Entwicklung einer Kalibriereinheit für das Goniometer

Miniforschung
Betreuer: Jonny Birkhan

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Inhaltsverzeichnis

1 Einführung in das Projekt

Diese Miniforschung soll bei einem Koinzidenzversuch am QCLAM-Spektrometer (A(e,e'x)B, wobei x = p,d,α,γ,n) dienlich sein. Dabei wird der Ort der Reaktion als Pivot-Punkt bezeichnet (Im Falle einer exakten Ausrichtung des Targets und des Strahls).

Die nach dem Stoß abgelenkten Elektronen werden im QCLAM-Spektrometer auf ihre Energie hin analysiert. Gleichzeitig sind um das Target herum Detektoren für die emmittierten Sekundärteilchen "x" unter verschiedenen ϕ- und θ-Winkeln aufgestellt. Um nun eine genaue Aussage über den Stoß machen zu können ist es wichtig zu wissen, unter welchem Winkel gegenüber dem primären Elektronenstrahl der Detektor steht, welcher das Sekundärteilchen detektiert hat. Bisher wurde die Position der Detektoren, welche in der technischen Zeichnung angegeben wurde als exakt angenommen. Tatsächlich muss dies nicht korrekt sein. Im Rahmen dieser Arbeit soll eine Sonde entwickelt werden, die imstande ist die Position der Detektoren zu bestimmen. Damit soll es ermöglicht werden, Abweichungen zwischen der technischen Zeichnung und dem tatsächlichen Aufbau zu zeigen.

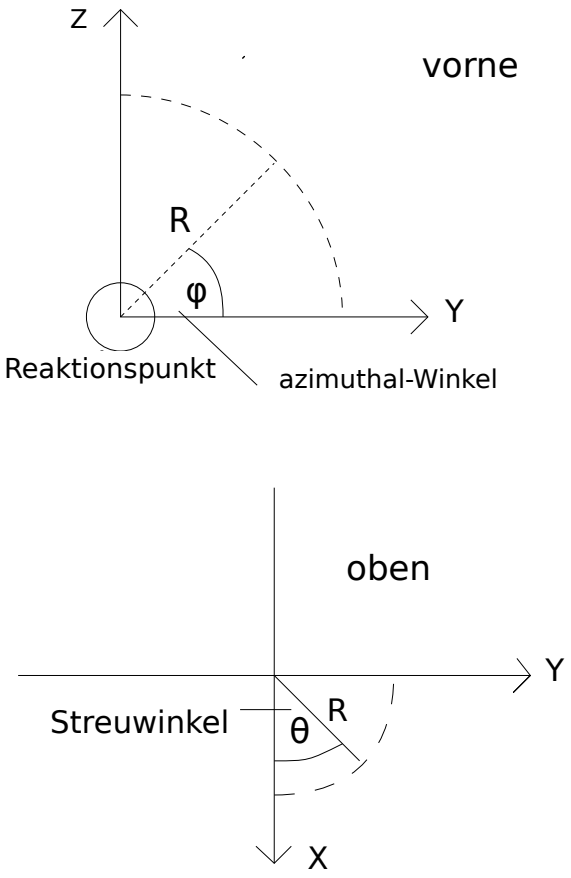

Abbildung 1: Winkeldarstellung

1.1 Messmethode

Zur Bestimmung der Position der Detektoren werden folgende Größen benötigt, die Distanz zwischen Reaktionspunkt und Detektoroberfläche und die beiden Raumwinkel der Detektorflächen. Um die Winkel auszumessen, wurde pro Winkel ein Schrittmotor verwendet. Somit ließ sich aus der Anzahl der Schritte des jeweiligen Motors der gedrehte Winkel rekonstruieren. Die Bestimmung der Entfernung erfolgte über die Verwendung einer Ultraschall- bzw. Infrarotsonde. Für die Messung gab es zwei Möglichkeiten. Entweder wurde ein Winkel gezielt angefahren und die Entfernung ausgemessen oder ein automatischer Scan fuhr den gesamten Raumwinkelbereich ab und nahm die Entfernung nach jedem Schritt auf.

Aus diesen Daten konnte ein Positions-Entfernungs-Spektrum erstellt werden, welches sich als Konturplot darstellen ließ.

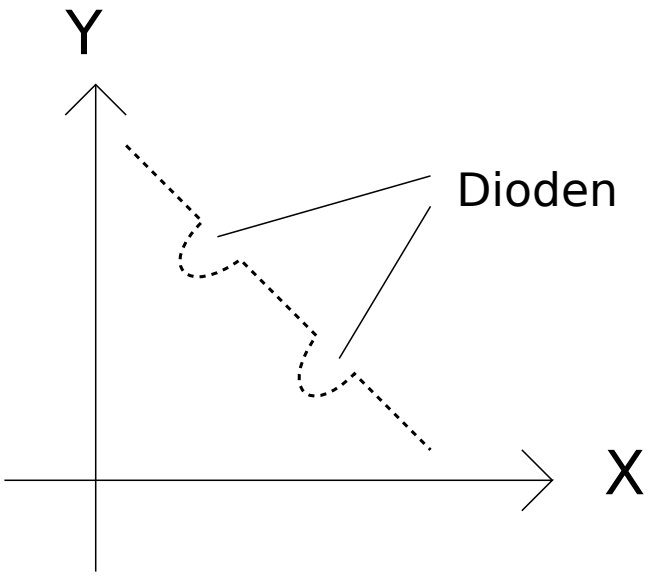

Abbildung 2: Abbildung der Dioden im Spektrum im Optimalfall

2 Technische Realisierung

Um die Schrittmotoren und Sonden gezielt anzusteuern wurde ein Microcontroller benötigt. Die Wahl fiel auf das *Arduino Duemilanove,* da dieses günstig und einfach zu bedienen war.

2.1 Arduino-Board

Das Duemilanove verfügt über 13 digitale, 6 analoge und einen Anschluss für 5 V stabilisierte Spannung. Außerdem gibt es Anschlüsse für Erdung sowie einen USB-Port, über den die Datenübertragung und die Stromversorgung läuft(Siehe Abb.3).

Die digitalen Anschlüsse werden im Betrieb zum Großteil von dem Motor Shield belegt und sind daher für die Ansteuerung der Sonden ungeeignet. Aus diesem Grund wurden die die analogen Anschlüsse zum Ansteuern und Auslesen der Sonde zu verwendet, da diese ebenfalls als digitale Anschlüsse verwendet werden können (in der Software durch *digitalPin(x)*, wobei die Integer *x* von 14 bis 19 die analogen Anschlüsse ansteuert). Die Betriebssprache des Boards ist C++. Da dies eine objektorientierte Sprache ist, vereinfachte es das Einbinden von Motoren als Objekte. Die Übertragung des Programmcodes lief über

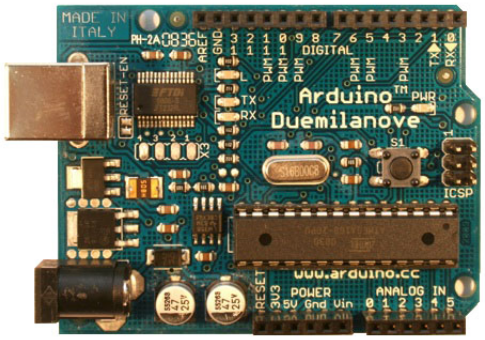

Abbildung 3: Arduino-Board

den USB-Port und das bereitgestellte Arduino-programming-kit [1]. Die Board-programmierung erlaubt das Auslesen und Ansteuern der Anschlüsse und ist in der Lage einfache mathematische Rechnungen durchzuführen.

2.2 Motor Shield und Schrittmotoren

Das *Motor Shield* dient der unkomplizierten Ansteuerung der Schrittmotoren. Ohne das Shield müssten die einzelnen Pole des Motors in richtiger Reihenfolge angesteuert werden um eine Drehung durchzuführen. Das Shield verfügt über Schaltkreise, die dies bereits implementiert haben. Somit muss im Programmcode lediglich eine Library eingebunden werden, welche die Funktionen des Shields zur Verfügung stellt.

Es verwendet alle digitalen Anschlüsse außer dem 2. und dem 13. und ist in der Lage bis zu zwei Schrittmotoren anzusteuern. Da diese eine Spannungsversorgung von 9 V benötigen wird eine zusätzliche Spannungsquelle benötigt, diese kann mit dem auf dem Shield angebrachten Anschluss verbunden werden(Siehe Abb.4)(es war möglich einen Motor ohne zusätzliche Spannungsversorgung zu betreiben, jedoch sollte dies vermieden werden um die Belastung für das Arduino Board so gering wie möglich zu halten).

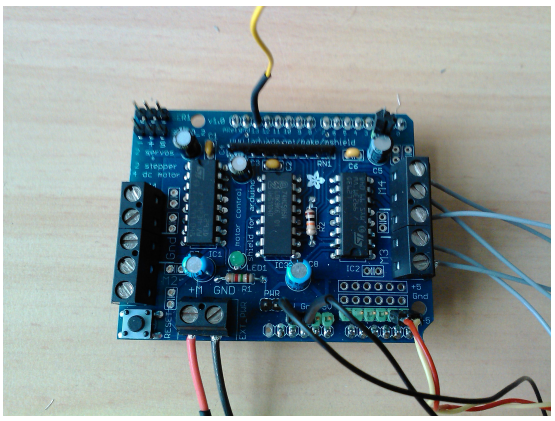

Abbildung 4: Motor-Shield

Es gibt zwei Sorten von Schrittmotoren, welche sich für die Verwendung in der Kalbriereinheit eignen. Dies sind Bi-Polare und Uni-Polare Motoren. Im Rahmen dieser Arbeit wurden Uni-Polare Motoren verwendet, welche sich durch 5 Pole (im Gegensatz zu 4 bei Bi-Polaren) auszeichnen. Damit die Mo-

[1] http://arduino.cc/hu/Main/Software

toren sich wie gewünscht drehen können, müssen die Pole in der richtigen Reihenfolge an das Board angeschlossen sein (Siehe Abb.5).

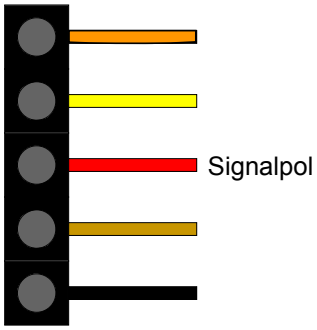

Abbildung 5: Anschlussschematik für die Motoren

2.3 Sonden

Zum Erfassen der Detektorflächen standen zwei Sensoren zur Verfügung, eine Ultraschall Sonde, welche sehr ausgiebig untersucht wurde, und eine Infrarot Sonde.

2.4 Ultraschall Sonde

Die Ultraschall Sonde vom Typ SRF05, verfügt über 2 Betriebsmoden. Sie kann in einer Anordnung verwendet werden, welche mit drei Anschlüssen auskommt, um Steckplätze auf dem Board zu sparen, oder mit vier Anschlüssen. Beiden Konfigurationen ist gemeinsam, dass die erste Lötstelle mit einer stabilisierten 5 V Spannungsquelle verbunden werden muss und die fünfte Lötstelle mit Ground (GND) verbunden wird. Für die Drei-Anschluss-Konfiguration musste die vierte Lötstelle der Sonde mit GND kurzgeschlossen werden. Danach konnte die Signal-IN-Verbindung ebenfalls als Echo für den Signalempfang verwendet werden.

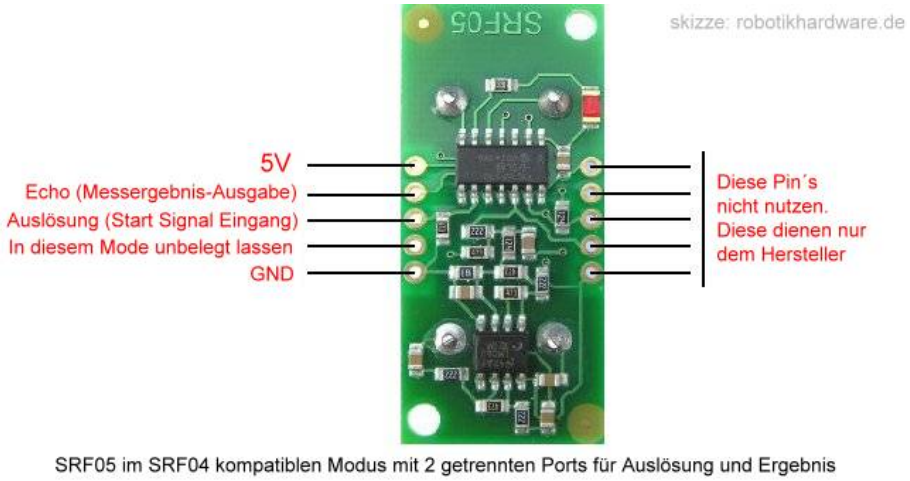

Abbildung 6: Anschlussschematik für die Ultraschallsonde

Erste Tests mit der Drei-Anschluss-Konfiguration zeigten, dass diese Konfiguration stark fehleranfällig war. Nach Aussenden des Pings wurde kein Echo empfangen. Nach diesem Ergebnis wurde entschieden, die weniger fehleranfällige Vier-Anschluss- Konfiguration zu verwenden, da kein Mangel an freien Anschlüssen bestand. Hierbei wurde die zweite Lötstelle als Signal Ausgang mit dem Board verbunden,

während die vierte Lötstelle unverbunden blieb. Nun konnte die Sonde über die dritte Lötstelle angesteuert werden und über die vierte Lötstelle Daten zurück senden. Diese Konfiguration lieferte bereits nach dem ersten Aufbau brauchbare Daten. Die Ultraschall Sonde wurde auf Software Ebene über die Methode *ping()* angesteuert.

2.5 Infrarot Sonde

Die Infrarot Sonde vom Typ GP2D12/15 wird über drei Anschlüsse mit dem Board verbunden, wobei der erste Anschluss die Spannungsversorgung und der dritte die Erdung(GND) darstellt. Die Sonde sendet über den zweiten Anschluss durchgängig analoge Daten, welche die von den Infrarotstrahlen induzierte Spannung darstellen. Um eine Entfernungsmessung mit der Infrarotsonde vornehmen zu können, ist es notwendig, die charakteristische Linie der Sonde aufzunehmen, um die gemessenen Werte in Verbindung mit Distanzen setzen zu können.

3 Software zur Steuerung und Datenaufnahme

3.1 Konzept zur Steuerung und Datenaufnahme

Die Hauptaufgabe des Programmcodes besteht darin, die Motoren zu steuern und die Sensoren auszulesen. Die Motoren sollen ausgewählte Winkel anfahren können und die Sensoren über eine GUI ein- bzw. ausgeschaltet werden. Die Schrittmotoren können jedoch nur, wie der Name bereits sagt, diskrete Schritte gehen. In einem späteren Testaufbau wird eine Übersetzung von $\frac{360^\circ}{530}$ gewählt. Im folgenden wird daher nur noch von Schritten und nicht von Winkeln gesprochen. Somit galt es, 2 Funktionen zu ermöglichen. Einerseits sollte der Motor diskrete Schritte anfahren können und die Sonden an dieser Position eine Messung vornehmen, oder es sollte ein automatischer Scan initialisiert werden, der sämtliche Schritte durchfährt und an jeder Position Entfernung und Winkel speichert.

3.2 Erste Messungen und gobetwino

Bei den ersten Messungen wurde *Gobetwino* verwendet. Dieses Programm hat die serielle Schnittstelle auf einlaufende Daten abgehört und diese gegebenenfalls abgespeichert. Damit waren jedoch nur statische Messungen möglich, d.h. vor jeder Änderung der Messvariablen (wie z.B. Schrittzahl/Position und Anzahl von Entfernungsmessungen) musste *Gobetwino* ausgeschaltet, dem Board die neuen Variablen hochgeladen und *Gobetwino* mit einem neuen Speicherverzeichnis eingeschaltet werden. Die Daten von dem Board bzw. der Ultraschallsonde wurden in einem txt-file für die weitere Auswertung abgespeichert. Damit war es bereits möglich charakteristische Linien von der Ultraschallsonde und einem Infrarotdetektor aufzunehmen und mit ersterem zusätzlich einen Rundscan in einem Testaufbau und im Goniometer vorzunehmen. Auf die Auswertung entsprechender Daten wird später näher eingegangen. Für den weiteren Verlauf der Programmierung war es zwar interessant, dass es *Gobetwino* mit wenig Aufwand erlaubte, Daten aus dem Arduino-board zu speichern, das Programm aber nicht annähernd alle Bedingungen für ein finales Programm erfüllte. Einerseits war es nur möglich, Daten, die über ein *print* vom Board geschickt wurden, zu empfangen, zum anderen konnte nur in einer Datei gespeichert werden, was den Programmfluss bei mehr als 100 Messdaten verlangsamte. Für die genannten Testmessungen reichte es jedoch aus.

3.2.1 Processing

Da die Möglichkeiten von gobetwino und der damit verbundene Aufwand sich als hinderlich erwiesen, war es nötig, eine sinnvollere Methode zur Steuerung der Motoren und Auswertung der Messdaten zu finden.

Um eine vollständige externe Steuerung des Arduinoboards zu ermöglichen, war der nächste Schritt ein Processing-Programm zu schreiben. Dieses sollte über die serielle Schnittstelle (genau genommen über eine virtuelle serielle Schnittstelle) Daten von dem Arduino-Board empfangen und auch an dieses Senden. Somit sollte eine Steuerung des Boards ohne Änderung der boardinternen Programmierung ermöglicht werden.

Der gesamten Programmierung der Steuerung liegt die Übergabe von Strings zugrunde, wobei das Programm um die Daten an die Serielle Schnittstelle zu übergeben, in Processing geschrieben wurde. Dem Arduino-board wird ein String übergeben. Das Board wertet nun die Stringeinträge, welche nacheinander in Form ihrer ASCI II - Codes übergeben werden, aus. Diese Form der Datenübertragung ausnutzend wird an erster Stelle des Strings stets das Steuerzeichen gesetzt, welches die folgenden Einträge des Strings an die richtige Funktion übermittelt.

$$10t : 10 \text{ Schritte in } \theta - Richtung \tag{1}$$

Die eingegebene Zeichenfolge wird vom Processingprogramm invertiert, womit das Board zunächst das Steuerzeichen und anschließend die Zahl in ansteigender 10er-Potenz übergeben bekommt. Somit kann die Zahl über eine Laufvariable für die Potenz aus den Stringeinträgen rekonstruiert werden. Um dem Programm das Ende des Strings zu übermitteln wird der ASCII-Code 10 (Enter) verwendet. Ist der String verarbeitet wird die Laufvariable der Potenz zurückgesetzt.

Dem String können nur Ziffern und Buchstaben übergeben werden, welche eine Bedeutung für den Arduino-Code besitzen. Innerhalb der Zeit der Miniforschung blieb keine Zeit für debugging und weitere Feinanpassungen. Demnach sollte sich an die Reihenfolge der Eingabe, wie folgt beschrieben wird, gehalten werden. Falsche Eingaben (z.B. mehrere Steuersignale hintereinander) sollten zwar ignoriert werden, können aber unter Umständen zu Problemen führen. Gleiches gilt für die Eingabe von sehr großen Zahlen ($> 10^6$), da diese unter Umständen nicht korrekt oder gar nicht gespeichert werden können und zu Problemen führen.

3.3 Programmierte Funktionen

Im folgenden soll auf die programmierten Funktionen des Programms eingegangen werden und wie diese angesprochen werden. Wird eine Funktion aufgerufen, die Daten speichern soll, werden im Verzeichnis des Programms entsprechende txt-files erstellt. Sämtliche Befehle müssen bündig und ohne Leerzeichen eingegeben werden. Den meisten der folgenden Funktionen wird eine Integer übergeben, welche die Schrittzahl oder die Anzahl der Messungen für eine Koordinate festlegt. Um die aktuelle Position der Motoren zu speichern, wird die Anzahl der gedrehten Schritte in einer Laufvariable abgespeichert. Deren Nullpunkt ist jedoch abhängig von der Motorposition beim Einschalten/anschließen des Arduino-Boards.

3.3.1 ϕ/θ-control()

Wird der Eingabe eine Ziffer, entsprechend der gewünschten Schrittzahl, und anschließend ”t” oder ”p” übergeben bewegt sich der entsprechende Motor die gewünschten Schritte. Dabei ist es sinnvoll die Verkabelung, bzw. die Erstellung der Motoren im Arduino derart anzupassen, dass mit t auch der θ-Motor und p der ϕ-Motor angesteuert wird. Ein näherer Test war leider nicht möglich, da am abschließenden

Testaufbau erst 1 Motor angeschlossen war, bzw. dieser Testaufbau nur eine Drehung um eine Achse zuließ (siehe Abb. 7).

$$10t : \text{10 Schritte in } \theta - Richtung \tag{2}$$

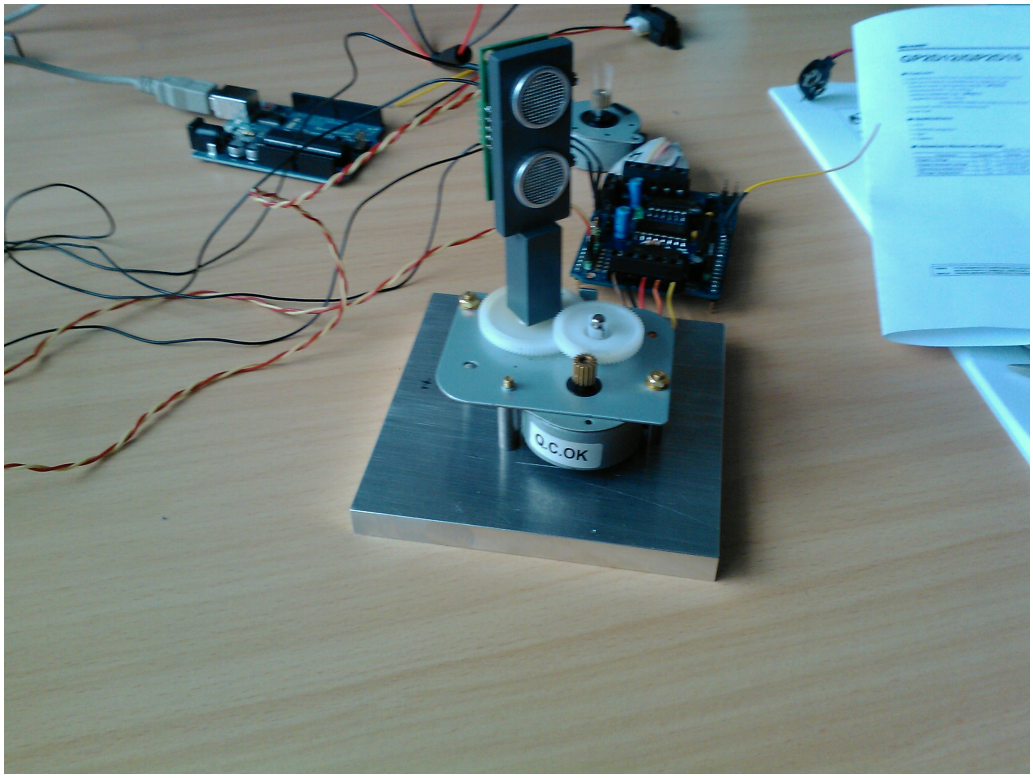

Abbildung 7: Aufbau zum Drehen der Schallsonde über einen Schrittmotor

3.3.2 ϕ/θpositioning()

Diese Funktion wird mit der Zahl und ”T” oder ”P”(Programm ist case-sensitive) gestartet. Es wird der Unterschied an Schritten zwischen der aktuellen und der gewünschten Position angesteuert. Für einen realen Aufbau im Goniometer gilt es, im Code des Arduino-board die Schrittanzahl für einen Umlauf, also die genutzte Übersetzung anzupassen.

$$10P : \text{Die Position 10 vom Nullpunkt des } \phi\text{-Motors wird angesteuert} \tag{3}$$

Ist die aktuelle Position mit der eingegebenen identisch, bewegt sich der Motor nicht, ist sie kleiner, bewegt sich der Motor den Unterschied rückwärts, ansonsten den Unterschied vorwärts.

3.3.3 Einschalten der Schallsonde - ping()

Wird dem Arduino-board ein ”s” gesendet, fängt dieses an, die Ultraschallsonde anzusteuern, welche pingt bis sie mit ”x” abgebrochen wird. Im Processing-Code wird nach jedem Ping die entsprechende Zeit in Mikrosekunden und die aktuelle θ/ϕ-Position in einem Array gespeichert und dieses nach Abschalten der Schallsonde in einer txt-file gespeichert. Für sämtliche Funktionen, die die Ultraschallsonde ansprechen und die Werte speichern sollen, sendet das Arduino-board das Steuerelement zurück an das

Processingprogramm, damit dieses die einkommenden Daten speichert, wenn erwünscht. Das heißt für den Ping, dass, falls das Board bereits vor Einschalten des Programms am Senden war, dieses erst wieder mit ”x” abgebrochen und per ”s” neu gestartet werden muss.

$$s: \text{ Startet Schallsonde} \tag{4}$$
$$x: \text{ Stoppt Schallsonde} \tag{5}$$

3.3.4 θscan

Diese Funktion wird mit einer Integer und anschließend ”X” initialisiert. Die Integer gibt an, wie viele Entfernungsmessungen pro Schritt vorgenommen werden sollen, wobei der Motor eine komplette Umdrehung vollführt. Nach einem Umlauf fährt der θ-Motor automatisch auf die Position $\theta = 0$ zurück. Da für jeden θ-Winkel ein eigenes txt-file erstellt werden soll, wird nach jedem Schritt vom Arduino-board ein ”x” an das Processing-Programm gesendet. Dieser Steuerbefehl veranlasst wie beim Ping ein Speichern des Arrays. Solange der Umlauf noch nicht komplett ist, wird nach dem Speichern ein ”X” vom Board an das Processing-Programm geschickt, damit dieses die einkommenden Daten in einem neuen Array ablegt. Somit erhält jeder Schritt ein eigenes txt-file.

$$10X: \text{ 10 Messungen pro Schritt bei einem Rundscan über } 360^\circ(\theta) \tag{6}$$

3.3.5 ϕscan

Diese Funktion läuft analog zu θscan ab, und erwartet ein ”Y” zur Initialisierung. Hier werden nach jedem ϕ-Schritt die Daten gespeichert.

3.3.6 fullscan

Dem Fullscan wird ebenfalls eine Integer und ”F” übergeben. Diese Funktion verhält sich wie die beiden oberen, nur wird der gesamte ϕ- und θ-Bereich abgescannt. Gespeichert wird nach jedem θ-Schritt, wobei zwischen den einzelnen θ-Schritten jeweils der gesamte ϕ-Winkel abgescannt wird. Allen 3 Scans liegt zugrunde, dass man sie jederzeit per ”x” im Processing-Programm abbrechen kann, wobei der Motor entsprechend des vorliegenden Scans zu $\theta = 0$ oder $\phi = 0$ oder beides zurückfährt. Bei einem realen Aufbau, sollte darauf geachtet werden, dass einer der beiden Motoren bei einem Winkel, der einer Schrittzahl von 180° ($\pm 90^\circ$) entspricht, abbricht.

3.3.7 actioncontrol

Diese Funktion hört die serielle Schnittstelle ab. Kommt ein String wird der erste Eintrag auf ein Steuerelement überprüft. Liegt eines vor, wird der Wahrheitswert (boolean) der entsprechenden Funktion auf true gesetzt und im Falle einer Funktion, für die das Processing-programm auch eine Initialisierung benötigt, das entsprechende Steuersignal zurückgesendet.

4 Messungen

4.1 Ultraschall Kalibrierung

Um die Daten der Ultraschallsonde sinnvoll interpretieren zu können, wurde die Sonde auf eine Halterung gesteckt. Diese wurde auf bekannte Distanzen gegenüber einem Hindernis eingestellt. Dies ermöglichte es, die gemessenen mit den eingestellten Entfernungen zu vergleichen, bzw. eine Kalibrierung der Sonde vorzunehmen.

4.2 Ultraschall Charakteristik

Um die Zuverlässigkeit der Ultraschallsonde zu bestimmen, wurde der Sensor vor einer Messschiene fixiert (siehe Abb. 8).

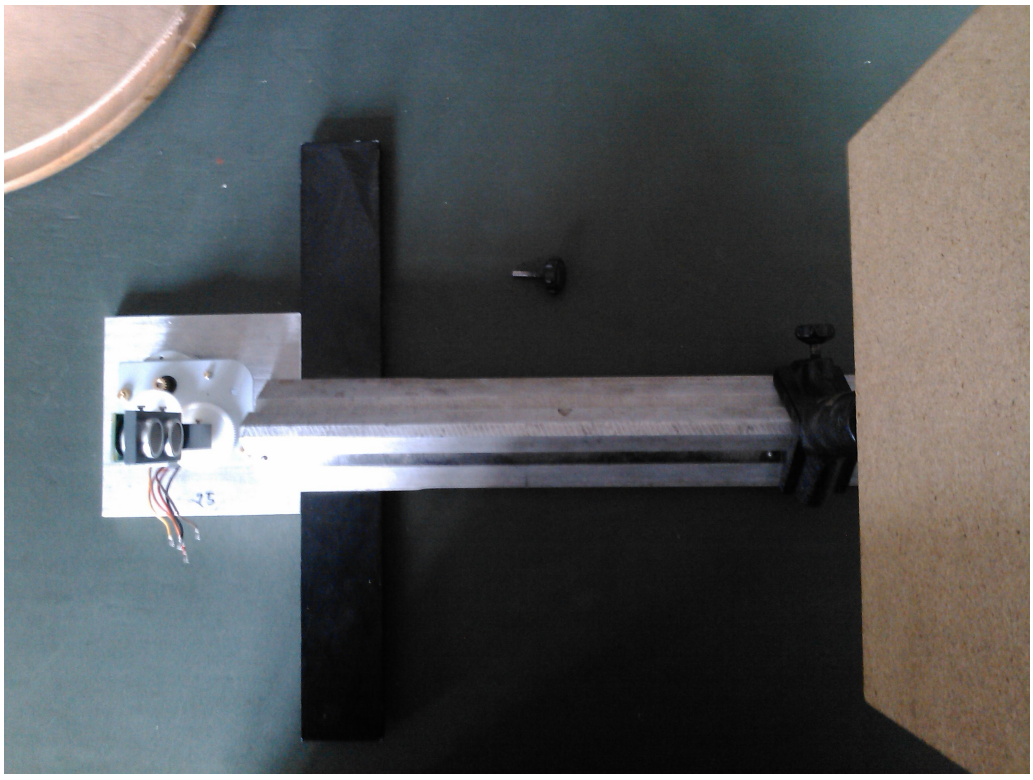

Abbildung 8: Aufbau für die Messung der Ultraschallcharakteristik, Ansicht von oben

Das auszumessende Ziel wurde durch eine Pressspan-Platte realisiert, welche mit einem Fuß über die Messschiene bewegt werden konnte. Unterschiedliche eingestellte Entfernungen wurden nun mit den Daten der Sonde verglichen.

Zum Vergleich der gemessenen Entfernungen, wurden diese durch die eingestellten Entfernungen geteilt und über die eingestellten Entfernung aufgetragen (siehe Abb. 9).

Dieser Plot zeigt deutlich, dass die von der Sonde gemessenen Entfernung gegenüber den eingestellten Entfernungen abweichen. Für große Entfernungen scheint diese Abweichung geringer zu werden. Weiterhin schwanken die Messwerte innerhalb der ersten 10 cm stärker als im übrigen Verlauf. Zudem

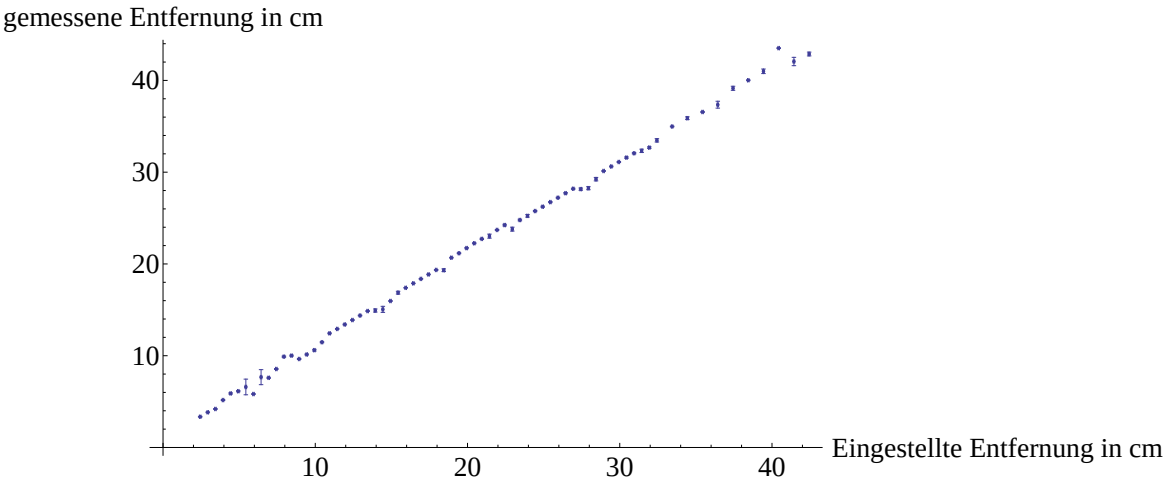

Abbildung 9: Ermittelte Entfernungen der Schallsonde, absolut gegenüber dem tatsächlichen Wert, Einheiten in cm, 1s-Umgebung, Die Unsicherheiten wurden aus der Standardabweichung der Messdaten konstruiert

fällt auf, dass in periodischen Abständen die Abweichung kurzfristig sinkt, dann jedoch wieder auf ca. 1,1 ansteigt (siehe Abb. 10). Dies deutet darauf hin, dass eine Verwendung der Ultraschallsonde auf Entfernungen unter 10 cm zu ungenauen Messwerten für die Entfernung führen könnte. Da die charakteristischen Entfernungen im Goniometer im Bereich von 5 bis 15 cm liegen, wäre es sinnvoll über die Verwendung einer Sonde mit besserer Kurzstreckenauflösung nachzudenken. Da eine systematische Abweichung von den erwarteten Werten vorliegt, wäre es sinnvoll gewesen, eine weitere Messung vorzunehmen und der Linie der Messdaten Entfernungen zuzuweisen. Dies konnte in der begrenzten Zeit jedoch nicht umgesetzt werden. Ebenso mangelte es an Zeit die periodischen Abweichungen der Sonde genauer zu untersuchen.

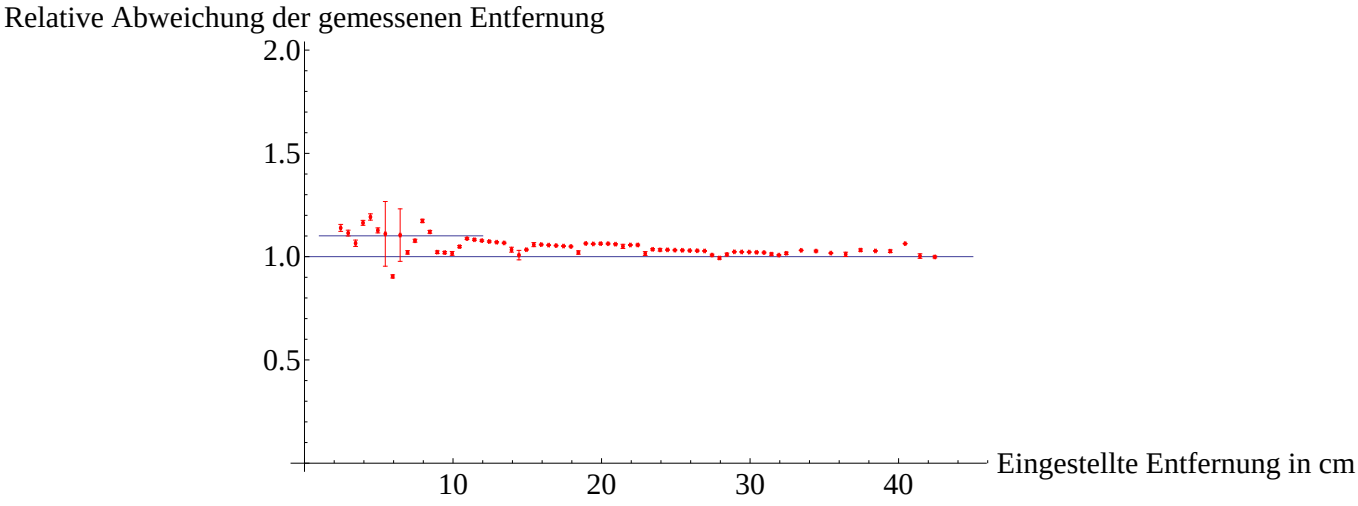

Abbildung 10: Ermittelte Entfernungen der Schallsonde, relativ gegenüber dem tatsächlichen Wert, Einheiten in cm, 1s-Umgebung, Die Unsicherheiten wurden aus der Standardabweichung der Messdaten konstruiert

4.3 Infrarot Charakteristik

Um die Daten der Infrarotsonde in Entfernungen umrechnen zu können, war es notwendig eine Kalibrierkurve aufzunehmen. Zu diesem Zweck wurde eine Anordnung gewählt, bei der die Sonde auf einer Messschiene fixiert und das Ziel auf bekannte Entfernungen eingestellt wurde.

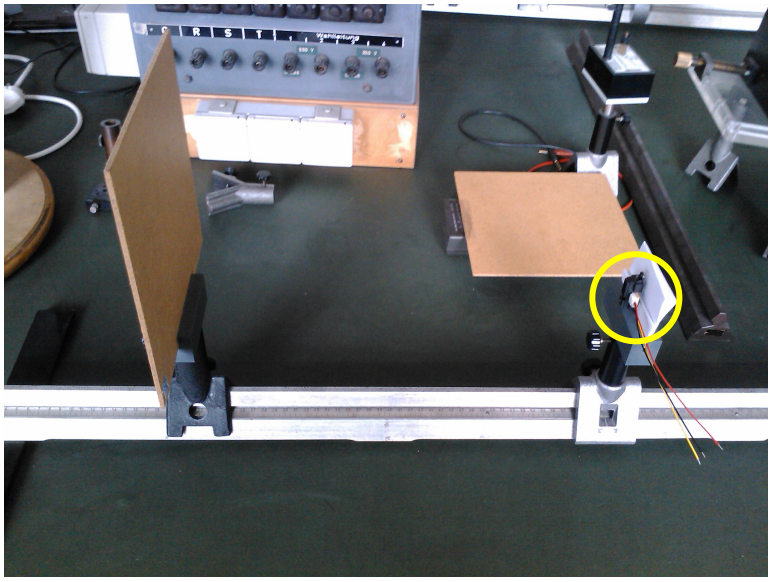

Abbildung 11: Infrarotcharakteristik

Der Plot zeigt die an der Sonde gemessenen Spannungen bei den eingestellten Entfernungen. Die Kurve spiegelt somit die charakteristische Kurve der Sonde wieder, die benötigt wird, um die Sensordaten in Entfernungen umzurechnen. Hierbei zeigt sich, dass die Schwankungen der Messwerte gegenüber der Ultraschallsonde geringer ausfallen. Es wurden keine Entfernungen unter 10 cm aufgenommen, da die Sonde nicht für diese Distanzen konstruiert ist, es wäre jedoch grunsätzlich möglich.

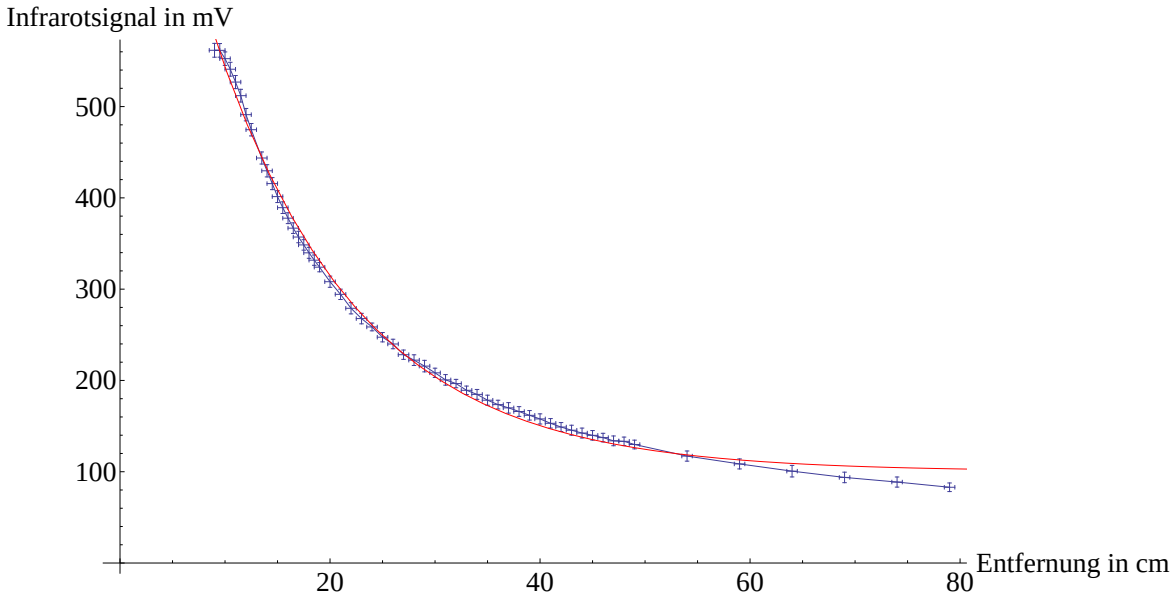

Abbildung 12: Charakteristische Linien der IR-Sonde in 1 Sigma Umgebung, Die Unsicherheiten wurden aus der Standardabweichung der Messdaten konstruiert, Es wurde ein Fit der Form $a \cdot Exp[-bx] + c$ mit a = 916; b = 0,073; c = 100,3 verwendet

4.4 Hindernis Parcours

Für die Aufgabe der Kalibriereinheit im Goniometer, ist die Winkelauflösung der Apparatur entscheidend, daher wurde ein Hindernis Parcours aufgebaut, in welchem die Verknüpfung von Ultraschallsonde und Motor, sowie die daraus resultierenden Daten untersucht werden konnten. Zu diesem Zweck musste zuerst ein Aufbau entwickelt werden, mit welchem ein Motor die Sonde ausrichten konnte(siehe Abb. 7). Nun konnte der Motor über ein Getriebe (Übersetzung: 530 Schritt entsprachen 360°) die Sonde drehen. Bei jedem Schritt wurden 2 Entfernungsmessungen durchgeführt (Problematik des *Gobetwinos* wie weiter oben beschrieben).

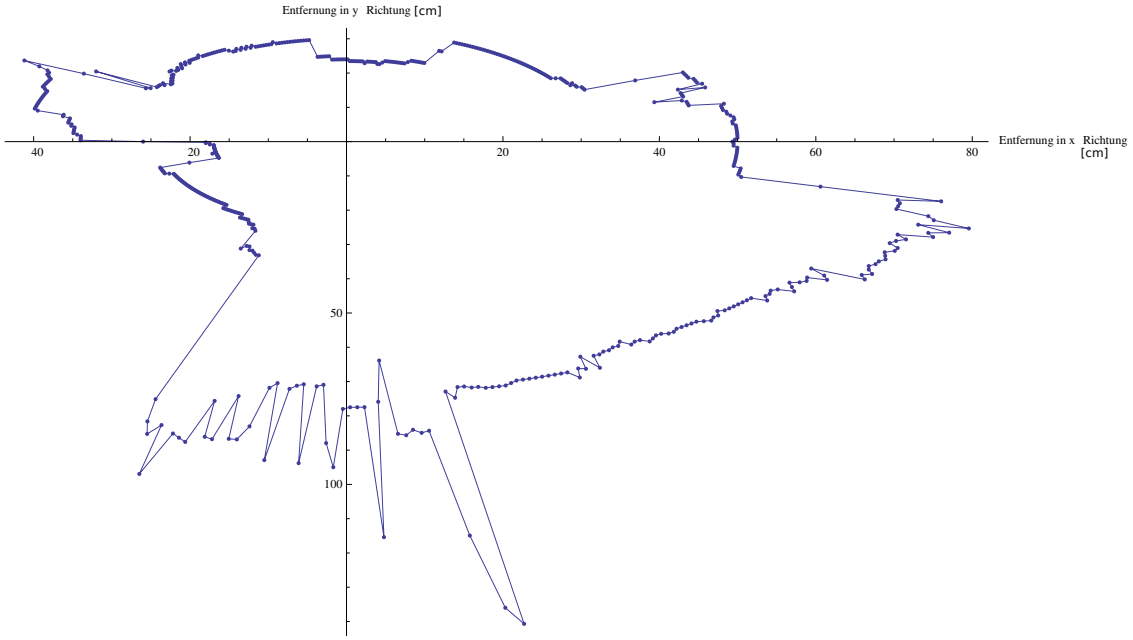

Abbildung 13: Rundplot der Aufnahme des Parkours

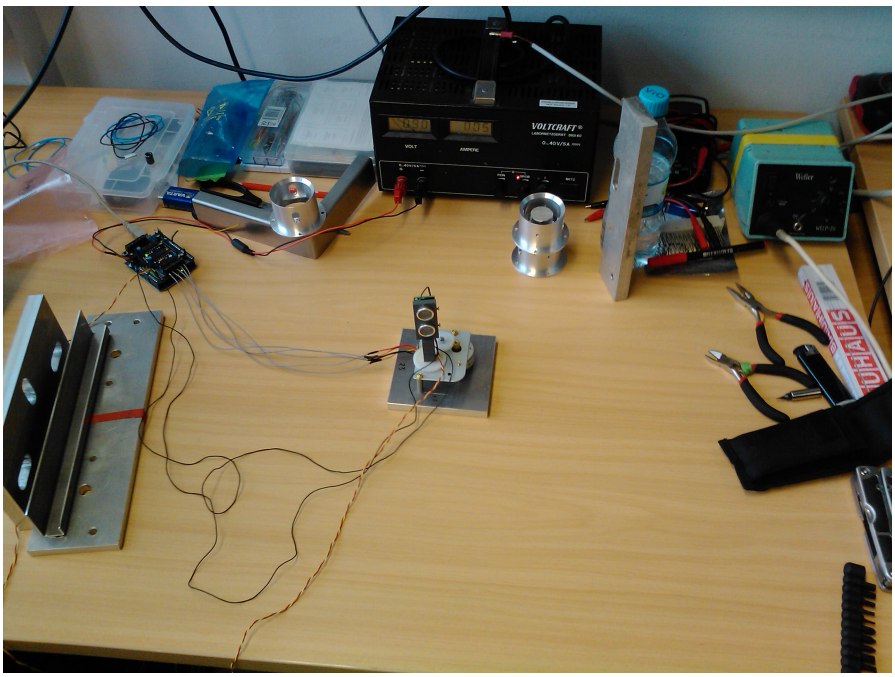

Abbildung 14: Hindernis Parcours

Da zu diesem Zeitpunkt der Softwareentwicklung noch *Gobetwino* benutzt wurde, war es nicht möglich mehr als zwei Messpunkte pro Schritt aufzunehmen. Dies resultiert aus der weiter oben beschriebenen Problematik mit dem Speicherverfahren von *Gobetwino*. Bei einem Vergleich des Aufbaus mit dem aufgenommenen Bild lassen sich die einzelnen Objekte grob zuordnen, ohne ein Bild des Aufbaus wäre dies jedoch nicht möglich.

4.5 Goniometer Test

Für einen Test der Sonde im Goniometer musste zuerst ein Aufbau entwickelt werden, welcher eine Montierung der Sonde im Inneren erlaubt (siehe Abb. 15). Zum Bewegen der Sonde wurde der zuvor entwickelte Aufbau verwendet und auf die Trägerschiene montiert.

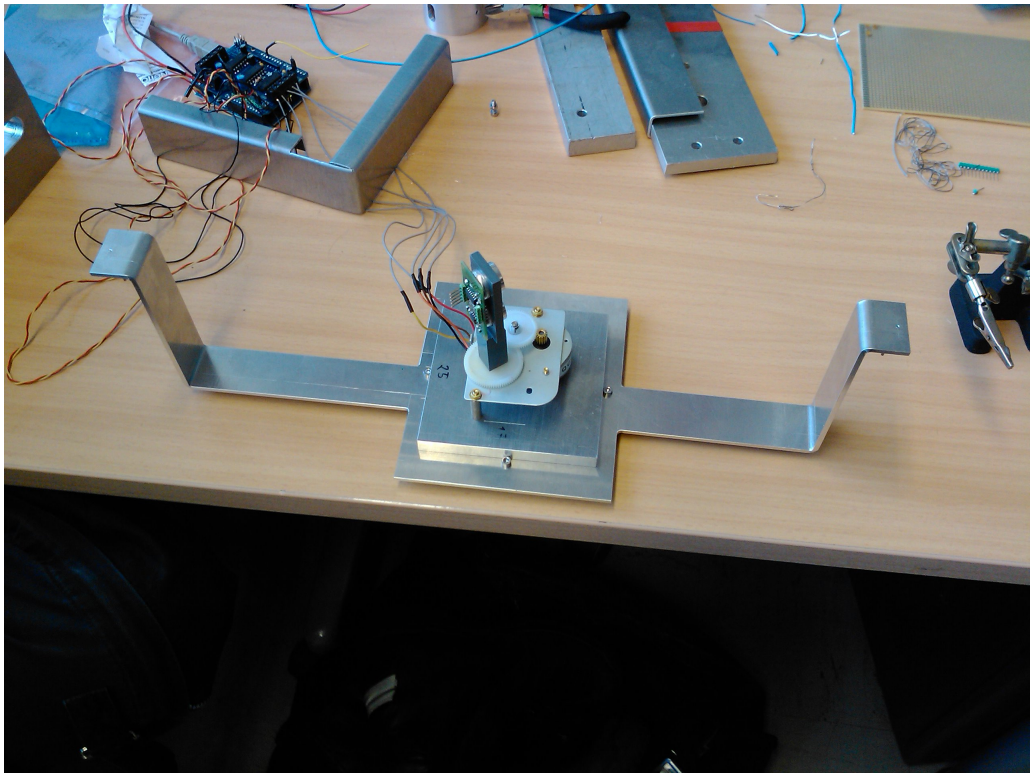

Abbildung 15: Sensorträger mit Sensor für den Gonjumetertest

Der Rundtest im Goniometer lieferte folgendes Bild

Zu diesem Zeitpunkt konnte die Processing-Software bereits verwendet werden, was eine Aufnahme größerer Datenmengen pro Schritt ermöglichte. Entsprechend wurden die statistischen Unsicherheiten gegenüber der Parcours-Messung reduziert. Auch hier ließen sich die Hindernisse grob zuordnen, wenn man den Aufbau kannte, es war jedoch nicht möglich die genauen Winkel oder Abgrenzungen der Hindernisse zu erkennen. Mit einer feineren Übersetzung wäre es möglich kleinere Winkelabstände zu vermessen. Dies sollte zu einer höheren Winkelauflösung führen.

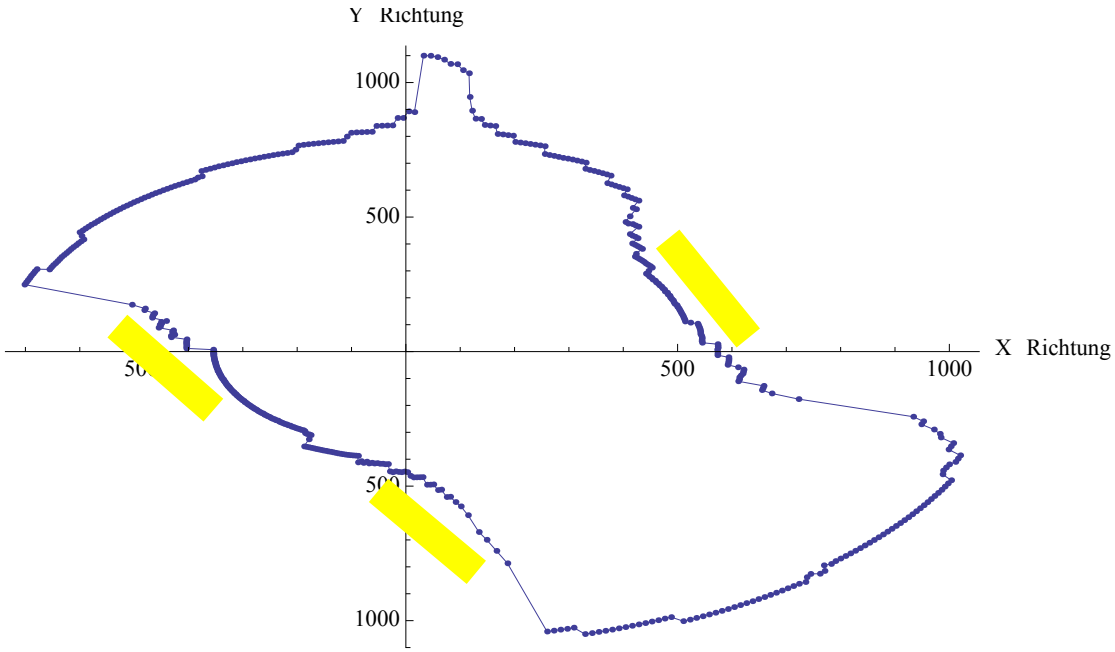

Abbildung 16: Rundplot der Aufnahme des Goniometers, Einheiten: Zeit des Schallpulses in μs, in gelb die ungefähre Position der Detektorflächen

5 Fazit

5.1 Verbesserungen

Aus den Messung geht hervor, dass eine Kalbriereinheit nach dem vorgestellten Prinzip durchaus realisierbar ist, jedoch an folgenden Punkten verbessert werden sollte.

- Die Sensoren: Der getestete Ultraschall-Sensor liefert zwar grobe Umrisse, verfügt aber über keine ausreichende Winkelauflösung. Das heißt, eine feinere Übersetzung des Getriebes würde es ermöglichen mehr Messpunkte pro Raumwinkel aufzunehmen. Über die Winkelauflösung des Infrarot-Sensor kann hier keine Aussage getroffen werden, da keine Messdaten vorliegen. Eine weitere Möglichkeit bestünde in der Verwendung eines Laser zur Entfernungsmessung, welcher mit großer Wahrscheinlichkeit über eine bessere Winkelauflösung als die IR- oder Ultraschall-Sonde verfügt, da ihm keine kegelförmige Strahlcharakteristik zugrunde liegt.
- Übersetzung: Bei der genutzten Übersetzung war es möglich, Winkelschritte von etwa 0,67° anzusteuern. Auch hier kann auf die Winkelauflösung Einfluss genommen werden, indem mehr Schritte pro Raumwinkel angesteuert werden können.

Es wurde deutlich, dass ein Projekt, welches augenscheinlich schnell abgehandelt werden kann, durchaus weit mehr Arbeitsaufwand erfordert. Das Projekt könnte unter größerem Zeitaufwand eine funktionsfähige Kalibriereinheit hervorbringen, die eine Verbesserung gegenüber den technischen Zeichnungen des Goniometers darstellt.

6 Anhang

6.1 Bilder

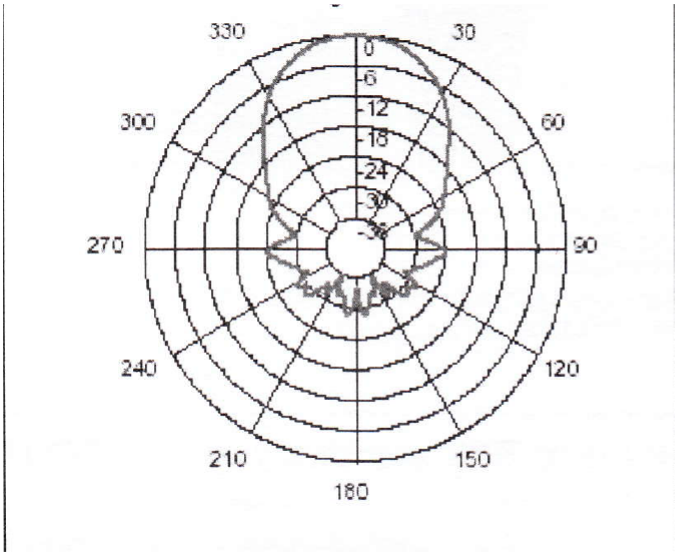

Abbildung 17: Abstrahlcharakteristk der Ultraschallsonde

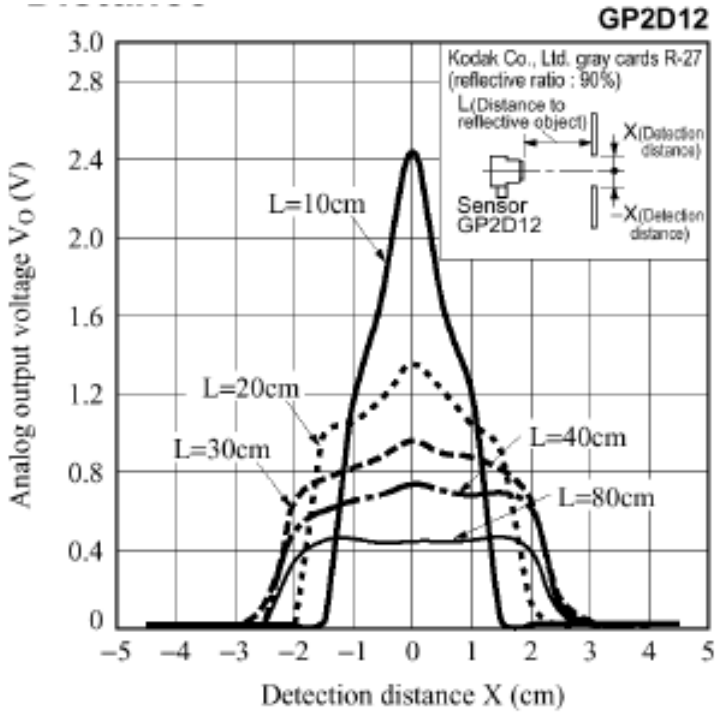

Abbildung 18: Abstrahlcharakteristk der Ultraschallsonde