Test des Goniometers und seiner Anbindung an EPICS für Elektronenstreuexperimente am QCLAM-Spektrometer

Miniforschung

Dirk Martin

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Inhaltsverzeichnis

1 Einleitung **3**

2 Aktueller Aufbau des Goniometers **4**
2.1 Belegung der Anschlüsse 5
2.2 Elektronische Bauteile des Goniometers 6
2.3 Derzeitige Elektronik 7
2.4 Dokumentation 8

3 Test der elektronischen Bauteile **9**
3.1 Test der Motoren 9
3.2 Test der Winkelschrittgeber 9
3.2.1 Theorie der Funktionsweise eines Winkelschrittgebers 9
3.2.2 Testaufbau für den Winkelschrittgeber 10
3.3 Simulation am Eigenbau 12
3.3.1 eitech-Konstruktion 12
3.3.2 Arduino 13
3.3.3 Plots 15

4 EPICS **17**
4.1 Grundlegendes 17
4.2 IOC-Server 17
4.3 Control System Studio 18
4.4 RDB Channel Archiver 18

5 Ausblick **20**

A Anhang **21**
A.1 Installation von Arduino unter Ubuntu 21
A.2 Quellcode des Arduino-Skripts 21
A.3 C++ Skript: sortA.cpp & sortB.cpp 22
A.4 Installation von EPICS 22
A.5 Einrichtung eines Beispiel-IOC-Servers 23

Literaturverzeichnis **25**

Abbildungsverzeichnis

2.1 Goniometer und Targetleiter 4
2.2 Streukammerdeckel des Goniometers 5
2.3 Belegung der Anschlüsse am Goniometer 6

3.1 Blockdiagramm zur Funktion eines Winkelschrittgebers 9
3.2 Schaltskizze des Winkelschrittgebertesters 10
3.3 Verbindungskabel mit umgesetzten Anschlüssen 11
3.4 Winkelschrittgebertester in der Box 11
3.5 Seitenansicht der eitech-Konstruktion 12
3.6 Ansicht der eitech-Konstruktion von oben 13
3.7 Schematische Darstellung der Verbindungen Arduino-Board - Winkelschrittgeber . 14
3.8 Zeitliche Abfolge der Winkelschrittgebersignale 16

1 Einleitung

Das Goniometer am QCLAM-Spektrometer wurde 1989 von D. Kleinhanss [1] für Koinzidenzexperimente vom Typ $(e, e'x)$ in einer Vakuumkammer (Streukammer) entwickelt. Die entsprechende Streukammer für Elektronenstreuexperimente am S-DALINAC konstruierte M. Kuss [2]. Zur selben Zeit entwickelte H. Diesener [3] eine Targetschleuse für dieselbe Streukammer, damit bei Austausch des Targets nicht jedes Mal die gesamte Apparatur, sondern nur ein kleines Volumen evakuiert werden muss. Zur Steuerung von Goniometer und Targetschleuse entschied man sich damals für die Verwendung von DC Motoren, da sich Schrittmotoren aus konstruktionstechnischen Gründen und aufgrund der zu starken Wärmeentwicklung in den Motoren nicht eigneten. Die Elektronik für das Goniometer entwarf G. Hartung [4] als NIM Crate Modul mit Mikroprozessor Controller. Die Software für den gesamten Aufbau wurde letztlich von E. Heid [5] in der Programmiersprache C als Kommando-Shell umgesetzt. Jene konnte über ein Terminal im Messraum und über eine Handsteuerung am QCLAM-Spektrometer gesteuert werden. Die Genauigkeit der Ansteuerung lag unter 0,1 ° bzw. 0,1 mm.
Auf lange Sicht ergaben sich bei der Benutzung dieses Programms jedoch zahlreiche Probleme an Hard- und Software. So kam es durch einen Variablenfehler dazu, dass die Targetleiter bei ihrer Bewegung „oszillierte“, d. h. dass sie sich zunächst wie gewünscht nach oben bewegte, dann aber plötzlich nach unten, kurze Zeit später wieder nach oben und so weiter. Weiterhin war es mit der vorhandenen Spannungsversorgung möglich die Motoren mit bis zu 25 V zu betreiben, wofür aber nicht alle Motoren ausgelegt waren. Ebenso wiesen einige der alten Winkelschrittgeber vom Typ HEDS 53XX Defekte auf, was auch daran lag, dass bei der Verkabelung die Signale der Kanäle teilweise mit der Stromversorgung der Winkelschrittgeber vertauscht waren. Noch gravierender war die Tatsache, das sämtliche Freiheitsgrade am Goniometer mit der Software über ihre mechanischen Grenzen steuerbar waren, wodurch Bauteile beschädigt werden konnten.
Der Großteil dieser Probleme wurde in den Jahren 2000 und 2001 von Y. Kalmykov und A. Shevchenko [6] behoben. Dennoch kam es seitdem immer wieder zu Abstürzen der Kommando-Shell und fehlenden Kontakten in den betagten Verbindungskabeln zwischen Steuerung und Goniometer. Da die Hauptplatine der Steuerung mittlerweile veraltet und der Zugriff auf das Steuerungsprogramm überaus kompliziert ist, entschied man sich für eine Neukonstruktion und -programmierung der Ansteuerung des Goniometers. Hierbei soll eine Anbindung an die Open Source Software EPICS erfolgen, die bereits bei den Netzgeräten des Beschleunigers S-DALINAC verwendet wird. Für EPICS hat man sich auch deshalb entschieden, weil es bereits an anderen Beschleunigern wie dem DESY in Hamburg eingesetzt wird und - im Gegensatz zu einer eigenen Lösung - bereits eine sehr große Community (inkl. Mailing-Liste) gibt, die Support und Dokumentationen bereitstellt und EPICS ständig weiterentwickelt.

Die Aufgabe der vorliegenden Arbeit bestand nun darin, die Motoren und Winkelschrittgeber des Goniometers auf Funktionalität zu überprüfen, die Belegung der Anschlüsse am Deckel der Streukammer des Goniometers zu ermitteln und eine Neuprogrammierung der Steuerung mit Anbindung an EPICS vorzubereiten.

2 Aktueller Aufbau des Goniometers

1. **Targetleiterführungsröhre**
2. **Motoren für die Targetleiter**
3. **Getriebebox**
4. **Targetschleuse**
5. **Ventil**
6. **Streukammerdeckel**
7. **Oberer Goniometerring**
8. **Targetleiter**
9. **Mittlerer Goniometerring**
10. **Verbindungsstrebe**
11. **Winkelschrittgeber**
12. **Unterer Goniometerring**
13. **Detektorteleskop**

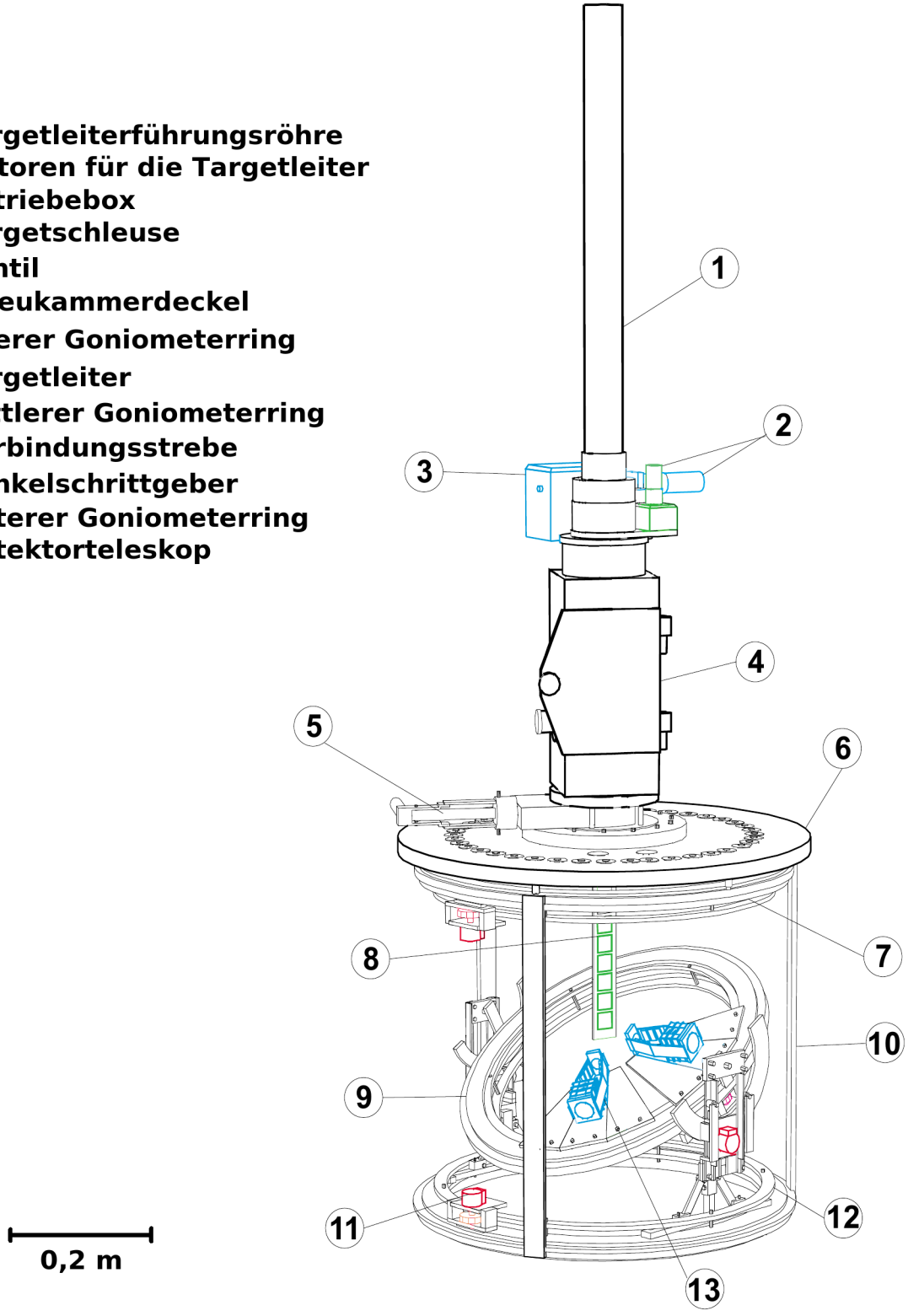

Abbildung 2.1: Goniometer und Targetleiter

2.1 Belegung der Anschlüsse

Um die Belegung der Anschlüsse am Streudeckel des Goniometers zu bestimmen, wurde jeder Anschluss einzeln mit einem Multimeter durchgemessen. Die folgende Abbildung zeigt eine schematische Darstellung der Anschlüsse auf der Oberseite des Streukammerdeckels, an die die Steuerung angeschlossen wird. Hierbei sind die BNC-Anschlüsse für die Detektoren nicht eingezeichnet.

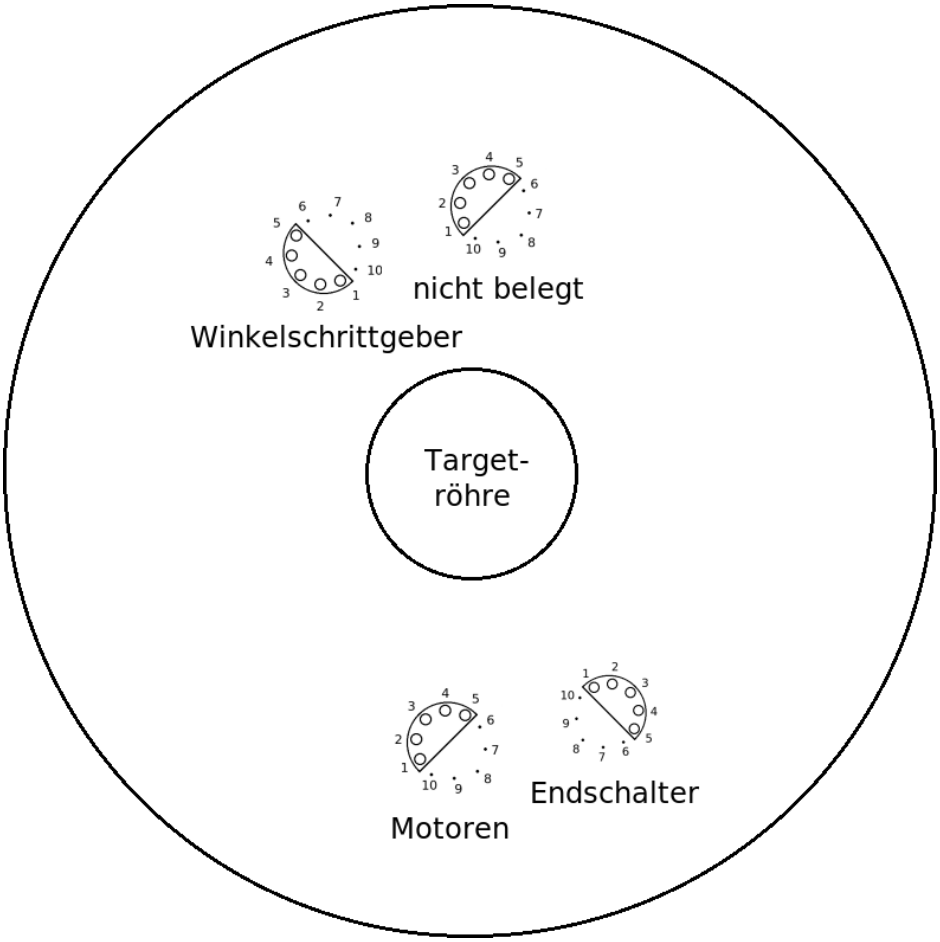

Abbildung 2.2: Streukammerdeckel des Goniometers

Die Anschlüsse auf der oberen Seite des Streukammerdeckels führen zu den Anschlüssen für die Motoren, die Endschalter und die Winkelschrittgeber, die sich direkt unter dem Streukammerdeckel, d. h. im Vakuum der Streukammer selbst, befinden. Bei den Durchführungen der Anschlüsse handelt es sich um spezielle Vakuumdurchführungen.
Nachfolgend sind die Verbindungen der Anschlüsse auf und unter dem Streukammerdeckel dargestellt. Die Ansicht (bzw. Durchsicht) erfolgt von oben auf (durch) den Deckel des Goniometers. Alle eingezeichneten Pfeile zeigen zu den entsprechenden anderen Anschlusstypen und dienen der Orientierung. Zusätzlich wurden die masseführenden Kabel der Verbindungen zu den Winkelschrittgebern angedeutet.

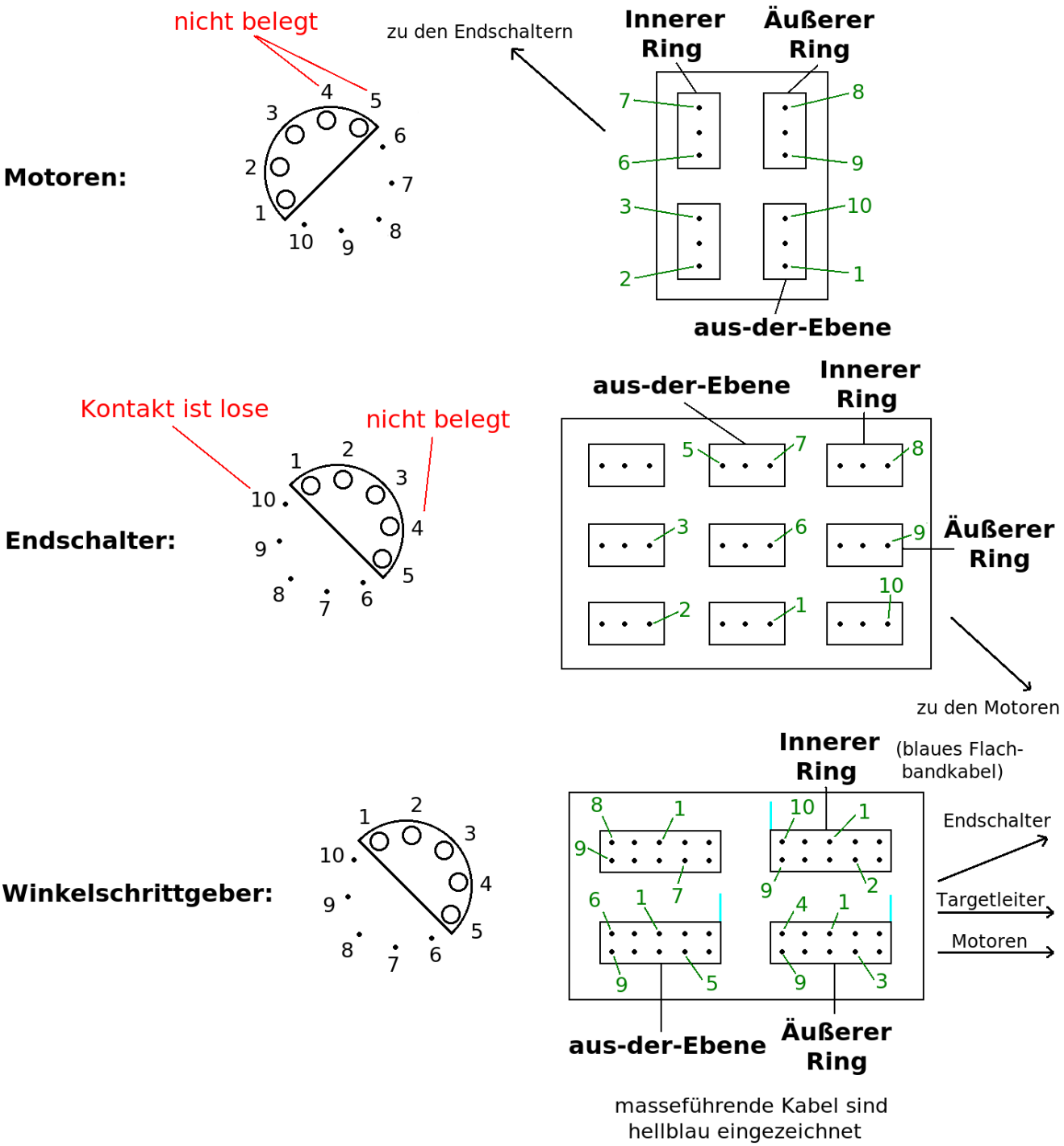

Abbildung 2.3: Belegung der Anschlüsse am Goniometer

2.2 Elektronische Bauteile des Goniometers

Die Freiheitsgrade des Goniometers werden über Gleichstrommotoren angesteuert, da die Motoren im Vakuum der Streukammer eingesetzt werden. Um die Motoren ausreichend mit Spannung zu versorgen, muss diesen eine Spannung von 24 V bereitgestellt werden. Jedoch erlaubt der Motor, der den äußeren Ring steuert, eine Maximalspannung von 18 V, so dass seitens der Software auf eine Beschränkung der Spannung an diesem Motor geachtet werden muss.
Y. Kalmykov und A. Shevchenko schlagen in ihrer Dokumentation über die Verbesserungen am

Goniometer vor, an der Targetleiter zukünftig Schrittmotoren zu verwenden, da diese Motoren außerhalb der Vakuumkammer liegen. [6]

Bis auf beim Inneren Ring wurden im Jahr 2000 alle Winkelschrittgeber gegen die neueren Modelle HEDS 5540/5640 A12 ausgetauscht, welche gegenüber den alten Typen einen weiteren Anschluss, den Indexpuls, besitzen. Der Indexpuls wird nach einer vollständigen Umdrehung der Lochscheibe gesendet und kann von der Software entweder als Kontrolle oder direkt für die Berechnung des Drehwinkels bei mehrfachen Umdrehungen der Lochscheibe verwendet werden. Die Bezeichnung A12 bedeutet, dass sie 500 Zyklen pro Umdrehung liefern (das entspricht 500 Rechtecksignalen auf Kanal A und Kanal B pro Umdrehung). Dadurch liegt die Auslesegenauigkeit der Winkelschrittgeber bei weniger als 0,1 °. Die beiden Modelle HEDS 5540 und HEDS 5640 unterscheiden sich nur dadurch, dass die 56XX-Reihe „Ohren“ zum Befestigen des Winkelschrittgebers auf einer Oberfläche besitzt. Auf die Theorie der Funktionsweise eines Winkelschrittgebers wird im Abschnitt 3.2.1 detailliert eingegangen.

An den Endpositionen der drei Freiheitsgrade Innerer Ring, Aus-der-Ebene und Äußerer Ring wurden Taster angebracht, die die jeweilige Endposition des Freiheitsgrades markieren. Sie wurden von der alten Software nach einem Neustart als Referenzpunkte benutzt.

2.3 Derzeitige Elektronik

Gegenwärtig sind zur Steuerung des Goniometers Motoren von den Typen ESCAP 23L21, 22V2R28, 22N2R28 und 28DT12 in Kombination mit den Getriebetypen ESCAP B24, K24, R32 und R22 eingebaut. Davon sind die Motoren der Typen ESCAP 22V2R28 und 22N2R28 sowie das Getriebe ESCAP K24 die neueren Modelle.
In der folgenden Tabelle sind die aktuellen Kombinationen von Motoren und Getrieben am Goniometer aufgelistet.

#	Freiheitsgrad	ESCAP Motor Typ	ESCAP Getriebe	Übersetzungsverhältnis
1	Innerer Ring	23L21 208E	B24	320:1
2	Aus-der-Ebene	22V2R28 208E.202	K24	800:1
3	Äußerer Ring	22N2R28 210E.286	K24	800:1
4	-	-	-	-
5	Target-Höhe	28DT12 222E55	R32 14.0	72,3:1
6	Target-Winkel	23L21 208E	R22.0	640:1

Tabelle 2.1: Motoren und Getriebe am Goniometer

Derzeit werden fast ausschließlich Winkelschrittgeber vom Typ HEDS 5540/5640 A12 zum Auslesen der Bewegung einzelner Freiheitsgrade verwendet, weil die älteren Modelle im Jahr 2000 defekt waren und ersetzt werden mussten. Nur am inneren Ring wird noch ein alter Winkelschrittgeber vom Typ HEDS 5310 836 D eingesetzt.
Die nachfolgende Tabelle zeigt, welche Winkelschrittgeber momentan zum Auslesen der jeweiligen Freiheitsgrade verwendet werden.

#	Freiheitsgrad	Winkelschrittgeber
1	Innerer Ring	HEDS 5310 836 D
2	Aus-der-Ebene	HEDS 5640 A12
3	Äußerer Ring	HEDS 5540 A12
4	-	-
5	Target-Höhe	HEDS 5540 A12
6	Target-Winkel	HEDS 5640 A12

Tabelle 2.2: Winkelschrittgeber am Goniometer

2.4 Dokumentation

Zur besseren Übersicht über die beim Goniometer vorhandene Elektronik sowie zur Archivierung wurde eine Dokumentation, die sämtliche Datenblätter zur Elektronik beinhaltet, in Form eines Wiki-Eintrages im ikpweb [7] erstellt. Es ist innerhalb des Netzwerkes im Institut für Kernphysik zugänglich. Auf diese Weise ist es sehr einfach, sich einen Überblick über alle Bauteile des Goniometers zu verschaffen, falls z. B. ein Defekt vorliegt oder weitere Anschlüsse gelegt werden sollen. Des Weiteren kann der Wiki-Eintrag beliebig erweitert werden, sobald neue mechanische oder elektronische Bauteile eingesetzt werden.

3 Test der elektronischen Bauteile

3.1 Test der Motoren

Die Motoren wurden mithilfe einer einfachen Schaltung in Betrieb genommen: Sie wurden mit einem Multimeter - über das kontrolliert werden sollte, dass die Stromstärke keine Werte erreicht, bei welchen der Motor Schaden nehmen könnte (teils waren schon 0,4 A die obere Grenze, nähere Hinweise finden sich in den Datenblättern der Motoren) - in Reihe geschaltet. Bei einer Spannung von 7,20 V (Maximalspannung des vorhandenen Netzteils) und Stromstärken zwischen 0,01 A und 0,28 A konnten alle Motoren in Bewegung gesetzt werden. Die maximale Stromstärke wurde bei dem Motor erreicht, der für die Targethöhe zuständig ist, erreicht, weil dieser mit 6,2 Ohm den geringsten Widerstand besitzt. Der Motor, welcher für den Parameter „aus-der-Ebene“ zuständig ist, bewegte sich verglichen mit den anderen Motoren deutlich langsamer, da er mit dem Verkippen der Detektor-Ebenen über ein schweres Zahnrad ein höheres Drehmoment aufbringen muss.

3.2 Test der Winkelschrittgeber

Dieser Test gestaltete sich etwas aufwendiger als der vorherige, da zum Betreiben der Winkelschrittgeber eine weitere Spannungsversorgung benötigt wird. Bevor ein Aufbau realisiert wurde, der einen Winkelschrittgeber auslesen kann, war es notwendig, sich zuerst mit der Theorie der Funktionsweise eines Winkelschrittgebers auseinanderzusetzen.

3.2.1 Theorie der Funktionsweise eines Winkelschrittgebers

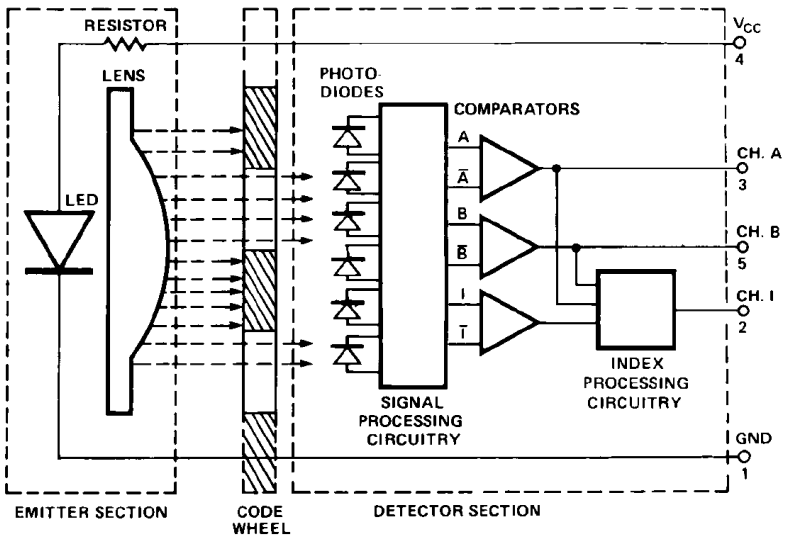

Abbildung 3.1: Blockdiagramm zur Funktion eines Winkelschrittgebers

Wie im Diagramm gezeigt, enthalten die Winkelschrittgeber der HEDS 55XX/56XX Serie eine LED als Lichtquelle [8]. Über eine Linse wird deren Licht parallel gerichtet und fällt auf eine Lochscheibe. Hinter der Lochscheibe befinden sich sechs Photodetektoren, die so angeordnet sind, dass ein Paar der Photodetektoren das Licht wahrnehmen kann, während das benachbarte Paar von Detektoren im Schatten liegt. Wegen der Rotation der Lochscheibe erzeugt der Lichteinfall auf die Photodetektoren nun eine Folge aus Licht und Schatten, das vom Radius und dem Design der Lochscheibe abhängt. Die Signale der einzelnen Photodetektoren werden an Komparatoren weitergeleitet, die nun die Ausgangssignale (Rechtecksignale) A, B und I erzeugen. Das Ausgangssignal I ist ein Indexpuls, der einmal bei jeder vollständigen Umdrehung der Lochscheibe - wenn A sowie B gerade ein tiefes Signal liefern - gesendet wird.
Die Rotationsrichtung der Lochscheibe lässt sich dadurch ermitteln, dass die Ausgangssignale A und B nie gleichzeitig sondern immer nacheinander ausgesendet werden. Geht das Signal A dem Signal B voraus, so liegt eine Rotation im Uhrzeigersinn - im umgekehrten Fall eine Rotation gegen den Uhrzeigersinn vor.

3.2.2 Testaufbau für den Winkelschrittgeber

Es wurde ein eigener Winkelschrittgebertester konzipiert, der an ein Oszilloskop angeschlossen werden kann. Damit der Aufbau kompakt ist, wurde er auf einer kleinen Loch-Raster-Platine umgesetzt.

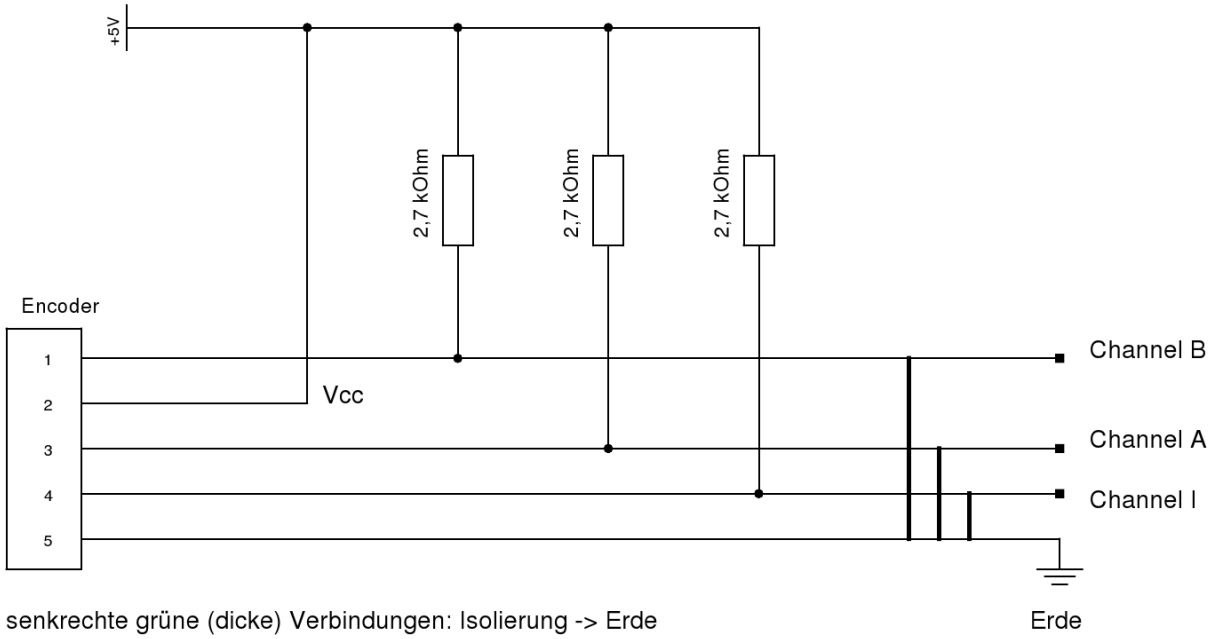

Abbildung 3.2: Schaltskizze des Winkelschrittgebertesters

Über den V_{cc}-Anschluss wird der Winkelschrittgeber mit +5 V versorgt. Die Kanäle A, B und I sind über 2,7 kOhm Widerstände ebenfalls mit der Spannungsversorgung (V_{cc}) verbunden und enden in BNC-Kabeln, deren Isolierung geerdet ist. An den Enden der Spannungsversorgung und der GND-Leitung (Erde) befinden sich einfache Bananensteckerkabel, die an Labornetzgeräte angeschlossen werden.
Es stellte sich heraus, dass für die Verbindung zu den Winkelschrittgebern eigene Verbindungs-

kabel (Flachbandkabel) entworfen wurden. Das war dadurch bedingt, dass die alten Winkelschrittgebermodelle noch keinen Indexpuls besaßen. Überdies konnten die verbleibenden vier Anschlüsse nur in vertauschter Reihenfolge an die von G. Hartung entworfene Hardware für die Steuerung angeschlossen werden, so dass die Adern der selbst entworfenen Verbindungskabel entsprechend geführt wurden.
Die untere Abbildung zeigt ein solches Verbindungskabel.

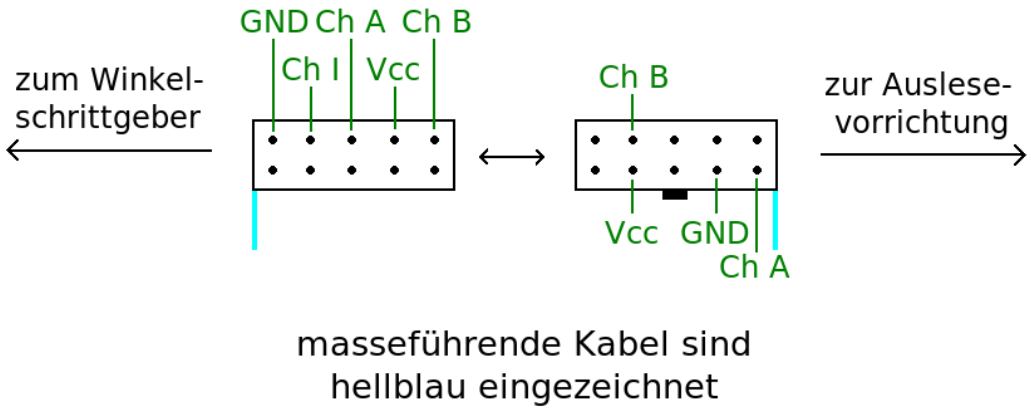

Abbildung 3.3: Verbindungskabel mit umgesetzten Anschlüssen

Daher wurden die Pins für den Winkelschrittgebertester so verlötet, dass sie an Verbindungskabel mit beliebiger Aderführung angeschlossen werden können. Der gesamte Aufbau wurde in einer Box untergebracht, die Zugfestigkeit für die einzelnen Kabel garantiert und verhindert, dass die Schaltung durch Berührung mit den Händen beschädigt wird.

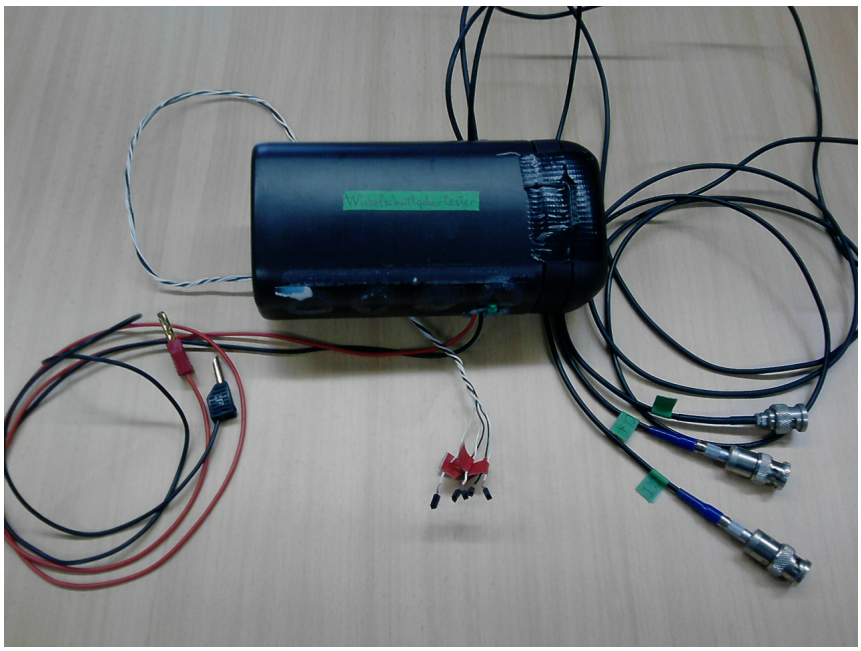

Abbildung 3.4: Winkelschrittgebertester in der Box

Bei einer Spannung von $U = 20,1$ V und einer Stromstärke von $I = 0,14$ A am Motor der Targetleiter wurden in deren Winkelschrittgeber Signale im Abstand von 4 ms erzeugt. Dies

entspricht einer Frequenz von $f = 250$ Hz und liegt somit über der Frequenz von 100 Hz, mit der die bisherigen Netzteile, die der Leiter der Elektronikwerkstatt, Herr Bonnes, entwickelt hat, umgehen können. Außerdem ist der Motor für bis zu 24 V ausgelegt, d. h. dass noch höhere Frequenzen erreicht werden können. Daher müssen neue Netzteile konzipiert werden, die mit diesen hohen Frequenzen umgehen können.

3.3 Simulation am Eigenbau

Zum Testen des Zusammenspiels von Motoren, Winkelschrittgeber und Taster wurde ein Testaufbau realisiert, an dem die Funktionsweise der Bauteile des Goniometers simuliert werden konnte.

3.3.1 eitech-Konstruktion

Der Testaufbau wurde mit Teilen des eitech-Construction-Sets aufgebaut. Dazu wurden zwei Zahnräder verwendet, von denen das kleine Zahnrad (10 Zähne), an dem der Motor sitzt, das große Zahnrad (58 Zähne) in Bewegung setzt. Am großen Zahnrad ist der Winkelschrittgeber befestigt. Die Übersetzung vom Motor zum Winkelschrittgeber beträgt 1:5,8. Damit Motor und Winkelschrittgeber nicht verrutschen, sobald der Aufbau in Betrieb genommen wird, wurden beide Bauteile durch zusätzliche Rahmenkonstruktionen fixiert.
Ferner wurde der Motor mit einem Taster, der den Stromkreis unterbricht, in Reihe geschaltet (dies entspricht der Funktion eines Endschalters). Die Enden dieser Schaltung wurden mit Bananensteckerkabeln versehen, so dass der Motor wieder von einem Labornetzteil angetrieben werden kann. Der Winkelschrittgeber wurde mit dem ebenfalls selbst gebauten Winkelschrittgebertester verbunden und ausgelesen.

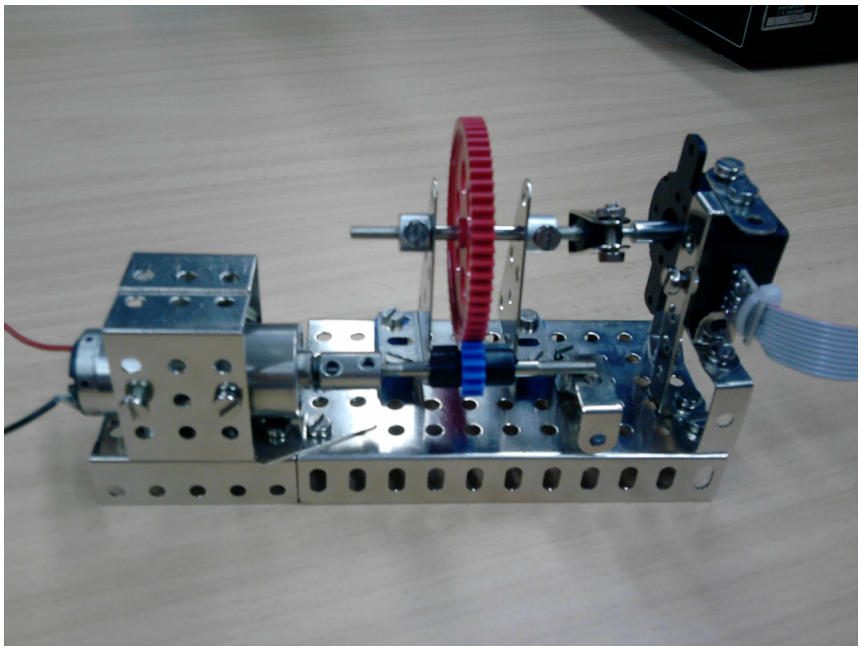

Abbildung 3.5: Seitenansicht der eitech-Konstruktion

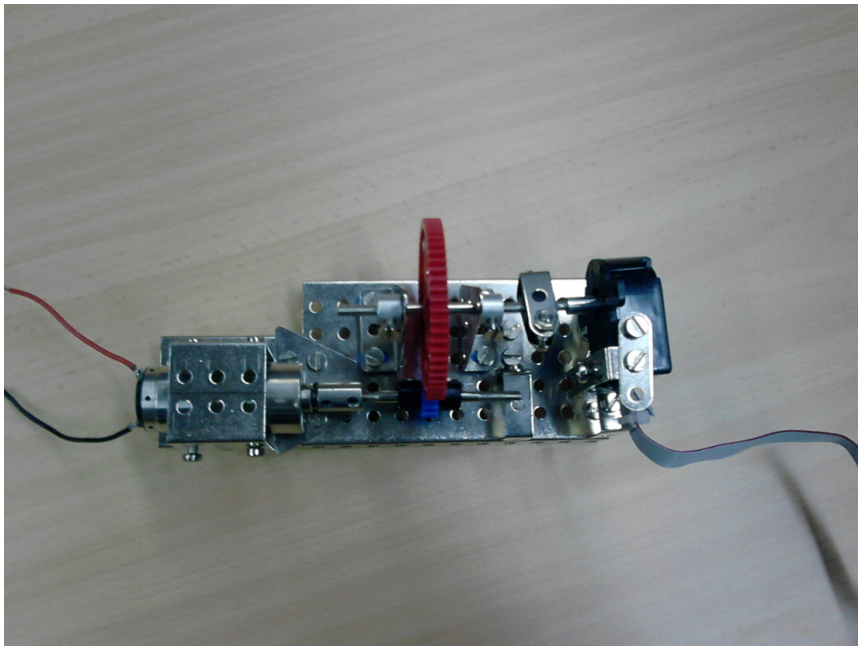

Abbildung 3.6: Ansicht der eitech-Konstruktion von oben; das kleine (blaue) Zahnrad sitzt direkt auf dem Motor (links im Bild) und treibt das große (rote) Zahnrad an, dessen Bewegung der Winkelschrittgeber (auf der rechten Seite) erfasst

3.3.2 Arduino

Nachdem das Auslesen des Winkelschrittgebers über das Oszilloskop erfolgreich erprobt worden war, wurde die obige Schaltung am Arduino Board der Arbeitsgruppe umgesetzt (Installation der Software: s. Anhang A.1). Die Schaltskizze ist in Abbildung 3.7 dargestellt und umfasst neben dem Auslesen des Winkelschrittgebers noch die Steuerung zweier LEDs. Da die Programmierung des Arduino-Mikrokontrollers (wie die von EPICS) auf C/C++ basiert und das Board eine Spannungsversorgung von 5 V besitzt, eignet es sich hervorragend für das Auslesen eines Winkelschrittgebers als Vorbereitung für die geplante Programmierung unter EPICS. Des Weiteren lassen sich die gemessenen Signale per USB-Verbindung an die Konsole eines Linux-Systems senden.
Der Quellcode eines selbstgeschriebenen Skripts zur Steuerung des Arduino-Boards befindet sich im Anhang A.2. Das Skript bewirkt einerseits, dass die derzeitige Signale der Kanäle A und B an die beiden LEDs weitergeleitet und somit in Form von deren Helligkeit angezeigt werden. Andererseits sendet der Arduino-Mikrokontroller bei jedem Durchgang der loop()-Schleife per USB die analogen Werte der Kanäle A und B an die Konsole. Arduino weist analogen Signalen Werte zwischen 0 und 1024 (10 Byte) zu. Die USB-Schnittstelle kann unter Linux mit dem Befehl

```
tail -f /dev/ttyUSB0 > datafile
```

abgehorcht und in die Datei namens `datafile` geschrieben werden. In dieser Datei liegen die Werte der beiden Kanäle nun gemischt vor, so dass sie sich noch nicht mit gnuplot darstellen lassen. Daher wurden zwei C++ Skripte `sortA.cpp` und `sortB.cpp` (s. Anhang A.3) geschrieben, die die Daten der Kanäle trennen.

```
./sortA.cpp > a.dat && sortB.cpp > b.dat
```

Nun müssen noch die Bezeichnungen „A“ bzw. „B“ aus den neuen Dateien `a.dat` und `b.dat` entfernt werden (z. B. über die Funktionen Suchen und Ersetzen eines einfachen Text-Editors). Dann können diese Datensätze unter gnuplot graphisch dargestellt werden.

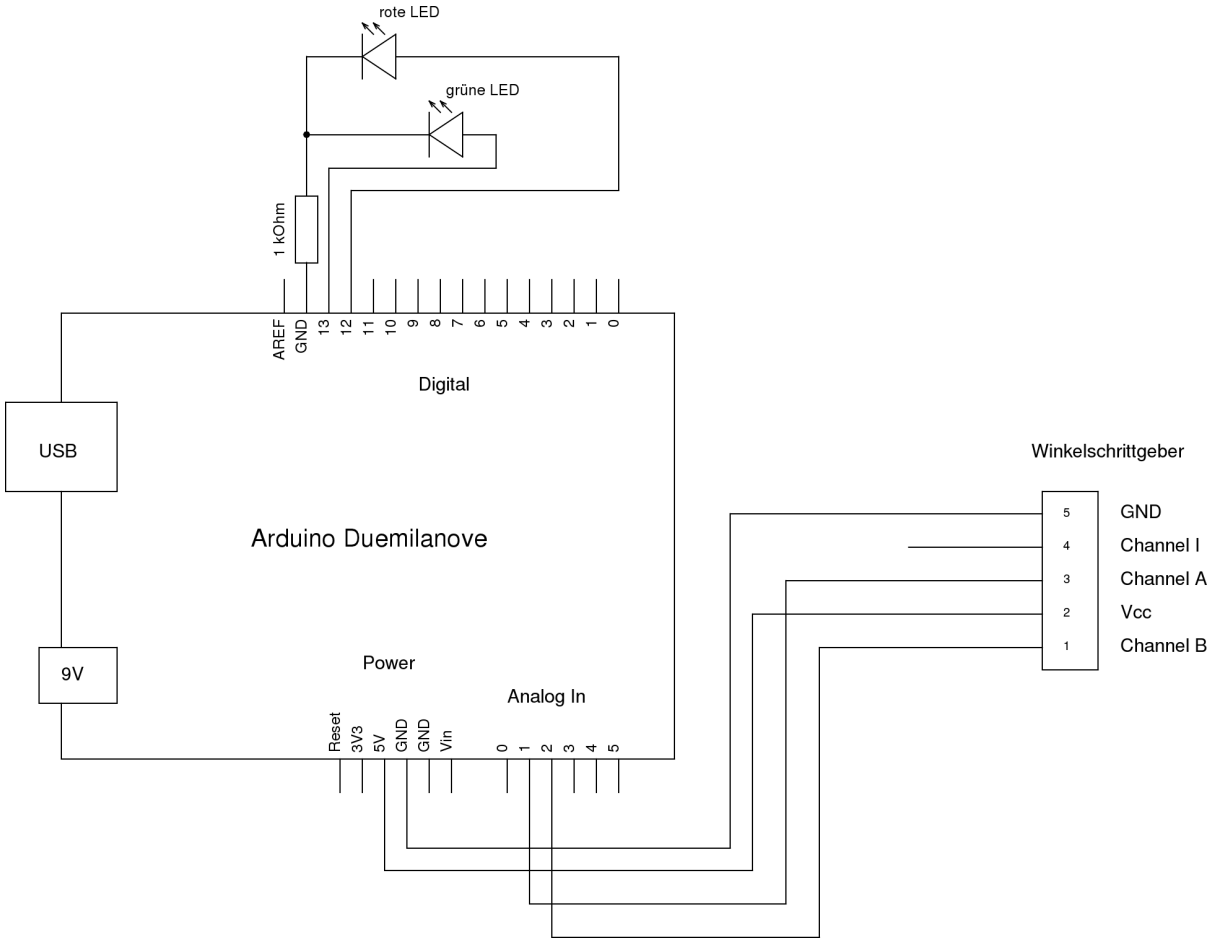

Abbildung 3.7: Schematische Darstellung der Verbindungen Arduino-Board - Winkelschrittgeber

Da der Arduino-Mikrokontroller, in der Lage ist, alle 10-15 ms ein Signal über die USB-Verbindung zu senden, lassen sich ab einer gewissen Winkelgeschwindigkeit, des großen Zahnrades (bzw. der Lochscheibe) nicht mehr alle Winkelschrittgebersignale mithilfe des Ardunio Boards registrieren. Eine einfache Abschätzung mit $2 \cdot 500$ Signalen (hohe und tiefe Signale) pro Umdrehung und Kanal ergibt eine minimale Periode von

$$\omega_{\min} = \frac{1 \text{ Signal}}{15 \cdot 10^{-3} \text{ s}} = \frac{2 \cdot 2 \cdot 500 \text{ Signale}}{T_{\min}} \qquad \Rightarrow \qquad T_{\min} = 30 \text{ s}$$

Ab einer Motorspannung von ca. 2,5 V (was einer Winkelgeschwindigkeit von 29,5 s entspricht) lässt sich beobachten, dass die beiden LEDs mit maximaler Frequenz leuchten. Das entspricht ziemlich genau dem, was nach der obigen Abschätzung zu erwarten war. Wird nun die Spannung erhöht, so erfasst das Arduino Board nicht mehr alle Winkelschrittgebersignale und es ist nicht mehr erkennbar, welcher der Känale dem anderen vorauseilt. Das führt dazu, dass der Drehsinn der Rotation nicht mehr ermittelt werden kann.

3.3.3 Plots

In der Abbildung 3.8 sind die Signale der Kanäle A und B über die Zeit nach Programmstart bei einer Motorspannung von 0,5 V aufgetragen. Das Signal des Kanals A wird jeweils durch die rote Linie und das des Kanals B durch die grüne Linie dargestellt. Die Form der Signale stimmt mit den theoretisch zu erwartenden Rechtecksignalen gut überein. Abweichungen kommen vor allem dadurch zustande, dass das Programm, wie im vorherigen Abschnitt beschrieben, die Signale nicht permanent, sondern nur alle 10-15 ms aufzeichnet.
Es lässt sich erkennen, dass bei Drehung im Uhrzeigersinn der Kanal A dem Kanal B zeitlich vorauseilt. Hingegen folgen die Kanäle bei einer Drehung gegen den Uhrzeigersinn in umgekehrter Weise aufeinander.

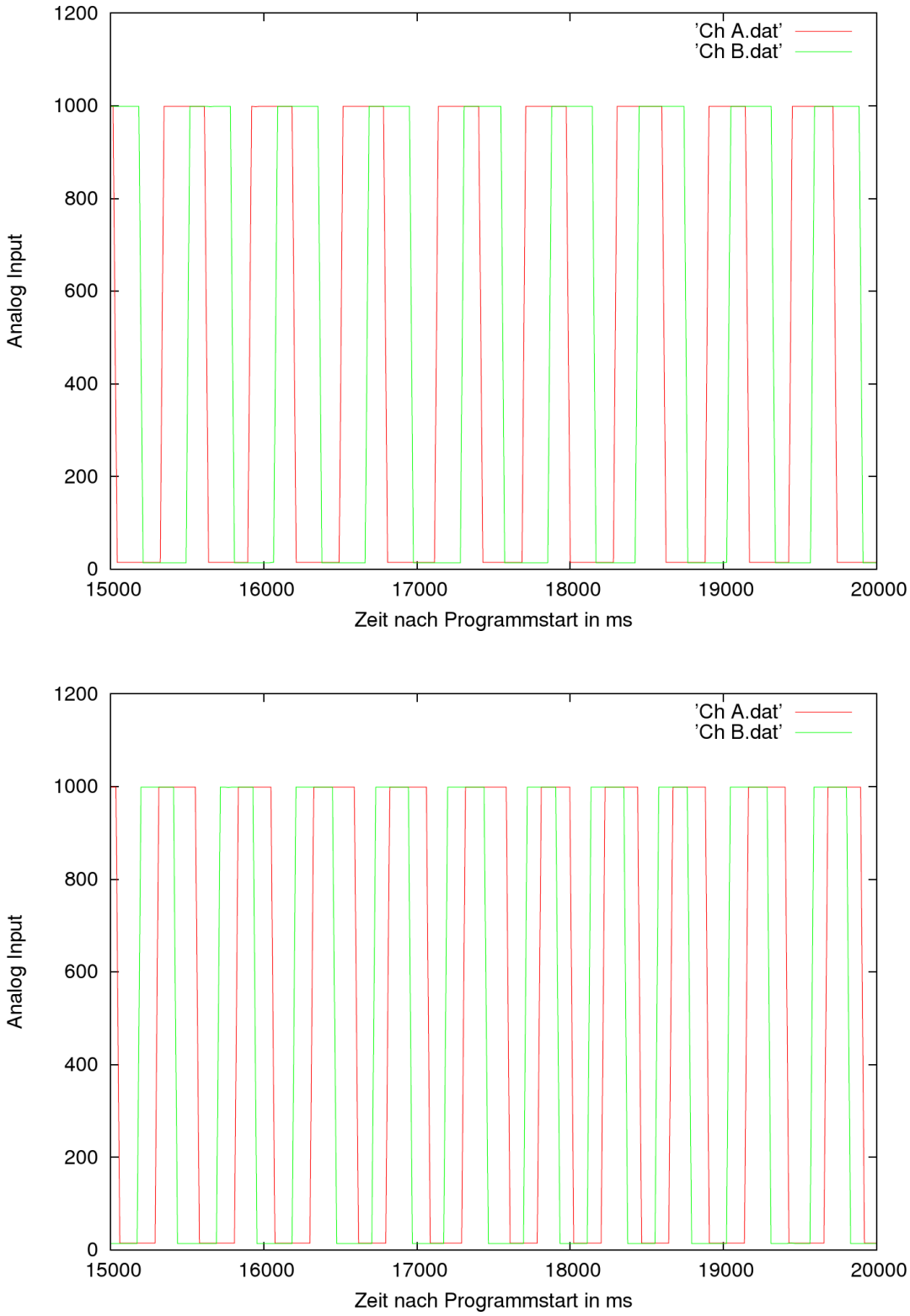

Abbildung 3.8: Auf dem oberen Teil der Abbildung ist die zeitliche Abfolge der Rechtecksignale bei einer Drehung im Uhrzeigersinn zu sehen. Unten ist die entsprechende Abfolge bei einer Drehung gegen den Uhrzeigersinn dargestellt.

4 EPICS

4.1 Grundlegendes

EPICS [9] (Experimental Physics and Industrial Control System) ist ein Open-Source-Kontrollsystem-Framework. Es ist auf diversen Plattformen lauffähig und bietet neben einem ausgereiften Netzwerkprotokoll diverse Tools und Treiber. Die Kontrollserver werden bei EPICS als Input Output Controller (IOC) bezeichnet, das Benutzerinterface als Operator Interface (OPI). Für den S-DALINAC werden derzeit IOCs zum Steuern und Überwachen der digitalen HF-Regelung sowie der CPS05-Netzteile zur Energieversorgung des Beschleunigers entwickelt. Als OPI wird derzeit Control System Studio (CSS) [11] erprobt.
EPICS arbeitet mit sogenannten Prozessvariablen (PV), die einen Parameter eines Geräts darstellen (z. B. Spannung, Temperatur, Position oder der Zustand eines Ventils). Sie werden in einem IOC über Records definiert, für die es wiederum - abhängig von der Verwendung - verschiedene Typen gibt (z. B. Analog In, Digital Out und Sequence). Eine PV hat einen eindeutigen Namen, der meist so gewählt wird, dass er dem entsprechenden Gerät zugeordnet werden kann (z. B. LINAC:BPM4:xPosition oder BOOSTER:gateValvePosition). Zusätzlich besitzt jede PV Attribute (Fields), in denen u. a. obere und untere kritische Werte, ein Zeitstempel oder die physikalische Einheit eingetragen werden können. Da all diese Informationen in EPICS-eigenen „Datenbanken“ gespeichert und miteinander verknüpft werden, hängt die Geschwindigkeit des gesamten Systems maßgeblich vom Design der „Datenbanken“ ab.
Man kann EPICS als Client/Server-Modell verstehen, dass ein eigenes, effizientes Kommunikationsprotokoll, den Channel Access, zum Datenaustausch zwischen Clientes und Servern verwendet. Dabei stellen die Channel Access Server (CAS) die Prozessvariablen über den Channel Access zur Verfügung, während die Channel Access Clients (CAC) auf diese zugreifen und über ihren Verwendungszweck bestimmen.
Die Installation eines EPICS-Systems wird im Anhang A.4 näher ausgeführt.

4.2 IOC-Server

Als IOC werden bei EPICS-Kontrollsystemen Server bezeichnet, an die Hardware angeschlossen ist, die über das Kontrollsystem beeinflusst/ausgelesen werden soll. Der IOC übernimmt dabei die Aufgabe die Kommunikation zur Hardware über gerätespezifische Schnittstellen/Protokolle abzuwickeln und zum Netzwerk hin standardisiert (über das EPICS-eigene Netzwerkprotokoll Channel Access) verfügbar zu machen.
Ein Client greift immer nur über den IOC auf die Hardware zu und muss daher keinerlei Kenntnisse über die konkrete Realisierung der am IOC angeschlossenen Hardware besitzen. IOCs laufen traditionell auf VME-Hardware, in den letzten Jahren jedoch auch vermehrt auf Linux- bzw. Windows-PC-Hardware, auf Macs sowie auf den unterschiedlichsten ins Gerät integrierten Embeddedsystemen.
Das Einrichten des Beispiel-IOC-Servers kann im Anhang A.5 nachgeschlagen werden.

4.3 Control System Studio

Control System Studio (CSS) ist eine Benutzeroberfläche für verschiedene Kontrollsysteme (z.B. EPICS), die am DESY entwickelt wurde. CSS basiert auf Eclipse und ist plattformunabhängig. Mittlerweile werden in der Version der Spallation Neutron Source (SNS) [11] sogar 64bit Versionen von Microsoft Windows und Linux unterstützt.
Nachdem das Programm auf der Seite des DESY [10] heruntergeladen wurde, kann das zip-Archiv entpackt werden und die ausführbare Datei css gestartet werden. Unter Linux gibt es möglicherweise Probleme mit der grafischen Oberfläche, deshalb sollte das Programm mit dem Aufruf

```
GDK_NATIVE_WINDOWS=1 ./css
```

gestartet werden. Beim Start erscheint ein Login-Fenster, das durch einen Klick auf Cancel jedoch ignoriert werden kann. Danach öffnet sich der Welcome-Bildschirm, in dem die wichtigsten Elemente der Oberfläche kurz erklärt werden. Als erstes sollte die IP-Adresse des IOCs unter `CSS -> Preferences -> CSS Core -> EPICS` im Feld addr_list eingetragen werden. Liegen der IOC und CSS auf demselben Computer, so ist hier die IP 127.0.0.1 einzutragen. Außerdem sollte der IOC gestartet sein, bevor die PVs in CSS verwendet werden.
Mithilfe von `CSS -> Display -> Synoptic Display Studio` wechselt man in die Ansicht des Synoptic Display Studio (SDS), mit dem Displays entwickelt werden können, die PVs anzeigen und verändern können. Um ein Display zu bearbeiten, sollte die Palette mit den Widgets (Anzeige- und Kontrollwerkzeuge) aktiviert werden: `Window -> Show View -> Other ... -> General > Palette`. Soll nun eine PV einem Widget zugeordnet werden, so muss in den Widget Properties (`Window -> Show View -> Other... -> Widget Properties`) der entsprechende Alias (z. B. Name: channel, Value: epics://rootHost:xxxExample) eingetragen werden. Ein Display wird per Klick auf den grünen Knopf in der oberen Leiste gestartet.
Weitere Anleitungen und Hinweise finden sich im Wiki des ikpweb [12].

4.4 RDB Channel Archiver

Sollen Daten über einen längeren Zeitraum gespeichert werden, so geschieht dies meistens entweder in Datenbanken oder einfachen Dateien. Weder für die erste noch für die zweite Variante gibt es im Moment ein Standardprogramm unter EPICS. Da aber bereits ein PostgreSQL im Netzwerk des IKP existiert, über den die gespeicherten Daten für das gesamte Netzwerk zugänglich sind, ist es geplant, die Daten der PVs in einer SQL-fähigen relationalen Datenbank (RDB) abzulegen. Der RDB Channel Archiver von SNS ist ein solches Programm, das sich aber noch in der Entwicklung befindet.

Im Folgenden soll beschrieben werden, wie mithilfe des RDB Channel Archivers in einer MySQL-Datenbank archivierte PVs dargestellt werden können.
In der CSS-Version 2.3.0 von SNS können Werte von Prozessvariablen, die in einer MySQL- oder ORACLE-Datenbank abgelegt wurden, im Datenbrowser geplottet werden, während die derzeitigen Werte dieser Prozessvariablen als Anzeige ausgegeben werden können. Dazu muss die URL zu der Datenbank unter `CSS -> Preferences -> CSS Application -> Trends -> Data Browser` im Feld `Archive Data Server URLs` eingetragen werden. Das Format der URL

unterscheidet sich, je nachdem welches Datenbanksystem verwendet wird.
Für MySQL:

```
jdbc:mysql://[host]:[port]/[database]?user=[user]&password=[pw]
```

Für ORACLE:

```
jdbc:oracle:thin:[user]/[pw]@//[host]:[port]/[database]
```

Evtl. ist es notwendig, in demselben Menü unter `RDB Archive` den Benutzernamen, das Passwort und die Archiv-Datenbank einzutragen.
Nun kann im Menü `Archive Search` (zu finden unter `Window -> Show View -> Other... -> Archive Search` die eben eingetragene URL ausgewählt und nach einer PV gesucht werden. Gibt man das Jokerzeichen * ein, werden alle PVs, die archiviert worden sind, angezeigt. Über `Rechtsklick -> CSS -> Data Browser` werden die archivierten Werte der ausgewählten PV geplottet. Falls eine Verbindung zu einem IOC besteht, der diese PV bereitstellt, so werden gleichzeitig auch die neuen Werte dieser PV im Datenbrowser angezeigt.
Der Aufbau einer RDB ist in dem Ordner `dbd/` des Pakets `org.csstudio.archive.rdb` beschrieben, dass sich in den SNS Sources befindet.

5 Ausblick

Im Rahmen dieser Miniforschung wurden wichtige erste Schritte zu einer neuen Goniometersteuerung geleistet. Ein Winkelschrittgeber vom Typ HEDS 5640 konnte mit einem Arduino Board auf einfache Weise ausgelesen und auf Drehrichtung hin untersucht werden. Ebenso wurde beschlossen, die gesamte Goniometersteuerung mithilfe von EPICS zu realisieren, für das eine Testumgebung binnen kurzer Zeit erstellt und erprobt werden konnte. Gleiches gilt für das CSS, über das die Steuerung letztlich erfolgen soll.
Dazu müssen jedoch erst Netzteile von Herr Bonnes entwickelt werden, welche die Motoren und insbesondere die Winkelschrittgeber versorgen. Hierbei muss noch explizit geklärt werden, ob und wie die neuen Winkelschrittgeber über die derzeitigen Anschlüsse versorgt und ausgelesen werden könnten. Die bestehenden Anschlüsse umfassen keine zusätzliche Spannungsversorgung für die Winkelschrittgeber, da diese bei den älteren Modellen vermutlich nicht notwendig waren.
Ferner ist es sinnvoll, die zurzeit eingesetzten DC Motoren an der Targetleiter, welche sich nicht im Vakuum befinden, durch Schrittmotoren zu ersetzen. Dadurch könnten diese Freiheitsgrade mit erhöhter Genauigkeit angesteuert werden. Im gleichen Zug sollte über Endschalter für die Targetröhre nachgedacht werden, weil es in der Vergangenheit immer wieder zu dem Problem kam, dass die Targetleiter am höchsten Punkt der Targetröhre nicht mehr beweglich war.
Sobald diese Punkte abgearbeitet sind, muss ein EPICS-Server vor Ort aufgesetzt werden, der sämtliche Prozessvariablen, die zur Steuerung benötigt werden, bereitstellt. Geplant ist, diesen hinter der Beton-Abschirmung am QCLAM zu plazieren. Eventuell müssen zusätzliche EPICS-Treiber für die neuen Netzteile entwickelt werden. Anschließend kann die Entwicklung der Goniometersteuerung auf Softwareebene beginnen. Die Programmierung kann dank dem EPICS-eigenen Netzwerkprotokoll auf einem beliebigen Rechner durchgeführt werden, der sich in demselben Netzwerk wie der EPICS-Server befindet. Wichtig ist, dass auf die in der Einleitung erwähnten Probleme der alten Software Rücksicht genommen wird, um eben solche Fehler zu vermeiden. Die höchsten Anforderung an die neue Steuerung wird jedoch sein, dass diese auf allen Ebenen leicht nachvollziehbar und erweiterbar ist.
Was außerem noch aussteht, ist das Protokollieren der Werte dieser PVs in eine Datenbank, um evtl. auftretende Fehlfunktionen nachvollzuziehen und zu beheben. Nach Angaben der EPICS-Mailing-Liste ist es problemlos möglich ca. 10000 PVs in einer RDB zu archivieren, solange nicht die Werte aller PVs jede Sekunde mitgeschrieben werden sollen. Folglich ist der RDB Channel Archiver auch bei den ca. 200 Netzteilen am Beschleuniger einsetzbar, von denen jedes durch ungefähr 50 PVs definiert ist.

A Anhang

A.1 Installation von Arduino unter Ubuntu

Damit die Arduino Software installiert werden kann, müssen folgende Pakete (z. B. mit Synaptic) installiert werden:

- die Java Runtime von sun (`sun-java6-jre`)
- `gcc-avr` in der Version 4.3.2 oder höher
- `avr-libc`

Außerdem sollte das Paket `brltty` entfernt werden, falls es installiert ist (das ist unter Ubuntu 10.04 standardmäßig der Fall). Anschließend lädt man die Arduino Software [13] herunter und entpackt bzw. kopiert die Dateien an den gewünschten Ort.
Hinweis: Falls ein 64bit System verwendet wird, so muss noch das Paket `librxtx-java` nachinstalliert und die Datei `librxtxSerial.so` im Verzeichnis `arduino-0019/lib` entfernt werden. Ferner kann es auf 64bit System bei Versionen vor Arduino 19 zu Problemen mit dem Arduino-eigenen Programm `avrdude` kommen.
Gestartet wird die Software nun über das ausführbare Skript `arduino` im entpackten Verzeichnis oder per Konsole:

```
sh arduino
```

A.2 Quellcode des Arduino-Skripts

```
1  #include <SimpleMessageSystem.h>
2
3  int ledPin[2] = {12,13}; // die LEDs sind an Pin 12 und 13 angeschlossen
4  int aPin = 1; // Pin für Channel A
5  int bPin = 2; // Pin für Channel B
6  int signAval = 0; // Signal von Channel A
7  int signBval = 0; // Signal von Channel B
8
9
10 void setup() {
11     // Setze die Pins der LEDs auf OUTPUT
12     for(int i=0;i<2;i++) pinMode(ledPin[i], OUTPUT);
13     // Setze die Pins für Ch A und Ch B auf INPUT
14     pinMode(aPin, INPUT);
15     pinMode(bPin, INPUT);
16     Serial.begin(9600);
17   }
18
19 void loop() {
20     signAval = analogRead(aPin); // lese Channel A
21     signBval = analogRead(bPin); // lese Channel B
22
23     // rote LED für A
24     if(signAval > 100){
25       digitalWrite(ledPin[0], signAval);
26     } else {digitalWrite(ledPin[0], LOW);}
```

```
27
28      // grüne LED für B
29      if(signBval > 100){
30        digitalWrite(ledPin[1], signBval);
31      } else {digitalWrite(ledPin[1], LOW);}
32
33      // Ausgabe in der Konsole
34      Serial.print("A");
35      Serial.print(millis());
36      Serial.print(" ");
37      Serial.print(signAval);
38      Serial.print('\n');
39
40      Serial.print("B");
41      Serial.print(millis());
42      Serial.print(" ");
43      Serial.print(signBval);
44      Serial.print('\n');
45  }
```

A.3 C++ Skript: sortA.cpp & sortB.cpp

Das C++ Skript sortA.cpp umfasst folgende Zeilen:

```
1   #include <fstream>
2   #include <iostream>
3   using namespace std;
4
5   int main(){
6       fstream f;
7       char cstring[256];
8       // Die zu bearbeitende Datei heißt 'datafile':
9       f.open("datafile", ios::in);
10      while (!f.eof()){
11          f.getline(cstring, sizeof(cstring));
12          // Wenn die Zeile als erstes Zeichen ein A hat,
13          // dann soll diese Zeile in der Konsole ausgegeben werden:
14          if(cstring[0]=='A') cout << cstring << endl;
15      }
16      f.close();
17  }
```

In sortB.cpp wird Zeile 14 entsprechend durch

```
        if(cstring[0]=='B') cout << cstring << endl;
```

ersetzt.

A.4 Installation von EPICS

Die Installation von EPICS erfolgte auf einem Linux-System, da hier C/C++-Compiler schon installiert waren und über die Kommandozeile sehr leicht verwendet werden konnten.
Nachdem das Archiv `baseR3.14.11.tar.gz` von der EPICS Homepage heruntergeladen wurde, legt man einen neuen Ordner unter `/opt` an (oder an einen beliebigen anderen Ort) und entpackt das Archiv dorthin.

```
cd /opt
sudo mkdir epics
sudo chown dirk epics
```

```
cd /opt/epics
tar -zxvf /home/dirk/Downloads/baseR3.14.11.tar.gz
ln -s base-3-14-11/ base
cd base
```

Anschließend müssen ein paar Umgebungsvariablen gesetzt werden, auf die EPICS zurückgreift.

```
export EPICS=/opt/epics
export EPICS_HOST_ARCH=linux-x86_64
export EPICS_BASE=/opt/epics/base
```

Achtung: Bei 32bit Systemen muss die Host-Architektur auf linux-x86 gesetzt werden. Damit diese Umgebungsvariablen auch beim nächsten Starten der Konsole gesetzt bleiben, müssen die drei export-Befehle auch in die Datei /etc/bash.bashrc eingetragen werden!

```
sudo apt-get install libreadline5 libreadline5-dev
make
```

Danach ist EPICS einsatzfähig. Getestet wurde diese Installation unter einem 64bit Debian Lenny und einem 64bit Ubuntu 10.04.

A.5 Einrichtung eines Beispiel-IOC-Servers

Das Einrichten des Beispiel-IOC-Servers erfolgt nach Kapitel 2 des Application Developers Guide [14], jedoch wird kein Sequencer verwendet.
Als erstes legt man einen Ordner (hier: myexample) an, in den der Beispiel-IOC letztlich installiert werden soll und ruft das EPICS-eigene PERL-Script `makeBaseApp.pl` auf, das C/C++-Dateien für das Beispiel erzeugt.

```
su
cd /opt/epics/
mkdir myexample
cd myexample/
/opt/epics/base/bin/linux-x86_64/makeBaseApp.pl -t example myexample
/opt/epics/base/bin/linux-x86_64/makeBaseApp.pl -i -t example myexample
```

Diese können nun initialisiert werden.

```
make
```

Jetzt ist der Beispiel-IOC fertig und kann gestartet werden.

```
cd iocBoot/ioctest/
../../bin/linux-x86_64/test st.cmd
```

In der nun gestarteten Kommandoshell kann man sich mit dem Befehl

```
dbl
```

sämtliche von diesem IOC verwalteten PVs anzeigen lassen. Der Befehl

```
dpbr rootHost:aiExample
```

bewirkt, dass einige Attribute dieser PV angezeigt werden. Führt man diesen Befehl einige Male hintereinander aus, so bemerkt man, dass diese PV immer wieder von 0 bis 9 hochzählt und dabei ihren Status verändert. Auf diese Art funktionieren auch die Widgets in CSS, welche dem Benutzer anzeigen, wie sich der Wert und der Status einer PV verhalten. Eine Liste aller Befehle kann mit dem Befehl

```
help
```

angezeigt werden.
Weiterhin kann ein Zufallszahlengenerator [15] als IOC aufgesetzt werden.

Literaturverzeichnis

[1] D. Kleinhanss, Diplomarbeit, Institut für Kernphysik, TH Darmstadt (1989)

[2] M. Kuss, Diplomarbeit, Institut für Kernphysik, TH Darmstadt (1990)

[3] H. Diesener, Diplomarbeit, Institut für Kernphysik, TH Darmstadt (1989)

[4] G. Hartung, Diplomarbeit, Institut für Kernphysik, TH Darmstadt (1989)

[5] E. Heid, Diplomarbeit, Institut für Kernphysik, TH Darmstadt (1993)

[6] Y. Kalmykow & A. Shevchenko, *Recovery and Further Improvement of the Goniometer and Target Ladder for the QCLAM Spectrometer at the S-DALINAC*, TU Darmstadt (2000 - 2001)

[7] http://ikpweb.ikp.physik.tu-darmstadt.de/mediawiki/index.php/QCLAM

[8] Agilent Technologies, *Quick Assembly: Two and Three Channel Encoders* (2002)

[9] http://www.aps.anl.gov/epics/

[10] http://css.desy.de/content/e413/index_eng.html

[11] http://ics-web.sns.ornl.gov/css/products.html

[12] http://ikpweb.ikp.physik.tu-darmstadt.de/mediawiki/index.php/CSS

[13] http://arduino.cc/en/Main/Software

[14] http://www.aps.anl.gov/epics/base/R3-14/11-docs/AppDevGuide.pdf

[15] https://pubweb.bnl.gov/~mdavidsaver/epics-doc/epics-devsup.html

Danksagungen

An erster Stelle gilt mein Dank Herrn Prof. Dr. Peter von Neumann-Cosel dafür, dass er mir die Möglichkeit gegeben hat, diese Arbeit in seiner Arbeitsgruppe durchzuführen sowie für das Vertrauen, das er mir entgegebrachte und das Interesse an meinen Ergebnissen.

Ganz besonders möchte ich mich bei meinen Betreuern Dipl.-Phys. Simela Aslanidou und Dipl.-Phys. Jonny Birkhan für die umfangreiche Unterstützung bei experimentellen, fachlichen, aber auch formalen Fragestellungen bedanken.

Ebenso danke ich Dipl.-Phys. Christoph Burandt und Dipl.-Phys. Martin Konrad für die Hilfe bei zahlreichen Fragen rund um EPICS.

Weiterhin bedanke ich mich bei allen weiteren Mitarbeitern der AG von Neumann-Cosel für die anregenden Diskussionen und das überaus positive Arbeitsklima.