UNIVERSITÄT OSNABRÜCK

Diplomarbeit

Entwicklung einer Software für eine SPA-LEED-Oktolpolansteuerung

eingereicht von
Tim Röwekamp
Matrikel-Nummer: 925881

Osnabrück, den 13. Dezember 2013

Fachbereich Physik
AG Dünne Schichten und Grenzflächen
Prof. Dr. Joachim Wollschläger
Prof. Dr.-Ing. Elke Pulvermüller

Inhaltsverzeichnis

Inhaltsverzeichnis

1. Einleitung

Seltene Erden sind aufgrund ihrer elektrischen und chemischen Eigenschaften in den letzten Jahren sehr gefragt, da sie in den verschiedensten Gebieten verwendet werden können. Die interessanten Eigenschaften sind die Elektronenmobilität und die Sauerstoffmobilität, welche in den Bereichen der Mikro- und Nanoelektronik und der heterogenen Katalyse eine Rolle spielen [1, 2]. Wegen der Knappheit der Ressourcen und der voranschreitenden Miniaturisierung der Elektronik wird mit immer dünneren Schichten experimentiert.
In der AG „Dünne Schichten und Grenzflächen“ an der Universität Osnabrück werden von den interessanten Elementen ultradünne Schichten hergestellt und charakterisiert. Um diese auf ihre Struktur und Morphologie zu untersuchen, werden die Oberflächen mit **S**pot **P**rofile **A**nalysis - **L**ow **E**nergy **E**lectron **D**iffraction (SPA-LEED) untersucht. Der Vorteil zum konventionellen LEED ist die Möglichkeit die Form des zentralen (00)-Reflexes zu messen.
Die bisher benutze Ansteuerung, bestehend aus einer BurrBrown-Karte und einem Addierer für die im SPA-LEED benötigten Oktopol-Platten, ist schon in die Jahre gekommen und könnte jederzeit ausfallen. Um diese zu ersetzen, wurde in Zusammenarbeit mit der Elektronikwerkstatt eine neue Oktopol-Ansteuerung gebaut. In dieser Arbeit wurde eine neue Software für die Ansteuerung entwickelt, die neben einer Vereinfachung der Bedienung auch die Möglichkeit bieten soll mehrere Messungen automatisch hintereinander durchzuführen.
In Kapitel 2 werden die theoretischen Grundlagen, auf denen das SPA-LEED basiert vorgestellt. Im Anschluss erfolgt ein Überblick über den experimentellen Aufbau in Kapitel 3 und die Ansteuerung in Kapitel 4. Daraufhin stelle ich die verwendete Programmiersprache LabVIEW und die umgesetzten Konzepte in Kapitel 5 vor. In Kapitel 6 stelle ich das Programm vor. Die ersten mit der Software erstellten Messergebnisse werden in Kapitel 7 gezeigt. Abgeschlossen wird die Arbeit mit einer Zusammenfassung und einem kurzen Ausblick in Kapitel 8.

2. Theoretische Grundlagen

Dieses Kapitel erläutert das physikalische Konzept des in dieser Arbeit verwendeten SPA-LEED (**S**pot **P**rofile **A**nalysis - **L**ow **E**nergy **E**lectron **D**iffraction), sowie die dazu nötigen theoretischen Grundlagen.

2.1. Oberflächen- und Überstrukturen

Ist ein Festkörper aus periodisch angeordneten Einheitszellen aufgebaut, wird er als idealer Kristall bezeichnet. Wird der Kristall geteilt, so entstehen zwei halbunendliche Kristalle und an den Grenzflächen bilden sich ungesättigte Bindungen aus, die die strukturellen, chemischen und physikalischen Eigenschaften verändern. Das hat zur Folge, dass die Oberfläche eine vom Volumen verschiedene Periodizität aufweist. Diese Grenzfläche ist einige Atomlagen dick und wird als Oberfläche bezeichnet. Die 14 Bravais-Gitter beschreiben die Einheitszellen im Volumen, für die Oberfläche reduziert sich die Anzahl auf 5 Gitter (siehe Abbildung 2.1).

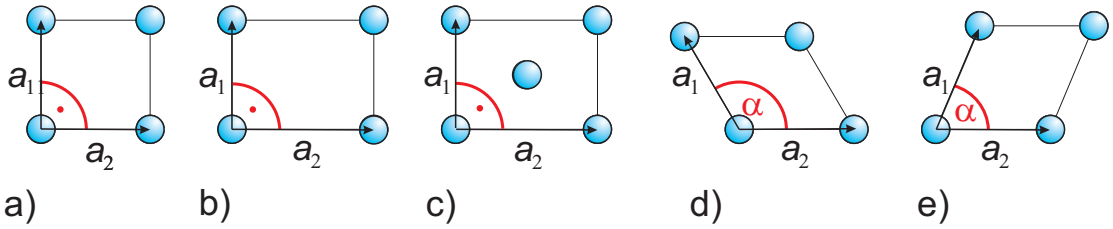


Abbildung 2.1: Die Bravais-Gitter für den zweidimensionalen Raum: a) quadratisch $[a_1 = a_2, \alpha = 90°]$, b) primitiv rechtwinklig $[a_1 \neq a_2, \alpha = 90°]$, c) rechtwinklig zentriert $[a_1 \neq a_2, \alpha = 90°]$, d) hexagonal $[a_1 = a_2, \alpha = 120°]$, e) schiefwinklig $[a_1 \neq a_2, \alpha \neq 90°]$. Entnommen aus [3].

Die Positionen der Atome auf den Gitterplätzen werden mathematisch beschrieben durch Translationsvektoren.

$$\vec{T} = h \cdot \vec{a}_1 + k \cdot \vec{a}_2 \tag{2.1}$$

Dabei sind $\vec{a}_1$ und $\vec{a}_2$ die Basisvektoren des Gitters und h und k die jeweiligen Miller'schen Indizes.

2.2. SPA-LEED-Theorie

Das SPA-LEED ist eine Weiterentwicklung des LEED durch Henzler et al. [4]. Das Beugungsbild wird nicht wie beim LEED auf einem Leuchtschirm abgebildet, durch die winkelauflösende Messmethode des SPA-LEEDs werden nur noch Elektronen vom Channeltron detektiert, die unter einem bestimmten Beugungswinkel gestreut wurden. Verwendet wird dies für laterale Scans im $\vec{k}$-Raum (vgl. Kapitel 2.5). Der Vorteil besteht darin, einen größeren Bereich des reziproken Raumes scannen zu können. Ein weiterer Vorteil durch eine bessere Auflösung ist die Möglichkeit, den zentralen Reflex zu messen und dessen Halbwertsbreite zu analysieren. Durch die Beugung von niederenergetischen Elektronen ist es möglich, die Beschaffenheit von kristallinen Oberflächen zu bestimmen. Insbesondere können Aussagen über die mittlere Terrassenbreite oder eine Facettierung der Oberfläche getätigt werden. Die De-Broglie-Wellenlänge

$$\lambda = \frac{h}{\sqrt{2m_e \cdot E}} \tag{2.2}$$

ordnet Elektronen einer bestimmten Energie eine Wellenlänge zu. Die verwendete Elektronenenergie liegt zwischen $10eV$ und $500eV$ was einer Wellenlänge zwischen $3,88\mathring{A}$ und $0,55\mathring{A}$ entspricht. Wegen der niedrigen Energie der Elektronen liegt auch die Eindringtiefe der Elektronen in das Material bei weniger als $10\mathring{A}$. Hiermit ist gewährleistet, dass die Elektronen nur an den obersten atomaren Lagen gestreut werden und damit nur die Oberfläche und nicht das Volumen zum Beugungsbild beiträgt.

2.3. Streuphase

Die Streuphase S ist eine dimensionslose Größe, mit der das Interferenzverhalten von an den atomaren Lagen oder Stufenkanten gestreuten Elektronen beschreiben kann. Sie beschreibt den Zusammenhang zwischen dem Gangunterschied Δg und der Energie der Elektronen E.

Mit Geometrieüberlegungen kommt man zur Bragg-Bedingung

$$\Delta g = S \cdot \lambda = 2d \cdot cos(\theta) \tag{2.3}$$

θ beschreibt den Einfallswinkel der Elektronenwelle. Der Gangunterschied ist damit als ein Vielfaches der Wellenlänge λ beschreibbar. Bei einem ganzzahligen Vielfachen und damit auch einer ganzzahligen Streuphase beobachtet man konstruktive Interfe-

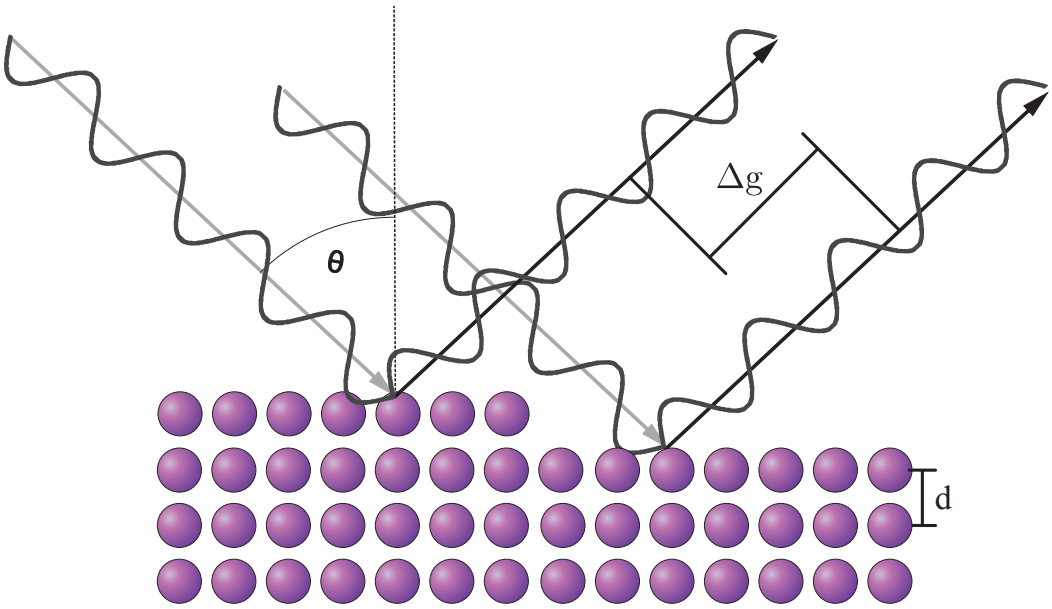


Abbildung 2.2: Beugung zweier Elektronenwellen an einer Stufenkante, d bezeichnet den Lagenabstand, Δg den Gangunterschied und θ den Einfallswinkel.

renz (In-Phase). Ist der Gangunterschied jedoch nur ein halbzahliges Vielfaches der Wellenlänge, so liegt destruktive Interferenz vor (Out-Phase). Mit der De-Broglie -Wellenlänge (siehe Gleichung 2.2), die den Zusammenhang zwischen der Energie der Welle und deren Wellenlänge beschreibt, stellt man nun die Energieabhängigkeit der Streuphase dar.

$$S = \frac{2d \cdot cos\theta \cdot \sqrt{2m \cdot E}}{h} \tag{2.4}$$

$$S \cong \sqrt{\frac{E[eV]}{37.6eV\mathring{A}^2}} \cdot d \cdot cos(\theta) \tag{2.5}$$

$$E[eV] \cong 37,6eV\mathring{A}^2 \cdot \left(\frac{S}{d \cdot cos(\theta)}\right)^2 \tag{2.6}$$

Mit bekanntem Lagenabstand d ist es somit möglich, die für die gewünschte Streuphase die nötige Elektronenenergie zu bestimmen. Insbesondere die Halbwertsbreite der Beugungsreflexe ist eine Funktion der Streuphase, während ein Reflex bei einer In-Phase scharf ist, verbreitet er sich in einer Out-Phase, die maximale Intensität nimmt ab, während die Integrale gleich bleibt (siehe Abbildung 2.3).

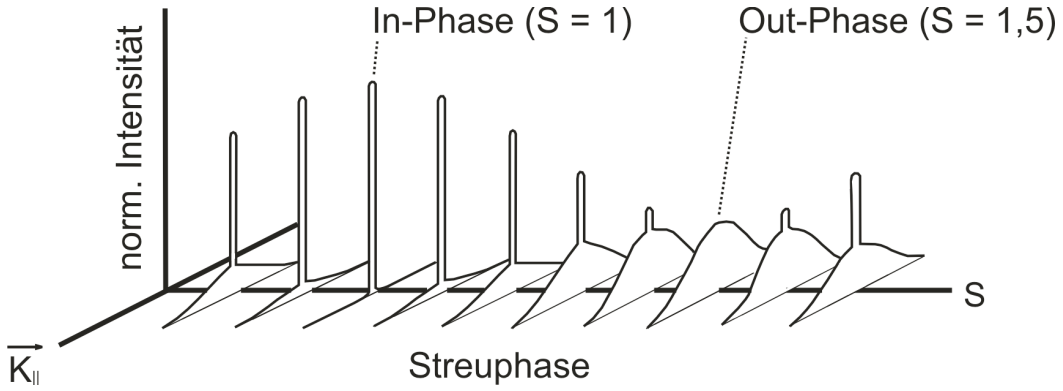


Abbildung 2.3: Darstellung der Abhängigkeit des Reflexprofils von der Streuphase. Entnommen aus [5].

2.4. Kinematische Beugungstheorie

Mithilfe der kinematischen Beugungstheorie ist es möglich, durch die Untersuchung des Reflexprofils und der Lage der Reflexe, Aussagen über Defekte und die Oberflächenmorphologie zu tätigen. Dazu wird die Oberfläche in eine periodische Anordnung von säulenförmigen Einheitszellen unterteilt.

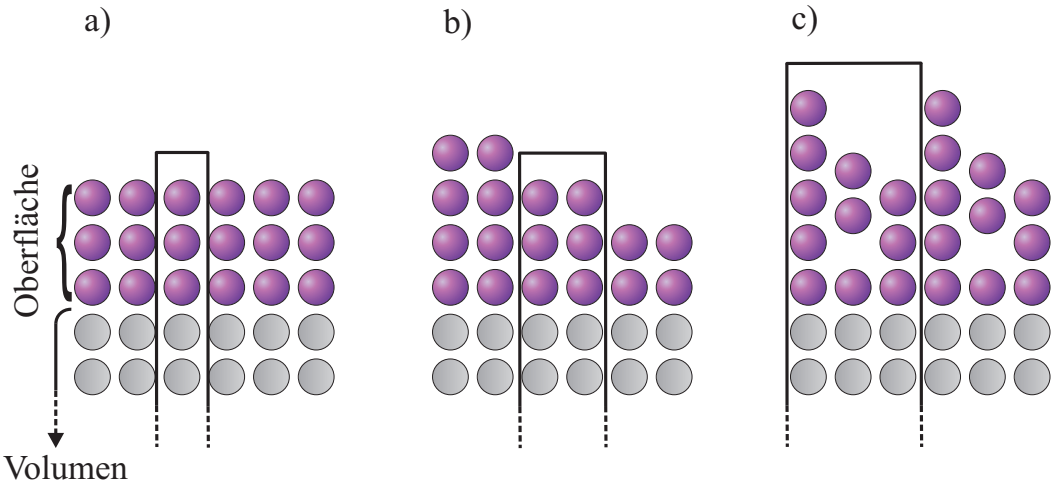


Abbildung 2.4: Schematische Darstellung möglicher Säuleneinheitszellen: a) ideale Oberfläche, b) Oberfläche mit atomaren Stufen und c) Überstrukturen. Entnommen aus [6].

Es wird dabei nur die Vorwärtsstreuung betrachtet, da die Seitwärtsstreuung vernachlässigt werden kann. Das Streuverhalten an der n-ten Säule wird durch den Formfaktor f_n beschrieben. Die Fraunhofer-Näherung besagt, dass Wellen als ebene Welle an-

genähert werden können, wenn die Entfernung von Probe und Detektor deutlich größer ist, als die Wellenlänge des Elektronenstrahls. Für die Wellenfunktion, der an der 0-ten Säule gebeugten Elektronenwelle kann, gilt

$$\Psi_0(\vec{k}, \vec{k}', \vec{r}) = f_0(\vec{k}, \vec{k}') \cdot e^{i\vec{k}' \cdot \vec{r}} \tag{2.7}$$

dabei beschreiben $\vec{k}$ und $\vec{k}'$ die einfallende und gestreute Welle. Um die Gleichung auf eine beliebige Säule an der Stelle $\vec{r}_n$ zu erweitern, wird die Streuamplitude um den Streuvektor $\vec{K} = \vec{k} - \vec{k}'$ phasenverschoben. Die Wellenfunktion ergibt sich damit zu

$$\Psi_n(\vec{k}, \vec{k}', \vec{r}_n) = f_n(\vec{k}, \vec{k}') \cdot e^{i\vec{k}' \cdot \vec{r}} \cdot e^{i\vec{K} \cdot \vec{r}_n} \tag{2.8}$$

Aus dem Beugungsbild geht jedoch nur die Gesamtintensität I hervor und nicht die Amplitude einer Wellenfunktion Ψ_n. Mit der Annahme das der Formfaktor für jede Säule identisch ist ($f_n = f_0$), kann man die Gesamtintensität als Betragsquadrat der Amplituden ausdrücken

$$\begin{aligned} I &= \left| \sum_{n=0}^{N} \Psi_n^2 \right| && (2.9) \\ &= \frac{I_0}{N} \cdot \overbrace{|e^{i\vec{k}' \cdot \vec{r}}|^2}^{=1} \cdot |\sum_{n=0}^{N} \overbrace{f_n(\vec{k}, \vec{k}')}^{=F_{\text{Säule}}} \cdot e^{i\vec{K} \cdot \vec{r}_n}|^2 && (2.10) \\ &= \frac{I_0}{N} \cdot |F_{\text{Säule}}(E)|^2 \cdot |\sum_{n=0}^{N} e^{i\vec{K} \cdot \vec{r}_n}|^2 && (2.11) \end{aligned}$$

Der Gitterfaktor $G = \sum_{n=0}^{N} e^{i\vec{K} \cdot \vec{r}_n}$ beschreibt die periodische Anordnung der Säulen und damit auch die Lage der Reflexe.

2.5. Reziprokes Gitter

Der reziproke Raum dient als Modellvorstellung und lässt sich durch Fourier-Transformation in den Realraum überführen. Da eine Oberfläche betrachtet wird, sind die nächsten Atome senkrecht zur Oberfläche in Richtung des Vakuums im realen Raum näherungsweise unendlich weit entfernt. Dies führt zu unendlich nah beieinanderliegenden Gitterpunkten im reziprokem Raum, diese bilden dann die Beugungsstangen. Das Gitter reduziert sich wegen der Beugung an Oberflächen auf 2 Dimensionen, da-

mit wird die Oberfläche eines Kristalls durch ein 2D-Gitter mit den Basisvektoren $\vec{a}_1$, und $\vec{a}_2$ beschrieben. Für das reziproke Gitter mit den Basisvektoren $\vec{a}_1^*$, und $\vec{a}_2^*$ ergibt sich damit:

$$\vec{a}_1^* = 2\pi \cdot \frac{\vec{a}_2 \times \vec{n}}{|\vec{a}_1 \times \vec{a}_2|} \tag{2.12}$$

$$\vec{a}_2^* = 2\pi \cdot \frac{\vec{n} \times \vec{a}_1}{|\vec{a}_1 \times \vec{a}_2|} \tag{2.13}$$

$\vec{n}$ ist der Einheitsvektor, der senkrecht auf der Oberfläche steht. Ein Vektor $\vec{k}$ aus dem reziproken Raum entspricht einer ebenen Welle der Form $e^{i\vec{k}\cdot\vec{r}}$ im realen Raum. Der reziproke Raum wird deshalb auch $\vec{k}$-Raum genannt. Nun definiert man den parallelen Anteil des Gittervektors $\vec{G}_\parallel$ mithilfe der ganzzahligen Miller'schen Indizes h und k, mit denen sich im reziproken Raum jeder Gitterplatz erreichen lässt.

$$\vec{G}_\parallel = h \cdot \vec{a}_1^* + k \cdot \vec{a}_2^* \tag{2.14}$$

2.6. Ewald-Konstruktion

Die Ewald-Konstruktion veranschaulicht die Laue-Bedingung. Diese ist das Analogon zur Bragg-Bedingung im reziproken Raum und besagt, das konstruktive Interferenz nur dort entsteht, wo der Streuvektor $\vec{K} = \vec{k}' - \vec{k}$ einem Gittervektor $\vec{G}$ entspricht. ($\vec{k}$ ist der Wellenvektor der einfallenden Welle und $\vec{k}'$ der Wellenvektor der reflektierten Welle). Die lässt sich nur durch die parallelen Anteile vom Streuvektor und Gittervektor $\vec{K}_\parallel = \vec{G}_\parallel$ erfüllen. Dies gilt an allen Stellen, an denen die Ewald-Kugel eine Beugungsstange schneidet.

Der Radius der Ewald-Kugel entspricht der Länge des einfallenden Wellenvektors $\vec{k}$. Vorausgesetzt wird, dass die Länge der einfallenden und reflektierten Wellenvektoren gleich sind, damit die Spitze von $\vec{k}'$ auch wieder auf der Ewald-Kugel liegt. Dies folgt aus der Energieerhaltung und es werden dementsprechend nur elastisch gestreute Elektronen betrachtet. Stimmt der Ort, an dem der reflektierte Wellenvektor auf die Ewald -Kugel trifft mit einer Beugungsstange überein, dann beobachtet man im Beugungsbild eine konstruktive Interferenz. Die Größe der Ewald-Kugel ist proportional zur gewählten Energie der Elektronen:

$$E = \frac{p}{2m} \tag{2.15}$$

$$= \frac{\hbar^2 \cdot \left|\vec{k}\right|^2}{2m} \tag{2.16}$$

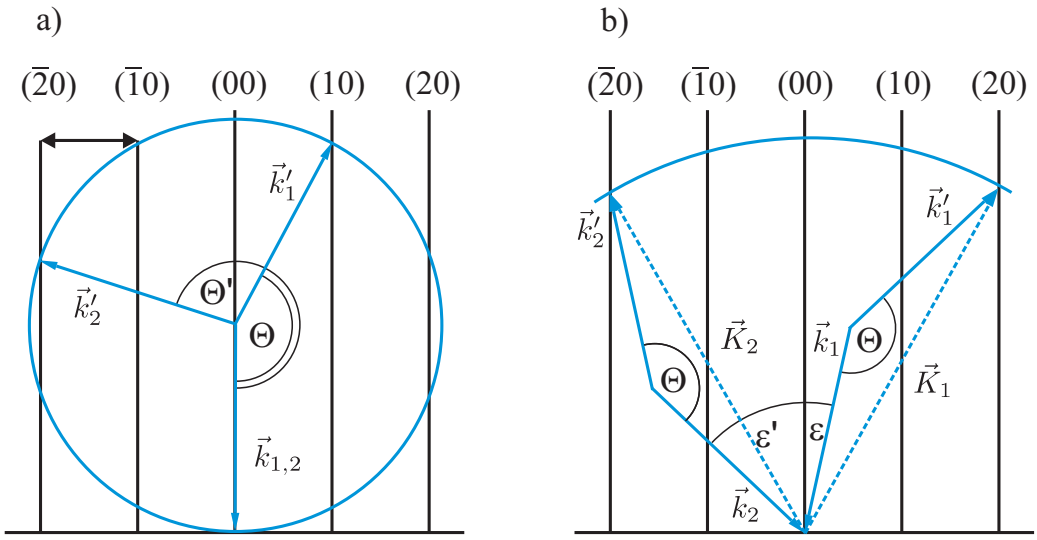


Abbildung 2.5: EWALD-Konstruktion für a) konventionelles LEED und b) für SPA-LEED. Dabei bezeichnen $\vec{k}_1$ und $\vec{k}_2$ die Vektoren der einfallenden Wellen und $\vec{k}_1'$ und $\vec{k}_2'$ die Vektoren der gebeugten Wellen. Der Beugungswinkel Θ zwischen einlaufender und gebeugter Welle beim konventionellen LEED, entspricht dem Winkel zwischen Elektronenkanone und Channeltron beim SPA-LEED. Der reziproke Streuvektor wird durch $\vec{K}_1$ und $\vec{K}_2$ beschrieben. ϵ beschreibt den Einfallswinkel der Elektronen auf die Oberfläche, der durch die Oktopolplatten variiert werden kann. Entnommen aus [5].

3. Experimentelle Grundlagen

3.1. Ultrahochvakuum

Als Ultrahochvakuum (UHV) werden Drücke zwischen 10^{-7} und 10^{-12}mbar bezeichnet. Es wird verwendet, um ein Verschmutzen der Probe durch Absorption und Reaktion mit Restgasteilchen zu verhindern. Die Drücke entsprechen Moleküldichten von 10^4 bis $10^9 \frac{1}{cm^3}$ und einer mittleren freien Weglänge von 10^5km. Dadurch verlängert sich die Zeit, in der sich eine geschlossene Lage von Fremdpartikeln auf die Probe legt auf mehrere Stunden. Ebenso stoßen die vom SPA-LEED emittierten Elektronen seltener mit Restgasmolekülen zusammen und können eher die Probe und den Detektor erreichen.
Der Druck wird mit einer Kombination aus verschiedenen Pumpen erreicht, da eine Pumpenart jeweils nur in einem bestimmten Druckbereich eingesetzt werden kann. Tabelle 1 zeigt die Pumpenarten und deren Einsatzbereich.

Pumpe	Druckbereich [mbar]
Drehschieberpumpe	10^3 bis 10^{-3}
Turbomulekularpumpe	10^{-2} bis 10^{-10}
Ionengetterpumpe	10^{-5} bis 10^{-11}

Tabelle 1: Einsatzbereich der verwendeten Pumpen.

3.2. Der SPA-LEED-Aufbau

Abbildung 3.1 zeigt die Kammer, in der das für diese Arbeit verwendete SPA-LEED eingebaut ist. Es basiert auf der im Jahre 1927 entdeckten Elektronenbeugung von Davison und Germer. Anders als beim LEED, werden beim SPA-LEED die Elektronen durch eine Ablenkeinheit, bestehend aus zwei Oktopolen, in einem bestimmten Winkel auf die Probe gelenkt. Dadurch lassen sich Beugungsbilder aufnehmen, bei denen der zentrale Reflex nicht durch die Elektronenkanone verdeckt ist. Das SPA-LEED (Abbildung 3.2) setzt sich zusammen aus einer Elektronenkanone, einer Ablenkeinheit und einem Channeltron als Detektor.

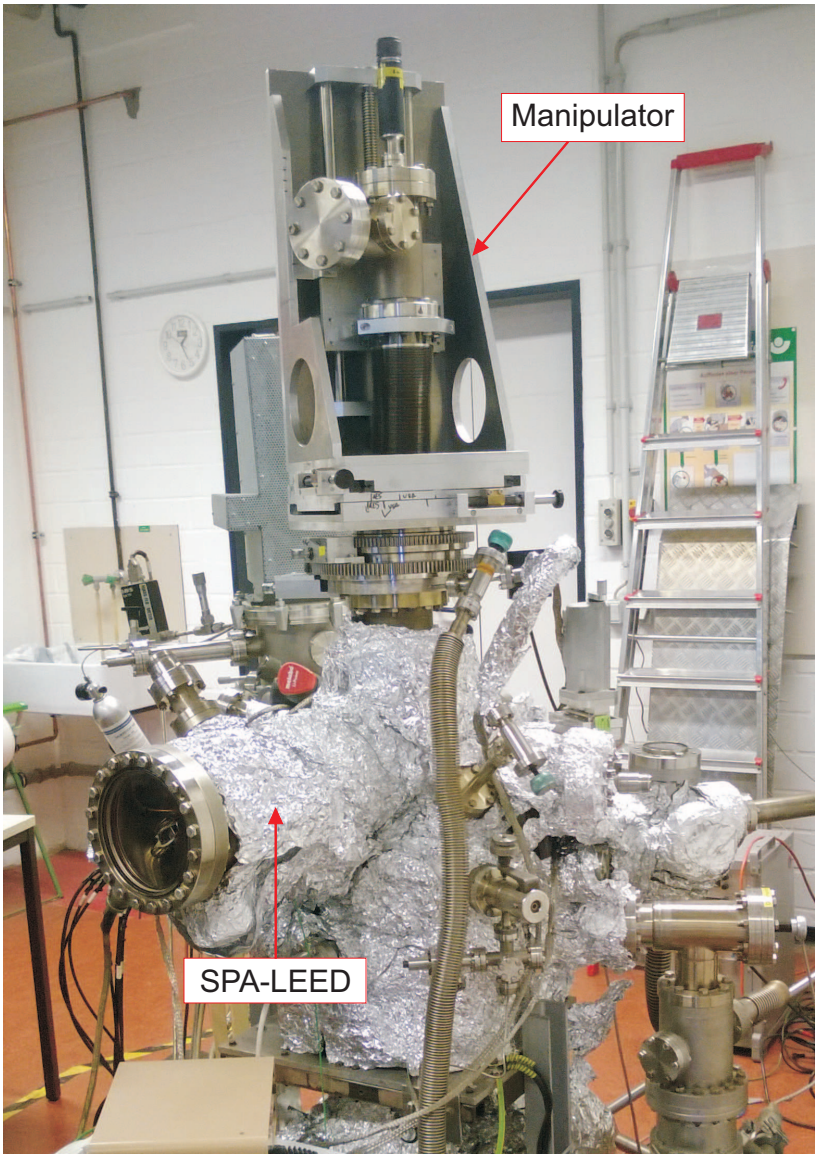


Abbildung 3.1: Bild der verwendeten UHV Kammer.

Der Hauptbestandteil der Elektronenkanone ist die Kathode, die aus einem dreieckigem Wolframfilament besteht, deren Spitze sich in einem Wehneltzylinder befindet. Das Potential zwischen der Kathode und der Anode beschleunigt die Elektronen und legt deren Energie E fest und damit auch Wellenlänge λ und Wellenvektor $\vec{k}$. Danach durchläuft der Strahl zwei Teillinsen, die diesen nochmal bündeln, und damit die Größe des Strahls auf der Probe festlegen.
Die Ablenkeinheit besteht aus drei Oktopolen. Die vorderen und hinteren Oktopolplatten weisen dabei eine entgegengesetzte Polung auf. Anders als die äußeren Plat-

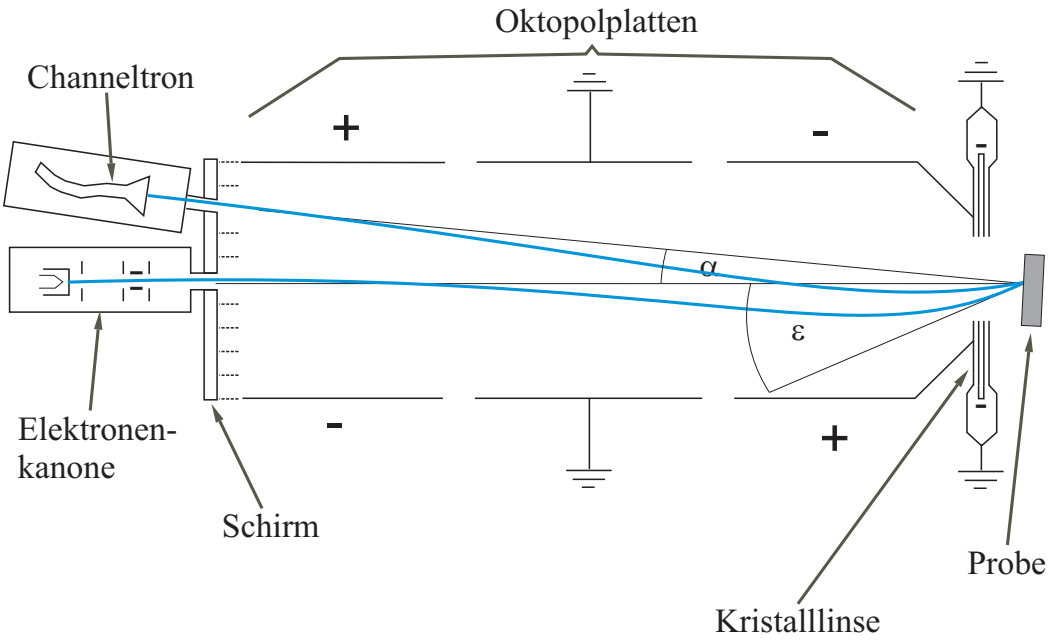


Abbildung 3.2: Schematischer Aufbau des verwendeten SPA-LEED-Aufbaus. Ein möglicher Strahlengang der Elektronen ist durch die blaue Linie gekennzeichnet. α bezeichnet den Winkel zwischen Elektronenkanone und Channeltron, und ϵ den Einfallswinkel der Elektronen auf die Probe. Entnommen aus [5].

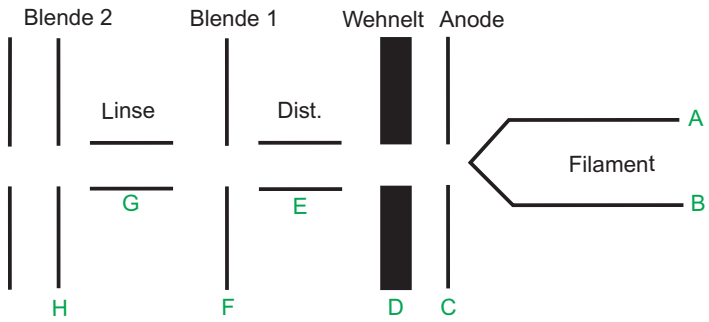


Abbildung 3.3: Schematischer Aufbau der Elektronenkanone. Die Anschlüsse wurden entsprechend Tabelle 6 markiert. Entnommen aus [7].

ten, liegen die Platten des mittleren Oktopols auf Masse. Durch den Aufbau kann der Elektronenstrahl mit verschiedenen Einfallswinkeln auf die Probe gelenkt werden. Dadurch kann der $\vec{k}$-Raum durch bestimmte Spannungsänderungen an den Platten in lateraler Richtung abgetastet werden. Die beiden Oktopole werden durch vier verschiedene Spannungen gesteuert. Diese werden durch eine Widerstandsmatrix (Span-

nungsverhältnisse sind in Abbildung A.1 dargestellt) auf die Platten aufgeteilt, wobei der hintere Oktopol auf einem kleineren Potential liegt als der vordere Oktopol. Das Verhältnis der Oktopole zueinander ist vom Abstand der Probe abhängig.
Zwischen der Ablenkeinheit und der Probe ist noch eine Kristalllinse verbaut. Um ein scharfes Bild im Detektor zu bekommen, werden diese Linse und die in der Elektronenkanone so eingestellt, dass der Elektronenstrahl gebündelt und auf die Detektorebene fokussiert wird.
Bevor die gebeugten Elektronen den Detektor erreichen, müssen diese einen Supressor passieren, der diejenigen Elektronen herausfiltert, die durch einen inelastischen Stoß Energie verloren haben. Als Detektor wird ein Elektronenvervielfacher (Channeltron) eingesetzt, dessen verstärkte Signale mit dem PC ausgewertet werden.
Für die Steuerung der Ablenkspannung und die Aufnahme der Signale diente bisher die Software „SPA5.6“. Mit dieser standen 2 Aufnahmemodi zur Verfügung einmal die Möglichkeit mit 2D-Scans eine Ebene des $\vec{k}$-Raums senkrecht zu den Beugungsstangen aufzunehmen und 1D-Scans („Linescan“), bei denen nur eine Linie in einer Ebene abgetastet wird. Bei beiden Modi wird jeweils die gemessenen Counts pro Sekunde über die Ablenkung aufgetragen.

4. Oktopol-Steuerung

Die bisher verwendete Ansteuerung der Oktopol-Platten besteht aus einer BurrBrown-Karte, diese ist über die ISA-Schnittstelle mit dem PC verbunden. Ebenso ist der verwendetet Addierer ein Eigenbau, für den keine genaue Dokumentation vorhanden ist. Da bei einem Ausfall einer Komponente ein Ersatz nicht leicht beschafft werden kann, wurde schon im Vorfeld dieser Arbeit damit begonnen, in Zusammenarbeit mit der Elektronikwerkstatt, eine neue Oktopol-Ansteuerung zu entwickeln. Im gleichen Zuge wurde ebenfalls der Vorverstärker verändert und getauscht.

4.1. Steuerung für Elektronenkanone und Linsen

Die Ansteuerung für die Elektronenkanone ist das einzige Steuerungsgerät was unverändert übernommen wurde. Das LEYBOLD-Gerät steuert die Einstellungen für die Elektronenkanone (Stromstärke für das Filament, Spannungen für die Anode und die Linsen), sowie die Einstellung der Kristalllinse zwischen Probe und Ablenkeinheit und die Betriebsspannung des Channeltrons.
Geändert wurde hier nur die Einstellung der Elektronenenergie. Die Eingabe erfolgt nicht mehr manuell über einen Drehknopf, dessen Eingabe immer etwas ungenau und zeitverzögert war, sie wird nun per Software geregelt, dabei wird der Energie-In-Anschluss des LEYBOLD-Geräts mit dem Energie-Out der neuen Ansteuerung verbunden. Das LEYBOLD-Gerät erwartet am Eingang eine Spannung zwischen 0V und 100V und stellt die Spannung für die Elektronenenergie proportional dazu ein, mit dem Verhältnis $1V \mathrel{\hat{=}} 50eV$.

4.2. Vorverstärker

Vorverstärker sind nach folgendem Prinzip aufgebaut: Zuerst verstärkt ein Ladungsverstärker das Signal. Danach folgt ein Differenzierer mit Pole-Zero-Cancellation[1], der das Signal in Pulse umwandelt. Am Ende wird das Signal durch einen Discriminator-Pulsformer für die Verarbeitung am PC aufbereitet. Der Pulsformer wandelt ein beliebiges Eingangssignal in Pulse mit festen Werten (Frequenz, Amplitude, Dauer, ...) um.

[1] Die Pole-Zero-Cancellation schneidet die Überschwingungen (Undershoot), die beim Differenzieren entstehen, ab.

Der alte Vorverstärker war aufgebaut aus einem High-Speed-Vorverstärker mit nachgeschaltetem Komparator und retriggerbarem Monoflop. Der Komparator (Schwellendiskriminator) schaltet das Signal nur durch, wenn dieses größer ist als eine angelegte Vergleichsspannung. Wird der Monoflop durch ein Signal getriggert, wechselt dieser seinen Zustand und leitet das Signal für eine bestimmte Zeit weiter an den PC. Sollte während der Monoflop getriggert ist ein weiteres Signal kommen, so wird die interne Zeit zurückgesetzt und das Signal verlängert sich. Der kritische Punkt bei diesem System ist die Einstellung der Komparatorschwelle, da der Aufbau wegen der hohen Bandbreite zu Eigenschwingungen neigt.

Der neue Vorverstärker wurde im Vergleich zum Alten nur an einer Stelle geändert, die Pole-Zero-Cancellation wurde durch eine Pulskompressionsstufe ersetzt. Da keine Pulshöhenanalyse benötigt wird, vereinfacht dies die Einstellung des Diskriminators. Ein Nachteil ist, das man den Rauschuntergrund mit dem Diskriminator schlecht trennen kann. Abhilfe schafft das Messen des Rauschuntergrund und spätere Abziehen von der Zählrate.

4.3. Ansteuerung

Die bisherige Ansteuerung der Oktopol-Platten bestand aus einer BurrBrown-Karte als Steuereinheit und dem Addierer, der die Oktopol-Platten mit Spannung versorgt. Bei diesem Aufbau musste der Offset Manuell am Addierer eingestellt.

Die Steuereinheit wurde durch einen Mikrocontroller (Atmel AVR AT-Mega 2560) ersetzt, dieser übernimmt die Kommunikation mit der Software, die Einstellung einzelnen Messparameter und die Durchführung der Messungen. Als Schnittstelle für die Kommunikation mit dem PC wurde USB gewählt, diese ist weit verbreitet und lässt sich mit jedem Betriebssystem verwenden. Die Vorspannungen vor den Endstufen werden nun von sechs 16bit-DACs[2] erzeugt, drei davon sind jeweils hintereinandergeschaltet und sind für die Spannungen für die X- bzw. Y-Komponente verantwortlich. Der erste DAC hat die Aufgabe der Messfenstergröße und ist einstellbar in einem Bereich von $U_{\text{Größe}} = 0\text{V}$ bis 10V. Der nächste DACs ist für die Positionierung des Elektronenstrahls verantwortlich, er lässt sich im Bereich $U_{\text{Position}} = (-1)\text{V}$ bis 1V einstellen und wird mit der Spannung des ersten DACs multipliziert. Der letzte DAC bietet die Einstellung für den Offset zwischen $U_{\text{Offset}} = (-5)\text{V}$ bis 5V, da dieser Bereich für die

[2] Ein DAC (Digital-Analog-Converter) erzeugt für einen digitalen Wert ein bestimmtes analoges Signal, die hier verwendeten 16bit-DACs erzeugen für einen Dezimalwert zwischen 0 und 65535 eine bestimmte Spannung.

Praxis viel zu groß ist und eine genauere Einstellung besser ist, so wird nur die Hälfte der eingestellten Spannung hinzuaddiert. Damit ergibt sich eine halb so große Einstellung für den maximalen Offset, dafür ist der Abstand zwischen zwei einstellbaren Spannung ebenfalls halbiert worden. Für die Spannung vor den Endstufen ergibt sich damit

$$U_{X,Y} = U_{\text{Größe}} \cdot U_{\text{Position}} + \frac{1}{2} U_{\text{Offset}} \tag{4.1}$$

Diese beiden Spannungen für X und Y werden dann durch die verbauten Endstufen vervielfacht. Insgesamt wurden acht einzeln auswechselbare Endstufen verbaut, jeweils vier für die vorderen[3] und hinteren[4] Oktopolplatten. Die Verstärkung kann durch ein Potentiometer, welches durch ein Loch in der Frontpartie erreichbar ist, einzeln verstellt werden. Dies ermöglicht es das Verhältnis des vorderen Oktopols zum hinteren Oktopol einzustellen, andererseits kann der Benutzer durch Verstellen der einzelnen Achsen Verzerrungen minimieren. Mithilfe der am Gerät verbauten Matrixanzeige und den internen Kalibrierungsmethoden lässt sich die jeweilige Verstärkung der Endstufen anzeigen. Die verschiedenen Methoden bieten jeweils unterschiedliche Vorspannungen für die Endstufen. Gemessen wird dabei die Spannung nach den Endstufen, dies erfolgt mit einem 16bit-ADC[5] mit einer Vergleichsspannung von 150V, dabei entspricht ein angezeigter Wert von +32767 einer Spannung von 150V und -32767 einer Spannung von -150V. Der Modus zum Anzeigen der Werte befindet sich im Menü unter dem Punkt *Calibrate Span*, hier werden Spannungen für X und Y generiert und die entsprechende Spannung nach den Endstufen angezeigt. Die vier Einstellungen stellen vor den Endstufen folgende Spannungen ein: MAX = 6,66V; MIN = -6,66V; 1\3 MAX = 3.33V und 0 = 0V. Das Gerät gibt am hinteren Ausgang (8pol. Stecker in Abbildung 4.1) die Spannung der Endstufen aus, diese wird dann noch durch eine Widerstandsmatrix, die sich in der Box mit dem Stecker befindet, auf die einzelnen Platten verteilt. Die Spannungsverhältnisse sind in Abbildung A.1 dargestellt. Über die Steckerbox laufen auch die Anschlüsse für die Kristalllinse und den Repeller.
Ein weitere Neuerung ist der eingebaute Tiefpassfilter, der hinter die Endstufen geschaltet wurde. Dieser soll das Rauschen der Oktopolplatten minimieren, wenn eine Messung mit sehr kleiner Messzeit pro Messpunkt durchgeführt wird, da bei diesen Messungen die X-Komponente sehr schnell geändert wird, sollen hierdurch Schwin-

[3] Verbaut wurden x10 bis x15-fach Multiplikatoren, bezeichnet mit +XF, -XF, +YF und -YF
[4] Verbaut wurden x8 bis x12-fach Multiplikatoren, bezeichnet mit +XR, -XR, +YR und -YR
[5] Analog-Digital-Converter

Pin im Gerät	A1	A2	A3	A4	A5	A6	A7	A8
Endstufe	+XF	-XF	+YF	-YF	+XR	-XR	+YR	-YR

Tabelle 2: Zuordnung der Pins im Anschluss zu den Endstufen.

gungen vermieden werden. Der Filter lässt sich je nach Bedarf vor einer Messung ein- oder ausschalten. Ebenfalls neu ist der Schalter, um zwischen dem LEED und dem SEM[6]-Modus umzuschalten. Im SEM-Modus werden die vorderen Oktopolplatten abgeschaltet, dadurch ist ein Scan des Realraums möglich. Mit diesem kann beispielsweise die Position der Probe überprüft werden.

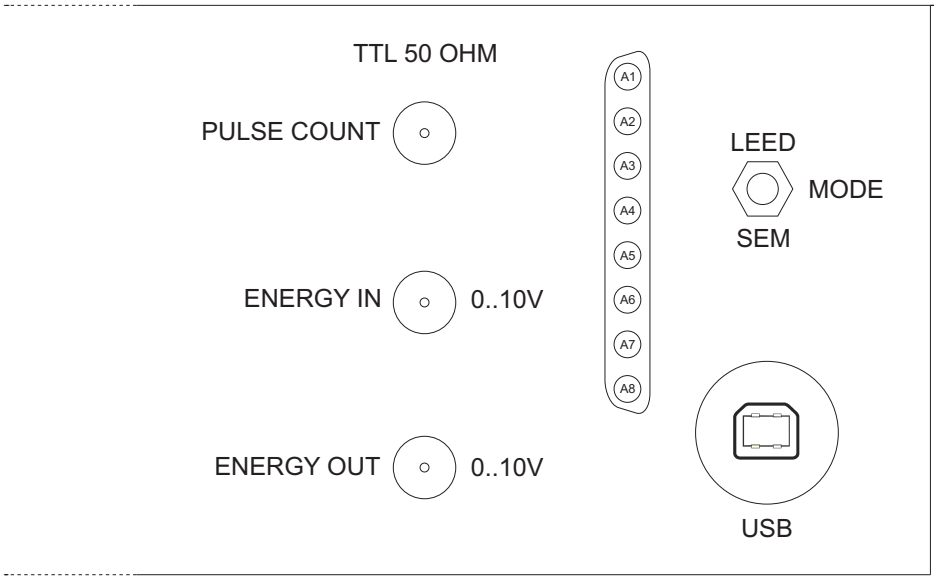


Abbildung 4.1: Rückansicht der neuen Oktopol-Steuerung, für die Pinbelegung des 8pol. Steckers siehe Tabelle 2.

4.4. Verbindung mit PC/Software

Angeschlossen wird die Steuerung an einem USB-Steckplatz am PC, dies ermöglicht den einfachen Austausch des Messcomputers, die Voraussetzung eines bestimmten Steckplatzes auf dem Mainboard entfällt damit. Obwohl das Gerät per USB angeschlossen ist, erfolgt die Kommunikation innerhalb der Software mit dem Übertragungsstandard RS-232, da das Betriebssystem den Anschluss als virtuellen COM-Port

[6]Scanning Electron Microscope

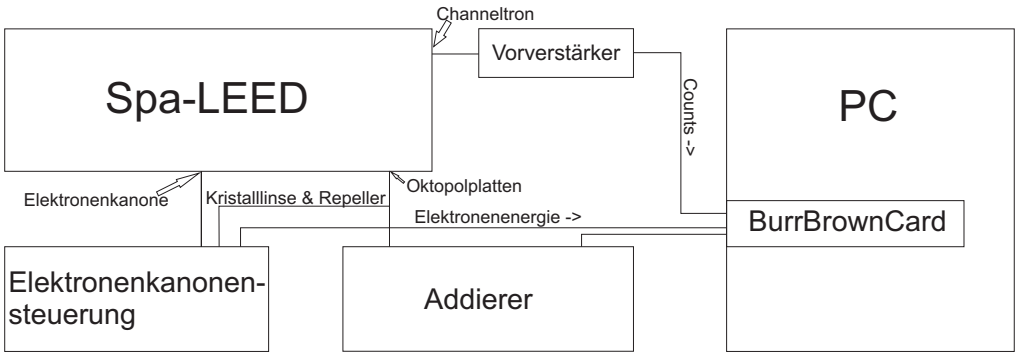


Abbildung 4.2: Schematischer Aufbau der Geräte und Verbindungen vor den Änderungen.

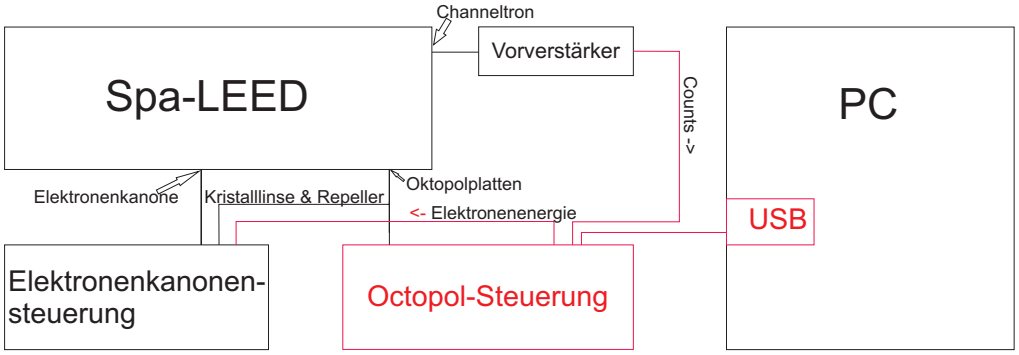


Abbildung 4.3: Schematischer Aufbau der Geräte und Verbindungen nach den Änderungen. Alle Änderungen sind rot markiert.

(VCP) zur Verfügung stellt. Für das Programm ist es nicht erkennbar, ob es über einen realen oder virtuellen COM-Port kommuniziert, da die Einstellungen gleich bleiben. Tabelle 3 zeigt die wichtigsten Einstellungsparameter und die Einstellungen um eine Verbindung mit der Oktopol-Steuerung herzustellen. Der verbaute Mikrocontroller wandelt die über USB kommenden Signale passend um und parst diese, um die entsprechende Methode mit den angegebenen Werten auszuführen.

4.4.1. Befehle

Eine Liste mit allen Befehlen und Rückgaben befindet sich in im Anhang Tabelle 7. Befehle sind wie folgt aufgebaut:

$$@Command\ P1\ P2*$$

Das @ und $*$ bezeichnen den Beginn und das Ende des Befehls. Die Art des Befehls wird mit *Command* angegeben, dadurch wird die Anzahl der erwarteten Werte und

Parameter	Einstellungen	Oktopolsteuerung
Baud	Anzahl der übertragenen Symbole pro Sekunde	115.200
Data bits	Anzahl der Datenbits eines Symbols	8
Parity	Bit für Fehlerkontrolle	n
Stop bits	Zeigt das Ende eines Symbols	1
Flow-Control	Verfahren für den Datenfluss	keine
Termination Char	Zeichen für das Ende eines Befehls	\r\n

Tabelle 3: Wichtige Parameter einer seriellen Verbindung.

deren Datentyp[7] festgelegt. Um Falscheingaben zu verhindern, werden diese Werte im Mikrocontroller direkt geprüft und an die entsprechenden Grenzen angepasst, damit wegen falschen Werten keine Bauteile beschädigt werden können. An jeden Befehl wird beim Versenden automatisch ein Wagenrücklauf[8] \r\n angehängt. Der Mikrocontroller sendet für jeden erhaltenen Befehl eine Rückgabe. Für Befehle die keine Messung starten, wird eine Bestätigung zurückgesendet, anhand derer es möglich ist zu kontrollieren, ob der Befehl richtig gesendet und ausgeführt wurde. Bei Scanbefehlen wird im Anschluss nach den Messwerten ebenfalls eine Bestätigung zurückgesendet. Die Rückgabe der Messwerte erfolgt direkt während der Messung, die Werte werden in der Reihenfolge der Messung zurückgegeben und sind durch ein Komma getrennt. Direkt nach Beendigung einer Messung/Zeile folgt ein \r\n um das jeweilige Messende/Zeilenende zu markieren.

4.5. Messungen

Die Positionierung der Messpunkte erfolgt in einem Gitter mit 65536x65536 Punkten, wobei der Punkt P(32767,32767) den Nullpunkt markiert, bei dem alle Oktopolplatten auf Masse liegen. Das Gitter und damit der Abstand zweier benachbarter Punkte skaliert mit der Messfenstergröße mit. Es gibt 3 verschiedenen Messarten: Punktscan, Linescan und 2D-Scan. Für den Punktscan und den 2D-Scan existiert jeweils nur eine Methode. Angegeben werden muss für den Punkt-Scan die Position des Messpunktes

[7] bspw. 16bit unsigned (Dezimalwertebereich: 0-65535), 16bit signed (-32767- +32767) oder 32bit unsigned (0-4294967295)

[8] Carriage Return kurz: <CR>

(P1,P2) innerhalb des Messfensters in Koordinaten von 0-65535, was (-)100% bis 100% des Messfensters entspricht und die Messzeit(P3) in μs.

Wegen dem DAC für die Positionierung des Elektronenstrahls, können nur diskrete Koordinaten verwendet werden. Daraus folgt, dass nicht jede beliebige Anzahl an Punkten auf eine Zeile/Spalte des Gitter verteilt werden kann, wenn die Endpunkte auf dem Rand liegen und der Abstand äquidistant sein soll. Wegen der variablen Messfenstergröße wurde entschieden die Punkte äquidistant zu setzen. Dabei muss der Benutzer darauf achten, das Messfenster entsprechend zu vergrößern, damit die äußersten Messpunkte bei der gewünschten Spannung (Einfallswinkel) liegen. Die Vergrößerung wird aus der Gittergröße und der Messpunkteanzahl berechnet. Zuerst wird der Abstand zwischen den Punkten bestimmt (Gittergröße $-$ 1 geteilt durch Anzahl $-$ 1, (-1) da der erste Punkt nicht zur Schrittweite beiträgt und einen Gitterpunkt belegt.). Danach wird abgerundet (um ganzzahlige Schrittweiten zu erhalten) und wieder mit der Anzahl $-$ 1 multipliziert. Danach wird die Größe des Gitters durch die berechneten Schritte (+1 für den ersten Punkt) geteilt. In Abbildung 4.4 ist die Berechnung des Faktors dargestellt, wie sie in LabVIEW umgesetzt wurde, in Abbildung 4.5 ist die Vergrößerung für einige Werte grafisch dargestellt.

$$\text{Faktor Messfenstervergrößerung} = \frac{\text{Gittergröße}}{\left(\left\lfloor \frac{\text{Gittergröße}-1}{\text{Punkteanzahl}-1} \right\rfloor \cdot \text{Punkteanzahl} + 1\right)} \tag{4.2}$$

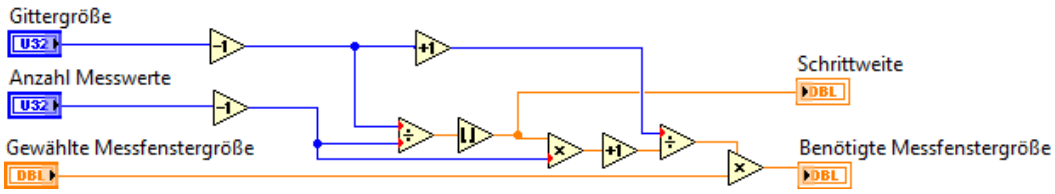


Abbildung 4.4: Blockdiagramm von *Messfenstervergrößerung.vi*. Berechnung des Faktors für die Messfenstervergrößerung in LabVIEW.

Die Punkte werden von der Mitte ausgehend gleichmäßig auf beiden Seiten verteilt. Dies passiert bei dem 2D-Scan für die Zeilen und Spalten. Um auch eine Messung des Nullpunktes durchzuführen, da in diesem der (00)-Reflex liegt, sollte die Anzahl der Messwerte stets ungerade gewählt werden, dann liegt der mittlere Wert der Messung im Nullpunkt.

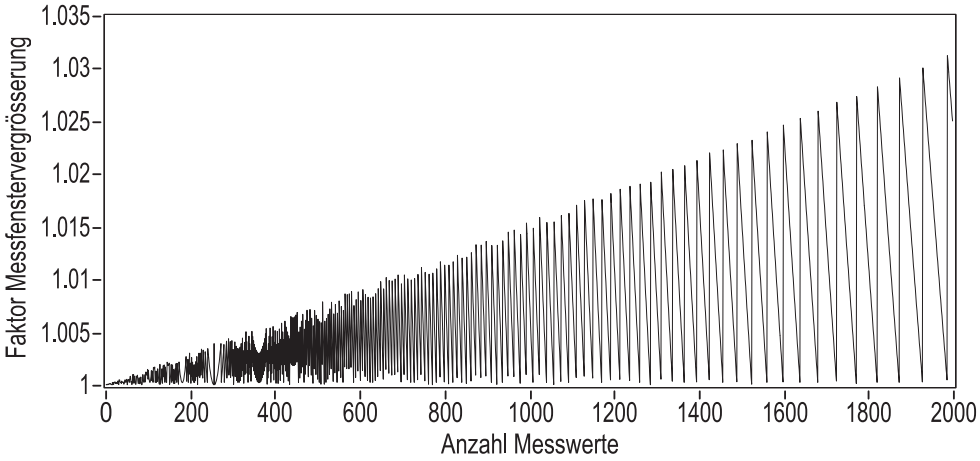


Abbildung 4.5: Benötigte Vergrößerung des Messfensters bei verschiedener Messwerteanzahl.

Für den Linescan gibt es vier verschiedene Arten:

- ***scanHline*** Horizontaler Scan mit äquidistanten Schritten auf der X-Achse.
- ***scanVline*** Vertikaler Scan mit äquidistanten Schritten auf der Y-Achse.
- ***scanray*** Gedrehter Linescan mit äquidistanten Schritten auf dem Strahl.
- ***scanTray*** Gedrehter Linescan mit äquidistanten Schritten auf der X- oder Y-Achse.

Für die Linescans wird die Anzahl der Messpunkte vorher mit dem Befehl *@scp P1** festgelegt. Aufgeteilt werden die Punkte für die Methoden *scanHline* und *scanVline* in der gleichen Art wie für den 2D-Scan. Weshalb für die Linescans ebenfalls immer eine ungerade Anzahl an Messpunkten gewählt werden sollte. Bei *scanTray* werden die Punkte ähnlich verteilt, hierzu wird zuerst die entsprechende Achse bestimmt, für Winkel zwischen 45 Grad und 135 Grad werden die Punkte gleichmäßig auf der X-Achse verteilt und dann der dazugehörige Y-Wert bestimmt. Bei Winkeln zwischen 0 Grad bis <45 Grad und >135 Grad bis 180 Grad erfolgt dies andersherum. Der *scanray* bestimmt zuerst den Abstand der Messpunkte im Gitter und verteilt diese auf dem Strahl. Der Unterschied zwischen den beiden Methoden für den Winkelscan ist der Abstand zwischen zwei benachbarten Punkten, dieser ändert sich bei *scanray* nicht für verschiedene Winkel, während er sich für *scanTray* mit den Winkel ändert, da nur die X- bzw. Y-Komponente gleich bleibt.
Da für die Koordinatenberechnung bei den Winkelscans trigonometrische Funktionen verwendet werden und das Gitter nur diskrete Koordinaten verwendet, kommt es zu

kleinen Abweichungen durch Rundungen. Die Abweichung beträgt maximal $\frac{1}{\sqrt{2}}$ Gitterpunkte, umgerechnet auf die Messfenstergröße $\frac{1}{\sqrt{2} \cdot 32768} \cdot U_{\text{Größe}}$.

5. Programmiersprache und Konzepte

In diesem Kapitel stelle ich kurz die verwendete Programmiersprache und deren Konzepte vor.

5.1. LabVIEW

LabVIEW (**Lab**oratory **V**irtual **I**nstrumentation **E**ngineering **W**orkbench) wurde speziell von National Instruments für Aufgaben in der Messtechnik entwickelt. Es setzt auf die grafische Programmiersprache „G“, welche den Programmcode nach dem Datenflussmodell darstellt. Ein in LabVIEW geschriebenes Programm wird als VI (Virtuelles Instrument) bezeichnet und besteht aus zwei Komponenten, dem Frontpanel (FP), das die Benutzerschnittstelle[9] bereitstellt und der grafischen Darstellung des Programmcodes in einem Blockdiagramm. Die VIs können beliebig verschachtelt werden, wobei die untergeordneten VIs als Sub-VI bezeichnet werden, mit diesen lassen sich größere Algorithmen oder wiederkehrende Abläufe unkompliziert und platzsparend integrieren. Vergleichbar sind diese SubVIs mit den Methoden in textbasierten Sprachen. Ein SubVI kann dabei mehrere Eingänge (Werte mit denen gerechnet werden soll) und Ausgänge (Ergebnisse) besitzen, mit der Ausführung wird erst begonnen, wenn alle Eingänge mit Werten gefüllt wurden. Die Drähte im Blockdiagramm verbinden die Ausgänge mit den Eingängen und stehen dabei für den Weg der Daten. Diese können gesplittet, d.h. ein Ausgang kann mit mehreren Eingängen verbunden werden. Diese System lässt sich mit Leiterplatten vergleichen, wobei eingesetzte Bauteile den SubVIs entsprechen. Ebenfalls lässt sich das Blockdiagramm mit Ablaufdiagrammen vergleichen, so können diese nahezu direkt umgesetzt werden.

Für den Großteil von Aufgaben stehen schon vorgefertigte SubVIs zur Verfügung, die sogenannten ExpressVIs, diese stellen alles Grundlegende für Dateioperationen, Aufnahme von Messdaten und Kommunikation mit anderen Geräten zur Verfügung (Öffnen und Schließen der Schnittstelle, Schreiben und Lesen von Daten). LabVIEW unterstützt für die Kommunikation die gängigsten Protokolle (USB, RS-232[10], TCP/IP, UPD, ...).

Um Daten schnell einzulesen und zu verarbeiten, stellt LabVIEW eine einfache Realisierung von Multitasking zur Verfügung. VIs oder Schleifen, die nebeneinander angelegt wurden und in keiner direkten Abhängigkeit zueinanderstehen, werden auto-

[9] GUI (Graphical User Interface)
[10] Standard für den COM-Port

matisch parallel ausgeführt. Um auftretende Race-Conditions[11] zu umgehen, bietet LabVIEW die gleichen Möglichkeiten wie andere Programmiersprachen, bspw. Semaphoren. Dies bedeutet auch, dass bei bestimmten Ausführungen auf die Reihenfolge geachtet werden muss. Hierzu wird meist der Fehlerdraht verwendet, um die Reihenfolge festzulegen.

Die Erstellung der GUI erfolgt nicht über aufwendige Textbausteine, sondern folgt dem „Drag and Drop“ Prinzip. Zuerst wird das Element ausgewählt und dann an die richtige Stelle verschoben. Danach können noch über das Kontextmenü verschiedene Eigenschaften (bspw. Größe, Farbe, Beschriftung,...) eingestellt werden. Im Anschluss werden die Elemente im Blockdiagramm mit den entsprechenden Funktionen und der Ausgabe verbunden und das VI ist fertig. Die Elemente der GUI und alle anderen Elemente können über Eigenschaftsknoten und Methodenknoten angesprochen werden. Mit diesen Knoten lassen sich verschiedene Eigenschaften wie Größe, Aussehen und Verhalten auslesen und ändern.

5.2. Zustandsautomat

Ein Zustandsautomat oder auch endlicher Automat ist ein Modell aus der Informatik. Er besitzt verschiedene Zustände zwischen den je nach Eingabe gewechselt wird. In den Zuständen wird eine Aktion ausgeführt oder die Ausgabe verändert. Dabei wird der Automat über drei verschiedene Mengen und zwei Funktionen definiert, die in Tabelle 4 dargestellt sind. Beginnend beim Startzustand wechselt der Automat je nach Eingabe in einen anderen Zustand, es existieren auch Zustände bei denen der Übergang automatisch durchgeführt wird bspw. nach einer gewissen Zeit. Dies passiert solange, bis ein Endzustand erreicht wird und sich der Automat damit selbst beendet.

Ein Zustandsautomat lässt sich durch ein Zustandsdiagramm ausdrücken, dabei werden Zustände durch Rechtecke und die Übergänge durch Pfeile dargestellt. In Abbildung 5.1 ist ein Beispiel eines Zustandsdiagramm anhand einer Fußgängerampel dargestellt.

Zustandsautomaten eignen sich gut für komplexe Messaufgaben, da hier viele verschiedene Schritte nacheinander ausgeführt werden müssen. Insbesondere wenn einige Schritte nicht oder mehrfach ausgeführt werden müssen, dabei ist eine reine sequentielle Struktur nicht von Vorteil. Hierbei müsste für jede Möglichkeit eine neue eigene Struktur oder viele verschachtelte Fälle angelegt werden. Da Sprünge oder Mehrfach-

[11] Gleichzeitiger lesender und schreibender Zugriff auf eine Variable von mehreren verschiedenen Threads

Name	Beschreibung
Eingabemenge	Benutzereingaben, Antwort von einem Gerät
Ausgabemenge	Anzeige auf Monitor, Befehl an eine Gerät
Zustände	Warten auf Eingaben
Übergangsfunktion	ermittelt welcher Zustand als nächstes eingenommen wird
Ausgabefunktion	definiert die Ausgabe

Tabelle 4: Mengen und Funktionen eines Zustandsautomaten.

ausführungen in einer Sequenz nicht vorgesehen sind. Der Zustandsautomat bietet hier Vorteile, da hier der Folgezustand nicht erst am Ende des Zustandes festgelegt wird.

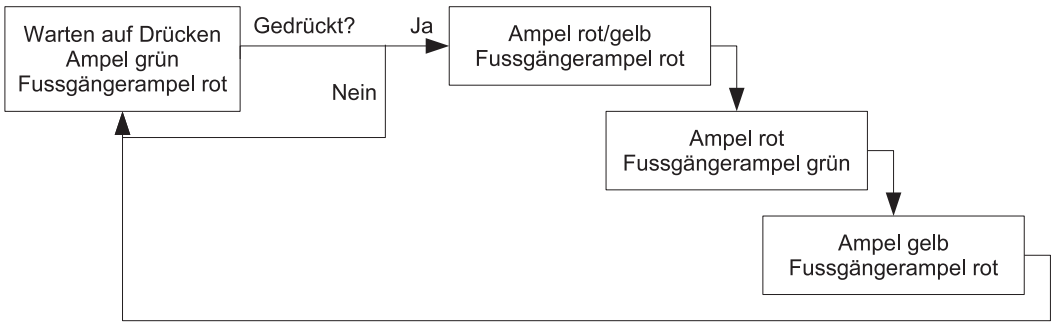


Abbildung 5.1: Beispiel eines Zustandsdiagramm einer Fußgängerampel, mit den verschiedenen Phasen/Zustanden die nacheinander durchlaufen werden.

5.2.1. Implementation des Zustandsautomaten

Einen Zustandsautomat ist in LabVIEW schnell erstellt, es wird nur eine While-Schleife, eine Case-Struktur und ein Enum benötigt. Während die While-Schleife dafür sorgt, dass nach einem abgearbeitetem Zustand ein Neuer aufgerufen werden kann, enthält die Case-Struktur die Zustände und mithilfe des Enums werden die Zustände festgelegt und ausgewählt. Der Folgezustand wird jeweils am Ende eines Zustands festgelegt und über ein Schieberegister[12] an der While-Schleife für den nächsten Durchlauf verfügbar gemacht. Enums[13] sind für diese Aufgabe ideal, da sie den einzelnen Zustän-

[12] Ein Schieberegister gibt den Wert den die Variable am Ende eines Schleifendurchlaufs besitzt, an den Anfang des nächsten Schleifendurchlaufs weiter.

[13] Enum ist ein Aufzählungsdatentyp, bei dem bestimmten Werten ein Name zugeordnet wird, so kann der Nutzer mit den Namen arbeiten, während das Programm intern mit Zahlen rechnet.

den nicht nur eine Nummer zuordnen, sondern einen Namen, womit diese einfacher zu identifizieren sind. Sie werden einmal definiert und können durch ein Dropdown-Menü entsprechend ausgewählt werden, was die Möglichkeit einer falschen Schreibweise bei der Verwendung von Strings verhindert. Durch die Typdefinition kann das Enum leicht geändert werden, da eine Änderung direkt für alle im Programm befindlichen Enums übernommen wird.

Alle Fenster die mehrere Funktionen implementieren, sind mithilfe eines Zustandsautomaten mit den gleichen Zuständen realisiert worden, um eine bessere Übersicht zu gewährleisten. Die Zustände sind *init, front, hide, data, display, wait* und *stop*. *stop* beendet das VI, *front* rückt das Fenster in den Vordergrund, *hide* minimiert das Fenster und *display* dafür sorgt, dass die angezeigten Daten aktualisiert werden. Der allgemeine Zustand *data* enthält verschiedene Unterzustände, unter anderem das Ändern der Achsen und Achsenbeschriftungen, die Auswahl erfolgt durch einen String. Die wichtigsten Zustände sind *init* und *wait*, während *init* einmalig am Start ausgeführt wird und die Grenzen für die Variablen festlegt oder Dateien einliest, wird *wait* immer wieder ausgeführt. In diesem Zustand wird wiederholt die Queue auf neue Befehle abgefragt und die Interrupts der Eventstruktur ausgeführt(siehe Kapitel 5.4). Damit sich die Funktionen nicht gegenseitig blockieren, besitzen beide einen Timeout. Aus der Queue können jeweils nur Elemente entnommen werden, wenn das „Target“ des Elements mit dem entsprechenden VI übereinstimmt.

Alle Zustände im Überblick:

- ***init*** Startzustand, wird einmal ausgeführt und legt die Variablengrenzen fest.
- ***front*** Holt das Fenster in den Vordergrund.
- ***hide*** Versteckt das Fenster.
- ***data*** Allgemeiner Zustand, enthält verschiedene Unterzustände, bspw. Ändern der Achsen.
- ***display*** Anzeigezustand, aktualisiert die angezeigten Daten.
- ***wait*** Standartzustand, liest die Queue aus und reagiert auf Frontpaneleingaben.
- ***stop*** Endzustand, stoppt das VI.

Main-Start.vi besitzt dabei andere Zustände, da es die Kommunikation mit der Steuerung übernimmt und die Messungen durchführt. Diese werden genauer in Kapitel 6.1 erläutert. Eine weitere Änderung ist die Auslagerung der Eventstruktur in eine weitere While-Schleife, um den Zugriff auf das Dateimenü während einer Messungen zu

ermöglichen.

5.3. Warteschleife Queue

Für die Übertragung von Daten zwischen zwei VIs bietet LabVIEW neben den globalen Variablen die Möglichkeit Warteschleifen (Queues) zu benutzen. Der Nachteil der globalen Variablen sind die auftretenden RaceConditions und die Möglichkeit immer nur einen Wert zu übergeben, ebenso muss der Empfänger benachrichtigt werden. Die Queue bietet die Möglichkeit die RaceConditions zu umgehen und sie kann beliebig viele Daten gleichzeitig aufnehmen. Durch die Benennung einer Queue ist es möglich gleichzeitig mit vielen verschiedenen VIs auf die Queue zuzugreifen, dabei haben alle VIs dieselben Rechte. Um mit einer Queue zu arbeiten, muss beim Initialisieren der Queue der Datentyp festgelegt werden, der verwendet werden soll. Hier bietet LabVIEW die Möglichkeit neue Datentypen zu erstellen, die Cluster, mit denen mehrere einzelne Datentypen zusammengefasst werden können. Das verwendete Verfahren der Queue ist das FIFO[14]-Verfahren, sie kann aber auch mit dem LIFO[15]-Verfahren betrieben werden. Durch Verbinden des Timeout-Eingangs mit einem Wert kann der Versuch ein Element zu lesen auf eine bestimmte Zeit beschränkt werden. Ohne einen Timeout besteht die Möglichkeit, dass die danebenliegende Eventstruktur blockiert wird, bis eine neuen Element zu Queue hinzugefügt wird.

Dies ist sinnvoll, wenn parallel noch auf andere Eingaben gewartet oder Berechnungen durchgeführt werden, damit diese nicht auf ein neues Element in der Queue warten müssen.

5.3.1. Implementation - Command Pipe und Data-Pipe

Für das Programm wurden zwei verschiedene Queues verwendet, damit Daten und Befehle getrennt sind. Die *Data-Pipe* ist für die Weiterleitung und Zwischenspeicherung der Daten zwischen Auslesen und Auswerten verantwortlich. Diese wird nur während einer Messung verwendet, damit das Lesen der Daten nicht auf die Auswertung warten muss, bzw. blockiert wird. Die Queue ermöglicht es auch durch die indirekte Abhängigkeit, dass die Daten schon ausgewertet werden, während die Messung noch läuft. Als Datentyp wurde String gewählt, da die Daten in diesem Format gelesen werden, und damit ohne Formatierung und Unterbrechung weitergegeben werden können.

[14] First In - First Out
[15] Last In - First Out

Als zweite Queue wurde die *Command-Pipe* erstellt, welche von allen VIs genutzt wird, die für Eingaben/Ausgaben verantwortlich sind. Als Datentyp wurde hierbei ein eigens erstellter Cluster aus verschiedenen Datentypen verwendet. Der Cluster besteht aus mehreren Enums um das Ziel „Target“ und den Befehl „Command“ bzw. „Command-Main“ festzulegen und aus verschieden Datentypen um bestimmte Werte zu übergeben. Ein Element kann nur vom angegebenem „Target“ aus der Queue entfernt werden, dies geschieht im jeweiligen *wait*-Zustand des VIs.

5.4. Event-Struktur

Um auf Eingaben der Tastatur oder Maus im Frontpanel zu reagieren, gibt es verschiedene Methoden, die beiden am Häufigst verwendeten Methoden stelle ich kurz vor. Die erste Variante wird Polling genannt, dabei wird immer wieder in einer Schleife der Zustand eines Elementes überprüft und mit dem vorherigem Zustand verglichen. Die Größe des Zeitintervalls, die ein Durchlauf benötigt, bestimmt die Reaktionszeit auf eine Eingabe. Je kleiner das Intervall gewählt wurde, desto schneller kann auf Eingaben im Frontpanel reagiert werden, dies führt aber zu einer erhöhten Systemlast, da die Schleife öfter durchlaufen wird. Durch Erhöhung des Intervalls, in dem die Schleife durchlaufen wird, lässt sich die Systemlast verringern, jedoch steigt die Reaktionszeit. Um Polling zu verwenden, muss der passende Mittelweg zwischen Reaktionszeit und Systemlast gefunden werden. Tritt nur selten eine Veränderung des Wertes ein, so wird die Schleife unnötig oft durchlaufen und erzeugt überflüssigen Stromverbrauch und Abwärme.

Die zweite Methode ist die Interrupt-Methode, die Änderung eines Bedienelements löst einen Interrupt aus. Diese werden direkt im Prozessor in einem Interruptzyklus verarbeitet. Die CPU sichert den aktuellen Zustand (Register)[16], lädt das Register für den Interrupt, führt dieses aus und sendet eine Benachrichtigung an die entsprechenden Programme. Im Anschluss lädt der Prozessor wieder den vorherigen Zustand und das Programm führt den Interrupt aus. Interrupts erzeugen dadurch wenig Overhead[17], da keine überflüssigen Schleifendurchläufe durchgeführt werden und bieten eine kurze Reaktionszeit. Die Reaktionszeit ist dabei abhängig von den Laufzeiten der Methoden der Interrupts. Diese sollten nur das Notwendigste enthalten und schnell ausführbar sein, um eine schnelle Abarbeitung der Interrupts zu garantieren. Für die Interruptverarbeitung stellt LabVIEW die Event-Struktur zur Verfügung. Damit ist es möglich auf

[16] Speicherbereich im Prozessor
[17] zusätzliche Daten, die nicht genutzt werden

alle zu erwartenden Ereignisse im Frontpanel zu reagieren, bspw. Eingaben von Tastatur und Maus über das Frontpanel oder Dateimenü, Änderungen der Fenstergröße, Schließen des Fenster, Werteänderungen von Variablen oder auch eigene Events.

5.4.1. Implementation der Event-Struktur

Den Ausschlag für die Interrupt-Methode und damit die Eventstruktur gaben die Häufigkeit der Eingaben und die damit verbundenen Methoden. Ein Großteil der Interrupts besteht nur aus dem Erstellen und Hinzufügen eines Clusters zur Command-Queue, um eine andere Methode (Zustand) aufzurufen, bzw. aus wenigen Rechenschritten plus Speicherung einer Variablen. Die Laufzeiten sind damit nur kurz und blockieren nicht das Auslösen neuer Events. Damit während einer Messung keine wichtigen Variablen geändert werden, kann durch Deaktivierung des Auslösers, die Ausführung der Events verhindert werden. Innerhalb der Energiescan-Methode wurde Polling eingebaut, um das Ende der Messung abzufragen. Es wurde jedoch eine große Schleifendurchlaufzeit gewählt und die Abfrage wird nur temporär und nicht dauerhaft durchgeführt.

6. Programm MultiSPA

Dieses Kapitel stellt die neu entwickelte SPA-LEED-Software namens „MultiSPA“ vor. Dabei gehe ich in den Unterkapiteln auf die erstellten Fenster ein und erläutere die wichtigen Funktionen und Elemente. Das Programm basiert dabei nicht auf einem festen Fenster, in dem alle Bedienelemente enthalten sind, sondern wurde aus vielen verschiedenen Fenstern aufgebaut. Der User kann selbst entscheiden was ihm angezeigt wird. Dadurch ist das Programm nicht auf eine Bildschirmauflösung bzw. ein Seitenverhältnis ausgerichtet. Die Abbildungen der einzelnen Frontpanel können sich je nach verwendetem Betriebssystem und eingestelltem Design unterscheiden, die gezeigten Screenshots wurden mit Win7 und dem Standarddesign erstellt. Eine Installationsanleitung zum Programm mit den benötigten Laufzeitbibliotheken befindet sich unter dem Punkt A.4.

6.1. Hauptfenster *Main-Start.vi*

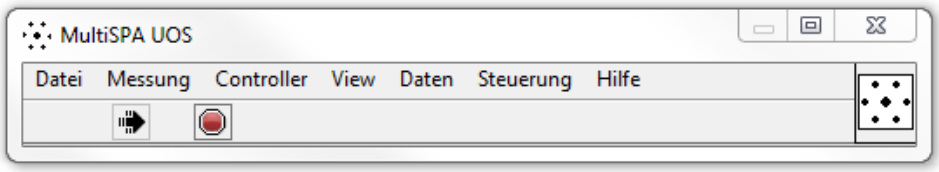


Abbildung 6.1: Frontpanel von *Main-Start.vi* mit dem Menü.

Nach dem Start der *MultiSPA.exe* wird zunächst *Main-Start.vi* ausgeführt. Diese VI initialisiert zuerst die beiden Queues (*Command-Pipe* und *Data-Pipe*), setzt die *Stop*-Variablen auf *FALSE* und legt den verwendeten Timeout fest. Im gleichen Zuge wird die Log-Datei geöffnet bzw. neu erstellt. Diese wird für jeden Tag neu angelegt, um die Übersichtlichkeit zu wahren. Die Log-Datei enthält alle gesendeten Befehle und deren Bestätigungen, um falsch ausgeführte Befehle nachverfolgen zu können.
Der Startzustand ist *connect*, dieser ruft *Main-Connect.vi* auf, welches die Verbindung herstellt. Sollte bei der Erstellung ein Fehler auftreten, wird der Zustand erneut aufgerufen. Bei erfolgreicher Herstellung der Verbindung fragt die Software mit dem Befehl *@?** die Geräteversion ab, um festzustellen, ob die Software mit der der Steuerung zusammenarbeiten kann. Nach erfolgreicher Prüfung wird der *init*-Zustand aufgerufen. Dieser liest die Daten aus der *specificData.spa* ein, die in Kapitel Tabelle 6.13 be-

schrieben ist. Ebenfalls werden alle Fenster-VIs gestartet, diese werden direkt in den Hintergrund geschoben und an die Positionen gesetzt, die in *Fensterpositionen.spa* eingetragen sind. Nur ein VI *Controller-Modus.vi* wird direkt angezeigt, mit dem der Benutzer die anderen Fenster aufrufen kann. Danach wird der Zustand *send* mit dem String *@clear** aufgerufen, welcher alle Parameter in der Oktopol-Steuerung in Mittelstellung bringt. Es folgt der Standardzustand *wait* der auf Eingaben der Queue wartet. Die Zustände *load* und *save* enthalten keine eigenen Aufgaben, sie reichen die Befehle des Menüs über die Queue weiter.

6.1.1. Zustand measurement

Der Zustand *measurement* enthält die Messmethoden für die 2D-, Line- und Optimierungs-Scans. Diese sind als Unterzustände realisiert worden, da alle unterschiedlich aufgebaut sind. Ausgewählt wird der entsprechende Zustand über den durch den Queue-Cluster mitgelieferten String. Für die einzelnen Schritte einer Messung wurde die gestapelte Sequenzstruktur von LabVIEW ausgewählt. Die einzelnen Strukturen laufen nacheinander ab und bieten mehr Platz im Blockdiagramm, da nur die aktuell ausgewählte Teilsequenz angezeigt wird. Die Messungen sind gleich aufgebaut, zuerst wird die Messfenstergröße (*@size X Y**) und im Falle der 1D-Scans die Anzahl der Messpunkte (*@scp P**) gesendet. Danach wird das Array für die Daten vorbereitet (während für 2D-Scans nur ein entsprechend großes 2dimensionales-Array angelegt wird, wird für die 1D-Scans ein 2xn Array angelegt, dabei wird eine Zeile mit den X-Werten schon gefüllt, die andere Zeile enthält dann die Y-Werte/Counts.). Außerdem werden über die Queue Befehle an den entsprechenden Graphen gesendet, damit diese die Achsen anpasst und die Messparameter für ein späteres Speichern aktualisiert. Gleichzeitig werden auch alle Eingabemöglichkeiten gesperrt, damit nicht schon die nächste Messung gestartet wird oder Parameter verändert werden, die dann die Queue blockieren. Im Folgenden wird der Befehl für die Messung erstellt und versendet. Die nächste Teilsequenz der Sequenzstruktur übernimmt das Auslesen und Auswerten der Daten, für Beide Schritte wurde ein Schleife angelegt. Der Datentransfer vom Auslesen zum Verarbeiten erfolgt über die *Data-Pipe*. In der Ausleseschleife wird zuerst die Verbindung abgefragt, ob und wie viele Bytes vorhanden sind und dann entsprechend ausgelesen. Mit der Abfrage der am Eingang liegenden Bytes werden unnütze Lesevorgänge verhindert, da diese einen Blue-Screen auslösen könnten. Das VI „VISA:Lesen“ liest dabei entweder die angegebene Anzahl an Bytes aus, oder sollte vor der angegebenen Anzahl ein Endzeichen \r\n kommen nur bis dahin. Ausgelesen wird

beim 1D-Scan nur bis zum ersten Endzeichen, da dieses das Ende der Messung markiert. Beim 2D-Scan wird bis zum n-ten Endzeichen gelesen, da hierbei das Endzeichen nur das Ende einer Zeile markiert, wobei n für die Anzahl der Messreihen steht. Die ausgelesenen Daten werden direkt in die Queue geschrieben. Die Verarbeitungsschleife liest solange Daten aus der Queue bis diese ein Endzeichen enthalten. Die Daten liegen als viele lange Einzelstrings vor, welche bei den Kommas geteilt und in Integer umgewandelt werden. Diese werden in der richtigen Reihenfolge in das Array geschrieben. Nach jedem verarbeitetem Einzelstring bekommt der 1D-Graph den Befehl sich zu aktualisieren. Eine Aktualisierung nach jedem Wert erzeugt einen sehr großen Overhead, denn die benötigte Zeit zum Aktualisieren des Graphen ist größer, als die Zeit, die zum Auslesen und Auswerten eines Wertes benötigt wird. Für Messzeiten größer als 0.1μs ist das Auslesen und Auswerten pro Messwert nicht langsamer als die Messzeit. Bei 2D-Scans wird der Graph nach jeder fertigen Zeile aktualisiert. Nach der Verarbeitung wird die GUI wieder entsperrt, die Data-Pipe geleert und die Rückgabe überprüft, ob die Messung auch erfolgreich durchgeführt wurde. Der Automat wechselt dann wieder in den *wait*-Zustand.

6.1.2. Menü

Main-Start.vi besitzt ein Menü, dessen Eingaben mithilfe einer Eventstruktur in einer eigenen Schleife neben dem Zustandsautomaten ausgewertet werden. Über die meisten Menüpunkte lassen sich die einzelnen Fenster in den Vordergrund holen, diese rufen jeweils den *front*-Zustand des jeweiligen VIs über die Queue auf. Als Standardeinstellung ist bei allen VIs die Einstellung „Verbergen, wenn LabVIEW nicht aktiv ist“ aktiviert. Dies bedeutet, wird ein anderes Programm als MultiSPA aktiviert, so werden alle Fenster direkt minimiert und erscheinen wieder, wenn MultiSPA wieder das aktive Programm ist. Diese Einstellung kann mithilfe des Menüpunktes „Datei→Minimieren bei Inaktivität“ aus- und eingeschalten werden, die aktuelle Einstellung wird mit einem Haken angezeigt. Der Menüpunkt „Hilfe→Reset Queue“ leert die *Command-Pipe*, falls es mal zu einer Blockade kommen sollte. Während „Hilfe→Fenster in rechte obere Ecke verschieben“ hilft, wenn ein Fenster mal außerhalb des Bildschirms geraten ist. “Hilfe→Update Graphen“ kann außerhalb der üblichen Aktualisierung der Graph erneuert werden, beispielsweise nach einem Queue-Reset. Mit „Steuerung→Offset und Energie neu setzen“ ist es möglich nach einem Neustart der Steuerung weiterzumachen, ohne das MultiSPA neuzustarten. Hierbei werden der Offset und die Elektronenenergie auf die alten Werte gesetzt. „Steuerung→Clear“ wird ein *@clear** ausgeführt. Wichtige

Menüpunkte wurden mit Tastenkombinationen versehen, um das Aufrufen zu erleichtern. Dies geschieht direkt in der Erstellung des Menüs. Aufgrund der Art des Aufbaus können diese nur benutzt werden, wenn *Main-Start.vi* das aktive Fenster ist.

6.2. Verbindungsfenster *Main-Connect.vi*

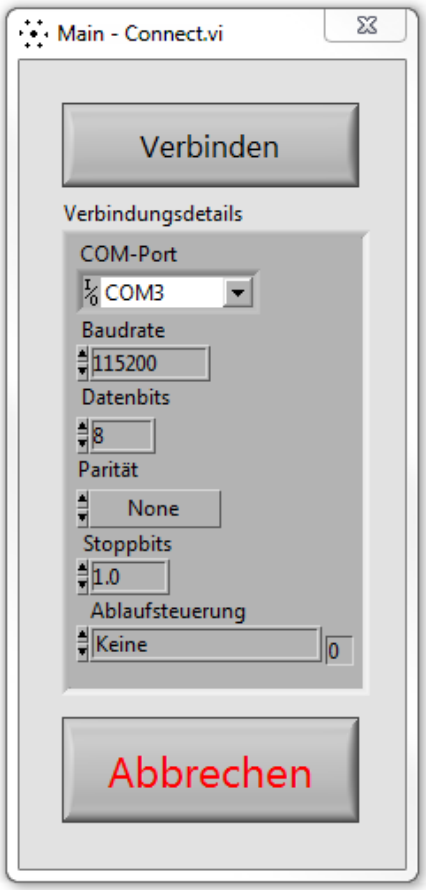


Abbildung 6.2: Frontpanel von *Main-Connect.vi*.

Main-Connect.vi wird direkt nach dem Start vom *connect*-Zustand von *Main-Start.vi* ausgeführt. Es enthält nur zwei Schaltflächen und das Clusterelement mit den Verbindungsdetails, die in Kapitel 4.4 vorgestellt wurden. Die Schaltfläche „Verbindung“ startet einen Verbindungsversuch mit den angegebenen Werten. Bei erfolgreichem Aufbau werden die Buffer geleert und die Verbindung wieder geschlossen. Die Referenz auf die Verbindung wird über die globale Variable *VISA-Ressourcenname* weitergegeben, sie enthält alle Parameter um die Verbindung aufzubauen. Kommt keine Verbindung

zustande, so wird der Benutzer durch ein Dialogfenster informiert und kann danach versuchen mit anderen Verbindungsdetails die Verbindung herzustellen. Durch Betätigen der *Abbrechen*-Schaltfläche oder des *X* des Fensters, wird das Programm beendet.

6.3. Modus *Controller-Modus.vi*

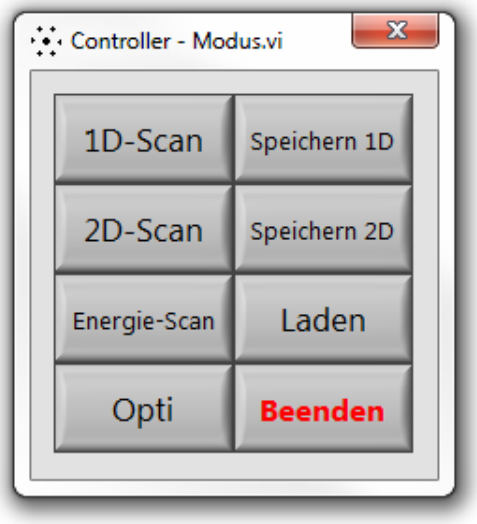


Abbildung 6.3: Frontpanel von *Controller-Modus.vi*.

Abbildung 6.3 zeigt das erste Fenster nach einer erfolgreichen Verbindung. Hier wählt der Benutzer die Funktion aus. Nach Betätigung einer Schaltflächen werden die benötigten Fenster in den Vordergrund geholt. Bei Klick auf *Speichern*[18] erscheint ein Dateidialog, in dem der Speicherort für die Messung festlegt wird. Es wurden zwei Schaltflächen dafür angelegt, da nicht automatisch geprüft werden kann, welches Graphenfenster sich aktuell im Vordergrund befindet. *Laden*[19] erzeugt ebenfalls ein Dateidialog, mit diesem können alte Messungen angeschaut werden. Es wird dazu die jeweilige Methode bei den Graphen-VIs aufgerufen und der Pfad der Messung mit dem *Command-Cluster* übergeben. Wird eine Messmethode (1D, 2D oder Optimierungsmodus) aufgerufen, so werden neben den Startfenstern auch der Graph und das Fenster für die Messparameter sichtbar.

[18] .spa1d und .spa2d
[19] erlaubte Dateitypen .spa1d; .spa2d; .d1d und .d2d

6.4. Messparameter *Controller-Einstellungen.vi*

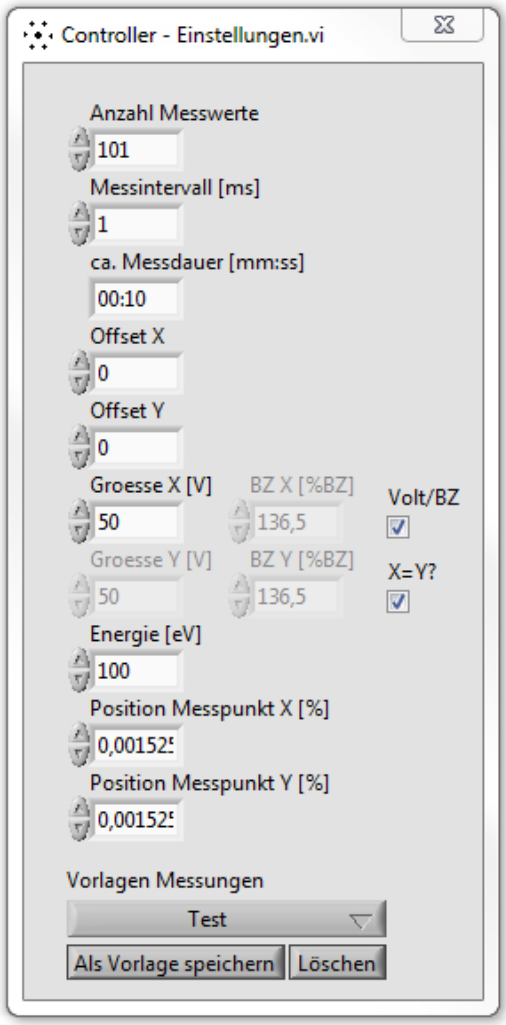


Abbildung 6.4: Frontpanel von *Controller-Einstellungen.vi*.

Das Fenster beinhaltet alle Einstellungen für die Messparameter. Einstellbar sind die Anzahl der Messwerte, Messintervall (Gatetime), Messfenstergröße, Messfensterverschiebung , Zeile für die X/Y-Achsen Messung und die Elektronenenergie. Damit falsche Eingaben nicht möglich sind, werden im *init*-Zustand alle Eingabefelder mit entsprechenden Grenzen versehen. Da für bestimmte Angaben nur diskrete Werte erlaubt sind (wegen den DACs) wird auch die Schrittweite (Inkrement) entsprechend angepasst. Die Felder sind so angepasst worden, dass sie alle Eingaben erlauben, den eingegebenen Wert aber auf den nächsten erlaubten Wert anpassen. Die Werte werden nach der Eingabe für die Befehlsstruktur umgerechnet. In Abbildung 6.5 ist ein

Beispiel einer Umrechnung angegeben.

$$Dezimalwert \cdot \frac{Maximum(Integerwert)}{Maximum(Dezimalwert)} = Integerwert \tag{6.1}$$

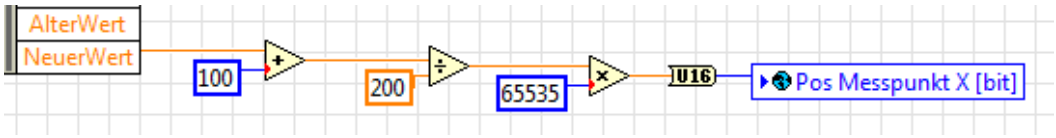


Abbildung 6.5: Umrechnung am Beispiel von Mittelpunkt X. Das Feld erlaubt Werte von -100% bis +100% und rechnet diese um in Werte von 0 bis 65535. Aus *Controller-Einstellungen.vi*.

Für die Anzahl der Messwerte sind theoretisch ganzzahlige Messwerte von 0-65535 erlaubt. Hier begrenzen die unterste Grenze für Linescans (10) und der Speicher als oberste Grenze für 2D-Scans den Wertebereich. Die oberste Grenze ist variabel einstellbar über die Datei *specificData.spa* die in Kapitel 6.13 beschrieben ist. Die Grenze hängt im Wesentlichen von der Menge des eingebauten Arbeitsspeichers ab. Wie schon erläutert sollte immer eine ungerade Anzahl an Messwerten verwendet werden.

Für das Messintervall wurde die Eingabe in ms gewählt, da Zeiten außerhalb dieser Maßeinheit nur selten gewählt werden. Einstellbar sind 0,1ms bis 1000ms in 0,1ms Schritten. Werte die außerhalb dieses Bereiches liegen, sind für eine Messung nicht praxistauglich. Unterhalb ist das Intervall zu kurz für eine vernünftige Messung und oberhalb wird die gesamte Messdauer zu groß. Die angezeigte Messdauer wird aus der Anzahl und dem Intervall für eine 2D-Messung berechnet, diese dient nur der Übersicht. Sollte die erwartete Messzeit über 10min legen, so färbt sich das Feld Gelb und der Benutzer bekommt vor einer Messung eine Warnung.

Die beiden Felder für Offset X und Y sind für die Verschiebung des Messfensters zuständig. Die Grenzen liegen bei -37,5V und +37,5V.

Über die Felder Größe und BZ (Brillouin-Zone) wird die Messfenstergröße eingestellt. Mit der Checkbox *Volt/BZ* kann ausgewählt werden, mit welchen Feldern (Einheit) die Größe festgelegt wird, dies wird auch für die Achsenbeschriftung übernommen. Ebenfalls ist es möglich mit der anderen Checkbox *X=Y?* auszuwählen, ob der Wert für X auch für Y übernommen werden soll, oder ob unterschiedliche Werte angegeben werden sollen. Die jeweils nicht benötigten Felder werden ausgegraut, die angezeigten Werte jedoch weiter angepasst. Die Umrechnung zwischen der Einstellung in Volt oder

BZ ist von der eingestellten Elektronenenergie und dem eingegebenem Skalierungsfaktor (Sensitivität) abhängig:

$$Volt = \frac{\sqrt{E}}{\text{Sensitivität}} \cdot \%BZ \qquad \text{und} \qquad \%BZ = \frac{\text{Sensitivität}}{\sqrt{E}} \cdot Volt \tag{6.2}$$

Mit der Dropdownbox unten im Fenster können verschiedene Vorlagen ausgewählt werden. Ebenfalls ist es möglich neue Vorlagen anzulegen. Dazu werden einfach die Felder mit den Werten gefüllt und mit *Als Vorlage speichern* gespeichert. Im Folgendem Dialogfenster muss noch ein Name eintragen werden und die Werte werden in der Messvorlagen.spa gespeichert. Mit *Löschen* kann die aktuell ausgewählte Messvorlage gelöscht werden.

6.5. Energiescan *Controller-Energiescan.vi*

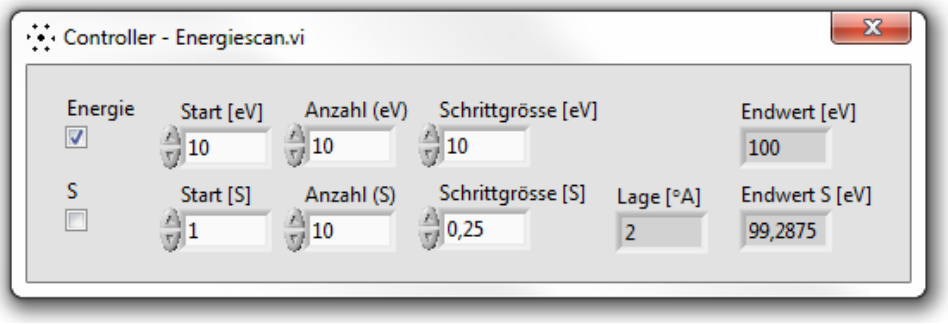


Abbildung 6.6: Frontpanel von *Controller-Energiescan.vi*.

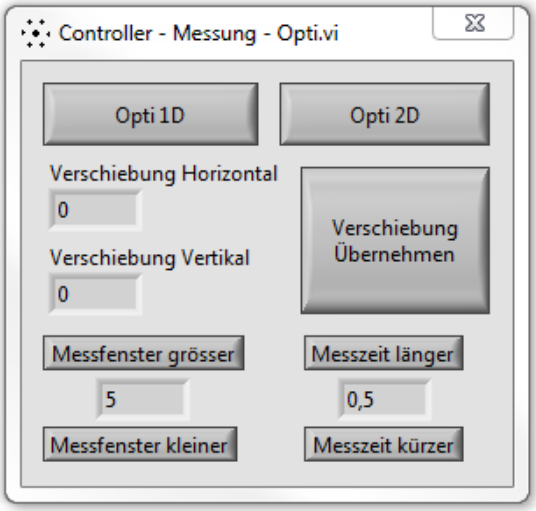
Das Fenster *Energiescan* bietet die Einstellungen, um mehrere Messungen hintereinander automatisiert durchzuführen. Die Einstellung kann entweder in Elektronenenergie oder in Streuphasen erfolgen. Angegeben werden muss der Startwert, die Anzahl der Schritte und die Schrittweite. Für die Streuphase werden noch der Lagenabstand und der Winkel α (vgl. Abbildung 3.2) für die Umrechnung in die Energie (vgl. Gleichung 2.6) benötigt. Diese werden über ein weiteres Fenster (Kapitel 6.11) oder über die Datei *specificData.spa* eingestellt. Die beiden Felder *Endwerte* zeigen die Energie der letzten Messung der Serie an. Steigt der Wert über das Maximum der erlaubten Energie (500eV), so wird ein bereits gesetzter Haken in der zugehörigen Checkbox wieder entfernt. Es kann jeweils nur eine Checkbox aktiviert werden, die jeweils andere wird bei einer Aktivierung zurückgesetzt. Wird bei der Schrittweite ein 0 eingetragen,

so werden mehrere Messungen mit derselben Energie durchgeführt. Die Werte werden jeweils direkt in globalen Variablen gespeichert, um sie einerseits für die Messung zur Verfügung zu stellen und andererseits für die Speicherung von Messvorlagen.

6.6. Messungen *Controller-Messung-1D/2D/Opti.vi*

(a) *Controller-Messung-1D.vi*

(b) *Controller-Messung-2D.vi*



(c) *Controller-Messung-Opti.vi*

Abbildung 6.7: Frontpanel der Fenster um die Messungen zu starten.

Mit den Schaltflächen in den Messungen-VIs lassen sich die einzelnen Messungen starten. Die einzelnen Methoden sind jeweils auf einzelne Fenster aufgeteilt, um ein zu großes Fenster mit allen Methoden zu vermeiden. Der 1D-Modus bietet drei verschiedene Modi, den normalen 1D-Scan, den Energiescan und den Conti-scan. Mit

dem 1D-Scan wird nur ein Scan durchgeführt, während beim Conti-scan der Scan immer wieder durchgeführt wird und die Counts der einzelnen Messpunkte addiert werden. Der Conti wird solange durchgeführt, bis die Schaltfläche *Abbrechen* betätigt wird. Bei Energiescan werden die Einstellungen aus dem VI *Controller-Energiescan.vi* verwendet und die Scans nacheinander durchgeführt. Für den Energiescan wird zuerst geprüft, ob und welcher Haken gesetzt wurde, im Anschluss wird der Benutzer nach einem Speicherplatz für die Messungen gefragt. Verlaufen beide Prüfungen erfolgreich, wird ein Array mit den Energien erstellt, mit denen eine Messung durchgeführt werden soll. Eine Messung besteht im Wesentlichen aus drei Schritten, Energie setzen, Messung durchführen und Messung speichern. Das Ende einer Messung wird mit der globalen Variablen *Energiescan* geprüft, diese wird zu Anfang einer Messung im Messzustand in *Main-Start.vi* auf *TRUE* gesetzt und am Ende auf *FALSE*. Die Methode, die die entsprechenden Schritte einleitet, befindet sich im Zustand *data→1D-Energiescan*. Für den 1D-Scan lassen sich mit der Checkbox *H/V* bzw. mit dem Feld *Winkel*, die Lage des 1D-Scans im Messfenster festlegen. Ein Scan mit einem Winkel führt immer durch den Mittelpunkt des Messfensters. Wird ein Horizontaler/Vertikaler Scan ausgeführt (Winkel auf 0, dann wird die Checkbox ausgewertet), kann der Linescan mit Position *Messpunkt X/Y* im Fenster verschoben werden. Mit der Schaltfläche *Gauß* wird *View-Graphen-Liniendiagramm.vi* gestartet, mit den aktuellen Messwerten und einer automatisch angefitteten Gauß-Funktion. Dazu wird die Halbwertsbreite, die aus der Standartabweichung σ berechnet wird, angezeigt. Die Formel lautet $Halbwertsbreite = 2 \cdot \sqrt{2 \cdot ln2} \cdot \sigma$. Für den 2D-Scan sind die gleichen Schaltflächen vorhanden, um eine Messung zu starten. Es fehlen die Checkbox und das Feld für den Winkel, da der 2D-Scan nicht gedreht durchgeführt werden kann[20]. Über die Schaltfläche Auswerten wird das VI *View-Auswertung.vi* gestartet, das in Kapitel 6.9 genauer erläutert wird. Das Opti-Messfenster bietet zwei Schaltflächen für die Messungen und weitere für die Einstellungen, welche über die unteren Schaltflächen eingestellt werden können, da die Messungen nicht mit den Messparameter aus *Controller-Einstellungen.vi* ausgeführt werden. Der Opti-1D führt zwei Messungen aus, einmal Horizontal und einmal Vertikal. Die beiden Messungen werden mit je 101 Messwerten durchgeführt und solange wiederholt, bis diese Schaltfläche erneut betätigt wird. Der 2D-Scan wird nur einmal durchgeführt mit 51x51 Messwerten. In den Feldern *Verschiebung* wird jeweils die Abweichung des Maximums der letzten Messung

[20] Dies wird durch den Speicher, bzw. Rechenleistung des Mikrocontroller begrenzt. Eine Vorausberechnung der Messpunkte benötigt mehr Speicher und eine Berechnung pro Punkt benötigt mehr Rechenleistung um eine akzeptable Messzeit zu erreichen.

von dem Nullpunkt angezeigt. Diese berechnet sich aus dem Abstand des Gitterpunktes zum Nullpunkt multipliziert mit der Messfenstergröße geteilt durch 32768. Mit der Schaltfläche *Verschiebung übernehmen* wird diese Verschiebung direkt als Offset übernommen. Der Opti-Modus wird benutzt, um den Hauptreflex in den Mittelpunkt des Messfensters zu verschieben. Zuerst sollte versucht werden die Probe mit dem Manipulator entsprechend zu positionieren. Der Offset wirkt durch die Ablenkeinheit direkt auf den Elektronenstrahl und führt für verschiedene Elektronenenergien zu anderen Verschiebungen.

6.7. Darstellung *View-Graphen-1D/2D/Opti1D/Opti2D.vi*

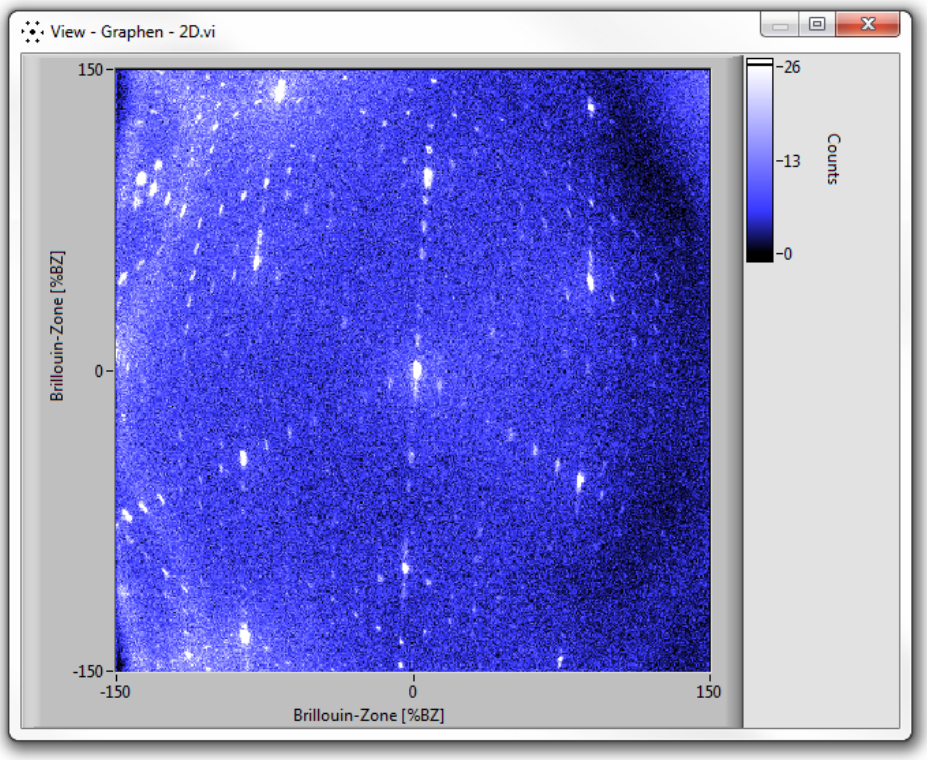


Abbildung 6.8: Frontpanel von *View-Graphen-2D.vi*. Messung zeigt das Beugungsbild (7x7) einer Si(111)-Probe.

Für die Darstellung der Messwerte gibt es vier verschiedene Fenster. Die jeweiligen Frontpanel bestehen nur aus einem oder zwei Graphen. Sie sind die einzigen VIs bei denen die Fenstergröße variabel angepasst werden kann, es wurde nur eine Minimalgröße vorgegeben. Alle Graphen sind ebenfalls als Zustandsautomat aufgebaut, die

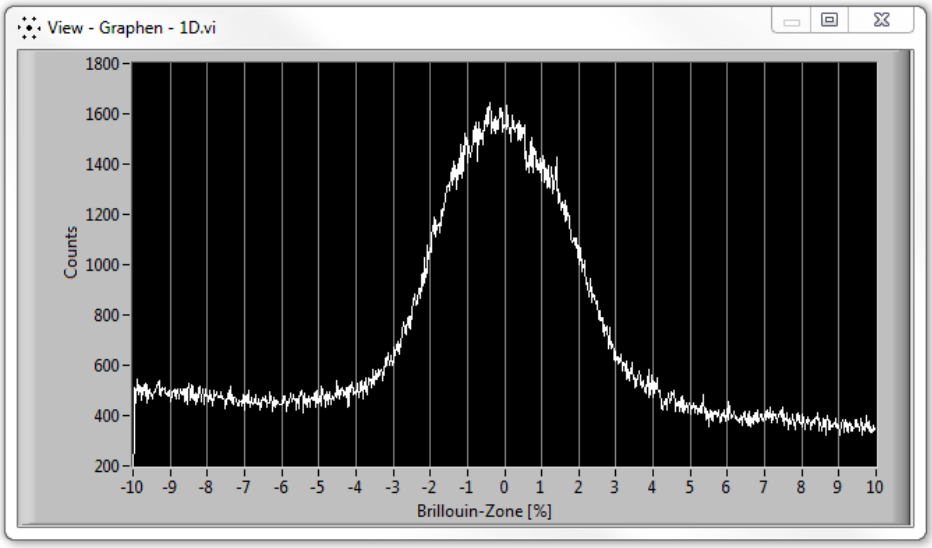


Abbildung 6.9: Frontpanel von *View-Graphen-1D.vi*. Messung horizontal durch den (00)-Reflex von Si(111).

beiden Hauptzustände sind init für die Ersterstellung des Graphen und display in dem die Werte aktualisiert werden. Der Zustand data enthält Unterzustände, um bestimmte Einstellungen der Graphen zu ändern (bspw. Achsen-beschriftung und -skalierung, speichern und laden von Daten.) Die beiden Optimierungsgraphen haben außer der Darstellung der Messwerte keine weitere Funktion. Die roten Linien zeigen die Position des Nullpunktes an. Der obere Graph der beiden 1D Diagramme zeigt die Messwerte der horizontalen Messreihe und der Untere die der vertikalen Messung. Die Y- bzw. Z-Achse skaliert automatisch linear mit den Messwerten. Der 1D und 2D-Graph besitzen neben den Messdaten auch Variablen für die Parameter der Messung. Diese werden nach der Sperrung der Frontpanel gespeichert, damit stehen die zugehörigen Messparameter bis zur nächsten Messung zum Speichern zur Verfügung. Wird eine Messung geladen, so werden die entsprechenden Parameter mit den Werten aus der eingelesenen Datei gefüllt. Wird ein alter Dateityp (.d1d oder .d2d) eingelesen, so werden die fehlenden Werte (siehe Kapitel 6.13) mit 0 gefüllt. Die Methoden um die Achsenbeschriftung bzw. die Skalierung der Achsen zu ändern befinden sich als Unterzustände im data-Zustand. Sie beschränken sich darauf bestimmte lokale Variablen mithilfe eines Eigenschaftsknoten für den Graphen zu übernehmen. Für die Auswertung von 2D-Messungen sind zwei Events zuständig, diese werten die Mausbewegung und die Maustasten aus. Für die Positionsbestimmung der Maus werden die Bildschirmkoordinaten mit dem Methodenknoten *Koordinaten in XY umrechnen* des 2D-Graphen in

die Koordinaten des Arrays umgerechnet und geprüft ob die Position innerhalb des Graphen liegt. Für die Mausbewegung wird dann mit dem Faktor und dem Offset der Achsen die aktuelle Position im Array berechnet und die 3 Koordinaten für das VI *View-Auswertung.vi* gespeichert. Bei einem Klick wird ebenfalls die Position berechnet und ein Cursor an die Stelle gesetzt. Wurden bei Maustasten betätigt, wird eine Linie zwischen den beiden Punkte gezeichnet. Danach werden die Messpunkte die auf der Linie liegen mit dem Bresenham-Algorithmus bestimmt. Zu dieser Kurve wird eine Gaußkurve angefittet mit dem VI *Gaußsche Spitzenwertanpassung* und die Halbwertsbreite aus der Standartabweichung berechnet. Neben diesen Berechnungen werden noch der Abstand der Punkte und der Winkel den beide mit der Vertikalen einschließen in globale Variablen gespeichert, und eine Updatebenachrichtigung an *View-Auswertung.vi* verschickt, die die Daten darstellt.

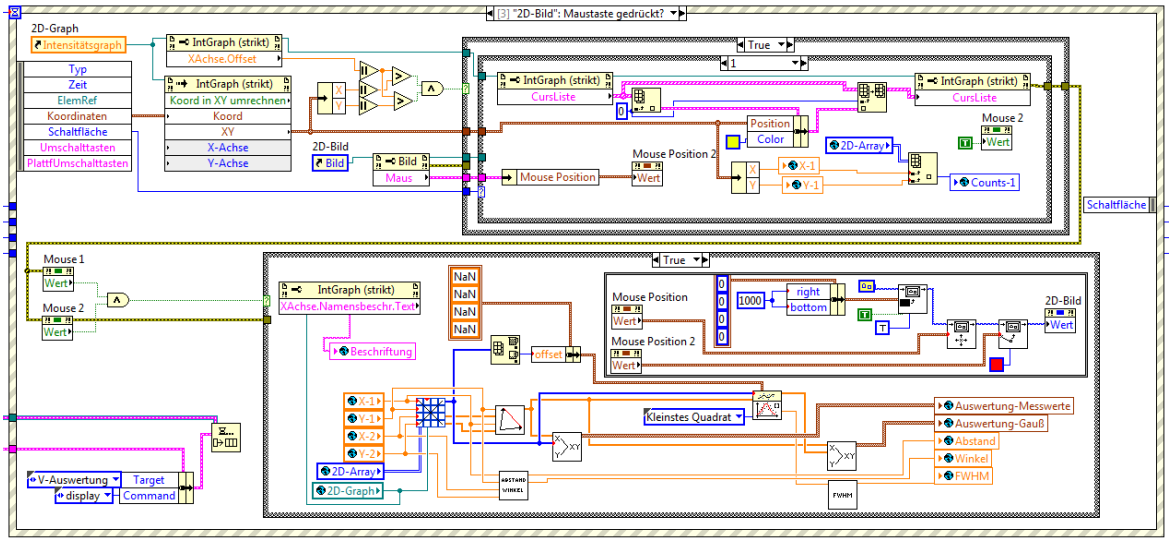


Abbildung 6.10: Blockdiagramm aus *View-Graphen-2D*, dargestellt ist im oberen Teil das setzen der Cursor und im unteren Teil die Berechnung der Linie.

Das Anzeigen der Linie geschieht nicht über den Intensitätsgraphen, da dieser keine Methode dafür enthält. Über den Graphen wurde ein Bild gelegt, das zumeist Transparent ist, und nur die Linie enthält.

Über das Menü von *Main-Start.vi* können die Werte der Graphen auch als .png (für 2D) bzw. .eps (für 1D) exportiert werden. Die jeweiligen Methoden befinden sich in den Graphen-VIs. Mit dem Methodenknoten des Intensitätsgraphen lässt sich direkt ein Bild erstellen, welches auch die Achsen beinhaltet und nicht pixelgenau ist, da beim

Graphen je nach Anzeigegröße interpoliert wird. Bei 2D-Messungen wird nur das Bild der Messwerte ohne Interpolation benötigt. Dazu wird das Array mit den Messwerten zuerst an die Grenzen des Counts angepasst, danach werden die Zeilen umgedreht[21] und von allen Werten der Minimale-Count abgezogen. Im Anschluss werden die Werte noch mit 255/(Max-Count - Min-Count) multipliziert. Somit wird ein Array erzeugt, mit Werten zwischen 0 und 255, auf das die Farbtabelle im VI *2D-Pixmap nach 1D* angewendet werden kann, danach wird das Bild gespeichert.
Der 1D-Graph kann zur weiteren Verwendung bspw. in Corel-Draw als .eps gespeichert werden. Für den im 1D-Graph verwendeten XY-Graphen ist dies mit dem Methodenknoten *Bild exportieren* möglich, hierbei tritt ein Bug von LabVIEW auf, der umgangen werden muss. LabVIEW schreibt `polyline32` statt `polyline`. Dazu wird die erstellte .eps-Datei im Anschluss noch einmal geöffnet und die entsprechenden Stellen ersetzt. An diesem Beispiel lässt sich auch die Benutzung des Fehlerdrahtes als Angabe der Reihenfolge erkennen. So wird *Datei öffnen* erst ausgeführt, wenn *Bild exportieren* beendet wurde, ohne den Fehlerdraht würden die Methoden zum Ersetzen gleichzeitig mit der Erstellung in der Datei arbeiten, was zu Fehlern führt. Sollte das 1D-Fenster zu groß sein, so wird nicht alles in der .eps-Datei gespeichert, sondern abgeschnitten. Um dies zu umgehen, wird der Graph für die Erstellung kurzzeitig verkleinert.

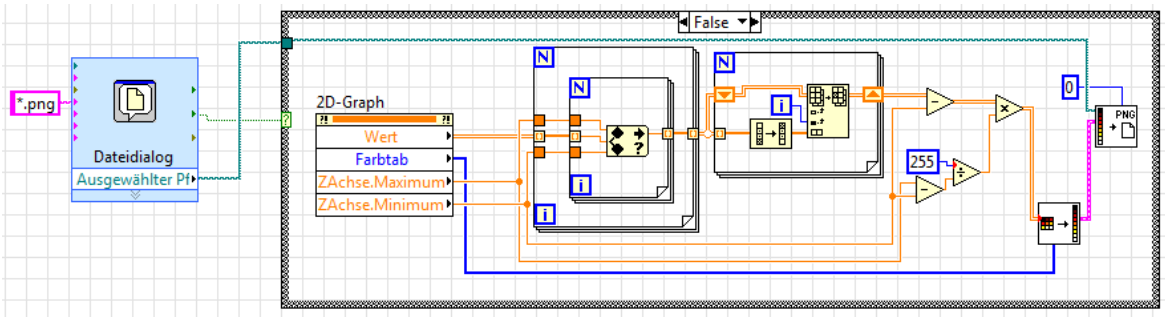


Abbildung 6.11: Blockdiagramm aus *View-Graphen-2D.vi* zeigt die Methode zum Exportieren einer 2D-Messung als png-Datei.

6.8. Counts *View-Counts.vi*

Mit diesem Fenster lässt sich die Achse anpassen die die Counts anzeigt (Y-Achse für 1D-Scans und Z-Achse für 2D-Scans). Die Änderungen werden in eine globale Variablen

[21] Dies kompensiert das Umdrehen des Arrays in *2D-Pixmap nach 1D.vi*.

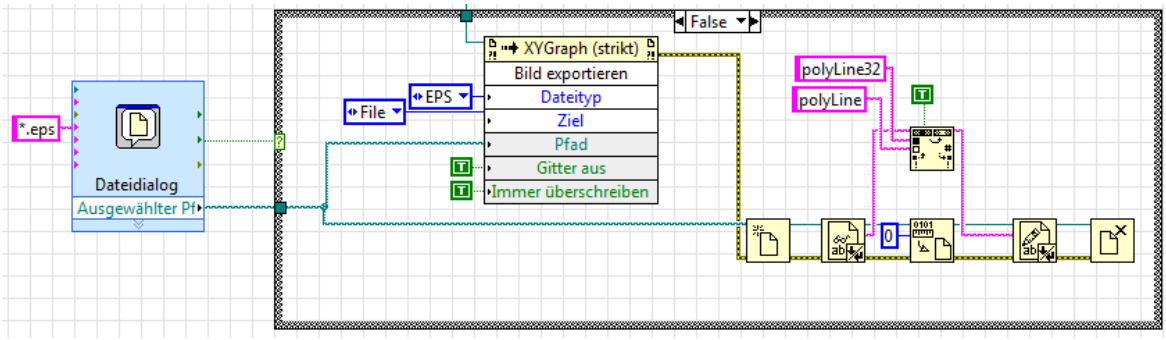


Abbildung 6.12: Blockdiagramm aus *View-Graphen-1D.vi* zeigt die Methode zum Exportieren einer 1D-Messung als eps-Datei.

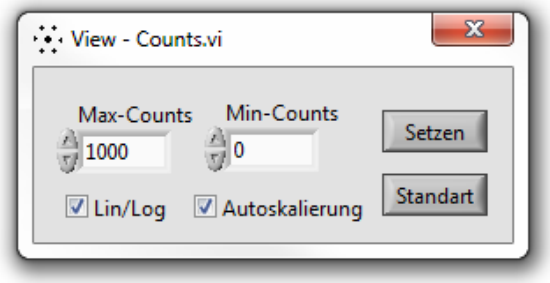


Abbildung 6.13: Frontpanel von *View-Counts.vi*.

geschrieben und eine Benachrichtigung zum Update an die Graphen geschickt. Ist die Checkbox Autoskalierung aktiviert, wird die Achse automatisch an das aktuelle Maximum und Minimum angepasst. Bei Deaktivierung werden die in Max-Counts und Min-Counts eingetragenen Werte für die Count-Achse verwendet. Bei Änderungen in Max-Counts oder Min-Counts werden diese direkt für die Graphen übernommen. Ist *Autoskalierung* aktiviert, so können die Werte bis zur nächsten Graphenaktualisierung mit der Schaltfläche *Setzen* übernommen werden. Die Checkbox *1D-Lin/Log* ändert die Achsenskalierung für den 1D-Graph zwischen linear und logarithmisch. Mit dem Button *Standard* werden die Graphen wieder auf die Ausgangswerte zurückgesetzt.

6.9. Auswertung *View-Auswertung.vi*

Das Auswertungsfenster ist eine Erweiterung zum 2D-Graphen, neben den Koordinaten für die aktuelle Mauszeigerposition werden die Koordinaten der beiden Zeiger angezeigt, die im 2D-Graph gesetzt werden können. In weiteren Feldern wird der Abstand

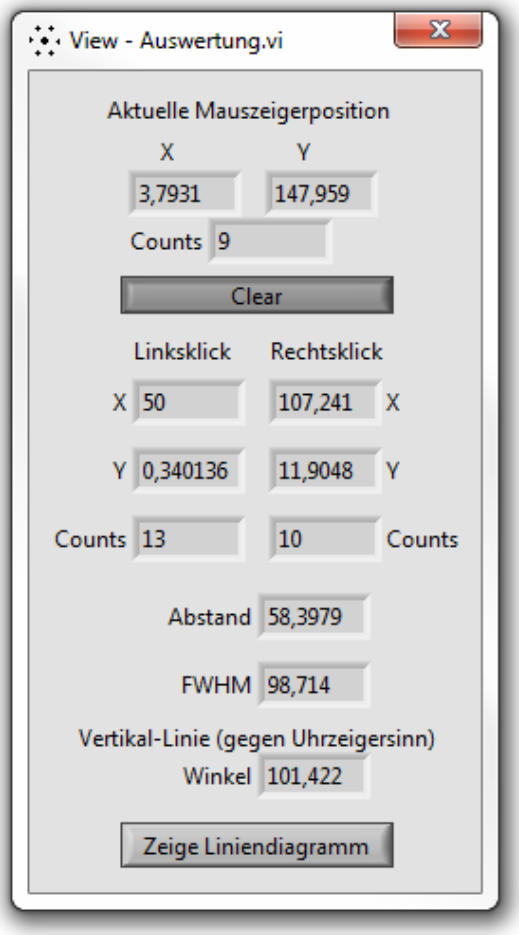


Abbildung 6.14: Frontpanel von *View-Auswertung.vi*.

der Punkte in der aktuell verwendeten Einheit [V,%BZ] angezeigt. Das Feld *Winkel* zeigt den von der Geraden durch die beiden Punkte und der Vertikalen eingeschlossenen Winkel an. Der Button *Zeige Liniendiagramm* zeigt die Werte an, die auf der Linie zwischen den beiden Punkten liegen und die automatisch angefittete Gauß-Funktion. In *FWHM* wird die aus der Gauß-Funktion berechnetet Halbwertsbreite angezeigt. Die Berechnungen finden wie in Kapitel 6.7 beschrieben statt. Die Übertragung der Daten erfolgt per globalen Variablen und einer Updatebenachrichtigung.

6.10. Farbverlauf *Controller-Color.vi*

Das in Abbildung 6.15 gezeigte Fenster ist für die Einstellung des Farbverlaufs im 2D-Graph verantwortlich. Der Farbverlauf wird mit den drei Grundfarben Rot, Grün

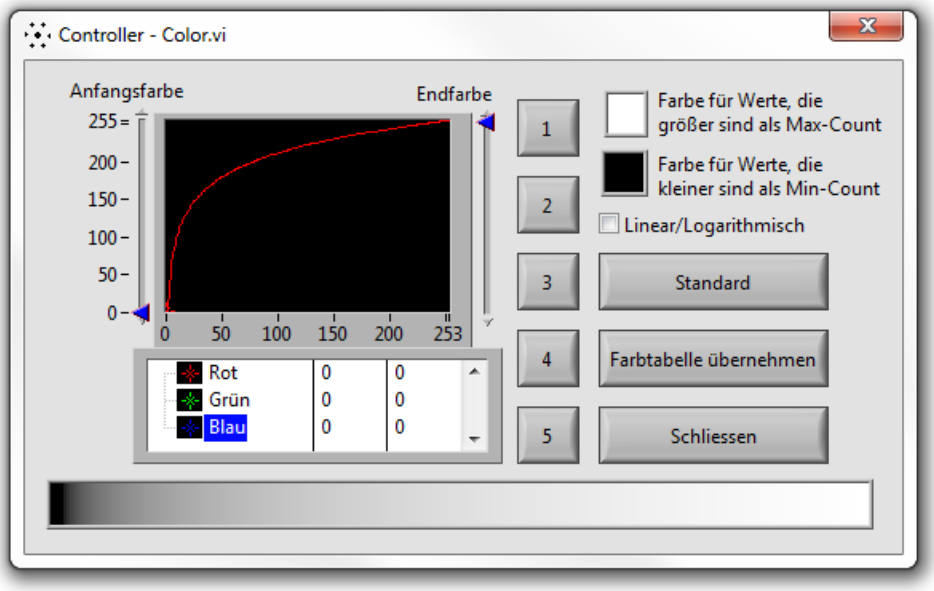
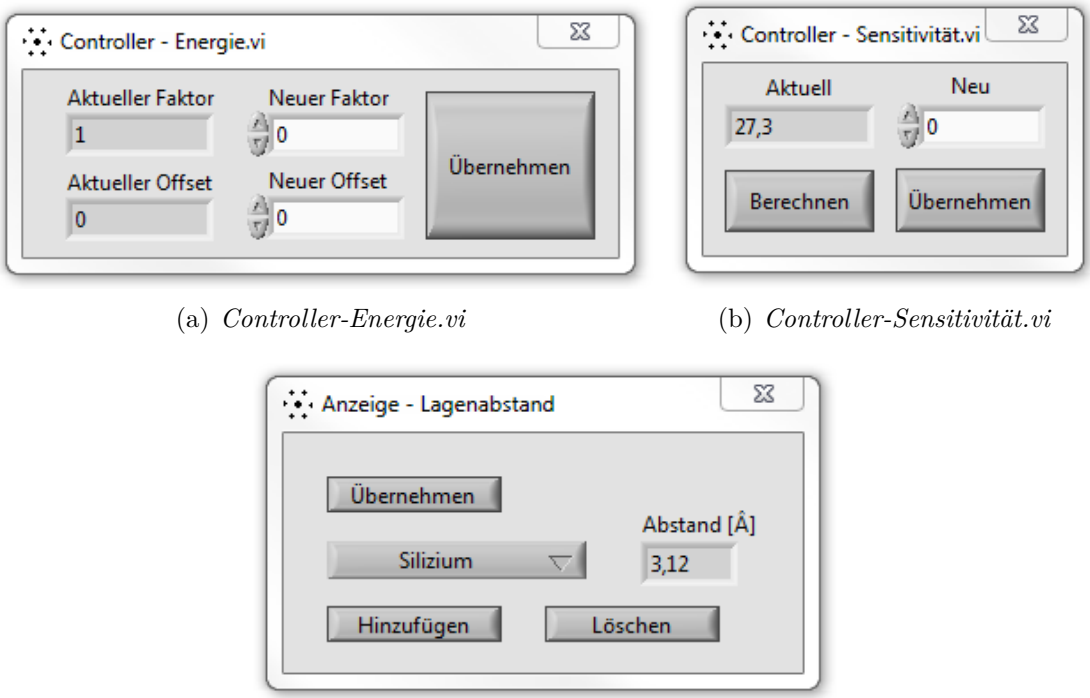


Abbildung 6.15: Frontpanel *Controller-Color.vi*.

und Blau eingestellt, für jede Farbe müssen jeweils drei Werte festgelegt werden. Der Start- und Endwert und ein Zwischenwert. Der Start- bzw. Endwert wird über den jeweiligen Regler neben dem Graphen eingestellt. Der Zwischenwert kann mit der Maus im Graphen angepasst werden, durch Anklicken des jeweiligen Zeiger im Graphen und verschieben mit der Maus oder per Eingabe im darunterliegendem Feld. Der Farbverlauf wird in Echtzeit mit angepasst. Mit der Checkbox *Linear/Logarithmisch* lässt sich die Art des Verlaufs ändern. Die Änderung wird ebenfalls direkt im Graph und dem Farbverlauf angezeigt. Die beiden Farbfelder oben rechts ändern die Farben, die angezeigt werden, wenn ein Wert über/unterhalb der aktuell eingestellten Grenzen für die z-Achse liegt. Der Standardverlauf (Schwarz-Blau-Weiß) lässt sich mit der Schaltfläche *Standard* wiederherstellen. Weitere oft benutzte Farbverläufe lassen sich über die Schaltflächen *1-5* aufrufen (Bunt, Weiß-Blau-Schwarz, Schwarz-Weiß-Schwarz, Schwarz-Weiß, Weiß-Schwarz). Betätigten der Schaltfläche *Farbtabelle übernehmen* gibt den aktuell ausgewählten Farbverlauf zur Verwendung an den 2D-Graphen weiter. Aufgerufen werden kann dieses Fenster nur über das Menü „View→Farbverlauf 2D-Graph“.



(a) *Controller-Energie.vi*

(b) *Controller-Sensitivität.vi*

(c) *Controller-Lagenabstand.vi*

Abbildung 6.16: Frontpanel der Einstellungsfenster.

6.11. weitere Einstellungsfenster

Diese VIs besitzen nur die Funktionen auslesen und setzen und sind nicht wie die anderen Fenster als Zustandsautomat aufgebaut. Sie sind im Wesentlichen nur aus einer While-Schleife mit einer Event-Struktur aufgebaut. Mit diesen Fenstern lassen sich aktuell verwendete Geräte- und Materialkonstanten anzeigen, ändern und speichern (für spätere Verwendung, siehe Kapitel 6.13). Das Fenster *Lagenabstand* kann für verschiedene Elemente die Lagenabstände speichern, aufgerufen und für die Berechnung der Energie aus der Streuphase übernommen werden.

Controller-Energie ist dazu da, um den Faktor und den Offset der Energie zu verändern. Die Steuerung gibt eine Spannung zwischen 0-10V aus. Die Anpassung der Elektronenenergie erfolgt in der Steuerung für die Elektronenkanone. Der Faktor bezeichnet hierbei die Umrechnung von 1Volt in eV. Der Offset wird in eV angegeben.

Die Sensitivität wird über das VI *Controller-Sensitivität.vi* eingestellt. Es ist möglich diese manuell einzugeben oder aber die Sensitivität für die aktuellen Parameter berechnen zu lassen. Dafür muss zuerst die Messfenstergröße für eine beliebige Energie

so eingestellt werden, dass der gesamte Bereich 200%BZ entspricht. Dann wird nach dem Betätigen der Berechnen-Schaltfläche die passende Sensitivität berechnet.

Mit der Schaltfläche *Übernehmen* wird bei allen VIs der Wert in das Programm übernommen und eventuelle Methoden benachrichtigt.

6.12. SubVIs

Befehl senden(SubVI).vi

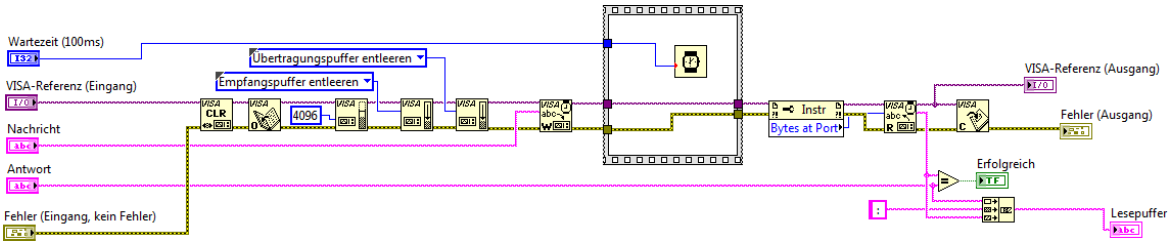


Abbildung 6.17: Blockdiagramm von *Befehl Senden(SubVI).vi*.

Abbildung 6.17 zeigt das Blockdiagramm zum Senden von Befehlen. Das VI benötigt als Angabe die VISA-Referenz, den zu sendenden Befehl, die erwartete Antwort und eine Angabe der Zeit, nach welcher eine Rückantwort erwartet wird. Die COM-Port-Session wird für jeden neuen Befehl einzeln geöffnet und die Puffer werden geleert. Durchgeführt wird dies wegen dem VCP-Treiber *usbser.sys* von Windows, dieser neigt bei vielen nicht erfolgreichen Lesezugriffen dazu einen Bluescreen zu verursachen. Danach wird der Befehl gesendet. Nach der Wartezeit[22] wird die Anzahl der aktuell im Puffer vorhandenen Bytes abgefragt. Diese Anzahl wird vom Puffer gelesen und mit der erwarteten Antwort verglichen. Die Rückgabe des VIs besteht aus der Referenz auf den Port, einem String mit der Antwort und ob der Befehl erfolgreich ausgeführt wurde. Verwendung findet diese SubVI nur in *Main-Start.vi*, da nur diese mit der Steuerung kommuniziert.

Abstand Winkel(SubVI).vi.

Bei diesem VI wurde eine andere Form der Berechnung gewählt. LabVIEW ermöglicht es mit dem Formelknoten mathematische Formeln in C-Syntax auszudrücken. Die

[22] Die Wartezeit liegt für Befehle bei denen nur eine Spannung gesetzt wird bei <100ms, und für Befehle bei denen zwei Spannungen gesetzt werden bei <250ms. Die Zeit ergibt sich aus dem Senden des Befehls per USB, der Bearbeitung und dem Senden der Antwort durch den Mikrocontroller.

Variablen werden mit Eingangs- und Ausgangsknoten übergeben. Es dient vor allem der Übersicht, denn bei komplexeren Formeln werden schnell Fehler eingebaut. Das VI selbst berechnet den Abstand von zwei Koordinaten und den Winkel den diese mit der Vertikalen einschließen. Verwendet wird dieses SubVI in *View-Graphen-2d.vi*.

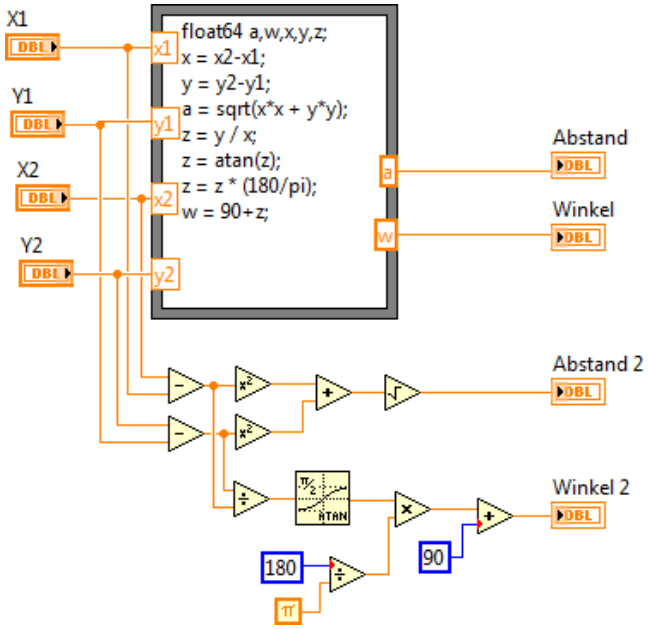


Abbildung 6.18: Blockdiagramm von *Abstand Winkel(SubVI).vi*, der obere Teil zeigt die Berechnung mithilfe des Formelknotens, der untere Teil die Berechnung nach dem Datenflussprinzip.

6.13. Dateien

Für die Speicherung der Messdaten wurden zwei neue Dateitypen (.spa1d und .spa2d) angelegt, deren Formatierung identisch ist. Als Zeichenkodierung für die Dateien wurde die ASCII-Kodierung gewählt, da bei dieser die Messparameter direkt mit einem Texteditor ausgelesen werden können. Gespeichert werden alle wichtigen Parameter (Messpunkteanzahl, Messzeit, Messfenstergröße in V und %BZ, Elektronenenergie, Winkel, Sensitivität, Offset) der genaue Aufbau ist in Kapitel A.3 dargestellt. Außerdem wurden noch vier Dateien erstellt, die sich im Unterordner „/lib“ des Programms befinden. Diese sind als Konfigurationsdateien angelegt, Eckige Klammen bezeichnen Abschnitte unter denen verschiedene Schlüssel mit Werten abgelegt werden können. Um einen Wert auszulesen, wird Kenntnis über den Abschnitt, den Schlüssel und den Datentyp benötigt. *specificData.spa* enthält verschiedene Werte die von der Steuerung und dem

verwendeten Messaufbau abhängen. Umrechnungen für die Messzeiteinstellung und die Elektronenenergie, den Winkel α (vgl. Abbildung 3.2) des SPA-LEED-Aufbaus, die Sensitivität und die Geräteversion, mit der diese Software kommunizieren kann. *Lagenabstand.spa* enthält für verschiedene Elemente den Lagenabstand, *Controller-Lagenabstand.vi* ist dazu da, die Werte aus der Datei auszulesen und zu übernehmen, und auch um neue Werte abzuspeichern. *Messvorlagen.spa* enthält die Werte für verschiedene Messungen, diese können über *Controller-Einstellungen.vi* ausgelesen und gespeichert werden. *Fensterpositionen.spa* enthält die Positionen und die Größe der Fenster nach der letzten Ausführung. Die Datei wird am Anfang von *Main-Start.vi* gelesen und am Programmende mit den veränderten Positionen neu geschrieben. So sind die Fenster nach einem Neustart immer wieder am selben Platz und müssen nicht erst wieder verschoben werden.

7. Erste Messungen

Das Kapitel stellt die ersten Messergebnisse vor, die mit der neu entwickelten Software aufgenommen wurden. 2D-Messungen mit der neuen Ansteuerung sind nicht direkt mit den alten Messungen vergleichbar, da die Ablenkung des Elektronenstrahls in einer anderen Reihenfolge erfolgt. Die Messungen werden im Folgenden für direkte Vergleiche mit alten Messungen um 180° gedreht dargestellt. Die ist leicht erkennbar, wenn die Messfenstergröße sehr groß gewählt wird, dass die Oktopolplatten ins Bild kommen, wie in Abbildung 7.4 dargestellt.

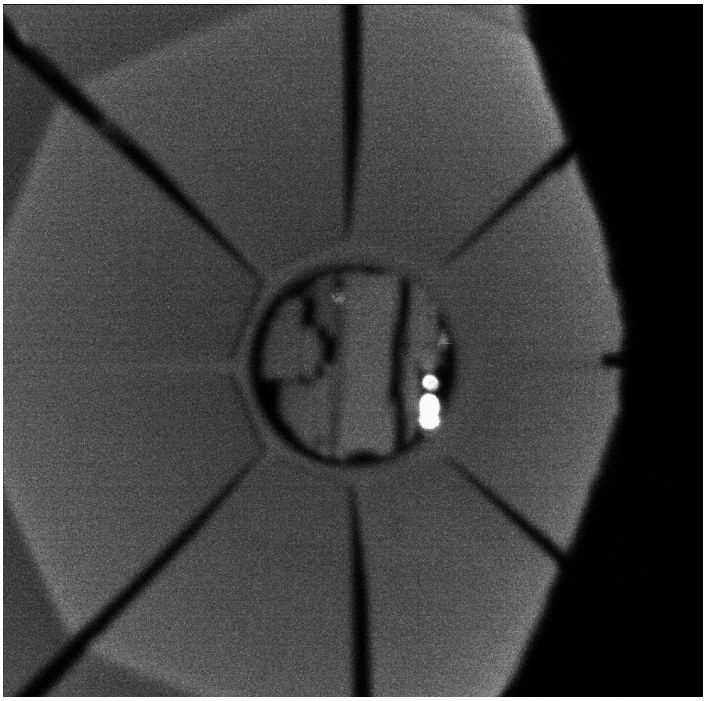


Abbildung 7.1: Messung im SEM-Modus.

Abbildung 7.1 zeigt die erste Aufnahme im SEM-Modus, dabei wurde ein 2001x2001-Scan ausgeführt mit einer Messzeit von 1ms, die Messfenstergröße lag bei 150V. Danach wurde eine zweite Messung (Abbildung 7.2) durchgeführt, bei der die Auflösung (200x200) und die Messfenstergröße (50V) verringert wurden, die Messzeit (10ms) jedoch erhöht wurde. Außerhalb des Kreises sind die verschiedenen vorderen Oktopolplatten gut zu erkennen. Die Strukturen innerhalb des Kreises zeigen den Manipulator und die Probe (längliches aufrechtes Rechteck). Beide Messungen zeigen, dass sich Teile des Manipulators aufladen (Weiße Punkte). Später wurde festgestellt, dass diese

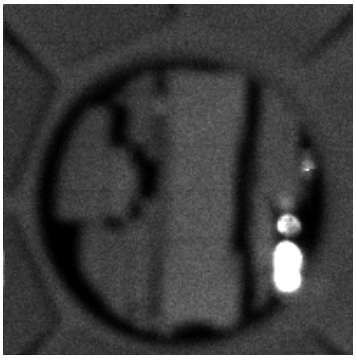
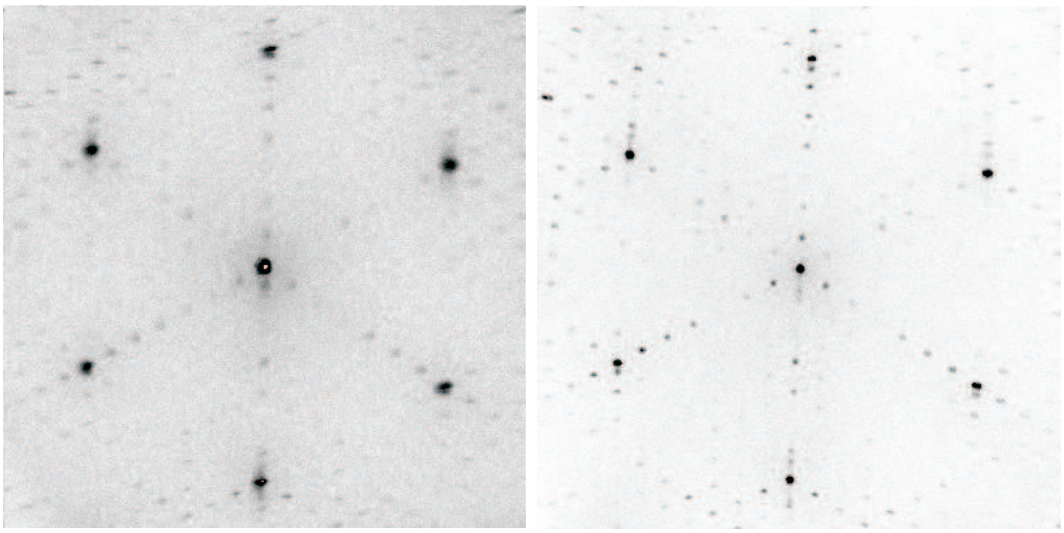


Abbildung 7.2: SEM-Modus kleiner Bereich mit weniger Punkten, aber mit längerer Messzeit.

Aufladung für eine Verbreiterung der Reflexe verantwortlich ist. Wird der Manipulator so eingestellt, dass der Teil mit der Aufladung nicht mehr im Scan-Bereich ist, zeigen die Aufnahmen weniger Rauschen und die Halbwertsbreiten der Reflexe werden kleiner.

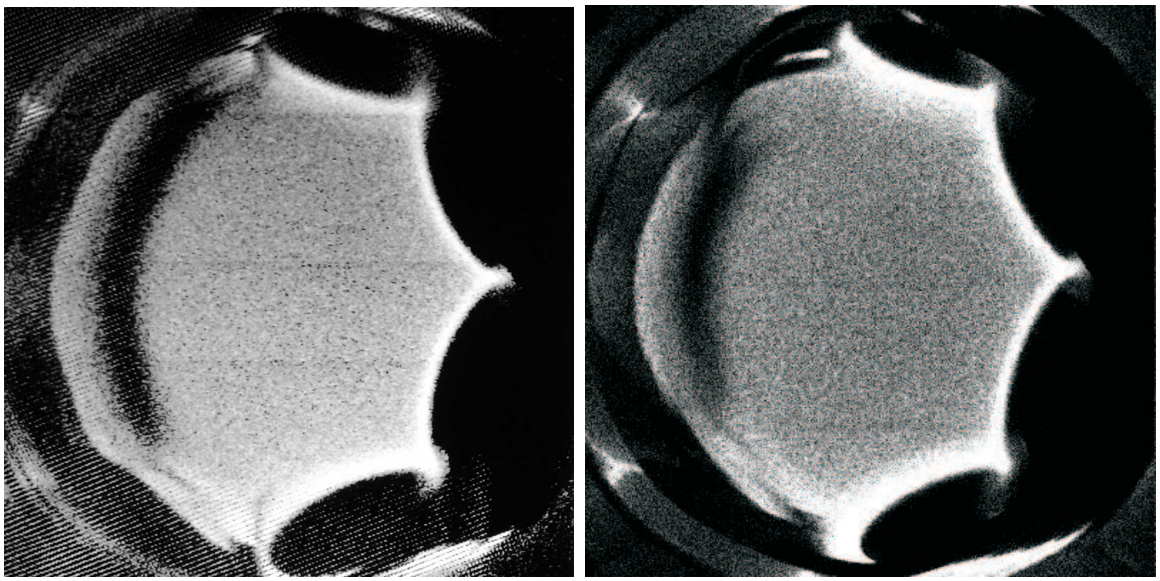
Die Messungen in Abbildung 7.3 zeigen einen Vergleich von Si(111)-Messungen von der alten und neuen Ansteuerung vor der Einstellung der Oktopol-Platten. Mit dem neuen Gerät sind die Reflexe deutlich schärfer und auch kleiner.



(a) alte Ansteuerung (b) neue Ansteuerung

Abbildung 7.3: Vergleich von Si(111).

Im Zuge eines Austausches des Filaments der Elektronenkanone wurden im Anschluss



(a) alte Ansteuerung (b) neue Ansteuerung

Abbildung 7.4: Messfenstergrösse 100V, Messzeit 1ms und Energie 91,9eV.

einige Messungen im 10min-Takt durchgeführt, um den Verlauf der Intensität anzuschauen (siehe Abbildung 7.5). Diese Messung wurde automatisch durchgeführt, dazu wurde ein zusätzlicher Zustand erstellt, der alle 10min eine Messung mit den gleichen Parametern durchgeführt hat. Nach jeder Messung wurde der höchste vorkommende Count in einen weiteren Graphen (siehe Abbildung 7.6) über die Zeit aufgetragen. Abbildung 7.6 zeigt das mit längerer Messdauer die Intensität abnimmt, so dass der Filamenttausch das Problem mit der fallenden Intensität nicht gelöst hat.

Abbildung 7.4 zeigt eine Messung, bei der die Probe schon lange im Vakuum war und daher keine Reflexe mehr bei Messungen zeigte. Deshalb wurde das Messfenster sehr groß (100V[23]) gewählt, um bei einer Messung die Oktopol-Platten zu sehen. Das Rauschen, welches im äußeren Bereich bei der alten Ansteuerung erkennbar ist, kann bei der neuen Ansteuerung nicht mehr beobachtet werden. Ebenso hat im Vergleich das allgemeine Rauschen abgenommen.

Die folgenden Messungen zeigen die Schritte, um die Verzerrung der Oktopolplatten zu minimieren. Verwendet wurde eine Si(111)-Probe, die vorher für zwei Tage ausgegast wurden und kurz vor der Messung geflasht wurde, um Verunreinigungen zu entfernen.

[23] bzw. 200V bei der alten Ansteuerung

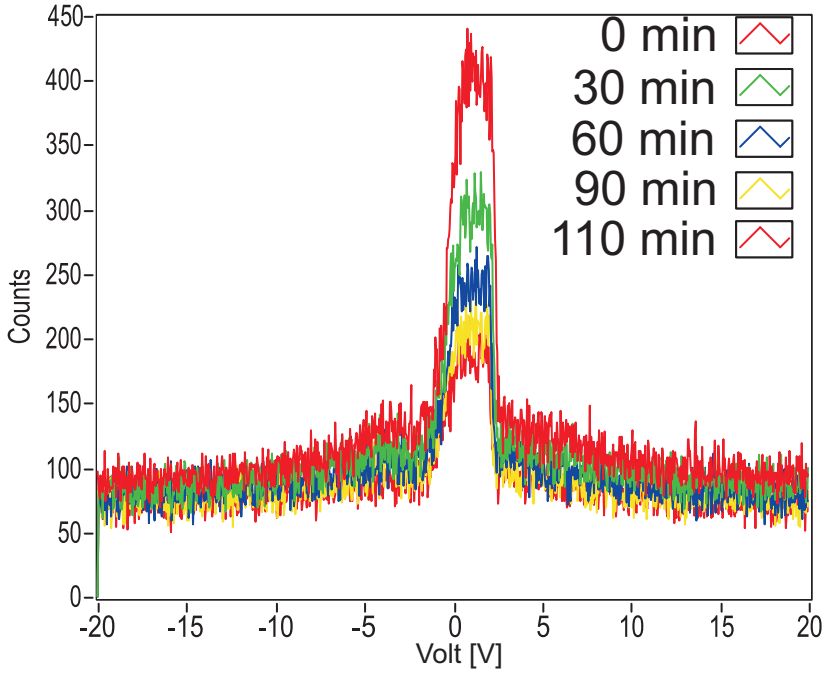


Abbildung 7.5: Linescans über den (00)-Hauptreflex von Si(111) (Messparameter: 100Messwerte, 1ms Gatetime und 20V Messfenstergröße).

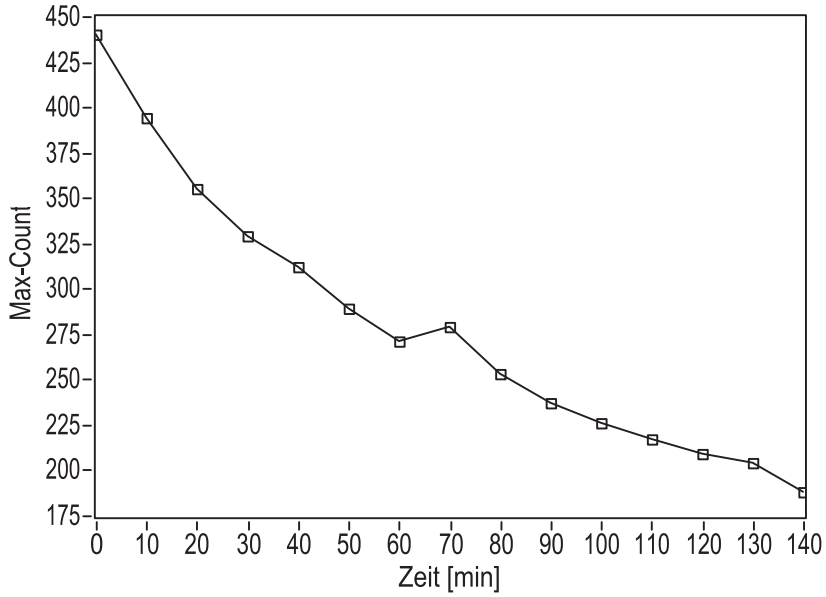


Abbildung 7.6: Verlauf der höchsten Counts über die Zeit.

Für die Messreihe wurde zuerst ein Vergleichsbild aufgenommen (Abbildung 7.7) und anschließend immer nur eine Endstufe für eine der vorderen Oktopolplatten geändert und nach der Messung wieder in Ausgangsstellung gebracht. Das Verhältnis der bei-

den Oktopole zueinander wurde für diese Messungen nicht beachtet. (Messparameter: Anzahl Messpunkt: 400x400, Messzeit: 2ms, Messfenstergröße: 43V (etwa 120%BZ))

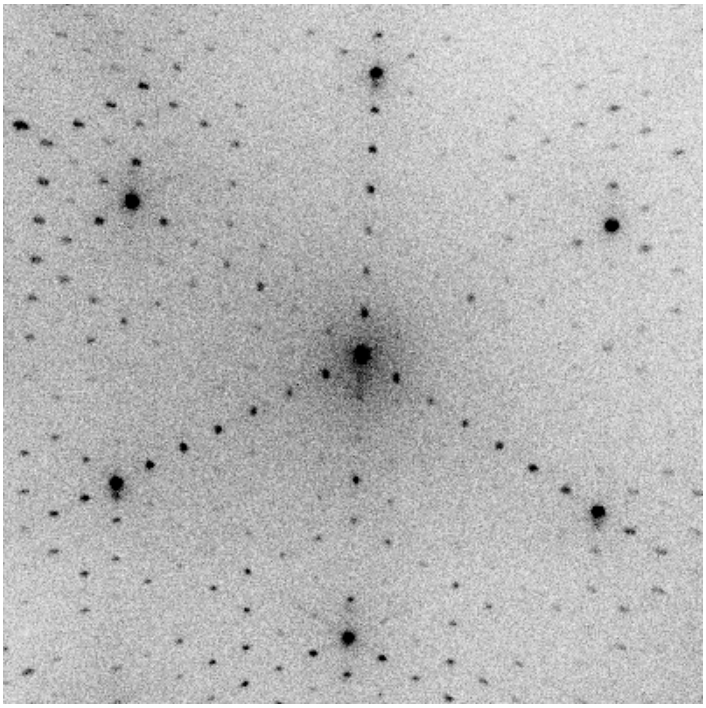
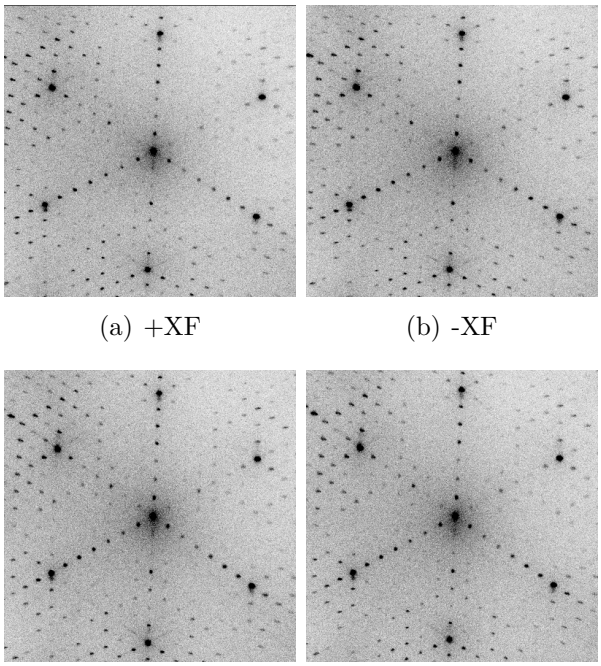


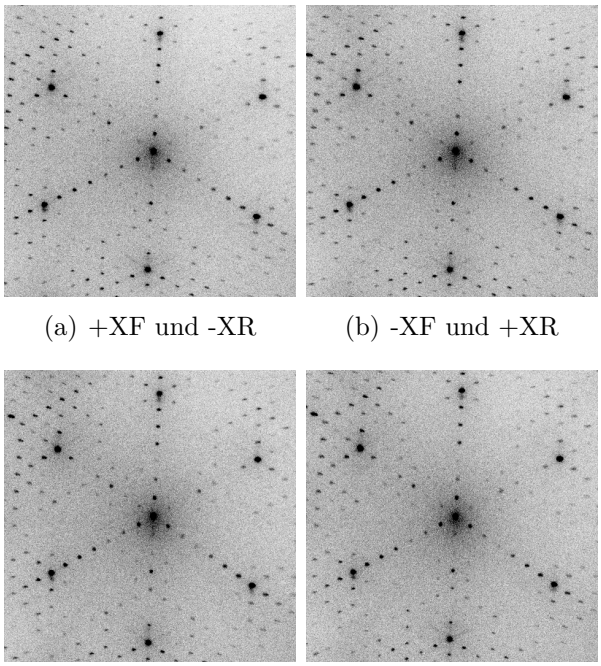
Abbildung 7.7: Messung vor der Variation der Endstufen.



(a) +XF (b) -XF

(c) +YF (d) -YF

Abbildung 7.8: Variation der vorderen Endstufen, es wurde jeweils eine Endstufe von 22300 auf 20000 heruntergeregelt.

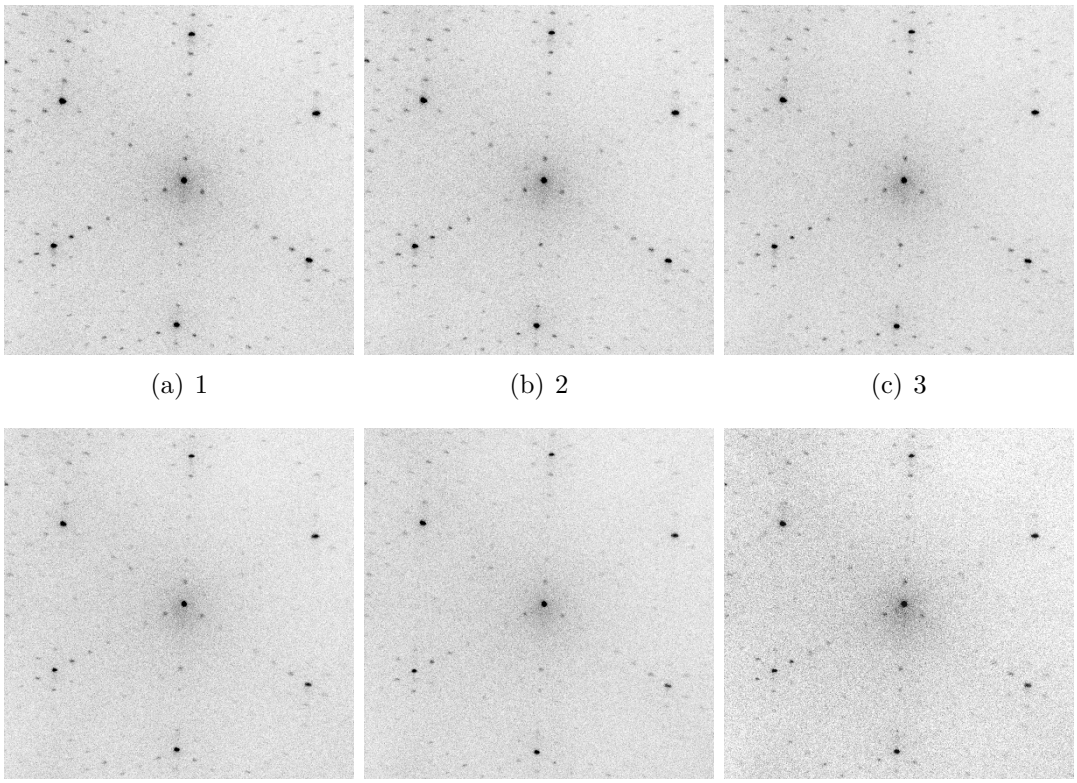


(a) +XF und -XR (b) -XF und +XR

(c) +YF und -YR (d) -YF und +YR

Abbildung 7.9: Variation der vorderen und hinteren Endstufen, Es wurde jeweils eine vordere Endstufe von 22300 auf 20000 heruntergeregelt und die dazugehörige hintere Endstufe von 13500 auf 12100.

Die Positionsveränderungen der Reflexe sind dabei jeweils nur im direkten Vergleich deutlich erkennbar. Die Reflexe wandern nach außen wenn der Multiplikator verringert wird und nach innen bei Erhöhung des Multiplikators. Durch diese Einstellungen ist es möglich die Endstufen einzustellen, um die Verzerrung der Reflexe zu minimieren. Um dies zu erreichen wurde die Probe zuerst in Mittelstellung gebracht, so dass der (00)-Reflex in der Bildmitte liegt. Damit der Offset keinen Einfluss auf die Messungen hat, wurde dieser in Nullstellung gelassen. Danach wurde jeweils der Abstand der Reflexe 1.Ordnung zum Hauptreflex gemessen. Dies geschieht mit dem im Programm integrierten Auswertefenster. Im Anschluss wurde die Sensitivität so eingestellt, dass der Abstand für alle Reflexe zum Mittelpunkt im Mittel 100% BZ entspricht. Für die Einstellung wurde wieder die Si(111) Probe benutzt, wegen der vielen Reflexe, bei der die Einstellungen überprüft werden können. Das Verhältnis der vorderen zu den hinteren Platten zueinander wurde nicht angetastet, die hinteren Endstufen wurden im gleichen Zuge mit den Vorderen angepasst. Die Abbildungen 7.10(a)-(e) zeigen die verschiedenen Zwischenstufen, wieder mit Abbildung 7.7 als Ausgangsmessung. Der Endzustand ist in Abbildung 7.10(f) dargestellt. Die endgültigen Einstellungen zeigt die Tabelle 5. Mithilfe der Anzeige kann dieser Zustand jederzeit wieder eingestellt werden. Angegeben wurden die Werte, die von der Matrixanzeige angezeigt wurden.



(a) 1 (b) 2 (c) 3

(d) 4 (e) 5 (f) Endzustand

Abbildung 7.10: Schritte bis zur endgültigen Einstellung.

Endstufe	Vor Einstellung	Nach Einstellung
+XF −XR	22300 14500	20700 12520
−XF +XR	22300 14500	21600 12950
+YF −YR	22300 14500	21700 13090
−YF +YR	22300 14500	21000 12610

Tabelle 5: Einstellungen der Endstufen vor und nach der Kalibrierung.

8. Zusammenfassung und Ausblick

Gegenstand dieser Arbeit war es eine Bediensoftware für die neue Oktopol-Steuerung zu entwickeln. Diese sollte die Funktionen der bisher benutzten SPA5.6 Software übernehmen, verbessern und erweitern. Das Ziel, all diese Punkte umzusetzen, wurde dabei vollständig erreicht und die Kombination aus neuer Software und neuer Ansteuerung wurde schon erfolgreich eingesetzt.
Vereinfacht wurden die Bedienung einmal durch die variable Benutzeroberfläche, die nicht mehr von der Bildschirmgröße abhängig ist und durch die Möglichkeit Messvorlagen zu erstellen, um so die Messparameter nicht immer eingeben zu müssen. Ebenso konnte die maximal mögliche Auflösung deutlich gesteigert werden. Die maximale Auflösung ist aber nicht Praxistauglich, da die Messdauer, selbst für kleine Messzeiten (<1ms) zu groß ist und damit die Qualität der Probe für nachfolgende Messung zu stark verringert wird. Erste Messungen haben gezeigt, dass sich das Rauschen insbesondere in den äußeren Bereichen deutlich gemindert hat. Die Messzeit, die für eine Messung benötigt wird, ist unverändert, da diese sich nur aus der Anzahl der Punkte multipliziert mit der Gatetime zusammensetzt. Mit dem neuen SEM-Modus lässt sich neben der Position der Probe auch die Probe selbst betrachten. Ebenso lassen sich eventuelle Aufladungen schnell erkennen.
Zukünftige Erweiterungen können durch die Umsetzung als Zustandsautomaten und die erstellten SubVIs einfach eingebaut werden. Erweiterungen wären beispielsweise neue Messarten einzubauen oder aber die Messmethoden aus dem Programm zu nehmen und dieses auf Bürocomputern als Auswertungssoftware zu benutzen, die entsprechenden Funktionen (Messung laden, speichern, auswerten und als Bild exportieren) sind bereits implementiert. Ein Tausch des in Abbildung 4.1 gezeigten LEED-SEM-Schalter mit einem Relais wäre eine weitere Möglichkeit. Damit wäre es möglich per Software zwischen den beiden Modi umschalten zu können.

Literatur- und Internetadressenverzeichnis

[1] J. Kwo, M. Hong, B. Busch, D.A. Muller, Y.J. Chabal, A.R. Kortan, J.P. Mannaerts, B. Yang, P. Ye, H. Gossmann, A.M. Sergent, K.K. Ng, J. Bude, W.H. Schulte, E. Garfunkel, and T. Gustafsson. Advances in high k gate dielectrics for Si and III-V semiconductors. *Journal of Crystal Growth*, 251:645–650, 2003.

[2] K. Asami, K. Kusakabe, N. Ashi, and Y. Ohtsuka. Synthesis of ethane and ethylene from methane and carbon dioxide over praseodymium oxide catalysts. *Applied Catalysis*, 156:43–56, 1997.

[3] Jari Rodewald. Thermische Stabilität von ultradünnen Ge-Schichten auf Pr_2O_3/Si(111). Bachelorarbeit, Universität Osnabrück, 2011.

[4] G. Meyer U. Scheithauer and M. Henzler. A new LEED Instrument for Quantitative Spot Profile Analysis. *Surface Science*, 178:441–451, 1986.

[5] Sebastian Gevers. SPA-LEED-Untersuchungen von Praseodymoxidschichten auf Si(111)-Substraten. Diplomarbeit, Universität Osnabrück, 2007.

[6] Daniel Bruns. SPA-LEED-Untersuchungen zur Epitaxie von Praseodymoxidschichten auf Si(111). Diplomarbeit, Universität Osnabrück, 2008.

[7] Alexander Knobeler. Praeseodymoxidschichten auf Bor-passivierten Si(111)-Oberflächen. Diplomarbeit, Universität Osnabrück, 2010.

[8] Wolfgang Georgi and Ergun Metin. *Einführung in LabView*. Carl Hanser Verlag, 2009.

[9] Prof. Dr. M. Horn von Hoegen. `http://www.spaleed.de/`. Universität Duisburg-Essen.

[10] Steffen Jentsch. Automatisierung der SPA-LEED-Control zur Erstellung von Heringsplots durch Editierung der SPA-LEED-Software (SPA5). Studienprojekt, Universität Osnabrück, Juni 2010.

[11] Ch. Kittel. *Einführung in die Festkörperphysik*. Oldenbourg, 2005.

[12] P.Zahl and M. Horn von Hoegen. Third generation conical spot profile analyzing low-energy electron difffraction. *Review of Scientific Instruments*, 73:2958, 2002.

[13] Oliver Vornberger. Computergrafik Skript, 2010.

[14] `http://www.labviewforum.de/`.

[15] `http://www-wnt.gsi.de/lvug/techniken.htm`.

[16] National Instruments. `http://forums.ni.com/`.

[17] Claudius Klein, C.Wiethoff, B.Krenzer, and C.Seifert. DnDLoad, Igor Procedurs File. SPA-LEED-Wiki, 2009.

Abbildungsverzeichnis

Tabellenverzeichnis

A. Anhang

A.1. Spannungsverhältnisse und Anschlüsse

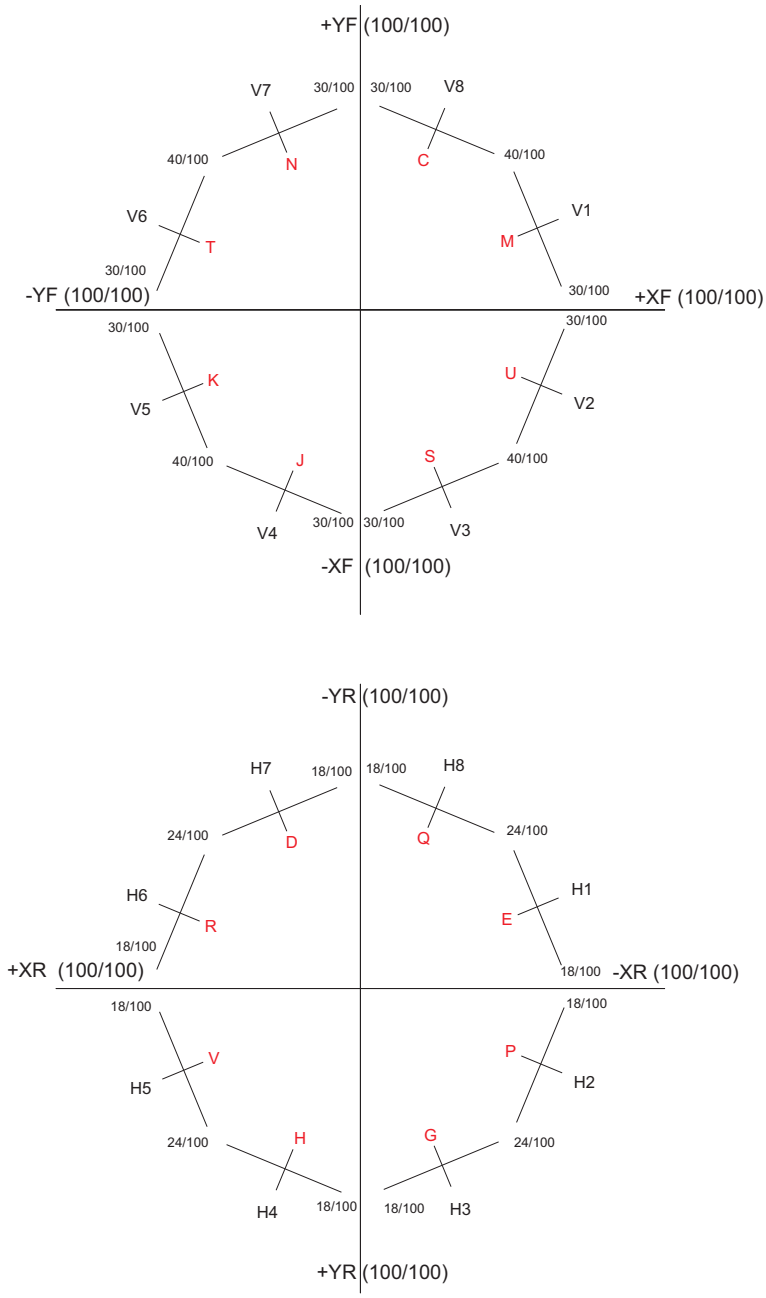


Abbildung A.1: Spannungsverhältnisse der vorderen und hinteren Oktopolplatten mit Steckerbelegegung nach Tabelle 6.

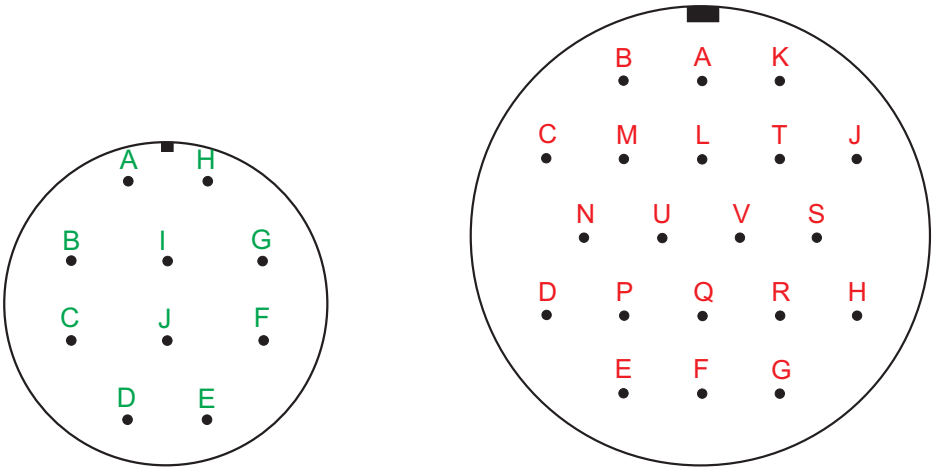


Abbildung A.2: Pinbezeichnung des 10pol. Elektronenkanonensteckers und des 20pol. Oktopolsteckers, Aufsicht. Pinbelegung siehe Tabelle 6. Entnommen aus [7].

A	Filament	A	Gehäuse
B	Filament	B	Repeller
C	Wehnelt („Energy“)	C	V8
D	Anode	D	H7
E	Dist.	E	H1
F	Blende 1	F	Kristalllinse - bzw. Masse
F	Linse („Focus 1“)	G	H3
H	Blende 2	H	H4
I	nicht belegt	J	V4
J	nicht belegt	K	V5
		L	Kristalllinse + („Focus 2“)
		M	V1
		N	V7
		P	H2
		Q	H8
		R	H6
		S	V3
		T	V6
		U	V2
		V	H5

Tabelle 6: Pinbelegung des 10pol. Elektronenkanonensteckers und des 20pol. Oktopolsteckers entsprechend der Abbildung A.2. Entnommen aus [7].

A.2. Kommandos für die Oktopol-Steuerung

Befehl[24]	Rückgabe	Beschreibung
@?*	Version x.xx\r\n	Gibt den Versionsstring zurück
@fon* @foff*	Filter High\r\n FilterLow	schaltet den Tiefpassfilter
@set P1*	Energie: P1\r\n	belegt die Energieausgabe mit dem Wert P1
@clear*	Clear \r\n	Das Messfenster wird geschlossen und die Scan- und Offset DACs in Mittelstellung (0V) gebracht.
@winX P1* @inY P1* @size P1 P1*	window_X P1\r\n windows_Y P1\r\n size w:P1 h:P1\r\n	stellt die Größe des Messfensters ein.
@offX P2* @offY P2* @move_abs P2 P2* @move_rel P2 P2*	offset_X P2\r\n offset_Y P2\r\n move_abs. x:P2 y:P2\r\n move_rel. x:P2 y:P2\r\n	stellt die Verschiebung des Messfensters (Fenstermitte) ein. move_abs verschiebt absolut move_rel verschiebt relativ.
@scanX P1* @scanY P1*	@scan_X P1\r\n @scan_Y P1\r\n	bestimmt die Position des Messpunktes innerhalb des Messfensters.
@scanP P1 P1 P3*	Scan Point X:P1 Y:P1 Z:P3\r\n Wert,\r\n	startet eine Messung an einem bestimmten Punkt innerhalb des Messfensters, mit dem Messintervall P3

Befehl[24]	Rückgabe	Beschreibung
@scp P1*	points\scan: P1\r\n	setzt die zu messende Punktmenge für die Scan-Befehle @scanVline; @scanHline; @scanray; @scanTray.
@scanVline P1 P3* @scanHline P1 P3*	Werte\r\n scanVline start Y:P1 interval:P3\r\n Werte\r\n scanHline start X:P1 interval:P3\r\n	startet eine Messreihe über (@scp) Punkte mit dem Messintervall P3. Diese werden gleichmäßig über die Zeile/Spalte verteilt. Die Zeile/Spalte wird mit P1 festgelegt. Die Ausmaße des Messfensters (@size) bestimmt die Distanz zwischen den Messpunkten.
@scanray P1 P3*	Werte\r\n scanray Angle:P1 interval:P3\r\n	startet eine Messreihe über (@scp) Punkte mit dem Messintervall P3. Diese werden gleichmäßig über den Messstrahl verteilt. Der Winkel zur Y-Achse wird über P1 als Grad*10 übergeben. Gültige Werte P1(0-1800). Die Ausmaße des Messfensters (@size) bestimmt die Distanz zwischen den Messpunkten.

Befehl[24]	Rückgabe	Beschreibung
@scanTray P1 P3*	Werte\r\n scanTray Angle:P1 interval:P3\r\n	startet eine Messreihe über (@scp) Punkte mit dem Messintervall P3. Diese werden gleichmäßig über die X-Achse (45-135) oder Y-Achse(0-44 & 136-180) verteilt. Der Winkel zur Y-Achse wird über P1 als Grad*10 übergeben. Gültige Werte P1(0-1800). Die Ausmaße des Messfensters (@size) bestimmt die Distanz zwischen den Messpunkten.
@scan P1 P3*	Werte\r\n scan points:P1 interval:P3\r\n	startet eine Messreihe über P1 Punkte mit einem Messintervall P3. Die Größe des Messfensters bestimmt die Rasterung der Messpunkte und muss im Vorfeld bestimmt werden.
@test*	Test \r\n Werte\r\n	startet eine Testmessung. Die Fenstergröße wird auf 2/3 festgelegt. Es werden 50*50 Punkte mit einem Messintervall von 2ms gemessen.
falscher Befehl	wrong command:Befehl\r\n	

Tabelle 7: Alle ausführbaren Kommandos der Oktopol-Steuerung mit Rückgabewerten und Beschreibung.

[24]Erlaubte Werte:
P1: 0-65535 (16bit unsigned)
P2: -32767 - +32767 (16bit signed)
P3: 0-494967295 (32bit unsigned)

A.3. Neue und Alte Dateitypen

Die Software kann mit insgesamt vier Dateitypen für die Messdaten umgehen, einmal den beiden Neuen .spa1d und .spa2d sowie den beiden alten Binärdateitypen .d1d und .d2d. Eingelesen werden können alle vier, gespeichert werden kann nur in den beiden neuen Dateiformaten. Der Aufbau der neuen ASCII-kodierten Dateien sieht wie folgt aus:

boolean; int; int; double; double; double; double; double; double; double;\r\n
(Werte als Integer durch Komma getrennt)\r\n
...
(Werte)\r\n // *Für 1D-Scans nur eine Zeile*

Die Daten in der ersten Zeile stehen dabei für:
TRUE=2D/FALSE=1D; Anzahl Messpunkte; Messzeit in μs; Messfenster in V; Messfenster in %BZ; Energie in eV; Winkel in Grad; Sensitivität; Offset X in V; Offset Y in V;

Die wichtigen Daten zum Einlesen der .d1d und .d2d-Dateien wurden dem Skript [17] entnommen, die entsprechenden Zeilen sind dabei in Code 1 und Code 2 dargestellt.

```
// Datenstruktur: taken from read_d1d for IDL
// 0: x:double
// 1: y:double
// 2: x1:double
// 3: x2:double
// 4: y1:double
// 5: y2:double
// 6: dx:double
// 7: dy:double
// 8: x0:double
// 9: y0:double
// 10: gatetime: double
// 11: xydist:double
// 12: cpshigh:double
// 13: cpslow:double
// 14: alpha: double
// 15: usr1:double
// 16: creg1:byte
```

```
19 // 17: creg2:byte
20 // 18: creg3:byte
21 // 19: creg4:byte
22 // 20: creg5:byte
23 // 21: creg6:byte
24 // 22: creg7:byte
25 // 23: creg8:byte
26 // 24: scanmarker: integer
27 // 25: points:integer
28 // 26: energy:double
29 // 27: focus1:double
30 // 28: current:double
31 // 29: usr2:double
32 // 30: usr3:double
33 // 31: usr4:double
34 // 32: focus2:double
35 // 33: extractor:double
36 // 34: dummy3:double
37 // 35: dateofscan: bytarr(21)
38 // 36: comment: bytarr(117)
```

Code 1: Zeilen 922-959 aus [17], zeigt den Aufbau von .d1d-Dateien

```
1  // Read in additional data
2  GBLoadWave/Q/O/B/N=data/T={4,4}/W=1/U=18 fileReferenz
3  variable/G energy = data0[17]
4  variable/G minCount = data0[13]
5  variable/G maxCount = data0[12]
6  variable/G volts = data0[11]
7
8  GBLoadWave/Q/O/B/N=data/T={16,4}/W=1/U=107 fileReferenz
9  variable/G xPix = data0[105]
10 variable/G yPix = data0[106]
11 variable xDim = xPix
12
13 // function reads in xDim + 1 arrays of signed integers
14 // with base name 'data'
15 GBLoadWave/Q/O/B/N=data/T={16,4}/S=386/W=(xDim+1)fileReferenz
```

Code 2: Zeilen 329-342 aus [17] zeigen teilweise die Datenstruktur von .d2d-Files

A.4. Installationsanleitung

Installieren der Treiber für den Arduino Mega 2560:

Zuerst die Steuerung mit einem USB Kabel an einen PC anschließen. Der Assistent zum Suchen neuer Hardware sollte sich automatisch öffnen. Dieser installiert dann automatisch den passenden Treiber. Startet der Assistent nicht, muss der Gerätemanager[25] aufgerufen werden, hier sollte ein unbekanntes Gerät angezeigt werden. Ein Rechtsklick und auf Treibersoftware aktualisieren klicken. Wird der Treiber nicht automatisch gefunden, dann die Datei „Arduino MEGA 2560.inf“ auswählen.

Installieren der Software:

Komplett:
Setup.exe ausführen und den Anweisungen folgen.

Einzeln:
Herunterladen und Installieren der „*LabVIEW Runtime 2012 (32bit)*“ oder neuer und der „*NI-VISA Runtime 5.4*“ oder neuer von der National Instruments Homepage.
Danach Setup.exe ausführen und den Anweisungen folgen.

Mit der Installation wird automatisch eine Verknüpfung auf dem Desktop angelegt und ein Eintrag im Startmenü erstellt.

[25] Systemsteuerung→System→Gerätemanger

B. Danksagung

An diese Stelle möchte ich mich bei allen Bedanken die zum Gelingen dieser Arbeit beigetragen haben. Zunächst bei Herrn Prof. Dr. Joachim Wollschläger, der es mir ermöglicht hat dieses interessante Thema zu bearbeiten.
Darüber hinaus bedanke ich mich bei Frau Prof. Dr.-Ing. Elke Pulvermüller, die sich bereit erklärt hat, die Position der Zweitgutachterin zu übernehmen. Meinen Betreuern Dr. Daniel Bruns, Henrik Wilkens und Frederic Timmer möchte ich ebenfalls danken, sie hatten immer ein offenes Ohr für technische und inhaltliche Fragen und auch Wünsche für neue Funktionen. Bedanken möchte ich mich auch bei Wanja und Robert, die als Versuchspersonen herhalten mussten und natürlich beim Rest der Arbeitsgruppe „Dünne Schichten und Grenzflächen“ für die super Atmosphäre.
Ebenso danke ich der Elektronikwerkstatt für den Bau der neuen Geräte, insbesondere Herrn Bernd Lemme, der die gewünschten Funktionen für den Mikrocontroller umgesetzt hat und die netten Unterhaltungen über die Funktionsweise.
Weiterhin danke ich meiner Familie, für die Unterstützung und die Ermöglichung des Physikstudiums.

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbständig angefertigt, außer der angegebenen Literatur keine weiteren Hilfsmittel verwendet und noch keinen Versuch eine Diplomarbeit zu schreiben unternommen habe.

Osnabrück, 13. Dezember 2013