Laserintensitätsstabilisierung mit dem Red Pitaya

von

Jonas Lehnen

Bachelorarbeit in Physik
vorgelegt dem Fachbereich Physik, Mathematik und Informatik (FB 08)
der Johannes Gutenberg-Universität Mainz
am 19. Februar 2019

1. Gutachter: Prof. Dr. Patrick Windpassinger
2. Gutachter: Prof. Dr. Jochen Walz

Ich versichere, dass ich die Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Mainz, den

Jonas Lehnen
QPQI, QUANTUM
Institut für Physik
Staudingerweg 7
Johannes Gutenberg-Universität D-55099 Mainz
jlehnen@students.uni-mainz.de

Inhaltsverzeichnis

1. Einleitung

Vor über 2000 Jahren erfand der Grieche Ktesibios einen Mechanismus, der den Wasserstand in einem Tank auf einer konstanten Höhe hält. Eine Schwimmblase öffnet dazu ein Ventil, wenn der Wasserstand unter ein bestimmtes Niveau fällt. [1]. Damit schuf er eines der ersten Systeme mit rückgekoppelter Regelung. Im großen Stil werden rückgekoppelte Regelungen aber erst seit der Erfindung der Dampfmaschine im Jahr 1769 verwendet. Ohne eine Vielzahl von rückgekoppelten Reglern, allen voran der Fliehkraftregler, wäre ein wirtschaftlicher Einsatz der Dampfmaschine nicht möglich gewesen. Das System des Fliehkraftreglers, der die Dampfzufuhr über die Drehzahl der Dampfmaschine kontrolliert, konnte durch Differentialgleichungen mathematisch beschrieben werden, was die Geburtsstunde der modernen Regelungstheorie darstellt [1].

Heute sind rückgekoppelte Regelungen nicht mehr aus technischen Anwendungen wegzudenken. Vom Thermostat über den Tempomat im Auto, bis hin zum Landen von Raketen auf schwimmenden Plattformen, überall wird rückgekoppelte Regelung verwendet. Auch aus der Experimentalphysik ist die Regelungstechnik nicht mehr wegzudenken. Die Entdeckung von Gravitationswellen, für die 2017 der Physik Nobelpreis verliehen wurde, war nur durch ausgefeilte Stabilisierung der verwendeten Lasersysteme möglich [3]. In der Quantenoptik müssen Laser auf feste Frequenzen stabilisiert werden, damit eine resonante Wechselwirkung zwischen Atomen und Licht stattfindet. Auch die Stabilisierung der Laserintensität ist für reproduzierbare Messergebnisse unabdingbar.
Wo früher noch analoge Regelsysteme verwendet wurden, kommen heute mehr und mehr digitale Regelsysteme zum Einsatz. Das liegt vor allem an der höheren Flexibilität und den gesunkenen Kosten dieser Systeme. Komplizierte Regelungsalgorithmen können relativ einfach implementiert und verändert werden, sollten sie die gewünschten Anforderungen nicht erreichen [4]. Wird eine bestehende Regelung nicht mehr benötigt, kann die digitale Hardware einfach umprogrammiert und an anderer Stelle eingesetzt werden.
In dieser Arbeit soll eine Laserintensitätsstabilisierung mit einem einem digitalen „Red Pitaya STEMlab 125-14“-Board (Red Pitaya) implementiert werden. Der Red Pitaya besitzt einen 125 MHz FPGA und genauso schnelle, analoge Ein- und Ausgänge. Damit eignet er sich perfekt für Regelungsaufgaben, da Signale im FPGA fast ohne Zeitverzögerung verarbeitet werden können.
Die Stabilisierung soll in ein bestehendes Dysprosium-Experiment [2] der Arbeitsgruppe Windpassinger eingebaut werden und Schwankungen der Teilchenzahl einer magnetooptischen Falle minimieren. Diese Arbeit soll dem Leser das nötige Wissen vermitteln, um die gleiche Laserstabilisierung aufzubauen. Als Erstes wird eine kurze

Einführung in die Regelungstheorie und ein Überblick über die experimentellen Aufbauten gegeben. In Kapitel 4 wird der Red Pitaya und die darauf verwendete Software vorgestellt. Das darauffolgende Kapitel ist als Anleitung für die Modifizierung der Red Pitaya Software zu verstehen. Die notwendigen Schritte für die Implementierung eines digitalen Triggers, der die Regelung ein- und ausschaltet, werden aus diesem Grund sehr detailliert beschrieben. In den Ergebnissen werden die Messungen zum Aufbau und Betrieb der Laserstabilisierung, sowie deren Einfluss auf die Stabilität der Teilchenzahl im Dysprosium-Experiment, vorgestellt. Im Fazit werden die gewonnenen Ergebnisse zum Aufbau der Intensitätsstabilisierung zusammengefasst. Außerdem wird ein kurzer Ausblick über weitere Verbesserungs- und Anwendungsmöglichkeiten der Stabilisierung gegeben.

2. Regelungstheorie

In diesem Kapitel soll dem Leser das zum Verständnis dieser Arbeit notwendige Wissen über Regelungstheorie vermittelt werden. Zuerst wird der Regelkreis erklärt, um ein Verständnis von Regelungsvorgängen zu etablieren. Danach wird analoge und digitale PID-Regelung vorgestellt, sowie Möglichkeiten diese zu optimieren.

2.1. Regelkreis

Der Regelkreis besteht aus einem System, das geregelt werden soll und einem Regler, der diese Regelung übernimmt. Das zu regelnde System wird als Regelstrecke bezeichnet, welche auf ein Eingangssignal $f(t)$ ein Ausgangssignal $x(t)$ ausgibt. Die Regelstrecke kann dabei als die Impulsantwort $r(t)$ auf ein Dirac-δ-Signal modelliert werden. Nimmt man an, dass die Antwort des Regelkreises linear und invariant unter Zeitverzögerung ist, so ergibt sich das $x(t)$ als

$$x(t) = \int_{\Re^n} r(\tau) f(t-\tau) d\tau = (r * f)(t), \tag{2.1}$$

also der Faltung von r und f. Durch eine Fourier-Transformation kann die Gleichung im Frequenzraum betrachtet werden und es ergibt sich unter Anwendung des Faltungstheorems

$$X(s) = R(s)F(s). \tag{2.2}$$

$R(s)$ wird dann als Übertragungsfunktion bezeichnet. Die in dieser Arbeit betrachteten Regelstrecken enthalten alle eine Photodiode, welche durch den eingebauten Operationsverstärker ein Tiefpassverhalten aufweist [4]. Der Regelkreis kann deswegen näherungsweise als Tiefpass erster Ordnung betrachtet werden, wobei w_0 die Grenzfrequenz ist. Die Übertragungsfunktion $R(s)$ ist dann

$$R(s) = \frac{R_0}{1 + s/w_0}. \tag{2.3}$$

Die Regelung wird von einem Regler C vorgenommen, welcher genau wie die Regelstrecke über die Fourier-Transformation seiner Impuls-Antwort als $C(s)$ modelliert werden kann. Das Ausgangssignal des Reglers $Y(s)$ ist definiert als

$$Y(s) = C(s)X(s) \tag{2.4}$$

und wird auch Stellgröße genannt.

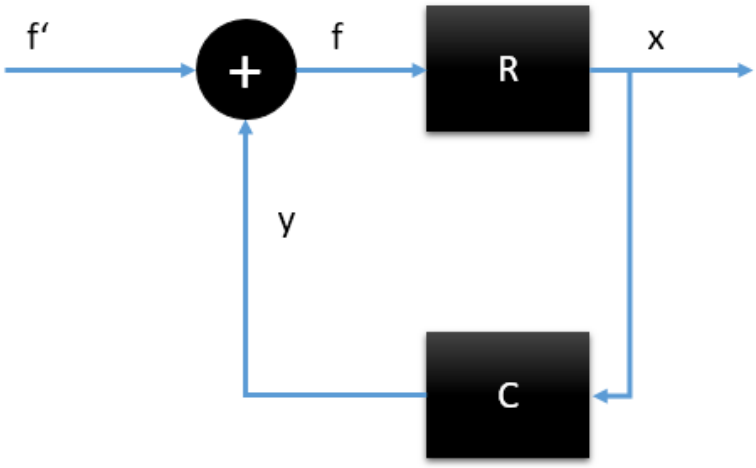

Abbildung 2.1.: Modell eines Regelkreises. f’ ist ein vom Regelkreis unabhängiges Eingangssignal z.B. rauschen. Das Eingangssignal wird mit dem Stellwert y summiert und auf den Regelkreis gegeben. Dieser gibt ein Antwortsignal aus, das zum Regler C geleitet wird, sodass eine rückgekoppelte Regelung entsteht.

Die Abbildung 2.1 zeigt einen geschlossenen Regelkreis. Ein unabhängiges Eingangssignal f' wird mit der Stellgröße y summiert und als $f = f' + y$ auf die zur Regelstrecke gegeben. Zusammen mit den Gleichungen 2.2 und 2.4 ergibt sich

$$\frac{x}{f'} = \frac{R}{1 - RC}. \tag{2.5}$$

Daraus folgt, dass für den Fall $RC \to 1$ das System instabil wird, da schon eine kleines Eingangssignal ausreicht, damit $x \to \pm\infty$ geht.
Es ist jedoch auch möglich, durch eine entsprechende Wahl von C, den Einfluss, welchen f' auf x hat, zu minimieren und damit auch externe Störungen oder Rauschen zu minimieren. Will man, dass x nicht auf Null, sondern einen konstanten Setzwert s geregelt wird, muss s vor dem Regler von x abgezogen werden, sodass das Fehlersignal $e = x - s$ auf 0 geregelt wird und $x + \delta e = s$ gilt.

2.2. PID-Regelung

Für die Intensitätsstabilisierung soll ein digitaler PID-Algorithmus verwendet werden. Zum besseren Verständnis wird zuerst die analoge PID-Regelung betrachtet. PID-Regler sind die wahrscheinlich meistverbreiteten analogen Regler. [4] Sie bestehen aus einem Proportional-, einem Integral- , und einem Differentialteil, woraus sich deren

Name ergibt. Im Zeitbereich wird der Regler dabei durch die Gleichung

$$C_{PID} = K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{de(t)}{dt} \tag{2.6}$$

beschrieben, wobei K_p,K_i und K_d Regelparameter sind, die für jeden Regelkreis passend gewählt werden müssen, damit dieser nicht instabil wird 2.1.
Ein anderer Effekt, der sich negativ auf die Regelung auswirkt, ist der „Integral-Windup“. Ist $e(t) > 0$ für eine lange Zeit, z. B. weil der Setzwert größer gewählt wurde als es der Regelkreis ermöglicht, so divergiert der Integralteil des PID. Da alle realen Systeme Grenzen für ihre minimalen und maximalen Ausgangswerte haben, ist es möglich, dass der I-Teil diesen um ein Vielfaches überrepräsentiert. Ist die Regelung auf den Setzwert zu einem späteren Zeitpunkt wieder möglich, so muss erst der I-Teil abgebaut werden, bevor eine Reglung auf den Setzwert erfolgt. Dies kann zu langen Zeitverzögerungen führen.

2.3. Optimale Regelung

Die Dimensionierung eines zu einem Regelkreis passenden Reglers C, wird in der Regelungstechnik als „Problem der optimalen Regelung“bezeichnet. Der Regelkreis hat die Aufgabe, $e(t)$ auf Null zu halten. Um die Güte der Regelung zu beschreiben, wird die skalare Größe Q eingeführt, welche die Abweichung vom gewünschten Regelverhalten misst.

$$Q = \frac{1}{T} \int_0^T e^2(t)dt \tag{2.7}$$

Eine Optimale Regelung ist erreicht, wenn Q minimal ist.

Faustformelverfahren

Eine Möglichkeit optimale PID-Parameter zu finden, stellen die Faustformelverfahren dar. Diese kommen ohne mathematisches Modell der Regelstrecke aus und verwenden stattdessen heuristische Methoden.

Zweite Einstellregel Ziegler Nichols

Die zweite Einstellregel von Ziegler und Nichols approximiert die Regelstrecke als Übertragungsglied erster Ordnung und analysiert eine Sprungantwort der Regelstrecke, um daraus PID-Parameter zu erhalten. Dazu wird die Amplitude nach langer Zeit $x(t = \infty)$ gemessen, aus der sich der stationäre Verstärkungsfaktor

$$K = \frac{x(t = \infty)}{f(y = \infty)} \tag{2.8}$$

ermitteln lässt. Außerdem wird eine Wendetangente an das Antwortsignal, wie in Abbildung 2.2 angelegt und die Verzugszeit T_u als Nullstelle dieser Tangente ermittelt, sowie die Ausgleichszeit T_g als Zeit zwischen Nullstelle und Schnittpunkt der Tangente mit $f(y = \infty)$. Die PID-Parameter lassen sich dann in Tabelle B.2 ablesen [6].

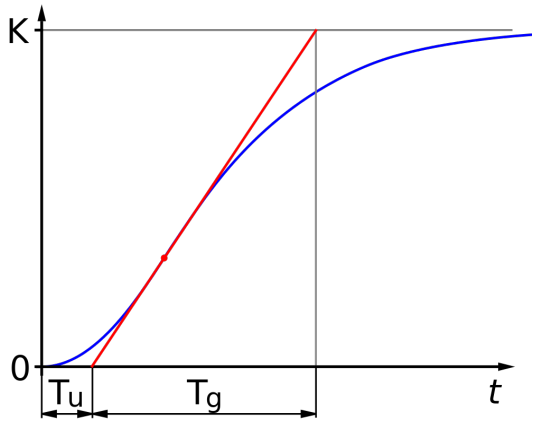

Abbildung 2.2.: Sprungantwort zur Ermittlung der PID-Parameter. Zum Zeitpunkt $t = 0$ wird eine Sprungfunktion auf den Regelkreis gegeben. Dessen Sprungantwort ist im blauen Graph dargestellt. Zur Ermittlung der Regelparameter wird eine Wendetangente $W(t)$ an die Sprungantwort angelegt. Aus der Höhe der Sprungantwort kann der Parameter K ermittelt werden. T_u und T_g ergeben sich aus den Schnittpunkten $W(t = T_u) = 0$ und $W(t = (T_g + T_u) = K)$. (Urheber: Matthias Krüger, Dresden, Quelle: `https://de.wikipedia.org/wiki/Faustformelverfahren_(Automatisierungstechnik)#/media/File:Tutg.svg`)

Minimierung der Regelabweichung

Eine weitere Möglichkeit die Regelung zu optimieren, bezieht sich direkt auf unsere Definition der optimalen Regelung. Hierbei wird der geschlossene Regelkreis mit einer Rechteckschwingung gestört und Q minimiert. Dies kann von Hand mit einem Oszilloskop und Augenmaß durchgeführt oder automatisiert werden.

2.4. Digitale PID-Regelung

Theoretisch unterscheidet sich die digitale PID-Reglung kaum von der analogen Variante. Die zur Beschreibung der P-,I-,D-Komponenten nötigen Gleichungen müssen lediglich diskretisiert werden. Die Gleichung 2.6 entspricht

$$u(t_n) = K'_p e(t_{n-1}) + K'_i \sum_{i=0}^{n-1} e(t_i) + K'_d e((_{tn-1}) - e(_{tn-2})) \tag{2.9}$$

in einer zeitdiskreten Darstellung.
Seien f_r die Regelfrequenz und Δt die Periodendauer des Regelalgorithmus, dann soll sich im Grenzfall von $\Delta t \to 0$ der analoge Regelalgorithmus wiederfinden. Deshalb

müssen K'_p, K'_i und K'_d wie folgt definiert werden:

$$\begin{aligned} K_{p_{analog}} = K_p &\Rightarrow K'_p e(t_{n-1}) \xrightarrow{\Delta t \to 0} K_p e(t) \\ K_{i_{analog}} = \frac{K'_i}{f_r} &\Rightarrow K'_i \sum_{i=0}^{n-1} e(t_i)\Delta t \xrightarrow{\Delta t \to 0} K_i \int_0^t e(t)dt \\ K_{d_{analog}} = K'_d f_r &\Rightarrow K'_d \frac{e(t) - e(t-\Delta t)}{\Delta t} \xrightarrow{\Delta t \to 0} K_d \frac{de(t)}{dt}. \end{aligned} \tag{2.10}$$

Für Regelfrequenzen die wesentlich größer sind als die des Fehlersignals e(t), verhält sich der digitale PID-Regler wie sein analoges Gegenstück, was man beim Vergleich 2.10 und 2.6 sieht.
Ein Problem, das bei der Digitalisierung auftreten kann, wird als Alias-Effekt bezeichnet. Dieser tritt auf, wenn die Frequenzen des Fehlersignals größer sind als $\frac{1}{2\Delta t}$.[4] Dabei wird ein hochfrequentes Signal als eines mit sehr viel niedrigerer Frequenz interpretiert. Um dies zu vermeiden, sollten Signale vor dem Digitalisieren durch einen Tiefpass gefiltert werden. Bei der theoretischen Betrachtung von PID-Reglern besteht keine Notwendigkeit, das Integral im analogen Fall oder die Integralsumme $K_{i_{analog}} \sum_{i=0}^{n-1} e(t_i)\Delta t$ zu beschränken. In der praktischen Anwendung sind diese jedoch immer beschränkt, da im analogen Fall ein Kondensator voll aufgeladen ist, oder im Digitalen der verfügbare Speicherplatz ausgefüllt ist. Diese Limitierung ist meist nützlich, da sie den Integral-Windup begrenzt.

3. Experimentelle Aufbauten

Das Ziel dieser Arbeit ist die Implementierung einer Laserstabilisierung in das Dysprosium-Experiment der AG Windpassinger [2]. Im Folgenden wird ein kurzer Überblick über das Experiment und die wichtigsten enthaltenen Lasersysteme gegeben.

3.1. Dysprosium-Experiment

Das Element Dysprosium ist ein silbergraues Metall mit der Ordnungszahl 66 und gehört zur Gruppe der Lanthanoide. Dysprosium hat ein magnetisches Moment von $10\mu_b$, wobei μ_b das Bohrsche Magneton bezeichnet. Die Dipol-Dipol-Wechselwirkung skaliert mit μ^2, weshalb deren Effekt bei Dysprosium etwa 100 mal stärker ist als bei Alkalimetallen mit einem magnetischen Moment von ungefähr $1\mu_b$. [8] Damit lassen sich Effekte der Dipol-Dipol-Wechselwirkung wesentlich besser von anderen Effekten unterscheiden. Die Arbeitsgruppe ist dabei[1], die Isotopieverschiebung zwischen verschiedenen Dysprosium-Isotopen zu messen. Es werden die Isotope 160Dy, 162Dy und 164Dy verwendet. Dazu wird ein Übergang von 1001 nm bei den verschiedenen Isotopen spektroskopiert und die relative Änderung der Frequenz des Übergangs gemessen. Abbildung 3.2 gibt einen Überblick über den Vakuumaufbau des Experiments. Dysprosium wird in einem Ofen auf 1250 °C geheizt, um ein Dysprosium-Gas zu erhalten. Aufgrund der hohen kinetischen Energie des Gases entsteht ein Teilchenstrahl, welcher sich nur entlang eines Rohrs in eine Vakuumkammer bewegen kann. Der Teilchenstrahl wird dabei von einem 421 nm-Lasersystem in transversaler Richtung gekühlt. Das selbe Lasersystem betreibt einen Zeemanslower, der den Teilchenstrahl in longitudinaler Richtung kühlt, bevor dieser in die Vakuumkammer gelangt. Sowohl beim transversalen Kühlen, als auch beim Zeemanslower wird der optische Dopplereffekt verwendet, um die Atome zu kühlen.
Ist der Laser rotverstimmt gegenüber eines Dysprosiumübergangs, wechselwirken nur noch Dysprosium-Atome mit dem Laser, welche einen Geschwindigkeitsanteil in dem Laserlicht entgegengesetzter Richtung haben, da sie dieses aufgrund des optischen Dopplereffekts mit einer höheren Frequenz als ein Beobachter im Laborsystem wahrnehmen. Atome, die sich in transversaler Richtung aus dem Strahl herausbewegen, werden dann vom Laserlicht resonant angeregt. Dabei findet ein Impulsübertrag von Photon auf Atom statt, wodurch das Atom zurück in den Teilchenstrahl schickt wird. Im Zeemanslower werden Teilchen in longitudinaler Richtung gekühlt. Um Atome nicht nur bei einer mit dem Laser resonanten Dopplergeschwindigkeit, sondern über

[1] (Stand Februar 2019)

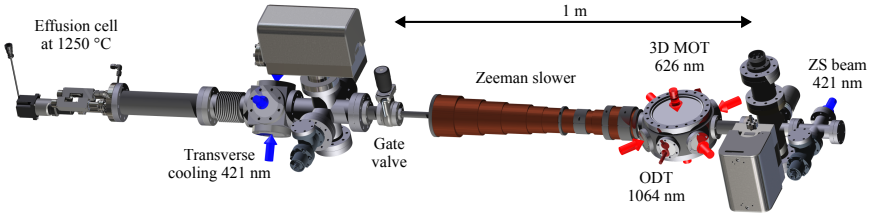

Abbildung 3.1.:

Abbildung 3.2.: Vakuumsystem des Dysprosium-Experiments. Effusion cell: Ofen in dem das Dysprosium erhitzt wird. Transverse cooling: Transversales Kühlsystem mit einer Wellenlänge von 421 nm. 3D MOT: Magneto-optische Falle zum einfangen der Dysprosium-Atome. ODT: Optische Dipol-Falle [8].

ein breites Band an Geschwindigkeiten zu kühlen, wird der Zeemaneffekt genutzt. Ein Magnetfeld mit positivem Gradienten in Richtung der Vakuumkammer verschiebt die Energieniveaus der Atome kontinuierlich, sodass diese den gesamten Weg durch den Zeemanslower hindurch resonant zum Laser sind und deshalb von diesem gekühlt werden.
Die so gekühlten Atome können anschließend von einer magnetooptischen Falle (MOT) oder einer Dipolfalle [8] eingefangen und untersucht werden. Auch hier wird der optische Dopplereffekt, in Verbindung mit einem ortsabhängigen Magnetfeld, verwendet, um die Atome mit einem 626 nm-Lasersystem zu fangen und zu kühlen. [2]
Die Laborerfahrung hat gezeigt, dass die Anzahl der gefangenen Atome in der MOT stark von der Leistung des Lichts im Zeemanslower abhängt. Um die Stabilität des Experiments zu verbessern und Schwankungen in der Teilchenzahl zu vermeiden, soll deswegen eine Laserintensitätsstabilisierung in das Lasersystem des Zeemanslowers eingebaut werden.

3.2. Regelkreise

Der für die Laserstabilisierung betrachtete Regelkreis des Dysprosium-Experiments beschränkt sich auf einen Abschnitt kurz vor dem Zeemanslower. Dieser besteht aus dem Strahlengang des Lasers, der für den Zeemanslower verwendet wird, und den elektrischen Bauteilen, die für die Regelung verwendet werden. Ein Foto ist in Abbildung C.1 zu sehen. Der Laserstrahl des Zeemanslowers wird im Rahmen dieser Arbeit auch nur solange betrachtet, wie er sich im Regelkreis befindet, da weder davor, noch danach, die Möglichkeit besteht, diesen durch eine Regelung zu kontrollieren.
Um Vorarbeiten für die Stabilisierung auch abseits des Dysprosium-Experiments durchführen zu können, wurde im Rahmen dieser Arbeit ein zusätzlicher Testaufbau aufgebaut.

3.2.1. Regelkreis Dysprosium-Experiment

Der Regelkreis, schematisch abgebildet in Abbildung 3.3, beginnt mit einem Laserstrahl, welcher durch einen akustooptischen Modulator, der in Abschnitt 3.3 beschrieben wird, in einen Strahl nullter und erster Beugungsordnung aufgeteilt wird. Die nullte Ordnung wird dabei auf eine Strahlfalle geschickt, während die erste Ordnung über zwei Spiegel in den Zeemanslower geleitet wird. Licht, das durch den letzten Spiegel vor dem Zeemanslower transmittiert wird, wird von einer Photodiode aufgefangen und ein Spannungssignal zu einem digitalen „Red Pitaya StemLab 14“-Board (Red Pitaya) geleitet. Das vom Red Pitaya gelesene Spannungssignal ist das Ausgangssignal $x(t)$ der Regelstrecke. Da die Spannung einer Photodiode proportional zur Laserintensität ist, sollte das Stabilisieren der Photodiodenspannung auch die Laserintensität stabilisieren. Anstelle des transmittierten Lichts wäre es wünschenswert, einen Strahlteiler in den Strahl erster Ordnung zu setzen, um ein stärkeres und besser kontrollierbares Ausgangssignal zu erhalten. Dies war jedoch aufgrund laufender Messungen und Platzbeschränkungen im Experiment nicht möglich.

Im Red Pitaya ist ein digitaler PID-Algorithmus implementiert, der einen Stellwert s von $x(t)$ abzieht und das resultierende Fehlersignal $e(t)$ auf Null regelt. Dazu gibt der Red Pitaya eine Stellgröße $y(t)$ aus. Diese wird dann von einem Signalverstärker[2] verstärkt und an den Amplitudenmodulationseingang eines AOM-Treibers[3] weitergegeben. In der Regelstrecke des Zeemanslowers wird das Signal auf zwei unterschiedliche Arten übertragen. Einmal als Leistung eines einfallenden Laserstrahls, dann als Spannungssignal einer Photodiode. Man müsste demnach eigentlich mehrere verschiedene Signale und Übertragungsfunktionen betrachten, um die Regelstrecke exakt zu beschreiben. Stattdessen kann aber auch die gesamte Regelstrecke als ein einziges System, welches durch das Produkt der einzelnen Übertragungsfunktionen (Faltungstheorem) beschrieben wird, angesehen werden. So kann die Regelstrecke als Ganzes, wie in Abschnitt 2.1, betrachtet werden.

3.2.2. Testaufbau

Für die Entwicklung der Laserintensitätsstabilisierung wurde außerdem ein Testaufbau errichtet, an dem die Funktionalität und Möglichkeiten der Laserstabilisierung unabhängig vom Dysprosium-Experiment getestet werden können. Es wird auch hier ein AOM verwendet, um die Laserleistung zu kontrollieren. Die erste Ordnung des Strahls nach dem AOM wird anders als im Dysprosium-Experiment direkt auf eine Photodiode geleitet. Anstatt eines 421 nm-Strahls, wird der Strahl einer Laserdiode mit 780 nm verwendet. Ansonsten unterscheidet sich der Testaufbau nicht vom Aufbau im Dysprosium-Experiment.

[2] INA111 - Texas Instruments
[3] Im Institut für Physik der Universität Mainz entwickelt

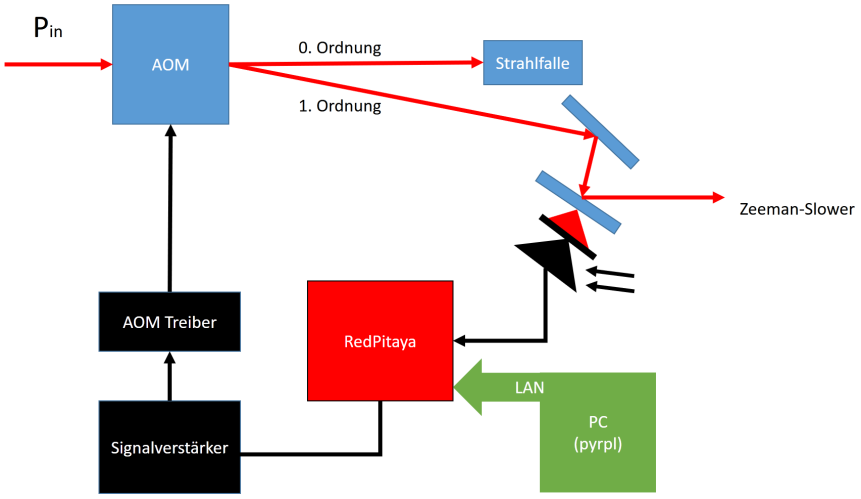

Abbildung 3.3.: Regelstrecke für das Lasersystem des Zeemanslowers. Rote Pfeile stellen den Laserstrahlen dar, schwarze Pfeile Spannungssignale.

3.3. Akustooptischer Modulator (AOM)

Um die Intensität des Laserstrahls zu kontrollieren, wird ein akustooptischer Modulator(AOM) verwendet, ein optisches Bauelement, durch das die Amplitude des Strahls sehr schnell ($\approx 500\,\mathrm{ns}$) variieren werden kann. Ein AOM besteht aus einem Kristall, an dem das einfallende Licht an einem Beugungsgitter gebeugt wird. Dieses Gitter entsteht durch das Einkoppeln einer stehenden oder laufenden Schallwelle in den Kristall, welche lokale periodische Änderungen der Dichte und damit auch des Brechungsindex im Kristall verursacht. Aus der Frequenz der eingekoppelten Schallwelle Ω_Λ und der Schallgeschwindigkeit im Kristall c_{Schall}, kann man die Wellenlänge der Schallwellen über

$$\Lambda = \frac{c_{Schall}}{\Omega_\Lambda} \tag{3.1}$$

berechnen. Λ entspricht dabei der Periode der Dichteänderungen und ist damit auch die Gitterkonstante des Beugungsgitters. Da die Lichtgeschwindigkeit sehr viel größer ist als die Schallgeschwindigkeit im Kristall, kann die Beugung näherungsweise wie die an einem statischen Gitter beschrieben werden. Für den Fall, dass die Strecke L, die das Licht im optischen Medium zurücklegt, groß ist, im Vergleich zu $\frac{\Lambda^2}{\lambda}$, befindet man sich im Bragg-Regime und die Lage der Lichtmaxima nach der Beugung kann mit der Bragg-Gleichung

$$2\Lambda\, sin(\Phi) = m\lambda, \qquad m = ..., -2, -1, 0, 1, 2, ... \tag{3.2}$$

beschrieben werden. Φ ist der Einfallswinkel des Lichts, der orthogonal zur Ausbreitungsrichtung der Schallwellen gemessen wird und m die Beugungsordnung. Je größer die periodischen Änderungen des Brechungsindex sind, desto mehr Licht wird gebeugt.

Die Größe dieser Änderungen kann durch die Amplitude der eingekoppelten Schallwelle variiert werden, wodurch eine direkte Möglichkeit zur Steuerung der Lichtintensität in den Beugungsordnungen entsteht.

3.4. Anforderungen an die Regelung

Um die Regelung in das Dysprosium-Experiment sinnvoll einbauen zu können, muss diese einige Anforderungen erfüllen. Die wichtigste Anforderung ist die Stabilisierung der Laserintensität über einen Zeitraum von mehreren Stunden, bis hin zu einem ganzen Labortag. Dabei soll die Regelung mindestens in einem Bereich von einigen Hertz arbeiten, um Langzeitdrifts der Intensität entgegenzuwirken. Eine schnellere Regelung ist in jedem Fall wünschenswert, da die Regelung dann in der Lage ist, Rauschen bis zu einer bestimmten Frequenz aus dem Lasersignal herauszufiltern. Des Weiteren soll es die Möglichkeit geben, den Setzwert während des Betriebs zu verändern, um darüber die Teilchenzahl in der MOT optimieren zu können. Ist das Dysprosium-Experiment in Betrieb, so werden von der Experimentsteuerung auf einem Laborcomputer kontinuierlich Messsequenzen gefahren. Jede dieser Sequenzen dauert etwa sieben Sekunden. Währenddessen es ist notwendig, den Laser des Zeemanslowers so schnell wie möglich ein- und auszuschalten. Das im Dysprosium-Experiment genutzte Experimentsteuerungssystem ist in der Lage, Schaltzeiten von 10 µs zu erreichen. Diese sollen mindestens für das Ein- und Ausschalten der Laserstabilisierung erreicht werden. Um den Einsatz der Intensitätsstabilisierung für den Laboralltag praktikabel zu gestalten, soll außerdem eine praxistaugliche Benutzeroberfläche entwickelt werden.

4. Red Pitaya

Zur Laserstabilisierung soll ein digitales „Red Pitaya STEMlab 125-14“-Board[1] (Red Pitaya) verwendet werden. Ein Bild vom Red Pitaya ist in Abbildung 4.1 zu sehen Der Red Pitaya ist ein programmierbares, multifunktionales Laborinstrument, das einen ARM 9 Prozessor mit einem „Xilinx Zynq 7010 FPGA“-Board, sowie je zwei 125 MHz analog digital Wandler (ADC) und digital analog Wandler (DAC) verbinden. Des Weiteren sind 16 Allzweckein-/ausgaben (GPIO) und je 4 „langsame“ADCs und DACs (100kS/s)[2] vorhanden. Der FPGA und die 125 MHz ADCs und DACs sollten eine Regelbandbreite bis zu 20 MHz bereitstellen [7].
Vom Hersteller wird eine Linux-Distribution für das Board bereitgestellt [**?**], die es ermöglicht mit dem Board über ein Webinterface, SSH-, oder einem SCPI-Server zu kommunizieren. Im Webinterface sind verschiedene Anwendungen wie z. B. ein Oszilloskop und ein Funktionsgenerator, ein Netzwerkanalyseprogramm, oder eine PID-Controller-Anwendung verfügbar. Die für diese Arbeit wichtigsten technischen Daten sind in Tabelle B.1 zusammengefasst, sowie einige Hinweise zum Einrichten eines Red Pitaya in einem Universitätsnetzwerk in A.1. Zum besseren Verständnis des Red Pitaya lohnt es sich, einen kurzen Blick auf die Funktionsweise und den Aufbau von Anwendungen für diesen zu werfen.

Field Programmable Gate Array (FPGA)

Der Regler für die Intensitätsstabilisierung wird auf einem FPGA implementiert, der direkt an je zwei Analog-Digital-Converter (ADC) und Ditigal-Analog-Converter (DAC) angeschlossen ist.
Ein FPGA besteht aus konfigurierbaren Logik- und Speicherblöcken die mit einem Netz von Leitungen verbunden sind, welche an konfigurierbaren Leitungskreuzungen zusammenlaufen. Beim Konfigurieren des FPGA wird jedem Block eine fest definierte Binäroperation zugewiesen, die er mit seinen Eingangssignalen durchführen sollen. Genauso erhalten die Kreuzungen Anweisungen, wie sie die Leitungen miteinander verbinden sollen. Da sich während der Laufzeit die Aufgaben der einzelnen Logikblöcke nicht ändern, können alle Blöcke gleichzeitig arbeiten, was die Parallelisierung vieler Prozesse ermöglicht. Dadurch können auch große Datenmengen schnell verarbeitet werden, was es ermöglicht, digitale Regelungen im MHz-Bereich durchzuführen. [5]
Um einen FPGA zu programmieren, ist es nicht notwendig jeden Block und jede Kreuzung einzeln zu konfigurieren. Stattdessen werden höhere Programmiersprachen, wie

[1] Das Board wurde von Red Pitaya 1.1 zu StemLab 125-14 umbenannt.
[2] Samples pro Sekunde

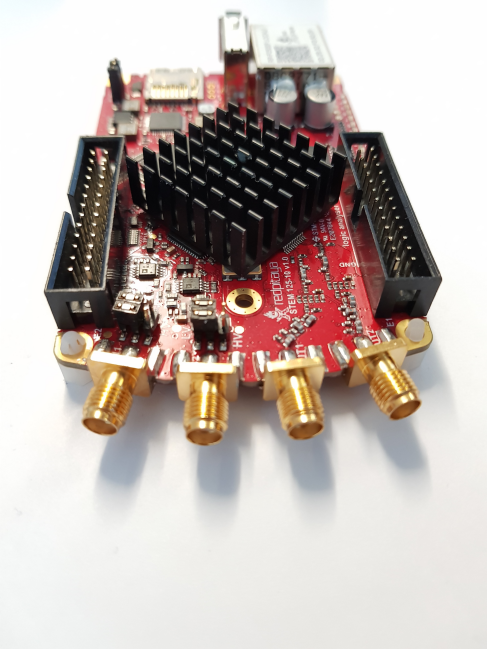

Abbildung 4.1.: Frontaufnahme des Red Pitaya. Bei den goldenen SMA-Anschlüssen handelt es sich von links nach rechts um: Eingang-1, Eingang-2, Ausgang-1, Ausgang-2. An den Rändern links und rechts befinden sich die GPIOs und die langsamen ADCs und DACs. Der schwarze Block in der Mitte ist das Kühlelement für den FPGA.

„Verilog“oder „VHDL“verwendet, die anschließend von einem Compiler in eine FPGA-spezifische Konfiguration umgewandelt werden. Funktionen werden bei „Verilog“von sogenannten „Modulen“übernommen, welche selbst wiederum Untermodule enthalten können. Der hierarchische Aufbau ermöglicht es, in gewissen Grenzen Methoden der objektorientierten Programmierung, zu verwenden.

Aufbau von Red Pitaya Anwendungen

Alle Anwendungen, die auf dem Red Pitaya standardmäßig vorhanden sind, bestehen aus der Benutzeroberfläche (Front-End), die der Anwender sieht und einem Back-End, das dieser nicht sieht. Das Back-End ist für den Ablauf des Programms zuständig. Das Front-End besteht aus einer Website, die über eine Programmierschnittstelle (API[3]) mit dem „Controller“, der das Back-End darstellt, kommuniziert. Der „Controller“ist ein C++ Programm, dass Parameter im FPGA ändert, Rechnungen durchführt und Signale an das Webinterface sendet [9].

[3] Application-Programming-Interface

5. Erweiterung von PyRPL

Für die Implementierung der Laserstabilisierung wird das Open-Source Softwarepaket PyRPL [7] (Python Red Pitaya Lockbox) verwendet, das ein eigenes FPGA-Image für den Red Pitaya, eine Benutzeroberfläche (GUI) und Python-API für den Client-PC mitbringt. Es sind schon viele nützliche Instrumente für die Implementierung einer Intensitätsstabilisierung vorhanden, wie digitale Filter, Signalgeneratoren und PID-Regler. Es wurde entwickelt, um eine Pound-Drever-Hall-Stabilisierung mit dem Red Pitaya zu ermöglichen [7]. Durch einen modularen Aufbau der im FPGA implementierten Instrumente und des Python-API ist es jedoch möglich, auch andere komplexe Anwendungen zu realisieren. Als Teil dieser Arbeit hat der Autor die FPGA-Belegung und Python-API erweitert, um die Anforderungen aus 3.4 zu erfüllen.
Zu Beginn des Kapitels wird ein Überblick über PyRPL gegeben. Nicht alle Anforderungen, die an die Regelung in 3.4 gestellt wurden, sind als Funktion in PyRPL eingebaut. Deswegen wird analysiert, welche Anforderungen an die Regelung schon von PyRPL erfüllt sind und welche noch erweitert werden müssen. Anschließend werden mögliche Lösungen für diese Anforderungen, sowie deren Vor- und Nachteile diskutiert. Für das Problem des Ein- und Ausschaltens der Regelung, wird für alle vorgeschlagenen Lösungen eine mögliche Implementierung besprochen. Das Ziel des Kapitels ist es, den Leser soweit mit dem Red Pitaya, PyRPL und der FPGA-Programmierung vertraut zu machen, dass dieser in der Lage ist, die Intensitätsstabilisierung selbst weiterzuentwickeln und für seine Zwecke anzupassen.

5.1. Überblick über PyRPL

PyRPL besteht aus drei Teilen, die auf unabhängigen Hardwaresystemen laufen und über Kommunikationsprotokolle miteinander interagieren können. Auf dem Client-PC läuft eine Python-Anwendung, die Befehle zu einem Programm auf dem Linux-System des Red Pitaya sendet. Dieses kommuniziert dann mit dem FPGA und schreibt oder liest Variablen darin, wodurch dessen Funktion angepasst oder Signale ausgelesen werden können.

5.1.1. FPGA-Design

Die gesamte Signalverarbeitung findet auf dem FPGA des Red Pitaya statt, da nur so die benötigte Regelbandbreite erreicht werden kann. Der folgende Abschnitt soll dem Leser ein Verständnis vom modularen Aufbau des FPGA in PyRPL vermitteln.
Da sowohl der vom Hersteller bereitgestellte, als auch der in PyRPL verwendete FPGA-Code in Verilog geschrieben ist, wird die Sprache auch im Rahmen der Ar-

beit verwendet.
An oberster Stelle steht bei PyRPL das „Top-Modul“, das alle anderen Module beherbergt und mit den Ein- und Ausgängen des Red Pitaya (ADCs, DACs, GPIO, usw.) kommuniziert. Darunter befindet sich ein digitaler Signalprozessor (DSP) und Multiplexer. Dieser beherbergt die eigentlichen Module der Signalverarbeitung (Oszilloskop, Funktionsgenerator, PID,...) und hat die Aufgabe, Ein- und Ausgangssignale vom Top-Modul zu den Signalverarbeitungsmodulen hin und zurück zu leiten, sowie die Signale zwischen den Signalverarbeitungsmodulen weiterzugeben. Dadurch kann ein Signal vom Eingang des Red Pitaya z.B. durch mehrere PIDs nacheinander und anschließend zum Oszilloskop und den Ausgängen gesendet werden. Alle Signale, die von den Modulen zu einem der beiden Ausgänge geschickt werden, werden dort summiert und vom DAC ausgegeben. Während es möglich ist, eine solche Funktionsweise direkt in das FPGA-Design zu schreiben, ermöglicht der modulare Aufbau, die Funktion des FPGA nach dem Kompilieren des FPGA-Designs zu ändern. [7]

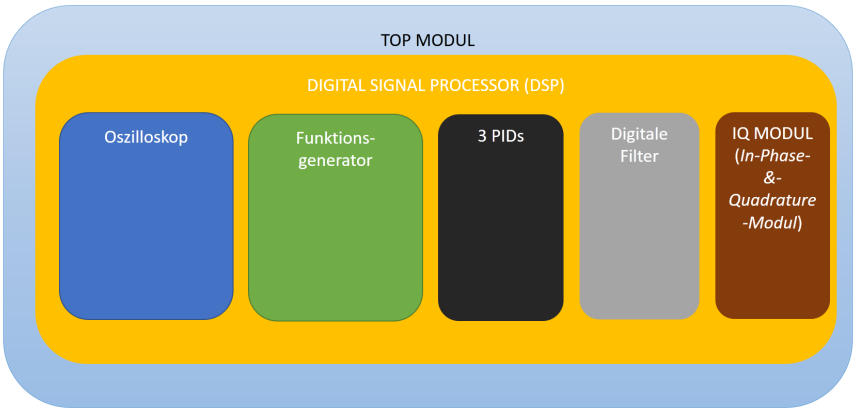

Abbildung 5.1.: Exemplarisches FPGA-Design mit Digitalem Signal Prozessor und Multiplexer.

5.1.2. Kommunikation zwischen Red Pitaya und Client-PC

Das Front-End von PyRPL ist ein Python-API, welche auf einem Client-PC läuft. Mit dem Red Pitaya kommuniziert dieses über ein Kommunikationsprotokoll, das kurz erläutert werden soll.
Wenn auf einem Client-Computer PyRPL gestartet wird, verbindet sich dieser über das SSH-Protokoll mit dem Linux-Betriebssystem auf dem Red Pitaya. Anschließend lädt der Client eine kompilierte FPGA-Konfiguration vom Client-PC auf den Red Pitaya herunter und führt diese über einen Linux-Befehl auf dem FPGA aus.
Dann wird ein C-Programm namens „monitor_server“auf den Red Pitaya geladen und ausgeführt, welches über das TCP/IP-Protokoll auf Anweisungen des Client wartet. Diese Anweisungen können entweder „lesen“oder „schreiben“sein, gefolgt von einer

32 Bit-Adresse und der Anzahl der zu lesenden oder schreibenden 32 Bit-Pakete. Über ein in den FPGA implementiertes Bus-System kann dann mithilfe der im FPGA konfigurierten Logik, über die 32 Bit-Adresse auf einzelne Register im FPGA zugegriffen werden. Alle Module auf dem FPGA werden auf dem Client-PC als Python-Objekte dargestellt. Diese enthalten die selben Variablen, die in den FPGA-Modulen einstellbar sind und kennen deren Adressen im FPGA. Dadurch bilden sie eine Brücke zwischen Python-API und FPGA, sodass Variablen im FPGA vom Nutzer des Python-API genauso verwendet werden können, wie normale Python-Variablen [7].

5.1.3. Bestandsaufnahme der PID-Anwendung in PyRPL

In PyRPL[1] sind drei PID-Module implementiert. Jedes dieser Module enthält Vier digitale Filter erster Ordnung und die Möglichkeit, sein Eingangssignal $x(t)$ frei zu wählen. K_p-Werte können, bis auf Rundungsfehler, zwischen [-2048, 2048] frei gewählt werden, sowie I-Werte zwischen $[-3 \cdot 10^4, 3 \cdot 10^4]s^{-1}$. Die differentielle Komponente des PID ist deaktiviert, da sie nicht für die Lockbox-Anwendung notwendig ist. Der Speicher für die Integralsumme überrepräsentiert das Eingangssignal um zwei Bit, sodass auch kleine Langzeitabweichungen vom Setzwert aufgenommen werden können, die nur im statistischen Mittel messbar sind. Weiterhin ist die Integralsumme beschränkt auf [-4,4] V. Vorteile einer beschränkten Integralsumme wurden bereits in Abschnitt 2.4 besprochen. Um zu verhindern, dass der PID-Regler aus seinem Regelbereich herausläuft, und sensible Apparaturen beschädigt, oder einfach nicht mehr regeln kann, ist es möglich, eine Minimal- und Maximalspannung des Stellwerts $y(t)$ einzustellen, die die Ausgabe des PIDs beschränkt. Diese beschränken jedoch nur das Ausgangssignal eines einzelnen PIDs, und nicht das summierte Signal an den Ausgängen. Verwendet man beispielsweise mehrere PIDs, die den selben DAC als Ausgang nutzen, kann die Maximalspannung überschritten werden.

5.2. Analyse der Regelanforderungen

Die Möglichkeit einer Intensitätsstabilisierung im Bereich von einigen Hertz ist, mit den in PyRPL vorhandenen PID-Modulen, ohne Weiteres gegeben. Welche Grenzen sich für die Regelgeschwindigkeit darüber hinaus ergeben, muss die Implementierung in die Experimente zeigen. Sowohl die Regelparameter, als auch der Setzwert der PIDs kann im laufenden Betrieb über die GUI oder das Python-API geändert werden. Das schnelle Ein- und Ausschalten, der Regelung ist nicht ohne Weiteres für die PID-Module vorgesehen. Dazu ist die Programmierung einer praxistauglichen Benutzeroberfläche für den Laboralltag vorgesehen. Außerdem hat sich die Frage ergeben, ob der in PyRPL deaktivierte D-Teil des PID in der Lage ist, die Regelung zu verbessern.

[1] Stand 21.8.2018

5.3. Ein- und Ausschalten der Regelung

Schnelles Ein- und Ausschalten der Regelung mit Schaltzeiten unter 10 μs stellte ein zentrales Problem in dieser Arbeit dar. Es wurden drei verschiedene Lösungsansätze getestet, die hier vorgestellt werden.

5.3.1. Python-Script

Am einfachsten ist es, den PID Regler über die PyRPL GUI manuell ein- und auszuschalten. Ein solcher Ansatz ist für das Dysprosium-Experiment mit automatisierter Experimentsteuerung natürlich nicht praktikabel. Es ist jedoch möglich, anstelle der GUI das Python-API zu verwenden, welches die selben Möglichkeiten bietet.

Vor- und Nachteile

Der Vorteil dieser Lösung ist dessen Einfachheit, da keine Änderungen am FPGA-Code oder der Python-API vorgenommen werden müssen. Ein kurzes Python-Script reicht aus, um die Funktionalität zu gewähren. Der Nachteil ist die geringe und nicht konstante Geschwindigkeit dieser Lösung. Die serverseitigen Abfragen des Red Pitaya finden in fest definierten Zeitintervallen statt und in jedem Intervall kann nur ein Befehl übertragen werden. Selbst wenn das TCP/IP-Protokoll bei jeder Übertragung perfekt funktioniert, braucht ein Befehl im Schnitt ein halbes Abfrageintervall bis er ausgeführt werden kann. Dies und die Ausführungsgeschwindigkeit von Befehlen die über die CPU des Red Pitaya laufen und Variablen im FPGA Register ändern, beschränken die Schaltzeit dieser Möglichkeit auf einige Millisekunden.

Implementierung

Das Ausgangssignal der Photodiode $x(t)$ wird an den Eingang-1[2] des Red Pitaya angeschlossen. Der Ausgang-1[3] des Red Pitaya wird an den Signalverstärker des Regelkreises angeschlossen.
In Listing 5.1 wird ein Code-Beispiel gezeigt, welches einen PID-Regler einschaltet, der mit den Parametern $K_p = 1$ und $K_i = 1000/s$ versucht, das Signal x(t) am Eingang-1 auf den Setzwert 0.3 V zu regeln. Dafür kann dieser eine Spannung zwischen 0 V und 0.5 V auf Ausgang-1 geben.

[2] Auf dem Red Pitaya mit „IN1“bezeichnet.
[3] Auf dem Red Pitaya mit „OUT1“bezeichnet.

```
1 from pyrpl import Pyrpl
2 #Hier wird eine Verbindung zum Red Pitaya aufgebaut
3 p=Pyrpl(ip=100.100.100.100)
4 r=p.rp
5 # FPGA-Parameter koennen als Argumente in die Funktion setup uebergeben
      werden
6 r.pid0.setup(input="in1", ouput_direct="out1",p=1,i=1000,inputfilter
      =[0,0,0,0], min_voltage=0)
7 # oder als Variablen einzeln gesetzt werden
8 r.pid0.setpoint=0.3
9 r.pid0.max_voltage=0.5
```

Listing 5.1:

Die Funktion Pyrpl (Zeile 3) erzeugt eine Socket-Instanz, die eine Verbindung zum Red Pitaya mit der IP 100.100.100.100 herstellt. Nachdem diese Verbindung erfolgreich hergestellt wurde, kann auf das p.rp Objekt zugegriffen werden, welches die Module der Signalverarbeitung als Objekte zur Steuerung der auf dem FPGA implementierten Hardware-Module enthält. Danach können die Attribute des Hardware-Moduls entweder über die setup-Funktion gesetzt werden, oder nacheinander als Variablen. Für das PID-Modul sind diese in Tabelle B aufgelistet. Um die Ausgabe des PID auszuschalten, kann man die Variable "output_direct" auf "off" setzen (Zur Installation von PyRPL siehe A.2).

5.3.2. Steuerung über analogen Setzwert

Die zweite Möglichkeit den PID ein- und auszuschalten beruht auf der Idee, diesem einen variablen Setzwert über einen der analogen Eingänge zu geben. Dieser kann dann zwischen beliebigen Werten variiert werden, also auch zwischen 0 und dem gewünschten Setzwert. Diese Funktion ist in der verwendeten PyRPL-Version nicht verfügbar. Durch den DSP kann das gewünschte Verhalten aber durch einfachen Python-Code erreicht werden, indem alle drei auf dem FPGA vorhandene PID-Module genutzt werden. Das Flussdiagramm 5.2 zeigt die Verschaltung der Module.
Der gewünschte Setzwert wird von der Experimentsteuerung in den Eingang-1 gegeben, das Antwortsignal der Regelstrecke in den Eingang-2[4]. Um ein Fehlersignal zu erhalten, muss der Setzwert vom Antwortsignal abgezogen werden. Dazu wird das Antwortsignal durch ein PID Modul mit $K_p = 1$ an den Ausgang-2 gesandt. Das Setzwert-Signal wird durch einen anderen PID mit $K_p = -1$ an den selben Ausgang geleitet. Alle Signale an einem Ausgang des Red Pitaya werden dort summiert. Deshalb liegt an Ausgang-2 ein Fehlersignal mit variablem Setzwert an. Der DSP-Multiplexer erlaubt es, ein Ausgangssignal im FPGA als Eingangssignal für weitere Module zu verwenden. PID-3 kann dann verwendet werden um eine PID-Regelung mit dem Fehlersignal auf Ausgang-2 durchzuführen und ein Regelsignal über Ausgang-1 an die Regelstrecke zu senden.
Diese Art der Steuerung bringt einige Vorteile mit sich. Genau wie die erste Möglichkeit ist sie ohne eine Veränderung des FPGA-Codes möglich, was spätere Verände-

[4] Auf dem Red Pitaya als OUT2 bezeichnet

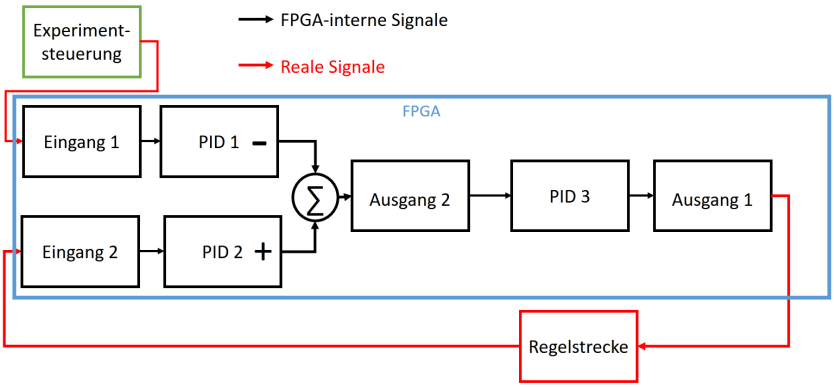

Abbildung 5.2.: Schematische Darstellung der Steuerung des PIDs über einen analogen Setzwert. PID 1 und PID 2 erzeugen ein Fehlersignal $e(t) = x(t) - s(t)$ auf Ausgang-2. Die eigentliche Regelung findet durch PID 3 statt, den Stellwert f auf den Regelkreis gibt.

rungen vereinfacht. Dazu ermöglicht es die Steuerung über einen analogen Setzwert die Regelung nicht nur ein- und auszuschalten, sondern auch stabilisierte Funktionen auf das Lasersignal zu modulieren, indem diese als Setzwert vorgegeben werden.Diese Möglichkeit der Steuerung ist wesentlich schneller als die Erste und nur beschränkt durch die Geschwindigkeit der Regelung von einem Setzwert auf einen anderen. Abhängig von der Regelstrecke und den Regelparametern können Schaltzeiten zwischen $5 - 100\mu s$ erreicht werden. Nachteilig sind der hohe Ressourcenaufwand, da alle Ein- und Ausgänge des Red Pitaya belegt sind und ein analoger Ausgang der Experimentsteuerung verwendet wird, um ein Gerät ein- und auszuschalten. Außerdem ist die Geschwindigkeit des Schaltens nicht optimal, da anstelle eines Schaltvorgangs ein Regelvorgang stattfindet. Ein Codebeispiel ist in D.1 zu sehen.

5.3.3. Steuerung über einen digitalen Trigger

Die GPIO-Pins des Red Pitaya können als digitale Trigger verwendet werden. Für das Oszilloskop und den Funktionsgenerator sind diese Funktionen in PyRPL vorhanden, beim PID jedoch nicht. Dies erfordert eine Anpassung des FPGA-Designs. Ein Trigger muss dabei die Signalausgabe des PID ausschalten und gleichzeitig die Integralsumme einfrieren, um einen „Integral-Windup“zu vermeiden. Bei einer guten oder optimalen Regelung sollte nach hinreichend langer Zeit der Regelwert dem Setzwert entsprechen. Das bedeutet, dass das Fehlersignal $e(t) = 0$ ist und damit auch der P- und D-Teil einen Stellwert von Null ausgeben. Die Integralsumme im I-Teil ist dann gleich dem Stellwert. Wird die Regelung über einen Transistor-Transistor-Logik-Trigger (TTL-Trigger), wie oben beschrieben, gesteuert, so ist der Stellwert beim Einschalten der Regelung wieder auf genau dem Wert, der vorher für eine optimale Regelung gesorgt hat, da die Integralsumme während des Ausschaltens der Regelung gespeichert ist.

Vor- und Nachteile

Der einzige Nachteil dieser Lösung liegt in der Änderung des FPGA-Codes. Dadurch ist die verwendete Software nicht mehr mit neuen Updates von PyRPL kompatibel. Der größte Vorteil ist die Geschwindigkeit des TTL-Triggers, die es erlaubt, Geschwindigkeiten unterhalb 1 µs für Schaltvorgänge zu erreichen. Da die Regelung beim Einschalten wieder zu einem vorher optimalen Stellwert zurückkehrt, sind solche Regelgeschwindigkeiten auch erreichbar. Darüber hinaus besteht mit dieser Methode prinzipiell die Möglichkeit, zwei Lasersysteme mit einem Red Pitaya zu stabilisieren.

5.3.4. Implementierung des Triggers

Die Implementierung des TTL-Triggers kann in drei Teile aufgeteilt werden. Als Erstes muss ein Trigger-Signal von einem GPIO zum PID-Modul des FPGA übergeben werden. Zweitens soll es möglich sein, das TTL-Triggern über die Python-API ein- und auszuschalten, sollte es nicht benötigt werden. Als Drittes muss die Integralsumme eingefroren und der Stellwert $y(t)$ auf Null gesetzt werden, wenn der TTL-Trigger die Regelung ausschaltet.

Übergabe des Trigger-Signals vom GPIO

Das erste Problem ist es, das Trigger-Signal vom GPIO durch den FPGA zum PID-Modul zu führen. Das Top-Modul (red_pitaya_top.v) bekommt die Werte der GPIOs als Variablen übergeben. Von dort aus muss eine dieser Variablen über das DSP-Modul (red_pitaya_dsp.v) an die PIDs (red_pitaya_pid_block.v) übergeben werden (siehe **??**). Als GPIO-PIN wird der DIO0_P (siehe C.3) verwendet, welcher auch schon als Trigger für das Oszilloskop- und den Funktionsgenerator-Modul verwendet wird.
Das Listing 5.2 zeigt ein Code-Beispiel, in welchem das PID-Modul Variablen wie bspw. das Trigger-Signal erhält, die vom übergeordneten DSP-Modul (siehe Listing D.2) übergeben wurden. Zu Beginn des Listings 5.2 werden Konstanten innerhalb des Moduls, sogenannte Parameter, definiert die anstelle von Zahlen im Code verwendet werden können. Alle Parameter im PyRPL-Code sind daran zu erkennen, dass sie nur aus Großbuchstaben bestehen. Danach wird definiert, welche Variablen dem PID-Modul vom übergeordneten Modul übergeben wird (input) und welche es an dieses zurückgibt (output). Die Variablen nach dem Kommentar „communication with PS“implementieren das Bus-System, welches zur Kommunikation zwischen Python-Client und FPGA verwendet wird. „addr“bezeichnet die statische Adresse einer Variable auf den FPGA, „wen“(schreiben eingeschaltet) oder „ren“(lesen eingeschaltet) legen die Art der Kommunikation fest. „wdata “und „rdata“sind die Register, in denen die eigentlichen Variablen enthalten sind. In Zeile 23 wird das Trigger-Signal an die Variable, „trigger_ext_i“übergeben.

```
1 module red_pitaya_pid_block #(
2   //parameters for gain control (binary points and total bitwidth)
    ,
3   parameter     ISR = 32          ,//official redpitaya: 18 ,
```

```
4     parameter      GAINBITS = 24      ,
5 )
6 (
7     // data
8     input                   clk_i           ,  // clock
9     input                   rstn_i          ,  // reset - active low
10    input       [ 14-1: 0]  dat_i           ,  // input data
11    output      [ 14-1: 0]  dat_o           ,  // output data
12
13    //externer Trigger
14    input                   trig_ext_i      ,  // external trigger ---Lehnen
15
16    // communication with PS
17    input       [ 16-1: 0]  addr,
18    input                   wen,
19    input                   ren,
20    output reg          ack,
21    output reg  [ 32-1: 0]  rdata,
22    input       [ 32-1: 0]  wdata
23 );
```

Listing 5.2: In diesem Codebeispiel erhält das PID-Modul Variablen des übergeordneten DSP-Modul gegeben. Die Reihenfolge muss dabei die Selbe sein, in der die Variablen im DSP-Module in Listing D.2 übergeben werden.
Zeilen die mit //—Lehnen kommentiert sind, wurden vom Autor ergänzt, um den TTL-Trigger zu implementieren. Der Originalcode wurde verkürzt

Kommunikation mit dem Python-API

In Listing 5.3 wird das Bus-System dargestellt, mit dem vom Benutzer gesetzte Variablen im FPGA-Modul gelesen oder geschrieben werden. In den ersten beiden Zeilen werden Register, also Speicherplätze, reserviert. In den eckigen Klammern steht dabei die Länge und Leserichtung des Speichers. Es wird ein Register für den Setzwert in Zeile Eins definiert und ein Register „TTL_enable“, welches speichert, ob der TTL-Trigger über das Python-API ein- oder ausgeschaltet ist.
In Zeile Fünf steht ein „always“-Block welcher jedes mal ausgeführt wird, wenn die Bedingung in der Klammer erfüllt ist, in diesem Fall bei jeder positiven Flanke des Takt-Signals. Jede Variable, auf die vom Client-PC zugegriffen werden soll, bekommt durch die if-Abfrage eine statische Adresse zugewiesen. Darüber kann der Client-PC bestimmen, welcher Variable die neuen Werte - gespeichert in „wdata“- zugewiesen werden sollen. Da das Übertragungsprotokoll vorsieht, dass jedes mal 32-bit übertragen werden, müssen je nach Registerlänge L der zugewiesenen Variable nur die entsprechenden unteren L Bit von „wdata“ausgelesen werden.
Das Auslesen von Registern funktioniert fast genauso, wie das Einlesen. Die Werte aus einem Register mit der passenden Adresse werden an den output „rdata“(read data) übergeben. Da auch dieser 32-Bit lang ist, werden kürzere Register mit führenden Nullen aufgefüllt.

```
reg [ 14-1: 0] set_sp;    // set point
reg TTL_enable; //  enabels TTL triggering ---Lehnen

//  System bus connection
always @(posedge clk_i) begin
   //Schreiben von Variablen im FPGA
      if (wen) begin
         if (addr==16'h104)   set_sp  <= wdata[14-1:0];
         if (addr==16'h130)  TTL_enable <=wdata[0]; // ---Lehnen
      end

    //Lesen von Variablen aus dem FPGA
    casez (addr)
       16'h104 : begin ack <= wen|ren; rdata <= {{32-14{1'b0}},set_sp};
    end
        16'h130 : begin ack <= wen|ren; rdata <= {{32-1{1'b0}},TTL_enable
    }; end //---Lehnen
```

Listing 5.3: Der Code in diesem Listing implementiert das Schreiben und Lesen von Variablen über das Bus-System im FPGA. Register können über die im Code definierte Adresse ausgelesen oder geschrieben werden. Jedem Register, auf das über das Python-API zugegriffen werden soll, muss bei der FPGA-Programmierung manuell eine Adresse zugewiesen werden. (Originalcode wurde verkürzt.)

Damit über das Python-API auf die Variable „TTL_enable“im FPGA zugegriffen werden kann, muss noch die zum PID zugehörige Python-Klasse pid.py verändert werden. Die Datei befindet sich im Order pyrpl/hardware_modules. Zur Übergabe der Werte reicht es aus ein neues „BoolRegister“, wie in Listing 5.4 gezeigt, zu erstellen und diesem die Adresse, welche vorher im FPGA festgelegt wurde zu übergeben. Alle Register und die dazugehörige Logik sind in der Datei pyrpl/attributes.py definiert. Die Variable „TTL_trigger “kann dann verwendet werden um die Triggerfunktion für den jeweiligen PID ein- oder auszuschalten.

```
    TTL_trigger = BoolRegister(0x130, doc="If True ext Trigger activates
    the pid)")
```

Listing 5.4: In dieser Zeile wird die pid.py Datei um ein BoolRegister erweitert. Dieses implementiert die Kommunikation mit dem FPGA Register für eine boolsche Variable. Auf das zugehörige Register im FPGA kann dann über das Python-API genau wie auf eine herkömmliche Python-Variable zugegriffen werden.

Stoppen der Regelung

Um die Regelung auszuschalten, müssen die Integralsumme eingefroren und der Stellwert des PID auf Null gesetzt werden. Die Integralsumme $Isum$ kann eingefroren werden, indem diese nicht mehr verändert wird, solange die Regelung nicht aktiv ist.

In jedem Takt des FPGA wird die Operation

$$Isum_n = Isum_{n-1} + ki\_mult; \quad mit \quad ki\_mult = e(t) * K_i \tag{5.1}$$

durchgeführt (Listing 5.5, Zeile 9). Zum Einfrieren von $Isum$ wird die Variable ki_mult auf Null gesetzt (Listing 5.5, Zeile 3), wenn die Regelung deaktiviert ist, sodass

$$Isum_n = Isum_{n-1} + ki\_mult = Isum_{n-1}; \quad mit \quad ki\_mult = 0'. \tag{5.2}$$

```
1 always @(posedge clk_i) begin
2       if(trig_ext_i==1'b0 && TTL_enable)  // wenn TTL_enable==True und
    ext_trig==low  //--- Lehnen
3          ki_mult  <= {15+GAINBITS{1'b0}}; // => ki_mult=0 ---Lehnen
4       else
5          ki_mult <= $signed(error) * $signed(set_ki) ;
6
7         int_reg <= int_sum[IBW-1:0]; // use sum as it is
8   end
9 assign int_sum = $signed(ki_mult) + $signed(int_reg) ;
```

Listing 5.5: Dieses Listing zeigt das Einfrieren der Integralsumme, wenn die neu hinzugefügte Variable TTL_enable über das API auf „True“gesetzt wurde und am TTL-Eingang des Red Pitaya eine keine Spannung anliegt. (Originalcode wurde verkürzt.)

Der Stellwert kann über eine äquivalente if-Abfrage ausgeschaltet werden, wenn die Regelung ausgeschaltet ist.

Kompilierung des Codes

Der neue FPGA-Code muss vor der Verwendung auf dem FPGA kompiliert werden. Die dafür notwendigen Programme und Befehle werden in A.3 beschrieben. Die erfolgreich kompilierte Binärdatei muss dann in das pyrpl/fpga Verzeichnis kopiert werden, in dem sich alle Python-Pakete befinden. Außerdem müssen die veränderten Python-Dateien im passenden Ordner das PyRPL-Python-Pakets ersetzt werden (siehe A.2). Anschließend kann PyRPL mit den neu hinzugefügten Funktionen verwendet werden.

5.4. Weitere Änderungen des FPGA-Codes

Es wurden außer der Implementierung des TTL-Trigger noch weitere Änderungen am FPGA-Code vorgenommen, die hier nicht im Detail besprochen werden. Es sollte der D-Teil des PID-Moduls, aktiviert und getestet werden, um die Frage zu beantworten, ob damit eine schnellere Regelung möglich ist. Außerdem wurde ein automatisches Zurücksetzen der Integralsumme implementiert, das bei der Stabilisierung von Systemen mit störanfälligen Regelstrecken hilfreich sein kann.

D-Teil

Der D-Teil des PID ist im PyRPL FPGA-Code schon enthalten, wurde aber auskommentiert, da dieser nicht für die Lockbox-Anwendung notwendig ist. Er kann also einfach wieder aktiviert werden, indem die Kommentare vor den Code-Zeilen gelöscht werden. Dann kann die Frage beantwortet werden, ob mit dem D-Teil eine bessere Regelung möglich ist, als nur mit einem PI-Regler.

Automatisches Zurücksetzen der Integralsumme

Im Fall eines Integral-Windup kann es hilfreich sein, wenn die Integralsumme automatisch zurückgesetzt wird, sodass der Stellwert wieder in einen linearen Regelbereich zurückkehrt. Dies kann jedoch oft auch Instabilitäten in ansonsten stabilen Systemen verursachen, weshalb diese Funktion nicht dauerhaft aktiv sein soll, sondern über das Python-API aktiviert werden kann. Die Variable „.ivalReset“[5] schaltet diese Funktion ein oder aus. Sie ist standardmäßig ausgeschaltet.

[5] kann auf True (Eingeschaltet) oder False (Ausgeschaltet) gesetzt werden.

6. Ergebnisse

Das primäre Ziel der Arbeit war die Stabilisierung der Laserintensität am Zeemanslowers des Dysprosium-Experiments. In diesem Kapitel werden alle nötigen Schritte zum Aufbau einer Intensitätsstabilisierung mit der im vorherigen Kapitel 5 erweiterten Software diskutiert. Danach werden verschiedene Methoden zum Finden optimaler Regelparameter im Experiment miteinander verglichen. Anschließend wird die im Dysprosium-Experiment erreichte Intensitäts- und Teilchenzahlstabilität analysiert. Zum Abschluss des Kapitels werden weitere Störeinflüsse auf die Teilchenzahl betrachtet.

6.1. Aufbau der Intensitätsstabilisierung

Zum Aufbau einer Intensitätsstabilisierung sollte als Erstes die Regelstrecke analysiert werden. Es kann hilfreich sein, die Frequenzcharakteristik der Regelstrecke zu bestimmen, um ein besseres Verständnis von dieser zu erhalten. Oft muss der Regelbereich des Reglers eingeschränkt werden, sodass ein linearer Zusammenhang zwischen y(t) und x(t) besteht. In jedem Fall müssen die Regelparameter passend zum Regelkreis gewählt werden, um eine gute Stabilisierung zu erreichen.

6.1.1. Charakterisierung der Regelstrecken

Um die Möglichkeiten und Einschränkungen der Stabilisierung zu bestimmen, sowie eine theoretische Analyse zu ermöglichen, ist es hilfreich, eine Frequenzanalyse in Form eines Bode-Diagramms zu erstellen. Die Frequenzcharakteristik der Regelstrecke kann dann durch bekannte elektronische Bauteile nachgebildet werden. Beim Dysprosium-Experiment ist ein Tiefpass erster Ordnung mit einer Grenzfrequenz von $4,84 \pm 0.73\,\mathrm{kHz}$ eine gute Näherung an die Regelstrecke, wie man in Abbildung 6.1 sehen kann.
Damit ist es möglich, die Langzeitstabilisierung im Bereich von mehreren Minuten oder Stunden durchzuführen und gleichzeitig das Signal im Millisekundenbereich zu stabilisieren. Außerdem ist es nicht nötig, die Diskretisierungsfehler der Signale zu betrachten, die bei einer digitalen Regelung entstehen können, da die Taktrate des FPGA mit $125\,\mathrm{MHz} \gg 5\,\mathrm{kHz}$ ist. Da das Signal auf der Photodiode, bedingt durch den Aufbau sehr schwach ist, muss ein Verstärkungsfaktor von $70\,\mathrm{dB\,V}$ verwendet werden. Dieser verringert die Bandbreite der Photodiode auf $5\,\mathrm{kHz}$.

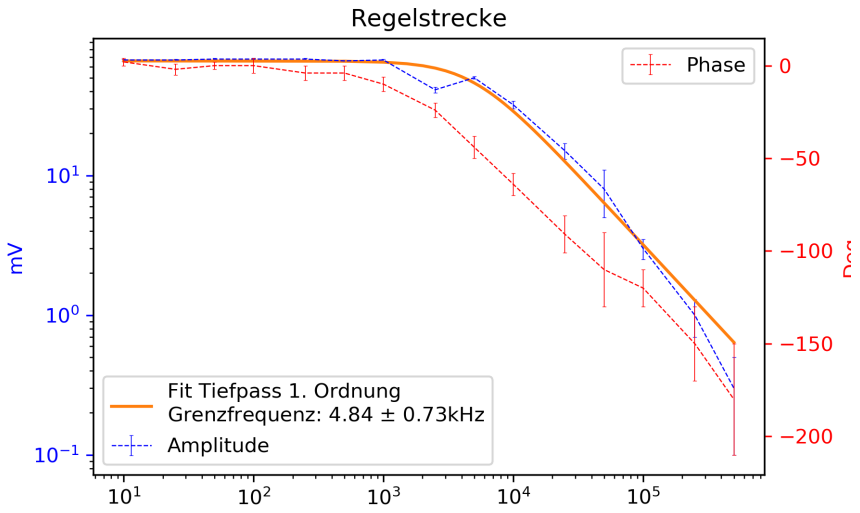

Abbildung 6.1.: Bode-Diagramm der Regelstrecke mit Fit eines Tiefpass erster Ordnung an die Amplitude. Die Grenzfrequenz beträgt dabei ungefähr 5 kHz. Die Datenpunkte wurden mit einem Funktionsgenerator und einem Oszilloskop aufgenommen.

Testaufbau

Um mit den selben Komponenten eine schnellere Regelung zu erhalten, ist es nötig, einen geringeren Verstärkungsfaktor der Photodiode zu verwenden. Die Photodiode hat nach Herstellerangaben [10] eine Bandbreite von 10 MHz bei 0 dB V. Da im Testaufbau der Laser durch den AOM direkt auf die Photodiode gelenkt werden kann, gibt es hier keine Limitierung der Bandbreite durch die Photodiode, und die Komponenten der Regelstrecke erlauben die Regelung mit einer Bandbreite von bis zu 100 kHz (siehe Anhang C.2). Auch hierbei können Diskretisierungsfehler der Regelung vernachlässigt werden.

6.1.2. Auswahl des Regelbereichs

Um eine PID-Regelung durchführen zu können, wird ein monotones und möglichst lineares Verhalten von $x(f(t))$ benötigt. Im Fall des Dysprosium-Experiments ist $x(f(t))$ weder monoton noch linear, wie man in 6.2 sieht. Deswegen wird die Stellgröße auf ein Intervall von $[0, 0.25]$ V beschränkt, in dem die Voraussetzungen erfüllt sind.
Eine zu hohe Laserleistung erzeugt in diesem Bereich ein positives Fehlersignal. Um die Laserleistung zu korrigieren, muss die Stellgröße y verringert werden, weil ein linearer Zusammenhang mit positiver Steigung zwischen Stellgröße und Regelwert besteht. Daraus folgt, dass die Regelparameter K_p, K_i, K_d negativ sein müssen. Wählt man

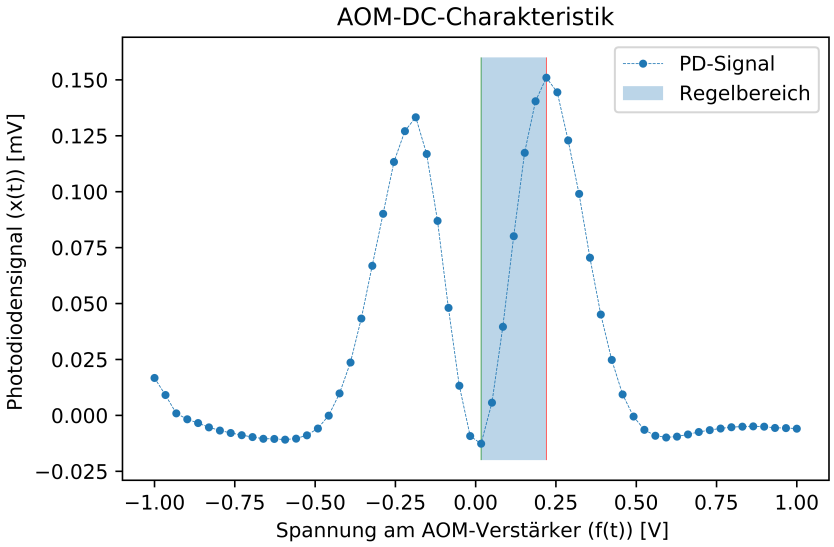

Abbildung 6.2.: Mit dem Red Pitaya gemessene DC-Antwort x(t) auf das Eingangssignal am Regelkreis f(t). Für f(t) wurde ein Gleichspannungssignal verwendet. Der Regelbereich wurde so gewählt, dass ein möglichst linearer Zusammenhang von der AOM-Spannung zur Photodiodenspannung besteht.

stattdessen einen Regelbereich mit negativer Steigung aus, müssen die Regelparameter positiv sein.

6.1.3. Optimale Stabilisierung

Ist der Regelbereich festgelegt, müssen nur noch die passende Regelparameter gefunden werden. Dabei werden nur die Parameter K_p und K_i genutzt. Es hat sich gezeigt, dass der D-Teil erst ab einer Frequenz von 300 kHz vom Rauschen, das er verursacht, zu unterscheiden ist. Auch die implementierten Filter konnten dieses Ergebnis nicht verbessern.
An der Teststrecke wurden verschiedene Verfahren getestet, mit denen Parameter für eine optimale Regelung gefunden werden können. Um ein besseres Verständnis vom Einfluss der Regelparameter auf die Optimierung der Regelung zu erhalten, wurde eine Rastersuche für verschiedene K_p und K_i durchgeführt und die Regelung mit der Güte Q gemäß 2.7 bewertet. Dazu war die Reglung aktiv, während eine Rechteckschwingung mit einer Amplitude von 200 mV und einer Frequenz von 1 kHz die Stellgröße y stört. In Abbildung 6.3 ist das Ergebnis dieser Suche dargestellt. Die optimale Reglung findet bei den Parametern $K_p = 0.53 \pm 0.09$ und $K_i = 3.9 \cdot 10^4\,\mathrm{s}^{-1}$ statt.
Das selbe Verfahren kann auch von Hand und „nach Augenmaß“durchgeführt werden. Dazu muss $x(t)$ nur auf einem Oszilloskop betrachtet und die Regelparameter solan-

ge geändert werden, bis die aufmodulierte Rechteckschwingung auf $x(t)$ nur noch als Spitzen auf dem Setzwert sichtbar sind, wie bei der optimalen Reglung in Abbildung 6.4 . Die nach Augenmaß ermittelten Werte decken sich mit denen der Rastersuche. Es wurden für eine optimale Reglung $K_i = 3.9 \cdot 10^4\,\mathrm{s}^{-1}$ gewählt, was dem maximal für K_i einstellbaren Parameter entspricht. Für $K_p = 5.20 \pm 0.16$ wurde gewählt. Diese Werte befinden sich im gelben Bereich der Gütelandschaft in Abbildung 6.3, was bedeutet, dass sie zu einer fast optimalen Regelung führen.
Außerdem wurde das Faustformelverfahren von Ziegler und Nichols aus Kapitel 2.3 verwendet, um optimale Regelparameter zu finden. Aus der Sprungantwort ergaben sich für $K_p = 0.477$ und $K_i = 288000\,s^{-1}$. Da K_i in PyRPL auf $3.9 \cdot 10^4\,\mathrm{s}^{-1}$ beschränkt ist, wird dieses Maximum verwendet.
Alle Verfahren zur Optimierung der Regelung geben ungefähr die selben Regelparameter, deren Q sich um maximal einen Faktor von zwei unterscheidet. Hinzu kommt, aufgrund eines nur näherungsweise linearen Zusammenhangs von x(t) und f(t), dass für unterschiedliche Setzwerte, unterschiedliche Regelparameter zu einer optimalen Regelung führen. Der selbe Effekt tritt auf, wenn die Eingangsleistung des Lasers mit der Zeit nachlässt und der AOM mehr Licht in den Strahl erste Ordnung geben muss, was effektiv zu einer Verschiebung des genutzten Regelbereichs führt. Aus diesen Gründen sollte es für viele Stabilisierungsaufgaben ausreichen, die optimalen Parameter über die Minimierung von Q nach Augenmaß zu bestimmen.
Die Parameter aus dem Faustformelverfahren deuten darauf hin, dass mit einem höheren K_i eine bessere Reglung möglich ist. Auch bei der Rastersuche scheinen Werte mit höheren K_i-Werten tendenziell zu einer besseren Reglung zu führen. Dies könnte entweder durch eine weitere Änderung im FPGA-Code erreicht werden, oder indem das Fehlersignal ähnlich wie bei 5.3.3 vor dem Regeln durch einen internen PID verstärkt wird, der das Fehlersignal $e(t)$ mit einem Faktor K_p multipliziert.

6.2. Intensitätsstabilität

Ausgerüstet mit optimalen Regelparametern kann die Regelung im Dysprosium-Experiment verwendet werden. Am Eindrucksvollsten zeigt sich der Effekt der Stabilisierung an einem direkten Vergleich vom unstabilisierten Lasersignal in Abbildung 6.5, mit dem Stabilisierten in Abbildung 6.6. Die Photodiodenspannung wird in beiden Abbildungen durch eine grüne Linie dargestellt. Als Maß zum Vergleich der Regelung wird die Standardabweichung der Photodiodenspannungen verwendet, da sie die Differenz der Spannung vom gewünschten Setzwert angibt. Die Stabilisierung verringert die Standardabweichung der Photodiodenspannung von 1,812% auf 0,016% in den Messungen vom 1.2.2019 und von 0,591% auf 0,016% in den Messungen vom 29.1.2019. Dieser Stabilität wird dabei durch das Rauschen der Photodiode begrenzt, welches sich im Bereich einiger mV befindet, wogegen die Regelung im Schnitt weniger als ein Zehntel mV vom Setzwert abweicht. Auch die Bitweite der ADCs und DACs könnte ein limitierender Faktor sein. Betrachtet man das Signal im Frequenzraum auf den Abbildungen C.6 und C.7, sieht man, dass Störungen des Signals bis zu einer

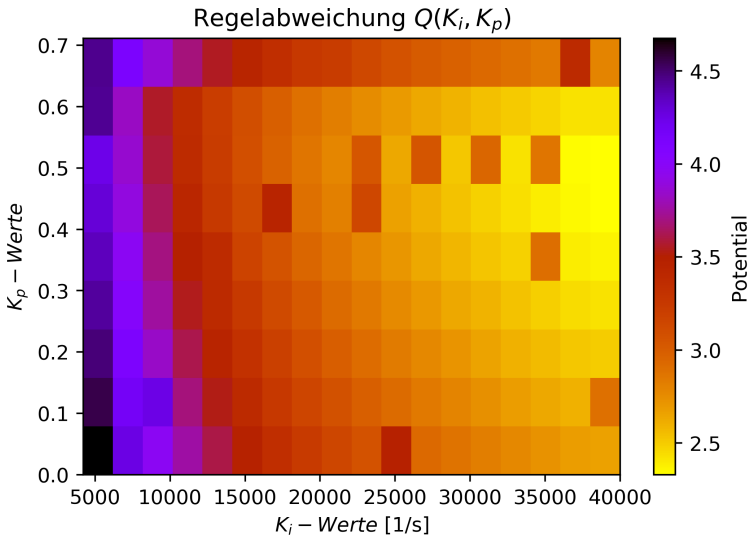

Abbildung 6.3.: Ausschnitt der Gütelandschaft für verschiedene K_p und K_i Parameter. Je niedriger das Potential, desto besser ist die Regelung. Die Gesamte Regellandschaft ist in C.4 zu sehen.

Größe von ca. 3 kHz herausgefiltert werden. Mit einem geringeren Verstärkungsfaktor an der Photodiode und damit einer größeren Bandbreite sollte auch die Rauschunterdrückung höherer Frequenzen möglich sein. Die Regelung ist also in der Lage, sowohl Langzeitdrifts, als auch Rauschen aus dem Signal herauszufiltern. Es ist wichtig, die Photodiode von anderen Lichteinflüssen abzuschirmen, da diese ansonsten auf den Laser aufmoduliert werden. Die Deckenbeleuchtung im Labor hat z. B. dazu geführt, dass die Regelung ein 100 Hz-Brummen auf den Laser moduliert.

6.3. Teilchenzahlstabilität

Die Motivation für diese Bachelorarbeit ist die Stabilisierung der Teilchenzahl in der MOT des Dysprosium-Experiments. Die Laborerfahrung der Gruppe hat gezeigt, dass die Leistung des Zeemanslower-Lasers einen großen Einfluss auf die Teilchenzahl in der MOT hat. Dieser Zusammenhang soll in diesem Abschnitt näher untersucht werden. Anschließend wird betrachtet, inwiefern die Intensitätsstabilisierung die Teilchenzahl in der MOT beeinflusst und ob sie zu einer Stabilisierung dieser beitragen kann. Abschließend werden weitere Quellen, die zur Destabilisierung der Teilchenzahl beitragen, analysiert und auf Korrelationen hin untersucht.

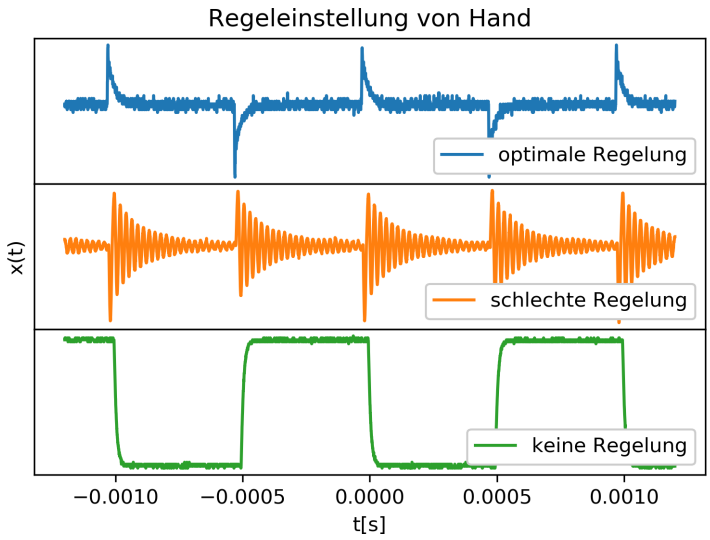

Abbildung 6.4.: Zur manuellen Optimierung wird das System mit einer kleinen Rechteckschwingung gestört. Bei guten Regelparametern sind nur noch Spannungsspitzen zu sehen (blau). Bei schlecht gewählten Regelparametern beginnt das System zu schwingen (orange).

6.3.1. Korrelation zwischen Photodiodenspannung und Teilchenzahl

In diesem Abschnitt wird die Korrelation zwischen der Photodiodenspannung am Zeemanslower und der Teilchenzahl untersucht. Bei einem eindeutigen Zusammenhang der beiden Größen sollte eine Stabilisierung der Photodiodenspannung auch zu einer Stabilisierung der Teilchenzahl führen. Um die Korrelation zu messen, werden mehrere Messsequenzen im Dysprosium-Experiment, mit einer aktiven Stabilisierung auf eine Reihe unterschiedlicher Setzwerte durchgeführt.
Es ist eine eindeutige Korrelation zwischen den Setzwerten und der Teilchenzahl in Abbildung 6.7 und Abbildung 6.8 zu erkennen. Die Teilchenzahl steigt mit der Laserleistung bis zu einem Maximum bei $s \approx 0.26\,\mathrm{V}$ an und fällt anschließend asymptotisch zu einer mittleren Teilchenzahl. Daraus lassen sich wichtige Erkenntnisse für den experimentellen Betrieb gewinnen. Aus der Lage des Maximums kann man auf Setzwerte schließen, welche die Teilchenzahl in der MOT optimieren. Außerdem kann aus der Korrelation geschlossen werden, dass für den nicht stabilisierten Fall, in dem die Laserleistung mit der Zeit abfällt, diese zu Beginn etwas oberhalb des für die Stabilisierung optimalen Setzwerts eingestellt werden sollte. Dadurch wird das Teilchenzahlplateau hinter dem Maximum genutzt und der steile Bereich unterhalb des Maximums gemieden.

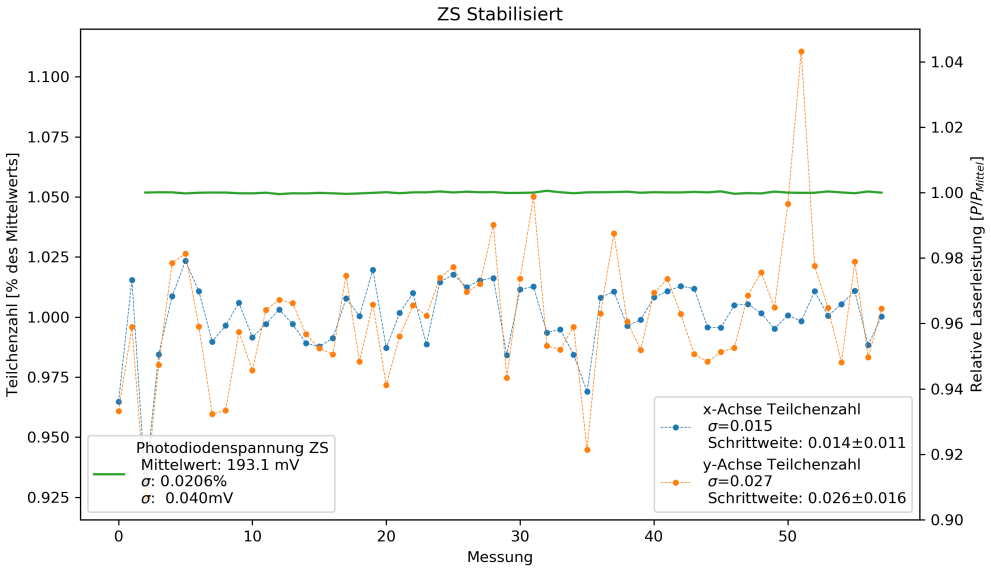

Abbildung 6.5.: Messung der Teilchenzahl in der MOT bei stabilisierter Zeemanslower-Leistung. Schwankungen in der Photodiodenspannung am Zeemanslower sind kaum zu erkennen.

6.3.2. Stabilität der Teilchenzahl

Während die Intensitätsstabilisierung wie erwartet funktioniert hat, stellte sich die Stabilisierung der Teilchenzahl als etwas problematischer heraus. Beim ersten Einsatz der Intensitätsstabilisierung am Zeemanslower fiel die Teilchenzahl bei aktiver Regelung der Intensität rapide ab, wie man in Abbildung C.5 sehen kann. Ohne die Regelung der Laserleistung blieb die Teilchenzahl relativ stabil. Bei genauerer Betrachtung des Regelkreises stellte sich heraus, dass das Licht mit einer undefinierten Polarisation in den AOM eintrat. Deshalb wurde der Strahlengang verändert, sodass der Laserstrahl mit linearer Polarisation in den AOM läuft. Danach trat dieser Effekt nicht mehr auf. Vermutlich lässt sich dieser Effekt mit einer Polarisationsänderung durch den AOM, zusammen mit der Polarisationsabhängigkeit des transmittierten Lichts, auf welches das System stabilisiert wurde, erklären.
Im Dysprosium-Experiment werden, zur Bestimmung der Teilchenzahl in der MOT, Bilder aufgenommen, an die automatisch eine Gauß-Kurve auf der x-Achse und eine Gauß-Kurve auf der y-Achse angepasst wird. Daraus bestimmt das Auswertungssystem automatisch die Teilchenzahl in der MOT. Der Effekt der Regelung auf die Stabilität kann in zwei Größen gemessen werden. Es kann zum Einen die Standardabweichung der Teilchenzahlen betrachtet werden. Solange die Teilchenzahlen um einen konstanten Mittelwert herum streuen, ist dies ein gutes Maß für die Stabilität des

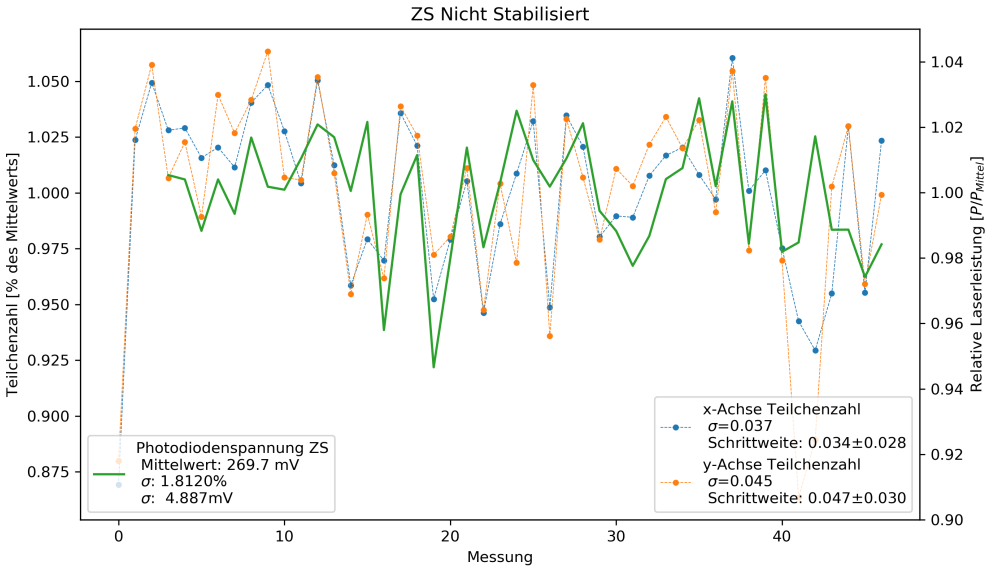

Abbildung 6.6.: Messung der Teilchenzahl in der MOT ohne Stabilisierung der Zeemanslower-Leistung. Es sind deutliche Schwankungen in der Photodiodenspannung zu sehen. Die y-Achse der Laserleistung ist genauso skaliert wie in 6.5, um einen direkten Vergleich zu ermöglichen.

Systems. Bei längeren Messungen über mehr als eine halbe Stunde wird es jedoch problematisch, diese Größe anzuwenden, weil der Mittelwert der Teilchenzahl, aufgrund äußerer Einflüsse, zu wandern beginnt. Deswegen wird als zweite Größe die absolute Änderung der Teilchenzahl von Messung zu Messung betrachtet und der Mittelwert daraus gebildet. Ändert sich der Wert, um den die Teilchenzahl streut nur um wenige Schrittweiten, so sollte diese Größe eine Aussage über die Stabilität der Messung liefern können. Die Teilchenzahlen in der MOT variieren stark, von einem Tag auf den Anderen. Deswegen werden alle Teilchenzahlen auf die mittlere Teilchenzahl der Messung normiert.
In den Abbildungen 6.5 und 6.6 sind Messungen der Teilchenzahl mit und ohne Stabilisierungen geplottet. Die Ergebnisse aller Messungen, sind in Tabelle B.4, zusammengefasst. In den Messungen vom 29.1.2019 verringert sich die Standardabweichung der Teilchenzahlen in x-Richtung um einen Faktor von 2,85 und in y-Richtung um einen Faktor von 2,14. Bei der mittleren Schrittweite sind ähnliche Verbesserungen zu erkennen. Die Stabilisierung der Laserintensität stabilisiert also auch die Teilchenzahl.
In den Abbildungen 6.9 und 6.10 ist die Verteilungen der Schrittweiten in y-Richtung mit und ohne Stabilisierung dargestellt. Beide Verteilungen werden als normalverteilt betrachtet und es wird eine Gauß-Verteilungsdichte an diese angepasst. Der Parameter

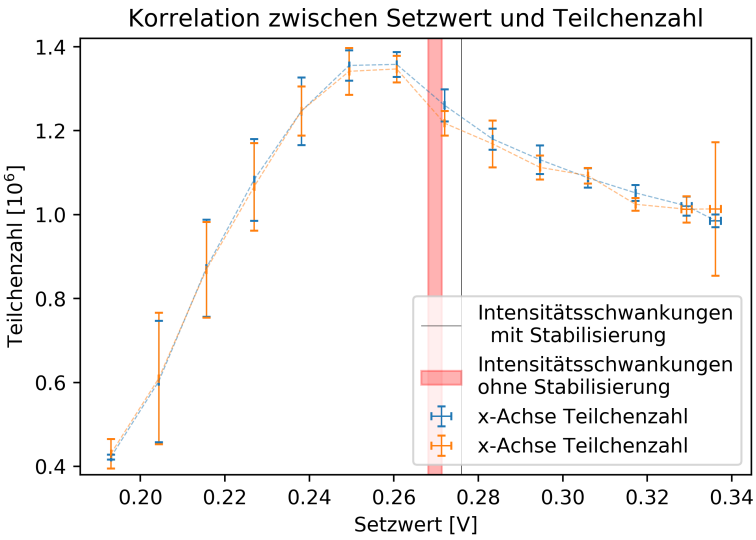

Abbildung 6.7.: Bei aktiver Stabilisierung wird der Setzwert der Regelung in konstanten Schritten erhöht. Die Teilchenzahl erhöht sich dabei nahezu linear einem Maximum und fällt danach asymptotisch ab.

μ beschreibt dabei, wie weit die Teilchenzahl sich pro Messung von einem Mittelwert verschiebt. Sowohl im stabilisierten Fall, als auch im nicht stabilisierten Fall sind diese Verschiebungen negativ. Dies stimmt mit der Beobachtung überein, dass sich die Teilchenzahl mit der Zeit verringert. Bei der Messung mit Stabilisierung ist diese Verschiebung 3,38±0,21 mal kleiner, als bei der Messung ohne Stabilisierung. Der Parameter σ beschreibe die Standardabweichung der Verteilungsfunktion. Sie ist ein Maß für die Streuung der Schrittweiten. Bei eingeschalteter Stabilisierung verringert sich σ auf der y-Achse um einen Faktor von 1,66±0,046. Damit liegt die Verbesserung fast im Bereich der am 29.1.2010 gemessenen Verbesserung der Standardabweichung. Währen auf der y-Achse eine klare Verbesserung der Teilchenzahl zu erkennen ist, wenn die Regelung eingeschaltet wird, bleibt die x-Achse davon unbeeinflusst. Sowohl μ, als auch σ, verändern sich nicht signifikant auf der x-Achse. Es ist möglich, dass die Methode der Teilchenzahlmessung je nach Ausrichtung der MOT mal mehr und mal weniger sensitiv auf Schwankungen in der Teilchenzahl ist. Auch die Wahl des Bildbereichs, der für das Anpassen einer Gauß-Kurve verwendet wird, hat einen großen Einfluss darauf, wie gut die Messungen auf den beiden Achsen sind.
Betrachtet man die in Abbildung 6.7 dargestellten Schwankungen der Laserleistungen und deren Korrelation mit der Teilchenzahl, würde man erwarten, dass sich die Schwankungen der Teilchenzahl, beim eingeschalteter Stabilisierung, auf einem viel geringeren Niveau befinden. Für die Diskrepanz zwischen der erwarteten und gemessenen Teilchenzahlstabilität werden verschiedene Erklärungsmöglichkeiten diskutiert. Es ist

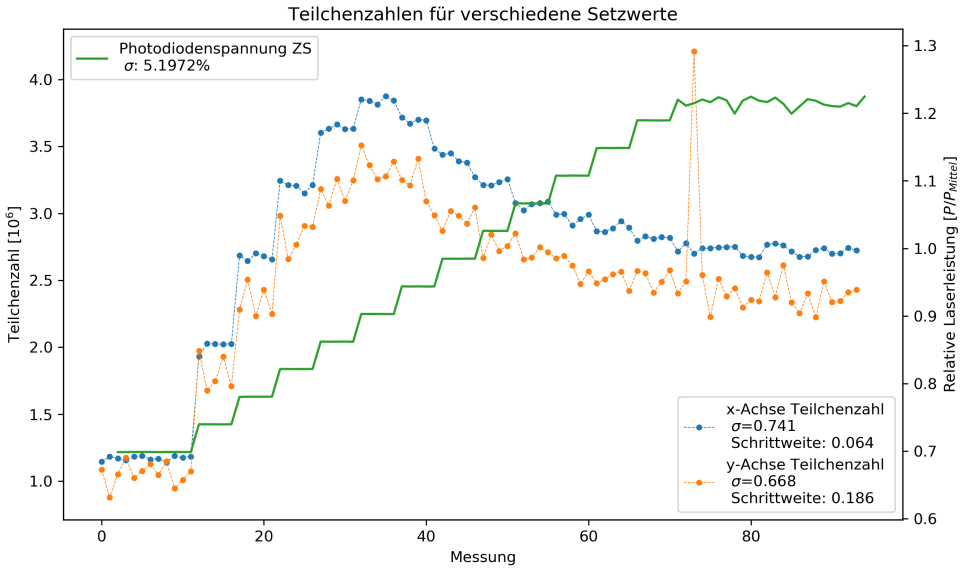

Abbildung 6.8.: Bei aktiver Stabilisierung wird der Setzwert der Regelung in konstanten Schritten erhöht. Ab Messung 72 ist der Setzwert höher als die zur Verfügung stehende Laserintensität. Die Regelung ist deshalb nicht mehr in der Lage die Intensität zu stabilisieren.

denkbar, dass die Stabilisierung die Laserleistung nur auf Zeiträume von mehreren Sekunden funktioniert, jedoch einen destabilisierenden Effekt auf kürzeren Zeitskalen hat. Dagegen spricht jedoch die in Abschnitt 6.2 besprochene Verringerung des Rauschens von Frequenzen bis zu 3 kHz. Ein destabilisierender Effekt auf kurzen Zeitskalen bis 5 kHz ist also nicht erkennbar.
Die zweite mögliche Erklärung beruht auf dem Problem, dass nur das durch einen Spiegel transmittierte Licht auf die Photodiode trifft, anstatt eines wohl definierten Laserstrahls. Der Zusammenhang zwischen der Photodiodenspannung und der Laserintensität am Zeemanslower ist nicht bekannt und es ist möglich, dass die Laserleistung schwankt, obwohl die Photodiodenspannung auf dem selben Setzwert ist. Dieses Problem scheint jedoch durch die Änderungen am Strahlengang, die zu Beginn des Kapitels beschrieben wurden, gelöst zu sein.
Die wahrscheinlichste Erklärung ist, dass der Einfluss der schwankenden Laserleistung auf die Teilchenzahl effektiv ausgeschaltet ist und die Schwankungen aus anderen Quellen stammen.

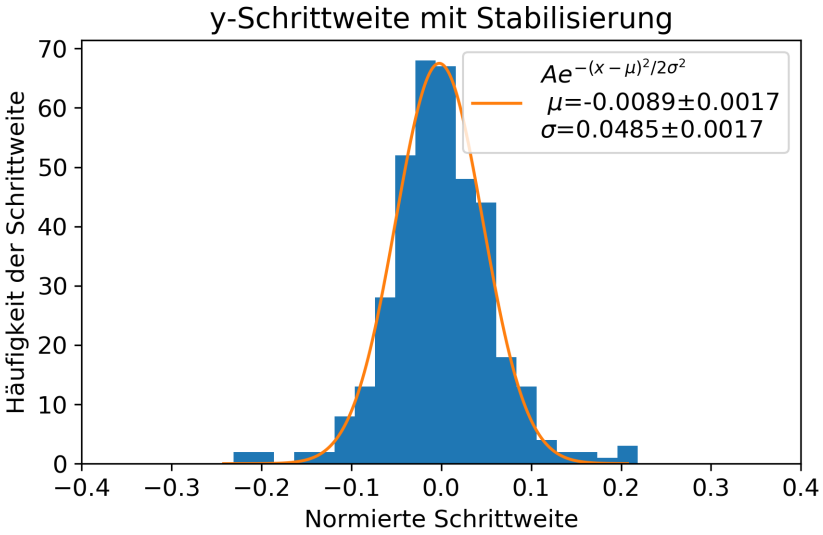

Abbildung 6.9.: Histogramm der in x-Richtung gemessenen Schrittweiten der Teilchenzahl mit Intensitätsstabilisierung. Es wurde angenommen, dass die Schrittweiten normalverteilt sind. μ beschreibt den Langzeitdrift der Teilchenzahl. Da sich die Schritte im Schnitt weiter in negativer Richtung bewegen, als in Positiver wird auch hier der Langzeitdrift der Teilchenzahl erkennbar. Der Parameter σ beschreibt die Streuung der Schrittweite und damit auch die der Teilchenzahl, von Messung zu Messung. Bei eingeschalteter Stabilisierung verringert sich sowohl der Langzeitdrift als auch die Streuung (siehe 6.10).

Andere Störquellen

Nachdem die Laserleistung im Zeemanslower stabilisiert ist, werden die beiden anderen Lasersysteme, das transversale Kühlsystem (TC) und das 626 nm-MOT-Lasersystem (626) untersucht. Dazu werden die Leistungen der Lasersysteme an Photodioden gemessen, während die Regelung für den Zeemanslower aktiv ist. Um Korrelationen zu finden wurde eine Messreihe mit über 350 Messsequenzen aufgenommen, was etwa 45 Minuten entspricht. Betrachtet man die Teilchenzahlen und Photodiodenspannungen im zeitlichen Verlauf auf Abbildung 6.11, wird sichtbar, dass sowohl die Teilchenzahl, als auch die Intensitäten der beiden anderen Lasersysteme mit der Zeit abfallen. Außerdem fällt auf, dass die Leistung des TC-Lasers eine periodische Schwingung durchführt und beide Laserleistungen kontinuierlich abnehmen. Betrachtet man die zeitunabhängigen Korrelationen von Teilchenzahl und Laserleistung in Abbildung C.8 und C.9, sieht man einen linearen Zusammenhang von Teilchenzahl und Photodiodenspannung. Die Stabilisierung dieser beiden Lasersysteme, zusätzlich zu der des Zeemanslower-Systems, sollte die Stabilität der Teilchenzahl noch weiter verbessern.

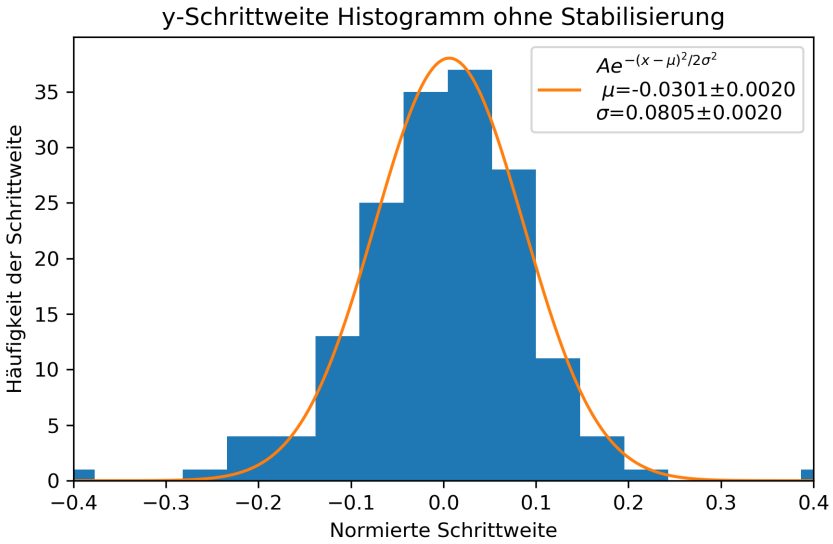

Abbildung 6.10.: Histogramm der in x-Richtung gemessenen Schrittweiten der Teilchenzahl ohne Intensitätsstabilisierung. Es wurde angenommen, dass die Schrittweiten normalverteilt sind. μ beschreibt den Langzeitdrift der Teilchenzahl .Da sich die Schritte im Schnitt weiter in negativer Richtung bewegen, als in Positiver wird auch hier der Langzeitdrift der Teilchenzahl erkennbar. Der Parameter σ beschreibt die Streuung der Schrittweite und damit auch die der Teilchenzahl, von Messung zu Messung. Bei eingeschalteter Stabilisierung verringert sich sowohl der Langzeitdrift als auch die Streuung (siehe 6.9).

Ohne die Intensitätsstabilisierung des Zeemanslowers waren diese Korrelationen nicht sichtbar.
Genau wie die Laser, sind auch andere Teile des Dysprosium-Experiments mögliche Ursachen für die schwankende Teilchenzahl. Spiegel ändern ihre Ausrichtung aufgrund von Temperaturschwankungen im Labor, weshalb die Laser nicht mehr optimal ausgerichtet sind. Die Frequenzstabilisierung der Lasersysteme kommt im Verlauf eines Tages mehrfach an die Grenzen ihres Regelbereichs, wodurch eine Frequenzstabilisierung nur noch teilweise möglich ist. Auch die Magnetfelder des Zeemanslowers und der MOT sind nicht aktiv stabilisiert und Schwankungen unterworfen. Die Anzahl der möglichen Störeinflüsse macht es sehr schwer den Einfluss, der jeweiligen Störquelle auf das Experiment, abzuschätzen. Durch das systematische Ausschalten der Störquellen kann die Stabilität des gesamten Experiments und damit der Teilchenzahl noch weiter erhöht werden. Die Stabilisierung aller vorhandenen Lasersysteme ist dafür ein wichtiger Schritt.

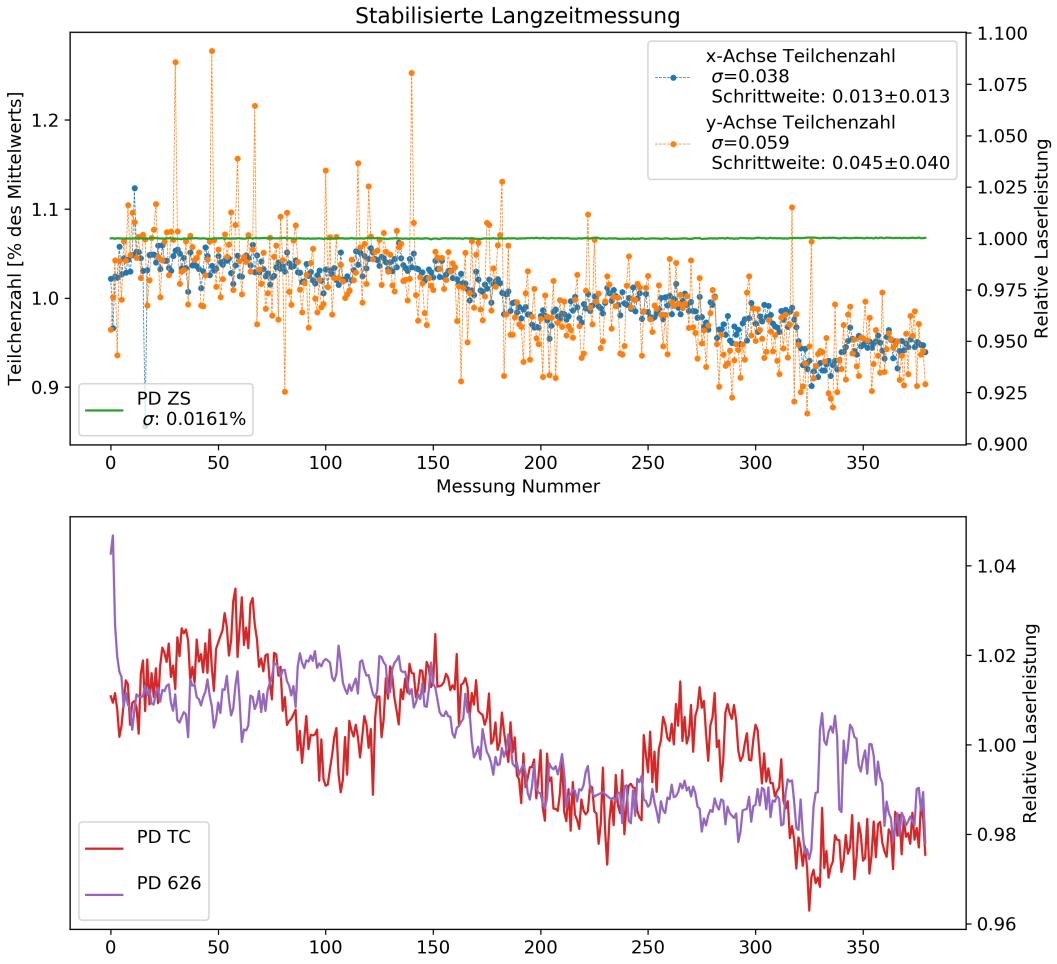

Abbildung 6.11.: Langzeitmessung der Teilchenzahl mit Intensitätsstabilisierung. Zusätzlich zur Laserintensität am Zeemanslower wurden die Laserintensitäten am transversalen Kühlen (TC) und am 626 nm-MOT-Lasersystem (626) gemessen, um weitere Korrelationen zwischen Laserintensitäten und Teilchenzahlen zu finden. Die Schwingungen des TC-Lasersystems scheinen sich auf die Teilchenzahl zu übertragen.

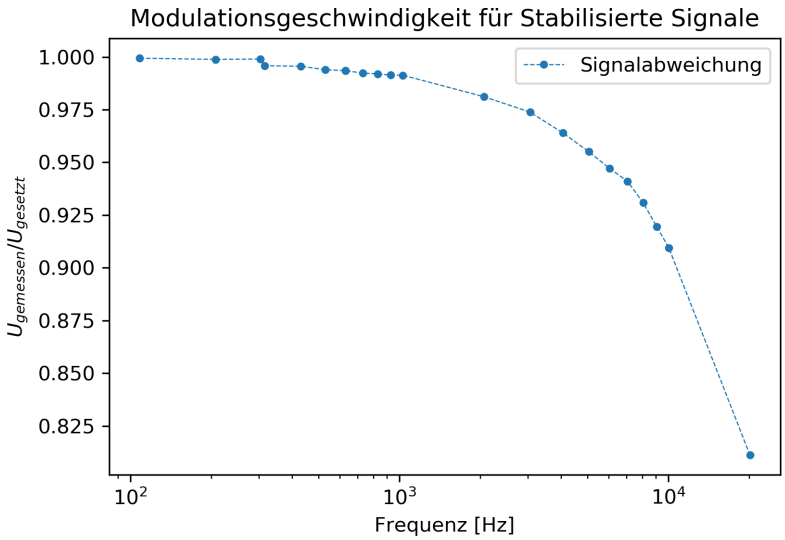

Abbildung 6.12.: Eine Dreieckspannung wird dem Red Pitaya als Setzwert übergeben. Der Plot zeigt, mit welcher Abweichung der Regelkreis dem Setzwert folgt, abhängig von der Frequenz des Setzwertsignals. Die Messung wurde am Testaufbau durchgeführt.

6.4. Veränderter Setzwert

Im Dysprosium-Experiment sollen Laserintensitäten nicht nur stabilisiert, sondern auch moduliert werden. Dass es möglich ist, mit dem Red Pitaya solche Modulationen stabilisiert durchzuführen, wurde bereits in 5.3.2 beschrieben. Hierbei versucht der Red Pitaya $x(t)$ auf einen variablen Setzwert $s(t)$ zu regeln. Abbildung 6.12 zeigt eine Untergrenze für die Geschwindigkeit solcher Modulationen. Modulationen über eine Bandbreite bis zu einem kHz, können nahezu ohne Abweichungen vom Setzwert durchgeführt werden und selbst bei einer Frequenz von 10 kHz beträgt die Abweichung weniger als 10%.

6.5. Verwendung im Laboralltag

Für die Verwendung der Stabilisierung im Labor wurde ein Jupyter-Notebook entworfen, das in Abbildung 6.13 zu sehen ist. Es sind Schieberegler vorhanden, mit denen K_p, K_i und der Setzwert eingestellt werden können. Der Schalter „New Setpoint “misst den aktuellen Ausgangswert x des Regelkreises und setzt diesen als neuen Setzwert für die Regelung ein. Die Schalter „Start Manual Optimi... “und „Stop Manual Optimi... “aktivieren oder deaktivieren eine Rechteckschwingung mit einer Amplitude von 50 mV und einer Frequenz von 1 kHz. Diese wird zusätzlich zur Stellgröße auf den

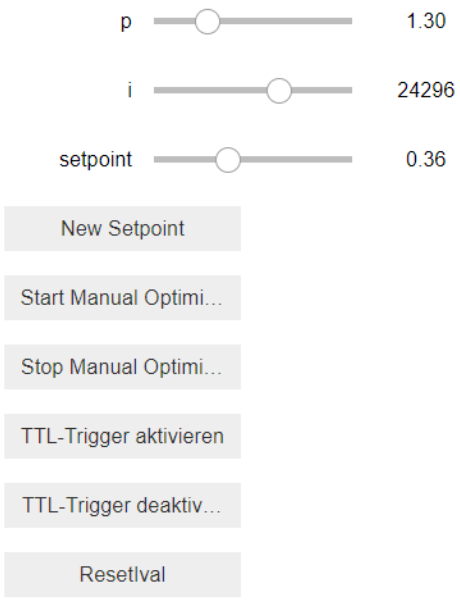

Abbildung 6.13.: Für die Laborarbeit entworfene Benutzeroberfläche. An den Schiebereglern können die Regelparameter und der Setzwert ausgewählt werden.

Regelkreis gegeben. Wie in Abschnitt 6.1.3 beschrieben, muss die Regelung optimiert werden. Anschließend sollte der Funktionsgenerator natürlich wieder ausgeschaltet werden. Die Knöpfe „TTL-Trigger aktivieren “und „TTL-Trigger deaktiv... “, legen fest, ob der Red Pitaya auf ein externes Trigger-Signal reagiert oder nicht. „ResetIval “setzt die Integralsumme auf Null. Alle Einstellungen werden in einer Konfigurations-Datei gespeichert, sodass die Regelung nicht jedes mal neu konfiguriert werden muss, wenn das Experiment z. B. nach einem Stromausfall neu eingeschaltet wird.

7. Fazit

Das Ziel dieser Arbeit war der Aufbau einer Laserintensitätsstabilisierung am Dysprosium-Experiment, um damit die Teilchenzahl in der MOT zu stabilisieren. Der Aufbau und die Optimierung der Stabilisierung konnten erfolgreich durchgeführt werden. Die Stabilisierung wurde über einen Zeitraum von mehr als 45 Minuten am Stück getestet und war dabei in der Lage, Schwankungen der Leistung von über 0,5% auf 0,016% zu verringern und Rauschen bis zu einer Frequenz von 3 kHz aus dem Laserstrahl zu filtern. Für die Implementierung in das Dysprosium-Experiment wurde die FPGA- und Python-Software von PyRPL um einen TTL-Trigger im PID-Modul erweitert. Damit lassen sich Schaltzeiten von einigen hundert Nanosekunden erreichen. Mit dieser Regelung war es möglich, die Stabilität der Teilchenzahl im Dysprosium-Experiment um einen Faktor von 2,85 auf der x-Achse und um einen Faktor von 2,14 auf der y-Achse, zu verbessern. Des Weiteren konnten Anhaltspunkte gesammelt werden, die darauf hindeuten, dass eine Intensitätsstabilisierung des TC- und 626-Lasersystems zu einer weiteren Stabilisierung der Teilchenzahl führt. Mit der entwickelten Laserintensitätsstabilisierung sollte es möglich sein, alle Lasersignale, deren Leistung elektronisch gesteuert werden kann, zu stabilisieren. Dazu ist möglich, die Software zu erweitern, sodass über das Python-API für jeden PID ein GPIO-Stift als Trigger-Eingang ausgewählt werden kann. Mit dieser Erweiterung können zwei Lasersysteme mit dem selben Red Pitaya stabilisiert werden.
Für die Anwendung der Regelung im Labor wurde eine Benutzeroberfläche entworfen, die unter anderem eine einfache Optimierung der Regelparameter ermöglicht. Anstatt die Regelparameter von Hand zu suchen, kann dazu ein Algorithmus verwendet werden. Eine Möglichkeit dafür stellt die in dieser Arbeit vorgestellte Suche über eine Gütelandschaft dar. Es können aber auch andere Optimierungsverfahren über das Python-API implementiert werden. Als langfristiges Ziel könnte eine Stand-Alone-Version der Stabilisierung mit dem Red Pitaya entworfen werden, die keinen Client-PC benötigt und sich automatisch für einen gegebenen Regelkreis optimiert, sodass nur noch ein Setzwert eingestellt werden muss. Kurzfristig können alle Lasersysteme im Dysprosium-Experiment mit einem Red Pitaya stabilisiert werden.

A. Anhang

A.1. Inbetriebnahme

Zur Inbetriebnahme des Red Pitaya müssen Nutzer von Windows 7/8 „Bonjour Print Services“installieren um Zugang zur Netzwerkadresse *.local zu erhalten. Nutzer von Windows 10 sollten schon Zugang dazu haben [9]. Nach dem Einschieben der mitgelieferten micro-SD-Karte sollte der Nutzer in der Lage sein, den RedPitaya mit einem Lan-Kabel an einen Rechner anzuschließen und mit einem Webbrowser über die Adresse „rp-xxxxxx-local
grqq auf den Red Pitaya zuzugreifen. In den meisten Universitätsnetzwerken ist es notwendig, den Red Pitaya vorher im internen Netzwerk freischalten zu lassen. Dabei wird ihm eine eigene IP zugewiesen, über die der Nutzer den Red Pitaya von jedem Rechner im Netzwerk ansteuern kann. Da bei der aktuellen Version jeder Rechner im Netzwerk auf den Red Pitaya zugreifen kann, und zumindest über das Webinterface auch ohne SSH-Passwort auf die Ein- und Ausgänge des Red Pitaya zugreifen kann ist es wichtig, entweder keine sensiblen Teile eines Experiments damit zu verbinden, oder die Red Pitaya nur über ein gesichertes Lokales Netzwerk zu verwenden. Weitere Informationen sind in der Red Pitaya Herstellerdokumentation [9] zu finden.

A.2. Installation von PyRPL

Um PyRPL zu verwenden, müssen die folgenden Python-Pakete über die Kommandozeile oder die Anaconda Konsole ausgeführt werden:

- „pip install pyrpl“
- „pip install quamash“

Alle weiteren für die Verwendung von PyRPL benötigten Pakete sollten automatisch von „pip“installiert werden. Um die in dieser Bachelorarbeit erweiterten Funktionen zu nutzen, muss der Ordner pyrpl im PATH-Verzeichnis durch den pyrpl-Ordner aus dem /Laserstab Ordner[1] ersetzt werden. Der Pfad zum PATH-Verzeichnis kann in den Windows Umgebungsvariablen eingesehen werden. Einfacher ist es jedoch, ein beliebiges Python-Paket zu importieren, beispielsweise numpy, und anschließend die folgende Zeile auszuführen:
print(numpy.__file__.replace(”\\”,”/”))
Dieser Dateipfad bis ../lib/site-packages ist der Pfad zum PATH-Verzeichnis in dem der pyrpl-Ordner ersetzt werden muss.

[1] Im Gruppenverzeichnis

A.3. Software zur FPGA-Bearbeitung

Um das FPGA-Design zu verändern, ist es nötig, eine Entwicklungsumgebung zu nutzen, die in der Lage ist, den Verilog-Code in eine für den FPGA lesbare Binärdatei zu kompilieren. Dazu muss „Vivado 2015.4“[2] von der Xilinx-Website Website heruntergeladen und eine Lizenz beantragt werden. Eine genaue Anleitung findet sich in [11] unter „1.5.2 Building the FPGA firmware“. Nach erfolgreicher Installation und dem Erhalten einer kostenfreien „HL WebPACK 2015 and Earlier License“, können FPGA-Bitfiles und Binärdateien erstellt werden. Die aktuelle Version vom PyRPL-Projekt mit FPGA-Quellcode und weiteren Programmen, die zur Erstellung einer Binärdatei für den FPGA benötigt werden, kann unter `https://github.com/lneuhaus/pyrpl` heruntergeladen werden. Dort befindet sich in ihrem_pyrpl_Verzeichnis/pyrpl/fpga (pyrpl/pyrpl/fpga) die Datei „make.bat“(Windows) welche den FPGA-Code kompiliert und zwischen 10 und 30 Minuten dafür benötigt.[11] Wird dieser Prozess erfolgreich abgeschlossen sollte im Verzeichnis der „make.bat“die Datei „red_pitaya.bin“, durch eine neuere Version ersetzt worden sein. Der Quellcode für die FPGA-Module befindet sich im Ordner pyrpl/pyrpl/fpga/rtl. Die Dateien können mit einem beliebigen Texteditor, möglichst mit Syntaxhervorhebung, bearbeitet werden. Treten Fehler bei der Kompilierung des Verilog-Codes auf, kann es hilfreich sein, ein Projekt in Vivado zu erstellen und darüber eine FPGA-Bitfile zu erstellen, da Fehlermeldungen beim Kompilieren hier mit Zeilenangabe und Fehlergrund gemeldet werden. Dazu muss in der Vivado-Kommandozeile in pyrpl/fpga Ordner navigiert und der Befehl „source red_pitaya_vivado_project.tcl“ausgeführt werden.

[2] Andere Versionen funktionieren nicht

B. Tabellen

Prozessor	dual core ARM A9
FPGA	FPGA Xilinx Zynq 7010 SOC Xilinx Zynq 7010 SOC
Abtastrate	125 MS/s
ADC Auflösung	14bit
Messbereiche	[-1,1] V , [-20,20] V (Durch umstecken von Steckbrücken wählbar)
Ausgangsspannungsbereich	[-1,1] V
Eingangsimpedanz	1 MΩ/10 pF
Lastimpedanz Ausgang	50 Ω

Tabelle B.1.: Technische Daten des Red Pitaya. Die komplette Liste kann `https://www.redpitaya.com/f130/STEMlab-board` entnommen werden.

Regler	K_p	K_i	K_d
P	$\frac{T_g}{KT_u}$		
PI	$\frac{0.9T_g}{KT_u}$	$\frac{.9T_g/K}{3.33{T_u}^2}$	
PID	$\frac{1.2T_g}{KT_u}$	$\frac{1.2T_g/K}{2{T_u}^2}$	$\frac{0.6T_g}{K}$

Tabelle B.2.: Regelparameter nach Ziegler und Nichols für P, PI und PID Regler [6].

Variable	Funktion	Mögliche Werte abfragen
input	Eingangssignal für ein PID-Objekt aus	in1 ,in2, out1, out2, pid0, pid1, ...
output_direct	Ausgäng, an die das PID Objekt den Stellwert sendet	off, out1, out2, both
setpoint	Setzwert, der vom Regelsignal abgezogen wird	[-1,1]
p	K_p	[-2048,2048]
i	K_i	$[-3.9 \cdot 10^{-4}, 3.9 \cdot 10^{-4}]$
d	K_d	$[-4.1 \cdot 10^{\cdot 5}, -4.1 \cdot 10^{\cdot 5}]$
inputfilter	Erste Ordnungsfilter auf das Eingangssignal.	$[-2^{25}, 2^{25}]$ (nur 0 und 2^n Werte möglich)
max_voltage	Maximal vom PID ausgegebene Spannung in V	[-1,1]
min_voltage	Minimal vom PID ausgegebene Spannung in V	[-1,1]
ival	Wert der Integralsumme	[-4,4]

Tabelle B.3.: Alle Variablen sind in einem PID Objekt (pid0,pid1,pid2) vorhanden und müssen auch als soche über pidX.Variable angesprochen werden. Die Variable pidX.setup_attributes ist ein Dictionary, das angibt, wie die Variablen gesetzt sind. Die Variable pidX.Variable_options ist eine Liste der möglichen Setzwerte für Variablen angibt.

Datum	Stabilisiert	σ-x	Mittlere x-Schrittw.	σ-y	Mittlere y-Schrittw.
1.2.2019	Nein	0,015±0,012	0,015	0,079	0,077±0,079
1.2.2019	Ja	(0,038)	0,013±0,015	(0,059)	0,045±0,041
29.1.2019	Ja	0,015	0,014±0,011	0,027	0,026±0,016
29.1.2019	Ja	0,014	0,010±0,006	0,028	0,031±0,019
29.1.2019	Nein	0,037	0,034±0,028	0,045	0,047±0,030
29.1.2019	Teilweise	0,021	0,017±0,011	0,030	0,030±0,021
29.1.2019	Ja	0,010	0,008±0,005	0,019	0,023±0,014
23.1.2019	Ja	0,155	0,023±0,049	0,159	0,032±0,052

Tabelle B.4.: Teilchenzahlschwankungen mit und ohne aktive Laserstabilisierung. Eingeklammerte Werte sind unbrauchbar, da sich der Mittelwert der Teilchenzahl über den Verlauf der Messung verschiebt. Bei der teilweisen Stabilisierung war die verfügbare Laserintensität zu gering, sodass es zu Einbrüchen in der Intensität am Zeemanslower kommt.

C. Abbildungen

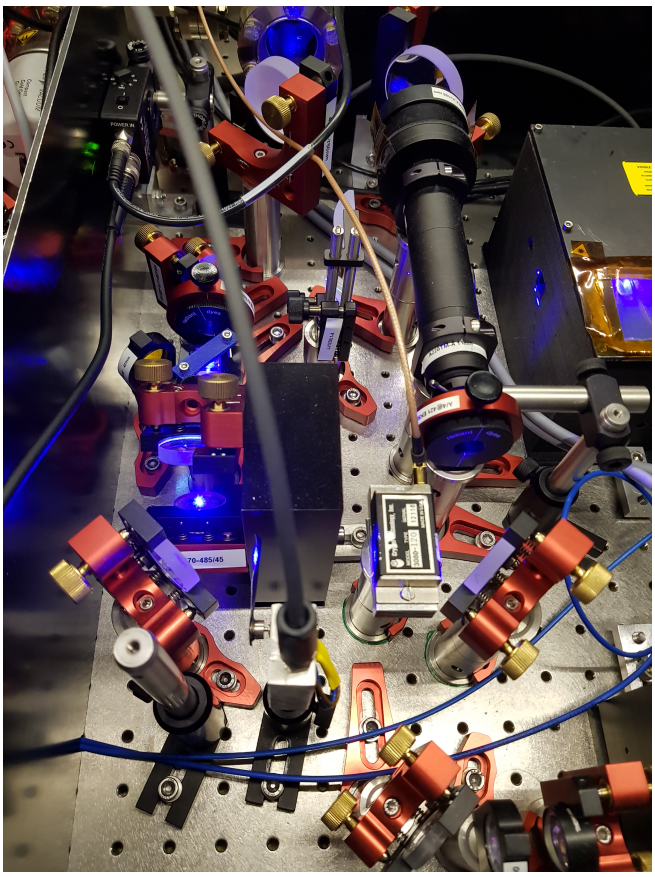

Abbildung C.1.: Bild des Dysprosium Regelkreises.

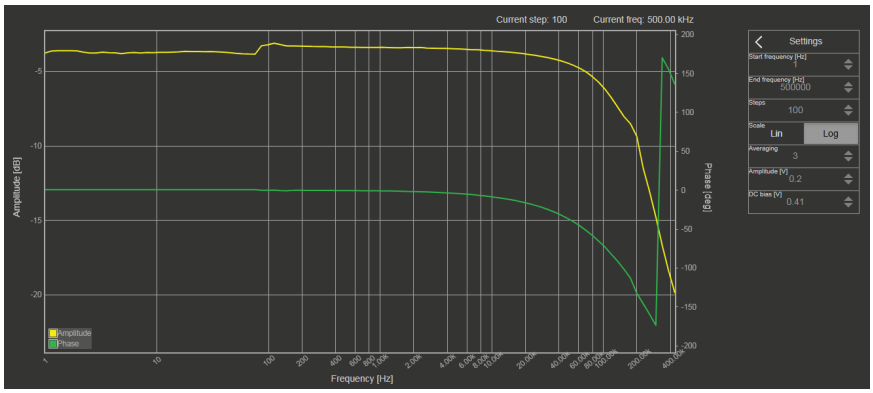

Abbildung C.2.: Bode-Diagramm der Regelstrecke des Testaufbaus. Aufgenommen mit einem 14-Bit Red Pitaya.

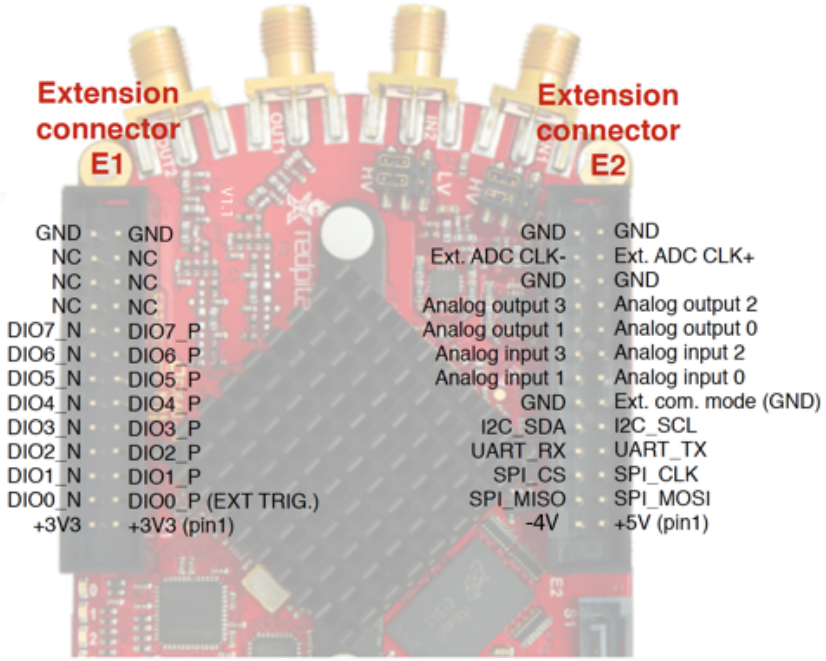

Abbildung C.3.: Exemplarisches FPGA-Design mit Digitalem Signal Prozessor und Multiplexer. Entnommen von: [9]

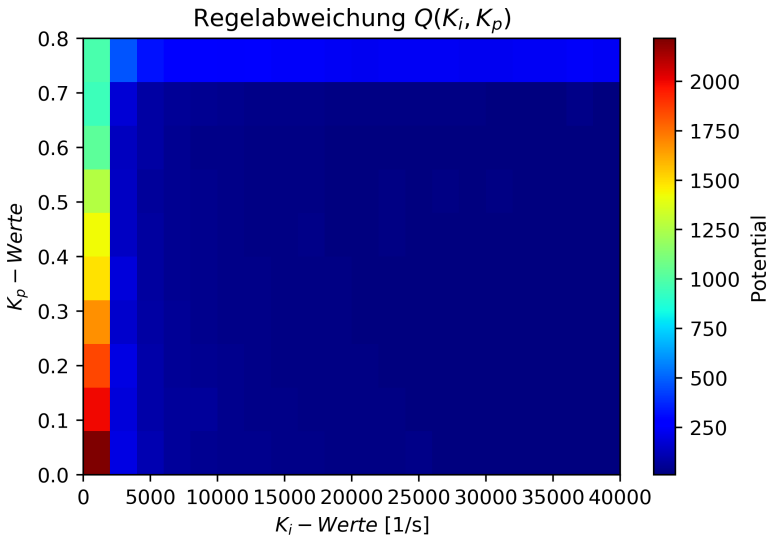

Abbildung C.4.: Gesamte Regellandschaft für die Teststrecke. Für eine optimale Regelung soll Q minimal sein.

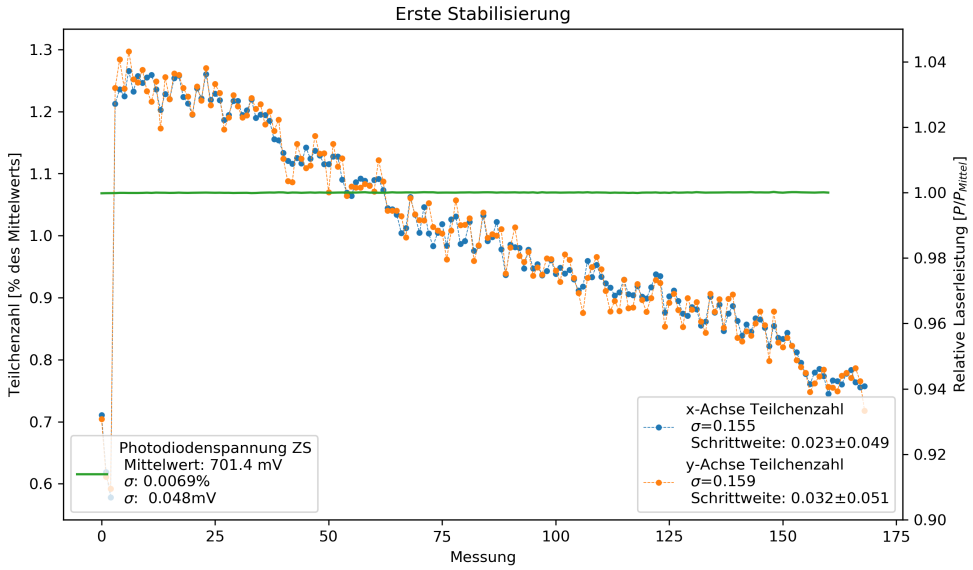

Abbildung C.5.: Erste Messungen der Teilchenzahl bei eingeschalteter Stabilisierung. Obwohl die Laserleistung konstant ist, fällt die Teilchenzahl rapide ab.

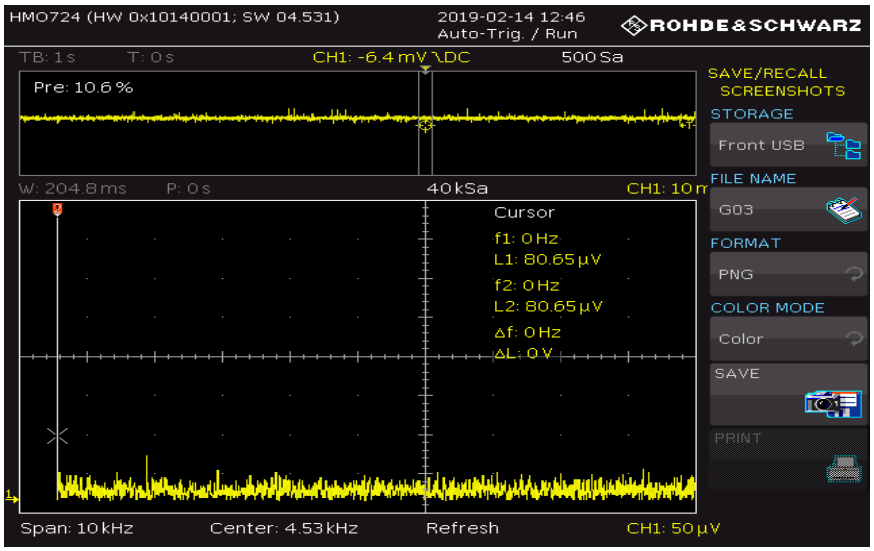

Abbildung C.6.: Rauschen der Photodiodenspannung am Zeemanslower mit Stabilisierung.

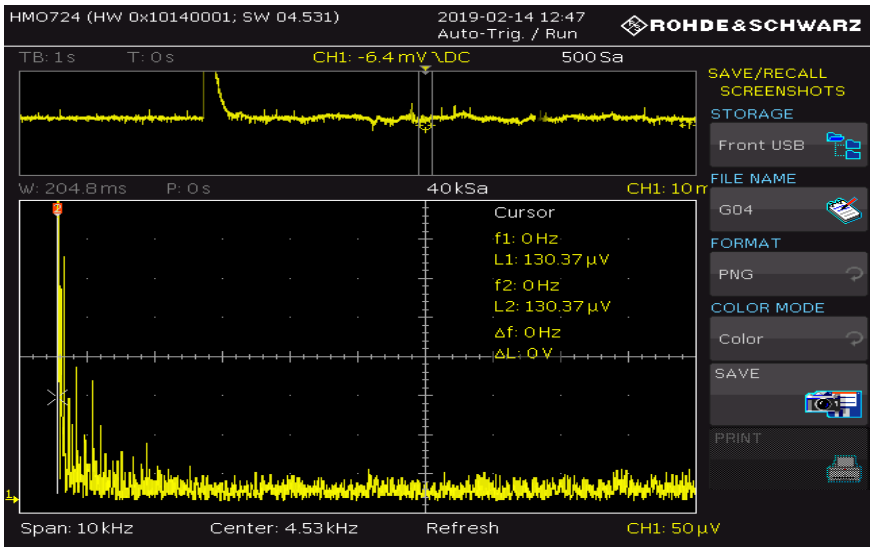

Abbildung C.7.: Rauschen der Photodiodenspannung am Zeemanslower ohne Stabilisierung.

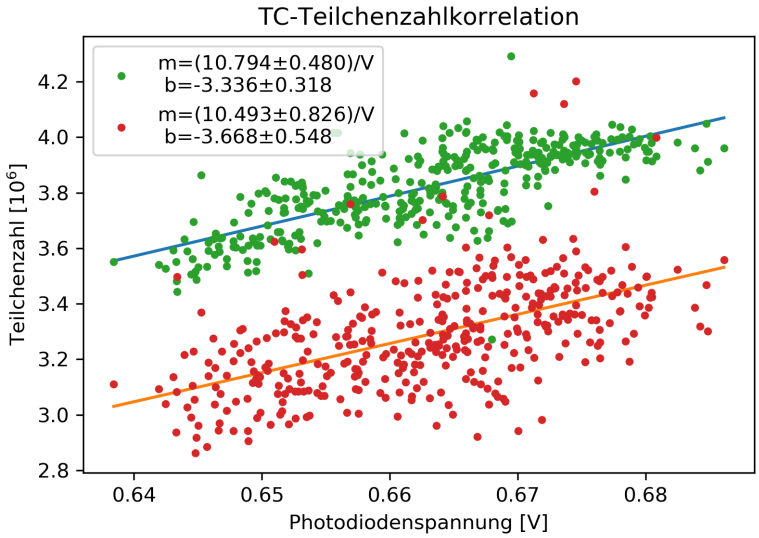

Abbildung C.8.: Korrelation zwischen der MOT-Teilchenzahl und der TC-Laserintensität. Es ist eine Korrelation erkennbar.

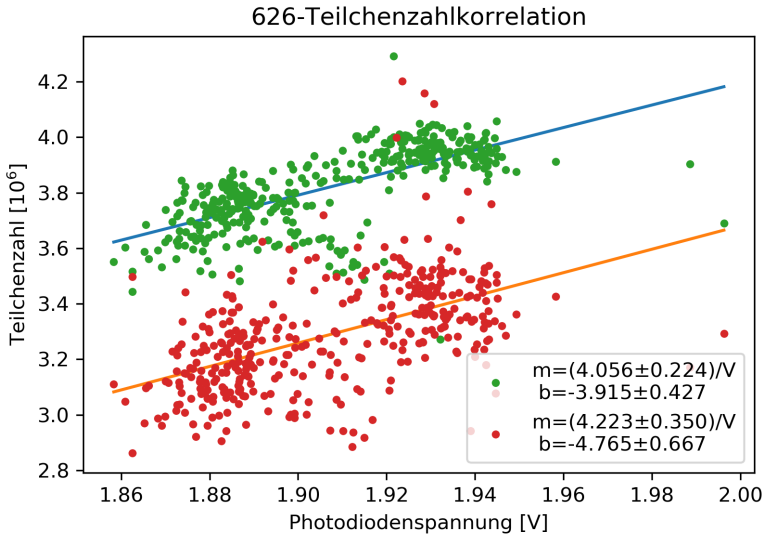

Abbildung C.9.: Korrelation zwischen der MOT-Teilchenzahl und MOT-Laserintensität. Es ist eine Korrelation erkennbar.

Abbildung C.10.: Messung der Teilchenzahl ohne Intensitätsstabilisierung. Zusätzlich zur Laserintensität am Zeemanslower wurden die Laserintensitäten am transversalen Kühlen (TC) und am 626 nm-MOT-Lasersystem (626) gemessen. Hierbei konnte keine Korrelation zwischen der Teilchenzahl und den Lasersystemen festgestellt werden.

D. Code

```
from pyrpl import Pyrpl
#Hier wird eine Verbindung zum Red Pitaya aufgebaut
p=Pyrpl(ip=100.100.100.100)
r=p.rp
# FPGA-Parameter koennen als Argumente in die Funktion setup uebergeben
    werden
pid1=r.pid1
pid2=r.pid2
pid3=r.pid0
pid1.setup(input="in1", ouput_direct="out2",p=-1)
pid2.setup(input="in2", output_direct="out2",p=1)
#Der Setzwert muss 0 sein, da der Setzwert von der Experimentsteuerung
    schon vom Eingangssignal abgezogen wurde
r.pid0.setup(input="out2", ouput_direct="out1",p=1,i=1000, min_voltage=0,
    max_voltage=0.5,setpoint=0)
```

Listing D.1:

```
   red_pitaya_pid_block i_pid (
     // data
     .clk_i        (  clk_i          ),  // clock
     .rstn_i       (  rstn_i         ),  // reset - active low
     .dat_i        (  input_signal [j] ),  // input data
     .dat_o        (  output_direct[j]),  // output data

     .trig_ext_i      (  trig_ext_i           ),  // external trigger
   ---Lehnen

   //communincation with PS
   .addr ( sys_addr[16-1:0] ),
   .wen  ( sys_wen & (sys_addr[20-1:16]==j) ),
   .ren  ( sys_ren & (sys_addr[20-1:16]==j) ),
   .ack  ( module_ack[j] ),
   .rdata (module_rdata[j]),
     .wdata (sys_wdata)
   );
```

Listing D.2:

Literaturverzeichnis

[1] Lewis F. L. , Applied Optimal Control and Estimation, Chapter 1: Introduction to Modern Control Theory, Prentice-Hall, (1992)

[2] Mühlbauer, F., Petersen, N., Baumgärtner, C. et al. Appl. Phys. B (2018) 124: 120. https://doi.org/10.1007/s00340-018-6981-2

[3] Kwee, P. , Laser characterization and stabilization for precision interferometry, Laser Interferometry & Gravitational Wave Astronomy, AEI-Hannover, MPI for Gravitational Physics, Max Planck Society, Doktorarbeit (2010).

[4] Bechhoefer J. , Feedback for physicists: A tutorial essay on control , Rev. Mod. Phys.77, (2005) 783-836

[5] Farooq, U. , Marrakchi Z., Mehrez H. , Tree-based Heterogeneous FPGA Architectures, Springer New York, ISBN: 9781461435945 (2012)

[6] Skogestad, S. , Probably the best simple PID tuning rules in the world. Norwegian University of Science and Technology (2001)

[7] Neuhaus, L. , Cooling a macroscopic mechanical oscillator close to its quantum ground state. Quantum Physics [quant-ph]. Université Pierre et Marie Curie (Paris 6), Doktorarbeit (2016)

[8] Türk, G. , Aufbau und Charakterisierung einer optischen Dipolfalle für Dysprosium, Bachelorarbeit, Johannes Gutenberg-Universität Mainz, Bachelorarbeit (2018)

[9] Red Pitaya (Jan 21, 2019): Red Pitaya STEMlab Documentation, Release 0.97

[10] THORLABS, PDA36A(-EC), Si Switchable Gain Detector User Guide (2017)

[11] Neuhaus L., pyrpl Documentation Release 0.9.4.0 (2017) https://media.readthedocs.org/pdf/pyrpl/latest/pyrpl.pdf

E. Danksagung

An dieser Stelle möchte ich mich bei allen bedanken, die mich beim Schreiben dieser Arbeit unterstützt haben. Ich möchte mich bei Prof. Dr. Windpassinger herzlich bedanken, der mich in seine Arbeitsgruppe aufgenommen hat und so diese Bachelorarbeit erst ermöglichte. Genauso herzlich möchte ich mich bei der ganzen Arbeitsgruppe bedanken, die vom ersten Tag an unglaublich nett und hilfsbereit war.
Außerdem möchte ich mich bei Prof. Dr. Jochen Walz dafür danken, dass der sich bereiterklärt hat, das Zweitgutachten für diese Arbeit, zu erstellen.
Mein ganz besonderer Dank geht an Marcel Trümper, der sich sehr viel Zeit genommen hat, um mir Fragen zu beantworten und diese Bachelorarbeit in die richtige Richtung zu lenken. Genauso möchte ich mich bei Niels Petersen bedanken, der mir immer bei Fragen und Problemen in den vergangenen Monaten geholfen hat.
Mein herzlichster Dank geht an Marina und Celine, die sich die Zeit genommen haben, diese Arbeit nach Fehlern zu durchsuchen. Ich möchte mich außerdem bei allen meinen Freunden bedanken, die mich in den letzten Wochen davor bewahrt haben, verrückt zu werden, genauso wie bei denen, die mit mir durch das Physikstudium gegangen sind. Ohne euch hätte ich das nicht geschafft, geschweige denn so viel Spaß dabei gehabt.
Zum Schluss möchte ich mich bei meiner Familie und vor allem meinen Eltern bedanken, weil ihr mich bei allem, was ich tue, unterstützt und an mich glaubt.