Besondere Lernleistung

Entwicklung und Bau einer Computersteuerungseinheit für das Schulröntgengerät des SGR

verfasst von
Moritz Kasemann

Fachanbindung: Informatik & Physik
Betreuer: Herr Faßbender
Abgabetermin: 20.03.2018

Inhaltsverzeichnis

1 Vorwort

Ich möchte mich ganz herzlich bei meinen Eltern bedanken, die es mir ermöglicht haben dieses Projekt umzusetzen. Weiterer Dank geht an das Max-Planck-Institut für Radioastronomie, das mir einen Lötplatz zur Verfügung gestellt hat.

2 Einleitung

2.1 Thema

Bau und Programmierung einer elektronischen, mikroprozessorbasierten Steuerungseinheit für den Motor des Goniometers am Schulröntgengerät und Entwicklung einer Software, die das automatisierte Aufnehmen eines Röntgenspektrums am Schulröntgengerät ermöglicht.

2.2 Projektübersicht

Um das Schulröntgengerät vollautomatisch ansteuern zu können, muss die aktuelle Messrate ausgelesen und der Messwinkel eingestellt werden können. Um eine einfache Bedienung zu ermöglichen sollte eine Benutzeroberfläche zur Verfügung stehen. Somit lässt sich dieses Projekt in drei große Teilbereiche unterteilen.

1. Ansteuerung des Digitalzählers zum Auslesen der aktuellen Messrate
2. Ansteuerung des Motors zur Veränderung des Messwinkels
3. Programmierung einer Software, mit grafischer Benutzeroberfläche, zum Aufnehmen eines Röntgenspektrums

Der Messbereich erstreckt sich über 60°. Es wurde angestrebt eine Granularität von 0,1° Schritten zu erreichen, um nutzbare Messergebnisse erhalten zu können.

Um diese Aufgaben erfüllen zu können wurde ein Raspberry Pi 3 Model 3 verwendet. Dies ist ein Computer in einem sehr kleinen Formfaktor. Er bringt mehrere Vorteile mit sich. Zum einen besitzt er digitale Inputs/Outputs (IOs), die nötig sind um die Hardware direkt ansteuern zu können und er besitzt alle gängigen Standards heutiger Computer, zum Beispiel HDMI oder USB. Somit muss sich nicht um die Ansteuerung des Bildschirms oder das Auslesen der Tastatur gekümmert werden. Auch das Speichern einer Messreihe auf einem USB-Stick ist ohne Probleme möglich.

3 Ansteuerung des Digitalzählers

Der Digitalzähler führt eine Messung von alleine durch, er muss nur gestartet bzw. gestoppt werden und der Messwert kann dann ausgelesen werden. Es gibt zwei Möglichkeiten dies zu tun. Zum einen hat der Digitalzähler eine RS232-Schnittstelle über die der Wert ausgelesen werden kann und zum anderen gibt der Digitalzähler eine analoge Spannung aus, die sich linear zu dem Messwert verhält. Letztere Möglichkeit ist aber nicht zu empfehlen, da der Digitalzähler digital arbeitet, er muss also um eine analoge Spannung ausgeben zu können den digital gespeicherten Wert in einen analogen Wert umwandeln. Um mit diesem Wert weiterarbeiten zu können, müsste man diesen analogen Wert erst wieder in einen digitalen umwandeln. Bei diesem Vorgang muss folglich zweimal gewandelt werden und da jede Wandlung eine Ungenauigkeit mit sich bringt, werden dabei falsche Ergebnisse, entsprechend der Wandler Genauigkeit, entstehen. Außerdem besitzt der Raspberry Pi bereits eine serielle Schnittstelle über die es möglich ist die RS232-Schnittstelle anzusteuern. Damit diese auch genutzt werden kann, wird ein Pegelwandler benötigt, da der Raspberry Pi einen Spannungspegel von 0V bis 3,3V besitzt, eine RS232-Schnittstelle aber einen Pegel von ±12V besitzt. In diesem Projekt wurde der Pegelwandler ADM3202 von Analog Devices genutzt. Die notwendige Beschaltung des Pegelwandlers ist dem Datenblatt zu entnehmen. [10]

Da es immer passieren kann, dass die serielle Schnittstelle nicht funktioniert, wurde aus Sicherheitsgründen auch die mögliche Ansteuerung über die analogen Ein- und Ausgänge auf der Platine realisiert. Da auf diese nicht zurückgegriffen werden musste, wird diese hier nicht weiter beschrieben.

Um die serielle Schnittstelle letztendlich nutzen zu können wird noch der Befehlssatz des Digitalzählers benötigt. Dieser wird aber nicht von dem Hersteller angegeben und muss somit selber herausgefunden werden. Die Herstellersoftware [1] hat alle Funktionen, die für dieses Projekt von Nutzen sind. Um die Daten, die über die Schnittstelle gesendet werden, herauszufinden, wurde die kostenlose Software HTerm genutzt. [16] Diese zeigt die gesendeten und empfangen Daten an. Außerdem gibt es die Parität (keine), die Anzahl der Stopbits (eins) und die Baudrate (9600) an. In Tabelle 1 sind alle nötigen Befehle des Befehlssatzes zu sehen.

Computer an Digitalzähler	Reaktion des Digitalzählers
C	<Speicher löschen>
S1	<Messung starten>
S0	<Messung stoppen>

Tabelle 1: die wichtigsten Befehle des Digitalzählers

Um den aktuellen Messwert zu erfahren, muss kein Befehl gesendet werden, da der Digitalzähler ihn immer sendet, falls er sich ändern sollte.

Die serielle Schnittstelle muss in Python nicht programmiert werden, da dafür bereits die Library „serial“ [2] zur Verfügung steht. Diese hält alles nötige für die Schnittstelle bereit.

4 Ansteuerung des Motors

4.1 Auslesen der aktuellen Position des Motors

Der zufahrende Winkelmessbereich erstreckt sich über 60°. Dreht man an dem Goniometer, so dreht man aber sowohl den Kristall, als auch den Detektor. Dies bedeutet, dass der Motor nur 30° zurücklegen muss, um den gesamten Bereich abzudecken.

Der zur Verfügung gestellte Motor gibt eine bipolare Spannung aus, die sich linear zu dem aktuellen Winkel verhält. Der mögliche Spannungsbereich liegt bei ±3,8V, die Spannungsdifferenz beträgt folglich 7,6V. Diese Differenz entspricht ungefähr 10 Umdrehungen des Motors. Dreht man den Motor darüber hinaus, so bleibt die Spannung konstant bei -3,8V bzw. 3,8V. Der eigentlich zu fahrende Bereich von 30° entspricht demnach einer Spannungsdifferenz von ungefähr 63,3mV. Es wurde vermutet, dass diese Spannung mittels eines Potentiometers erzeugt wird. Da sich Potentiometer nicht linear verhalten, wenn an ihnen eine Last anliegt, wurde, um kein Risiko einzugehen, diese Spannung mithilfe eines Operationsverstärkers (OP) gepuffert. Die zugehörige positive Versorgungsspannung beträgt 5V und wird von dem

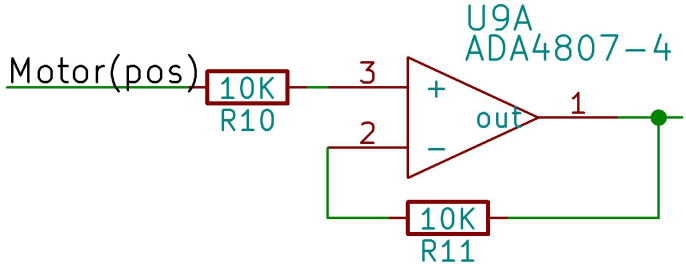

Abbildung 1: Puffer der positionsangebenden Spannung

Raspberry Pi selber bereitgestellt. Die entsprechende negative Versorgungsspannung muss erst noch erzeugt werden. Dies geschieht mithilfe des Spannungswandlers ADM660 von Analog Devices. Dieser wurde entsprechend des Datenblattes beschaltet. [5] Auch die weiteren OPs haben die gleiche Versorgungsspannung und um kein Risiko einzugehen, wird an jedem OP die Versorgungsspannung mithilfe mehrerer Kondensatoren geglättet.

Um diese Spannung weiter verarbeiten zu können muss sie erst von einem A/D-Wandler in einen digitalen Wert umgewandelt werden. Die typische höchste Auflösung eines A/D-Wandlers liegt bei 16 Bit. Dabei muss man aber bedenken, dass A/D-Wandler eigentlich nicht bipolar sind. Würde man also auf die Spannung exakt 3,8V addieren und dem Wandler die Spannungsdifferenz als Referenz geben, so würde eine Einheit 7,6V / 2^{16} = 0,115mV entsprechen. Die kleinste Einheit die man anfahren können möchte entspricht aber 63,3mV / (60°/0,1°)= 0,105mV. Da dieser Wert noch kleiner ist, als die kleinste Einheit, die man auflösen kann, wird eine Art „Lupe“ benötigt. Um diese „Lupe“ umzusetzen wurde erneut ein OP genutzt. Die Beschaltung ist in Abbildung 2 zu sehen. Die beiden Widerstände R21 und R23 erzeugen eine Verstärkung von 68KΩ / 2,2KΩ = 30,91. Das Eingangssignal wird also um circa den Faktor 31 verstärkt und zusätzlich umgepolt. Um den A/D-Wandler vor negativen Spannungen zu schützen, wurden mithilfe der Diode D2 alle negativen Spannungen verhindert. Der A/D-Wandler AD7689 von Analog Devices wurde genutzt. [9] Er besitzt acht Kanäle und als mögliche interne Referenzen 2,5V oder 4,096V. Jegliche Spannungen über 3,5V werden über die (blaue) LED D1 abgeleitet, somit wird verhindert, dass der Kanal des Wandlers beschädigt werden kann. Nutzt man nun die interne Referenz von 2,5V, so erhält man als kleinste mögliche Einheit 2,5V / 2^{16} = 0,0381mV. Die kleinste Einheit, die angefahren werden soll entspricht nun (63,3mV*30,91) / (60°/0,1°) = 3,27mV. Die Position des Motors kann nun genau genug ausgelesen werden, da man nicht auf die kleinstwertigen Bits angewiesen ist. Zwischen den einzelnen anzufahrenden Punkten liegen nun 3,27mV / 0,0381mV = 85,73 Einheiten.

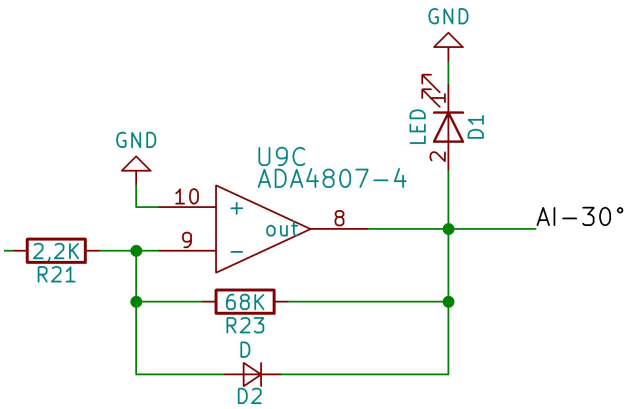

Abbildung 2: Lupen OP

Es kann aber nicht garantiert werden, dass die Position des Motors, zum Start des Programms, in dem Bereich der „Lupe“ liegt. Sollte dies nicht der Fall sein, so wird der Kanal des Wandlers zwar nicht beschädigt, aber es ist auch nicht möglich herauszufinden an welcher Position sich der Motor nun befindet. Um dieses Problem vorzubeugen wird noch ein weiterer Kanal des Wandlers genutzt. Dieser soll den gesamten Bereich abdecken, sodass zuerst die ungefähre Position des Motors ausgelesen werden kann und dann der Motor in den Bereich der „Lupe“ gefahren werden kann. In Abbildung 3 ist nun die Beschaltung eines OPs zu sehen, der aus der bipolaren Spannung eine unipolare machen soll und gleichzeitig die Spannung so vorbereiten soll, dass der Wandler diese mit der internen Referenz auslesen kann. Der Widerstand R13 und der Kondensator C29 bilden dabei, aus reiner Sicherheitsmaßnahme, einen Tiefpassfilter, um gegebenenfalls die Spannung zu glätten. Die Widerstände R13, R14 und R16 erzeugen einen Verstärkung, von 18KΩ / (9,1KΩ + 47KΩ) = 0,3208. Folglich wird damit der mögliche Spannungsbereich auf ±1,2193V reduziert. Um diese bipolare Spannung in eine unipolare umzuwandeln, wird mit den beiden Widerständen R15 und R16 die Hälfte von 2,5V addiert. Da die Spannung von 2,5V nicht von dem Raspberry Pi zur Verfügung gestellt wird, wird diese mithilfe des Referenzgenerators ADR4525 von Analog Devices erzeugt. [4] Dieser ist entsprechend des Datenblattes beschaltet, wobei die Versorgungsspannung und die Referenzspannung mit mehreren Kondensatoren geglättet wird. Der Kondensator C31 aus Abbildung 3 glättet die Ausgangsspannung noch einmal. Somit liegt nun der mögliche Spannungsbereich bei -0,0307V bis -2,4693V. Der Bereich liegt nun im negativen, da ein OP, der als Addierer beschaltet ist, immer auch invertierend ist. Deswegen folgt auf den OP aus der Abbildung 3 noch ein weiterer OP, der die Spannung wieder invertiert, sodass sie nun positiv ist. Nun ist es möglich mit einem weiteren Kanal des A/D-Wandlers den gesamten Bereich abzudecken und falls nötig in den Bereich der „Lupe“ zu fahren.

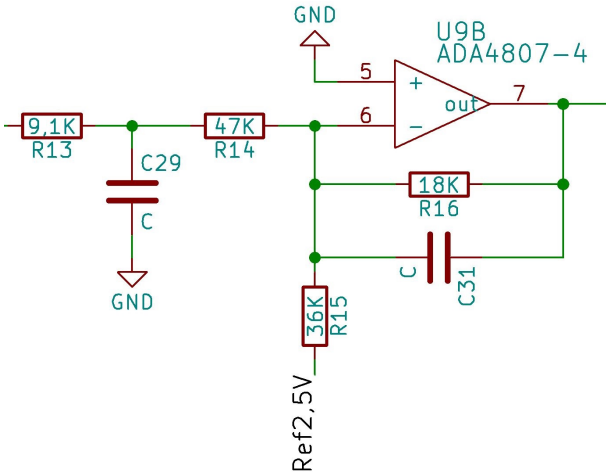

Abbildung 3: Umwandlung der bipolaren Spannung in eine unipolare

4.2 Drehen des Motors

Der Motor besitzt einen analogen Eingang, über den sich die Geschwindigkeit und die Fahrtrichtung des Motors bestimmen lässt. Die Spannung die dort angelegt werden muss, darf nur in dem Spannungsbereich von ±8V liegen. Das Vorzeichen der Spannung entscheidet über die Fahrtrichtung und die Größe der Spannung entscheidet über die Geschwindigkeit. Da der Raspberry Pi keine analoge Spannung erzeugen kann, muss ein digitales Potentiometer verwendet werden. (Abbildung 4) In diesem Projekt wurde das digitale Potentiometer AD5292 von Analog Devices genutzt. [7] Dieses wird über ein SPI-Interface angesteuert und besitzt eine Auflösung von 10 Bit. Als Bezugspunkt für das Potentiometer wurden 2,5V gewählt. Dies bedeutet, dass an dem Ausgang (Pin 4) 2,5V anliegen, wenn man den maximalen Wert ($2^{10} - 1 = 1023$) einspeichert. Da sich auch dieses Potentiometer bei starker Last nicht linear verhält, folgt ein entsprechender Puffer, wie er auch in Kapitel 4.1 verwendet wurde. Der mögliche Spannungsbereich, der ausgegeben werden kann, ist aber unipolar und muss somit erst noch in einen bipolaren umgewandelt werden. Mithilfe der Schaltung aus Abbildung 5 wurde dies umgesetzt. Dabei invertiert der erste OP die Spannung und der zweite addiert, durch die beiden Widerstände R8 und R9, die Hälfte von 2,5V. Der Spannungsbereich von 0V bis 2,5V wird also zuerst invertiert, folglich liegt er dann bei -2,5V bis 0V, und danach um 1,25V erhöht. Somit liegt er dann bei ±1,25V. Mit diesem Spannungsbereich ist es nun bereits möglich den Motor anzusteuern. Da mit dem Motor aber sehr kleine Schritte zu fahren sind (0,05°), ist es unerlässlich die Steuerspannung für nur sehr kleine Zeiten anlegen zu können. Bei der beschriebenen

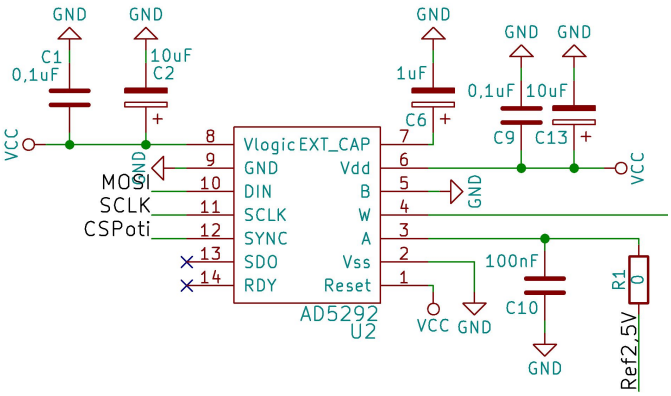

Abbildung 4: Beschaltung des digitalen Potentiometers

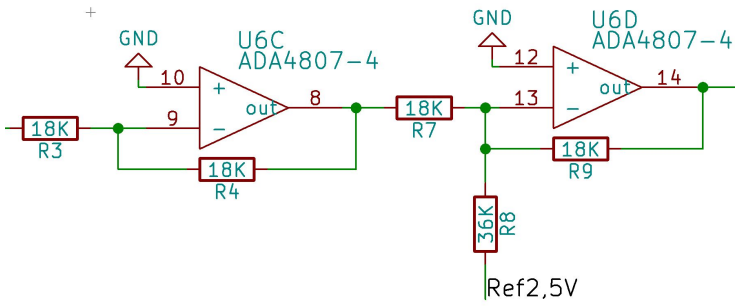

Abbildung 5: Schaltung um die unipolare Schaltung in eine bipolare umzuwandeln

Schaltung kann die Steuerspannung nur geändert werden, indem in das digitale Potentiometer ein neuer Wert eingespeichert wird. Der Vorgang, einen neuen Wert einzuspeichern, dauert aber verhältnismäßig lange. Um nicht auf diesen Vorgang angewiesen zu sein wurde der digitale Schalter ADG1636 von Analog Devices genutzt. (Abbildung 6) [8] Legt man an Pin 1 des Schalters eine logische Null an, so liegt an dem Ausgang die Spannung von Pin 4 an. Legt man eine logische Eins an, so liegt an dem Ausgang die Spannung von Pin 2 an, also 0V. Bei dem Hochfahren des Raspberry Pis ist der Status der einzelnen IOs nicht definiert, somit können sowohl eine Null als auch eine Eins an allen IOs anliegen. Es könnte deshalb theoretisch passieren, dass der Motor in die falsche Richtung losläuft und er dabei das Goniometer des Röntgengerätes beschädigen kann. Pin 14 des Schalters ist die Enable-Leitung. Liegt dort eine logische Null an, so kommen an allen Ausgängen immer 0V heraus. Es muss dort also zwingend eine Eins anliegen, um den Motor fahren zu können. Das erwähnte Problem wird mithilfe des Widerstandes R25 und dem Kondensator C33 vorgebeugt. Es liegt an Pin 14 erst eine Eins an, wenn der Kondensator voll aufgeladen ist. Dabei wurden Widerstand und Kondensator so gewählt, dass das Aufladen des Kondensators so lange benötigt, dass die Undefinierbarkeit der IOs nicht mehr vorhanden ist. Der IC ADG1636 beherbergt zwei digitale Schalter, wovon einer nicht genutzt wird. Als Sicherheitsmaßnahme sind alle Leitungen, des nicht genutzten Schalters, auf Masse gelegt, damit diese auch auf keinen Fall den anderen Schalter beeinflussen.

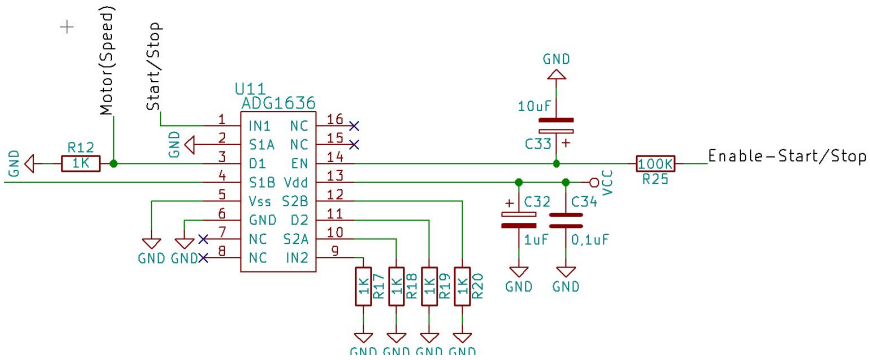

Abbildung 6: Beschaltung eines digitalen Schalters (switch)

Durch diese Schaltung ist es nun möglich eine bipolare Spannung zu erzeugen, die den Spannungsbereich von ±1,25V hat. Zudem kann diese sehr schnell angelegt und wieder entfernt werden. Sollte dieser Spannungsbereich zu klein sein, könnte man noch, dank des Jumpers R1 aus Abbildung 5, manuell den Bezugspunkt des Potentiometers von 2,5V auf 5V erhöhen. Zudem müssten die Widerstände R8 und R9 aus Abbildung 5 so geändert werden, dass sie gleich groß sind. Dadurch würde sich der Spannungsbereich

auf ±2,5V erhöhen.

5 Motorproblematik und deren Behebung

Die Strecke, die der Motor zurücklegt, lässt sich über die Geschwindigkeit und die Einschaltdauer regulieren. Reduziert man beide Variablen auf ein Minimum, sodass der Motor gerade noch los fährt, dann legt der Motor mindestens eine Strecke von circa 1° zurück. Die kleinste Wegstrecke die zurückgelegt werden soll liegt aber bei 0,05°. Dieses Problem wurde erst bei der Inbetriebnahme entdeckt, weil es nicht von Hand möglich ist eine Einschaltdauer, von dem Bruchteil einer Sekunde, zu simulieren. Um dieses Problem zu beheben gibt es nur die Möglichkeit, den Motor zu ersetzen. Ein anderer Motor mit Getriebe stand zur Verfügung. Das Getriebe besitzt eine Übersetzung von 180 zu 1 und ist zudem absolut spielfrei. Außerdem besitzt es einen Referenzschalter, der immer dann heruntergedrückt ist, wenn der Motor auf null Grad steht. Ausgelesen wird der Referenzschalter mit der Hilfe eines digitalen IOs. Der Motor ist ein fünfphasiger Schrittmotor. Da es so gut wie keine Treiber für solche Motoren gibt, wurde das Getriebe auf einen zweiphasigen Schrittmotor umgerüstet. Dies war dank einheitlicher Maße gut möglich. Damit dieser Motor mit dem Getriebe auch auf die zu drehende Achse aufgesteckt werden kann, wurde eine entsprechende Kupplung aus Aluminium, auf einer Drehbank, gedreht. Für eine längere Haltbarkeit wurden alle Einzelteile eloxiert. Um den Motor tatsächlich aufstecken zu können, wird eine Halterung benötigt. Diese wurde mittels eines 3D-Druckers gefertigt. (9.1 Abbildung 7)

Für die Ansteuerung wurde das Shield „DC & Stepper Motor HAT“ von Adafruit gewählt. [14] Der Motor und das Shield benötigen nur eine Versorgungsspannung von 5V, somit kann beides direkt von dem Raspberry Pi mit Strom versorgt werden. Der Schrittmotor macht 200 Schritte pro Umdrehung, folglich entsprechen 100 Schritte des Motors einem Grad des Goniometers und 5 Schritte der kleinst gewollten Einheit von 0,05°. Somit wäre es möglich nicht nur 20 Punkte pro Grad anfahren zu können, sondern sogar 100 Punkte. Die Library „Adafruit_MotorHAT“, wird von Adafruit kostenlos zur Verfügung gestellt und verfügt über alle nötigen Befehle um den Motor ansteuern zu können. [12]

Bei dem ersten Testlauf an dem realen Versuchsaufbau stellte sich heraus, dass das Goniometer auch ein Getriebe besitzt und somit nicht, wie am Anfang angegeben, nur 30° gefahren werden müssen. Da die Übersetzung des Getriebes nicht bekannt war,

musste das Verhältnis erst herausgefunden werden. Auf dieses Verhältnis wurde geschlossen, indem die Schritte des Schrittmotors gezählt wurden, die er benötigt um den gesamten Messbereich abzufahren. Es ergab sich, dass 60° auf der Skala ungefähr 992,39° auf der Achse des Motors entsprechen. Somit ergibt sich circa ein Verhältnis von 1 zu 16,54. Dieses Verhältnis wurde in der Software entsprechend berücksichtigt.

6 Programmierung der Steuerungseinheit

6.1 Betriebssystem und Programmiersprache

Als Betriebssystem, für den Raspberry Pi, wurde die Linux Distribution Debian Jessie gewählt, da diese, obwohl sie sehr klein gehalten wurde, alles Nötige für den normalen Betrieb mitbringt. Für die Entwicklung der nötigen Software wurde die Programmiersprache Python gewählt, weil sie über einen sehr großen Umfang an Librarys, die zum Beispiel die nötigen Befehle für das Ansteuern der digitalen IOs bereitstellen, verfügt. Für das angeführte Beispiel wurde die Library „RPi.GPIO“ gewählt. Für die Bereitstellung einer grafischen Benutzeroberfläche wurde die Library „Tkinter“ gewählt, da diese alle benötigten Komponenten zur Verfügung stellt.

6.2 Programmsetup

Bevor das Hauptbedienungsfenster generiert wird, muss die Hardware vorbereitet werden. Zu aller erst wird verhindert, dass Hardwareschäden durch das Abstürzen des Programms entstehen können. Eine mögliche Fehlersituation wäre zum Beispiel, dass das Programm während einer Motorfahrt abstürzt. Da nicht bekannt ist, wie Adafruit die Motoransteuerung tatsächlich realisiert hat, muss auch von dem Fall ausgegangen werden, dass der Motor dann beliebig weiterfährt. Dieses mögliche Problem wird durch die Library „atexit“ behoben. Diese Library ermöglicht das Ausführen von weiteren Befehlen nach dem Abstürzen des Programms. In diesem Fall wird ihr die Methode „turnOffMotors“ übergeben, die alle angeschlossenen Motoren stoppt und ihnen die Stromzufuhr verwehrt. Nach dieser Sicherheitsvorkehrung, wird die Verbindung zu dem Digitalzähler aufgebaut. Dies geschieht, indem zuerst der Port des Gerätes freigegeben wird und danach der Library „serial“ [2] alle nötigen Informationen übergeben werden, zum Beispiel der Port oder die Baudrate. Die Portfreigabe müsste eigentlich durch den Nutzer über ein Terminal erfolgen, aber die Library „os“ stellt den direkten Zugriff auf das System zur Verfügung. Somit kann das Programm sich selber die Portfreigabe

geben. Um nun noch den Motor vorzubereiten, wird erst einmal überprüft, ob der Motor gerade auf dem Referenzschalter steht. Sollte dies der Fall sein, so wird der Nutzer gefragt, ob das Goniometer des Röntgengerätes auf zwei Grad (Theta) steht, da erst ab diesem Punkt gemessen werden darf. Sollte dies der Nutzer bestätigen, so ist keine weitere Vorarbeit von Nöten. Sollte das Goniometer aber nicht auf zwei Grad stehen, so wird der Nutzer aufgefordert den Motor von dem Goniometer zu entfernen. Sobald der Nutzer dies getan hat, wird er aufgefordert das Goniometer so genau wie möglich auf zwei Grad zu drehen und danach den Motor wieder anzustecken. Darauffolgend hat der Nutzer noch einmal die Möglichkeit den genauen Winkel dem Programm zu übergeben, da es nicht gewährleistet ist, dass das Aufstecken des Motors ohne verstellen des Goniometers möglich ist. Tritt aber der andere Fall ein, nämlich dass der Motor nicht auf dem Referenzschalter steht, so wird der Nutzer ebenfalls aufgefordert den Motor von dem Goniometer zu entfernen. Auch danach wird der Nutzer aufgefordert das Goniometer so genau wie möglich auf zwei Grad zu drehen. Währenddessen fährt der Motor immer weiter in eine Richtung, bis er den Referenzschalter gefunden hat. Sobald beides erledigt ist, muss der Nutzer den Motor wieder auf das Goniometer aufstecken. Auch danach hat er die Möglichkeit den genauen Winkel einzugeben.

6.3 Hauptteil des Programms

Nachdem die Hardware vorbereitet wurde, wird das Hauptbedienungsfenster geöffnet. (9.1 Abbildung 10) Dieses lässt sich in zwei Bereiche unterteilen. Der eine Bereich bietet, in Form von Knöpfen, alles nötige um das Röntgengerät zu bedienen. Der zweite Bereich ist rein passiv, er reagiert nur auf die Eingabe des Nutzers. Er besteht aus einer großen Leinwand, an der der zugehörige Graph gezeichnet wird, aus zwei Ladebalken die den Fortschritt der aktuellen Messung angeben und aus einem Ausgabebereich, in dem zum Beispiel Fehlermeldungen ausgegeben werden können.

Im ersten Bereich werden dem Nutzer folgende Möglichkeiten gegeben:

- eine neue Messreihe starten
- eine Messreihe speichern
- eine Messreihe laden
- den aktuellen Graphen durchzugehen und sich die Messwerte genau anzuschauen

Möchte der Nutzer eine neue Messreihe starten so öffnet sich ein neues Fenster, in dem

zuerst alle Parameter der Messreihe eingegeben werden müssen. (9.1 Abbildung 8) Dazu gehört der Name der Messreihe, der Kristall der genutzt wird, die Messdauer pro Messung, in welchem Winkelbereich gemessen werden soll und die Granularität der Messreihe. Bestätigt der Nutzer nun seine Eingabe so müssen alle Parameter geprüft werden, damit es nicht zu Abstürzen oder Beschädigungen kommen kann. Dabei wird zum Beispiel geprüft, ob alles ausgefüllt ist, ob der Name keine Sonderzeichen enthält, dass die Granularität nicht zu klein ist, usw.. Sollten alle Eingaben korrekt sein so wird das Eingabefenster geschlossen und die Messreihe wird in einem neuen Thread gestartet. Dies bedeutet, dass die Messreihe parallel zu dem Hauptfenster bearbeitet wird, es entstehen also zwei verschiedene Ablaufstränge. Der Sinn dahinter ist, dass das Hauptfenster immer noch ansprechbar sein soll, damit der Graph auch aktualisiert werden kann. Dies wird durch die Library „thread“ umgesetzt. Die anzufahrenden Winkel der Messreihe werden dann, in diesem neuen Thread, in einer Schleife abgearbeitet. Zuerst wird der nächste Winkel angefahren, danach wird die Messung gestartet und nach der angegebenen Zeit gestoppt. Der Messwert wird dann, mit dem Messwinkel, in einer zweidimensionalen Liste abgespeichert. Danach wird ein neuer Thread gestartet, der den Graphen aktualisieren soll. Dies wurde so realisiert, um die Zeit zwischen den einzelnen Messungen möglichst gering zu halten. Wurden alle Messungen durchgeführt, wird der Motor wieder auf den Referenzschalter zurückgefahren.

Möchte der Nutzer die aktuelle Messung speichern, so wird diese unter dem angegebenen Namen auf dem Desktop abgespeichert. Als Dateiformat wurde dabei csv gewählt, da dies ein genormtes Format ist, das von allen gängigen Programmen, zum Beispiel Excel, geöffnet werden kann. Für die Umwandlung der Liste mit allen Messwerten in das csv-Format wird die Library „csv“ verwendet.

Für das Laden einer Messreihe wird ein neues Fenster geöffnet, in dem der Pfad zu der Datei eingegeben werden muss. (9.1 Abbildung 9) Sollte der Pfad nicht existieren, wird eine Fehlermeldung ausgegeben und der Nutzer wird aufgefordert den Pfad zu korrigieren. Für die Umwandlung der Datei in eine Liste wurde ebenfalls die Library „csv“ verwendet.

Möchte der Nutzer sich den Graphen genauer anschauen, dann kann er mit zwei Knöpfen den Graphen durchgehen. Dabei wird der Messpunkt, der angeschaut wird, mit einem Kursor gekennzeichnet und die Daten der Messung werden genau angegeben.

Der Graph wird auf einer, von „Tkinter“ zur Verfügung gestellten, Leinwand gezeichnet. Da die Messzeit bei jeder Messreihe frei gewählt werden kann, wurde keine starre Y-Achse gewählt. Das bedeutet, dass die Skala sich variabel an den größten Messpunkt anpasst. Damit ist sichergestellt, dass der ganze Graph angezeigt werden kann und bei kleinen Messzeiten, der Graph immer noch gut erkennbar bleibt.

7 Platine

Um alle benötigten Bauteile aus den Kapiteln 3, 4.1 und 4.2 unterzubringen musste eine Platine konstruiert werden. Die Maße der Platine wurden entsprechend den offiziellen HAT-Maßen gewählt. [11] Dies sind die Maße, die für alle offiziellen Erweiterungsplatinen für den Raspberry Pi gelten. Für die Umsetzung wurde das frei erhältliche Programm KiCad verwendet. Zuerst wurde die gesamte Schaltung in KiCad übertragen. Danach musste für jedes Bauteil ein entsprechender Footprint hinterlegt werden. Da die Platine mit der Länge von 65mm und der Höhe von 56,5mm sehr klein ist, um alle Bauteile auf ihr zu platzieren, wurde für jedes Bauteil die SMD Baugröße gewählt. Dann wurde für jedes Bauteil die genaue Position auf der Platine festgelegt und alle Leiterbahnen verlegt. Dabei wurde darauf geachtet, dass zusammengehörige Bauteile möglichst nah aneinander platziert sind und dass alle Masseleitungen möglichst dick sind. Mithilfe von KiCad wurden dann Gerberdaten von dieser Platine erzeugt, die an den Produzenten der Platine weitergeleitet wurden. Die fertige Platine wurde dann entsprechend der Schaltung bestückt. (Der zugehörige Schaltplan ist der digitalen Version beigefügt)

8 Fazit

Das Ziel, mit einer Auflösung von 0,1° ein Röntgenspektrum aufnehmen zu können, wurde nun, nachdem die Motorproblematik behoben wurde, erreicht. Die Hardware, des Röntgengerätes, wird optimal ausgenutzt und die Software bietet dem Nutzer eine sehr komfortable Bedienung.

Durch dieses Projekt habe ich sehr viel gelernt. Aus technischer Sicht habe ich gelernt, wie man einen Schaltplan erstellt und aus diesem dann eine Platine entwickelt. Zudem konnte ich auch einen Einblick in die Nutzung einer Drehbank erhalten. Von der technischen Seite abgesehen, habe ich aber auch gelernt, wie man ein solches Projekt umsetzt. Dass man zum Beispiel immer Zeit einplanen sollte um Fehler suchen zu

können, sowohl in der Software, als auch in der Hardware. Bezüglich meiner Studienwahl hat mir dieses Projekt gezeigt, dass mein Plan, Elektrotechnik zu studieren, für mich genau das richtige ist.

9 Anhang

9.1 Bilder des Projektes

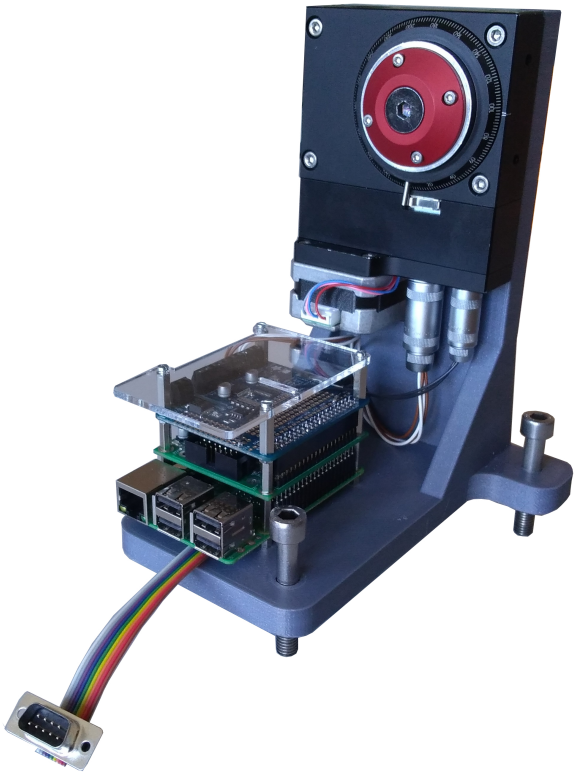

Abbildung 7: Computersteuerungseinheit (komplett)

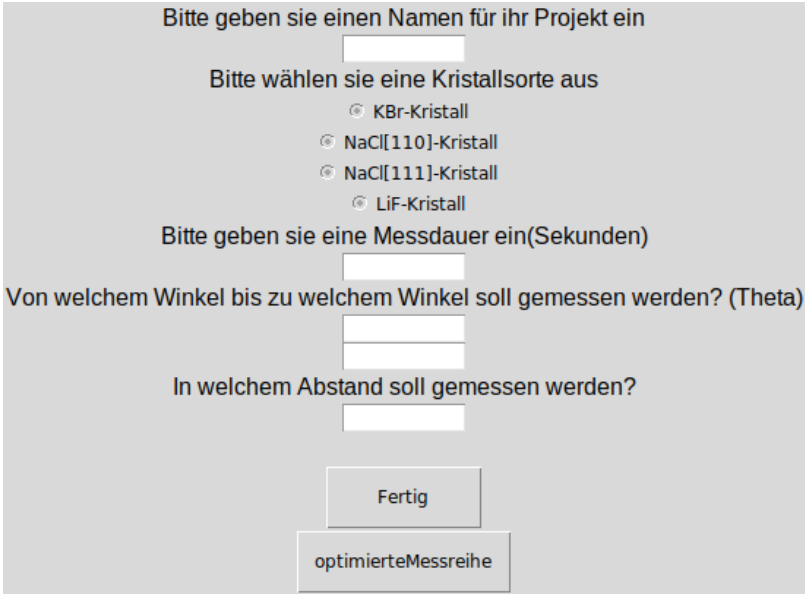

Abbildung 8: Parametereingabe für eine neue Messreihe

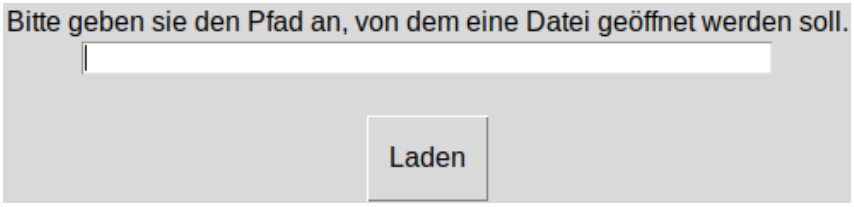

Abbildung 9: eine Messreihe laden

Abbildung 10: Benutzeroberfläche

9.2 Hilfsmittel

Es wurden folgende Hilfsmittel genutzt:

- entsprechende Werkzeuge zum löten
- Oszilloskop
- Voltmeter
- Drehbank und die zugehörigen Werkzeuge
- Serial Port Monitor [15]
- HTerm [16]
- KiCad [17]

10 Literaturverzeichnis

1. Leybold, Digitalzählersoftware: https://www.leybold-shop.de/physik/physik-geraete/elektrik-elektronik/messgeraete/digitalzaehler/digitalzaehler-57548.html – 19.01.2018

2. Python Library „serial“: https://github.com/pyserial/pyserial – 19.01.2018

3. Reichelt 9polig sub-d Stecker weiblich, Datenblatt: https://www.reichelt.de/SUB-D-gewinkelt/D-SUB-BU-09US/3/index.html?ACTION=3&GROUPID=7418&ARTICLE=6954 – 19.01.2018

4. Referenzgenerator ADR4525, Datenblatt: http://www.analog.com/en/products/linear-products/voltage-references/adr4525.html – 19.01.2018

5. Spannungswandler ADM660, Datenblatt: http://www.analog.com/en/products/power-management/switching-power-converters/switched-capacitor-converters/adm660.html – 19.01.2018

6. Operationsverstärker ADA4807-4, Datenblatt: http://www.analog.com/en/products/amplifiers/operational-amplifiers/rail-to-rail-amplifiers/ada4807-4.html – 19.01.2018

7. Digitalpotentiometer – Datenblatt Download: http://www.analog.com/en/products/digital-to-analog-converters/digital-

potentiometers/ad5292.html#product-overview – 19.01.2018

8. digitaler Schalter ADG1636 – Datenblatt Download: http://www.analog.com/en/products/switches-multiplexers/analog-switches-multiplexers/adg1636.html – 19.01.2018

9. A/D-Wandler AD7689 – Datenblatt Download: http://www.analog.com/en/products/analog-to-digital-converters/ad7689.html – 19.01.2018

10. Pegelwandler ADM3202 – Datenblatt Download: http://www.analog.com/en/products/interface-isolation/rs-232-rs-422-rs-485/interface-rs-232/adm3202.html – 19.01.2018

11. Raspberry Pi 3 HAT Maße – Download: https://github.com/raspberrypi/hats – 05.10.2017

12. Adafruit Motor HAT - Library Download: https://github.com/adafruit/Adafruit-Motor-HAT-Python-Library – 14.03.2018

13. Adafruit Motor HAT - Schematics Download: https://learn.adafruit.com/adafruit-dc-and-stepper-motor-hat-for-raspberry-pi/downloads – 14.03.2018

14. Adafruit Motor HAT – Überblick: https://learn.adafruit.com/adafruit-dc-and-stepper-motor-hat-for-raspberry-pi – 14.03.2018

15. Serial Port Monitor – Download: https://www.eltima.com/de/products/rs232-data-logger/ - 12.7.2017

16. HTerm – Download: https://www.heise.de/download/product/hterm-53283 – 12.7.2017

17. KiCad – Download: http://kicad-pcb.org/download/ - 17.03.2018

11 Erklärung der eigenständigen Verfassung

Ich versichere, dass ich die vorliegende Arbeit einschließlich evtl. beigefügter Zeichnungen, Kartenskizzen, Darstellungen u. ä. m. selbstständig angefertigt und keine anderen als die angegebenen Hilfsmittel benutzt habe. Alle Stellen, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen sind, habe ich in jedem Fall unter genauer Angabe der Quelle deutlich als Entlehnung kenntlich gemacht.

_____, den _____
(Ort) (Datum)

_____