 ist die aufgebaute Schaltung zu sehen. Die drei Potentiometer sind mit ihren Bezeichnungen aus dem Schaltplan markiert. Einige wichtige Bauteile sind ebenfalls markiert. Zu sehen sind auch die Anschlussklemmen für die 5 V Versorgung (oben links), die Ausgangsspannung V_{out} und für den Temperatursensor (beide rechts mittig). In diesem Aufbau wurde dieser direkt an die Klemmen angeschlossen. Er kann jedoch auch mit einem Verlängerungskabel weiter von der Platine entfernt betrieben werden.

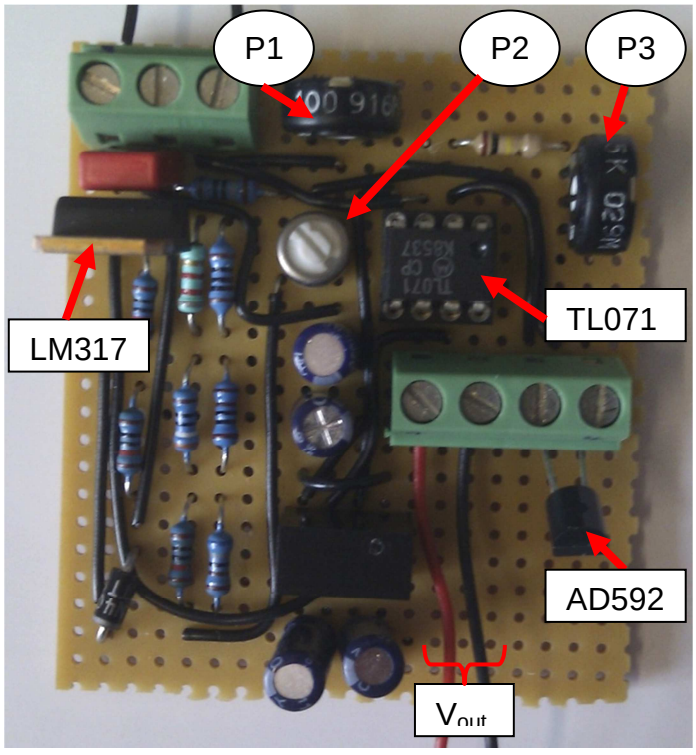

Abb. 47 Schaltung zum Umwandeln der Signale des Temperatursensors ein analoges Ausgangssignal.

Benutzung der Schaltung

Damit die Schaltung eingesetzt werden kann muss sie lediglich mit 5V Spannung versorgt werden. An den Anschlussklemmen V_{out}+ und V_{out}- kann die Spannung abgenommen und den analogen Eingangsmodul NI-9215 zugeführt werden.

5.6 Musterlösung Versuch 6

Dieser Versuch ist prinzipiell so aufgebaut, dass die Platine den Studierenden in Einzelteilen gegeben werden kann. Mit Hilfe des Schaltplans und der zugehörigen Datenblätter zu den Bauteilen sollte es möglich sein die Schaltung nachzubauen. Jedoch wurde in der hier vorgestellten Versuchsbeschreibung auf diese Möglichkeit verzichtet. Die Gründe hierfür sind eher praktischer Natur. Denn auch der Nachbau, einer als Bild gezeigten Platine dieser Größe, kann ein bis zwei Stunden dauern.
Deshalb wurde die Schaltung in diesem Versuch nur vorgestellt und die Kalibrierung als Aufgabe gestellt. Für die Kalibrierung wird natürlich ein möglichst genaues Thermometer benötigt. Der Nullpunkt wurde bei Raumtemperatur eingestellt. In einer durchgeführten Messung wurde bei eine Temperatur von 20°C das Potentiometer *P2* wurde so eingestellt, das 2 V an V_{out} ausgegeben wurden. Für die Einstellung von *P3* wurde der Temperatursensor mit einem Leistungswiderstand erwärmt. Die Temperatur wurde mit dem zweiten Thermometer gemessen und mit dem angezeigten Wert der Schaltung verglichen. Eine signifikante Abweichung konnte bis zum höchsten Messwert von 50°C nicht festgestellt werden.
Das zugehörige LabVIEW-Programm ist sehr einfach aufgebaut. Die ausgelesene Spannung an einer der analogen Eingänge wird mit zehn multipliziert und in einem Signalverlaufsdiagramm ausgegeben.

6 Labor zur Aktorik

In der letzten Laborveranstaltung werden das erlernte Wissen und die Grundlagen aus den ersten beiden Laborveranstaltungen angewendet, um Aktoren mit dem cRIO-System zu steuern. Die Auswahl der Aktoren wurde in Kapitel 2.1 durchgeführt. Es wurden ein DC-Motor, DC-Ventilator, Lautsprecher, Peltier-Element und ein Schrittmotor für die Laborveranstaltung ausgewählt.
Im Laborversuch 7 werden der DC-Motor, DC-Ventilator, Lautsprecher und das Peltier-Element angesteuert. Dafür werden zwei Module (NI-9472 und NI-9403) verwendet und deren Unterschiede erarbeitet. Dabei wird auf die von NI gegebenen Möglichkeiten hinsichtlich der Ausgangssignale eingegangen. In diesem Versuch wird zudem der Begriff der Pulsweitenmodulation erklärt. Diese wird für die beiden nachfolgenden Versuche benötigt.
Im 8. Laborversuch erfolgt dann die Entwicklung einer optimierten Steuerung für einen DC-Motor. Mit dieser Steuerung lässt sich der Motor nicht nur in seiner Drehzahl, sondern auch in seiner Drehrichtung steuern.
Im letzten Laborversuch wird schließlich ein Schrittmotor gesteuert. Dies gescheit mit Hilfe des in Versuch 4 vorgestellten Joysticks und der in Versuch 8 vorgestellten H-Brücke. Das Programm für die Steuerung wird auf der FPGA konfiguriert.
Eine weitere Verwendung des Peltier-Elements und Ideen für den Ausbau der Schrittmotorsteuerung werden in Kapitel 8 geschildert.

6.1 Versuch 7: Ansteuerung von verschiedenen Aktoren

In diesem Laborversuche werden vier verschiedenen Aktoren mit Hilfe der NI Hardware angesteuert. Das cRIO-System soll mit der Scan-Engine betrieben werden.
Die Ausgangsgröße eines Aktors ist eine Energie oder Leistung. Diese wird in den meisten Fällen als Rotations- oder Translationsenergie bereitgestellt. Kann aber auch in anderer Form vorliegen. Aktoren sind im Allgemeinen Stelleinrichtungen, welcher eine bestimmte physikalische Größe im System gemäß ihrer Eingangssignale einstellen. Dazu zählen auch Heizstäbe und sogar Lampen. In Gewächshäusern werden die Pflanzen auch nachts mit Licht versorgt, um das Wachstum voranzutreiben. In diesem Fall sind Lampen also Aktoren, welche in dem System Gewächshaus die Beleuchtungsstärke (*physikalische Größe*, gemessen in Lux)

einstellen. In der Mechanik werden solche Aktoren, aufgrund ihrer doch einfachen Funktionsweise und Ansteuerung nicht behandelt. Häufiger vertreten sind elektromagnetische und fluidische Aktoren.
In diesem Laborversuch sollen ein DC-Motor, ein DC-Ventilator, ein Lautsprecher und ein Peltier-Element angesteuert werden. Der Motor arbeitet bei einer Spannung von 12 V, der Ventilator bei 24 V, dem Lautsprecher reicht eine Spannung von wenigen Volt und das Peltier-Elemente darf maximal mit 1,9 V betreiben werden. NI stellt für die Ansteuerung von Aktoren verschiedenste Module bereit. In diesem Versuch sollen einige davon eingesetzt und die verschiedenen Möglichkeiten objektiv bewertet werden.

Ansteuerung der Aktoren mit dem digitalen Ausgangsmodul NI-9472:

Die Ansteuerung des 12 V DC-Motors soll zunächst mit Hilfe des digitalen Ausgangsmoduls NI-9472 durchgeführt werden. Dazu muss der Motor an das Modul angeschlossen und ein entsprechendes Programm geschrieben werden. Dieses soll zunächst lediglich einen der Ausgänge des Moduls steuern.
Das Modul muss mit einer externen Spannungsquelle verbunden werden, hier können die vom Motor verlangten 12 V eingestellt werden.
Ebenso kann mit dem DC-Ventilator verfahren werden. Da dieser eine Spannung von 24 V benötigt muss die externe Spanungsquelle entsprechend eingestellt werden.

1. Welche Einschränkungen hinsichtlich der Steuerbarkeit von Motor und Ventilator zeigen sich?
2. Warum kann das Peltier-Element nicht mit dem NI-9472 angesteuert werden?

Damit Elektromotoren mit dem Modul besser verwendet werden können, hat NI eine wichtige Funktionalität in die Module integriert. Elektromotoren werden in der Automation mit so genannten *PWM*-Signalen gesteuert. PWM steht für Pulsweitenmodulation, eine genauere Beschreibung des Verfahrens erfolgt gesondert. Eine Änderung der Betriebsart, der digitalen Ausgänge, muss in den *Eigenschaften* des Moduls unter *Specialty Digital Configuration* vorgenommen werden. Der Modus *Pulse-Width Modulation* soll ausgewählt werden. Für den DC-Motor sollte eine Periode von 1 kHz eingestellt werden. Der Ventilator kann ebenfalls bei dieser Periode betrieben werden. Das Programm muss aufgrund der geänderten Betriebsart der Ausgänge angepasst werden.

3. Welche Verbesserung der Steuerbarkeit kann mit dem PWM-Signal erreicht werden?
4. In welcher Hinsicht unterscheiden sich Motor und Ventilator?

Anschließend soll der Lautsprecher mit dem Modul verbunden werden. Dieser kann mit einer Versorgungsspannung von 7 V betrieben werden. Die maximale Stromstärke ist sicherheitshalber auf 0,06 A zu begrenzen.

5. Welche Betriebsart wird für den Lautsprecher benötigt?

Ansteuerung mit Hilfe des digitalen Ein-/Ausgangs-, TTL-Moduls NI-9403:

Das Modul NI-9403 kann im Gegensatz zum vorherig verwendeten Modul nicht für die direkte Ansteuerung von Aktoren verwendet werden. Dafür ist eine Leistungsschaltung notwendig. Eine solche Schaltung wurde bereits im Laborversuch 4 für die Lichtschranke verwendet. Nun soll die gleiche Schaltung für die Umsetzung von Steuersignalen auf einen Aktor verwendet werden.
In den *Eigenschaften* des Moduls können einzelne Kanäle als Ein- oder Ausgänge eingestellt werden. Für diesen Versuch wird ein Kanal mit der Einstellung als Ausgang benötigt. Dafür wird der *Channel DIO0* ausgewählt und die *Initial Line Direction* auf *Output* gesetzt. An diesen Anschluss wird die Schaltung aus Versuch 4 angeschlossen. Hierfür müssen *DIO0* mit *TTL+* und *COM* mit *TTL-* verbunden werden. Die Versorgungsspannung für die Aktoren wird an die V_{in} Anschlüsse der Schaltung angelegt. An V_{OUT} kann nun ein Aktor angeschlossen werden. Für die Steuerung mit diesem Modul ist ein passendes Programm zu schreiben. Steuern Sie den DC-Motor, den Ventilator und den Lautsprecher mit diesem Modul an. Zu beachten ist die richtige Einstellung der Versorgungsspannung.

6. Welche Vorteile ergeben sich bei einer Ansteuerung von Aktoren mit einer Kombination aus NI-9403 und der Transistorschaltung?
7. Welche Nachteile gegenüber der Steuerung mit dem vorherig untersuchten Modul ergeben sich?
8. Kann das Peltier-Element mit diesem Modul und der Transistorschaltung angesteuert werden?

Pulsweitenmodulation

Pulsweitenmodulation oder auch Pulslängenmodulation beschreibt ein Verfahren, bei welchem die Impulslänge eines Rechtecksignals bei konstanter Periodendauer moduliert wird.

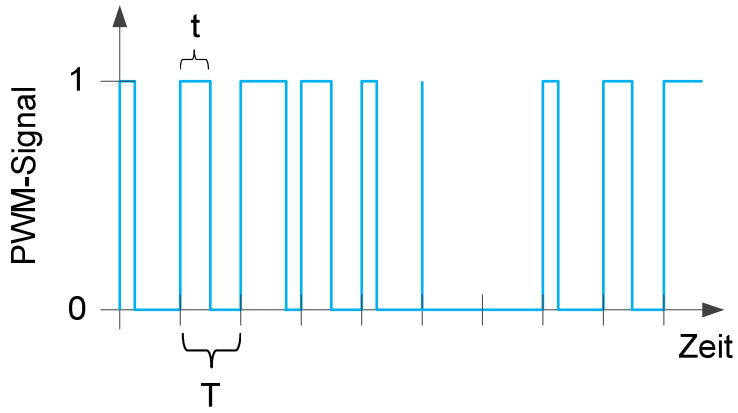

Abb. 48 Ein PWM-Signal mit einer festen Periode (T) und einer variablen Impulslänge (t).

In Abbildung 48 ist ein exemplarisches PWM Signal abgebildet. Ebenso sind die beiden wichtigen Größen der Periodendauer (*T*) und der Impulslänge (*t*) zu erkennen. Die Periodendauer eines PWM-Signals ist in den meisten Fällen konstant. Angepasst wird die Impulslänge. Aus der Periodendauer und der Impulslänge lässt sich die Einschaltdauer (duty cycle, *D*) in Prozent errechnen (siehe 6.1).

$$D = \frac{t}{T} \cdot 100\% \qquad (6.1)$$

In der Steuerungstechnik werden die induktiven oder kapazitiven Eigenschaften von elektrischen Geräten verwendet, um diese mit Hilfe eines PWM-Signals zu steuern. An dem Gerät stellt sich der arithmetische Mittelwert des Signals ein. Ein PWM-Signal mit einer Periodendauert T = 1 ms und einer Impulslänge von t = 0,5 ms würde an dem Gerät einem Eingangssignal von 0,5 entsprechen.
So lassen sich Motoren, welche sonst mit analogen Eingangssignalen gesteuert werden, direkt mit digitalen Signalen steuern ohne einen Digital-Analog-Wandler zu verwenden.

6.2 Musterlösung Versuch 7

Die benötigten Programme für die Ansteuerung der Aktoren über die beiden Module umfassen jeweils den entsprechenden Ausgang und ein Bedienelement, welches innerhalb einer zeitgesteuerten While-Schleife eingestellt werden kann. Auf eine Darstellung soll an dieser Stelle verzichtet werden.
Lediglich die Steuerung des digitalen Ausgangs des NI-9472 im PWM-Modus erfordert das erweiterte Wissen über PWM-Signale. Denn in diesen Modus wird der Ausgang über die Einschaltdauer gesteuert. Die Einschaltdauer ist als eine Zahl von 0 bis 100 (%) direkt an den Ausgang weiterzugeben.

Zu den gestellten Fragen sollen nun einige Antworten gegeben werden:

1. Welche Einschränkungen hinsichtlich der Steuerbarkeit von Motor und Ventilator zeigen sich?
    - Es ist keine Geschwindigkeits- und keine Steuerung der Drehrichtung möglich.
2. Warum kann das Peltier-Element nicht mit dem NI-9472 angesteuert werden?
    - Die minimale Versorgungsspannung für das NI-9472 beträgt 6 V. Da das Peltier-Element aber nur eine Spannung von 1,9 V verträgt ist eine Steuerung mit diesem Aufbau nicht möglich.
3. Welche Verbesserung der Steuerbarkeit kann mit dem PWM-Signal erreicht werden?
    - Der Motor ist nun in seiner Drehgeschwindigkeit steuerbar.
4. In welcher Hinsicht unterscheiden sich Motor und Ventilator?
    - Der Ventilator lässt sich entgegen dem Motor nicht in seiner Drehgeschwindigkeit steuern. Dies liegt aber an dem gewählten Ventilator. Eine temperaturgeregelte Steuerung von Ventilatoren mit PWM-Signalen ist sonst sehr verbreitet.
5. Welche Betriebsart wird für den Lautsprecher benötigt?
    - Der Lautsprecher muss mit einem PWM-Signal angesteuert werden. Aufgrund der Funktionsweise entsteht so, je nach Einstellung des Signals, ein Ton.

6. Welche Vorteile ergeben sich bei einer Ansteuerung von Aktoren mit einer Kombination aus NI-9403 und der Transistorschaltung?
    - Die Versorgungsspannung der Aktoren ist unabhängig von der Elektronik zur Erstellung der Steuersignale. An einem NI-9403 können, mit entsprechend vielen Aufbauten von Transistorschaltungen, beliebig viele Aktoren mit unterschiedlichen Anforderungen bezüglich der Eingangsspannungen betrieben werden.
7. Welche Nachteile gegenüber der Steuerung mit dem vorherig untersuchten Modul ergeben sich?
    - Es gibt keinen eingebauten Modus zum Senden von PWM-Signalen.
8. Kann das Peltier-Element mit diesem Modul und der Transistorschaltung angesteuert werden?
    - Da die Versorgungsspannung der Aktoren unabhängig vom Modul ist, kann das Peltier-Element mit diesem Aufbau angesteuert werden.

6.3 Versuch 8: Steuerung eines DC-Motors

Wie im letzten Laborversuch festgestellt wurde, ist die Steuerung eines DC-Motors mit den dort vorgestellten Mitteln noch nicht zufriedenstellend zu erreichen. Zwar konnte mit Hilfe des PWM-Signals die Drehgeschwindigkeit gesteuert werden, aber nicht die Drehrichtung. In diesem Versuch soll eine Möglichkeit untersucht werden, wie diese beiden Anforderungen gleichzeitig umgesetzt werden können. Dazu wird das Konzept einer H-Brücke erläutert und schließlich eine solche bereitgestellt.
Die H-Brücke wird an das cRIO-System angeschlossen. Für die Ausgabe von Signale steht das digitale Ein-/Ausgangs-, TTL-Modul NI-9403 zur Verfügung. Da dieses Modul keinen eingebauten PWM-Signal-Generator enthält, muss ein entsprechendes Programm geschrieben werden. Da für die Erstellung eines PWM-Signals ein gewisses Maß an Schnelligkeit und Genauigkeit notwendig ist, wird die FPGA im cRIO-Chassis für diese Aufgabe verwendet.
Das Programm soll zwei Ausgänge des NI-9403 schalten. Ein Ausgang soll das PWM-Signal weitergeben, der andere soll das Signal für die Richtung ausgeben. Diese beiden Ausgänge werden mit den entsprechenden Eingängen an der H-Brücke verbunden. Das erzeugte PWM-Signal soll mit einem Oszilloskop betrachtet werden.

1. Welcher Unterschied im Vergleich zum Versuch 3 (Informationsverarbeitungslabor) ist bei der möglichen maximalen Frequenz zu erkennen?
2. Woher kommt dieser Unterschied?
3. Was fällt an dem Ausgangssignal der H-Brücke auf?

H-Brücke

Der Name H-Brücke kommt von dem Aufbau einer solchen und der Ähnlichkeit mit dem Buchstaben H. In Abbildung 49 ist eine einfache H-Brücke bestehend aus vier Schaltern abgebildet.

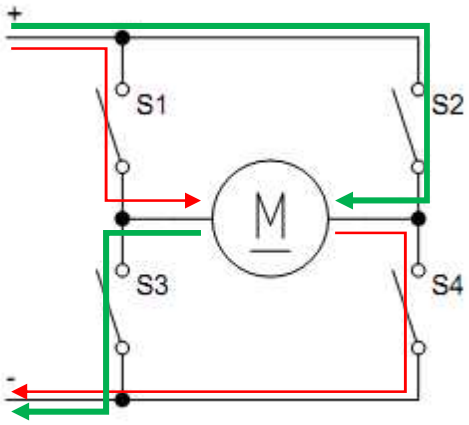

Abb. 49 H-Brücke mit vier Schaltern und einem Gleichstrommotor.

In der gezeigten H-Brücke würde das Schließen der Schalter *S1* und *S4* dazu führen, dass der Motor von links nach rechts mit Elektronen durchflossen wird (siehe Abbildung 49, roter Pfeil). Ein Schließen der Schalter *S2* und *S3* würde zu einer entgegengesetzten Fließrichtung der Elektronen führen (siehe Abbildung 49, grüner Pfeil). Wenn alle Schalter geöffnet sind fließt kein Strom durch den Motor, so kann durch gezieltes Öffnen der Schalter ein PWM-Signal erzeugt werden.

Die Elektrotechnik liefert solche H-Brücken als ICs, welche für verschiedenste Leistungsanforderungen verfügbar sind. In diesem Labor soll, der Anschaulichkeit halber, eine solche H-Brücke aus Transistoren vorgestellt werden. Die Transistoren übernehmen, wie schon in der Schaltung aus Laborversuch 4 für die Lichtschranke, die Funktion von Schalter.

In Abbildung 50 ist der Schaltplan der H-Brücke zu sehen. Die Transistoren *T1*, *T2*, *T5* und *T6* bilden die Schalter, wie sie zuvor in Abbildung 48 zu sehen waren. Die beiden zusätzlichen Transistoren dienen dem Zweck, dass die Transistoren, welche

ein „Schalterpaar“ bilden gleichzeitig geschaltet werden. Der Transistor *T3* schaltet die beiden Transistoren *T1* und *T6*. Der Transistor *T4* schaltet *T2* und *T5*. Mit Hilfe der Logikgatter *V1/1*, *V1/2* und *V1/3* können nun entsprechend der Eingangssignale die entsprechenden „Schalter“ gestellt werden, damit der Motor wie gewünscht läuft. Die Logikgatter sind in dem IC TC74HC02P [33] enthalten, welcher insgesamt vier NOR Gates enthält. Diese werden so hintereinander geschaltet, dass zu keinem Zeitpunkt beide „Schalterpaare“ gleichzeitig geöffnet sein können.

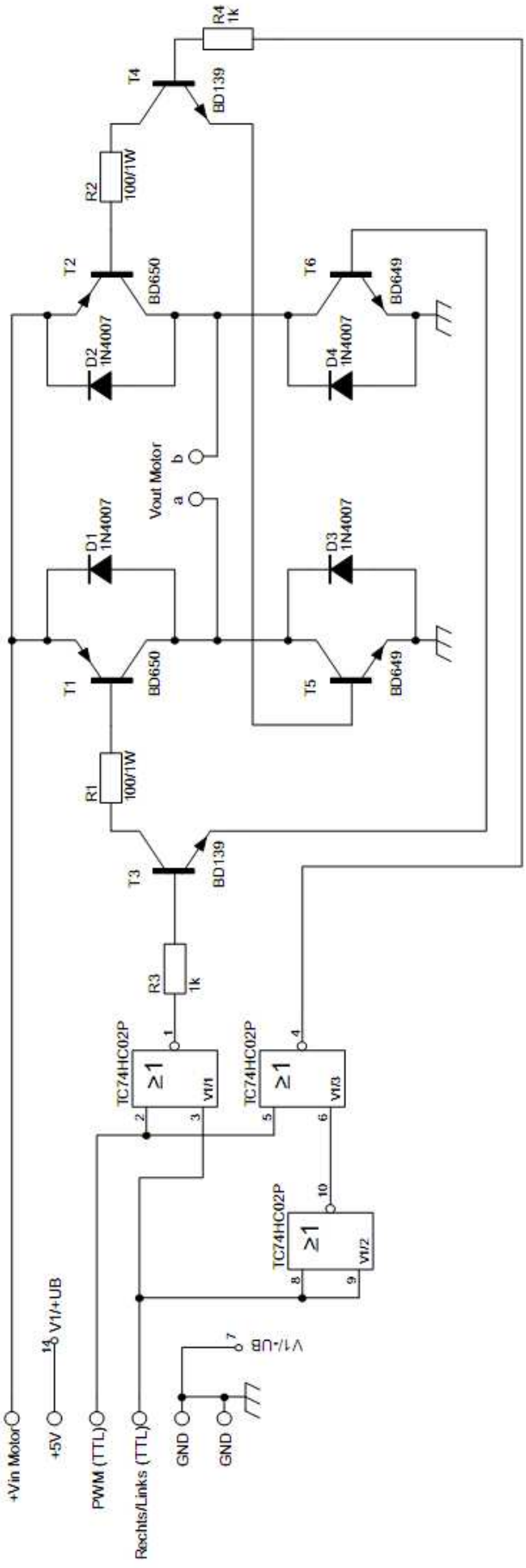

Abb. 50 Schaltplan der H-Brücke für die Motorsteuerung.

Abbildung 51 zeigt die aufgebaute H-Brücke. Links unten sind die Anschlüsse für *+Vin Motor*, *+5V*, *PWM (TTL)*, *Rechts/Links (TTL)* und zweimal *GND* (von links nach rechts) zu sehen. Rechts unten befinden sich die Anschlüsse $V_{out\ a}$ und $V_{out\ b}$ für den Motor. Der mittlere Anschluss ist nicht belegt. Zu Testzwecken kann an die Anschlüsse für den Motor auch ein Widerstand angeschlossen werden. Die sich ergebenden Spannungskurven mit angeschlossenem Wiederstand sind auf dem Oszilloskop deutlich besser erkennbar, als mit angeschlossenem Motor.

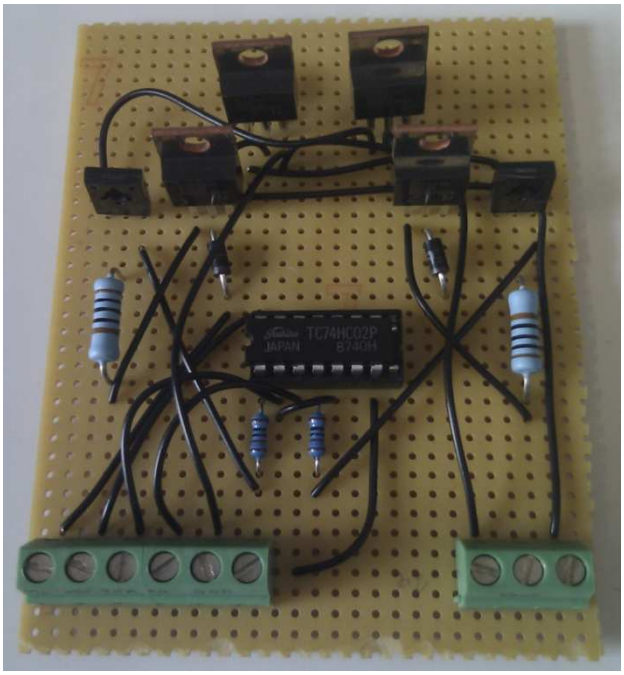

Abb. 51 Die H-Brücke für die Motorsteuerung.

Das FPGA-Programm

Um ein PWM-Signal mit der FPGA zu erzeugen können die beiden Funktionen *Loop Timer* und *Wait*, welche sich unter der Kategorie *Timing* finden, verwendet werden. Eine genaue Beschreibung der Funktionen kann der LabVIEW Hilfe entnommen werden. Ebenfalls sinnvoll ist die Verwendung einer *flachen Sequenzstruktur*.

6.4 Musterlösung Versuch 8

Für die Erzeugung des PWM-Signals wurde das in Abbildung 52 zu sehende Programm geschrieben. Die *Loop Timer* Funktion (siehe Abbildung 52, 1) bekommt die Periodendauer des PWM-Signals (in Ticks) übergeben. An die *Wait* Funktion

siehe Abbildung 52, 2) wird die *PWM-Laenge(Ticks)* übergeben. Beide Variablen sind in dem Cluster *PWM-Daten* zusammengefasst. In dieser Lösung wurden die Eingaben für die Steuerung des Signals in einem extra Programm getätigt. Für das Labor wäre es aber auch denkbar, die Eingaben direkt im FPGA Programm durchzuführen.

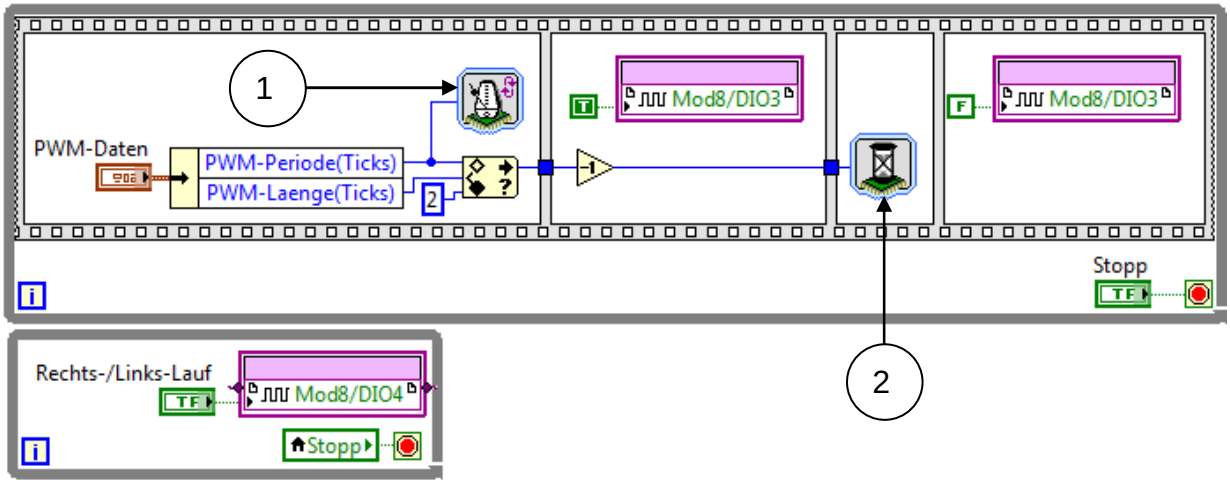

Abb. 52 Das Programm auf der FPGA für die Erzeugung eines PWM-Signals.

In Abbildung 53 ist das Host-Programm zu sehen. Die Kommunikation zwischen einem Host-Programm und einem FPGA-Programm wurden in Versuch 3 bereits durchgeführt. Hier soll nur noch einmal kurz die Funktionsweise der Blöcke geschildert werden.

Damit die Variablen im FPGA Programm vom Host aus manipuliert werden können, muss die *Funktion FPGA-VI-Referenz öffnen* (siehe Abbildung 53, 1) verwendet werden. Mit der Funktion *Lesen/Schreiben-Elemente* (siehe Abbildung 53, 2) können nun alle Bedienelemente der FPGA-VI mit Werten versehen werden. Die Periode des PWM-Signals in Ticks errechnet sich aus der Anzahl an Ticks, welche die FPGA pro Sekunde durchführt geteilt durch die gewünschte Frequenz. Die Anzahl der Ticks, der gesamten Periode mal eine Zahl von Null bis Eins (Einschaltdauer) ergibt die Länge des PWM-Signals in Ticks. Nachdem das Host-Programm mit dem Betätigen der Stopp-Taste beendet wird, wird auch die Verbindung zum FPGA-Programm mit der Funktion *FPGA-VI-Referenz schließen* (siehe Abbildung 53, 3) beendet.

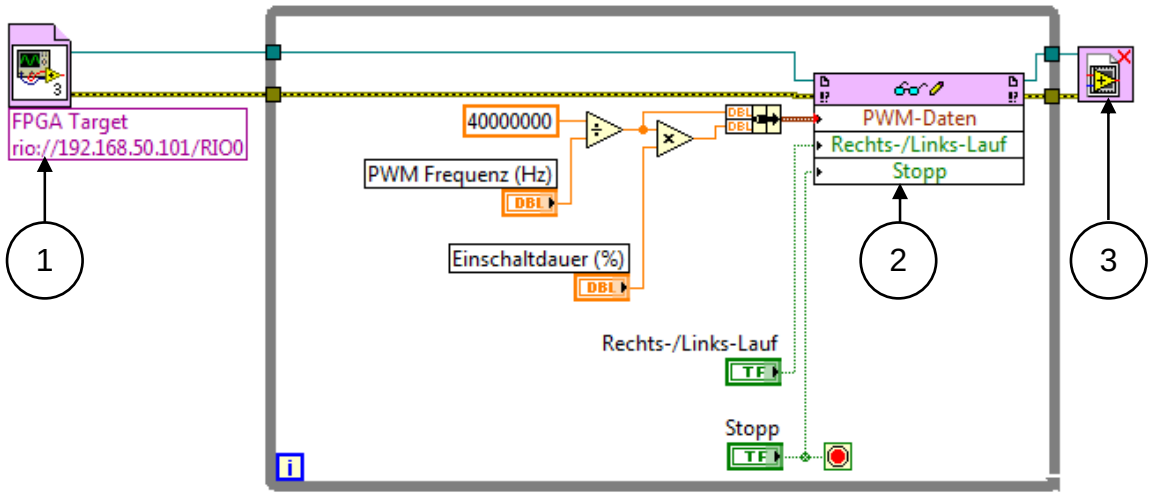

Abb. 53 Das Host-Programm, in welchem die Eingaben für die Steuerung des PWM-Signals getätigt werden können.

Zu den drei gestellten Fragen werden nun wieder mögliche Lösungen genannt.

1. Welcher Unterschied im Vergleich zum Versuch 3 (Informationsverarbeitungslabor) ist bei der möglichen maximalen Frequenz zu erkennen?
    - Die maximale Frequenz des Signals liegt um ein vielfaches höher. Es sind Schaltzeiten von weit über 10000 Hz möglich.
2. Woher kommt dieser Unterschied?
    - Das verwendete Modul NI-9403 weist im Vergleich zum NI-9472 eine Schaltverzögerung von maximal 330 ns auf (maximal 100 µs beim NI-9472).
3. Was fällt an dem Ausgangssignal der H-Brücke auf?
    - Das Signal ist hinsichtlich der eingestellten Einschaltzeit invertiert, siehe Abbildung 54. In der Abbildung sind zwei Signale zu erkennen. Das erste Signal (gelb) wurde direkt an den Ausgangsklemmen des NI-9403 gemessen. Das zweite Signal (hellblau) wurde an einen an die H-Brücke geschalteten Widerstand gemessen. Durch die verwendeten NOR Gates wird das Signal invertiert.

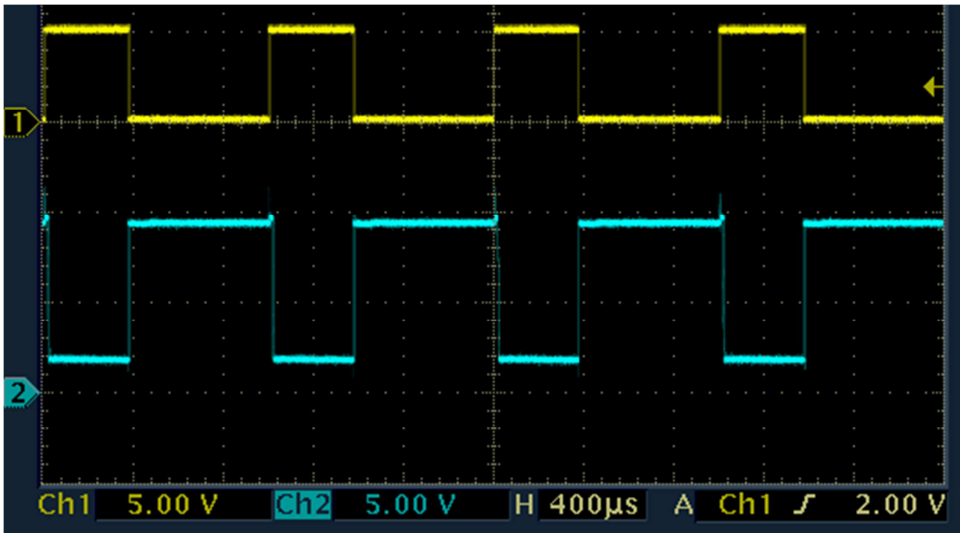

Abb. 54 Darstellung des Erzeugten PWM-Signals aus dem NI-9403 (gelb) und dem resultierenden Signal an den Ausgangsklemmen der H-Brücke (hellblau).

6.5 Versuch 9: Ansteuerung eines Schrittmotors

Schrittmotoren spielen in der Automatisierung von Prozessen eine, wenn nicht sogar die, zentrale Rolle. Sie machen zusammen mit hoch genauen Sensoren erst moderne automatisierte technische Anlagen wie CNC-Dreh oder -Fräsmaschinen möglich.

In diesem Versuch soll ein solcher Schrittmotor mit Hilfe des cRIO-Systems angesteuert werden. Für den Versucht steht ein vorbereitetes Programm für die FPGA bereit. Dieses muss an einigen Stellen ergänz werden.

Als Endergebnis soll das Programm den Schrittmotor in den beiden üblichen Betriebsarten Voll- und Halbschritt-Betrieb steuern können. Zudem sollen die Drehrichtung sowie die Geschwindigkeit eingestellt werden können. Für die Eingabe von Steuersignalen wird der in Versuch 4 verwendete Joystick benutzt. Der Schrittmotor wird mit Hilfe von zwei H-Brücken, wie sie in Versucht 7 vorgestellt wurde, betrieben. Die Kommunikation zwischen der Hardware und dem cRIO-Systems läuft über das ebenfalls schon verwendete Modul NI-9403.

Grundlagen zum Schrittmotor

Ein Schrittmotor ist ein Synchronmotor, bei dem der Rotor durch ein gezielt gesteuertes, sich schrittweise drehendes elektromagnetisches Feld der Statorspulen, um einen bestimmten Winkel gedreht wird. In Abbildung 55 ist ein einfacher Schrittmotor schematisch dargestellt.

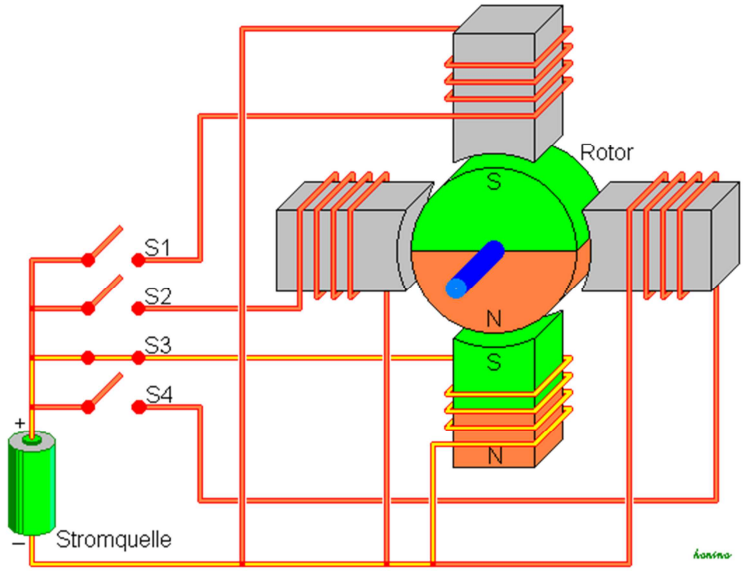

Abb. 55 Schematische Darstellung eines Schrittmotors mit vier Schritten für eine Umdrehung und unipolarer Beschaltung [27].

Durch das Schließen der Schalter *S1* bis *S4* wird in den vier Statorspulen ein elektrisches Feld erzeugt. Wenn die Schalter nacheinander geschaltet werden, dreht sich der Rotor entsprechend des äußeren magnetischen Feldes jeweils um 90°. In diesem Versuch wird nun eine spezielle Bauart von Schrittmotor verwendet. Die so genannten Reluktanzschrittmotoren besitzen einen gezahnten Weicheisenkern. Diese arbeiten nach dem Prinzip, das ein in einem Magnetfeld frei bewegliches magnetisches Materialien stets nach minimaler Reluktanz streben und sich entsprechend ausrichten wird. Reluktanz kann als magnetischer Widerstand bezeichnet werden. Dieser Effekt ist vergleichbar mit dem Ausrichten von Metallspänen in einem Magnetfeld. Bei dem verwendeten Motor (siehe Abbildung 56) werden an den Spulen magnetische Felder erzeugt. Der Rotor wird von diesem Magnetfeld durchflossen und richtet sich mit seinen Zähnen so aus, das minimale Reluktanz vorliegt. Durch die Anzahl der Zähne auf dem Rotor kann so die Schrittweite vorgegeben werden.

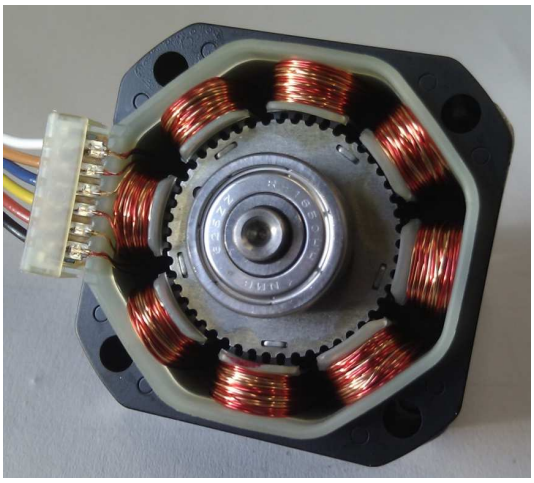

Abb. 56 Der geöffnete Schrittmotor. Zu sehen sind die acht Spulen, sowie der Weicheisenkern mit den Zähnen.

Aus dem Datenblatt zum Motor kann die interne Vorschaltung der Spulenpaare entnommen werden (siehe Abbildung 57). Der Motor besitzt sechs Anschlusskabel und soll später im bipolaren (im Gegensatz zum unipolaren Betrieb wechseln die Spulen im Betrieb ihre Polarität) Betrieb sowohl Halbschritte als auch Vollschritte durchführen können.

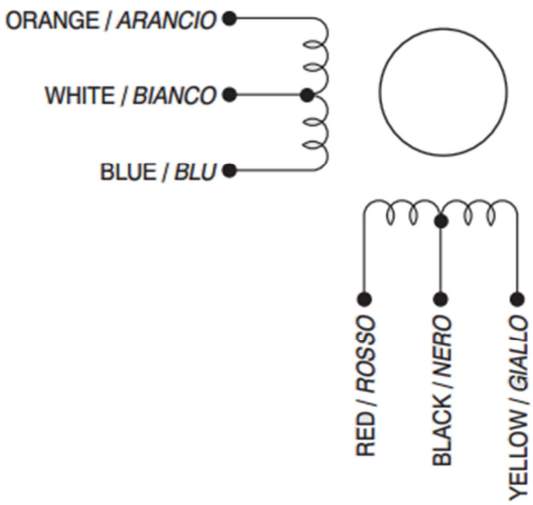

Abb. 57 Schaltung der Spulen im Schrittmotor [29].

Die erforderlichen Beschaltungen der Anschlüsse sind im Folgenden tabellarisch festgehalten.

Bipolare Ansteuerung Vollschritt						
	orange	blau	weiß	rot	gelb	schwarz
Schritt 1	+	-	0	+	-	0
Schritt 2	+	-	0	-	+	0
Schritt 3	-	+	0	-	+	0
Schritt 4	-	+	0	+	-	0
- steht für Minus; + steht für Plus; 0 steht für nicht belegt						

Tabelle 3 Ansteuerung der Spulen für den bipolaren Betrieb mit Vollschritten, nach [24].

Bipolare Ansteuerung Halbschritt						
	orange	blau	weiß	rot	gelb	schwarz
Schritt 1	+	-	0	+	-	0
Schritt 2	+	-	0	0	0	0
Schritt 3	+	-	0	-	+	0
Schritt 4	0	0	0	-	+	0
Schritt 5	-	+	0	-	+	0
Schritt 6	-	+	0	0	0	0
Schritt 7	-	+	0	+	-	0
Schritt 8	0	0	0	+	-	0
- steht für Minus; + steht für Plus; 0 steht für nicht belegt						

Tabelle 4 Ansteuerung der Spulen für den bipolaren Betrieb mit Halbschritten, nach [24].

Die beiden Spulen werden an je einer H-Brücke angeschlossen. Prinzipiell spielt es keine Rolle wie die Spulen mit den H-Brücken verbunden werden. Dies könnte später in der Programmierung angepasst werden. Für diesen Versuch ist die Beschaltung in Tabelle 5 vorgegeben.

Anschluss Schrittmotor	**Anschluss H-Brücke**
Orange	H-Brücke 1 Ausgangsklemme b (*H1b*)
Blau	H-Brücke 1 Ausgangsklemme a (*H1a*)
Weiß	Nicht verbunden
Rot	H-Brücke 2 Ausgangsklemme b (*H2b*)
Gelb	H-Brücke 2 Ausgangsklemme a (*H2a*)
Schwarz	Nicht verbunden

Tabelle 5 Anschluss des Schrittmotors an den H-Brücken.

Damit die Spulen wie in Tabelle 3 und 4 festgehalten beschaltet werden können, müssen die H-Brücken mit den richtigen Signalen versorgt werden. Zur Erinnerung (siehe Tabelle 6): ein Signal auf dem Eingang *PWM(TTL)* sorgt dafür, dass keine Spannung an den Ausgangklemmen anliegt. Ein Signal am Eingang *Rechts/Links(TTL)* ändert die Polarität der Ausgänge.

Anschluss an der H-Brücke	**Eingangssignal**	**Resultierendes Ausgangssignal**
PWM(TTL)	false	Spannung
	true	Keine Spannung
Rechts/Links(TTL)	false	Ausgangsklemme *b* Minus
	true	Ausgangsklemme *a* Minus

Tabelle 6 Verhalten der H-Brücke bei verschiedenen Eingangssignalen.

Mit Hilfe den Tabellen 3 bis 6 kann ein Signal für die H-Brücken programmiert werden, damit die geforderten Beschaltungen der Spulen für den Voll- und Halbschrittbetrieb eingehalten werden. Hier soll für den Vollschritt Betrieb der erste Schritt exemplarisch erarbeitet werden.

Gefordert ist nach Tabelle 3: Plus an den Anschlüssen *orange* und *rot,* sowie Minus an den Anschlüssen *blau* und *gelb*. Nach Tabelle 5 ergibt sich demnach Plus an den Anschlüssen *H1b* und *H2b* sowie entsprechend Minus an den Anschlüssen *H1a* und *H2a*. Das Signal welches an die beiden H-Brücken gesendet werden muss, kann mit Tabelle 6 abgeleitet werden. Danach ergibt sich das in Tabelle 7 in der Zeile „Schritt 1“ zu sehende Signal.

Signal an die H-Brücken für Vollschrittbetrieb				
	H-Brücke 1		**H-Brücke 2**	
	PWM(TTL)	*Rechts/Links(TTL)*	*PWM(TTL)*	*Rechts/Links(TTL)*
Schritt 1	false	true	false	true
Schritt 2				
Schritt 3				
Schritt 4				

Tabelle 7 Signalfolge für den Vollschrittbetrieb des Schrittmotors.

Signal an die H-Brücken für Halbschrittbetrieb				
	H-Brücke 1		**H-Brücke 2**	
	PWM(TTL)	*Rechts/Links(TTL)*	*PWM(TTL)*	*Rechts/Links(TTL)*
Schritt 1				
Schritt 2				
Schritt 3				
Schritt 4				
Schritt 5				
Schritt 6				
Schritt 7				
Schritt 8				

Tabelle 8 Signalfolge für den Halbschrittbetrieb des Schrittmotors.

Für die spätere Programmierung werden die Signale noch benötigt, deshalb sollen die Tabellen 7 und 8 vollständig ausgefüllt werden.

Eingaben am Joystick

Für die Steuerung des Schrittmotors wird der gleiche Joystick verwendet, wie er bereits in Versuch 4 benutzt wurde. Die fünf Eingabemöglichkeiten (siehe Abbildung 30) werden im Programm mit verschiedenen Funktionen belegt (siehe Tabelle 9).

Eingabe	Funktion
Vor	Geschwindigkeit erhöhen
Zurück	Geschwindigkeit verringern
Links	Schritt nach links
Rechts	Schritt nach rechts
Aktion	Umschalten zwischen kontinuierlichem und schrittweisem Betrieb

Tabelle 9 Übersicht der Eingaben am Joystick und deren Funktionen.

Vorstellung des Programms und weitere Programmieranweisungen

Wie bereits erwähnt wurde das Programm für die FPGA vorbereitet. Es besteht im wesentlichem aus drei Teilen: der Schrittsteuerung, dem Tickgeber und der Geschwindigkeitssteuerung. Der Tickgeber (siehe Abbildung 58) sorgt dafür, dass die Schrittsteuerung einen weiteren Schritt durchführt. Die Geschwindigkeitssteuerung regelt die Wiederholrate der Tickgeber-Schleife im kontinuierlichen Betrieb. Nachfolgend werden die drei Teile des Programms nacheinander erläutert und weitere Programmieranweisungen gegeben.
In der Schleife des Tickgebers wird zwischen kontinuierlichem und schrittweisem Betrieb umgeschaltet.

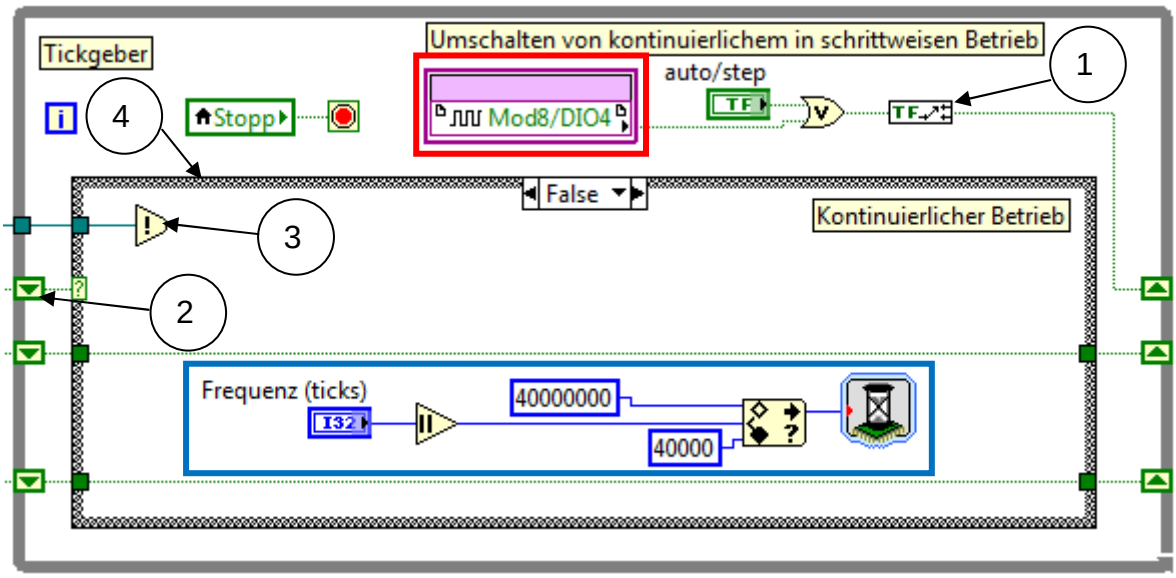

Abb. 58 Die Tickgeber-Schleife im FPGA Programm, zum Steuern eines Schrittmotors.

Die Umschaltung erfolgt entweder über eine Schaltfläche auf dem Frontpanel des Programms oder per Eingabe am Joystick (siehe Abbildung 58, roter Kasten). Im Labor muss die Schnittstelle angepasst werden. Wie bereits beschrieben soll ein Druck auf den Aktion Schalter am Joystick eine Umschaltung bewirken. Das Signal des Schalters oder der Schaltfläche werden einem Toggle-VI (Abbildung 58, 1) übergeben. Anschließend wird der Wert in ein Schieberegister (Abbildung 58, 2) geschrieben, dessen Wert über die Betriebsart bestimmt. Sollte der kontinuierliche Betrieb gewählt sein, wird gemäß der aktuell eingestellten *Frequenz (ticks)* die Schleife erneut ausgeführt (Abbildung 58, blauer Kasten) und mittels der *Occurrence festlegen* Funktion (Abbildung 58, 3) die wartende Schleife zur Schrittsteuerung aktiviert. Im schrittweisen Betrieb schaltet die *Case-Struktur* (Abbildung 58, 4) auf *True* um (siehe Abbildung 59). In dieser Struktur soll eine Logik eingebaut werden, welche beim Betätigen der zuvor festgelegten Schalter am Joystick ein *true* an die *Case-Struktur* (Abbildung 59, 1) zum Aktivieren der *Occurrence festlegen* Funktion übergibt.

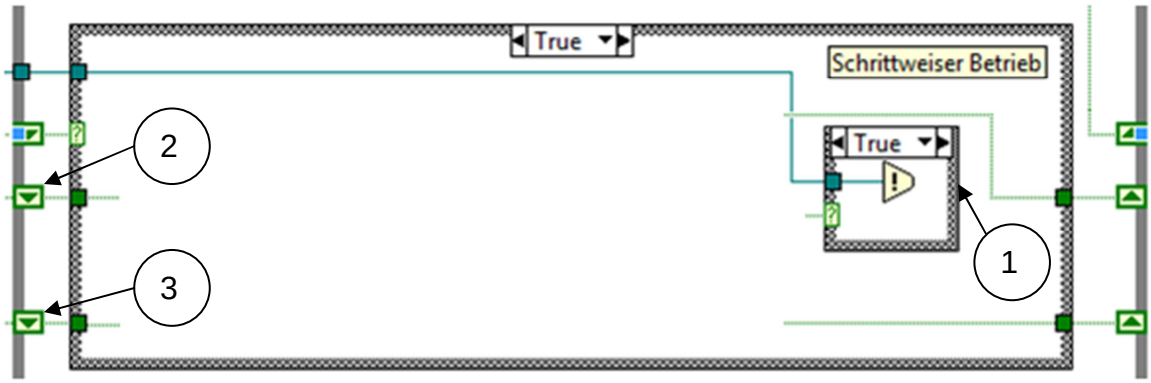

Abb. 59 Die Case-Struktur für den schrittweisen Betrieb.

Zu diesem Zweck sind bereits zwei Schieberegister (Abbildung 59, 2 und 3) vorhanden, welche die Werte der Schalter aus dem letzten Durchlauf der Schleife enthalten sollen. Die Case Struktur (Abbildung 59, 1) soll nur ausgeführt werden, wenn eine Wertänderung von *false* auf *true* vorliegt, entweder für den einen oder den anderen Schalter. Zusätzlich können zu Testzwecken auch zwei Schaltflächen für das Frontpanel eingebaut werden.

In der Schleife für die *Schrittsteuerung* (siehe Abbildung 60) werden die programmierten Zustände für jeden Schritt an Anschlüsse am NI-9403 weitergegeben. Es hat sich als sinnvoll herausgestellt mit Schrittnummern zu arbeiten, da so auch eine schrittweise Ansteuerung im manuellen Betrieb mit Richtungswechseln flüssig möglich ist. Für die Speicherung der letzten Schrittnummer wird ein Schieberegister verwendet. In dieser Schleife wird prinzipiell nach Voll- und Halbschritt-Betrieb unterschieden. Eine Umschaltung erfolgt über die Schaltfläche *half/fullstep* (Abbildung 60, 1). In der *Case-Struktur* „Schritt +/-1“ (Abbildung 60, 2) wir nach einer zu programmierenden Logik die Schrittnummer um Eins erhöht oder verringert. Die Logik muss die bereits vorhandenen (Abbildung 60, 5) und die Anschlüsse des NI-9403 verwenden, an denen die *Schritt nach links/rechts* Schalter des Joysticks angeschlossen sind. Nachdem die Schrittnummer erhöht oder verringert wurde durchläuft sie eine Kontrolle (Abbildung 60, 3) in der je nach Betriebsart ein Sprung, je nach Drehrichtung, von 3 auf 0 oder von 0 auf 3 für den Vollschritt-Betrieb oder von 7 auf 0 oder 0 auf 7 für den Halbschritt-Betrieb durchgeführt wird.

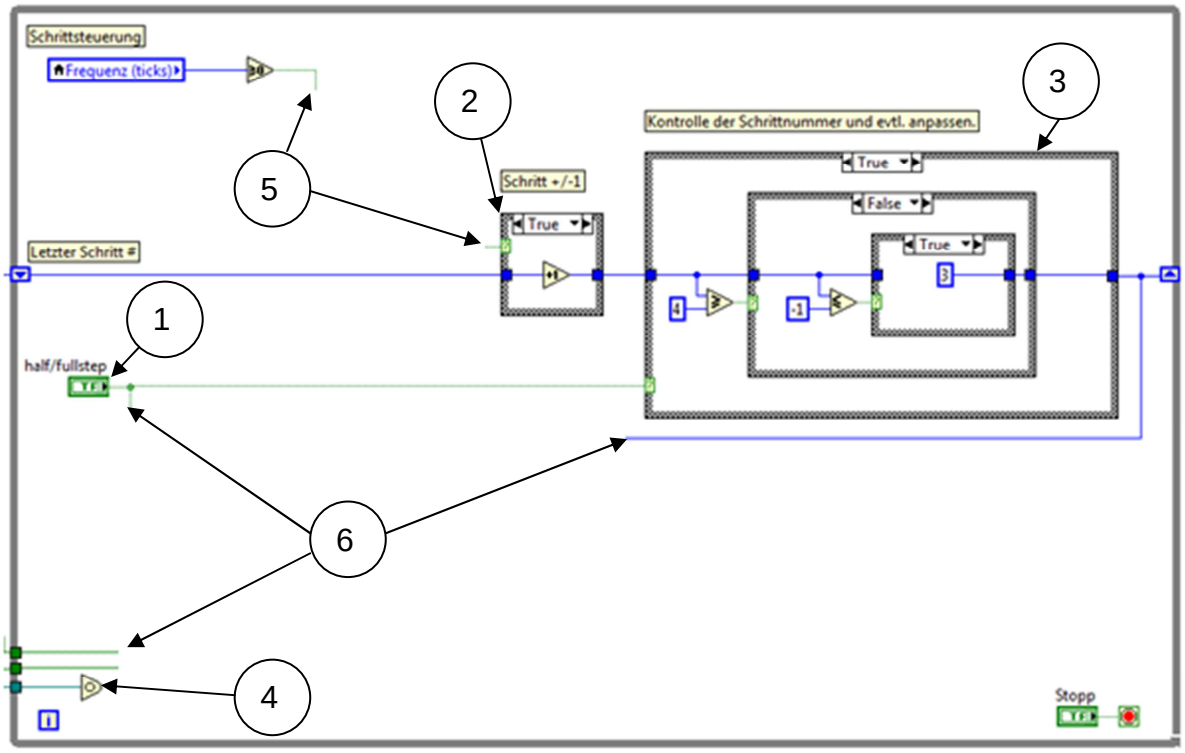

Abb. 60 Schleife für die Schrittsteuerung.

Hinter dieser Kontrolle wird die aktuelle Schrittnummer abgegriffen. Zusammen mit den beiden Anschlüssen der Zustände und der Schaltfläche (Abbildung 60, 6) soll eine Übergabe der Zustände an die Anschlüsse am NI-9403 für die Steuerung der H-Brücken programmiert werden.

Die Funktion *Auf Occurrence warten* (Abbildung 60, 4) lässt ein Ausführen der Schleife erst zu, wenn in der Tacktgeber-Schleife die zuvor beschriebene Funktion *Occurrence festlegen* aufgerufen wurde.

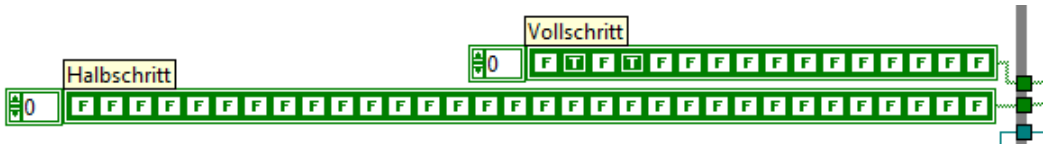

Abb. 61 Arrays mit den Zustandsinformationen für die H-Brücken im Vollschritt- und Halbschritt-Betrieb.

Außerhalb der Schleife finden sich die beiden Arrays *Vollschritt* und *Halbschritt* (siehe Abbildung 61). In diesen müssen die Zustände aus den Tabellen 7 und 8 eingetragen werden. Dazu können die booleschen Variablen entsprechend der Zeilen der Tabellen nacheinander eingestellt werden. Für das Array Vollschritt wurde exemplarisch die erste Zeile der Tabelle bereits eingetragen.

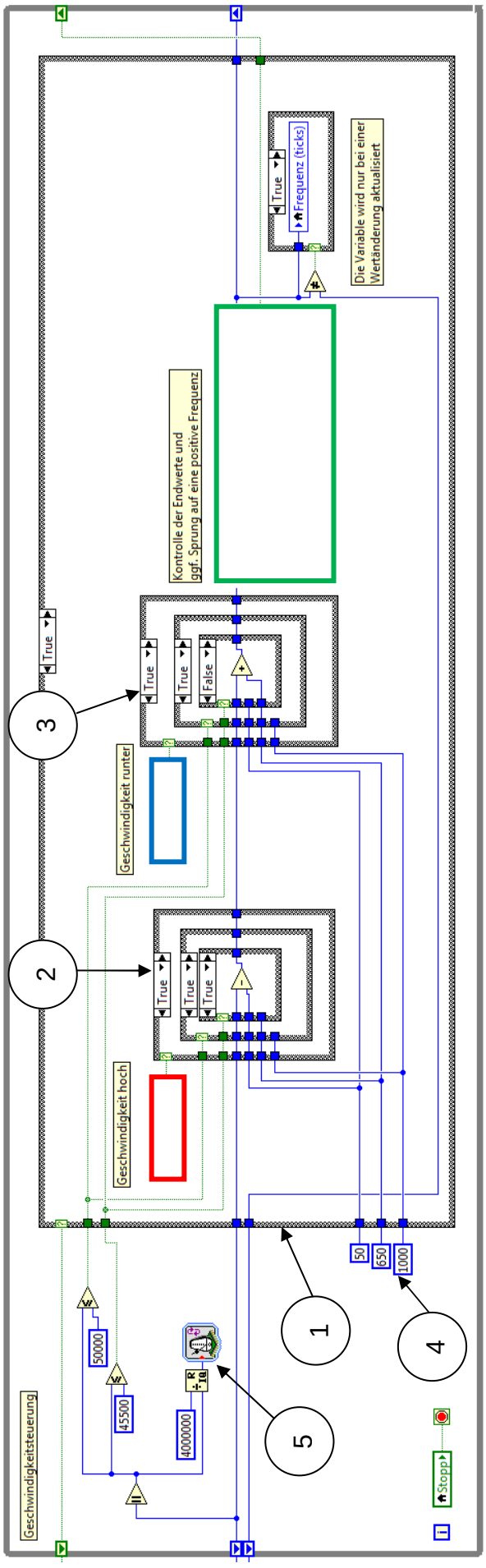

Abb. 62 Die Geschwindigkeitssteuerung.

Der dritte Teil des Programms, die *Geschwindigkeitssteuerung* (siehe Abbildung 63), setzt Eingaben am Joystick in eine Erhöhung oder Verringerung der Drehgeschwindigkeit um. Da das Programm mit einer Angabe in Ticks arbeitet ist das Einstellen der richtigen Werte mit einigen Überlegungen verbunden. In Abbildung 62 sind die Zusammenhänge zwischen Drehrichtung, -geschwindigkeit, Frequenz in Herz und Ticks zu sehen. Es sind zwei Zustände des Motors eingezeichnet (Abbildung 62, blaue und rote Linie), an denen die zugehörigen Befehle zum Verringern und Erhöhen der Drehgeschwindigkeit angetragen sind. Im Zustand der roten Linie soll mit dem Betätigen des Schalters, mit der Funktion *Geschwindigkeit verringern,* die Drehgeschwindigkeit in negative Richtung weiter bis zu einem Maximalwert erhöht werden. Ein Betätigen des Schalters *Geschwindigkeit erhöhen* soll die Drehgeschwindigkeit verringern.

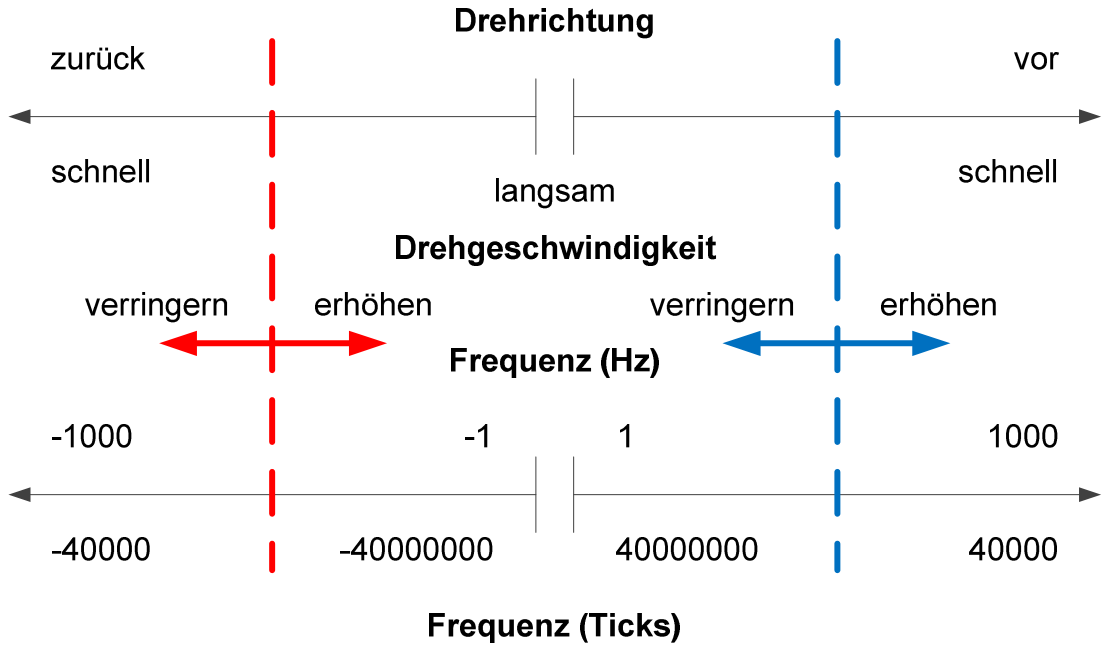

Abb. 63 Zusammenhänge zwischen Drehrichtung, -geschwindigkeit und Frequenz in Herz und Ticks bei der Schrittmotorsteuerung.

Im Falle des Zustands der blauen Linie kehren sich diese Befehle um. Ein Betätigen des Schalters mit der Funktion *Geschwindigkeit verringern* soll nun die Drehgeschwindigkeit verringern und der Schalter *Geschwindigkeit erhöhen* zu einer Zunahme der Drehgeschwindigkeit bis zu einem Maximalwert führen.

Die Unterscheidung zwischen negativen und positiven Frequenzen dient in der Schleife zur Schrittsteuerung für die Auswahl, ob ein Schritt vor oder zurück durchgeführt werden soll (siehe Abbildung 60, 5). Aus der Abbildung 62 lassen sich

die zu programmierenden Sprünge für die Frequenz in Ticks bei einem Richtungswechsel ablesen. Ebenso sind die Maximalwerte in der negativen und positiven Drehrichtung zu sehen. Für die Unterscheidung zwischen Rechts- und Linkslauf ist in der Schleife eine *Case-Struktur* (Abbildung 63, 1) vorhanden. In dieser wird je nach Fall, entsprechend der Abbildung 62, mit den Steuersignalen des Joysticks umgegangen. In der *Case-Struktur* (Abbildung 63, 1) sind einige Programmteile vorgegeben. Es befindet sich dort je eine *Case-Struktur* (Abbildung 63, 2 und 3), die die *Frequenz (ticks)* um einen Wert erhöht oder verringert. Zum Aufrufen dieser Strukturen sind die Anschlüsse des NI-9403 einzubauen (Abbildung 63, roter und blauer Bereich), welche mit den Funktionen *Geschwindigkeit erhöhen* und *Geschwindigkeit verringern* des Joysticks verbunden sind, dabei muss die Abbildung 62 beachtet werden.
Hinter den beiden Strukturen zum Addieren und Subtrahieren (Abbildung 63, grüner Bereich) soll eine Kontrolle der *Frequenz (ticks)* durchgeführt und der Sprung von -40000000 auf 40000000 Ticks programmiert werden. Es muss ebenfalls eine boolesche Variable ausgegeben werden, welche zwischen positivem und negativem Frequenzbereich umschaltet. Es ist darauf zu achten, dass diese Kontrolle und der Sprung in beiden Fällen der äußeren *Case-Struktur* (Abbildung 63, 1) programmiert werden muss.
Die weitere Funktionalität dieser Schleife umfasst, ein je nach aktueller Drehgeschwindigkeit angepasstes addieren oder subtrahieren von Werten (Abbildung 63, 4) zur *Frequenz (ticks).* Mit den gegebenen Werten zum Addieren und Subtrahieren kann im Labor experimentiert werden. Für eine bessere Bedienung wurde die Wiederholrate der Schleife mit der aktuellen Drehgeschwindigkeit des Motors verknüpft (Abbildung 63, 5).

6.6 Musterlösung Versuch 9

In diesem Versuch mussten einige Grundlagen zu Schrittmotoren erarbeitet werden. Hierzu gehörte die Umsetzung der geforderten Beschaltungen der Spulen, auf Schaltzustände der H-Brücken. Die ausgefüllten Tabellen 7 und 8 finden sich im Anhang 10.3. Im Halbschrittbetrieb waren bei Schritt 2, 4, 6 und 8 die Spulen als nicht belegt angegeben. Dies heißt, dass in diesem Schritt keine Spannung anliegen soll. Das Signal *PWM(TTL)* ist in diesen Fällen als *true* zu setzen. Wogegen das

Signal *Rechts/Links(TTL)* beliebig gewählt werden kann. Entsprechend der Tabellen 13 und 14 (siehe Anhang 10.3) wurden die Arrays *Vollschritt* und *Halbschritt* eingestellt (siehe Abbildung 64).

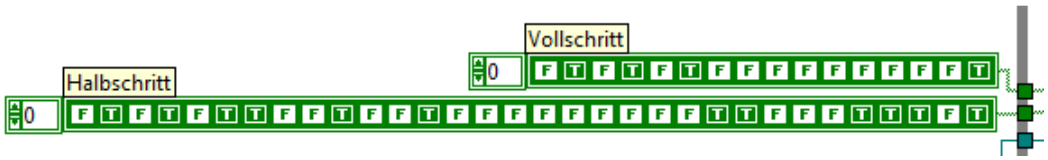

Abb. 64 Die ausgefüllten Arrays mit den Zustandsinformationen für die H-Brücken im Vollschritt- und Halbschritt-Betrieb.

In der Schrittsteuerung waren zwei Programmteile zu ergänzen. Diese sind in Abbildung 65 und 66 zu sehen.

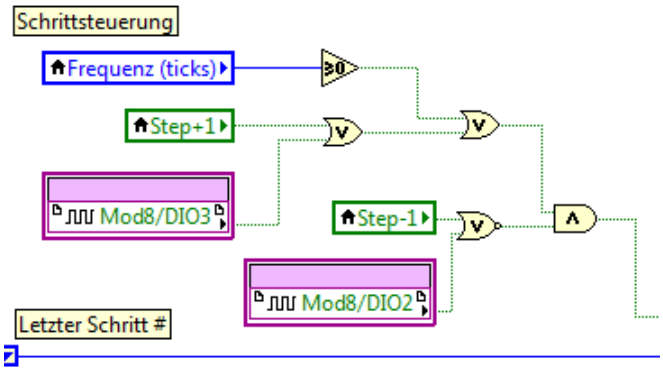

Abb. 65 Die Logik für die Auswahl ob ein Schritt addiert oder subtrahiert werden soll.

Für die Auswahl ob ein Schritt addiert oder subtrahiert werden soll muss die Abfrage *Frequenz (ticks)* ≥*0* mit den Eingangssignalen des Joysticks zusammengefasst werden. Ein Schritt wird dann addiert wenn:

$$\begin{aligned}&\big(Frequenz\ (ticks) \geq 0 \vee (Step + 1 \vee Joystickschalter\ Rechts)\big)\\ &\qquad \wedge \neg\ (Step - 1 \vee Joystickschalter\ Links)\end{aligned} \tag{6.2}$$

ein *true* ausgibt andernfalls wird ein Schritt abgezogen.

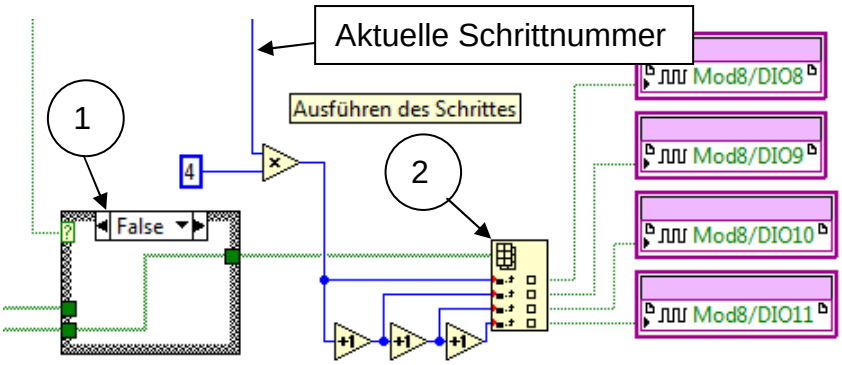

Abb. 66 Die Übergabe der Werte aus den Arrays an die Ausgänge des NI-9403.

Für die Übergabe der Werte aus den Arrays wurde eine Case-Struktur (Abbildung 66, 1) eingebaut, welche je nach Betriebsart, das Array Vollschritt oder Halbschritt zum Auslesen der zu setzenden Zustände auswählt. Die Werte wurden schließlich mit der Funktion *Array indizieren* (Abbildung 66, 2) ausgegeben. Diese Funktion gibt entsprechend der zugeführten Werte die jeweiligen Elemente eines Arrays aus. Für den ersten Schritt (Schrittnummer 0) werden gemäß der Programmierung die Werte der Elemente 0 bis 3 ausgegeben.

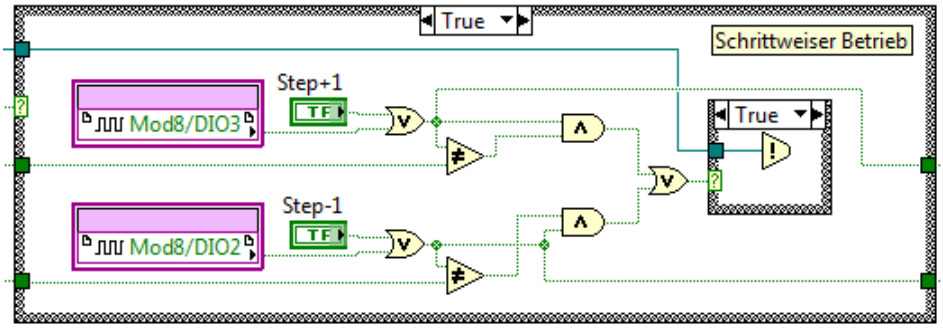

Abb. 67 Die Logik für die Tickgeber-Schleife, im schrittweisen Betrieb.

In der Tickgeber-Schleife sollte ebenfalls eine Logik programmiert werden. Diese ist in Abbildung 67 zu sehen. Es werden die Werte aus dem letzten Durchlauf der Schleife mit den aktuellen an den Eingängen des NI-9403 verglichen. Bei einer Wertänderung und wenn der aktuelle Wert *true* ist wird die *Occurrence festlegen* Funktion ausgeführt.

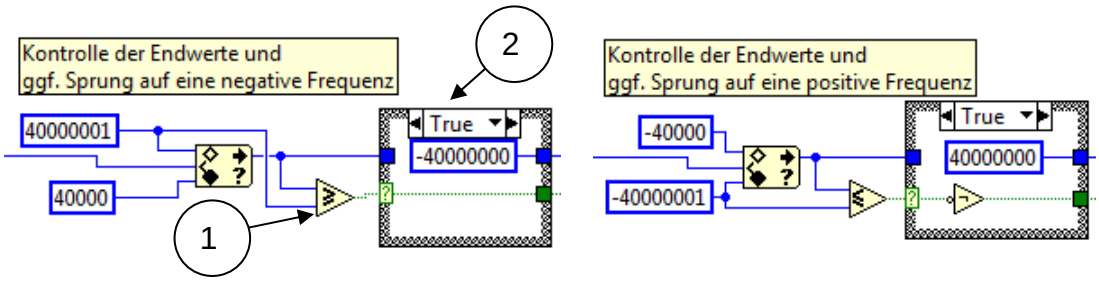

Abb. 68 Darstellung der fertigen Programmteile der Geschwindigkeitssteuerung. Links für den positiven und rechts für den negativen Frequenzbereich.

In Abbildung 68 sind die zu programmierenden Teile der Geschwindigkeitssteuerung abgebildet. Für den Übergang wurde ein Vergleich des aktuellen Frequenzwertes mit einen Maximalwert vorgenommen. Der Maximalwert der Frequenz kann entweder -40000001 oder 40000001 annehmen. Sollte dies der Fall sein (siehe Abbildung 68, 1) wird auf den jeweils anderen Bereich gesprungen. Dazu wird die Frequenz mit einem festgelegten Wert versehen und die geforderte boolesche Variable ausgegeben (Abbildung 68, 2), welche die äußere Case Struktur (Abbildung 63, 1) umschaltet.

Wenn alle Programmteile richtig programmiert wurden lässt sich der Schrittmotor mit Hilfe des Joysticks in seiner Geschwindigkeit und Drehrichtung steuern. Außerdem kann er im schrittweisen Betrieb, mit einzelnen Schritten in eine beliebige Richtung, bewegen werden.

7 Fazit

Das Ziel dieser Arbeit war es geeignete Versuche für das Mechatronik Labor an der HAW-Hamburg zu entwickeln. Dafür wurde der Begriff der Mechatronik im ersten Teil der Arbeit erläutert und auf dieser Grundlage ein Laborkonzept entworfen. Zum besseren Verständnis der komplexen Zusammenhänge bei mechatronischen Systemen wurden die Labore aus mehreren kleineren Versuchen zusammengesetzt, die thematisch zusammengehören. Zudem bauen die einzelnen Versuche aufeinander auf und gewährleisten eine Vertiefung des Erlernten aus den vorherigen Modulen. Die Versuche wurden unter die drei Überschriften Informationsverarbeitung, Sensorik und Aktorik entwickelt.

Für die Laborversuche wurden geeignete Versuchsaufgaben, sowie die dazu gehörigen Hardwarekomponenten und Programme erarbeitet. Nach einer Betrachtung der Anforderungen an die Informationsverarbeitung wurde für die Mechatronik Laborveranstaltung Hard- und Software von NI gewählt. Die Auswahl von LabVIEW hat zu einigen interessanten Herausforderungen für diese Arbeit geführt. Denn die Verwendung von LabVIEW und der dazu gehörigen Hardware sind an der HAW-Hamburg zum Zeitpunkt der Erstellung dieser Arbeit noch nicht sehr verbreitet. Die Versuche wurden so aufgebaut, dass sich ein einfacher Einstieg in die LabVIEW Programmierung ergibt. Durch die grafische Programmierung, welche einem Flussdiagramm sehr nahe kommt, wurde dieses Vorgehen positiv unterstützt.

In den Laborversuchen 1 bis 3 ist den Studierenden die verwendete Hardware schrittweise erklärt worden. Mit geeigneten Programmen wurden die Leistungsgrenzen des cRIO-Systems aufgezeigt und Grundlagen der Programmierung von embedded systems vermittelt.

Für die Industrie besonders interessant, ist der im cRIO-Chassis verbaute FPGA Chip. Zu diesem Chip wurden im dritten Labor wichtige Informationen gegeben, sowie die Programmierung an einem leicht verständlichen Beispiel verdeutlicht.

Für die Versuche in der zweiten Laborveranstaltung (Sensorik) wurden im Kapitel 2.2 einige geeignete Sensoren ausgewählt. Es werden typische Anwendungsfälle bearbeitet, so dass die Studierenden einen guten Einblick in die Verwendung unterschiedlicher Sensoren erhalten.

Für den praktische Anteil des Mechatroniklabors sind einige Schaltungen entwickelt worden, die sich in verschiedenen Versuchen vielfältig einsetzten lassen. Diese

Schaltungen werden von den Studierenden eigenständig aufgebaut oder fertig bereitgestellt.
In den letzten Laborversuchen 7 bis 9 werden schließlich Steuerungen von Aktoren durchgeführt. Für die Aktoren wird die Steuerungsmethode der Pulsweitenmodulation mit anschaulichen Beispielen gezeigt. Im letzten Versuch wird abschließend eine Steuerung, samt notwendiger Hardware für einen Schrittmotor untersucht. Den Studierenden wir in diesem Versuch die große Bedeutung von Schrittmotoren in der Automatisierung vermittelt. Mit der gestellten Aufgaben wir zudem auf die relevanten Steuerungsmethoden von Schrittmotoren eingegangen.
Das zum Beginn der Arbeit formulierte Ziel, drei Versuche für drei Laborveranstaltungen zu entwickeln, ist mit den vorgestellten neun Versuchen erfolgreich erreicht worden. Durch eine Verteilung von, für den letzten Versuch notwendiger Komponenten, ist auch die Forderung, nach aufeinander aufbauenden Versuchen erfüllt worden.

8 Ausblick

Mit dem entwickelten Labor für die Mechatronikvorlesung werden die wichtigsten Grundlagen mechatronischer Systeme den Studierenden anschaulich vermittelt. Die Versuche bieten viele Ansatzpunkte für einen weiteren Ausbau des Labors.

In den Versuchen wurden offenen Regelkreise entwickelt. Die Aufgaben lassen sich mit gezielten Anpassungen zu geschlossenen Regelkreisen umwandeln. Dabei sind vor allem die Lichtschranke und der Temperatursensor, bei den Sensoren und das Peltier-Element und der Schrittmotor, bei den Aktoren zu nennen. Diese können mit nur wenig Aufwand für den Aufbau eines geschlossenen Regelkreises verwendet werden. Für den Temperatursensor kann das Peltier-Element, zusammen mit der in Versuch 4 vorgestellten Transistorschaltung, als Aktor verwendet werden. So ergibt sich ein System, in dem die Temperatur als Regelgröße gesteuert werden kann.

Für die Lichtschranke ergeben sich im Zusammenspiel mit dem Schrittmotor einige interessante Ideen. So kann diese z.B. für die Geschwindigkeitsregelung eingesetzt oder die Anzahl der Schritte gezählt werden. Eine dafür erforderliche Lochscheibe ist ebenfalls schon vorhanden. Der Lautsprecher könnte so in das Programm der Schrittmotorsteuerung integriert werden, das er z.B. bei jedem Richtungswechsel einen Signalton abgibt. Die Möglichkeiten zur Erweiterung der Versuche sind nur durch den damit verbundenen Zeitaufwand im Labor begrenzt. Die Steuerung des Schrittmotors lässt sich auch durch die Verwendung von ICs vereinfachen, womit sich anstelle der Erzeugung, mehr Zeit für die Anwendung der Bewegung ergibt. Eine Regelungsaufgabe kann damit einfacher im Zeitrahmen eines Labors durchgeführt werden.

Die Informationsverarbeitung wurde in dieser Arbeit aus den genannten Gründen mit Hard- und Software von NI durchgeführt. Wie in Kapitel 2.3 beschrieben gibt es eine ganze Reihe von anderen Systemen. Der große Bereich der Mikrocontroller für embedded-systems wird in noch keiner Veranstaltung behandelt. Hier kann es sinnvoll sein, diese auch in das Labor mit aufzunehmen.

Die vielen weiteren möglichen Anschlussprojekte und Erweiterungen des Labors lassen sich gut in der vierten Laborveranstaltung unterbringen.

9 Literaturverzeichnis

[01] 3S-Smart Software Solutions: *CODESYS – IEC 61131-3 Development Tool for Industrial Controllers.* Stand 2013. http://www.codesys.com/products/codesys-engineering/development-system.html (abgerufen am 16.02.2013)

[02] Arduino: *Arduino - an open-source electronics prototyping platform*. Stand 11.02.2013. http://www.arduino.cc/ (abgerufen am 16.02.2013)

[03] Jens von Aspern: *SPS Grundlagen – Aufbau, Programmierung (IEC 61131, S7), Simulation, Internet, Sicherheit*. Hüthing Verlag, Heidelber u.a. 2009. ISBN-13: 978-3-7785-4060-2

[04] Torsten Bertram: *Entwicklungsmethodik für mechatronische Systeme*. Vorlesungsunterlagen zu Einführung in die Mechatronik 2. 20.04.2009

[05] Rick Bitter, Taqi Mohiuddin, Matt Nawrocki: *LabVIEW – Advanced Programming Techniques*. Taylor & Francis Group, Boca Raton 2007. ISBN-10: 0-8493-3325-3

[06] Anton Braun: *Grundlagen der Regelungstechnik – Kontinuierliche und diskrete Systeme*. Carl Hansen Verlag, München u.a. 2005. ISBN-10: 3-446-40305-1

[07] Klaus Dinners: *FPGA-Technologie in der Automatisierungstechnik*. In: *SPS-MAGAZIN 1+2 2011*. Stand: 27.01.2011. http://www.sps-magazin.de/?inc=artikel/article_show&nr=58922 (abgerufen am: 18.12.2012)

[08] ERIKA Enterprise: *ERIKA Enterprise and RT-Druid Features.* Stand 19.07.2011. http://erika.tuxfamily.org/drupal/features-and-beneficts.html (abgerufen am 16.02.2013)

[09] Wolfgang Georgi, Ergun Metin: *Einführung in LabVIEW*. Carl Hansen Verlag, München 2012. ISBN-13: 978-3-446-41560-7

[10] Gregor Häberle, u.a.: *Elektronik – Grundlagen*. Verlag Europa-Lehrmittel, Haan-Gruiten 1993. ISBN-10: 3-8085-3202-5

[11] Christian Hebel: „*Eine Sekunde bis zum Stopp*“. In: *Spiegel-Online (Wirtschaft)*. Stand: 24.11.2012. http://www.spiegel.de/wirtschaft/unternehmen/spiegel-neue-ice-modelle-haben-probleme-mit-steuerungssoftware-a-869137.html (abgerufen am 17.12.2012)

[12] Bodo Heimann, Wilfried Gerth, Karl Popp: *Mechatronik – Komponenten, Methoden, Beispiele*. Carl Hansen Verlag, München u.a. 2007. ISBN-10: 3-446-40599-2

[13] KW-Software GmbH: *IEC 61131 Programmiersysteme.* Stand 2013. https://www.kw-software.com/de/iec-61131-control/programmiersysteme (abgerufen am 16.02.2013)

[14] Jan Lunze: Automatisierungstechnik – Methoden für die Überwachung und Steuerung kontinuerlicher und ereignisdiskreter Systeme. Oldenbourg Verlag, München 2012. ISBN-13: 978-3-486-71266-7

[15] MathWorks, Inc.: *MATLAB – Die Sprache für technische Berechnungen*. Stand: 2013. http://www.mathworks.de/products/matlab/ (abgerufen am 16.02.2013)

[16] Stefan Möhringer: *Die neue Richtlinie VDI 2206: Entwicklungsmethodik für mechatronische Systeme*. Vortragsunterlagen zum Richtlinienausschuss A127/VDI 2206, 09.05.2003

[17] National Instruments: *Einführung in LabVIEW - Dreistündiger Einführungskurs.* Stand: 2005. www.ni.com/pdf/academic/d/labview_3_hrs.pdf (abgerufen am 06.02.2013)

[18] National Instruments: *Verwendung des NI CompactRIO Scan Mode mit NI LabVIEW.* Stand: 19.12.2011. http://www.ni.com/white-paper/7338/de (abgerufen am 22.01.2013)

[19] National Instruments: *Wie funktionieren FPGAs?.* Stand: 18.09.2012. http://www.ni.com/white-paper/6983/de (abgerufen am 17.12.2012)

[20] Gerhard Pahl u.a.: *Konstruktionslehre – Grundlagen erfolgreicher Produktentwicklung*. Springer Verlag, Berlin u.a. 2007. ISBN-13: 978-3-540-34060-7

[21] Patrik Schnabel: *Bipolarer Transistor.* In: *Elektronik Kompendium*. Stand: 14.01.2013. http://www.elektronik-kompendium.de/sited/bau/0201291.htm (abgerufen am: 14.01.2013)

[22] Elmar Schrüfer, Leonhard Reindl, Bernhard Zagar: *Elektrische Messtechnik – Messung elektrischer und nichtelektrischer Größen*. Carl Hansen Verlag, München 2012. ISBN-13: 978-3-446-43079-2

[23] Siemens AG: *SIMATIC STEP 7: das umfassende Engineeringsystem*. Stand 2013. http://www.automation.siemens.com/mcms/simatic-controller-software/de/step7/seiten/default.aspx (abgerufen am 16.02.2013)

[24] RoboterNETZ: *Schrittmotoren*. Stand 15.01.2013. http://www.rn-wissen.de/index.php/Schrittmotoren (Abgerufen am 07.02.2013)

[25] Günter Wellenreuther, Dieter Zastrow: *Automatisieren mit SPS – Theorie und Praxis*. Vieweg + Teubner, Wiesbaden 2008. ISBN-13: 978-3-8348-0231-6

[26] Alexander Weber: *Shake, rattle ‘n’ roll Mikrocontroller-Programmierung mit dem gewissen Dreh*. In: *c’t magazin 16/09*. Stand: 20.07.2009 http://www.heise.de/ct/artikel/Shake-rattle-n-roll-292164.html (abgerufen am: 04.02.2013)

[27] Wikipedia: *Schrittmotor*. Stand 15.01.2013. http.//de.wikipedia.org/wiki/Schrittmotor (abgerufen am 24.01.2013)

[28] Heinz Wörn, Uwe Brinkschulte: *Echtzeitsysteme – Grundlagen, Funktionsweisen, Anwendungen.* Springer-Verlag, Berlin u.a. 2005. ISBN-10: 3-540-20588-8

[29] Datenblatt: *103-547-52500 – Stepping Motor*. SANYO DENKI 2011

[30] Datenblatt: *AD592 - Low Cost, Precision IC Temperature Transducer*. Analog Devices 2004

[31] Datenblatt: *BD139 - NPN Silicon Transistor*. STMicroelectronics 2001

[32] Datenblatt: *LM317 - 1.2V to 37V Voltage Regulator*. STMicroelectronics 2004

[33] Datenblatt: *TC74HC02AP – Quad 2-Input NOR Gate*. TOSHIBA 2001

[34] Datenblatt: *TL071C,AC – Low Noise, JFET Input Operational Amplifiers*. Motorola 1997

10 Anhang

10.1 Auflistung der benutzten Hardware für jeden Versuch

Im Folgenden wird die Hardware, welche bei der Erstellung der Laborversuche benutzt wurde festgehalten. Zum Durchführen der Laborversuche sollte möglichst die gleiche oder vergleichbare Hardware verwendet werden.

	Verwendete Hardware	
#	**Art**	**Beschreibung**
1	Real-Time-Controller	National Instruments NI cRIO-9022 Sr.Nr. 1670C6D Inv.Nr. 35102004940-0
2	Rekonfigurierbares Chassis	National Instruments NI cRIO-9114 Sr.Nr. 165427B Inv.Nr. 35102004664-0
3	Digitales Eingangs Modul	National Instruments NI-9421 Sr.Nr. 1653B18
4	Digitales Ausgangs Modul	National Instruments NI-9472 Sr.Nr. 1650FCA
5	Analoges Eingangs Modul	National Instruments NI-9215 Sr.Nr. 1675755
6	Analoges Ausgangs Modul	National Instruments NI-9263 Sr.Nr. 1683F47
7	Digitales Ein-/Ausgangsmodul	National Instruments NI-9403 Sr.Nr. 167E40E

8	Netzgerät	DC POWER SUPPLY DF-3010-A Sr.Nr. 0010
9	Netzgerät	ROHDE & SCHWARZ DC POWER SUPPLY NGT20 Inv.Nr. 536
10	Oszilloskop	Tektronix TDS3014C Inv.Nr. 35102004337-0
11	Funktionsgenerator	TOPWARE ELECTRIC INSTRUMENTS CO., LTD:.TFG-462 Inv.Nr. 989/15.03
12	Multimeter	GROSSEN-METRAWATT GmbH METRA HIT ONE
13	Lötkolben	fixPOINT EP 5
14	Beschleunigungssensor	PCB PIEZOTRONICS 3741D4HB2G/M005JJ Sr.Nr. 3711
15	Neigungssensor	KÜBLER 8.IS40.22321
16	DC-Motor	IGARASHI Motoren GmbH N2738-125 CKD1/244475
17	Schrittmotor	SANYO DENKI 103-547-52500 LotNr. 09820F
18	Ventilator	WALLAIR 20.100.308
19	Peltier-Element	Conrad TEC1-01703 BestNr. 193550

Tabelle 10 Auflistung der benutzten Hardware.

In der nachfolgenden Tabelle werden die in jedem Versuchen verwendeten Komponenten festgehalten.

Versuch	**Verwendete Hardware und Komponenten**
1	1, 2, 4,8, 10 und 1,2 kΩ (6 W) Widerstand
2	1, 2, 5, 6, 8, 10 und 11
3	1, 2, 4, 8, 10 und 1,2 kΩ (6 W) Widerstand
4	1, 2, 3, 7, 8, 9, 10, 13, Lötzinn, Joystick, 1,2 kΩ (6 W) Widerstand und die folgenden Bauteile: 2x Leiterplatte 1x Fotodiode (SFH 203 FA) 1x IR-LED (HE3-290AC) 2x Anschlussklemmen (MKDSN 1,5/3) 1x Transistor (BD 139) 1x Widerstand 1k Ω 2x Widerstand 10k Ω 1x Diode (N1 4007) 3x Anschlussklemmen (MKDSN 1,5/3) 1x Draht
5	1, 2, 5, 8, 9, 10, 14 und 15
6	1, 2, 5, 8, 9, 10, 12, Temperatursensor AD 592 AN, Schaltung entsprechend Abbildung 44, Thermometer und Leistungswiderstand oder andere Wärmequelle
7	1, 2, 4, 7, 8, 9, 10, 16, 18, 19 und einen Lautsprecher
8	1, 2, 7, 8, 9, 10, 16 und eine H-Brücke entsprechend der Abbildung 48
9	1, 2, 7, 8, 9, 10, 17, zwei H-Brücken entsprechend der Abbildung 48 und einen Joystick

Tabelle 11 Verwendete Hardware und Komponenten für jeden Versuch.

10.2 Messwerte zum Neigungssensor Kübler 8.IS40.22321

Winkel (°)		**Spannung (V)**	
x-Achse	y-Achse	U_x	U_y
0	0	*2,59*	*2,66*
+ 26	0	*4,93*	
- 31	0	*0,19*	
0	*+ 30*		*4,98*
0	*- 27*		*0,26*
+10	0	*3,51*	
		3.60	
-10	0	*1,76*	
		1,94	
0	+10		*3,43*
			3,32
0	-10		*1,73*
			1,67

Tabelle 12 Messwerte zum Neigungssensor Kübler 8.IS40.2232, Gemessene Werte kursiv, errechnete Werte fett und kursiv.

10.3 Signale an die H-Brücken für Voll- und Halbschrittbetrieb

Signal an die H-Brücken für Vollschrittbetrieb				
	H-Brücke 1		**H-Brücke 2**	
	PWM(TTL)	*Rechts/Links(TTL)*	*PWM(TTL)*	*Rechts/Links(TTL)*
Schritt 1	false	true	false	true
Schritt 2	false	true	false	false
Schritt 3	false	false	false	false
Schritt 4	false	false	false	true

Tabelle 13 Ausgefüllte Tabelle mit den Signalen an die H-Brücken für Vollschrittbetrieb.

Signal an die H-Brücken für Halbschrittbetrieb				
	H-Brücke 1		**H-Brücke 2**	
	PWM(TTL)	*Rechts/Links(TTL)*	*PWM(TTL)*	*Rechts/Links(TTL)*
Schritt 1	false	true	false	true
Schritt 2	false	true	true	false
Schritt 3	false	true	false	false
Schritt 4	true	false	false	false
Schritt 5	false	false	false	false
Schritt 6	false	false	true	true
Schritt 7	false	false	false	true
Schritt 8	true	true	false	true

Tabelle 14 Ausgefüllte Tabelle mit den Signalen an die H-Brücken für Halbschrittbetrieb.

10.4 Inhalt der beiliegenden CD

Die beiliegende CD enthält die Abschlussarbeit in pdf- und Word 2010 Format (docx) im Hauptverzeichnis. Im Unterverzeichnis *Datenblätter* finden sich die in der Arbeit verwendeten Datenblätter der elektronischen Bauteile [29], [30], [31], [32], [33], und [34]. Im Unterverzeichnis *Internetquellen* finden sich die Druckversionen der im Literaturverzeichnis aufgelisteten Internetquellen.
Im Verzeichnis *Programme* sind die, für jeden Versuch erstellten LabVIEW-Programme in den separaten Unterverzeichnissen *Lösungen* zu finden. Für die Versuche 3 und 9 waren in den Aufgabenstellungen Programmteile vorgegeben, dies sind in den Unterverzeichnissen *Vorgegebene Programmteile* zu finden.

Verzeichnisstruktur der CD:

- Hauptverzeichnis
    - Literatur
        - Datenblätter
        - Internetquellen
    - Programme
        - Versuch 1
            - Lösung
        - Versuch 2
            - Lösung
        - Versuch 3
            - Lösung
            - Vorgegebene Programmteile
        - Versuch 4
            - Lösung
        - Versuch 5
            - Lösung
        - Versuch 6
            - Lösung
        - Versuch 7
            - Lösung
        - Versuch 8

        - Lösung
- Versuch 9
    - Lösung
    - Vorgegebene Programmteile

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences
Department Maschinenbau und Produktion

Formblatt **Erklärung zur selbstständigen Bearbeitung einer Bachelorthesis**

Zur Erläuterung des Zweckes dieses Blattes:

§ 16 Abs. 5 der APSOTIBM lautet:

„Zusammen mit der Thesis ist eine schriftliche Erklärung abzugeben aus der hervorgeht, dass die Arbeit bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit (§18 Absatz 1) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt wurden. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich zu machen."

Dieses Blatt mit der folgenden Erklärung ist nach Fertigstellung der Arbeit durch jede/n Kandidat/en/in auszufüllen und jeweils mit Originalunterschrift als letztes Blatt des als Prüfungsexemplar der Bachelorthesis gekennzeichneten Exemplars einzubinden.

Eine unrichtig abgegebene Erklärung kann - auch nachträglich - zur Ungültigkeit des Bachelorabschlusses führen.

Erklärung

Hiermit versichere ich,

Name: Quade

Vorname: Peter

dass ich die vorliegende Bachelorthesis – bzw. bei einer Gruppenarbeit die entsprechend gekennzeichneten Teile der Arbeit – mit dem Thema

Entwicklung eines praktischen Mechatronik-Laborversuches

ohne fremde Hilfe selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

- die folgende Aussage ist bei Gruppenarbeiten auszufüllen und entfällt bei Einzelarbeiten -

Die Kennzeichnung der von mir erstellten und verantworteten Teile der Bachelorthesis ist erfolgt durch

Hamburg _____ _____ _____

Ort Datum Unterschrift im Original