Die approbierte Originalversion dieser Diplom-/Masterarbeit ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.
http://www.ub.tuwien.ac.at

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology.
http://www.ub.tuwien.ac.at/eng

Unterschrift BetreuerIn

TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

DIPLOMARBEIT

# Entwicklung einer Instrumentansteuerung für das qBounce Experiment

ausgeführt am Atominstitut
der Technischen Universität Wien

unter der Anleitung von
**Dipl.-Phys. Dr. Tobias Jenke und**
**Univ.Prof. Dipl.-Phys. Dr. Hartmut Abele**

durch

**Jörg Herzinger**

Obere Amtshausgasse 19/10, 1050 Wien

15. September 2014

Unterschrift StudentIn

# Zusammenfassung

Im Rahmen dieser Diplomarbeit wurde die Ansteuerung der Geräte des qBounce Experiments überarbeitet und die Ausrichtung normal auf den Erdschwerpunkt verbessert.

Die Ansteuerung wurde mittels der Entwicklungsumgebung LabView von Grund auf neu konzipiert und programmiert. Hier wurde speziell eine Methode entwickelt mit der das Experiment möglichst einfach mit neuen Geräten erweitert werden kann, indem alle Komponenten voneinander separiert wurden. Im weiteren wurde hier auch die Datenaufnahme über die Abstraktionsebene des Netzwerks mit Hilfe des Syslog Protokolls neu implementiert.

Die Ausrichtung des Experiments wurde mittels neuer Winkelmesser und Messelektronik umgesetzt. Hier galt es speziell das Rauschverhalten des Regelkreislaufs und der einzelnen Komponenten zu bestimmen.

Der Vakuumkreislauf wurde mittels maschinell steuerbarer Ventile automatisiert, was eine Fehlbedienung der manuellen Ventile im Laborbetrieb ausschließt.

# Abstract

Within this diploma thesis the device control of the qBounce experiment was revised and and the leveling of the experiment normal to the earths center of gravity was improved.

The device control was implemented and designed from scratch using the development environment LabView. Here a new method was developed which easily allows to expand the experiment with new devices. This was achieved by separating all components from each other. Additionally the data acquisiton was reimplemented, with the network as a level of abstraction, using the syslog protocol.

Leveling the Experiment was implemented using new tiltsensor and measurement electronics. Here special attention was paid to the noise characteristics of the control circuit and the separate components.

The vacuum circuit was automated using machine controllable valves which removes the risk of an incorrect control of the manual valves during lab operation.

# Inhaltsverzeichnis


**1. Einleitung** **3**

**2. Das qBounce Experiment** **5**
2.1. Grundlagen . . . . . 5
2.2. Quantenmechanische Betrachtung . . . . . 6
2.3. Experimentaufbauten (Auswahl) . . . . . 7
2.3.1. Quantum Bouncing Ball . . . . . 7
2.3.2. Rabi Spektroskopie . . . . . 8
2.3.3. Ramsey Spektroskopie . . . . . 9
2.4. Instrumentaufbau . . . . . 10
2.5. Ablauf einer Messung am PF2/ILL . . . . . 12
2.6. Bisherige Lösung . . . . . 13
2.6.1. Nivellierung . . . . . 13
2.6.2. Software . . . . . 14
2.6.3. Vakuumkreislauf . . . . . 14

**3. Instrumentsteuerung** **17**
3.1. Verwendete Hardware . . . . . 17
3.2. LabView . . . . . 17
3.2.1. Variablen als Bindeglieder . . . . . 18
3.2.2. Aufbau des LabView Projekts . . . . . 18
3.2.3. Jobstruktur . . . . . 18
3.2.4. Konfigurationsdateien für LabView Programme . . . . . 20
3.2.5. Zentrale Verwaltung der LabView Programme . . . . . 21
3.2.6. Scriptmodus . . . . . 21
3.2.7. Datenein- und Ausgabe über den ADC/DAC . . . . . 24
3.2.8. Nivellierung . . . . . 25
3.3. Syslog . . . . . 25
3.3.1. Zeilenformat Syslog . . . . . 26
3.3.2. Dateistruktur und Datenformat . . . . . 27
3.3.3. Komprimieren der Daten und Backup . . . . . 29
3.3.4. Mathematica Paket zum Einlesen der Daten . . . . . 29

**4. Nivellierung des Experiments** **31**
4.1. Grundlagen . . . . . 31
4.2. Regelkreis . . . . . 31
4.3. ADC/DAC Datenkarte . . . . . 32
4.4. Winkelmesser . . . . . 33
4.4.1. Rauschbestimmung . . . . . 33
4.5. Auswertung . . . . . 35
4.5.1. Piezo Elemente . . . . . 36
4.5.2. ADC . . . . . 36
4.5.3. DAC . . . . . 36
4.5.4. Ergebnis . . . . . 37
4.6. Nivellierung . . . . . 37
4.6.1. Ausrichtung des Winkelmessers . . . . . 38
4.6.2. PID Regelung . . . . . 40
4.6.3. Schaltung High/Low Gain . . . . . 41
4.6.4. Analyse der Regelung . . . . . 42
4.6.5. Vergleich mit Ergebnissen der Vorjahre . . . . . 43
4.7. Verbau der Elektronik . . . . . 45

**5. Automatisierung des Vakuumkreislaufs** **47**
5.1. Halbautomatisches Testen . . . . . 48
5.2. Vollautomatisierte Steuerung der Ventile . . . . . 48
5.3. Auswertung der Ergebnisse . . . . . 50

**6. Zusammenfassung und Ausblick** **51**
6.1. Nivellierung . . . . . 51
6.2. Vakuumkreislauf . . . . . 51
6.3. Software . . . . . 52

**Literatur** **i**

**A. Script zur Kompression und Backup der Daten** **vii**

**B. Mathematica Paket** **ix**

**C. Einstellungen an der DCU** **xxv**
C.1. Normalbetrieb über Knopf . . . . . xxv
C.2. Testbetrieb . . . . . xxv

**D. Automatisierung Vakuum** **xxvii**

# 1. Einleitung

Im Rahmen des qBounce-Experiments soll das Newtonsche Gravitationsgesetz bei Mikrometerabständen bewiesen werden. Dazu werden ultrakalte Neutronen auf speziellen Spiegeln reflektiert und dann deren Sprunghöhe vermessen. Die Quelle der ultrakalten Neutronen, die für dieses Experiment verwendet werden, befindet sich am Institut Laue Langevin in Grenoble. Der speziell zu Forschungszwecken gebaute Kernreaktor bietet einen sehr hohen Fluss thermischer Neutronen im Moderator. Die thermischen Neutronen werden zuerst in Deuterium moderiert und im weiteren mittels einer sogenannten „Steyerl Turbine“ zu den Experimenten geleitet. Nach diesem Prozess haben die Neutronen eine Geschwindigkeit von ca. 5 m/s und fallen somit in den Bereich ultrakalter Neutronen. Die ultrakalten Neutronen werden dann über ein Strahlrohr an die verschiedenen Experimentierplätze geleitet. Dort erfolgt für das qBounce Experiment eine weitere Geschwindigkeitsselektion über Blendensysteme. Durch diese Geschwindigkeitsselektionen sinkt die Zählrate bereits deutlich. Auf Grund dieser sehr geringen Zählraten sind meist sehr lange Messzeiten von mehreren Tagen bis Wochen notwendig, was wiederum die Notwendigkeit eines stabilen Instruments und einer guten Instrumentansteuerung aufzeigt.

Für das qBounce Experiment wird eine relativ große und dadurch auch schwere Vakuumkammer benötigt in der die Neutronenspiegel mit der Experimentansteuerung Platz finden. Allen Experimenten ist gleich, dass Neutronen auf speziellen Spiegelsystemen reflektieren. Durch μ-Metalle und Spulensysteme können Einflüsse von Magnetfeldern minimiert werden und so verbleibt als größter Einfluss das Gravitationsfeld der Erde.

Da bei Verkippungen der Neutronenspiegel die Teilchen zur Seite reflektieren würden, ist es wichtig die Spiegeloberfläche normal auf den Erdschwerpunkt zu halten. Eine Verkippung würde mit dem Faktor $cos\theta$ eingehen, wodurch kleine Änderungen weniger problematisch werden. Um solche Verkippungen möglichst klein zu halten wurde im ersten Teil dieser Diplomarbeit die Nivellierung des Experimentaufbaus mit Hilfe eines sehr genauen Winkelmessers zur Auslese der Verkippung und Piezo Elementen zur Korrektur der Verkippung modernisiert. Der Winkelmesser erlaubt eine Auslese der Verkippung mit einer Auflösung von wenigen μrad im Bereich von ± 800 μrad. Mittels der Piezo Elemente konnte diese zeitnah in einem Bereich von ca. ± 100 μrad korrigiert werden.

Die Regelung der Verkippung wurde mithilfe der Entwicklungsumgebung und

Programmiersprache LabView implementiert. Besonderen Wert bei der Programmierung wurde darauf gelegt alle Komponenten möglichst modular und einfach zu halten. Ein übergeordnetes Programm sollte es ermöglichen alle einzelnen Komponenten zentral aufzurufen und Konfigurationsdateien sollten es ermöglichen alle wichtigen Parameter zentral in einfachen Textdateien zu bestimmen. Als zusätzliche Bedingung musste das übergeordnete Programm einfach erweiterbar sein um für zukünftige Änderungen und Erweiterungen geeignet zu sein. Speziell wurde hierbei auf den Einsatz moderner Technologien aus dem Bereich der Computer-Systemadministration geachtet.

Im weiteren wurde der Vakuumkreislauf analysiert und automatisiert. Um große Volumina zu evakuieren ist es oft von Vorteil über ein Bypass-System zu arbeiten, wie in Abbildung 2.5 ersichtlich. Derartige Bypass-Systeme erlauben es die Vorpumpe und die Turbomolekularpumpe durchgehend eingeschaltet zu lassen und ermöglichen so einen schnelleren Arbeitsablauf, falls Arbeiten in der Kammer notwendig sind. Der große Nachteil dieser Arbeitsweise besteht darin, dass die händische Bedienung der vielen Ventile zu Fehlern führen kann. Im schlimmsten Fall kann es passieren, dass die Turbomolekularpumpe im eingeschalteten Zustand belüftet wird. Um diese menschlichen Fehler zu beseitigen wurde der Vakuumkreislauf (Einschalten der Pumpe und aussaugen der Kammer, sowie Herunterfahren der Pumpen und sicheres Belüften der Kammer) mittels automatischer Ventile und einer Relaisschaltung automatisiert.

# 2. Das qBounce Experiment

## 2.1. Grundlagen

Das qBounce Experiment nutzt ultrakalte Neutronen und untersucht deren Verhalten im Gravitationsfeld der Erde. Diese Neutronen springen klassisch betrachtet analog zu einem Ball auf einem Neutronenspiegel auf, woher auch der Name „quantum bouncer“ stammt. In unterschiedlichen Experimentaufbauten können so unterschiedliche Eigenschaften des quantenmechanischen Systems der Neutronen im µm Bereich betrachtet werden.

Neutronen sind für diese Klasse an Experimenten besonders geeignet, da, im Gegensatz zu Atomen, ihre Polarisierbarkeit sehr klein ist. Eine weitere besonders günstige Eigenschaft ultrakalter Neutronen ist, dass sie unter allen Einfallswinkeln von Oberflächen total reflektiert werden [3]. Auf Grund der geringen De-Broglie-Wellenlänge der ultrakalten Neutronen von

$$\lambda_{dB} = \frac{h}{p} \approx 60\,\mathrm{nm}, \tag{2.1}$$

wobei p hier die kinetische Energie der Neutronen ist, die um ca. 6 Größenordnungen über einem typischen Kernabstand von $f \approx 20\,\mathrm{fm}$ liegt, sieht das Neutron effektiv ein gemitteltes Potential über mehrere Atomkerne, welches als Fermipotential $U_{eff}$ bezeichnet wird. Die Reflexionsbedingung für Neutronen ist somit erfüllt, wenn die kinetische Energie der ultrakalten Neutronen kleiner als das Fermipotential der Oberfläche ist, also $E_{kin} < U_{eff}$. Die im qBounce-Experiment verwendeten Neutronenspiegel aus BK7 Glas weisen ein Fermipotential von $U_{eff} \approx 100\,\mathrm{neV}$ [4] auf, wodurch die Reflexionsbedingung für Neutronen bis zu einer Geschwindigkeit von $v_\perp \approx 4.4\,\mathrm{m/s}$ erfüllt ist.

Eine Notwendigkeit, die ultrakalte Neutronen mit sich bringen, ist das Arbeiten in Vakuum. Die durch den Experimentaufbau bedingte (siehe hierzu 2.3) bereits sehr geringe Neutronen-Zählrate würde durch Streuung der Neutronen an Luftmolekülen weiter gesenkt. Beschrieben werden kann dieses Verhalten durch das Lambert-Beer Gesetz

$$N(x) = N_0 e^{-\mu x} = N_0 e^{-n\sigma x} \tag{2.2}$$

welches die Intensität eines Teilchenstrahls mit Anfangsintensität $N_0$ beim Durchgang durch ein Material bzw. ein Gas mit Absorptionskoeffizient $\mu$ beschreibt. Der

Absorptionskoeffizient $\mu$ wiederum setzt sich zusammen aus der Teilchenzahl pro Kubikmeter $n$ und dem Wirkungsquerschnitt $\sigma$. Aufgrund des exponentiellen Zusammenhangs der Teilchenzahl pro Kubikmeter, sprich des Luftdrucks, und der transmittierten Intensität ist ein möglichst geringer Luftdruck von Vorteil.

## 2.2. Quantenmechanische Betrachtung

Ein kurzer Auszug aus der quantenmechanischen Beschreibung von Neutronen im Gravitationsfeld der Erde sei folgend gegeben, eine genauere Beschreibung findet sich in [5] und [6]. Für die quantenmechanische Betrachtung eines solchen Neutrons ergibt sich die Schrödingergleichung zu

$$\left(-\frac{\hbar^2}{2m}\Delta + mgz\right)\psi\left(\vec{r},t\right) = i\hbar\frac{\partial\psi\left(\vec{r},t\right)}{\partial t}. \tag{2.3}$$

Hierbei bezeichnet $m$ die Neutronenmasse, $g$ die Erdbeschleunigung und $z$ die z-Komponente der Ortskoordinate des Teilchens. Mittels Separationsansatz lässt sich die allgemeine Lösung der z-Komponente der Wellenfunktion $\psi\left(\vec{r},t\right)$ schreiben als

$$\widetilde{\psi}\left(z\right) = c_1 A_i\left(\frac{z}{z_0} - \frac{E}{E_0}\right) + c_2 B_i\left(\frac{z}{z_0} - \frac{E}{E_0}\right), \tag{2.4}$$

wobei

$$z_0 = \sqrt[3]{\frac{\hbar^2}{2m^2 g}}\ ,\ E_0 = mgz_0. \tag{2.5}$$

Die Eigenenergien der Neutronen $E_0$ ergeben sich aus den Randbedingungen. Die Funktionen $A_i$ und $B_i$ sind die linear unabhängigen Airy Funktionen und die Konstanten $c_1$ und $c_2$ ergeben sich aus den Randbedingungen und Normierungen. Da Totalreflexion für ultrakalte Neutronen mit geeigneten Spiegeln gut realisierbar ist, lässt sich die Randbedingung $\psi(z=0)=0$ gut erfüllen.

Zur Präparation der Neutronen in den Grundzustand befindet sich auf dem Spiegel ein Streuer in Form eines aufgerauten Spiegels. Dieser streut Neutronen aus dem System die in weiterer Folge absorbiert werden. Der Abstand zwischen dem unten liegenden Neutronenspiegel und dem darüber liegenden Streuer liegt üblicherweise in der Größenordnung einiger µm ist so gewählt, dass nach Durchquerung des Systems nur der erste Zustand selektiert ist.

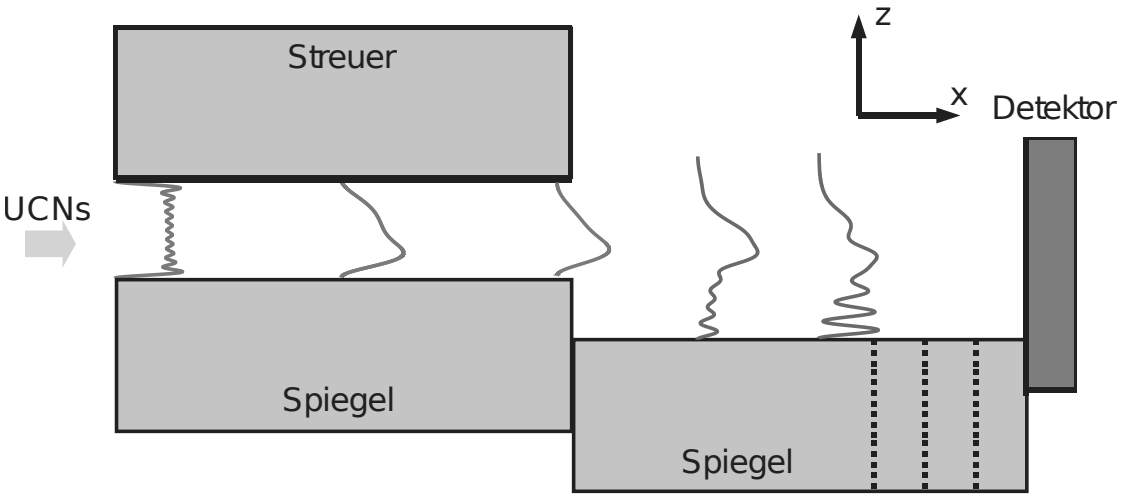



Abbildung 2.1.: Schematischer Aufbau des quantum bouncing ball Experiments (Quelle [3])

## 2.3. Experimentaufbauten (Auswahl)

In diesem Kapitel soll ein kurzer Überblick über die bisherigen Versuchsaufbauten der qBounce Experimente gegeben werden. Diese fußen auf Experimenten von H. Abele und V. Nesvizhevsky in den Jahren 1998 bis 2005. Begonnen wurden die Experimente zum „quantum bouncing ball“ (siehe Kapitel 2.3.1) im Jahr 2008 in dem erste Tests zur Machbarkeit eines solchen Spiegelsystems und dessen Transmissionsraten bestimmt wurden. Im weiteren wurden die Aufbauten schrittweise erweitert, bis im Jahr 2012 ein Rabi-Experiment (siehe Kapitel 2.3.2) realisiert werden konnte. In zukünftigen Aufbauten ist dann ein deutlich komplexerer Ramsey-Aufbau (siehe Kapitel 2.3.3) geplant.

### 2.3.1. Quantum Bouncing Ball

Die Experimentaufbauten für den quantum bouncing ball bestehen aus zwei Regionen und benutzen einen ortsauflösenden Detektor um die Zeitentwicklung der Neutronen zu beobachten. Ein solcher Aufbau ist schematisch in Abbildung 2.1 zu sehen und genauer beschrieben in [7], Kapitel 2. Die über ein Blendensystem bereits nach Geschwindigkeit selektierten Neutronen [8] werden in einer ersten Region in den Grundzustand präpariert, „fallen“ dann über eine wohldefinierte Stufe auf einen Spiegel in Region zwei und können sich hier zeitlich entwickeln. Die Stufenhöhe $l$ geht hier in das Übergangselement der Wellenfunktionen zwischen Region eins und zwei ein.

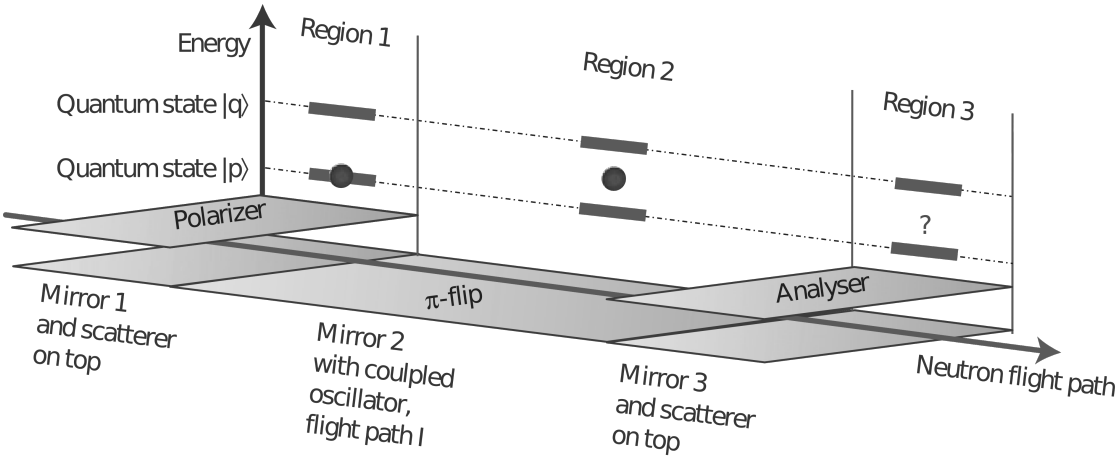



Abbildung 2.2.: Schematischer Aufbau einer Rabi Spektroskopie (Quelle: [12])

$$\langle \psi_I | \psi_{II} \rangle = \int_s^{s+l} \psi_I \psi_{II} dz \tag{2.6}$$

Als Detektor werden bei diesen Experimentaufbauten mit Bor beschichtete CR-39 Kunststoffplättchen [9, 10, 11] eingesetzt, welche eine örtliche Auflösung der detektierten Neutronen erlauben. Auf die Bor-Schicht auftreffende Neutronen erzeugen über Kernreaktionen Li-Ionen und α-Teilchen, welche wiederum einen Defekt im Kunststoffplättchen verursachen. Über ein spezielles Ätzverfahren kann die Bor-Schicht entfernt werden und die Defekte im Plättchen können somit mittels eines Mikroskops ortsaufgelöst registriert werden. Auf diese Weise kann mittels Variation der Länge des Spiegels in Region zwei die Zeitentwicklung des Neutrons unter Einfluss der Gravitation bestimmt werden.

### 2.3.2. Rabi Spektroskopie

Diese 1938 von Isidor Isaac Rabi vorgeschlagene [13] und 1944 mit dem Nobelpreis prämierte Form der Resonanzspektroskopie erlaubt Präzionsexperimente, da hier die Bestimmung der Energie bzw. Messgröße auf die Messung einer Frequenz $\nu$ zurückgeführt werden kann

$$E = h\nu. \tag{2.7}$$

Quantenmechanisch betrachtet gilt es die zeitabhängige Schrödingergleichung

$$V(t)\psi(t) = i\hbar \frac{\partial \psi(t)}{\partial t} \tag{2.8}$$

mit dem Störpotential

$$V(t) = \begin{pmatrix} 0 & \frac{1}{4}\hbar\Omega_R e^{-i(\omega_{pq}-\omega)t} \\ \frac{1}{4}\hbar\Omega_R e^{i(\omega_{pq}-\omega)t} & 0 \end{pmatrix} \tag{2.9}$$

zu lösen, wobei $\omega_{pq} = (E_q - E_p)/\hbar$ und $\omega$ die Frequenz der Störung bezeichnet. Die Frequenz $\Omega_R$ ist die sogenannte Rabifrequenz, die angibt mit welcher Frequenz die Besetzung der Zustände oszilliert. Mit dem Ansatz

$$|\psi(t)\rangle = c_p(t)\,|p(t)\rangle + c_q(t)\,|q(t)\rangle \tag{2.10}$$

lässt sich unter der Anfangsrandbedingung, dass nur Neutronen im Grundzustand die Region I verlassen dürfen, $c_p(t=0) = 1$, die Lösung direkt als Übergangswahrscheinlichkeit angeben

$$P_{p\to q} = |c_p(t)|^2 = \frac{\Omega_R^2}{\Omega_R^2 + (\omega_{pq}-\omega)^2} \sin^2\left(\sqrt{\Omega_R^2 + (\omega_{pq}-\omega)^2} + t\right). \tag{2.11}$$

Im Falle von qBounce ergibt sich folgender Aufbau in drei Regionen, der auch in Abbildung 2.2 zu sehen ist:

1. Präparation der Neutronen in den Grundzustand $|p\rangle$.
2. Anregung der Neutronen über einen vibrierenden Spiegel $|q\rangle$.
3. Messung des Zustands, bzw. Selektion des Zustands analog zu Region I.

Bei erreichen der Resonanzfrequenz $\Delta\omega = \omega_{pq} - \omega = 0$ tritt ein sogenannter π-Flip auf. Im Experiment wäre hier ein deutlicher Einbruch der Messrate am Detektor zu sehen, da die angeregten Neutronen aus Region II in Region III aus dem Experiment gestreut würden.

Experimentell ergeben sich bei einem solchen Aufbau große Herausforderungen. Zum einen gilt es alle Spiegel genau zu justieren und die Stufen zwischen den Regionen im Bereich weniger µm zu halten. Ebenfalls sollten die Spiegel eine gegeneinander möglichst geringe Verkippung aufweisen. Der in Region II vibrierende Spiegel stört die Justage der umliegenden Spiegel naturgemäß. Ebenfalls ist zur Steuerung der diversen Geräte ein gutes Management der Software von Nöten. Eine technische Grenze ist der Länge der Region II gesetzt. Die Realisierung der Vibrationen unter den gegebenen Qualitätsanforderungen ist nur unter sehr hohem technischem Aufwand machbar. Einen Ausweg bietet die Ramsey Spektroskopie.

### 2.3.3. Ramsey Spektroskopie

Die von Norman Ramsey 1950 entwickelte [14] und 1989 mit dem Nobelpreis gewürdigte Methode der separierten oszillierenden Felder lässt sich ebenfalls auf Neutronenstrahlen umlegen. Der Aufbau hier besteht wie in Abbildung 2.3 zu sehen aus 5 Regionen:

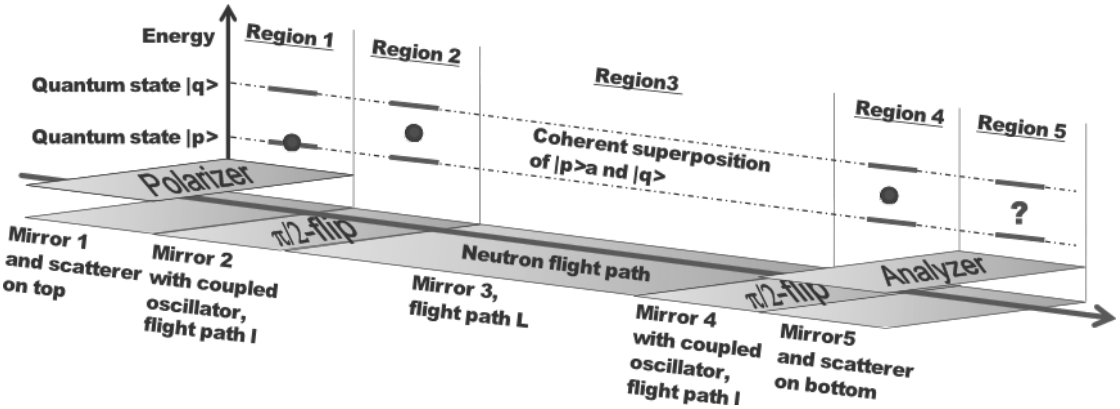



Abbildung 2.3.: Schematischer Aufbau von Ramseys Methode separierter oszillierender Felder (Quelle: [6])

1. Präparation der Neutronen in den Grundzustand $|p\rangle$.
2. Anregung der Neutronen, $\pi/2$-Flip, über einen vibrierenden Spiegel in den Zustand $|q\rangle$.
3. Kohärente Superposition der Zustände $1/\sqrt{2}\,|p\rangle + 1/\sqrt{2}\,|q\rangle$.
4. Weitere Anregung der Neutronen, $\pi/2$-Flip, über einen vibrierenden Spiegel.
5. Analyse des Neutronenstrahls.

Der große Vorteil gegenüber Rabis Methode liegt hier darin, dass die beiden vibrierenden Regionen relativ kurz gewählt werden können und die Messgenauigkeit durch die Länge des dritten Abschnitts gegeben ist, der prinzipiell relativ lang gewählt werden kann.

Experimentell ergeben sich hier weitere Herausforderungen. Alle fünf Regionen mit den vier Übergängen müssen in einer Ebene mit Stufen von $< 1\,\mu$m gehalten werden. Dies wird mit steigender Länge der Spiegel komplexer und die Vermessung dauert länger. Des weiteren setzt diese Methode voraus, dass die beiden Spiegel in Region II und IV in Phase vibrieren, was ebenfalls experimentell erwiesen werden muss. Durch die zwei weiteren Spiegel und die zusätzliche vibrierende Region gegenüber Rabis Methode ist hier ebenfalls eine sehr gute Ansteuerung der Geräte von der Softwareseite aus eine Grundvoraussetzung.

## 2.4. Instrumentaufbau

Durchgeführt wurden die Messungen für das qBounce Experiment am PF2 des Forschungsreaktors des ILL[1]. Eine anschaulicher Überblick über das PF2 Instrument ist in Abbildung 2.4 gegeben.

[1] Institut Laue Langevin, 71 avenue des Martyrs - CS 20156 - 38042 GRENOBLE CEDEX 9

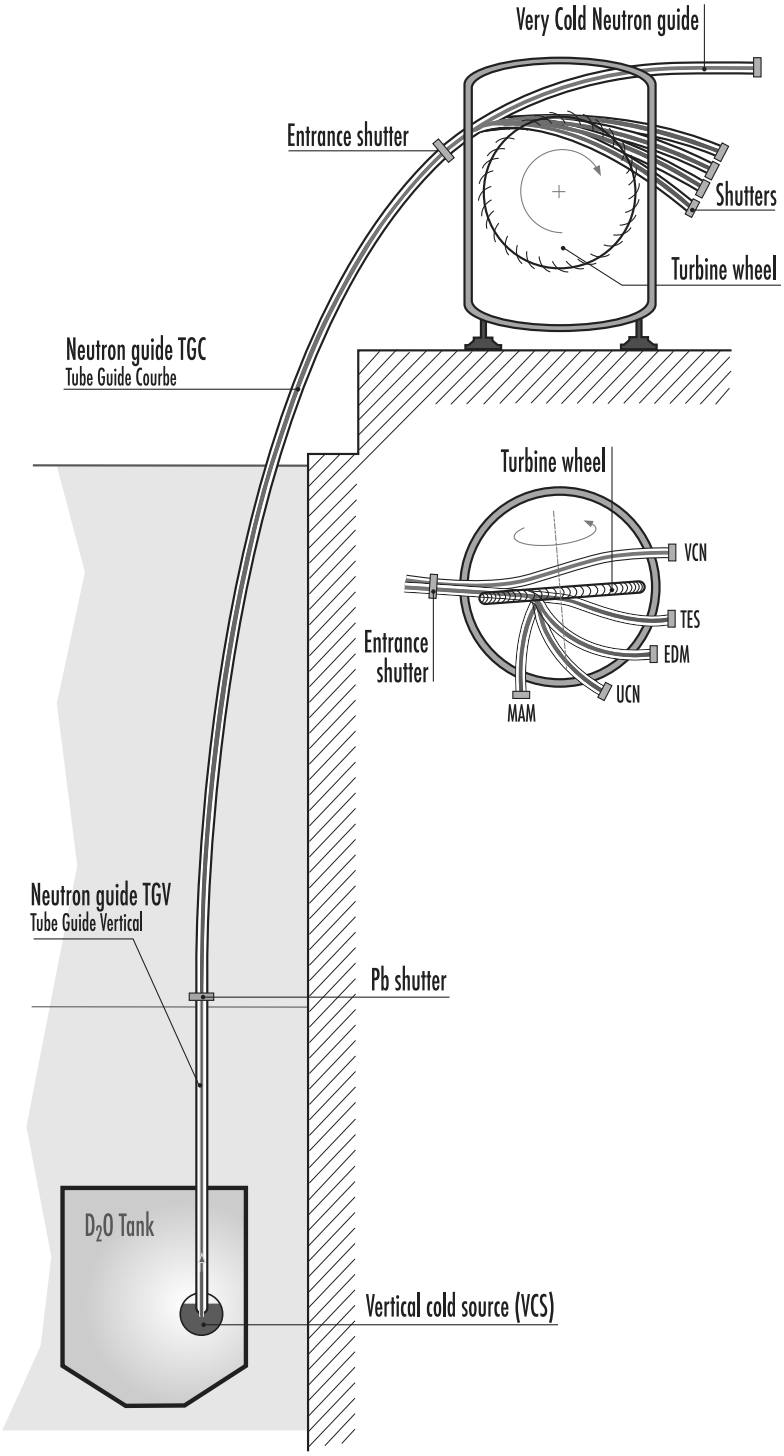



Abbildung 2.4.: Das PF2 Instrument des Instituts Laue Langevin in Grenoble (Quelle: www.ill.eu).

| # | Bezeichnung | Zeit [s] | qBounce Bez. | Messneutronen |
|---|---|---|---|---|
| ① | Filling | 10 | Vorbereiten | Vorhanden |
| ② | Cleaning | 190 | Messen | |
| ③ | Storing | 3 | Wartezeit | Unbekannt |
| ④ | Storing A | 3 | | |
| ⑤ | Storing B | 3 | | |
| ⑥ | Emptying | 3 | | |
| ⑦ | Sonstiges | * | Untergrund | Nicht vorhanden |

Tabelle 2.1.: Aufzählung der vorhandenen PF2 Signale und deren Verwendung für das qBounce Experiment

Die thermischen Neutronen des Reaktors werden in einer kalten Quelle, die Deuterium, das auf einer Temperatur von 25 K gehalten wird, enthält, zu kalten Neutronen moderiert. Über ein gekrümmtes Strahlrohr, das eine Höhendifferenz von mehreren Metern überbrückt, werden die Neutronen bereits zu sehr kalten Neutronen abgebremst, im weiteren über eine sogenannte „Steyerl Turbine“ weiter zu ultrakalten Neutronen abgebremst und an die verschiedenen Strahlplätze geleitet.

Über einfache Strahlrohre werden die Neutronen dann zu der Kammer des qBounce Experiments geleitet. Vor Eintritt in die Kammer werden die Neutronen durch ein spezielles Blendensystem [8] in einem spezifischen Geschwindigkeitsbereich selektiert.

## 2.5. Ablauf einer Messung am PF2/ILL

Grundsätzlich ist die Steuerung der Steyerl Turbine und damit die Aufteilung der Neutronen auf die einzelnen Strahlplätze auf Speicherexperimente ausgelegt. Bei Speicherexperimenten werden die erhaltenen Neutronen in einer sogenannten „Flasche“ gespeichert, welche nach einiger Zeit geleert und ausgewertet wird. Auf diese weise ist es prinzipiell möglich, dass mehrere Experimente gleichzeitig messen können. Da das qBounce Experiment nicht speichert, müssen hier die vorhandenen Steuersignale entsprechend angepasst werden. Ersichtlich sind die vorhandenen Steuersignale bzw. deren Verwendung für das qBounce Experiment in Tabelle 2.1.

Die Angegebenen Zeiten können von den Experimentierplätzen frei gesetzt werden und unterliegen nur zwei Regeln des PF2, die festsetzen, dass

- ① + ② $\leqslant$ 200 s,
- $\sum$ ③ .. ⑥ $\geqslant$ 12 s.

Die erste Regel dient der fairen Aufteilung der Messzeit unter den Experimentierplätzen, die zweite Regel ergibt sich aus der Verfahrzeit der Turbine. Die Signale werden in Form von standardisierten TTL Signalen ausgegeben und können mit

Hilfe der in Kapitel 3.2.7 beschriebenen Datenkarte ausgelesen und verarbeitet werden.

**Vorbereiten** in diesem Zyklus ist der Shutter zwischen der Turbine und dem Experiment bereits geöffnet. Da es einige Zeit dauert bis die volle Neutronenrate erreicht ist, können hier die Geräte für eine Messung vorbereitet werden.

**Messen** In diesem Zyklus erreichen die ultrakalten Neutronen das Experiment und es kann gemessen werden.

**Wartezeit** In diesem Zyklus fährt die Turbine eventuell zu einem der anderen Strahlplätze. Diese Zeit kann also für das qBounce Experiment nicht genutzt werden, da hier der genaue Status der Turbine nicht bekannt ist und es sein kann, dass entweder ein Untergrund Zyklus, oder direkt ein weiterer Messzyklus folgt.

**Untergrund** In diesem Zyklus zeigt die Turbine auf einen der anderen Messplätze und die Zeit wird genutzt um den Untergrund des Detektorsignals zu vermessen. Außerdem können weitere Messungen, wie Stufenvermessungen, Stufennachregelungen etc., durchgeführt werden, die während des regulären Messzyklus aufgrund der elektromagnetischen Felder der Geräte nicht durchgeführt werden können.

## 2.6. Bisherige Lösung

Um die genaue Problem- bzw. Aufgabenstellung dieser Diplomarbeit aufzuzeigen soll hier kurz die bisherige Lösung einiger Komponenten diskutiert werden.

### 2.6.1. Nivellierung

Die bisherige Lösung zur Nivellierung des Experiments ist unter anderem beschrieben in [7] und [15]. Es wurden Winkelmesser der Firma Tech-Sys Instruments[2], Modell „755“, mit der dazugehörigen Elektronik verwendet. Da diese das beste am Markt erhältliche Winkelauflösungsvermögen bieten, wurden sie auch weiterhin verwendet. Genauere technische Angaben sind daher in Kaptiel 4.4 zu finden. Die Elektronik der Winkelmesser gibt ein analoges Spannungssignal zwischen $\pm 8\,\mathrm{V}$ aus, welches über einen an der Universität Heidelberg entwickelten Analog-Digital-Wandler (im weiteren ADC genannt) [15] namens „Logic Box“ in ein vom PC auslesbares Signal umgewandelt wird. Dieser ADC besitzt eine Auflösung von 14 bit, was, im Messbereich von $\pm 10\,\mathrm{V}$, einer theoretischen Auflösung in Schritten von

$$\frac{10-(-10)}{2^{14}-1} = 0.0012\,\mathrm{V} \tag{2.12}$$

[2] Tech-Sys Instruments, Floridalane 64, 1180 Brussels, Belgium, http://www.tech-sys.eu

entspricht. Diese Auflösung bzw. diese technische Limitierung stellt den ersten Wert dar, der im Rahmen dieser Diplomarbeit verbessert werden sollte.

Ein LabView Programm verarbeitete die Daten des Winkelmessers dann zu einer notwendigen Höheneinstellung von Piezoelementen, die, dreipunktgelagert, die Lage des Experiments ausglichen. Die Ausgabe der errechneten Höheneinstellung erfolgte in Form einer Spannung über einen Digital-Analog-Wandler (im weiteren DAC genannt), welcher ebenfalls in der „Logic Box“ enthalten ist. Ein Austausch der Piezoelemente auf Niedervoltaktoren wäre ebenfalls wünschenswert gewesen, konnte aus Kostengründen jedoch im Zuge dieser Diplomarbeit nicht durchgeführt werden. Die verwendete Hardware ist daher in Kapitel 4.5.1 zu finden.

### 2.6.2. Software

Das Signal des Wandlers wurde am PC mit Hilfe der Entwicklungsumgebung LabView (genaueres hierzu siehe 3.2) eingelesen und verarbeitet. Das Programm hierzu war über mehrere Jahre gewachsen und erweitert worden, weshalb in seiner letzten Iteration diese Erweiterungen schwer lesbar waren. Des weiteren wurden bisher mehrere Computer für die Steuerung der Geräte verwendet, was die Kommunikation der Geräte bzw. Programme untereinander erschwerte. Ein weiteres Ziel dieser Diplomarbeit war es daher die softwareseitige Implementierung der Nivellierung und im weiteren des gesamten qBounce Experiments neu zu konzipieren und zu implementieren. Die größten grundlegenden Probleme die es zu behandeln galt waren:

1. Zur Steuerung aller Geräte waren bis zu sieben PCs mit teils unterschiedlichen Treiberkonfigurationen notwendig. Dies sollte möglichst auf einen Rechner reduziert werden.
2. Die meisten LabView Programme waren über die Jahre angewachsen, sehr statisch programmiert und unübersichtlich geworden. Dies sollte mit Hilfe von Konfigurationsdateien dynamischer gestaltet werden und die Programme selbst sollten überarbeitet und unterschiedliche Funktionalitäten möglichst in Unterprogramme ausgelagert werden.
3. Da jedes LabView Programm die Datenausgabe selbst definierte und die Dateien dann auf bis zu sieben PCs verteilt waren, waren der Speicherort der Daten, die Dateibenennung und das Datenformat selbst stark fragmentiert. Ziel war es deshalb Daten- und Dateiformat zu vereinheitlichen und die Dateien an einem zentralen Speicherort bzw. Server abzulegen.

### 2.6.3. Vakuumkreislauf

Der Aufbau des Vakuumkreislaufs der bis vor dieser Diplomarbeit verwendet wurde ist schematisch in Abbildung 2.5 zu sehen und in [16] genauer beschrieben. Die

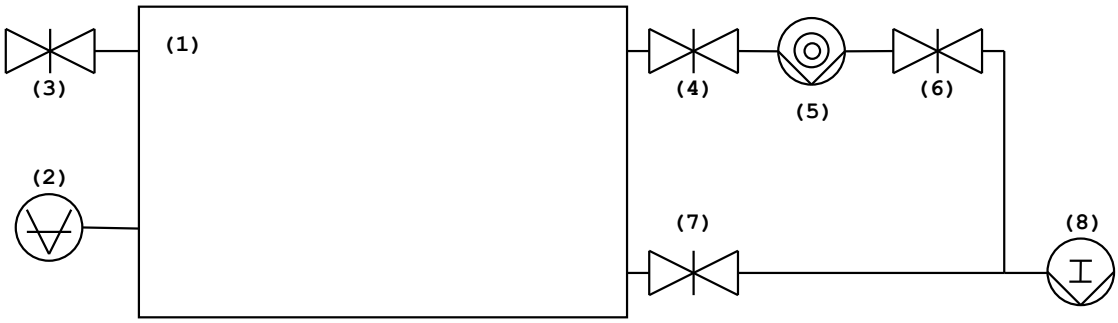



Abbildung 2.5.: Vakuumkreislauf mit Bypass. (1) Vakuumkammer, (2) Drucksonde, (3) Belüftungsventil, (4,6,7) Ventile, (5) Turbopumpe, (8) Vorpumpe

folgenden Ventilnummern beziehen sich jeweils auf Abbildung 2.5. Die Vakuumkammer des qBounce Experiments, mit einem Volumen von ca. $0.3\,\mathrm{m}^3$ wurde über ein Bypass-System über das Ventil (7) von der Vorpumpe auf einen Druck von ca. $10^{-3}$ mbar abgesaugt. Die Ventile (4) und (6) vor und hinter der Turbopumpe waren dabei geschlossen. War der Vordruck erreicht wurde, das Bypass-Ventil (7) geschlossen, die Turbopumpe eingeschaltet und die Ventile (6) und (4) der Reihe nach vorsichtig geöffnet. Diese Methode hatte den Vorteil, dass man im Falle von Wartungsarbeiten am Experiment, sprich in der Kammer, die Turbopumpe eingeschaltet lassen konnte. In diesem Fall wurden die Ventile (6) und (4) sowie das Bypass-Ventil (7) geschlossen und die Kammer über das Belüftungsventil (3) belüftet. Waren die Wartungsarbeiten beendet, wurde analog zum erstmaligen Evakuieren zuerst über den Bypass die Kammer auf Vordruck gebracht und danach die bereits laufende Turbopumpe zugeschaltet. Ein Arbeiten mit derartigen Bypass-Systemen ist in der Vakuumtechnik besonders bei großen Volumina üblich, da so die Abpumpzeiten verkürzt werden können. Ein bedeutender Nachteil dieser Methode ist, dass bei Fehlschaltung der manuellen Ventile die Turbopumpe über das Bypass-Ventil und den Auslass der Turbopumpe belüftet werden kann. Ein solcher Fehler kann bei der hohen Drehzahl der Turbopumpe von 833 Hz fatale Folgen haben. Im Normalfall muss hier die Turbopumpe zu einem Servicebetrieb geschickt werden, was einen Ausfall mehrerer Wochen bedeutet, im schlimmsten Fall kann dies zu einem Totalschaden der Turbopumpe führen. Eine weitere Fehlerquelle ist es, wenn die Turbopumpe ausgeschaltet wird, die Ventile (4) und (6) jedoch geöffnet bleiben. In diesem Fall kann es dazu kommen, dass der Unterdruck in der Vakuumkammer Öl aus der Vorpumpe in die Experimentumgebung saugt. Hierbei würden sowohl die Turbopumpe, als auch das innere der Vakuumkammer verunreinigt und dadurch das Experiment gefährdet. Da dies bereits in der Vergangenheit passiert ist, wurde als zusätzliche Sicherheit eine ölfreie Vorpumpe

verwendet.

Die Aufgabenstellung hier war es daher diese Fehlerquelle zu eliminieren und das Evakuieren und Belüften der Kammer zu automatisieren. Außerdem sollten systematische Tests der neuen, automatisierten Evakuierungsmethode gemacht und der „alten“ Bypass-Methode gegenübergestellt werden. Speziell die Evakuierungs- und Belüftungsdauer sind hier von Interesse.

# 3. Instrumentsteuerung

In diesem Kapitel wird ein Überblick über die softwareseitige Implementierung der Instrumentsteuerung gegeben. Um die Steuerung der einzelnen Komponenten von Instrument und Experiment von einem eizelnen Rechner aus zu erlauben, wurde ein übergeordnetes LabView Projekt konzipiert und implementiert. Dabei wurde speziell darauf geachtet die einzelen Komponenten einfach und möglichst unabhängig voneinender zu halten. Ein Satz fixer Bindeglieder erlaubt die Kommunikation zwischen den Komponenten. Des weiteren wurde die Datenaufnahme neu konzipiert und von der LabView Seite getrennt, um hier einheitliche Daten- und Dateistrukturen zu fixieren.

## 3.1. Verwendete Hardware

Zur Steuerung der Geräte wurden zwecks Redundanz zwei baugleiche Rechner mit rackmontierbaren Gehäusen gekauft. Diese sollten auch softwareseitig immer auf dem selben Stand gehalten werden um auch wirklich jederzeit gegeneinander austauschbar zu sein. Als zentraler Datenspeicher und Syslog Server (genaueres hierzu siehe Kapitel 3.3) wurde ein rackmontierbares NAS (Network Accessed Storage) der Firma Synology[1] gekauft. Mit 4 Festplatten in einem RAID 10 Verband ist hier auch die Datensicherheit gewährleistet. Um hier auch Unabhängigkeit von äußeren Netzwerkausfällen zu erreichen wurde ein rackmontierbarer 16-facher Netzwerkswitch gekauft, der virtuell in mehrere unabhängige Netzwerkswitches getrennt werden kann. Hier wurden zwei Netzwerke eingerichtet, eines für das externe Netzwerk, das auch Zugang zum Internet ermöglicht, und ein internes Netzwerk, das die Unabhängigkeit von äußeren Netzwerken gewährleistet.

## 3.2. LabView

Die Entwicklungsumgebung LabView[2] von der Firma National Instruments bietet eine einfache und dennoch funktionsreiche Möglichkeit sowohl einfache, als

[1] Synology Inc., 3F-3, No.106, Chang An W. Rd., Taipei 10351, Taiwan

[2] National Instruments, 11500 North Mopac Expressway Austin, Texas 78759-3504 USA, http://www.ni.com/labview

auch sehr komplexe Regelungen zu programmieren. Zusätzlich bieten sowohl die mitgelieferte Programmbibliothek von National Instruments als auch die große Community rund um LabView vorgefertigte Lösungen für die meisten Probleme.

### 3.2.1. Variablen als Bindeglieder

Da es ein Ziel der gesamten Programmierung war alle Komponenten der Instrument- und Experimentansteuerung möglichst unabhängig zu halten, mussten gewisse minimale Bindeglieder definiert werden, um die Kommunikation zwischen den Komponenten und damit Unterprogrammen zu gewährleisten. Als Bindeglieder wurden SharedVariables von LabView gewählt, die somit für Ein- und Ausgabe aller Daten zu und von den einzelnen Programmen/Geräten zuständig waren. Für diese wurde zur Strukturierung eine eigene LabView Library (siehe variables.lvlib in Abbildung 3.1) angelegt, was insgesamt einige vorteilhafte Eigenschaften mit sich bringt:

- Sie erlauben feste Datentypen, die jedoch frei definiert werden können. Auch Cluster sind erlaubt die es ermöglichen mehrere unterschiedliche Datentypen (Arrays, Bool Werte, Strings etc.) in ein Element zusammenfassen.
- Die Positionierung der Variablen in einer LabView Library erlaubt es Variablen später einfach auszugliedern.
- SharedVariables erlauben Timestamping, was bedeutet, dass bei jeder Änderung der Variablenwerte automatisch Zeitstempel mitgeschrieben werden.
- Für zukünfitge Anwendungen besteht die Möglichkeit die Variablen über das Netzwerk zu publizieren und so von anderen Computern aufzurufen und zu verarbeiten.

### 3.2.2. Aufbau des LabView Projekts

Als erster Schritt zur Vereinheitlichung der LabView Programmstruktur wurde ein LabView Projekt angelegt. LabView Projekte fassen mehrere LabView Programme, Libraries, Controls etc. innerhalb eines Dateibaumes zusammen und ermöglichen so einen schnellen Überblick über alle Komponenten. Die Ordnerstruktur wurde in Anlehnung an GNU/Unix Softwareprojekte erstellt und ist in Abbildung 3.1 ersichtlich. Zusätzlich wurde dieser Ordner mit dem LabView Projekt unter Versionskontrolle mit der Software GIT [17] gehalten. Die notwendige Integration von LabView in GIT wurde in einer früheren Projektarbeit [18] entwickelt.

### 3.2.3. Jobstruktur

Zur Vereinheitlichung der Ansteuerung der diversen Geräte wurde ein generisches Gerät entwickelt, das verschiedene Aufgaben (Jobs) ausführen kann und auf mög-

```
/
├── config ............... Enthält die Konfigurationsdateien und
│                          das Programm zum einlesen und speichern
│                          dieser in den SharedVariables.
├── control .............. Enthält die control Dateien die für
│                          die SharedVariables benötigt werden.
│                          Diese legen die genauen Datentypen der
│                          SharedVariables fest.
├── lib .................. Die Library enthält Ordner mit den
│                          Namen der Job-Programme, welche die
│                          zahlreichen Unterprogramme enthalten.
├── jobs ................. Enthält die Job-Programme, die von
│                          der main.vi aufgerufen werden und
│                          direkt Daten messen bzw. ausgeben. Die
│                          Ordnernamen in /lib decken sich mit den
│                          Programmnamen.
├── visualisation ....... Enthält die grafischen Visualisierungen
│                          der gemessenen bzw. berechneten Daten.
│                          Diese Programme werden unter anderem
│                          direkt vom Hauptprogramm gestartet.
├── main.vi .............. Das Hauptprogramm, das die
│                          Konfigurationsdateien einlesen
│                          lässt, Variablen initialisiert, die
│                          verschiedenen Unterprogramme aufruft
│                          und Abhängigkeiten zwischen den
│                          Unterprogrammen beachtet.
├── variables.lvlib ..... LabView Library, die die Definitionen
│                          der SharedVariables enthält. Siehe dazu
│                          Kapitel 3.2.1.
└── ...
```

Abbildung 3.1.: Orndertruktur des LabView Projekts

lichst alle denkbaren Geräte anwendbar ist. Dieses generische Gerät besteht im Prinzip aus drei Sektionen, die der Reihe nach ausgeführt werden.

**Sektion 1, Initialisierung** In dieser Sektion wird das Gerät mithilfe der notwendigen Informationen aus dem Konfigurationsfile initialisiert und eventuell notwendige und für alle Jobs gültige Einstellungen werden getroffen. Außerdem wird hier eine Nachricht über die Initialisierung des Geräts mit den Informationen über Inhalt und Einheiten der Spalten zukünftiger Daten an Syslog (siehe Kapitel 3.3) geschickt.

**Sektion 2, Ausführung des aktiven Jobs** In dieser Sektion wird in einer Endlosschleife aus einer SharedVariable (siehe hierzu 3.2.1) der aktuell auszuführende Job ausgelesen und entsprechend ausgeführt. Standardmäßig sollte der Job „Idle“ ausgeführt werden, der nichts macht bzw. passiv Daten aus dem Gerät ausliest. Um Rechenressourcen zu sparen wurde in diesem Job ebenfalls ein Wartebefehl für 50 ms untergebracht der die Ausführung der äußeren Schleife verlangsamt. Abbruchbedingung für die äußere Endlosschleife ist immer entweder ein Fehler des Geräts, oder das drücken des Start/Stop Knopfes der main.vi (Ersichtlich in Abbildung 3.2a).

**Sektion 3, Schließen der Geräteverbindung** In dieser Sektion werden eventuell für die Beendigung des Geräts notwendige Aktionen ausgeführt und die Verbindung zu diesem Gerät geschlossen. Ebenfalls wird eine Nachricht an Syslog geschickt die das Ende der Messung kennzeichnet.

### 3.2.4. Konfigurationsdateien für LabView Programme

Bei der Arbeit mit grafischen Oberflächen zeigt sich im allgemeinen, und bei LabView speziell, oftmals das Problem der Handhabung der Konfigurationseinstellungen. Es wird oft ein sehr hoher Aufwand betrieben um eine grafische Oberfläche für die unterschiedlichen Konfigurationseinstellungen zu erstellen, die dann jedoch relativ selten genutzt wird, da die meisten Einstellungen sich nur sehr selten ändern. Zusätzlich besteht das menschliche Problem, dass Einstellungen in einem Laborbuch oder ähnlichem, zwecks Nachvollziehbarkeit des Experiments, dokumentiert werden müssen. Zur Lösung dieser Probleme wurden simple textuelle Konfigurationsdateien verwendet, die in LabView eingelesen und entsprechend verarbeitet werden. Die Konfigurationsdateien sind vom Format:

```
[sektion1]
; Kommentar
variable1 = wert1

[sektion2]
variable2 = wert2
```

```
variable3 = wert3
```

Allgemein ist dies aus der Windows Welt als „INI“ Dateiformat bekannt, weshalb die Dateien auch meistens die Endung „.ini“ tragen, und steht für Initialization bzw. Initialisierung. Dieses Format wurde gewählt, da es sehr weit verbreitet ist und von LabView vollständig unsterstützt wird. Das Format selbst wurde jedoch nie standardisiert, weshalb heute viele unterschiedliche Unterformate existieren. Um in LabView auf keine unerwünschten implementierungsspezifischen Überraschungen zu stoßen werden, in den Sektions- und Variablennamen ausschließlich Buchstaben, Zahlen und Unterstriche (_) verwendet.

Mit diesem Ansatz können zusätzlich die Kommentare genutzt werden, um Informationen über Einheiten der Werte und ähnliches zu dokumentieren. Um die Übersichtlichkeit zu gewährleisten, werden mehrere Konfigurationsdateien verwendet, von denen jede nur eine Sektion enthält. Zusätzlich werden zur Dokumentation jeglicher Änderungen an der Konfiguration beim Einlesen der Dateien (siehe hierzu Abbildung 3.2a, Kästchen 1) diese Änderungen über Syslog (siehe Kapitel 3.3) mitgeschrieben.

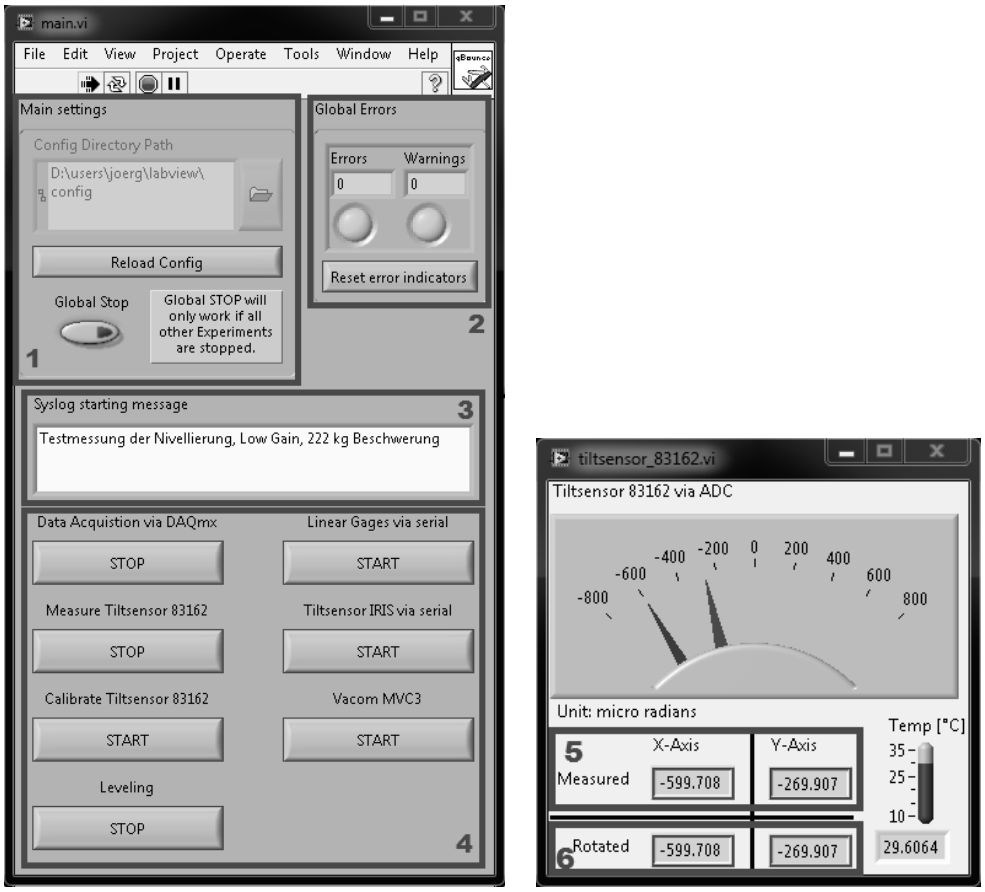
### 3.2.5. Zentrale Verwaltung der LabView Programme

Um die vielen unterschiedlichen Programme bzw. Jobs die für das qBounce Experiment notwendig sind möglichst einheitlich und einfach benutzen zu können, wurde ein zentrales Programm entwickelt mit dem es möglich ist die Konfigurationsdateien einzulesen, Fehlerberichte zu lesen und die unterschiedlichen Unterprogramme zu starten. Die grafische Aufarbeitung dieses Hauptprogramms ist zu sehen in Abbildung 3.2a. Die grafischen Oberflächen bzw. Visualisierungen der unterschiedlichen Unterprogramme und auch des Hauptprogramms sollten möglichst simpel und auf die wesentlichsten Elemente reduziert gehalten werden.

Wird in diesem zentralen Hauptprogramm der Start/Stop Knopf für ein Gerät gedrückt, wird das entsprechende Job-Programm für das Gerät im Hintergrund gestartet, welches das Gerät initialisiert und dann im „Idle“ Status verweilt, bis die Job Variable geändert wird. Optional für das Gerät wird ebenfalls ein kleines Programm gestartet, das die Ausgabe des Job Programms ausliest und visualisiert, wie beispielhaft in Abbildung 3.2b zu sehen ist.

### 3.2.6. Scriptmodus

Um die Jobs der einzelnen Geräte aufzurufen bzw. zu setzen wurde ein Scriptmodus entwickelt, mithilfe dessen die diversen Jobs der Geräte innerhalb eines textuellen Scripts gesetzt werden können. Ein Beispiel eines solchen Scripts wäre:




(a) Hauptprogramm (b) Tiltsensor

Abbildung 3.2.: Hauptprogramm und Visualisierung zu Tiltsensor mit Elektronik 83162

1. Einstellung des Ordners in dem die Konfigurationsdateien zu finden sind, sowie Knöpfe zum neu Laden der Konfiguration und zum Stoppen des Hauptprogramms.
2. Indikator über die Anzahl der gesammelten Fehler.
3. Nachricht, die beim Starten einer Messung an den Syslog Server gesendet wird. Siehe hierzu auch 3.3.2.
4. Knöpfe zum Starten bzw. Stoppen der einzelnen Geräte bzw. Komponenten.
5. Gemessene X und Y Verkippung des Winkelmessers.
6. Gedrehte X und Y Verkippung, siehe hierzu Kapitel 4.6.1.

Map_zur_Stufenvermessung.txt

```
1  beginfunction map
2    set_micos_position 1 20
3    if {!micos_pos 1 20} {wait_and_repeat 20}
4
5    label mapstart
6    measure_pi_517
7    if {!job pi517 idle} {wait_and_repeat 20}
8    syslog_data qbb micos1 pi517
9
10   if {!pf2_signal untergrund} {endfunction}
11
12   move_micos_relative 1 5
13   if {micos_moving 1} {wait_and_repeat 20}
14   if {micos_pos_less 1 121} {goto mapstart}
15 endfunction
16
17 label start
18 if {pf2_signal untergrund} {function map}
19 goto start
```

**Zeile 1** Kennzeichnet den Beginn einer Funktion namens „map“.
**Zeile 2** Setzt den Job des Lineartisches Micos 1 derart, dass dieser auf Position 20 fährt.
**Zeile 3** Überprüft, ob die Position des Lineartisches 1 bereits 20 ist. Das Ausrufezeichen negiert die Bedingung. Ist die Position somit ungleich 20, wird der Befehl „wait_and_repeat 20“ ausgeführt, der 20 Millisekunden wartet und dann den if Befehl erneut ausführt.
**Zeile 5** Setzt in dieser Zeile das Label „mapstart“, zu dem man mit dem Befehl „goto mapstart“ springen kann.
**Zeile 6** Setzt den Job der kapazitiven Abstandssensoren PI 517 derart, dass diese messen.
**Zeile 7** Wartet, bis der Messjob der Sensoren PI 517 beendet und der Job wieder „Idle“ ist.
**Zeile 8** Gibt die Werte der Position des Lineartisches Micos 1, sowie die Daten der letzten Messung der Kapazitiven Sensoren über Syslog mit dem Programmnamen „qbb“ aus.
**Zeile 10** Beendet die Funktion, falls das Signal des PF2 Instruments nicht mehr auf dem qBounce Wert für Untergrund steht.
**Zeile 12** Gibt dem Lineartisch Micos 1 den Befehl sich um 5 mm relativ zu seiner aktuellen Position zu bewegen.
**Zeile 13** Wartet, bis die Bewegung des Lineartisches Micos beendet ist.

**Zeile 14** Springt zum Label „mapstart“, falls die Position des Lineartisches Micos 1 weniger als 121 ist.
**Zeile 15** Kennzeichnet das Ende einer Funktion.
**Zeile 17** Setzt in dieser Zeile das Label „start“.
**Zeile 18** Startet die Funktion „map“, falls das Signal des PF2 Instruments auf dem qBounce Wert für Untergrund steht.
**Zeile 19** Springt zum label „start“.

Dieses kurze Script steht beispielhaft für eine Messung des quantum bouncing ball, und ist an sich nicht vollständig, demonstriert jedoch die Funktion des Scriptmodus. Die äußere Schleife, von Zeile 15 bis 17, läuft endlos und startet eine Map, falls der aktuelle Zyklus des PF2 auf „Untergrund“ steht. In der Map fährt der Lineartisch jeweils in Schritten von 5 mm weiter und misst die Abstände zu den Spiegeln mittels Kapazitiver Sensoren (PI 517). In jedem Schritt werden außerdem die Experimentdaten, sprich die Position des Lineartisches Micos1 und die gemessenen Abstände der Sensoren, über Syslog mit dem Programmnamen „qbb“ ausgegeben. Auf diese Weise können sehr einfach die Gerätedaten, die ebenfalls direkt über Syslog ausgegeben werden, und die Experimentdaten, die meist aus mehreren Geräten bestehen, getrennt werden. Der besondere Vorteil dieser Methode besteht darin, dass die Experimentdaten mehrerer Geräte so mit einem Zeitstempel vorhanden sind und nicht aus mehreren Dateien mit unterschiedlichen Zeitstempeln zusammengesammelt werden müssen.

Der Scriptmodus wurde derart gestaltet, dass er einfach und modular erweiterbar ist und so konzeptuell alle zukünftigen Geräte, die mithilfe der Jobstruktur implementiert sind, steuern kann.

### 3.2.7. Datenein- und Ausgabe über den ADC/DAC

Da die ADC/DAC Datenkarte nicht von mehreren Programmen gleichzeitig verwendet werden kann, dies jedoch für das qBounce Experiment notwendig ist wurde ein Unterprogramm entwickelt, das als zentrale Schnittstelle für die Datenkarte fungiert. Ein Nachteil der dadurch entsteht, ist, dass die Auslesegeschwindigkeit (von Herstellerseite angegeben mit 250.000 Datenpunkten pro Sekunde) stark reduziert werden muss, da sonst die Datenmengen zu groß werden. Da für die diversen qBounce Experimente die die Datenkarte verwenden jedoch nur Zeitspannen in der Größenordnung von 0.1 Sekunden und mehr relevant, sind stellt auch dies kein Problem dar.

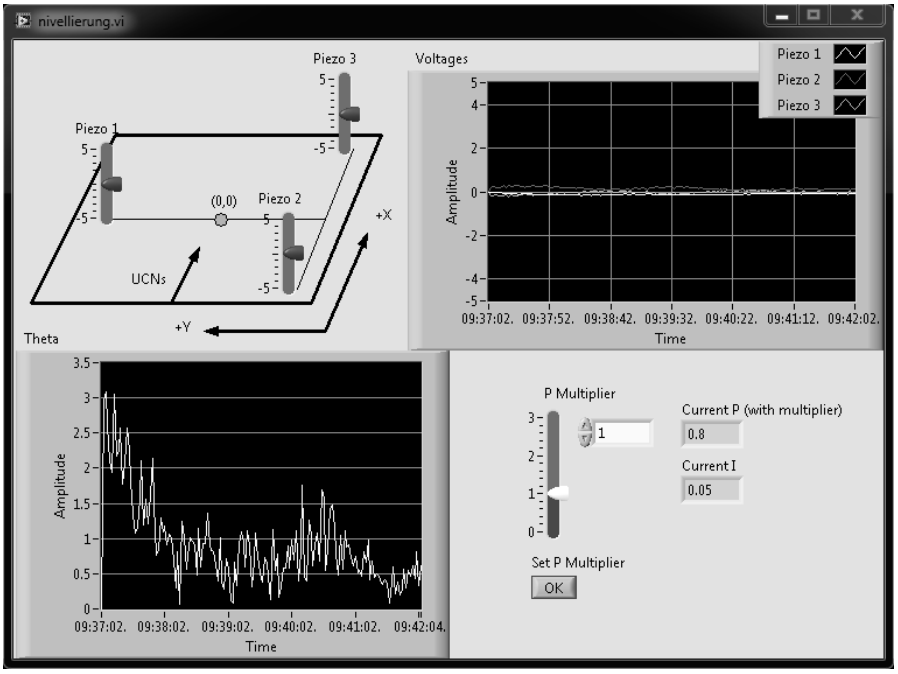



Abbildung 3.3.: Visualisierung zum Programm der Nivellierung

### 3.2.8. Nivellierung

Die mathematische Grundlage zur Berechnung der Nivellierung ist in Kapitel 4.6 gegeben. Zusätzlich erweist es sich als hilfreich die berechneten Werte vor der Ausgabe an den Vorverstärker in die Mitte des Regelbereichs zu verschieben um ein Driften des gesamten Aufbaus nach oben oder unten zu vermeiden. Die Visualisierung zur Nivellierung ist in Abbildung 3.3 zu sehen.

## 3.3. Syslog

Bei früheren Messreihen ergab sich immer wieder das Problem, dass Dateibenennungen, Datenformate, Dateiformate und Speicherorte schnell fragmentierten und die Organisation der Dateien, sowie die Auswertung der Daten immer schwieriger wurde. Es wurde daher auf das aus der Unix und Linux Welt bewährte Protokoll Syslog [19, 20] zurückgegriffen. Dieses ermöglicht es die Daten an einen zentralen Server zu senden, der wiederum sowohl die Daten- als auch die Dateistrukturierung übernimmt. Diese Methode bringt neben der vereinheitlichten Struktur einen weiteren Vorteil mit sich: Es ist möglich die Zeit des empfangenden Servers als Zeitstempel der Nachrichten zu verwenden. Dadurch werden gleichzeitige Events, die auf verschiedenen Rechnern aufgezeichnet werden, unabhängig von der Systemzeit

der Rechner mit gleichem absolutem Zeitstempel mitgeschrieben. Eine geringe Abweichung der Zeitstempel ergibt sich lediglich aus der Verzögerung im Netzwerk. Diese Verzögerung wurde mittels des Befehls „ping“ gemessen, was einen Wert von „<1 ms“ ergab. Da beim qBounce Experiment allgemein und speziell bei der Instrumentierung kaum Zeiten unterhalb des Sekundenbereichs interessant sind ist eine solche Verzögerung vernachlässigbar.

### 3.3.1. Zeilenformat Syslog

Die wichtigste Komponente der Syslog Spezifikation ist das Zeilenformat. Es definiert wie eine Zeile auszusehen hat, die an den Syslog Server geschickt wird. Dem Server obliegt dann die Verarbeitung und Sicherung der Daten in einem beliebigen Format. Für das qBounce Experiment werden als Format einfache Textdateien (siehe hierzu Kapitel 3.3.2) verwendet. In vielen Anwendungen, vor allem bei großen Firmen, ist es üblich die Daten direkt in Datenbanken abzulegen. Das Syslog Zeilenformat ist allgemein definiert als:

```
<Code>Zeitstempel ComputernameSender Programmname: Daten
```

**Code** Der Code setzt sich aus den Werten für die Facility und die Severity zusammen. Diese sind für die interne Verwendung von Syslog gedacht weshalb der Code meist nicht in der Ausgabe des Syslog Servers erscheint. Die Severity ist ein Wert zwischen 0 und 23, die speziell für die Verwendung auf Linux/Unix Systemen gedacht ist und für dieses Projekt nicht verwendet wurde. Die Facility erlaubt Werte zwischen 0 und 7 die, in aufsteigender Reihenfolge, definiert sind als „Emergency“, „Alert“, „Critical“, „Error“, „Warning“, „Notice“, „Info“ und „Debug“. Den numerischen Code Wert erhält man mit

$$Code = Facility + 8 * Severity. \tag{3.1}$$

Damit ist auch die Rückrechnung sehr einfach über

$$Severity = \lfloor Code/8 \rfloor, \tag{3.2}$$

$$Facility = Code - 8 * \lfloor Code/8 \rfloor. \tag{3.3}$$

Der Facility Code wurde für die Dateistrukturierung verwendet, siehe dazu Kapitel 3.3.2.

**Zeitstempel** Der Zeitstempel ist vom Format nicht streng definiert und es existieren unterschiedliche Standards hierzu. Für dieses Projekt wurde ein Zeitstempel nach dem ISO [21] Standard gewählt. Dieser hat das Format:

```
2014-03-20T17:57:00.000+01:00
```

welches sowohl das genaue Datum, die genaue Uhrzeit in beliebiger Genauigkeit der Sekunden, als auch die aktuelle Zeitzone enthält.

**ComputernameSender** Der Computername des Senders enthält die Information welcher Rechner im Netzwerk die Nachricht geschickt hat. Diese kann in Form des am Rechner eingestellten Hostnamens sein, oder in Form seiner IP Adresse. Sollten mehrere Computer Nachrichten an den syslog Server schicken, kann so unterschieden werden, welcher Rechner die entsprechende Zeile geschickt hat.

**Programmname** Mit Hilfe des Programmnamens kann unterschieden werden, welches Programm die entsprechende Zeile generiert und an den syslog Server geschickt hat. Für das qBounce Experiment wurde dieser Parameter genutzt um zu unterscheiden welches Gerät bzw. welches Modul die Zeile generiert hat.

**Daten** Die Daten enthalten eine beliebige Zeichenkette, die auch Zeilenumbrüche enthalten darf.

Auf diese Weise konnten die notwendigen Informationen zum Schreiben von Daten auf die gemessenen bzw. errechneten Daten und den Programmnamen reduziert werden. Zeitstempel und Computername wurden automatisch generiert. Der Facility und der Severity Code wurden mithilfe dreier Unterprogramme zum Senden von Fehlern bzw. Warnungen, Mitteilungen und der eigentlichen Daten wegabstrahiert.

### 3.3.2. Dateistruktur und Datenformat

Um die unterschiedlichen Informationen zu unterscheiden, wird der Syslog Server derart konfiguriert, dass er abhängig vom Facility Code in vier unterschiedliche Vorlagen an Dateinamen aufteilt:

| Facility | Dateiname |
|---|---|
| 0, 1, 2, 3 | qBounce.err |
| 4 | qBounce.warn |
| 5 | qBounce.<Programm>.notice |
| 6, 7 | qBounce.<Programm>.<Jahr>-<Monat>-<Tag> |

LabView unterscheidet intern bereits zwischen schwerwiegenden Fehlern (Error, zugeordnet den Facility Codes 0, 1, 2 und 3) die ein Programm meist zum Abbruch zwingen, und kleineren Fehlern (Warnings, zugeordnet dem Facility Code 4) die meist noch keinen vollständigen Abbruch erzwingen. Diese beiden Fehlerzustände werden von einem Unterprogramm gesammelt und ein Status über die Anzahl der gesammelten Fehler wird im Hauptprogramm angezeigt (siehe Abbildung 3.2a, Kästchen 2). Auf diese Weise sind auch Fehlerberichte langzeitig dokumentiert und können zur Fehlerbehebung herangezogen werden ohne die grafische Oberfläche unnötig zu überladen.

Über die Mitteilungen (Notices, Facility Code 5) werden spezielle Statusänderungen dokumentiert. Als Beispiel wird jede Änderung an den Konfigurationsdateien in der Datei `qBounce.config.notice` mitgeschrieben und auch jedes Starten bzw. Stoppen einer Messung wird, mit einer zusätzlichen Nachricht die im Hauptprogramm einstellbar (siehe Abbildung 3.2a, Kästchen 3) ist, in diesen Mitteilungen dokumentiert. Diese Mitteilungen automatisieren somit viele der sonst nötigen Einträge im Laborbuch die für die Nachvollziehbarkeit von Experimenten notwendig sind.

Die regulären Daten (Facility Codes 6 und 7) werden auf Grund der erhöhten Datenmenge in Dateien geschrieben die im Dateinamen den Programmnamen und das aktuelle Datum enthalten. Zusätzlich werden diese Dateien, der Übersichtlichkeit halber, in einen Unterordner mit dem Namen „Programmname“ geschrieben. Mit Hilfe der Mitteilungen wann eine Messung gestartet bzw. gestoppt wird, ist es somit sehr einfach den entsprechenden Datensatz zu finden und durch den sich in diesem Zeitraum befindlichen „HEADER“ sind die Spalten inklusive Einheiten eindeutig definiert. Ein Beispiel solcher Syslog Dateien ist im Folgenden zu sehen:

qBounce.leveling.notice

```
1  ...
2  2014−03−31T09:37:11.445+02:00 qBounceServer1 leveling: Starting
       measurement; Testmessung der Nivellierung, Low Gain, 222 kg
       Beschwerung
3  2014−03−31T09:37:11.446+02:00 qBounceServer1 leveling: HEADER (
       x_measured − x_target) (y_measured_y_target) [microradians]
4  2014−03−31T09:59:37.450+02:00 qBounceServer1 leveling: Stopping
       measurement
5  ...
```

leveling/qBounce.leveling.2014-03-31

```
1  ...
2  2014−03−31T09:37:11.446+02:00 qBounceServer1 leveling: 0.010476 −2.957027
3  2014−03−31T09:37:13.446+02:00 qBounceServer1 leveling: 1.378432 −2.758973
4  2014−03−31T09:37:15.445+02:00 qBounceServer1 leveling: 1.780773 −1.907344
5  2014−03−31T09:37:17.445+02:00 qBounceServer1 leveling: 0.935858 −1.887539
6  ...
7  2014−03−31T09:59:27.449+02:00 qBounceServer1 leveling: −0.009641 0.429684
8  2014−03−31T09:59:29.449+02:00 qBounceServer1 leveling: −0.311396
       −0.124865
9  2014−03−31T09:59:31.450+02:00 qBounceServer1 leveling: 0.251880 0.073188
10 2014−03−31T09:59:33.450+02:00 qBounceServer1 leveling: −0.492449 0.350463
11 2014−03−31T09:59:35.450+02:00 qBounceServer1 leveling: −0.271162
       −0.204086
```

### 3.3.3. Komprimieren der Daten und Backup

Um die Datenmenge insgesamt gering zu halten, wurde ein simples Ash [22] Script entwickelt, das Dateien, die reguläre Daten enthalten, komprimiert. Dieses Script wird einmal täglich zu einer bestimmten Zeit automatisch ausgeführt. Als Komprimierungsalgorithmus wurde GZip [23] gewählt, da es einerseits höhere Kompressionsraten als das aus der Windows Welt bekannte ZIP Format bietet und andererseits von Mathematica, siehe Kapitel 3.3.4, direkt eingelesen und verarbeitet werden kann. In einigen Tests ergaben sich Kompressionsraten zwischen 80 und 90 Prozent, was speziell bei Backups auf andere Server über das Internet von großem Vorteil ist.

Mit Hilfe der EDV Abteilung des Atominstituts der TU Wien wurde dieses Script erweitert, um nach der Kompression ein Backup der Daten auf den Servern des Atominstituts abzulegen. Der Quellcode des Scripts ist in Anhang A zu finden.

### 3.3.4. Mathematica Paket zum Einlesen der Daten

Um die mit Syslog gespeicherten Daten einfach verarbeiten zu können wurde ein Zusatzpaket für die Software Wolfram Mathematica[3] entwickelt, das die Daten möglichst automatisiert einliest und zur Weiterverarbeitung bereitstellt. Der Quellcode zu diesem Paket ist ersichtlich in Anhang B.

[3]Wolfram Research, The Wolfram Centre, Lower Road, Long Hanborough, Oxfordshire, OX29 8FD, United Kingdom, http://www.wolfram.com/mathematica

# 4. Nivellierung des Experiments

In diesem Kapitel wird die Ausrichtung des Experiments normal auf den Erdschwerpunkt erläutert. Es werden die dazu verwendeten Winkelmesser, die verwendete Elektronik sowie die Piezoelemente zur Steuerung der Lage beschrieben. Ebenso wird ein Überblick über den Regelkreis sowie dessen Grenzen und Genauigkeit gegeben. Zuletzt wird die erhaltene Genauigkeit der Regelung mit denen der Vorjahre verglichen.

## 4.1. Grundlagen

Für das qBounce Experiment ist eine Ausrichtung der Neutronenspiegel normal auf den Erdschwerpunkt besonders wichtig, da die Neutronenzählraten mit wenigen zehn Neutronen pro Stunde sehr gering ist und daher eine Messzeit von vielen Tagen notwendig wird. Eine Verkippung der Spiegel würde die Erdbeschleunigung um einen Faktor $cos\theta$ ändern und die Neutronen würden abgelenkt. Änderungen oder ein Driften der Verkippung wäre für das Experiment besonders störend.

## 4.2. Regelkreis

Für die Nivellierung des Experiments wird ein Regelkreis, bestehend aus einem Winkelmesser mit dazugehöriger Elektronik, einem Analog-Digital-Wandler (ADC), einem Digital-Analog-Wandler (DAC) und drei Piezoelementen mit zugehörigem Vorverstärker verwendet. Das analoge Signal des Winkelmessers wird mit dem ADC eingelesen und mittels eines LabView Programms in die notwendige Dehnung der Piezoelemente umgerechnet um das Experiment in der korrekten Lage zu halten. Die berechneten Werte werden dann über den DAC an den Vorverstärker der Piezoelemente weitergegeben. Es entsteht so ein geschlossener Regelkreis, siehe Abbildung 4.1, der in einem vordefinierten Intervall die Lage des Experiments angleicht.

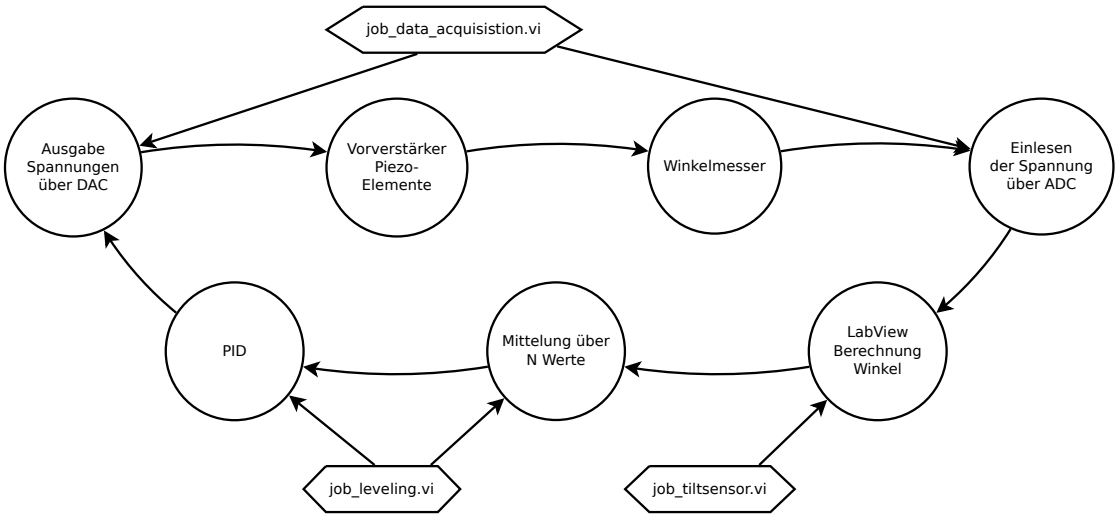



Abbildung 4.1.: Regelkreis der Nivellierung inklusive der steuernden Programme

## 4.3. ADC/DAC Datenkarte

Als zentrales Element der Instrumentsteuerung dient eine ADC/DAC Platine der Firma National Instruments[1], Modell „NI USB-6229 OEM“ . Da diese Platine eine eigene Spannungsversorgung benötigt, musste getestet werden welchen Einfluss eine Änderung der Spannungsversorgung auf das Messergebnis hat. Zu diesem Zweck wurde für die Spannungsversorgung ein Schaltnetzteil verwendet, dessen Ausgangsspannung manuell zwischen 11 und 30 Volt variiert wurde. Die Ausgangsspannung des DAC wurde per Software auf 5 mV gesetzt und mit einem Multimeter der Firma Fluke[2] Typ „8846A 6-1/2 Digit Precision Multimeter“ gemessen. Die Messgenauigkeit dieses Multimeters liegt laut Hersteller bei 0.1 µV, was für die 16 bit Auflösung des DAC (enspricht messbaren Schritten von 0.3 µV, siehe hierzu Formel 4.1), ausreichend ist.

Das Ergebnis der Messungen ist in Abbildung 4.2 ersichtlich. Es ergibt sich eine mittlere Ausgangsspannung von $\overline{U} = 5.15 \pm 0.007$. Man sieht sofort, dass die Ausgangsspannung von der verwendeten Eingangsspannung nur sehr gering abhängt. Es ist daher selbst für sehr genaue Messungen mit dem ADC/DAC nur eine Spannungsversorgung mit einer Restwelligkeit im mV Bereich notwendig.

[1]National Instruments, 11500 North Mopac Expressway Austin, Texas 78759-3504 USA, http://www.ni.com

[2]Fluke, Liebermannstraße F01 A-2345 Brunn am Gebirge, Österreich, http://www.fluke.com

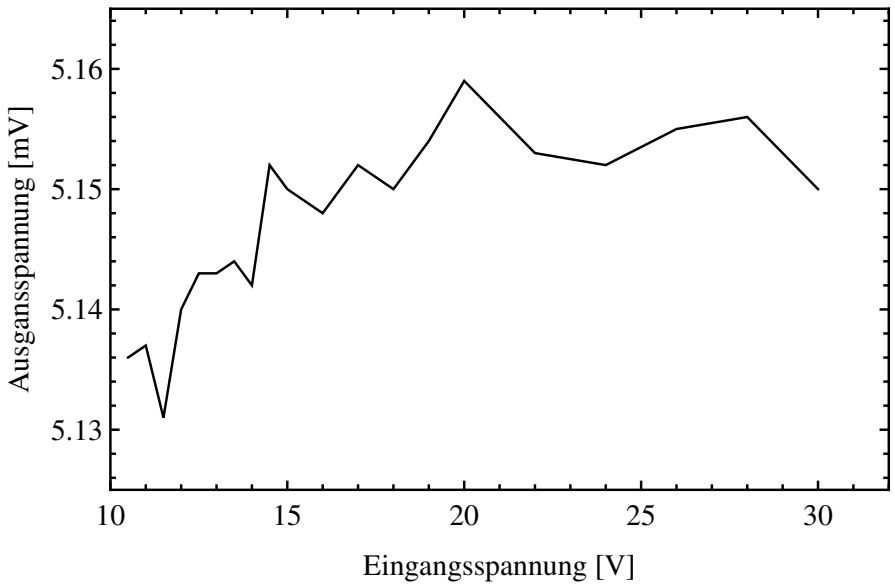



Abbildung 4.2.: Abhängigkeit der Ausgangs- von der Eingangsspannung des DAC

## 4.4. Winkelmesser

Zum Messen der Auslenkung werden Winkelmesser der Firma Tech-Sys Instruments[3], Modell „755“ mit der dazugehörigen Elektronik Modell „83162“ verwendet. Das Funktionsprinzip dieser Winkelsensoren ist mit dem einfacher Wasserwaagen vergleichbar und in Abbildung 4.3 genauer ersichtlich. Die Winkelsensoren selbst bieten laut Handbuch eine Auflösung von „0.1 µrad or better“, bei einem maximal messbaren Bereich von ± 8000 µrad. Die zugehörige Elektronik bietet die Umschaltmöglichkeit zwischen „Low gain“ und „High gain“, mit dem Effekt einer 10:1 Übersetzung der Genauigkeit und des auslesbaren Bereiches. Somit verbleibt bei der Stellung „High gain“ ein maximal messbarer Bereich von ± 800 µrad bei einer vom Hersteller angegebenen Skalierung von 0.1 µrad/mV. Bei der Stellung „Low gain“ steigt der maximal messbare Bereich auf ± 8000 µrad bei einer Skalierung von 1.0 µrad/mV. Die Winkelsensoren wurden mit der zugehörigen Elektronik und Verkabelung von der Herstellerfirma geeicht und die notwendigen Skalenfaktoren zur Umrechnung der Spannungswerte in Winkel angegeben.

### 4.4.1. Rauschbestimmung

Um nun die Genauigkeit des Systems, bestehend aus Winkelmesser mit Ausleseelektronik, ADC und den zugehörigen Spannungsversorgungen zu bestimmen

[3]Tech-Sys Instruments, Floridalane 64, 1180 Brussels, Belgium, http://www.tech-sys.eu

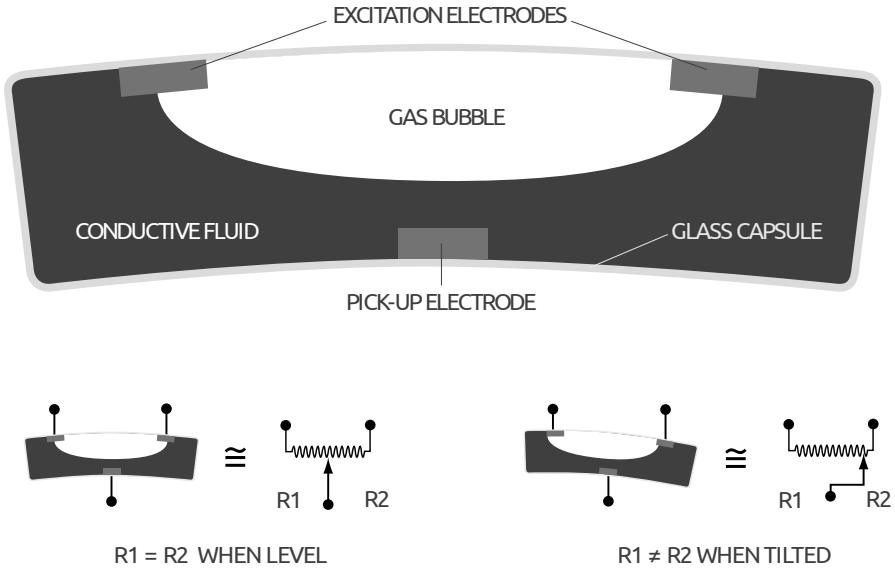



Abbildung 4.3.: Funktionsprinzip der Winkelsensoren 755 der Firma Tech-Sys Instruments (Quelle: [24]).

wurde mehrere Male über Nacht eine Leermessung vorgenommen um das Rauschen des Gesamtsystems zu bestimmen. Ein Beispiel einer solchen Messung ist in Abbildung 4.4 zu sehen, an der mehrere Details erkennbar sind.

- Die gesamte Platte auf der der Winkelmesser gelagert war, ist über Nacht leicht eingesunken. Dies dürfte an der Dreipunktlagerung und dem Laminatboden liegen.
- Zwischen ca. 20 und 23 Uhr sind starke Ausschläge erkennbar. Dies liegt nach etwas Nachforschung an zwei Kolleginnen, die in dieser Nacht den Büroraum zum Basteln eines Geschenkes genutzt haben.
- Zwischen ca. 00 und 05 Uhr war das System am ruhigsten. Diese Zeit wurde auch für die Rauschbestimmung herangezogen.
- Ab ca. 5 Uhr morgens verbreitert sich das Spektrum merklich. Dies dürfte auf den vermehrten Verkehr ab dieser Uhrzeit auf der Autobahn vor dem Institutsgebäude zurückzuführen sein.

### Auswertung der Daten

Für die Auswertung müssen die Datensätze erst normiert werden. Dazu wurden zwei Ansätze verfolgt und miteinander verglichen:

- Fit einer Kurve niedriger Ordnung an die Daten und Subtraktion der Kurvenwerte von den Datenpunkten.
- Mittelwertbildung von Gruppen von N Datenpunkten und Subtraktion des

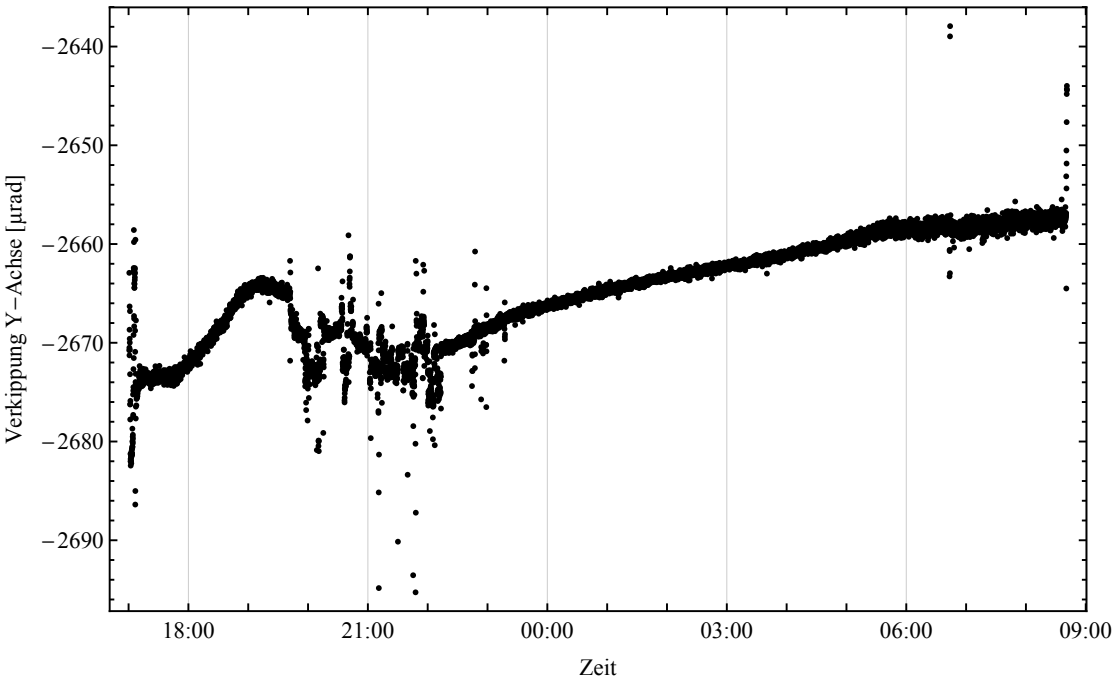



Abbildung 4.4.: Beispiel einer Rauschmessung

Mittelwerts von diesen.

Für lange Messungen über mehrere Stunden mit vielen tausend Datenpunkten erwies sich die erste Methode als brauchbar. Diese Daten erwiesen sich als weitestgehend linear und so konnten Kurven erster bis zweiter Ordnung leicht an die Daten gefittet werden. Kürzere Messungen unter einer Stunde erwiesen sich als problematischer, da hier schon kleine Schwankungen (z.B. durch zufallende Türen im Institutsgebäude) große Unlinearitäten mit sich brachten. Insgesamt erwies sich somit die zweite Methode als besser.

Nach Normierung der Datensätze ist es nun möglich die Daten auf eine gaussförmige Verteilung zu überprüfen und die Standardabweichung zu bestimmen.

## 4.5. Auswertung

Um das vorhandene System aus Winkelmesser, Elektronik, ADC, DAC und Piezo Elementen auszuwerten, muss die Genauigkeit jedes Elements separat analysiert werden.

### 4.5.1. Piezo Elemente

Die Piezo Elemente, die verwendet werden, um das Experiment zu heben bzw. zu senken stammen von der Firma Piezomechanik[4]. Die für die Dreipunktlagerung nötigen Aktoren waren vom Typ „PSt1000/25/80“ mit dem dazugehörigen Vorverstärker Typ „SVR 1000/3“. Der maximale Hub der Piezoaktoren beträgt für den Spannungsbereich des Vorverstärkers von 0 bis 1000 V 80 µm was in etwa einer Auslenkung des Systems von ± 100 µrad entspricht.

Das Rauschen der Vorverstärker ist mit 1 mVpp (Peak to Peak Voltage) angegeben. Für den gegebenen Spannungsbereich und die gegebene maximale Auslenkung ergibt dies ein Rauschen der Piezo Elemente von 0.08 nm/mV, beziehungsweise für den gegebenen Aufbau eine Verkippung von ca. 0.23 nrad/mV. Das Rauschen des Vorverstärkers ergibt somit um Größenordnungen kleinere Verkippungen als durch den Winkelmesser auflösbar wäre und ist somit vernachlässigbar.

### 4.5.2. ADC

Die Auflösung des ADC ist von Herstellerseite mit 16 bit in den möglichen Bereichen von ±10, ±5, ±1 und ±0.2 V angegeben. Für den größten Messbereich ergibt sich somit die maximale Auflösung in Schritten von:

$$\frac{10-(-10)}{2^{16}-1} = 0.0003\,\mathrm{V} \tag{4.1}$$

Da der ADC/DAC zusätzlich für weitere Komponenten des Experiments verwendet werden soll ist es nicht möglich den Messbereich zu verkleinern und somit die Auflösung zu erhöhen. Da jedoch die Skalenfaktoren der Platine 83162 für die Einstellung „High Gain“ als 0.1 µrad/mV angegeben sind und die Auflösung der Winkelmesser 0.1 µrad ist, ist eine Auflösung von 0.3 mV ausreichend. Für die Einstellung „Low Gain“ wäre der Skalenfaktor 1.0 µrad/mV, was hieße, dass eine Auflösung des ADC von mindestens 0.1 mV notwendig wäre.

### 4.5.3. DAC

Für die Auflösung des DAC von 16 bit im Spannungsbereich von ± 10 V ergibt sich eine Schrittweite von 0.3 mV. Somit ergibt ein Schritt des DAC eine Änderung der Verkippung von 6.9 nrad. Die maximale Auflösung des DAC ist somit ebenfalls um Größenordnungen besser als die Auflösung der Winkelmesser und ist somit vernachlässigbar.

[4]Piezomechanik, Dr. Lutz Pickelmann GmbH, Berg am Laim Str. 64, 81673 München, Germany, http://www.piezomechanik.com

### 4.5.4. Ergebnis

Es zeigt sich also, dass die einzelnen Komponenten der Instrumentsteuerung qualitativ die notwendigen Anforderungen erfüllen um die bestmögliche Auflösung zu bieten. Die Komponente mit dem stärksten Rauschverhalten stellt der Winkelsensor mit zugehöriger Elektronik dar. In mehreren Messungen über jeweils mehrere Stunden wurde das Rauschen auf ca. 0.3 µrad bestimmt.

## 4.6. Nivellierung

Zur Nivellierung des Instruments wurden mehrere mögliche Methoden in Betracht gezogen und verglichen (siehe auch Abbildung 4.6):

- Nivellierung um den Koordinatenmittelpunkt (siehe Abbildung 4.5).
- Nivellierung um einen beliebigen, variablen Punkt der Ebene mit dem Ziel den einstellbaren Winkelbereich zu maximieren.
- Nivellierung um den Ansatzpunkt des Strahlrohres.
- Nivellierung um den Schnittpunkt der Piezoachsen (diese Methode war bis vor dieser Diplomarbeit implementiert).

Die Nivellierung um den Ansatzpunkt des Strahlrohres hätte den Vorteil, dass man den Anschluss an das evakuierte Rohr der Ultrakalten Neutronen in der Höhe konstant halten könnte und so mögliche Lecks im Vakuumsystem ausschließen bzw. minimieren könnte. Ein Nachteil dieser Methode wäre ein großer Verlust an Reichweite des einstellbaren Winkels.

Aus Gründen der Einfachheit wurde die Nivellierung um den Koordinatenmittelpunkt implementiert. Ein Diagramm des Aufbaus der Instrumentierung mit Wahl der Achsenrichtungen sowie Koordinatenmittelpunkt ist in in Abbildung 4.5 ersichtlich.

Die Berechnung der notwendigen Einstellung der Piezoelemente um eine gemessene Verkippung auszugleichen ist sehr schnell aus Abbildung 4.7 abzulesen. Sie ergibt sich für gegebene Abstände x und y der Piezo Elemente zueinander und gemessene Verkippungen $\phi_x$ und $\phi_y$ zu

$$\begin{aligned} 2\Delta z_1 &= y \tan \phi_y \\ l &= x \tan \phi_x \\ \Delta z_2 &= \Delta z_x - l \\ \Delta z_3 &= \Delta z_x + l \end{aligned} \tag{4.2}$$

Die Abweichung $\Delta z_x$ ergibt sich auf Grund der Dreipunktlagerung und des gewählten Koordinaten-Mittelpunktes zu $\Delta z_x = \Delta z_1$. Diese Höheneinstellung kann mit Hilfe der gegebenen maximalen Auslenkung der Piezo Elemente leicht in die

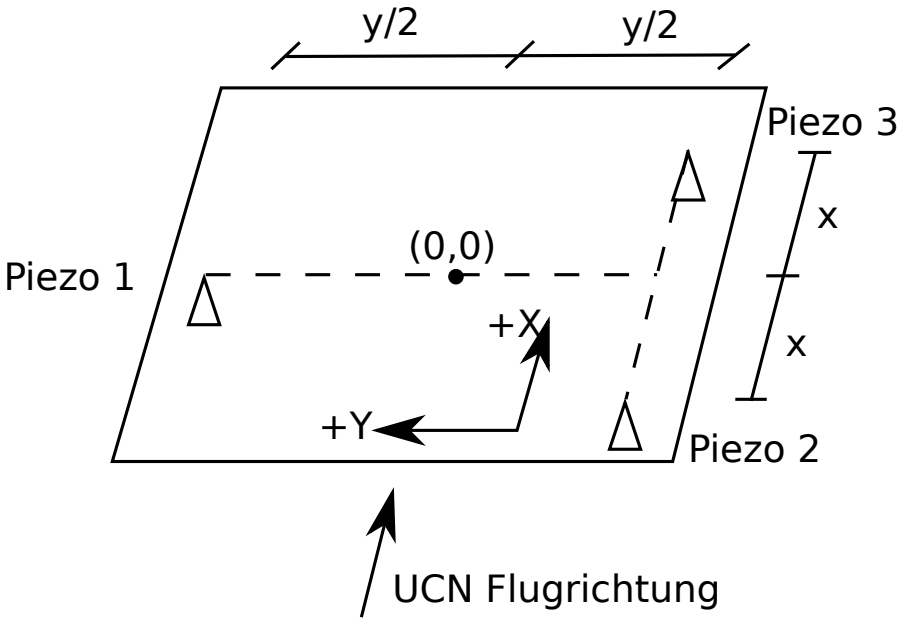



Abbildung 4.5.: Schematische Darstellung zum Aufbau der Nivellierung mit dem gewählten Koordinatenmittelpunkt, wobei für den aktuellen Aufbau x = y = 0.35 cm ist.

Eingangsspannung des Vorverstärkers umgerechnet werden, da diese zueinander linear skalieren.

### 4.6.1. Ausrichtung des Winkelmessers

Die drei Auflagepunkte der Piezos legen die Achsen der Kipprichtungen fest, wie in Abbildung 4.5 zu sehen ist. Praktisch ist es kaum möglich die Achsen des Winkelmessers und die der Kipprichtungen parallel zu legen, da dies bei der hohen Genauigkeit der Winkelmesser eine sehr gute Mechanik bräuchte. Daher ist eine Kalibrierung zur Bestimmung des Korrekturwinkels notwendig, um den die Achsen des Winkelmessers gegenüber denen des Tisches verschoben sind. Die Schematik der Kalibrierung ist ersichtlich in Abbildung 4.8, wobei $X_T$ und $Y_T$ die Achsen des Tisches und $X_W$ und $Y_W$ die Achsen des Winkelmessers bezeichnen. Daraus ergibt sich nach kurzer Überlegung der Korrekturwinkel $\theta$ zu

$$\begin{aligned} \phi_x &= \phi \sin(\theta)\,, \\ \phi_y &= \phi \sin(90-\theta)\,, \\ \frac{\phi_x}{\phi_y} &= \frac{\sin(\theta)}{\sin(90-\theta)} = \frac{\sin(\theta)}{\cos(\theta)} = \tan(\theta)\,. \end{aligned} \tag{4.3}$$

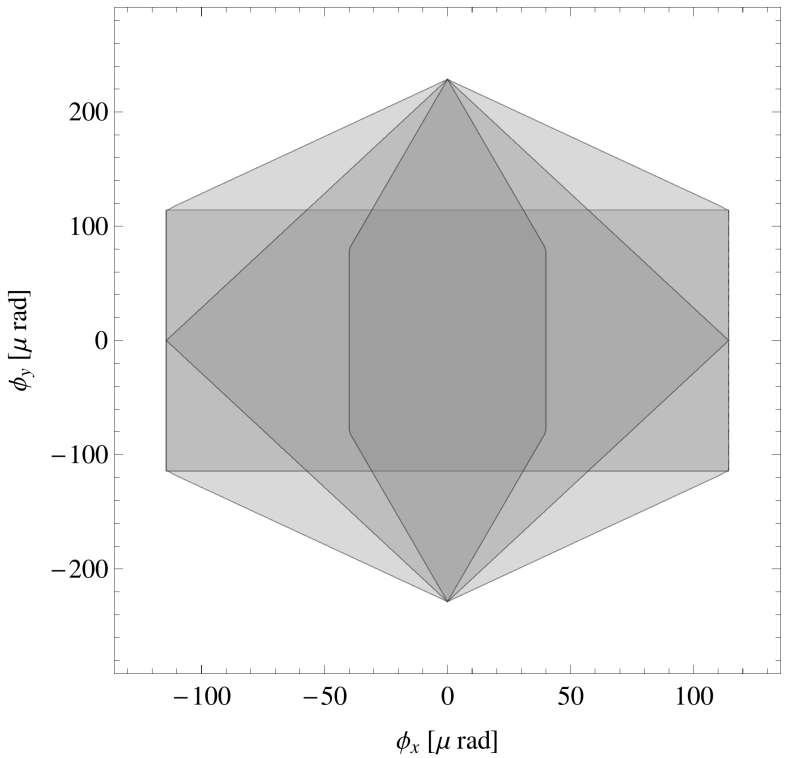



Abbildung 4.6.: Vergleich der möglichen maximalen korrigierbaren Schiefstellungen der Nivellierung. Das dunkelste Sechseck in der Mitte ergibt sich bei dem Drehpunkt um den Ansatzpunkt des Strahlrohres (-0.65,0), die Raute ergibt sich bei dem Drehpunkt um den Granitmittelpunkt (0,0), das Rechteck ergibt sich bei einem Drehpunkt um den Piezoachsenschnittpunkt (-0.175,0) und das äußerste und größte Sechseck ergibt sich bei einem dynamisch variablen Drehpunkt entlang der Y-Achse.

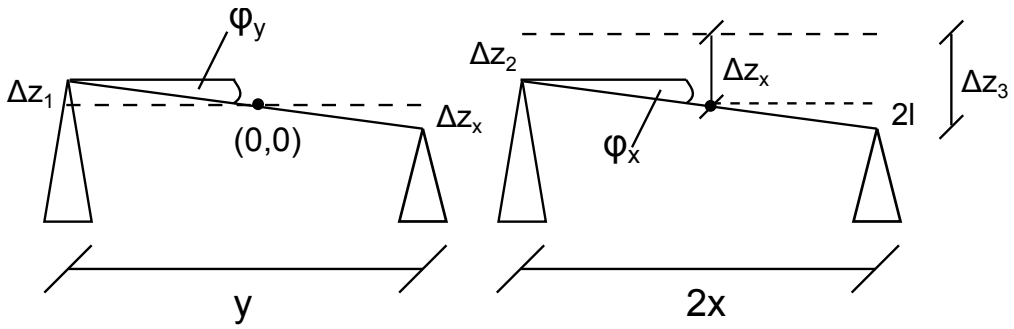



Abbildung 4.7.: Schematische Darstellung zur Berechnung der Nivellierung

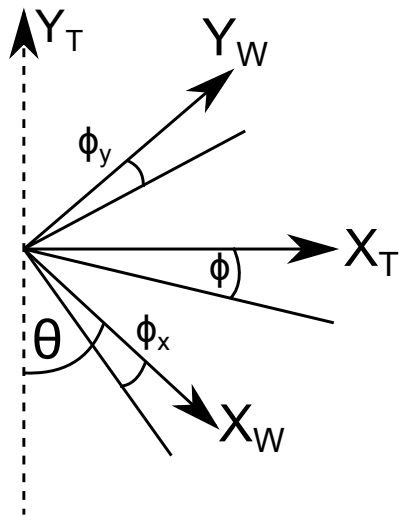



Abbildung 4.8.: Schematische Darstellung zur Ausrichtung der Winkelmesser

Implementiert wurde dies indem die Auslenkung von Piezo Nummer 1 zwischen seinem Maximum und Minimum in $N$ Schritten abgerastert und die Änderung der Winkel $\phi_x$ und $\phi_y$ aufgezeichnet und gemittelt wird. Der Abweichende Winkel $\theta$ ergibt sich dann zu

$$\theta = \arccos\left(\frac{\sum_{i=1}^{N} \Delta\phi_x^i / N}{\sum_{i=1}^{N} \Delta\phi_y^i / N}\right) \tag{4.4}$$

Da diese Methode nur Winkel zwischen $\pm 90°$ ergibt muss anhand der Mittelwerte von $\Delta\phi_x$ und $\Delta\phi_y$ auf den ganzen Kreis korrigiert werden.

### 4.6.2. PID Regelung

Zur Nivellierung des Instruments wird eine sogenannte PID-Regelung [25] verwendet. Diese Regelung besteht aus einem Proportionalen- (P), einem Integralen- (I) und einem Differentiellen- (D) Anteil. Die jeweiligen Anteile sind direkt aus der Formel ersichtlich:

$$u(t) = Pe(t) + I\int_0^t e(\tau)d\tau + D\frac{d}{dt}e(t) \tag{4.5}$$

Hier steht $e(t)$ für die Regelabweichung zwischen Soll- und Sensor- bzw. Ist-Wert. Eine PID Regelung dient dem Zweck, einen gemessenen Sensorwert mittels eines Regelkreises an einen gewünschten Sollwert anzunähern. Durch den proportionalen

Anteil wird der gemessene Sensorwert um einen Bruchteil der Differenz zwischen aktuellem und erwünschtem Wert verändert. Der integrale Anteil berücksichtigt vorhergegangene Werte und verhindert somit einerseits eine Überregulierung bei kurzfristigen starken Änderungen und reagiert andererseits mit stärkerer Änderung, falls die Regelung für längere Zeit nicht oder zu schwach reagiert. Der differentielle Anteil bezieht die Steigung der Regelung mit ein und betrachtet somit anschaulich zukünftige Werte. Für die Implementierung eines PID Algorithmus in LabView wurden Beispielprogramme der Herstellerfirma National Instruments [26] verwendet.

Da ein differentieller Anteil bei kurzen Ausschlägen schnell zu Überreaktionen führen kann und so ein starkes Rauschen des Sensorsignals verstärken würde, wird er für die qBounce Instrumentierung Null gesetzt und man erhält so effektiv eine PI Regelung. Um ein zu starkes Nachschwingen zu verhindern wird der integrale Anteil meist sehr klein gehalten und der proportionale Anteil wird nach Einschwingen des Systems ebenfalls klein gesetzt.

Spezielle Anforderungen an das System ergeben sich durch das hohe Gewicht, das nivelliert werden muss, wodurch das System sehr träge wird, und das starke Rauschen des Systems. Das Rauschverhalten ergibt sich hier aus mehreren Faktoren:

- Das Rauschen der Winkelmesser selbst.
- Das elektrische Rauschen der Messelektronik.
- Das Rauschen, das durch die Personen und Experimente der Umgebung erzeugt wird.

Durch die mitunter sehr starken Ablenkungen, die schon durch eine Person, die an dem Instrument vorbeigeht, verursacht werden, darf der proportionale Anteil der Regelung nicht zu klein sein um ein schnelles Nachregeln zu ermöglichen. Durch die Trägheit des Systems ist jedoch ein Nachregeln nur alle paar Sekunden möglich. Nach vielen Langzeittests war es jedoch möglich sehr gute P, I und D Werte für das Regelsystem zu finden.

$$\begin{aligned} P &= 0.8 \\ I &= 0.05 \\ D &= 0 \end{aligned} \tag{4.6}$$

### 4.6.3. Schaltung High/Low Gain

Die Platine des Winkelmessers bot die Möglichkeit über manuelle Schalter zwischen High und Low Gain zu wechseln, bzw. einen Filter ein- und auszuschalten. Der Filter mittelt elektronisch über mehrere Messwerte und erlaubt es so kurze Ausschläge wegzufiltern. Da jedoch die genaue Art der Mittelung und vor allem

die Dauer nicht bekannt sind, wurde diese Einstellung im Zuge dieser Arbeit nicht weiter untersucht und der Filter standardmäßig auf „Aus“ gestellt. Die Einstellung „High Gain“ erlaubt gegenüber der Einstellung „Low Gain“ eine um einen Faktor 10 erhöhte Winkelauflösung bei einem gleichermaßen großen Verlust des messbaren Winkelbereichs. Für das qBounce Instrument wäre es wünschenswert immer im Bereich „High Gain“ zu messen, da hier der Winkelbereich von $\pm 800\,\mu$rad immer noch ausreichend wäre (siehe 4.5) und die erhöhte Winkelauflösung von Vorteil ist. Da jedoch schon leichte Erschütterungen Ausschläge von über $800\,\mu$rad erzeugen können wurde die manuelle Schaltung durch eine elektrisch schaltbare ersetzt.

Mit Hilfe der Elektronikwerkstatt der TU Wien wurden die manuellen Schalter auf der Platine durch zwei 5 V schaltbare Relais ersetzt, die von den digitalen TTL Ausgängen der ADC/DAC Platine schaltbar sind. Auf diese Weise können beide Einstellungen (Filter und Gain) auf Wunsch per Software geändert werden. Da Relais für die Schaltung kleine Spulen verwenden, die dementsprechend elektromagnetische Felder erzeugen, wurden einerseits die Standardschaltungen (bei nicht stromdurchflossenem Relais) mit Bedacht gewählt und andererseits wurden die Relais in gewissem Abstand von der Winkelmesserplatine plaziert. Die Filterschaltung wurde standardmäßig auf „Aus“ gestellt und wird derzeit von Softwareseite nicht geschaltet. Die Gain Schaltung wurde standardmäßig auf „High“ gestellt und wird per Software bei Über- bzw. Unterschreiten eines einstellbaren Winkels für mehrere Messiterationen umgeschaltet.

Da bei den früheren Winkelmessern und der Elektronik bei Umschaltung zwischen High- und Low Gain ein Offset des gemessenen Winkels von mehreren $100\,\mu$rad festgestellt wurde, wurde dies auch bei der neuen Elektronik untersucht. In längeren Messreihen wurde jeweils nach mehreren Sekunden umgeschaltet und das Ergebnis auf vorhandene Stufen untersucht. Es ergaben sich kleine Stufen von wenigen $\mu$rad die in ihrer Größe variierten. Auf Grund des variablen und sehr geringen Offset wurde dieser nicht weiter in der Software berücksichtigt.

### 4.6.4. Analyse der Regelung

Die Regelung für das qBounce Instrument wurde anhand einer relativ leichten Stahlplatte, die auf die Piezo Elemente aufgelegt wurde, überprüft und erwies sich schnell als funktionell. Der eigentliche komplette Aufbau behinhaltet jedoch eine 350 kg schwere Granitplatte mit der darauf aufgesetzten Vakuumkammer, in der sich das Experiment befindet. Das Gesamtgewicht, das im Experimentierbetrieb zu nivellieren ist beträgt damit ca. 1 t. In den nächsten Jahren soll die Vakuumkammer noch erweitert werden, wodurch sich das Gesamtgewicht des Experiments noch signifikant erhöhen wird.

Um die Regelung unter erhöhter Belastung zu testen wurde die Stahlplatte mit Bleigewichten im Ausmaß von 222 kg beschwert. Die Ergebnisse einer solchen Mes-

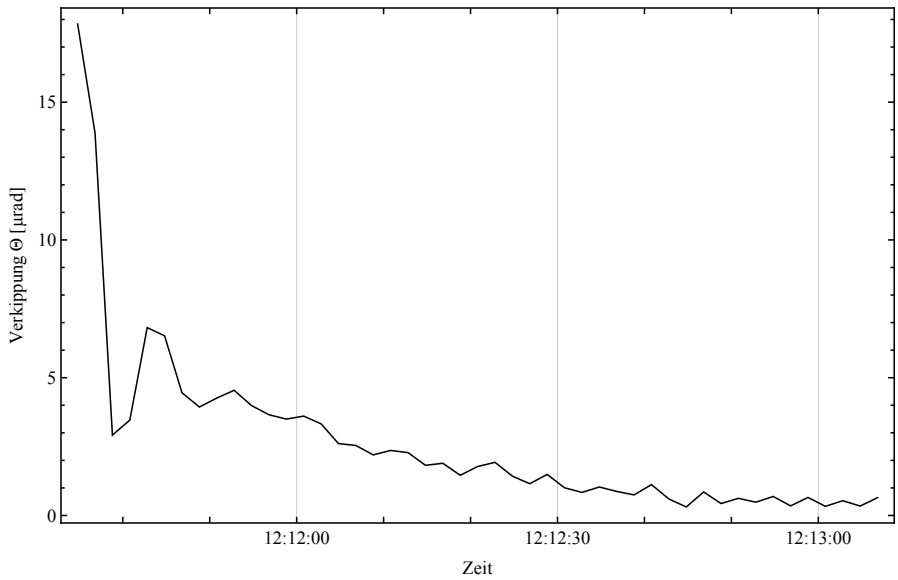



Abbildung 4.9.: Einschwingvorgang mit 222kg Beschwerung

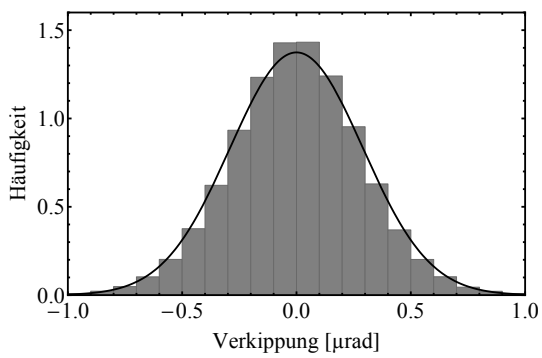
sung über eine Arbeitswoche sind in Abbildung 4.10 ersichtlich. Für ein P von 0.8 und I von 0.05 ergaben sich die Standardabweichungen $\sigma_{x,y,\theta}$, eine Mittelwert der Verkippung $\bar{\theta}$ bzw. eine wahrscheinlichste Verkippung $\widetilde{\theta}$ von

$$\begin{aligned} \sigma_x &= 0.29\,[\mu\text{rad}], \\ \sigma_y &= 0.24\,[\mu\text{rad}], \\ \sigma_\theta &= 0.19\,[\mu\text{rad}], \\ \bar{\theta} &= 0.33\,[\mu\text{rad}], \\ \widetilde{\theta} &= 0.25\,[\mu\text{rad}]. \end{aligned} \tag{4.7}$$

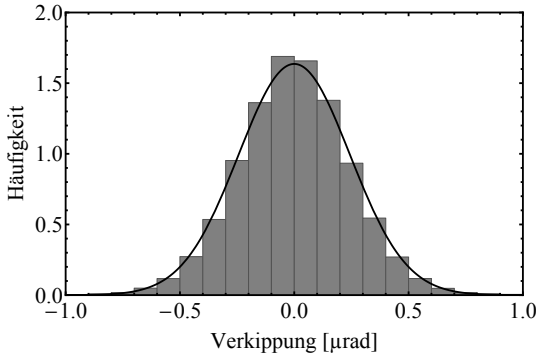
Die Fits der Verkippungen der X- und Y-Achse zeigen eine perfekt gaussförmige Verteilung, deren Standardabweichung erst in der fünften Nachkommastelle von der direkt errechneten Standardabweichung abweicht. Die rms-Verkippung $\theta = \sqrt{\Delta x^2 + \Delta y^2}$ folgt einer Rayleigh Verteilung, wie in Abbildung 4.10c ersichtlich ist.

### 4.6.5. Vergleich mit Ergebnissen der Vorjahre

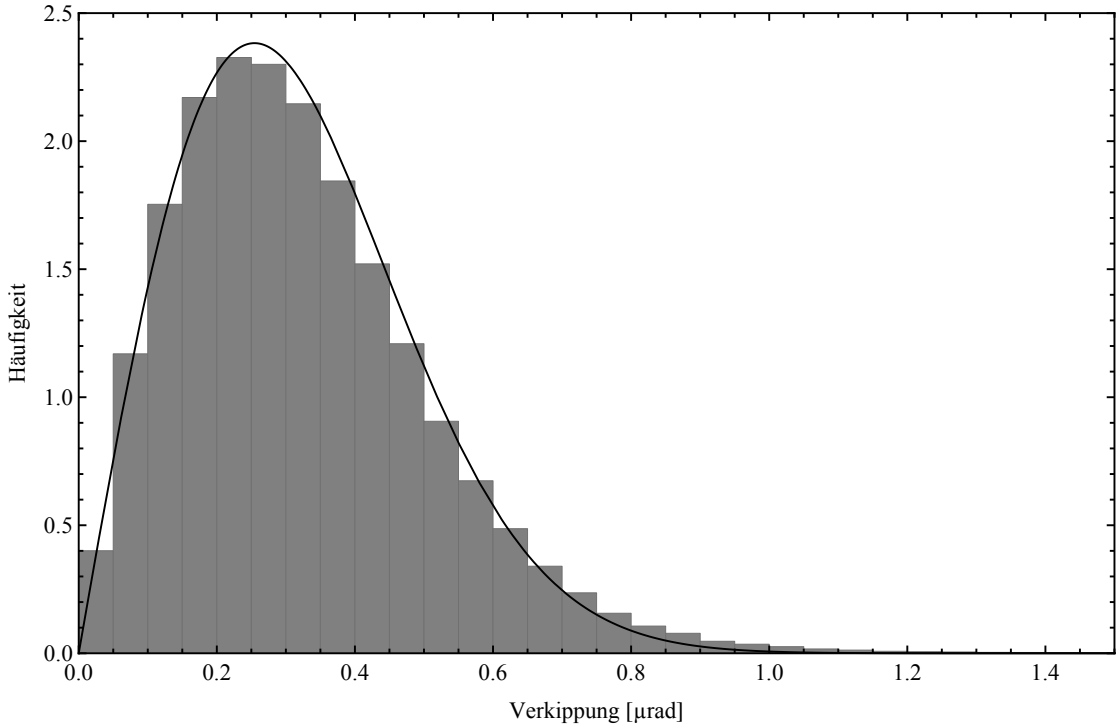
Eine Auswertung der Nivellierung aus dem Jahr 2009 ist in [7] gegeben, deren Daten am Experimentierplatz PF2 am Institut Laue-Langevin in Grenoble aufgenommen wurden. Es ergab sich eine wahrscheinlichste Verkippung von 0.771 µrad




(a) Verkippung $\Delta x$




(b) Verkippung $\Delta y$




(c) rms-Verkippung $\theta = \sqrt{\Delta x^2 + \Delta y^2}$

Abbildung 4.10.: Histogrammierung der Verkippungen gemessen über mehrere Arbeitstage mit einer Beschwerung von 222 kg und einem P Wert von 0.8 inklusive gefitteter Verteilungskurven. Verkippungen gemessen in µrad. Die Abbildungen (a) und (b) zeigen die Verkippungen der x bzw. y Achse, die Abbildung (c) zeigt die rms-Verkippung $\sqrt{\Delta x^2 + \Delta y^2}$

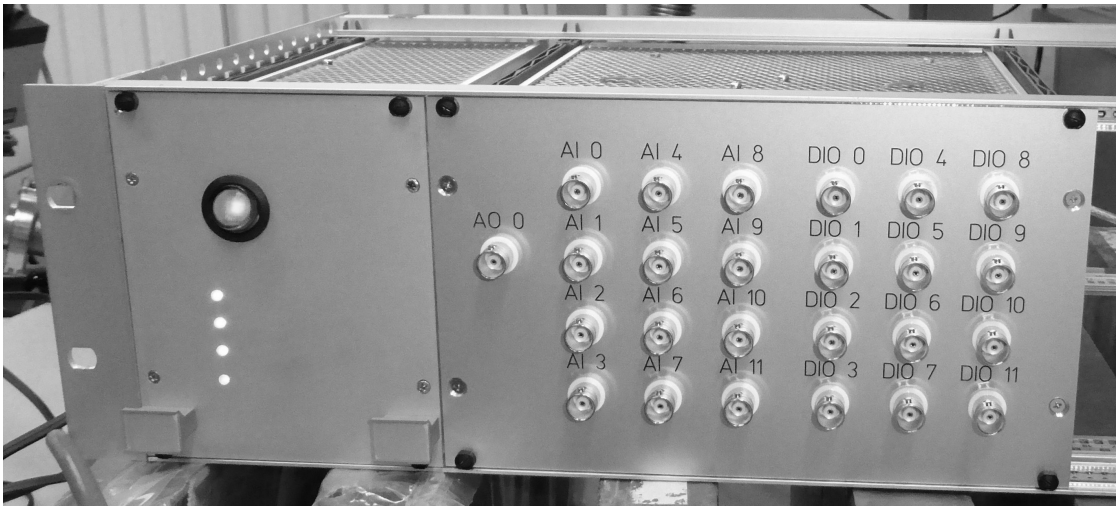



Abbildung 4.11.: Fertig Verbaute Elektronik. Links die Spannungsversorgung, rechts eine Gehäuse mit verbauter ADC/DAC Platine, Winkelmesser Platine und zusätzlichen Ein- und Ausgängen auf der Vorderseite

und eine mittlere Verkippung von 0.895 µrad. Die beiden Datensätze sind insofern vergleichbar, als dass beide zu normalen Arbeitsbedingungen mit Erschütterungen und Vibrationen der Umgebung aufgenommen wurden. Es konnte somit durch die verbesserte Auslese- und Ansteuer-Elektronik der Winkelsensoren und Piezoelemente die Nivellierung deutlich verbessert werden.

## 4.7. Verbau der Elektronik

Da die Vorbereitungen und der Aufbau des Instruments und Experiments am Atominstitut der TU Wien stattfanden, die Messungen jedoch am ILL in Grenoble, musste für den sicheren Transport aller Komponenten gesorgt werden. Die Auslese- und Ansteuerelektronik bestehend aus ADC, DAC, Verkabelung, der Platine 83162 sowie den zugehörigen Spannungsversorgungen wurde daher fest verbaut und in einem Rack fixiert. Ein Foto des Verbaus ist in Abbildung 4.11 ersichtlich. Um möglichst viele der Funktionen des ADC/DAC nutzen zu können, wurden einige der nicht verwendeten Analog und Digital Ein- und Ausgänge durch BNC-Koaxialstecker Anschlüsse an der Vorderseite zugänglich gemacht. Ebenfalls wurde von Maximilian Zach, Leiter der Elektronikwerkstätte der TU Wien, eine Spannungsversorgung geplant und gebaut (siehe Abbildung 4.11 linke Box) die die ADC/DAC Platine, die Winkelmesser Platine, sowie einige in dieser Arbeit nicht weiter behandelte Magnetfeldsensoren versorgt.

Um die diversen Ein- und Ausgänge möglichst einfach für verschiedenste Geräte nutzen zu können, wurde die Software speziell darauf ausgelegt über eine einfache Textdatei dynamisch konfiguriert zu werden. Eine genauere Beschreibung dazu ist gegeben in Kapitel 3.2.7.

# 5. Automatisierung des Vakuumkreislaufs

Der geplante Aufbau für einen automatisierten Vakuumkreislauf ist in Abbildung 5.1 zu sehen. Alle folgenden Ventilnummern beziehen sich auf diese Abbildung. Das manuelle Belüftungsventil (3) und das Schmetterlingsventil (5) wären nicht notwendig, wurden jedoch im Aufbau beibehalten, da sie in zeitkritischen Situationen den Belüftungsprozess beschleunigen können. In diesen Fällen kann, während die Turbopumpe langsam abbremst, das Schmetterlingsventil (5) geschlossen werden und die Kammer über das Belüftungsventil (3) belüftet werden. Für das Ventil (7) wurde ein pneumatisches gewählt, da so für das Experiment störende elektrische und magnetische Felder gegenüber einem elektrischen Ventil minimiert werden können. Als erster Schritt wurden die vorhandenen Komponenten, die Turbopumpe Typ TMU 521P mit Antriebselektronik TC 600 der Firma Pfeiffer[1] mit Steuergerät DCU 300 der Firma Pfeiffer, sowie die Vorpumpe Typ iXL 120 der Firma Edwards[2], auf Tauglichkeit für eine Automatisierung geprüft.

**Vorpumpe** Bei der Vorpumpe iXL 120 von Edwards handelt es sich von der Bauweise um eine sogenannte Drehschieberpumpe. Bei der Vorpumpe stellte sich heraus, dass eine Ansteuerung nur über ein Zusatzmodul möglich wäre, welches aus Kostengründen nicht gekauft wurde. Die Vorpumpe muss im vorhandenen Setup daher entweder manuell Ein- und Ausgeschaltet werden oder durchgehend laufen. Da es für die Mechanik und Lebensdauer von Drehschieberpumpen im allgemeinen von Vorteil ist, wenn sie durchgehend laufen, ist diese Option auch gegenüber einem automatisierten Ein- und Ausschalten zu bevorzugen.

**Turbopumpe** Die Turbopumpe TMU 521P mit Antriebselektronik TC 600 ist als Kernstück des Vakuumkreislaufs für die automatisierte Steuerung zuständig. Die grundlegende Idee war es, beim Einschalten der Turbopumpe das Belüftungsventil (4) zu schließen und das pneumatische Ventil (7) zur Vorpumpe zu öffnen. Beim Ausschalten der Turbopumpe soll das pneumatische Ventil (7) geschlossen werden und nach Unterschreiten einer gewissen Drehzahl der Turbopumpe die Kammer über das Belüftungsventil (4) belüftet werden. Auf diese Weise kann die Turbopumpe nicht im Betrieb bei voller Drehzahl belüftet werden, was zu einem

[1] Pfeiffer Vakuum GmbH, Berliner Straße 43, 35614 Asslar, Germany

[2] Edwards, Crawley Business Quarter, Manor Royal, Crawley, West Sussex, RH10 9LW, UK

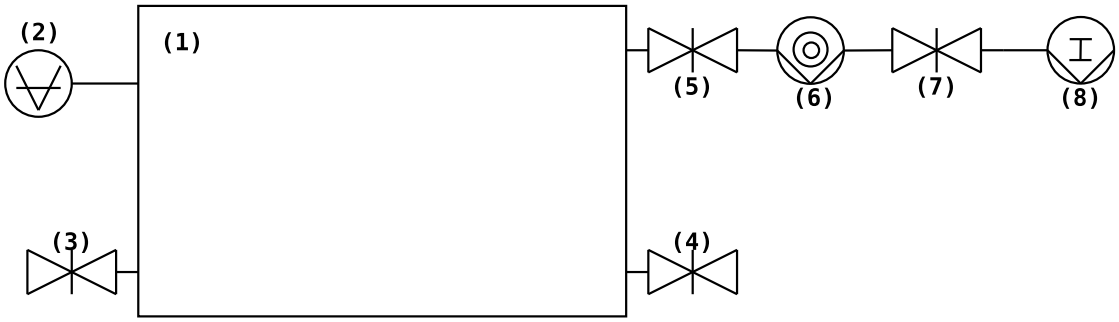



Abbildung 5.1.: Vakuumkreislauf der automatisierten Lösung. (1) Vakuumkammer, (2) Drucksonde, (3) Manuelles Belüftungsventil, (4) Elektrisches Belüftungsventil, (5) Manuelles Schmetterlingsventil, (6) Turbopumpe, (7) Pneumatisches Ventil, (8) Vorpumpe.

Schaden führen könnte. Es erwies sich, dass die dafür notwendigen Einstellungen (siehe Anhang C) am Steuergerät DCU 300 möglich und die dafür notwendigen Ein- und Ausgänge an der Antriebselektronik TC 600 vorhanden sind.

## 5.1. Halbautomatisches Testen

Um den geplanten Aufbau zu testen wurden die notwendigen Ein- und Ausgänge der Antriebselektronik TC 600 provisorisch verkabelt und der Ablauf so halbautomatisch getestet. Die Ausgangssignale wurden mit Hilfe eines Multimeters ausgelesen und die Ventile daraufhin händisch bedient. Die resultierende Druckkurve ist in Abbildung 5.2 als gepunktete Linie zu sehen. Gegenüber der manuellen Methode mit Bypass erweist sich, dass die Kammer schneller einen Druck unter $10^{-2}$ mbar erreicht. Aufgenommen wurden die Druckkurven mit Hilfe unterschiedlicher Drucksonden der Firmen Pfeiffer und Vacom[3]. Ebenfalls wurde das Signal zum Belüften der Kammer mit Hilfe eines Mutimeters ausgemessen. Die Funktionalität eines automatisierten Vakuumkreislaufs ist somit gegeben.

## 5.2. Vollautomatisierte Steuerung der Ventile

Für die vollständige Automatisierung wurde eine einfache Schaltung mithilfe zweier Relais von der Elektronikwerkstätte des Atominstituts der TU Wien entwickelt, die in Anhang D ersichtlich ist. In dieser Schaltung wurde zusätzlich ein Zeitre-

[3]VACOM Vakuum Komponenten & Messtechnik GmbH, Gabelsbergerstraße 9, 07749 Jena, Germany

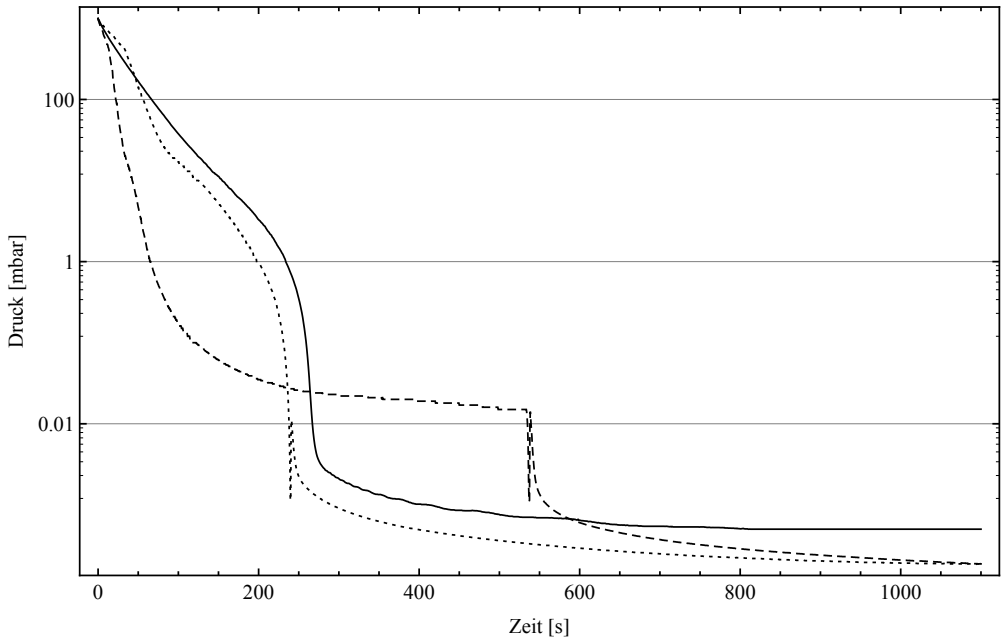



Abbildung 5.2.: Vergleich des Aussaugevorgang mit Bypass (strichlierte Linie), ohne Bypass und manuell simulierter automatischer Ventilschaltung (gepunktete Linie), sowie des finalen vollautomatisierten Prozesses (durchgezogene Linie).

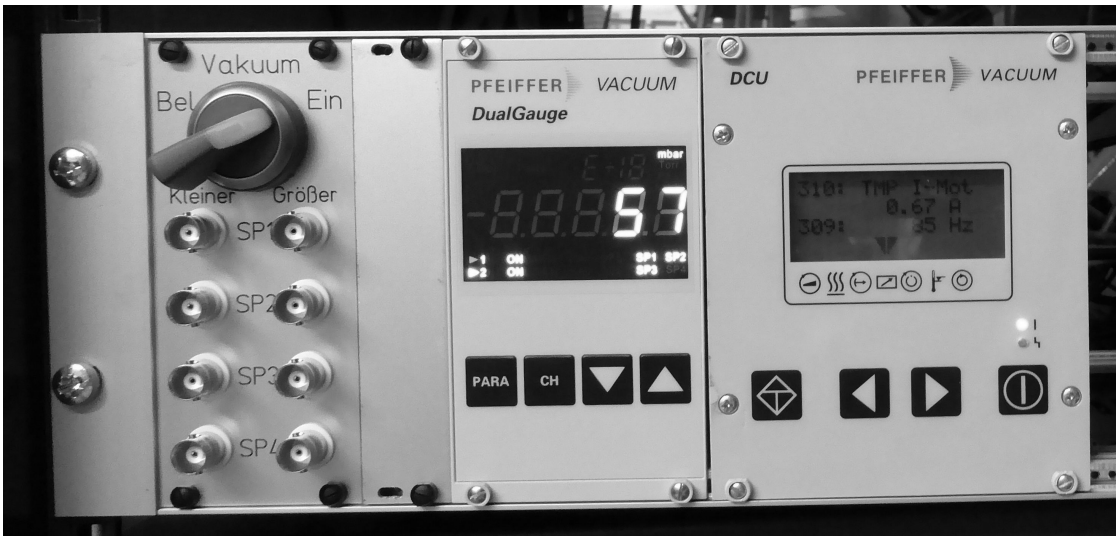



Abbildung 5.3.: Links der Schalter für den vollautomatisierten Vakuumkreislauf, darunter BNC Anschlüsse zum Auslesen der Druckschaltpunkte der Pfeiffer Dual Gauge (Mitte). Rechts DCU 300 zur Steuerung von Stromversorgung der Turbopumpe.

lais der Firma Tele[4] Typ E1ZM10 verwendet. Mit diesem ist es möglich das Belüftungsventil für eine einstellbare Zeit zu öffnen nachdem die Turbopumpe eine gewisse Drehzahl unterschreitet. Dies hat den Vorteil, dass so das Belüftungsventil nicht durchgehend stromdurchflossen ist, wenn die Turbopumpe ausgeschaltet und unterhalb einer gewissen Drehzahl ist. Die Front des fertigen Verbaus mit der Druckauslese DualGauge und Steuereinheit DCU 300 ist in Abbildung 5.3 zu sehen. Der Schalter zum Evakuieren wurde mit den möglichen Einstellungen „Ein“ zum Evakuieren und „Bel“ zum Belüften bzw. händischen Öffnen des Belüftungsventils beschriftet. Weiters ist eine neutrale Mittelstellung des Schalters möglich. In der Stellung „Ein“ wird die Turbopumpe aktiviert und die Kammer evakuiert. Wird der Schalter in die neutrale Mitellstellung versetzt fährt die Turbopumpe herunter und nach unterschreiten einer gewissen Drehzahl wird das Belüftungsventil für die am Zeitrelais eingestellte Zeit geöffnet. Die Stellung „Bel“ öffnet das Belüftungsventil immer, wenn die Drehzahl der Turbopumpe unterhalb eines gewissen Wertes ist. Die Lampe des Schalters wurde so geschaltet, dass sie bei geöffnetem Belüftungsventil leuchtet.

## 5.3. Auswertung der Ergebnisse

Ein Vergleich der unterschiedlichen Druckkurven ist in Abbildung 5.2 ersichtlich. Die Druckkurve der automatisierten Lösung ist als durchgezogene Linie dargestellt und wurde am Institut Laue Langevin in Grenoble aufgenommen. Hier waren alle Kammerflansche ausgetauscht und einige Geräte bereits in der Kammer, was den unterschiedlichen Enddruck im Graphen erklärt.

Im Praxisbetrieb erwies sich die automatisierte Lösung gegenüber dem händischen Betrieb mit Bypass, besonders wegen der deutlich geringeren Fehleranfälligkeit, als praktikabel. Einzig die Zeit zum Belüften der Kammer erwies sich im oft stressigen Labor- und Messbetrieb als unpraktisch, da es ca. 10 Minuten dauert bis die Turbopumpe unterhalb der kritischen Drehzahl ist und sich das Belüftungsventil öffnet. Diese Zeit wäre nur dann zu verkürzen, wenn das Schmetterlingsventil (5) durch ein automatisches Ventil ersetzt würde, was aufgrund der Größe des Flansches zum einen mit sehr hohen Kosten verbunden wäre und zum anderen auf Grund der Tiefe eines solchen Ventils nicht passen würde.

[4] Tele Haase Steuergeräte Ges.m.b.H., Vorarlberger Allee 38, 1230 Wien

# 6. Zusammenfassung und Ausblick

In einem dreiwöchigen Aufenthalt am ILL in Grenoble während der ersten Strahlzeit 3-14-331, vom 25 Juni bis 7 Juli 2014, des qBounce Experiments konnten alle Teile dieser Diplomarbeit im laufenden Messbetrieb getestet und ihre Funktionalität überprüft werden.

## 6.1. Nivellierung

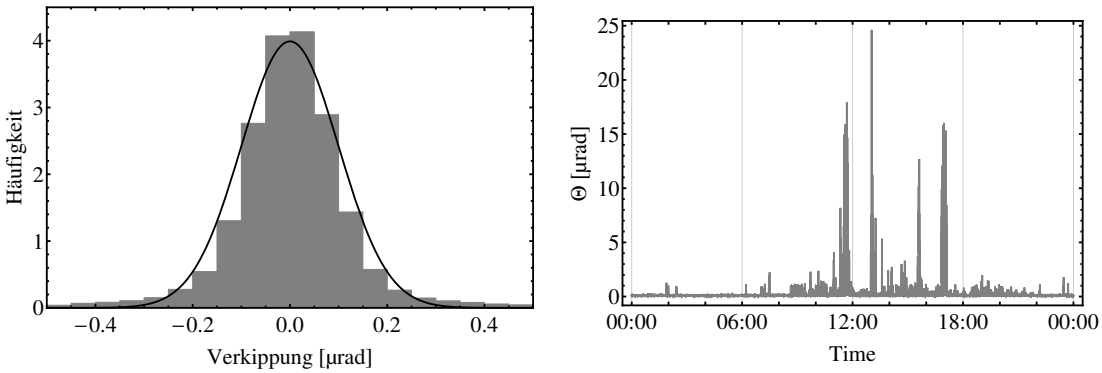
Da die Nivellierung des Experiments in Wien nur mit einer Beschwerung von 222 kg getestet wurde, war es wichtig zu testen ob diese auch bei vollem Gewicht funktioniert. Schon der erste Test zeigte, dass die Nivellierung wie erwartet funktioniert und wie in Wien ergab sich eine Standardabweichung der rms-Verkippung von $\sigma_\theta \approx 0.3\,\mu$rad. Für die Verkippung ergaben sich wieder sehr schöne Gaussverteilungen, wie in Abbildung 6.1a, einer Auswertung über einen ganzen Tag im laufenden Experimentbetrieb, zu sehen ist. Selbst größere Erschütterungen wie sie durch den Hallenkran und die Arbeit an den benachbarten Experimenten verursacht werden, ersichtlich in Abbildung 6.1b, konnten durch die Steuerung ausgeglichen werden.

Außerdem konnte die Nivellierung im Laufe des Betriebs noch leicht verbessert werden, unter anderem durch stärkere Mittelung der Winkelmesserwerte und durch Berücksichtigung der Trägheit des Systems, und es wurde so ein Wert der Standardabweichung der rms-Verkippung von $\sigma_\theta \approx 0.1\,\mu$rad erreicht. Dies entspricht einem um einen Faktor 5 bis 10 besseren Wert gegenüber dem System der Vorjahre.

Da bei der Implementierung der Software bereits darauf geachtet wurde, dass das Programm nur leicht abgeändert werden muss um das für 2015 geplante, deutlich größere Experiment zu nivellieren ist hier ebenfalls für die Zukunft vorgesorgt.

## 6.2. Vakuumkreislauf

Der Arbeitsaufwand während der Strahlzeit erwies sich als enorm, weshalb die Automatisierung dieses kritischen Systems sich als sehr vorteilhaft erwies. Mit




(a) Verkippung $\Delta x$ (b) rms-Verkippung $\theta = \sqrt{\Delta x^2 + \Delta y^2}$

Abbildung 6.1.: Histogrammierung der Verkippungen gemessen während der Strahlzeit 3-14-331 am ILL Grenoble im laufenden Experimentbetrieb.

der automatischen Schaltung ist ein Fehlverhalten beim Evakuieren bzw. Belüften der Vakuumkammer nicht mehr möglich. Lediglich die relativ lange Dauer der Belüftung erwies sich als problematisch, da das Belüftungsventil der Kammer erst öffnet, sobald die Turbopumpe unterhalb einer kritischen Drehzahl von ca. 5% der Maximaldrehzahl fällt. Dies dauert ca. 10 Minuten mit einer anschließenden Belüftungsdauer von ca. 20 Minuten.

Aufgrund der am Zeitrelais einstellbaren Belüftungsdauer ist diese Lösung auch für die deutlich größere Vakuumkammer geeignet, die für 2015 geplant ist.

## 6.3. Software

Auch die Softwareseite mit dem LabView Projekt und der Job Struktur der Geräte erwies sich als sehr gut geeignet. Während der ersten Wochen der Strahlzeit mussten einige Funktionen der Geräte implementiert werden, was aus Zeitgründen davor nicht möglich war. Einzelne neue Jobs der Geräte waren aufgrund des allgemeinen Aufbaus schnell implementiert und auch die Einbindung neuer Geräte in das Projekt und das Hauptprogramm erwies sich als problemlos. Durch das Versionskontrollsystem GIT ist es des weiteren möglich, dass mehrere Personen gleichzeitig auf unterschiedlichen Rechnern an einem Projekt arbeiten und die unterschiedlichen Versionen dann wieder zusammengefügt werden.

Auch die Datenaufnahme über Syslog erwies sich als sehr praktikabel. Aufgrund der Abstraktionsebene über das Netzwerk können so alle Daten zentral aufgenommen werden. Das Format enthält alle notwendigen Daten wie Uhrzeit, Name des sendenden Rechners und des Programms/Geräts, das die Daten produziert. Die

Aufteilung in die unterschiedlichen Dateien erwies sich als angenehm, im besonderen, weil Messdaten und Metadaten getrennt abgelegt sind. Speziell durch das vorgefertigte Mathematica Paket zum Einlesen und Filtern der Daten war diese Form der Datenablage schnell akzeptiert.

# Literatur


[1] Dipl.-Phys. Dr. Tobias Jenke. URL: https://tiss.tuwien.ac.at/adressbuch/adressbuch/person/69861 (besucht am 30. 03. 2014).

[2] Univ.Prof. Dipl.-Phys. Dr. Hartmut Abele. URL: https://tiss.tuwien.ac.at/adressbuch/adressbuch/person/69169 (besucht am 30. 03. 2014).

[3] David Stadler. „Dynamik ultrakalter Neutronen im Gravitationsfeld der Erde“. Diplomarbeit. Physikalisches Institut der Universität Heidelberg, 2009.

[4] Alexander Westphal. „Quantum Mechanics and Gravitation“. Diploma thesis. University of Heidelberg, 2001.

[5] Dipl.-Phys. Dr. Tobias Jenke. „Weiterentwicklung eines Experiments zur Realisierung eines Quantum Bouncing Balls und Suche nach Extradimensionen der Raumzeit“. master thesis. Ruprecht Karls Universität Heidelberg, Fakultät für Physik und Astronomie, 2008.

[6] Heiko Saul. „Weiterentwicklung des Detektor- und Auslesekonzepts für das Gravitationsexperiment qBounce“. master thesis. Technische Universität Wien, 2011.

[7] Dipl.-Phys. Dr. Tobias Jenke. „qBounce - Vom Quantum Bouncer zur Gravitationsresonanzspektroskopie“. Dissertation. Technische Universität Wien, 2011.

[8] Thomas Bittner. „Entwicklung eines Blendensystems zur Geschwindigkeitsselektion für das qBounce Experiment“. master thesis. Technische Universität Wien, 2013.

[9] Hanno Filter. „Neutronendetektor auf CR-39 Basis“. Projektarbeit. University of Heidelberg, 2009.

[10] Claude Krantz. „Quantum States of Neutrons in the Gravitational Field“. University of Heidelberg, 2006.

[11] Sophie Louise Nahrwold. „Developement of a Detector for Bound Quantum States of Neutrons in the Earths Gravitational Field“. Diploma Thesis. University of Heidelberg, 2004.

[12] Tobias Jenke, Peter Geltenbort, Hartmut Lemmel and Hartmut Abele. „Realization of a gravity-resonance-spectroscopy technique“. In: *Nature Physics* 7 (2011).

[13] I. I. Rabi u. a. „A New Method of Measuring Nuclear Magnetic Moment“. In: *Phys. Rev.* 53 (4 Feb. 1938), S. 318–318. DOI: 10.1103/PhysRev.53.318. URL: http://link.aps.org/doi/10.1103/PhysRev.53.318.

[14] Norman F. Ramsey. „A Molecular Beam Resonance Method with Separated Oscillating Fields“. In: *Phys. Rev.* 78 (6 Juni 1950), S. 695–699. DOI: 10.1103/PhysRev.78.695. URL: http://link.aps.org/doi/10.1103/PhysRev.78.695.

[15] Markus Spanring. „Kalibrierung der analogen Eingänge, sowie der digitalen Ausgänge des Steuerungs- und Datenerfassungssystems ‚Logic Box‘“. Projektarbeit. Technische Universität Wien, 2013.

[16] Sebastian Keerl. „Systematische Tests des Vakuumsystems des qBounce Experiments“. Bachelorarbeit. Technische Universität Wien, 2012.

[17] *GIT*. URL: http://www.git-scm.org/ (besucht am 30. 03. 2014).

[18] *LabViewGitEnv*. URL: http://www.github.com/joerg/LabViewGitEnv/ (besucht am 30. 03. 2014).

[19] *The syslog protocol - RFC 5424*. URL: http://tools.ietf.org/html/rfc5424 (besucht am 30. 03. 2014).

[20] *Transmission of syslog messages over UDP - RFC 5426*. URL: http://tools.ietf.org/html/rfc5426 (besucht am 30. 03. 2014).

[21] *Date and Time on the Internet: Timestamps - RFC 3339*. URL: http://tools.ietf.org/html/rfc3339 (besucht am 30. 03. 2014).

[22] *Almquist Shell*. URL: http://www.in-ulm.de/~mascheck/various/ash/ (besucht am 30. 03. 2014).

[23] *GNU GZip*. URL: http://www.gnu.org/software/gzip/gzip.html (besucht am 30. 03. 2014).

[24] *Produktkatalog Tech-Sys Instruments*. URL: http://www.tech-sys.be/PDF/2010%20AGI%20catalog.pdf (besucht am 21. 07. 2014).

[25] Karl J. Åström und Richard M. Murray. „PID Control“. In: *Feedback Systems: An Introduction for Scientists and Engineers*. .Chapter 10. Cloth, 2008. Kap. 10. ISBN: 9780691135762. URL: http://www.cds.caltech.edu/~murray/amwiki/index.php/Main_Page (besucht am 15. 03. 2014).

[26] National Instruments. *PID Theory Explained*. 2011. URL: http://www.ni.com/white-paper/3782/en/ (besucht am 30. 03. 2014).

[27] Rick Astley. *Never Gonna Give You Up*. 1987. URL: http://www.youtube.com/watch?v=oHg5SJYRHA0.

# Danksagung

Ich möchte allen danken, die mich in den vielen Jahren meines Studiums unterstützt haben. Ganze besonders aber möchte ich meiner Mutter danken und ihr diese Arbeit widmen. Mehr Fürsorge und Stolz auf die eigenen Kinder kann kein Mensch aufbringen. Wie sonst niemand hast du dich darüber gefreut, dass ich nun doch endlich fertig werde und dennoch war es dir nicht vergönnt dies auch noch zu erleben.

*Danke Mama (1957 - 2014)*

# A. Script zur Kompression und Backup der Daten

code/qBounce_logrotate.sh

```
#!/bin/ash

for FILE in $(find /volume1/qBounceLogs/syslog/)
do
 # Select only files that contain a valid date at the end of the filename
 DATE=$(echo $FILE | grep -oE '[0-9]{4}-[0-9]{2}-[0-9]{2}$')
 # Skip FILE if DATE is empty or today
 test -z $DATE && continue
 test $(date +%F) = $DATE && continue
 # Now zip the file with maximum compression
 gzip -9 $FILE
 chmod 664 "${FILE}.gz"
 chown root:users "${FILE}.gz"
done
rsync -rl --del --perms --chmod=Dug+rwx,Do-rwx,Fug+rw,Fug-x,Fo-
    rwx /volume1/qBounceLogs grenoble@128.131.38.218:/home/qbounce/
    Backups/2014_ILL_Grenoble/
if [ $? -eq 0 ]
then
 /usr/syno/bin/synonotify NetBkpOK
else
 /usr/syno/bin/synonotify NetBkpError
fi
```

# B. Mathematica Paket

code/syslog_reader.m

```
(* ::Package:: *)

(* Package syslog_reader
 Author: J\[ODoubleDot]rg Herzinger
Credits for original version go to Tobias Jenke and Leonid Shifrin (http://stackoverflow.com/questions/7525782/import
    -big-files-arrays-with-mathematica)
This is a heavily modified version of their original work.

ToDo:
 - Nothing
*)


BeginPackage["syslogReader`"];

syslogLineParse::usage =
"syslogLineParse[ line , since , dateStyle->\"iso\", fromHost->\"hostname\"]
line (mandatory)                : A single or multiple Syslog lines to parse. This can either be a String or a List.
since (optional)                : A String or DateList. The resulting timestamps will be in Seconds since 'since'.
dateStyle (optional, default iso) : Valid values are iso or bsd. Defines the style of the Date/Time
 string in syslog. bsd is of the form \"Nov 20 12:14:22\" without year or timezone information,
```

```
 iso is of the form \"2014-01-09T14:55:51.194+01:00\".
fromHost (optional)                 : The hostname that has sent this message.

This parses any number of syslog formatted lines into a List of the form { {DateList, Values*}, {DateList, Values*}, ...
    }.";

syslogDateParse::usage =
"syslogDateParse[ date, dateStyle->\"iso\"]
date (mandatory)                    : A Date/Time string, or a List of Date/Time Strings of ISO or BSD style.
dateStyle (optional, default iso) : Valid values are iso or bsd. Defines the style of the Date/Time
 string in syslog. bsd is of the form \"Nov 20 12:14:22\" without year or timezone information,
 iso is of the form \"2014-01-09T14:55:51.194+01:00\".

This will convert syslog Date strings to Mathematica DateLists.";

syslogReadFile::usage =
"syslogReadFile[ file , dateRange, chunkSize->2500, takeEvery->0, secondsSince->False, dateStyle->\"iso\",
    fromHost->\"hostname\"]
 file (mandatory)                       : The syslog formatted line to read.
dateRange (optional, default empty)  : Is a List of DateLists specifying a filter for start and end date.
chunkSize (optional, default 2500)   : Number of Lines to read consecutively. Helpful for huge files .
takeEvery (optional, default 0)      : Will only take every nth line of the file . This correlates with
 chunk size. With a chunkSize of 20 and takeEvery of 3 lines 1,4,7,10,13,16 and 19 will be taken.
secondsSince (optional, default False): Valid values are either False, True or a DateList. If True dateRange
 must be given and start will be the first value of dateRange. If a DateList is given this will be the start .
dateStyle (optional, default iso)    : Valid values are iso or bsd. Defines the style of the Date/Time
 string in syslog. bsd is of the form \"Nov 20 12:14:22\" without year or timezone information,
 iso is of the form \"2014-01-09T14:55:51.194+01:00\".
fromHost (optional, default empty)   : The hostname that has sent this message.

```

```
This will read a syslog file into a List of the form { { DateList, Values* }, { DateList, Values* }, ... }.";

syslogReadDirectory::usage =
"syslogReadDirectory[ directory, experiment , dateRange , chunkSize->2500, takeEvery->0, fromHost->\"hostname\",
 getHeader->True, headerKey->\"HEADER\"]
directory (mandatory)                  : The syslog directory.
experiment (mandatory)                 : The name of the experiment.
dateRange (mandatory)                  : The range to read files from.
  These first three mandatory options will read files named '<directory>\\qbounce.<experiment>.<YEAR>-<
      MONTH>-<DAY>'
chunkSize (optional, default 2500)     : Number of Lines to read consecutively. Helpful for huge files .
takeEvery (optional, default 0)        : Will only take every nth line of the file . This correlates with
 chunk size. With a chunkSize of 20 and takeEvery of 3 lines 1,4,7,10,13,16 and 19 will be taken.
secondsSince (optional, default False): Valid values are either False, True or a DateList. If True dateRange
 must be given and start will be the first value of dateRange. If a DateList is given this will be the start .
dateStyle (optional, default iso)      : Valid values are iso or bsd. Defines the style of the Date/Time
 string in syslog. bsd is of the form \"Nov 20 12:14:22\" without year or timezone information, iso
 is of the form \"2014-01-09T14:55:51.194+01:00\".
fromHost (optional, default emtpy)     : The hostname that has sent this message.
getHeader (optional, default True)     : Will additionally execute syslogGetHeader.
headerKey (optional, default HEADER) : The Header indicator. See syslogGetHeader for documentation.

This will read all files in directory belonging to experiment within given dateRange.";

syslogGetHeader::usage =
"syslogGetHeader[ directory, experiment, dateRange, chunkSize->2500, fromHost->\"hostname\", headerKey->\"
    HEADER\" ]
directory (mandatory)                  : The syslog directory.
experiment (mandatory)                 : The name of the experiment.
dateRange (mandatory)                  : The range to read files from.
```

```
These first three mandatory options will read files named '<directory>\\qbounce.<experiment>.<YEAR>-<MONTH>-<DAY>'
chunkSize (optional, default 2500)     : Number of Lines to read consecutively. Helpful for huge files .
dateStyle (optional, default iso)      : Valid values are iso or bsd. Defines the style of the Date/Time
 string in syslog. bsd is of the form \"Nov 20 12:14:22\" without year or timezone information, iso
 is of the form \"2014-01-09T14:55:51.194+01:00\".
fromHost (optional, default empty)     : The hostname that has sent this message.
headerKey (optional, default HEADER) : The keyword indicating a header line.

This will read the according notice file and Print the line that contains the string given by headerKey.
The Return value is the number of Headers found in given dateRange.";

syslogToSecondsSince::usage =
"syslogToSecondsSince[ data, since ]
data (mandatory)                : The List to convert. Starting date will be the first date in the list
 if since is not given/empty.
since (optional, default empty) : A DateList giving the starting date of the conversion.

This will convert a given list with dates into a list with seconds.";

syslogSelectBetween::usage =
"syslogSelectBetween[ data, dateRange ]
data (mandatory)      : The list to convert.
dateRange (mandatory) : The range to select entries from. Has to be a List of DateLists.

This will select elements from data in between dateRange.";


Begin["`Private`"];

```

```
Clear[syslogDatePartsToTakeBSD,syslogDatePartsToTakeISO,syslogDateRegexBSD,syslogDateRegexISO];
(* BSD Style Time: "Nov 20 12:14:22". No year included or timezone included which is error prone! *)
syslogDateRegexBSD=RegularExpression["^(<\\d+>)*(\\w+)\\s+(\\d+)\\s+(\\d+):(\\d+):(\\d+\\.?\\d+?)"];
syslogDatePartsToTakeBSD={DateString[{"Year"}] <> " $2 $3 $4 $5 $6"};
(* ISO Style Time: "2014-01-09T14:55:51.194+01:00". Year and timezone included and faster to parse! *)
syslogDateRegexISO=RegularExpression["^(<\\d+>)*(\\d+)-(\\d+)-(\\d+)T(\\d+):(\\d+):(\\d+\\.?\\d*)\\+(\\
    d+:\\d+)"];
syslogDatePartsToTakeISO={"$2 $3 $4 $5 $6 $7"};

(* Host and Data Parts for BSD and ISO Style lines when impoted with ImportString *)
syslogHostPartBSD=4; syslogHostPartISO=2;
syslogDataPartBSD=6; syslogDataPartISO=4;


Clear[slDD]; (* syslogDateDifference, shorthand which is often used *)
slDD[a_, b_] := DateDifference[a, b, "Second"][[1]]


Clear[slReplBSDMonth]; (* BSD Style Dates have a string type month that has to be replaced by a number *)
slReplBSDMonth[l_List] := StringReplace[l,
   {"Jan"->"1", "Feb"->"2", "Mar"->"3", "Apr"->"4", "May"->"5", "Jun"->"6",
    "Jul"->"7", "Aug"->"8", "Sep"->"9", "Oct"->"10", "Nov"->"11","Dec"->"12"}]


Clear[slLineListToString];
slLineListToString[l_] := If[ ListQ[l],
  StringJoin[Riffle[l, "\n"]],
  l
 ]
```

```
Clear[syslogLineParse];
syslogLineParse[line_, OptionsPattern[{dateStyle->"iso", fromHost->""}] ] :=
 Module[{lineList,hostPart,dataPart},
  Switch[OptionValue[dateStyle],
   "iso", hostPart=syslogHostPartISO; dataPart=syslogDataPartISO; datesList:=lineList[[All,1]],
   "bsd", hostPart=syslogHostPartBSD; dataPart=syslogDataPartBSD;
    spaceList:=Array[" "&,Length[lineList]];
    datesList:=StringJoin/@Transpose[
     {lineList [[All,1]], spaceList, ToString/@lineList[[All,2]],spaceList, lineList [[All,3]]}
    ],
   "_", Print["Error, dateStyle is not iso or bsd"]
  ];
  lineList =ImportString[slLineListToString[line],"Table"];
  If[ OptionValue[fromHost] != "",
   lineList = Select[lineList, #[[hostPart]] == OptionValue[fromHost]& ]
  ];
  FlattenAt[#,2]&/@Transpose[
   {syslogDateParse[datesList,dateStyle->OptionValue[dateStyle]],lineList [[All,dataPart ;;]]}
  ]
 ]

syslogLineParse[line_, since_, OptionsPattern[{dateStyle->"iso", fromHost->""}] ] :=
 Module[{sinceList,lineList},
  lineList =syslogLineParse[line, dateStyle->OptionValue[dateStyle], fromHost->OptionValue[fromHost]];
  sinceList = slDateStringOnlyParse[since, dateStyle->OptionValue[dateStyle]];
  syslogToSecondsSince[ lineList , sinceList , dateStyle->OptionValue[dateStyle]]
 ]
```

```
Clear[syslogDateParse];
syslogDateParse[ date_, OptionsPattern[{dateStyle->"iso"}] ] :=
 Switch[ OptionValue[dateStyle],
  "iso", ImportString[
   StringJoin[
     Riffle [
      Flatten[
       StringCases[date, syslogDateRegexISO -> syslogDatePartsToTakeISO]
      ]
     ,"\n"]
   ]
  ,"Table"],
  "bsd", ImportString[
   StringJoin[
     Riffle [
      Flatten[
       StringCases[slReplBSDMonth[date], syslogDateRegexBSD -> syslogDatePartsToTakeBSD]
      ]
     ,"\n"]
   ]
  ,"Table"],
  _, Print["Incorrect dateStyle given!"]
 ]

(* For internal use only. This function only parses the Strings within a list of dates. *)
Clear[slDateStringOnlyParse];
slDateStringOnlyParse[ date_, OptionsPattern[{dateStyle->"iso"}] ] :=
 Switch[ Head[date],
  String, syslogDateParse[date, dateStyle->OptionValue[dateStyle]][[1]],
```

```
  List, If[ Length[date] > 0 && (StringQ[date[[1]]] || ListQ[date[[1]]]),
   If[StringQ[#], syslogDateParse[#, dateStyle->OptionValue[dateStyle]][[1]], #]&/@date,
   date
  ],
  _, date
 ]


Clear[syslogReadFile];
(* Without dateRange *)
syslogReadFile[file_String?FileExistsQ,
 OptionsPattern[{chunkSize->2500, takeEvery->0, secondsSince->{}, dateStyle->"iso", fromHost->""}] ]:=
 Module[{stream, dataChunk, result, slp, since},
  stream = StringToStream[Import[file,"String"]];
  since = slDateStringOnlyParse[OptionValue[secondsSince], dateStyle->OptionValue[dateStyle]];
  If[since == {} || TrueQ[!since],
   slp[a_]:=syslogLineParse[a, dateStyle->OptionValue[dateStyle], fromHost->OptionValue[fromHost]];
  ];
  If[ ListQ[since] || StringQ[since],
   slp[a_]:=syslogLineParse[a, since, dateStyle->OptionValue[dateStyle], fromHost->OptionValue[fromHost]];
  ];
  (*main code*)
  result={};
  While[dataChunk=!={},
   dataChunk=ReadList[stream,"String",OptionValue[chunkSize]];
   If[ OptionValue[takeEvery] > 0, dataChunk = Take[dataChunk,{1,-1,OptionValue[takeEvery]}]];
   (*somehow sometimes dataChunk is empty, if so skip iteration*)
   If[ dataChunk== {} || Length[dataChunk]== 0, Continue[], Indeterminate];
   (* now parse dataChunk *)
   dataChunk=slp[dataChunk];
```

```
    result=If[ Length[result] == 0,dataChunk,Join[result,dataChunk]];
   ];
   (*clean-up*)
   Close[stream];
   result
  ]

(* With date Range *)
syslogReadFile[file_String?FileExistsQ, dateRange_List,
 OptionsPattern[{chunkSize->2500, takeEvery->0, secondsSince->False, dateStyle->"iso", fromHost->""}] ]:=
 Module[{stream, dataChunk, firstAndLast, result, since, diff, slp, dr, dateRangeList, optList, getDate},
  dateRangeList = slDateStringOnlyParse[dateRange, dateStyle->OptionValue[dateStyle]];
  stream=StringToStream[Import[file,"String"]];
  (*main code*)
  optList={dateStyle->OptionValue[dateStyle], fromHost->OptionValue[fromHost]};
  If[ TrueQ[!OptionValue[secondsSince]], (* secondsSince is False *)
   since = dateRangeList[[1]];
   diff [a_,b_] := slDD[a,b];
   slp [a_] := syslogLineParse[a, #]&[optList];
   dr = dateRangeList;
   getDate[s_] := syslogLineParse[s, dateStyle->OptionValue[dateStyle]];
  ];
  If[ TrueQ[OptionValue[secondsSince]], (* secondsSince is True *)
   since = dateRangeList[[1]];
   diff [a_,b_] := Subtract[b,a];
   slp [a_] := syslogLineParse[a,since, #]&[optList];
   dr = DateDifference[since,#,"Second"][[1]]&/@dateRangeList;
   getDate[s_] := syslogLineParse[s, since, dateStyle->OptionValue[dateStyle]];
  ];
  If[ ListQ[OptionValue[secondsSince]] || StringQ[OptionValue[secondsSince]], (*secondsSince is String or List *)
```

```
  since = slDateStringOnlyParse[OptionValue[secondsSince], dateStyle->OptionValue[dateStyle]];
  diff [a_,b_] := slDD[a,b];
  slp [a_] := syslogLineParse[a, since, #]&[optList];
  dr = DateDifference[since,#,"Second"][[1]]&/@dateRangeList;
  getDate[s_] := syslogLineParse[s, since, dateStyle->OptionValue[dateStyle]];
 ];
 result={};
 While[dataChunk=!={},
  dataChunk=ReadList[stream,"String",OptionValue[chunkSize]];
  If[ OptionValue[takeEvery] > 0, dataChunk = Take[dataChunk,{1,-1,OptionValue[takeEvery]}]];
  (*somehow sometimes dataChunk is empty, if so skip iteration*)
  If[ dataChunk== {} || Length[dataChunk]== 0, Continue[], Indeterminate];
  (* first speedup: if last is too early skip this loop iteration, if first is too late then break loop*)
  firstAndLast={getDate[First[dataChunk]][[1,1]], getDate[Last[dataChunk]][[1,1]] };
  If[ diff [ firstAndLast [[2]], dr [[1]] ] > 0 , Continue[],Indeterminate];
  If[ diff [ firstAndLast [[1]], dr [[2]] ] < 0 , Break[],Indeterminate];
  (* now parse dataChunk *)
  dataChunk=slp[dataChunk];
  (*if first is too early and last is late enough then filter, the same if first is late enough and last is too late*)
  If[ ( ( diff [ firstAndLast [[1]], dr [[1]] ] < 0 && diff [ firstAndLast [[2]], dr [[2]] ] > 0) ||
  ( diff [ firstAndLast [[1]], dr [[2]] ] > 0 && diff [ firstAndLast [[2]], dr [[1]] ] < 0) ),
   dataChunk=Select[dataChunk,diff[#[[1]], dr[[1]] ]<0 && diff [#[[1]], dr [[2]]] >0&];
  ];
  result=If[ Length[result] == 0,dataChunk,Join[result,dataChunk]];
 ];
 (*clean-up*)
 Close[stream];
 result
]
```

```
Clear[syslogReadDirectory];
syslogReadDirectory[directory_String, experiment_String, dateRange_List,
 OptionsPattern[{chunkSize->2500, takeEvery->0, secondsSince->False,
  dateStyle->"iso", fromHost->"", getHeader->True, headerKey->"HEADER"}] ]:=
 Module[{days, files, result , getSecondsSince, tmpFile, dateRangeList, since, myOptList, experimentReal},
  dateRangeList = slDateStringOnlyParse[dateRange, dateStyle->OptionValue[dateStyle]];
  days=Floor[DateDifference[ dateRangeList [[1]][[1;;3]], dateRangeList [[2]][[1;;3]] , "Day" ][[1]]];
   files = result = {};
  Do[
    files =Append[files,DateString[ DatePlus[ dateRangeList[[1]], {i,"Day"}] ,{"Year","-","Month","-","Day"} ] ];
    ,{ i ,0, days}
  ];
  experimentReal=StringSplit[experiment,"_"][[1]];
   files = FileNameJoin[{directory,experimentReal,"qBounce."<>experiment<>"."<>#}]&/@files;
  If[ TrueQ[OptionValue[secondsSince]], since = dateRangeList[[1]] ];
  If[ TrueQ[!OptionValue[secondsSince]], since = False ];
  If[ ListQ[OptionValue[secondsSince]], since = OptionValue[secondsSince] ];
  If[ StringQ[OptionValue[secondsSince]],
   since = slDateStringOnlyParse[OptionValue[secondsSince], dateStyle->OptionValue[dateStyle]][[1]]
  ];
  myOptList={chunkSize->OptionValue[chunkSize], dateStyle->OptionValue[dateStyle],
   fromHost->OptionValue[fromHost], headerKey->OptionValue[headerKey]};
  If[ OptionValue[getHeader], syslogGetHeader[ directory, experiment, dateRange, #]&[myOptList] ];
  For[i=1,i<=Length[files],i++,
   myOptList={chunkSize->OptionValue[chunkSize], takeEvery->OptionValue[takeEvery],
    secondsSince->since, dateStyle->OptionValue[dateStyle], fromHost->OptionValue[fromHost]};
   tmpFile = If[ FileExistsQ[ files [[ i ]] <> ".gz" ], files [[ i ]] <> ".gz", files [[ i ]] ];
   result= If[i==1 || i==Length[files],
    Append[result,syslogReadFile[tmpFile, dateRangeList, #]&[myOptList] ],
```

```
      Append[result,syslogReadFile[tmpFile, #]&[myOptList] ]
      ];
    ];
  Join@@result
  ]


Clear[syslogGetHeader];
syslogGetHeader[directory_String, experiment_String, dateRange_List,
 OptionsPattern[{chunkSize->2500, dateStyle->"iso", fromHost->"", headerKey->"HEADER"}]] :=
 Module[{data, header, experimentReal, myOptList},
  experimentReal=StringSplit[experiment,"_"][[1]];
  myOptList={chunkSize->OptionValue[chunkSize], dateStyle->OptionValue[dateStyle], fromHost->OptionValue[
      fromHost]};
  data=syslogReadFile[directory<>"qBounce."<>experimentReal<>".notice", dateRange, #]&[myOptList];
  header=DeleteDuplicates[Select[ data, #[[2]] == OptionValue[headerKey]& ][[All,2;;]]];
  Switch[ Length[header],
   0, Print["No header found!!"],
   1, Print[StringJoin[Riffle[ToString[#]&/@header[[1,2;;]], " "]]],
   _, Print["More than one header found!!"]
  ];
  Length[header]
 ]


Clear[syslogToSecondsSince];
syslogToSecondsSince[data_List, since_:{}, OptionsPattern[{dateStyle->"iso"}] ]:=
 Module[{sinceList},
  sinceList = slDateStringOnlyParse[since, dateStyle->OptionValue[dateStyle]];
  sinceList = If[ sinceList ==={},data[[1,1]], sinceList ];
```

```
  ReplacePart[#,1->DateDifference[sinceList , #[[1]] ,"Second"][[1]] ]&/@data
 ]


Clear[syslogSelectBetween];
syslogSelectBetween[data_List, dateRange_List, OptionsPattern[{dateStyle->"iso"}]]:=
 Module[{dateRangeList},
  dateRangeList = slDateStringOnlyParse[dateRange, dateStyle->OptionValue[dateStyle]];
  Select[data,DateDifference [#[[1]],  dateRangeList [[1]]  ]<0 && DateDifference[#[[1]], dateRangeList [[2]]]  >0&]
 ]


End[]; EndPackage[];
```

code/syslog_statistics.m

```
(* ::Package:: *)

(* Package syslog_statistics
   Author: J\[ODoubleDot]rg Herzinger
   Statistics and helper functions for syslog data
*)


Needs["syslogReader`"]
BeginPackage["syslogStatistics`"];


syslogNormalizeData::usage=
"syslogNormalizeData[ data, steps] returns normalized syslog data in steps bins
data (mandatory) : The syslog data to parse.
steps (mandatory) : The number of bins to use to normalize the data.";


Begin["`Private`"];


Clear[syslogNormalizeData];
syslogNormalizeData[data_, steps_] :=
 Module[{parts,dim,tmpDataNormalized,column,remaining},
  dim=Dimensions[data];
  tmpDataNormalized=List[];
  For[i=2,i<=dim[[2]],i++,
   parts = Partition[data[[All,i]], steps];
   column = Flatten[parts-Mean/@parts];
```

```
    parts = data[[All,i ]][[ dim[[1]]-Mod[dim[[1]],steps]+1;;dim [[1]]]];
    column = Join[column, parts-Mean[parts] ];
    tmpDataNormalized = Append[tmpDataNormalized, column];
   ];
  Transpose[ Prepend[tmpDataNormalized,data[[All,1]] ] ]
 ]


End[]; EndPackage[];
```

# C. Einstellungen an der DCU

## C.1. Normalbetrieb über Knopf

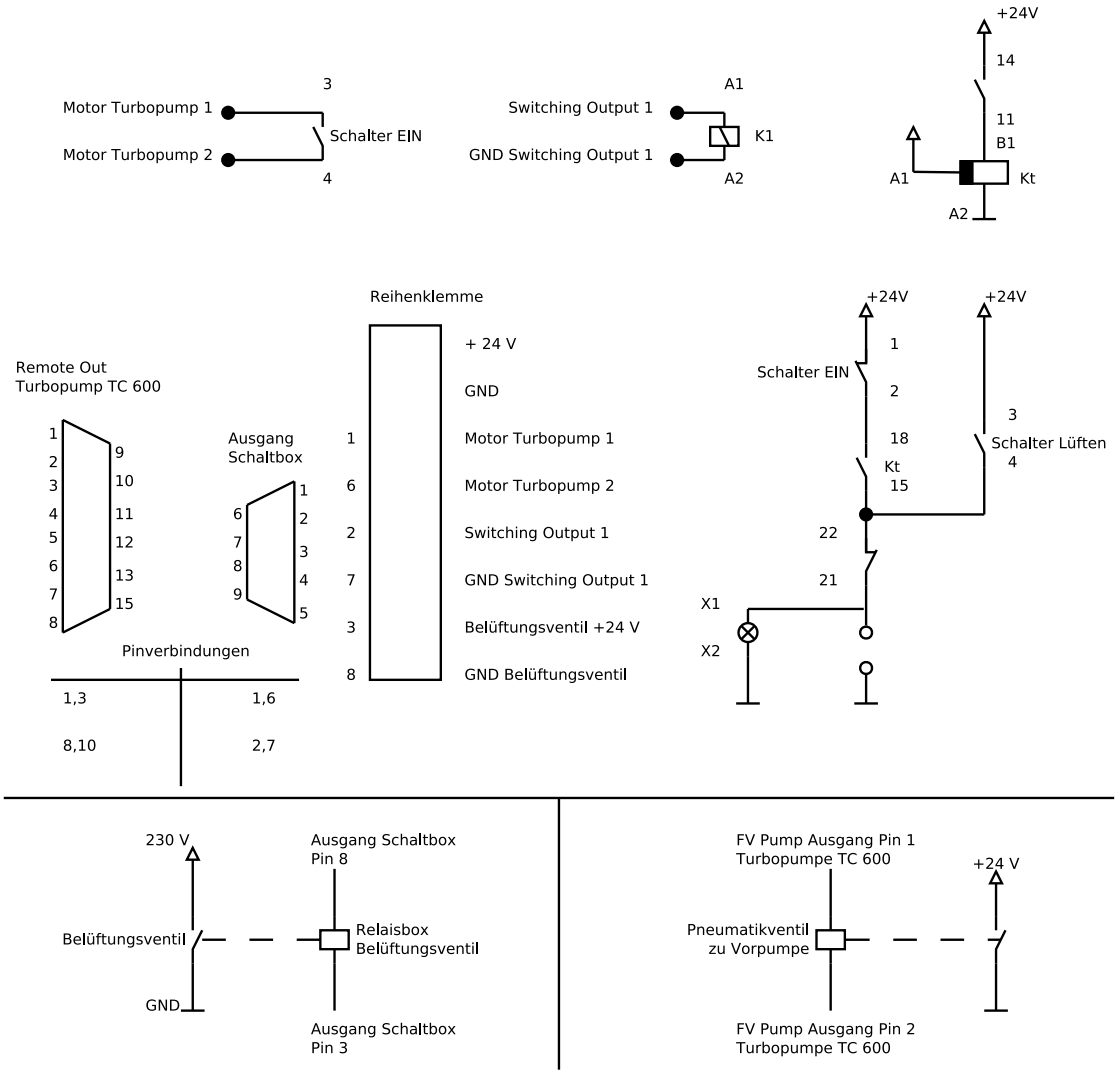
| Nummer | Name | Einstellung | |
|---|---|---|---|
| 010 | Pump Stat | OFF | wird über Knopf gesteuert |
| 023 | Motor TMP | OFF | |
| 794 | Param Set | 1 | DCU Manual Seite 10 |
| 025 | OpMode BkP | 2 | |
| 028 | OpMode rem | 0 | |
| 012 | Vent enab | ON | Turbopumpe automatisch bei 40% Drehzahl belüften |
| 030 | Vent mode | 0 | |
| 720 | Vent frequ | 40% | |
| 721 | Vent time | 600 | |
| 024 | Conf out 1 | 2 | siehe Kommentar unter Tabelle |
| 701 | Switch Pnt | 50% | |
| 719 | Switch Pnt 2 | 10% | |

Setzt zwei Schaltpunkte, die über den Ausgang der Turbopumpe „Switching output 1“ (siehe Manual Turbopumpe S14) ausgelesen werden können. Der im Manual eingezeichnete „Switching output 2“ ist ein Fehler und existiert nicht. Bei Erreichen von 50% der Maximaldrehzahl wird der Ausgang „Switching output 1“ aktiviert (24V), bei Unterschreiten von 10% der Maximaldrehzahl wird der Ausgang wieder deaktiviert.

## C.2. Testbetrieb

Die folgenden Einstellungen erlauben es das Ventil zur Vorpumpe zu öffnen, die Turbopumpe jedoch ausgeschaltet zu lassen.

| Nummer | Name | Einstellung | |
|---|---|---|---|
| 010 | Pump Stat | ON | |
| 023 | Motor TMP | OFF | |
| 794 | Param Set | 1 | DCU Manual Seite 10 |
| 025 | OpMode BkP | 0 | |

# D. Automatisierung Vakuum

Relaisschaltplan entwickelt von Maximilian Zach, Leiter der Elektronikwerkstatt des Atominstitus der TU Wien.