FH Westküste
Wirtschaft und Technik

HAW HAMBURG

Masterarbeit

Nils Voß

Quasistatische Ansteuerung und Charakterisierung von MEMS-Spiegeln

Fachhochschule Westküste
Fachbereich Technik

Fachhochschule Westküste - University of Applied Sciences
Faculty of Engineering

Hochschule für Angewandte Wissenschaften Hamburg
Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Hamburg University of Applied Sciences
Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Nils Voß

Quasistatische Ansteuerung und Charakterisierung von MEMS-Spiegeln

Masterarbeit eingereicht im Rahmen der Masterprüfung
im gemeinsamen Masterstudiengang Mikroelektronische Systeme
am Fachbereich Technik
der Fachhochschule Westküste
und
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Sönke Appel
Zweitgutachter: Prof. Dr. Peter Schulz

Eingereicht am: 21. Nov. 2025

Nils Voß

Thema der Arbeit

Quasistatische Ansteuerung und Charakterisierung von MEMS-Spiegeln

Stichworte

Quasistatische Ansteuerung, MEMS-Spiegel, MEMS

Kurzzusammenfassung

In dieser Arbeit wird eine hardwarenahe Ansteuer- und Auswertekette für quasistatisch betriebene piezoelektrische MEMS-Spiegel entwickelt und experimentell charakterisiert. Zunächst werden die statische Auslenkung, die Sensorspannungen sowie das Stoß- und Frequenzverhalten untersucht, wobei dominierende Eigenmoden bei etwa 268 Hz (Kippmode) und 466 Hz (Hubmode) identifiziert werden. Auf Basis dieser Charakterisierung wird eine Open-Loop-Ansteuerung auf Mikrocontrollerbasis sowie eine FPGA-basierte Proportionalregelung einschließlich analoger Signalaufbereitung implementiert. Die Verifikation zeigt, dass der erste Regelungsansatz die Kippmode leicht dämpft, die Hubmode jedoch verstärkt und die Abklingzeit nach Stoßanregungen nicht verkürzt. Die Ergebnisse liefern damit wesentliche Systemkenngrößen und eine funktionsfähige Hardwareplattform als Grundlage für weiterführende Regelungsstrategien und eine zukünftige ASIC-Integration.

Nils Voß

Title of Thesis

Quasistatic Control and Characterization of MEMS Mirrors

Keywords

Quasistatic Control, MEMS Mirrors, MEMS

Abstract

This thesis presents the development and experimental characterization of a hardware-oriented drive and readout system for quasistatically actuated piezoelectric MEMS mirrors. The static deflection, sensor response, and shock- and frequency-domain behavior are analyzed, revealing dominant eigenmodes at approximately 268 Hz (tilt mode) and 466 Hz (piston mode). Based on these results, a microcontroller-based open-loop drive and an FPGA-based proportional control scheme, including analog signal conditioning, are implemented. Verification shows that the initial control approach slightly attenuates the tilt mode but amplifies the piston mode and does not reduce the decay time after shock excitation. The results provide key system parameters and a functional hardware platform as a foundation for improved control strategies and future ASIC integration.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

ADC Analog digital Umsetzer.

AlN Aluminiumnitrid.

AlScN Aluminiumscandiumnitrid.

ASIC anwendungsspezifische integrierte Schaltung.

AVT Aufbau und Verbindungstechnik.

CNC Computerized Numerical Control.

DAC Digital-Analog-Umsetzer.

FMD Feder-Massen-Dämpfersystem.

FOV Field of View.

FPGA Field Programmable Gate Array.

HDL Hardware Description Language.

IP-Block Intellectual Property Block (wiederverwendbarer Funktionsbaustein).

LDV Laserdoppler-Vibrometer.

LiDAR Light Detection and Ranging.

MEMS Mikro-Elektromechanisches System.

PCB Printed circuit board.

PZT Blei-Zirkonat-Titanat.

SAR Successive Approximation Register.

SPI Serial Peripheral Interface.

TOSA Total optical scan angle.

1 Einleitung

Mikro-Elektromechanisches System (MEMS)-Spiegel sind entscheidend für die Entwicklung vieler moderner Technologien. Sie ermöglichen die präzise Steuerung von Lichtstrahlen in Anwendungen wie Laserscanning, Light Detection and Ranging (LiDAR) und Beamer-Technologien. Durch ihre Miniaturisierung und hohe Flexibilität bieten sie eine kostengünstige Alternative zu herkömmlichen mechanischen Spiegeln. MEMS-Spiegel sind daher eine Schlüsseltechnologie für die Verbesserung der Effizienz und Leistungsfähigkeit optischer Systeme. Ein bekanntes Problem quasistatischer MEMS-Spiegel ist jedoch ihr Schwingungsverhalten bei mechanischen Stoßeinwirkungen [26].

Am Fraunhofer-Institut für Siliziumtechnologie wurden bereits MEMS-Spiegelanwendungen mit resonant betriebenen Spiegeln realisiert. Dabei wird der Spiegel kontinuierlich mit seiner Resonanzfrequenz angeregt, wodurch in vielen Anwendungen dichte Lissajous-Figuren erzeugt werden. Im ersten Teil dieser Arbeit wird eine quasistatische Ansteuerung entwickelt, mit der ein beliebiges Scanmuster erzeugt werden kann. Dies bietet unter anderem die Möglichkeit, relevante Stellen häufiger oder präziser mit dem Laser abzutasten. Die Einschränkung, stets das gleiche Scanmuster durchfahren zu müssen, entfällt, da bei einer quasistatischen Ansteuerung jede Koordinate direkt angefahren werden kann.

Es ist bekannt, dass auch quasistatische Spiegel mehrere Resonanzfrequenzen besitzen und deren Anregung zu starken Auslenkungen führen kann. Solche extremen Auslenkungen können dazu führen, dass LiDAR-Systeme ihren vorgesehenen Scanbereich verlieren.

Im zweiten Teil dieser Arbeit wird das unerwünschte Anregen der Resonanzen infolge eines mechanischen Stoßes untersucht. Die Detektion der Stoßeinwirkung basiert auf einem neuartigen Sensorelement, das eine Messung der Aktuatorbewegung unabhängig vom Treibersignal ermöglicht. Anhand der gewonnenen Messdaten wird anschließend eine Closed-Loop-Steuerung entwickelt, die das Abklingen nach einem Stoß verkürzen soll. Als Plattform für die Regelung dient ein Field Programmable Gate Array (FPGA), das hohe

Flexibilität bietet und einen ersten Meilenstein für eine spätere Implementierung der Steuerung in einem anwendungsspezifische integrierte Schaltung (ASIC) darstellt, die zukünftig direkt auf dem Chip integriert werden könnte.

Die quasistatischen MEMS-Spiegel müssen dafür zunächst charakterisiert werden, um die Grenzen der Anwendung zu bestimmen. Die Spiegel sind über ein Federsystem aufgehängt, das mechanische Spannungen an kritischen Biegestellen reduziert. Dieses Federsystem kann jedoch dazu führen, dass der Spiegel bei Stoßeinwirkungen unkontrolliert schwingt. Ein Ziel dieser Arbeit ist es daher, diese Schwingungen zumindest teilweise zu dämpfen.

Damit sowohl die teilweise Dämpfung des Spiegelsystems als auch eine geeignete Steuerung möglich ist, wird das Frequenzverhalten des Chips aufgenommen und mit dem Laserdoppler-Vibrometer (LDV) analysiert. Auf Basis der Ergebnisse aus der Charakterisierung soll anschließend eine künstliche Dämpfung über eine Closed-Loop-Regelung realisiert werden.

2 Grundlagen

2.1 Die Resonanzfrequenz

Bei MEMS-Spiegeln besitzt jede relevante Schwingungsform, insbesondere die translatorische und die rotatorische Mode, eine jeweils eigene Resonanzfrequenz. Diese ergibt sich aus der mechanischen Dynamik des Systems und wird maßgeblich durch die Geometrie, die effektive Masse sowie die Steifigkeit der zugehörigen Federstrukturen bestimmt. Die Resonanzfrequenz kennzeichnet den Betriebszustand, in dem der Spiegel mit minimalem Energieaufwand eine maximale Auslenkung erreicht. Während dieses Verhalten bei rotatorischen Moden gezielt zur effizienten Winkelauslenkung genutzt werden kann, führt eine unbeabsichtigte Anregung translatorischer Moden häufig zu unerwünschten Kopplungen oder Instabilitäten.

Das Auftreten von Resonanzen kann sowohl funktionale Vorteile bieten, etwa eine erhöhte Empfindlichkeit oder größere Amplituden, als auch Nachteile wie zusätzliche Schwingungen bis hin zur potenziellen Überlastung oder Zerstörung des Bauelements verursachen. Aus Gleichung 2.1 wird deutlich, dass sich die Resonanzfrequenz durch Änderungen der effektiven Masse oder der Federkonstanten gezielt beeinflussen lässt. In dieser Arbeit erfolgt die Unterdrückung unerwünschter Resonanzen jedoch primär auf elektronischem Wege. Der Schwerpunkt liegt auf einer quasistatischen Ansteuerung, bei der die Betriebsfrequenzen so gewählt werden, dass keine der relevanten Eigenmoden, weder rotatorisch noch translatorisch, angeregt wird.

$$f_0 = \frac{1}{2\pi}\sqrt{\frac{k}{m}} \qquad \text{(vgl. [24, S.6])} \tag{2.1}$$

2.2 Der Stoß

Der Stoß wird als eine plötzliche, kurzzeitige Kraft definiert, die auf ein Objekt wirkt. Zur Nachbildung dieser Eigenschaften wird häufig die Sprungfunktion herangezogen. Sie eignet sich insbesondere zur Modellierung von Umwelteinflüssen, wie beispielsweise Erschütterungen in Fahrzeugen, die durch Schlaglöcher verursacht werden können. Neben der Rechteckfunktion zählt die Sprungfunktion zu den am häufigsten verwendeten aperiodischen Testsignalen. [16]

Die Sprungfunktion lautet:

$$\varepsilon(t) = \begin{cases} 0, & \text{für } t < 0 \\ 1, & \text{für } t \geq 0 \end{cases} \tag{2.2}$$

Die Rechteckfunktion lautet:

$$rect(t) = \begin{cases} 0, & \text{für } |t| < \frac{1}{2} \\ 1, & \text{für } |t| > \frac{1}{2} \end{cases} \tag{2.3}$$

Obwohl die Theorie sprunghafte Signaländerungen zulässt, weisen reale Signale aufgrund der physikalischen Eigenschaften der Übertragungsmedien stets endliche Anstiegszeiten auf. Diese begrenzen die effektive Bandbreite des Signals [20].
Mit der Faustformel 2.4 lässt sich eine schnelle Abschätzung der Bandbreite einer Sprungantwort vornehmen. Die Formel liefert eine Näherung der Bandbreite im Bereich der 3 dB-Grenzen. Standardmäßig wird hierfür die Anstiegszeit zwischen 10 % und 90 % des Endwertes herangezogen. Es ist zu beachten, dass diese Beziehung nur für einfache Tiefpasssysteme erster Ordnung gilt.

$$B \approx \frac{0{,}35}{t_r} \qquad \text{(vgl. [8, S.103])} \tag{2.4}$$

2.3 Indirekter piezoelektrischer Effekt

Bei piezoelektrischen Aktuatoren wird der indirekte piezoelektrische Effekt genutzt. Durch das Anlegen einer elektrischen Spannung an ein piezoelektrisches Material entsteht eine

mechanische Verformung. Abbildung 2.1 zeigt schematisch die Veränderung der Kristallstruktur eines hexagonalen Systems. Abhängig von der Feldrichtung kann das Material gestaucht oder gedehnt werden. Der Polarisationsvektor richtet sich dabei entgegen dem elektrischen Feld entlang der c-Achse aus und bewirkt die Verformung des Materials. In Abbildung 2.1 (a) ist die Stauchung des Kristallgitters parallel zum elektrischen Feld (entlang der c-Achse) dargestellt. Wird die Spannung an den Elektroden umgepolt, kann man in Abbildung 2.1 (b) eine Streckung des Materials erkennen. Diese Streckung resultiert aus der Stauchung des Kristallgitters orthogonal zum elektrischen Feld.[2]

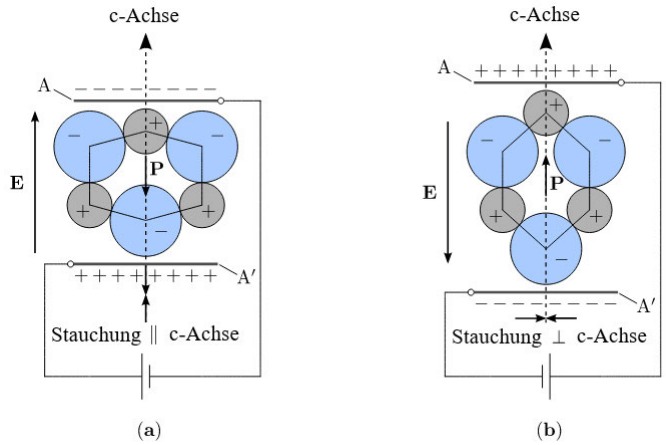

Abbildung 2.1: Indirekter piezoelektrischer Effekt aus [2, S.6]

2.4 Direkter piezoelektrischer Effekt

Beim piezoelektrischen Sensor wird, ebenso wie bei der Energieerzeugung, der direkte piezoelektrische Effekt genutzt. Dabei wirkt eine mechanische Kraft auf das piezoelektrische Kristallgitter. Die dadurch verursachte Verformung verschiebt den Ladungsschwerpunkt der Kristallstruktur, was zu einer elektrischen Polarisation führt. Je nach Richtung der angelegten Kraft stellt sich eine positive oder negative Spannung ein. In Abbildung 2.2 sind die resultierenden Verformungen sowie die zugehörigen Spannungsrichtungen dargestellt. In Abbildung 2.2(a) ist die Krafteinwirkung entlang der c-Achse gezeigt, was zu einer Stauchung des Materials führt, die eine Polarisation P auslöst. In Abbildung 2.2(b) ist die Krafteinwirkung seitlich über die Flächen B schematisch dargestellt. Bei der Krafteinwirkung über die Flächen B richtet sich der Polarisationsvektor entgegengesetzt

zu Fall (a) aus. Die Spannung an den Elektroden A und A' ist einmal positiv und einmal negativ.[2]

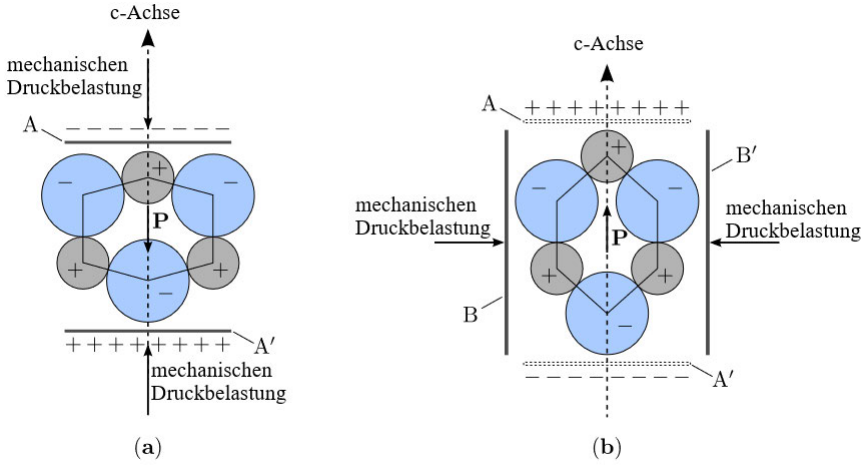

Abbildung 2.2: Direkter piezoelektrischer Effekt aus [2, S.5]

2.5 Laserdoppler-Vibrometer (LDV)

Das LDV ist ein berührungsloses Messverfahren zur Bestimmung von Schwingungen und geradlinigen Bewegungen mit hoher Genauigkeit. Grundlage ist die Interferometrie, mit der kleinste Abstandsänderungen der Oberfläche erfasst werden können, sowie der Dopplereffekt, mit dem die momentane Geschwindigkeit der Oberfläche bestimmt werden. Dadurch lassen sich die Bewegungen kleiner MEMS-Strukturen analysieren und mit Simulationsergebnissen vergleichen. Mit ausreichend vielen Messpunkten und können die Ergebnisse zudem animiert werden, sodass Aussagen zur Verformung der einzelnen Bauteile möglich sind.

Beim LDV wird ein Laserstrahl auf das Messobjekt gerichtet, dort reflektiert und anschließend detektiert. Parallel dazu wird ein Referenzstrahl mittels Strahlteiler abgezweigt und ebenfalls dem Detektor zugeführt. Bewegt sich die Oberfläche in Richtung oder entgegen der Strahlquelle, erfährt der zurücklaufende Messstrahl eine Dopplerverschiebung. Die Überlagerung von Referenz- und dopplerverschobenem Messstrahl erzeugt eine Schwebung, deren Frequenz die Bewegung der Oberfläche codiert und damit die Be-

stimmung der Geschwindigkeit ermöglicht. Aus der Amplitude der Schwebung lässt sich der momentane Abstand ablesen. [13]

In Abbildung 2.3 ist der schematische Aufbau eines LDV dargestellt. Die monochromatische Quelle emittiert mit der Grundfrequenz

$$f_0 = \frac{c_0}{\lambda_0}.$$

Für die vom Objekt reflektierte Dopplerfrequenz $f_d(t)$ gilt

$$f_d(t) = \frac{2\,v(t)}{\lambda_0} \qquad \text{[11, S. 97].}$$

Der Faktor 2 resultiert aus dem Hin- und Rückweg des Lichtes. Da $v(t) \ll c_0$, wird die Wellenlängenänderung vernachlässigt und λ_0 als konstant angenommen, die Bewegung wird im Folgenden über $f_d(t)$ beschrieben.

Die Interferenz beider Strahlen liefert eine Modulations- bzw. Schwebungsfrequenz f_m, die betragsmäßig der Dopplerfrequenz entspricht und deren Vorzeichen die Bewegungsrichtung angibt:

$$f_m(t) = \begin{cases} +\,f_d(t), & \text{Bewegung vom Detektor weg,} \\ -\,f_d(t), & \text{Bewegung zum Detektor hin.} \end{cases}$$

Aus der gemessenen Dopplerfrequenz folgt unmittelbar die Oberflächengeschwindigkeit

$$v(t) = \frac{\lambda_0}{2}\, f_d(t),$$

und durch Integration die Wegänderung

$$s(t) = \frac{\lambda_0}{2} \int_0^t f_d(\tau)\,\mathrm{d}\tau.$$

Damit lässt sich die Bewegung des Messobjekts vollständig aus dem Frequenzverlauf rekonstruieren. [11]

In der praktischen Anwendung muss bei LDV-Messungen darauf geachtet werden, dass die Messpunkte den Laserstrahl möglichst vollständig in den Detektor zurückreflektieren. Bei großen Neigungswinkelunterschieden zwischen den reflektierenden Messpunkten kann dies problematisch sein, sodass eine vollständige Aufnahme des gesamten Chips nicht bei

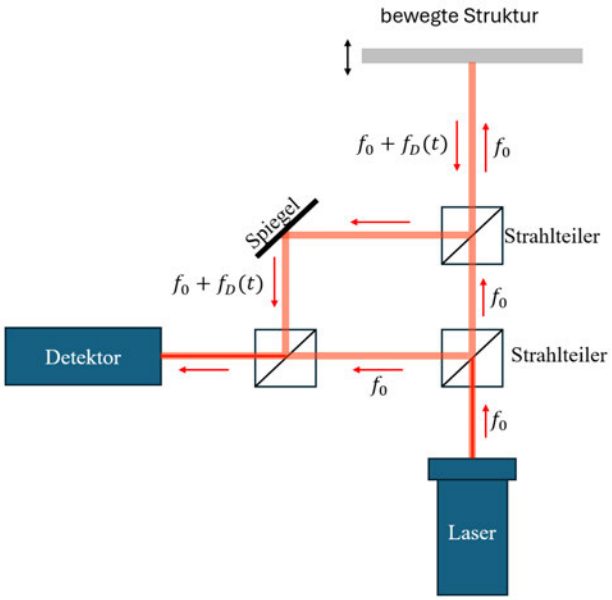

Abbildung 2.3: Interferometrisches Messprinzip eines Laser-Doppler-Vibrometers angelegt an [13, S.4]

jedem Exemplar möglich ist, da die Ruhelage der Aktuatoren von der Orthogonalen abweicht. Diese Limitierung hängt zudem stark vom verwendeten Objektiv ab, welche sich im Strahlgang vor der bewegten Struktur befindet.

Bereits leichte Verschmutzungen oder geringe Fehlstellungen der Biegebalken (Aktuatoren) können dazu führen, dass ein Chip nicht mehr zuverlässig messbar ist, da der Laser nicht zum Detektor gelangt. Beim Polytec MSA-500 kann der Chip mithilfe eines ausrichtbaren Messtisches so positioniert werden, dass alle Messpunkte ausreichend reflektieren.

2.6 Grundlagen des Feder-Masse-Dämpfer-Systems

Ein Feder-Masse-Dämpfer-System lässt sich durch die lineare Bewegungsgleichung

$$m\ddot{x} + c\dot{x} + kx = 0 \tag{2.5}$$

beschreiben. x steht hier für die Auslenkung, m für die Masse, k für die Federkonstante und c für den Dämpfungskoeffizienten, der eine dem Bewegungszustand entgegengesetzte Kraft proportional zur Geschwindigkeit beschreibt. Die Stärke der Dämpfung kann auch über die Dämpfungskonstante λ angegeben werden, welche die exponentielle Abklingrate der Schwingung charakterisiert und durch

$$\lambda = \frac{c}{2m} \qquad \text{(vgl. [14])} \tag{2.6}$$

mit dem Dämpfungskoeffizienten verknüpft ist.[14]

Für unterdämpfte Systeme ergibt sich die Bewegung als gedämpfte Schwingung:

$$x(t) = A\, e^{-\lambda t} \cos(\omega_d t + \phi) \tag{2.7}$$

wobei $\omega_0 = \sqrt{\frac{k}{m}}$ die ungedämpfte Eigenfrequenz ist [28, S.44]. Da ϕ lediglich von den Anfangsbedingungen abhängt und die Systemparameter nicht beeinflusst, wird es in der weiteren Betrachtung vernachlässigt.

2.7 MEMS Spiegel

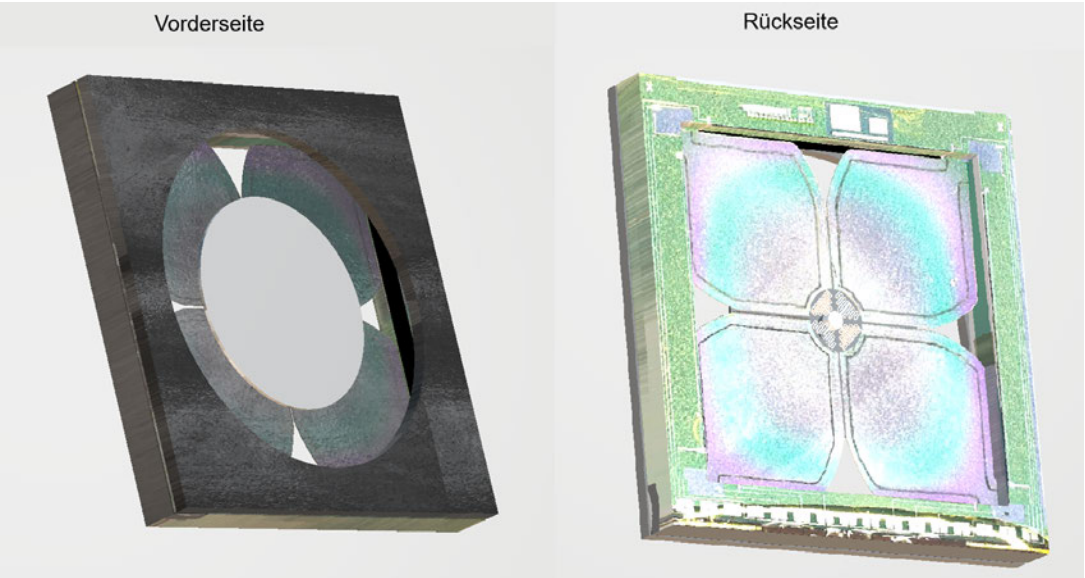

Abbildung 2.4: Dreidimensionale Darstellung der MEMS-Spiegel

Die verwendeten Chips haben die Außenmaße 10,7 × 11 mm und besitzen einen runden Spiegel mit einem Durchmesser von 5 mm. In Abbildung 2.4 ist die runde Spiegelplatte auf der Vorderseite zu sehen, auf der Rückseite sind die Federsysteme sowie die Sensor- und Aktorflächen zu erkennen. Die Chips verwenden Aluminium-Scandium-Nitrid (AlScN) als piezoelektrisches Material. Als Top- und Bottom-Elektroden, welche das piezoelektrische Material mit einem elektrischen Feld beaufschlagen, wird Molybdän eingesetzt. Das Trägermaterial besteht aus epitaktisch abgeschiedenem polykristallinem Silizium (Epi-Si) und bildet mit 25 µm dicke den Hauptanteil des Aktuatorquerschnitts.

Die vier Aktuatoren sind über ein Federsystem mit einer Säule verbunden, wie in Abbildung 2.5 dargestellt. An dieser Säule ist der eigentliche Spiegel befestigt. Das Federsystem dient dazu, den mechanischen Stress zwischen Aktuatoren und Säule zu reduzieren und gleichzeitig größere Auslenkungen zu ermöglichen. Die Federn sind in Abbildung 2.6 dargestellt, sie bestehen aus mäanderförmigem Epi-Si. Die Federsteifigkeit kann durch die Anzahl der Mäander verändert werden. In dieser Arbeit werden nur Spiegel verwendet, die im vergleich zu ähnlichen Proben eine geringe Federkonstante aufweisen.

Der 5 mm große Spiegel ist mit Glas Fritte auf der zentralen Säule des Chips befestigt. Die aufgeklebte Spiegelplatte befindet sich auf der in Abbildung 2.5 gezeigten Oberseite.

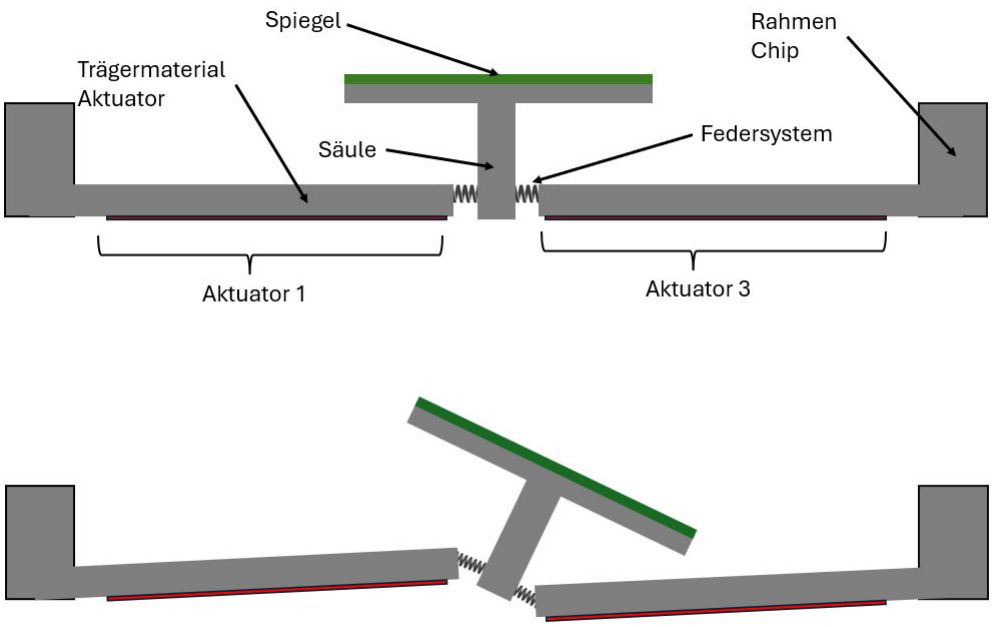

Abbildung 2.5: Schematischer querschnitts Darstellung des MEMS-Chips im Ruhezustand (oben) und im ausgelenkten Zustand (unten). Die Bewegung der reflektiven Fläche erfolgt über die Aktuatoren und das integrierte Federsystem

Bewegen sich die Aktuatoren (Biegebalken), wird die Säule ausgelenkt und damit auch der daran befestigte Spiegel, wie in Abbildung 2.5 schematisch dargestellt. Die Bewegung in Abbildung 2.5 wird durch das Ausdehnen des piezoelektrischen Materials von Aktuator 1 und das Stauchen des piezoelektrischen Materials von Aktuator 2 hervorgerufen. Dies wird durch die Ansteuerung der Aktuatoren im Gegentakt, also mit unterschiedlicher Polarität erreicht.

Der Spiegel besteht aus vier Aktuatoren, die gegenüberliegend angeordnet sind. Das ursprüngliche Design enthält pro Aktuator eine einzelne piezoelektrische Fläche. In der weiterentwickelten Version des Chips besitzt jeder Aktuator zwei elektrisch voneinander isolierte piezoelektrische Flächen. Dabei befindet sich eine kleinere Fläche am Rand des Aktuators und eine größere Fläche in der Mitte.

In Abbildung 2.6 wird die große Fläche als Aktuator und die kleine Fläche als Sensor bezeichnet. Es ist prinzipiell auch möglich, die Sensorfläche zum Antrieb zu verwenden,

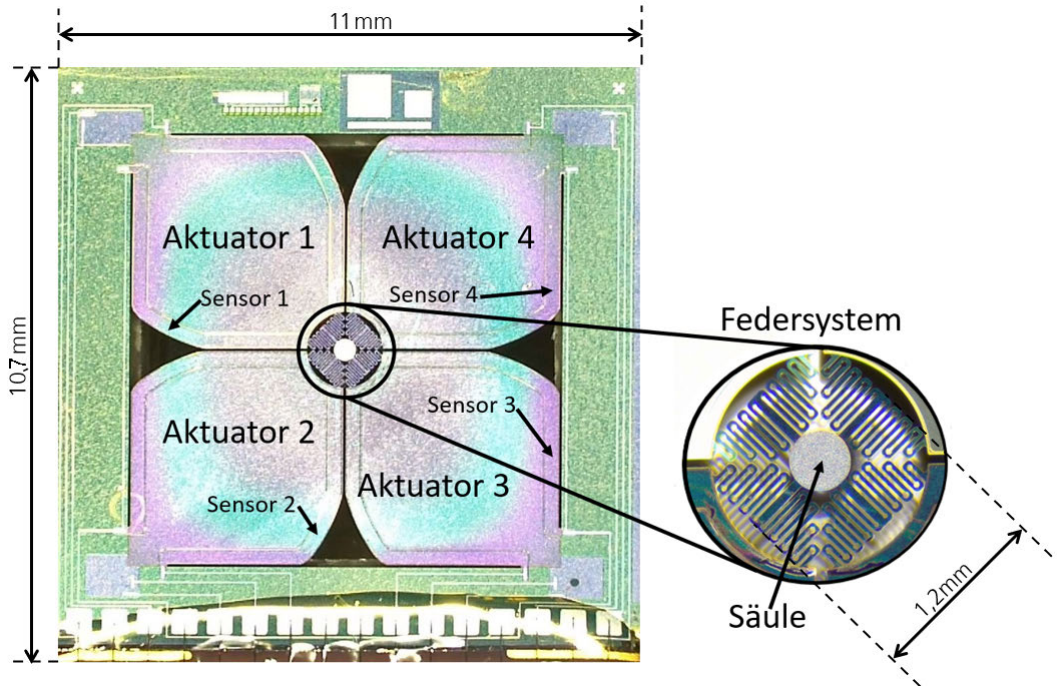

Abbildung 2.6: Mikroskopbilder der verwendeten MEMS-Spiegel

was jedoch eine geringere quasistatische Auslenkung zur Folge hat. Für die Messung der Aktuatorstellung wird daher die kleinere Fläche als Sensor genutzt, während zum Auslenken (Antrieb) die größere Fläche verwendet wird.

2.7.1 Aktuator

In dieser Arbeit werden monomorphe Aktuatoren verwendet, die überwiegend aus dem Trägermaterial polykristallinem Silizium bestehen. Als monomorphe Aktuatoren werden solche Aktuatoren bezeichnet, die ein piezoelektrisches Material auf einem nichtpiezoelektrischen Trägermaterial aufgebracht ist. Die resultierende Bewegung geht somit ausschließlich von einem Material aus. [3]

Die Bewegung wird durch den piezoelektrischen Werkstoff Aluminiumscandiumnitrid (AlScN) erzeugt, dieser ist als Dünnfilm auf dem Trägermaterial abgeschieden. AlScN stellt eine Weiterentwicklung von AlN dar, die darauf abzielt, mittels des indirekten piezoelektrischen Effektes mehr elektrische Energie in mechanische Energie umzuwandeln. Durch die Beimischung von Scandium wird die Leistungsfähigkeit des Materials für MEMS-

und Hochfrequenzanwendungen deutlich gesteigert. Diese Eigenschaften machen AlScN zu einer vielversprechenden Alternative zu anderen piezoelektrischen Materialien wie Blei-Zirkonat-Titanat (PZT) und Aluminiumnitrid (AlN). [15]

In der Mitte des Aktuators befindet sich der piezo-aktive Teil der zum treiben des Spiegels verwendet wird. In Abbildung 2.6 die Aktuator Flächen als Aktuator 1,2,3,4 gekennzeichnet. Die treibenden Flächen betragen etwa 80 % der gesamten Aktuatorfläche, was etwa 3600 µm^2 sind, in Abbildung 2.7 ist die die Fläche blau illustriert.

2.7.2 Piezoelektrischer Sensor

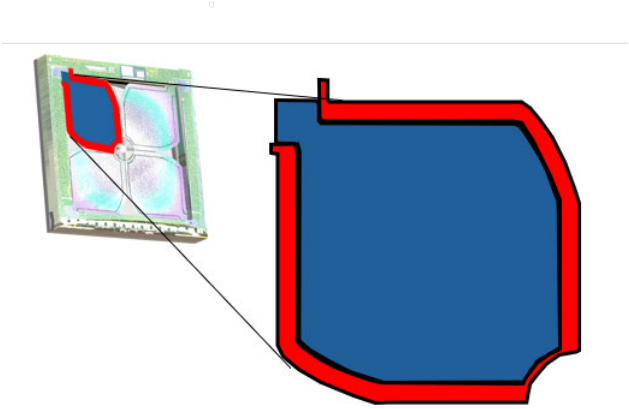

Abbildung 2.7: Zuordnung der piezoelektrischen Flächen, in blau die Treiber und rot die Sensorfläche

Die Sensorelemente sind piezoelektrische Flächen, die um die treibenden Flächen herum angeordnet sind. In Abbildung 2.7 ist das Sensorelement für den Aktuator 1 rot eingefärbte dargestellt. Sie bestehen aus AlScN und haben die gleiche Schichtdicke wie die Flächen die zum Treiben verwendet werden, da sie im gleichen Fertigungsschritt aufgebracht werden.

Mit den Sensorelementen soll die mechanische Auslenkung des Spiegels gemessen werden, unabhängig davon, ob diese durch den Antrieb der Aktuatoren oder durch äußere Einwirkungen wie z. B. mechanische Erschütterungen verursacht wird.

Die von einem Sensorelement abgegebene Spannung hängt sowohl von der Geschwindigkeit der Verformung als auch vom Ausmaß der Kristallgitterdeformation ab. Theoretisch bleibt die erzeugte Ladung erhalten, praktisch fließt jedoch durch den Innenwiderstand

des Messinstruments ein Teil der Ladung ab. Die Sensorfläche ist in Abbildung 2.6 mit *Sensor* gekennzeichnet und beträgt etwa 20 % der gesamten Aktuatorfläche, was ungefähr 400 µm^2 entspricht.

2.7.3 Spiegelplatte

Zur Reflexion wurde eine Spiegelplatte aus Epi-Silizium verwendet, auf der eine Aluminiumschicht abgeschieden wurde, ähnlich der in [19] beschriebenen Konstruktion. Die für diesen Aufbau verwendeten Spiegelplatten wurden jedoch in einem gesonderten Fertigungsprozess hergestellt und anschließend auf die Säule geklebt.

Die Spiegel sind in verschiedenen Ausführungen erhältlich, je nach zu reflektierender Wellenlänge des Lichts werden Gold, Aluminium oder andere hochreflektierende Materialien eingesetzt. Für diese Arbeit werden Spiegel mit einer Aluminiumschicht verwendet, da ein sichtbarer Laserstrahl für die Experimente reflektiert werden soll.

Die Spiegelplatten sind mit Durchmessern von 2 mm, 3 mm oder 5 mm erhältlich. In dieser Arbeit wurden Spiegelplatten mit einem Durchmesser von 5 mm eingesetzt, da die größere Masse zu einer niedrigeren Resonanzfrequenz des Chips führt, wie in Gleichung 2.1 dargestellt. Diese niedrigere Resonanzfrequenz begünstigt das Aufschwingen des Systems bei Erschütterungen mit kurzer Anstiegszeit.

3 Charakterisierung der MEMS Spiegel

3.1 Statische Auslenkung

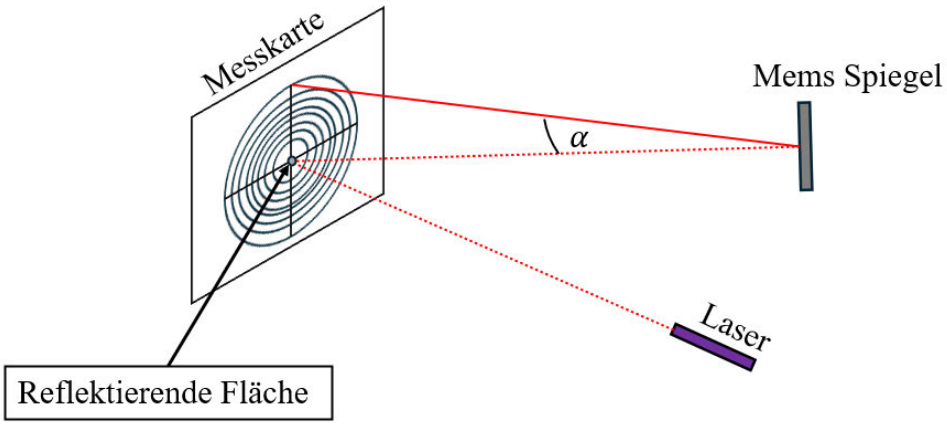

Abbildung 3.1: Messaufbau der TOSA-Messung der MEMS-Spiegel

Die Spiegel sollen quasistatisch betrieben werden, das heißt, sie werden mit Spannung angesteuert und können an jeder angefahrenen Auslenkung theoretisch beliebig lange verharren. In Abbildung 3.2 sind die Spiegel mit Gleichspannungen im Bereich von $-50\,\mathrm{V}$ bis $+50\,\mathrm{V}$ angeregt, und der resultierende Winkel wurde mit einer Auflösung von 0,05° ermittelt. Zur Bestimmung des Auslenkwinkels wurde eine Messkarte aus Papier verwendet, die in der Mitte ein Loch für die reflektierende Fläche aufweist. In Abbildung 3.1 sind Messkarte und Messaufbau schematisch dargestellt. Die Messkarte enthält konzentrische Kreise, deren Radien jeweils um 1,5 mm zunehmen. Die Winkelauflösung der einzelnen Ringe wird durch den Abstand des MEMS-Spiegels zur Messkarte bestimmt. Im Folgenden wird von einer Auflösung von 0,05° pro 1,5 mm Radius ausgegangen.

$$\tan(\alpha) = \frac{a}{b} \tag{3.1}$$

$$b = \frac{a}{\tan(\alpha)} \tag{3.2}$$

$$b = \frac{1{,}5\,\mathrm{mm}}{\tan(0{,}05°)} \tag{3.3}$$

$$b \approx 1{,}718\,\mathrm{m} \tag{3.4}$$

Mit einer Messkarte, die eine Auflösung von 0,05° pro Millimeter aufweist, lässt sich die Entfernung mithilfe des Tangens berechnen. Die Berechnung über Gleichung 3.1 stellt einen einfachen Weg dar, die Entfernung des Spiegels zur Messkarte zu bestimmen. Nichtlinearitäten, die dadurch entstehen, dass die Messkarte eine flache Fläche besitzt und nicht sphärisch ausgeführt ist, können vernachlässigt werden, da die maximale Auslenkung der Spiegel mit etwa 2,1° sehr klein ist. Es wird weiterhin angenommen, dass der Laser exakt senkrecht aus der Karte austritt. Der MEMS-Spiegel wird während des Versuchs möglichst parallel zur Messkarte ausgerichtet. Anschließend wird der reflektierte Laser manuell auf die Mitte der Messkarte eingestellt, sodass der MEMS-Spiegel in Ruhelage den Laser genau in die Mitte reflektiert. Um die maximale Auslenkung des MEMS-Spiegels zu erreichen, werden für die Y-Achse die Aktuatoren 1 und 3 aus Abbildung 2.6 und für die X-Achse die Aktuatoren 2 und 4 verwendet. Der Spiegel wird dabei um 45° nach rechts gedreht betrieben. Aktuator 3 wird mit umgekehrter Polarität zu Aktuator 1 angesteuert, sodass beispielsweise Aktuator 1 sich nach vorne bewegt, während Aktuator 3 gleichzeitig mit der gleichen Kraft nach hinten arbeitet.

In Abbildung 3.2 sind repräsentativ die quasistatischen Auslenkungen von vier Spiegeln dargestellt. Es ist ein lineares Verhältnis zwischen der Aktuatorspannung und der Auslenkung erkennbar. Dieses Verhalten ist für AlScN charakteristisch. Zwar zeigt AlScN grundsätzlich ein Hystereseverhalten, dieses tritt jedoch erst bei Spannungen auf, die in diesem Aufbau bereits zum elektrischen Durchbruch führen würden.[9]

Aufgrund dieses gut linearen Verhaltens lassen sich die Laserablenkungskoordinaten einfach durch die Spannungswerte beschreiben. Zusätzlich zeigen Spiegel mit schlechterer Total optical scan angle (TOSA) ein ähnliches Verhalten auf beiden Achsen. Die unterschiedlichen TOSA-Werte zwischen den getesteten Spiegeln deuten auf Fertigungstoleranzen hin, wie sie typischerweise bei MEMS-Prozessen auftreten (z. B. Rand- oder

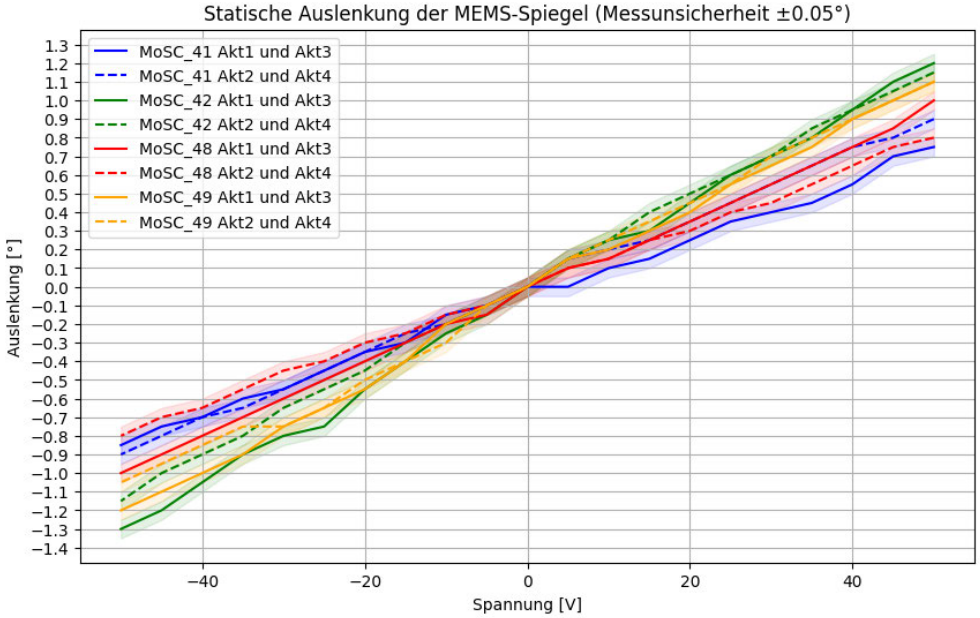

Abbildung 3.2: Statische Auslenkung der MEMS-Spiegel

Zoneneffekte auf dem Wafer). Eine systematische Analyse dieser Effekte wurde im Rahmen dieser Arbeit jedoch nicht durchgeführt und ist daher nicht Bestandteil der weiteren Betrachtungen.

3.2 Auswertung der Sensorspannung am Aktor

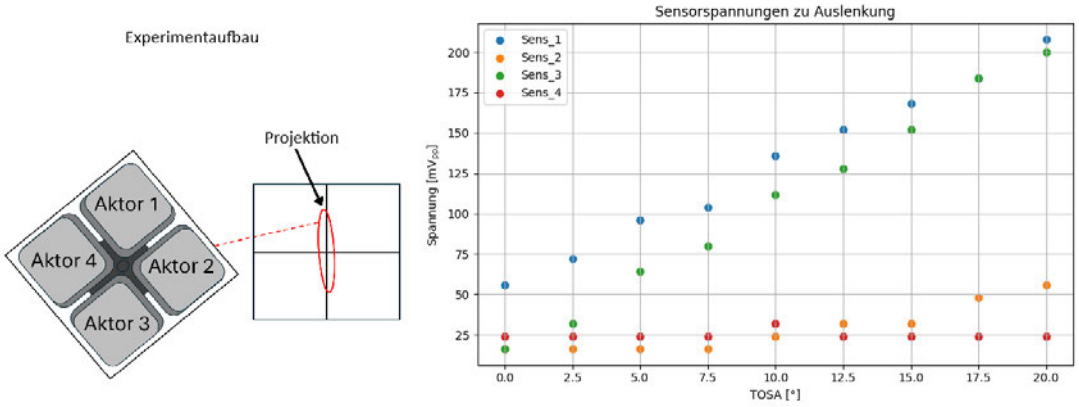

Abbildung 3.3: Sensorspannung in Abhängigkeit von der Auslenkung.

Die Sensorspannung wird genutzt, um ein Signal zu erhalten, aus dem die Bewegung des Spiegels rekonstruiert werden kann. Damit die Sensorspannung zuverlässig ausgewertet werden kann, müssen die maximale Amplitude und die korrekte Zuordnung gewährleistet sein. In Abbildung 3.3 sind die Spitzen-zu-Spitzen-Spannungen der vier Sensoren bei verschiedenen Auslenkungen dargestellt. Die Messungen stammen vom Spiegel MoSc_42, der über Aktuator 1 in Resonanz angeregt wurde. Das projizierte Bild ergibt eine Ellipse, deren Hauptachse auf der vertikalen Achse des Projektionsschirms liegt. Der Spiegel ist so angeordnet, dass Sensor 1 und 3 die vertikale Achse und Sensor 2 und 4 die horizontale Achse darstellen. Die Messwerte in Abbildung 3.3 zeigen eine erhöhte Spannung an Sensor 1 und 3, was zur projizierten Ellipse passt. Über den gesamten Messbereich ist bei Sensor 1 eine größere Sensorspannung zu erkennen, was durch eine stärkere Verformung des piezoelektrischen Sensors hervorgerufen wird, da Aktuator 1 als einziger das System treibt und damit die größte Bewegung verursacht. Für die weitere Auswertung ist jedoch ausschließlich das lineare Verhalten zwischen Auslenkwinkel und Sensorspannung relevant, sodass die Sensorspannungen als Rückmeldesignal für die Spiegelauslenkung verwendet werden können.

3.3 Schockmessung

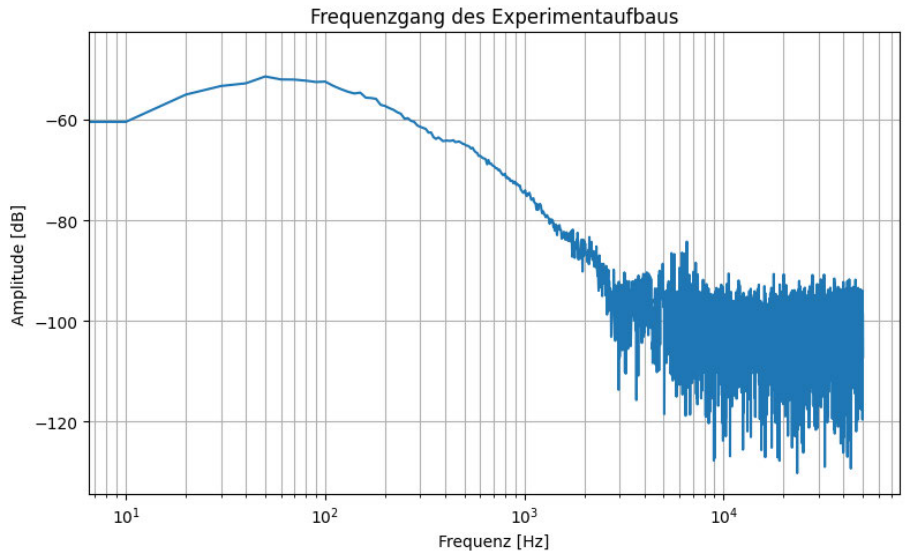

Abbildung 3.4: Frequenzanalyse des eingebrachten Schocks mit montierter Spiegelhalterung.

Ein zentraler Bestandteil dieser Arbeit ist die Reduktion des schwingenden Spiegelverhaltens bei mechanischen Erschütterungen. Grundlage für die Entwicklung geeigneter Gegenmaßnahmen ist die detaillierte Charakterisierung der Spiegelantwort unter Stoßanregung. Für die niederfrequente Rechteckanregung wurde der Hoch-g-Erregerkopf Typ 4811 in Kombination mit dem Exciter Body Typ 4801 und dem Leistungsverstärker Typ 2707 verwendet. Die Beschleunigungsmessung erfolgte mit dem miniaturisierten Beschleunigungssensor Typ 4344 der Fa. Brüel & Kjær, der lediglich 2 g wiegt. Der Hoch-g-Erregerkopf wurde außerhalb der üblichen Spezifikation als vereinfachter Schocktisch betrieben, indem bei niedriger Frequenz eine harte Rechteckanregung aufgebracht wurde. Für alle Messungen wurde der Hoch-g-Erregerkopf mit einem 1 Hz Pulssignal betrieben, das einen Duty Cycle von 20 % aufweist. Die Stoßkraft wurde über die Amplitude des Pulssignals gesteuert. Laut Datenblatt beträgt die maximale Beschleunigung bei 10 Hz 3 g. Für den Versuchsaufbau wurde die maximale Schockeinwirkung auf 24 $\frac{\mathrm{m}}{\mathrm{s}^2}$ (ca. 2,45 g) begrenzt, um Beschädigungen der mechanischen Anschläge des Hoch-g-Erregerkopfs zu vermeiden [4]. Die Spiegel besitzen Resonanzfrequenzen bei etwa 270 Hz und 466 Hz. Um sicherzustellen, dass diese Frequenzen auch angeregt werden, wurde eine Frequenzanalyse der Shockeinwirkung durchgeführt. In Abbildung 3.4 ist zu erkennen, dass Frequenzen bis

500 Hz noch vorhanden sind. Ab etwa 1300 Hz nehmen die Spektralanteile stark ab und verschwinden im Rauschteppich. Für die Charakterisierung wurden die beiden orthogonalen Extremfälle der Schockeinwirkung untersucht: einmal die Krafteinwirkung senkrecht zur Spiegelebene und einmal parallel zur Spiegelebene. Diese Konfigurationen erlauben es, die unterschiedlichen Schwingungsmoden, Hub- und Kippmoden, gezielt anzuregen und zu analysieren. Beliebige Zwischenwinkel lassen sich als Linearkombinationen dieser Grundorientierungen darstellen und wurden daher nicht gesondert betrachtet.

3.3.1 Schockmessung mit liegendem MEMS Spiegel

Bei der Schockmessung in der liegenden Orientierung ist der Spiegel des Chips nach oben gerichtet, so dass die Erdanziehungskraft und die Schockeinwirkung orthogonal zur Spiegelfläche wirken. In Abbildung 3.5 ist zu erkennen, dass die Sensorspannung mit zunehmender Stoßkraft proportional ansteigt. Bei einem Stoß von etwa 23,5 $\frac{m}{s^2}$, entsprechend ca. 2,4 g, liefern die Sensoren eine Spitzen-Spitzen-Spannung von 30 mV. Diese ±15 mV werden im weiteren Verlauf als maximale Spannung für die Messmimik verwendet.

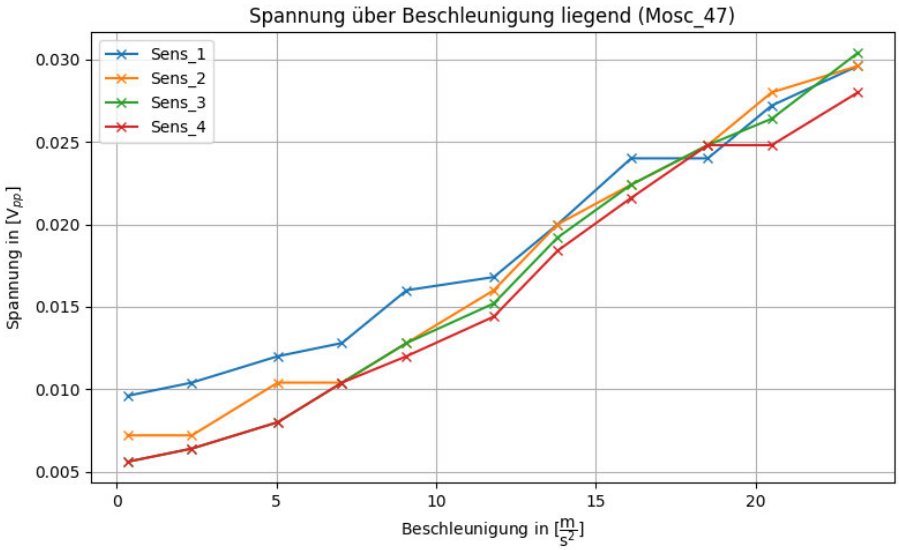

Abbildung 3.5: Gemessenes Sensorsignal in Abhängigkeit von der Stoßkraft bei liegendem Chip.

Abbildung 3.6 zeigt die Signale von Sensor 1 sowohl im Zeitbereich als auch im Frequenzbereich. Beim Anregen des Systems mit einem Stoß wird das System, wie in Kapitel 2.2

beschrieben, mit nahezu allen Frequenzen angeregt. Bei Schockeinwirkung in liegender Position ist ein deutlicher Frequenzanteil bei 466 Hz zu erkennen. Die 50 Hz in den Sensordaten stammen vom Versorgungsnetz und wurden in das Experiment eingeschleppt. Diese Störung wird hier nicht weiter betrachtet, da für den Versuchsaufbau lange Messleitungen erforderlich sind. Darüber hinaus sind schwache Signale bei etwa 268 Hz erkennbar. Frühere Messungen zeigen, dass die Resonanzfrequenzen der Spiegel bei 268 Hz liegen. Da in dieser Messung die Kraft orthogonal zur Spiegelplatte eingebracht wurde, deuten die Daten in Abbildung 3.6 darauf hin, dass hauptsächlich die Hubmode angeregt wird.

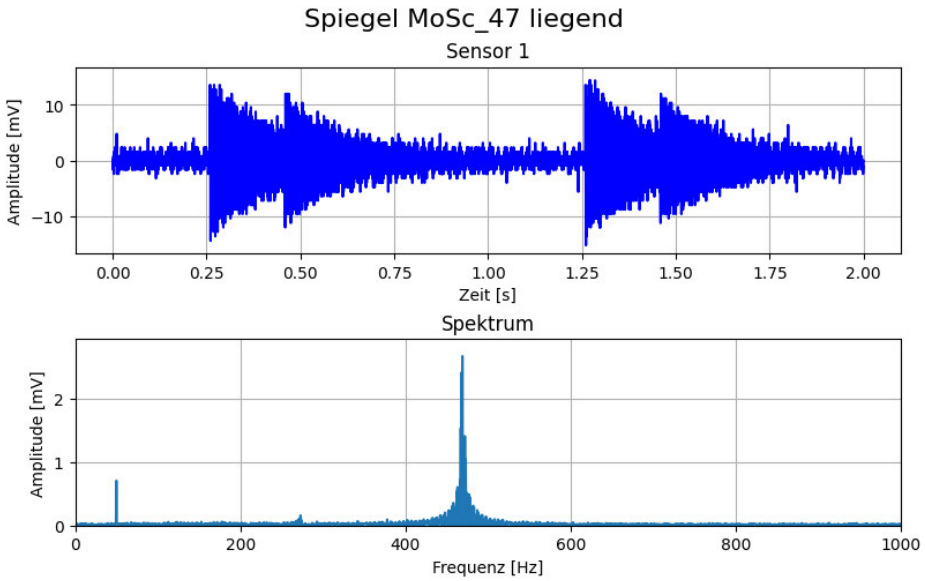

Abbildung 3.6: Gemessene Sensorsignale in Zeitbereich und Frequenzbereich bei liegendem Chip.

3.3.2 Schockmessung mit stehendem MEMS Spiegel

Die Schockmessung in stehender Position entspricht am ehesten einem möglichen Einsatz in einem Lidar-System, wie es in einem Fahrzeug verbaut wäre. In Abbildung 3.7 ist zu erkennen, dass die Sensoren erst bei einer Beschleunigung von 10 $\frac{\mathrm{m}}{\mathrm{s}^2}$ eine nennenswerte Reaktion zeigen. Die Spannung bei einer Beschleunigung von 23,5 $\frac{\mathrm{m}}{\mathrm{s}^2}$ liegt in dieser Anordnung des Spiegels bei etwa ±7 mV. Darüber hinaus zeigt sich, dass Sensor 1 und 3 höhere Spannungen aufweisen, da die entsprechenden Aktuatoren parallel zum Kraftstoßvektor angeordnet sind. Die Sensoren an den Aktuatoren 2 und 4, die die horizontale Achse repräsentieren, zeigen dagegen kleinere Spannungen. Dies ist darauf zurückzuführen, dass der Spiegel wie eine Masse am Ende der Säule wirkt und die Aktuatoren 1 und 3 dadurch stärker ausgelenkt werden. Der Rückgang der Spannung im Frequenzbereich bei 466 Hz und der gleichzeitige Anstieg bei 268 Hz lassen darauf schließen, dass bei 268 Hz die Kippmode dominant angeregt wird.

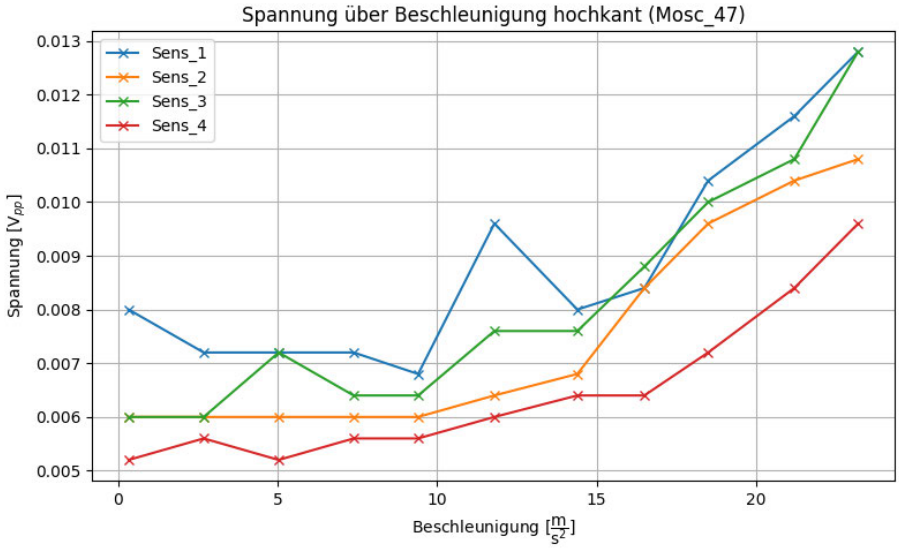

Abbildung 3.7: Gemessenes Sensorsignal in Abhängigkeit von der Stoßkraft bei stehendem Chip.

In Abbildung 3.8 sind die Sensordaten von Kanal 1 in stehender Position dargestellt. Es ist zu erkennen, dass die Signale deutlich kleiner ausfallen als bei der liegenden Stoßanregung. So ist das Signal bei 466 Hz von 13 mV auf 2 mV abgefallen, während das

Signal bei 268 Hz von 1 mV auf 3 mV angestiegen ist. Die in den Messdaten auftretende 50-Hz-Komponente wird in der weiteren Analyse nicht berücksichtigt.

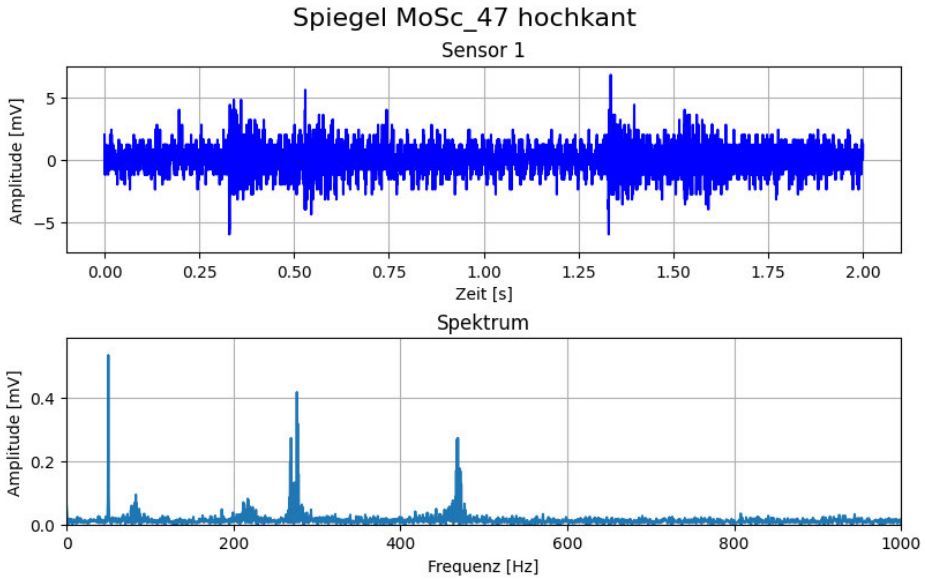

Abbildung 3.8: Gemessenes Sensorsignale in Zeitbereich und Frequenzbereich bei stehendem Chip.

3.3.3 LDV Messung an signifikanten Frequenzen

In den Kapiteln 3.3.2 und 3.3.1 wurden die Sensoren der MEMS-Spiegel bei Schockeinwirkung untersucht. Um die Schwingungsmoden der Spiegel zu bestimmen, wurde zusätzlich eine LDV-Messung durchgeführt. Dabei wurde der Spiegel ausschließlich über Aktuator 1 elektrisch angeregt, und die resultierende Schwingung mit dem LDV gemessen. Die LDV-Messung erfolgte bei den Frequenzen 268 Hz und 466 Hz, da diese Frequenzen in den Schockmessungen relevant aufgefallen sind. In Abbildung 3.9 sind die Animationen der LDV-Messung anhand jeweils dreier Einzelbilder dargestellt. Die Zahlen in den unteren linken Ecken der Bilder geben die fortschreitende Zeit an. Die Federstruktur des MEMS-Spiegels ist im Hintergrund gräulich erkennbar. Um die Auslenkung des Spiegels sichtbar zu machen, wurden die Messpunkte des LDVs entlang der Kippachsen des Spiegels platziert. Da die Spiegelplatte mit einem Durchmesser von 5 mm die Sicht auf die Federstruktur verdeckt, wurden die Messpunkte auf der Rückseite der Säule platziert.

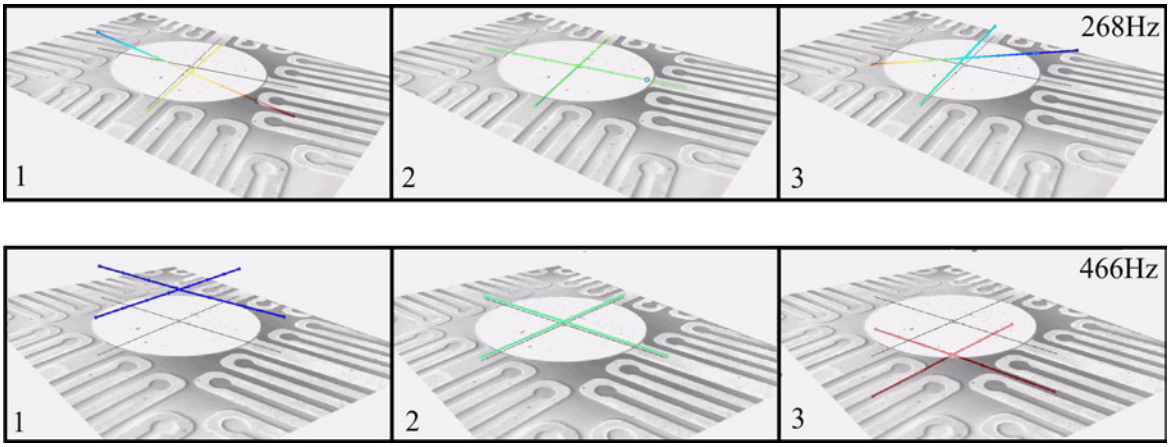

Abbildung 3.9: LDV Messung bei 268 Hz, Anregung über Aktuator 1.

Durch die feste Verbindung zwischen Säule und Spiegelplatte entspricht die gemessene Auslenkung an dieser Position der tatsächlichen Auslenkung der Spiegelplatte. Die oberen drei Bilder zeigen die Messung bei 268 Hz. Hier ist eine Kippbewegung der Messpunkte zu erkennen, was darauf hinweist, dass bei einer Schockeinwirkung in stehender Position die Kippmode des Spiegels stärker angeregt wird als die Hubmode. Die unteren drei Bilder von Abbildung 3.9 zeigen die LDV-Messungen bei 466 Hz. Bei dieser Frequenz ist eine Hubbewegung des Spiegels sichtbar. In einem System, in dem der Laser direkt auf den Spiegel trifft, hat diese Bewegung keine Auswirkungen auf die Auslenkung. Da der Spiegel jedoch mit Glasfritte an der Säule befestigt ist, was zu Asymmetrien führen kann, muss dieses Signal ebenfalls berücksichtigt werden.

3.4 Analyse des Abklingverhaltens

Für die Systemanalyse kann die Dämpfungskonstante über das Abklingverhalten des Spiegels ermittelt werden. Die Charakterisierung der Störeinwirkung ist für eine spätere Kompensation von entscheidender Bedeutung. Für die Analyse der Dämpfungskonstante wurde ein kleiner Abschnitt aus den Daten von Abbildung 3.6 ausgeschnitten. Dieser Ausschnitt beinhaltet ausschließlich das Abklingverhalten eines Spiegels bei Stoßanregung. Die Abklingkurve lässt sich durch die folgende Formel beschreiben:

$$x(t) = A_0 e^{-\lambda t} \sin(\omega t) \qquad \text{vgl. [10, S.196]} \tag{3.5}$$

bezeichnet werden. A_0 steht für die Anfangsamplitude der Schwingung, ω für die Kreisfrequenz und λ für die Dämpfungskonstante, welche die Abnahme der Schwingungsamplitude beschreibt. Die Kreisfrequenz ω entspricht in diesem Experiment etwa 466 Hz. In der Realität treten mehrere Frequenzanteile auf, die jedoch so gering sind, dass sie in Formel 3.5 vernachlässigt werden können. Mit der Funktion `find_peaks()` aus dem Python-Modul `scipy.signal` wurden die positiven Peaks der Schwingung bestimmt, sodass nur die obere Einhüllende für das Fitting verwendet wird. Auf dieser Grundlage kann Formel 3.5 vereinfacht werden zu:

$$\hat{x}(t) = A_0 e^{-\lambda t} \tag{3.6}$$

Der Ausdruck aus Formel 3.6 wird in die Python-Funktion 3.1 übertragen. Diese Funktion wird anschließend mithilfe der Funktion `curve_fit` aus dem Modul `scipy.optimize` an die experimentellen Daten angepasst. Dabei werden die Parameter A_0 und λ als Fitgrößen bestimmt, um die Anfangsamplitude und die Dämpfungskonstante des Schwingungssignals zu ermitteln.

```
1 # Exponentialfunktion definieren
2 def exp_func(t, A, lambda_):
3     out = A * np.exp(-lambda_ * (t - t_peaks[0]))
4     return out
```

Listing 3.1: Exponentialfunktion zur Modellierung des Abklingverhaltens

Die Daten, die in liegender Position aufgenommen wurden, zeigen ein sehr gutes Fit-Ergebnis. Dieses gute Ergebnis ist auf die hohe Amplitude und den damit verbundenen großen Rauschabstand zurückzuführen. Der Fit der Daten aus Abbildung 3.10 ergab eine Dämpfungskonstante von $\lambda = 5{,}350\,\frac{1}{\mathrm{s}}$. Mit diesem Wert kann der mechanische Dämpfungskoeffizient c berechnet werden. Für die Berechnung von c wurde die Masse des Spiegels geometrisch bestimmt. Dabei wird angenommen, dass nur die Säule und die Spiegelplatte das Volumen der beweglichen Masse ausmachen. Die Masse lässt sich somit vereinfacht durch zwei Zylinder und die Dichte von Silizium berechnen. Es wird der Einfachheit halber davon ausgegangen, dass die Masse vollständig aus Silizium ($\rho = 2328\,\mathrm{kg/m^3}$) [10, S.315] besteht und die Aluminiumschicht keinen signifikanten Einfluss auf die Masse hat.

$$
\begin{aligned}
V_{\text{Säule}} &= \pi \cdot r_{\text{Säule}}^2 \cdot h_{\text{Säule}} \quad &&= \pi \cdot (100\,\mu\text{m})^2 \cdot 500\,\mu\text{m} \\
V_{\text{Spiegelplatte}} &= \pi \cdot r_{\text{Spiegelplatte}}^2 \cdot h_{\text{Spiegelplatte}} \quad &&= \pi \cdot (2.5\,\text{mm})^2 \cdot 80\,\mu\text{m} \\
m &= (V_{\text{Spiegelplatte}} + V_{\text{Säule}}) \cdot \rho \quad &&\approx (1{,}571 \cdot 10^{-9}\,\text{m}^3 + 15{,}71 \cdot 10^{-12}\,\text{m}^3) \cdot 2328\,\text{kg/m}^3 \\
m &\approx 3{,}657\,\text{mg} \quad &&\approx 3{,}657 \cdot 10^{-6}\,\text{kg}
\end{aligned}
$$

Mit der ermittelten Masse und der Dämpfungskonstante kann nun der Dämpfungskoeffizient mit folgender Formel festgestellt werden.

$$
c = 2 \cdot m \cdot \lambda = 2 \cdot 3{,}657 \cdot 10^{-6}\,\text{kg} \cdot 5{,}350\,\frac{1}{\text{s}} = 39{,}13 \cdot 10^{-6}\,\frac{\text{kg}}{\text{s}}
$$

Für die Messung mit liegendem Spiegel ergibt sich ein mechanischer Dämpfungskoeffizient von $c = 39{,}13 \cdot 10^{-6}\,\frac{\text{kg}}{\text{s}}$. Zum Vergleich der gefitteten Werte in liegender und stehender Position kann der Gütefaktor aus den Parametern λ und ω_0 berechnet werden.

Für schwach gedämpfte Systeme lässt sich der Q-Faktor wie folgt bestimmen:

$$
\text{Gegeben:} \quad \omega_0 = \sqrt{\frac{k}{m}}, \quad Q = \frac{\sqrt{k \cdot m}}{c} \qquad \text{(vgl. [28, S.44])} \tag{3.7}
$$

Definiere die Dämpfungskonstante λ für einen gedämpften Oszillator:

$$
\lambda = \frac{c}{2m} \qquad \text{(vgl. Gl. 2.6)} \tag{3.8}
$$

Daraus folgt:

$$
c = 2m\lambda \tag{3.9}
$$

Setze c in die Definition von Q ein:

$$
Q = \frac{\sqrt{km}}{c} = \frac{\sqrt{km}}{2m\lambda} = \frac{\sqrt{k/m}}{2\lambda} \tag{3.10}
$$

Mit $\omega_0 = \sqrt{k/m}$ ergibt sich:

$$Q = \frac{\omega_0}{2\lambda} \tag{3.11}$$

Dabei bezeichnet

- $\omega_0 = 2\pi f_0$ die **Eigenkreisfrequenz**, die aus der experimentell bestimmten Eigenfrequenz $f_0 = 466\,\text{Hz}$, die aus Abbildung 3.6 erkennbar ist,
- $\lambda = 5{,}350\,\frac{1}{\text{s}}$ die aus dem exponentiellen Fit in Abbildung 3.10 ermittelte **Dämpfungskonstante**.

Daraus ergibt sich ein Gütefaktor für die Daten Schockmessung des liegenden Spiegels:

$$Q = \frac{466\,\text{Hz} \cdot 2\pi}{2 \cdot 5{,}35\,\frac{1}{\text{s}}} = 273{,}6$$

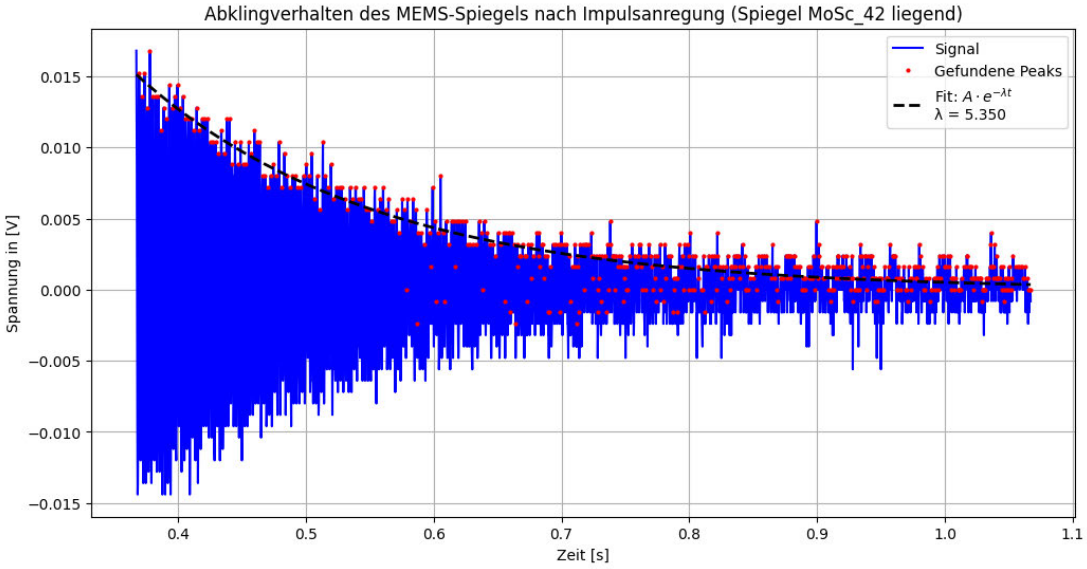

Abbildung 3.10: Abklingkurve von Spiegel MoSc_42 in liegender Position mit Fit

Bei der Schockmessung in stehender Position sind die Daten stark verrauscht, weshalb vor dem Fitting eine Filterung durchgeführt wurde. Zur Glättung des Messrauschens und zur besseren Erkennung der charakteristischen Signalverläufe wird ein Mittelwertfilter eingesetzt. Der Mittelwertfilter erstreckt sich über 7 Werte bei einer Abtastrate von

5 kHz, was einem Filter mit einer Grenzfrequenz von $f_{-3\mathrm{dB}} \approx 0{,}443 \cdot \frac{f_s}{N} \approx 316\,\mathrm{Hz}$ entspricht. Dieser wirkt als Tiefpassfilter und unterdrückt hochfrequente Störungen, ohne den langsamen Abklingverlauf des Signals wesentlich zu beeinflussen. Da alle schwingungsrelevanten Moden des MEMS-Spiegels insbesondere Kipp- und Hubmode deutlich unterhalb von 5 kHz liegen und höhere Moden aufgrund starker Dämpfung kaum zum Sensorsignal beitragen, ist diese Filterwahl vollständig ausreichend, um die für das Fitting benötigte Dynamik unverfälscht abzubilden. Die Daten wurden zudem offsetbefreit, indem der Mittelwert aller Samples berechnet und das gefilterte Signal entsprechend korrigiert wurde. Aus diesen Daten lassen sich, wie zu Beginn, die Dämpfungskonstante, der Dämpfungskoeffizient und der Q-Faktor durch Fitting bestimmen. Da die Daten stärker verrauscht sind, wurde dem maximalen Peak im Fitting-Algorithmus ein höheres Gewicht zugewiesen. Das Gewicht des maximalen Peaks wurde durch iterative Tests und visuelle Analyse des Fits festgelegt.

$$
\begin{aligned}
\lambda &= 5{,}318\,\frac{1}{\mathrm{s}} \\
c &= 2m\lambda = 2 \cdot 3{,}657 \cdot 10^{-6}\,\mathrm{kg} \cdot 5{,}318\,\frac{1}{\mathrm{s}} = 4{,}293 \cdot 10^{-6}\,\frac{\mathrm{kg}}{\mathrm{s}} \\
Q &= \frac{\omega_0}{2\lambda} = \frac{268\,\mathrm{Hz} \cdot 2\pi}{2 \cdot 5{,}318\,\frac{1}{\mathrm{s}}} = 158{,}3
\end{aligned}
$$

Die aktuellen Messungen zeigen, dass die Dämpfungskonstanten λ und die mechanischen Dämpfungskoeffizienten c in beiden Orientierungen des Spiegels nahezu identisch sind. Die Gütefaktoren Q unterscheiden sich jedoch deutlich. Dies lässt sich darauf zurückführen, dass der Gütefaktor nicht allein von der Dämpfung, sondern auch von der jeweiligen Eigenfrequenz ω_0 des Systems abhängt, wie in Gleichung 3.11 dargestellt. Bei den unterschiedlichen Stoßeinwirkungen werden verschiedene Schwingungsmoden angeregt, wodurch die dominierenden Frequenzen der abklingenden Schwingung variieren. Obwohl die absolute Dämpfung unverändert bleibt, führt die unterschiedliche Eigenfrequenz zu abweichenden Gütefaktoren. Dies bedeutet, dass das System bei gleicher Dämpfung je nach angeregter Mode unterschiedlich viel Energie pro Zyklus verliert. Die Güte des Systems ist somit nicht ausschließlich ein Maß für die Dämpfung, sondern spiegelt das Zusammenspiel von Dämpfung und Eigenfrequenz wider. Diese Erkenntnis ist insbesondere für die spätere Auslegung von Regelungsstrategien relevant, da sie zeigt, dass die

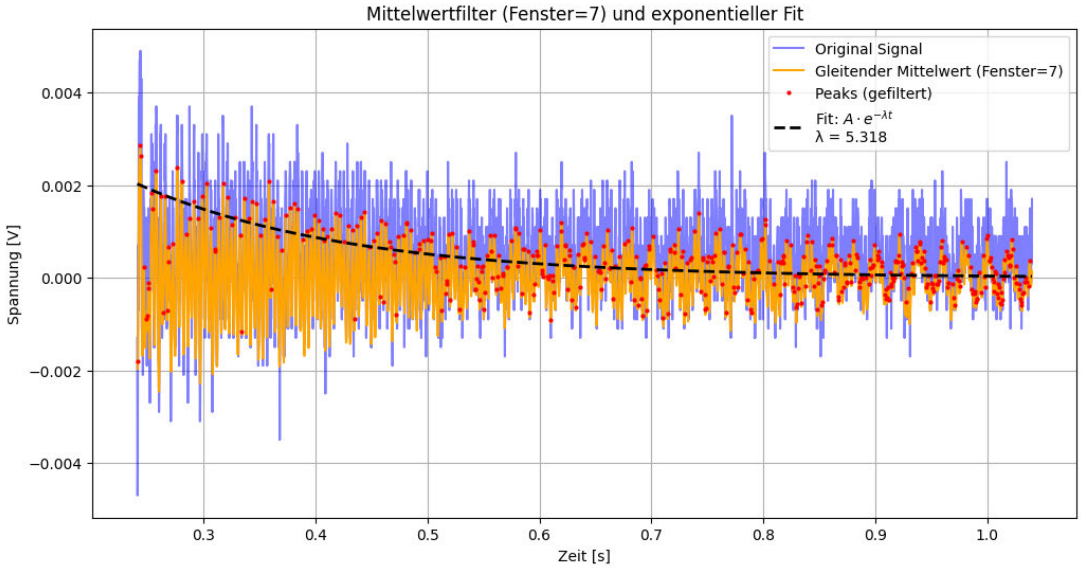

Abbildung 3.11: Abklingkurve von Spiegel MoSc_42 in stehender Position mit Fit

energetischen Verluste pro Zyklus nicht nur durch die Dämpfung, sondern auch durch die jeweils angeregte Schwingungsfrequenz bestimmt werden.

3.5 Frequenzverhalten MEMS-Spiegel

In diesem Experiment soll die Wechselwirkung zwischen der Bewegung des Aktuators und der Spannung am Sensorelement untersucht werden. Auf dieser Grundlage sollen die Systemparameter für die spätere Sollwertregelung extrahiert werden. Um die Wechselwirkung der piezoelektrischen Aktuatoren und Sensoren zu bestimmen, wird der Spiegel über zwei Aktuatoren betrieben. Die Aktuatoren werden dabei entgegengesetzt gepolt angesteuert, um die maximale Auslenkung des Spiegels zu erreichen. Ein Piezoelement erzeugt nur dann eine Spannung, wenn es in Bewegung ist. Wird der Aktuator auf eine bestimmte Position gebracht, bleibt das Signal konstant, sobald die Position erreicht ist. Hieraus ergibt sich ein Hochpassverhalten der Sensormodule in Bezug auf die Aktuatoren. Wie in [25] beschrieben, wird die Grenzfrequenz des Hochpasses hauptsächlich durch den angeschlossenen Messverstärker bestimmt. Bei vielen Geräten ist diese wählbar. Bei der Verwendung von Spannungsverstärkern ergibt sich die Grenzfrequenz aus der RC-Zeitkonstante, die sich aus dem Verstärkereingangswiderstand sowie den Kapazitäten von Sensor, Kabel und Verstärkereingang zusammensetzt. In diesem Experiment werden ausschließlich Oszilloskope verwendet, deren Eingang auf 1 MΩ eingestellt ist. Eine gängige Methode zur Charakterisierung eines Systems besteht darin, die Übertragungsfunktion zu bestimmen. Hierbei wird das Übertragungsverhalten des Spiegels beschrieben, wobei die Aktuatoren als Eingang und die Sensoren als Ausgang dienen. Im sogenannten Konstant-TOSA-Experiment wird der Spiegel so angesteuert, dass er über die Frequenz einen konstanten TOSA-Wert beibehält. Ein TOSA von 1,5° wird dabei als quasistatische Vollauslenkung des Spiegels angenommen. Zwar existieren Spiegel, die eine größere quasistatische Auslenkung erreichen können, jedoch wurde der Wert von 1,5° gewählt, um eine ausreichende Auswahl an Prüflingen sicherzustellen und im Falle einer Beschädigung noch Ersatzexemplare verfügbar zu haben.

In Abbildung 3.12 sind die Messwerte des Spiegels MoSc_42 repräsentativ für eine Reihe von Spiegeln mit guter quasistatischer Auslenkung dargestellt, Fertigungstoleranzen werden an dieser Stelle nicht weiter betrachtet.

Bei den ersten vier Messpunkten aus Abbildung 3.12 ist die Eingangsspannung nahezu konstant, was den tatsächlichen quasistatischen Frequenzbereich des Spiegels widerspiegelt. Die Sensorspannungen in diesem Bereich steigen mit zunehmender Frequenz an, sodass die Winkelgeschwindigkeit abgelesen werden kann. Ab etwa 100 Hz wird der Einfluss der Resonanzfrequenz deutlich, sodass die Antriebsspannung reduziert werden muss, um den TOSA von 1,5° konstant zu halten.

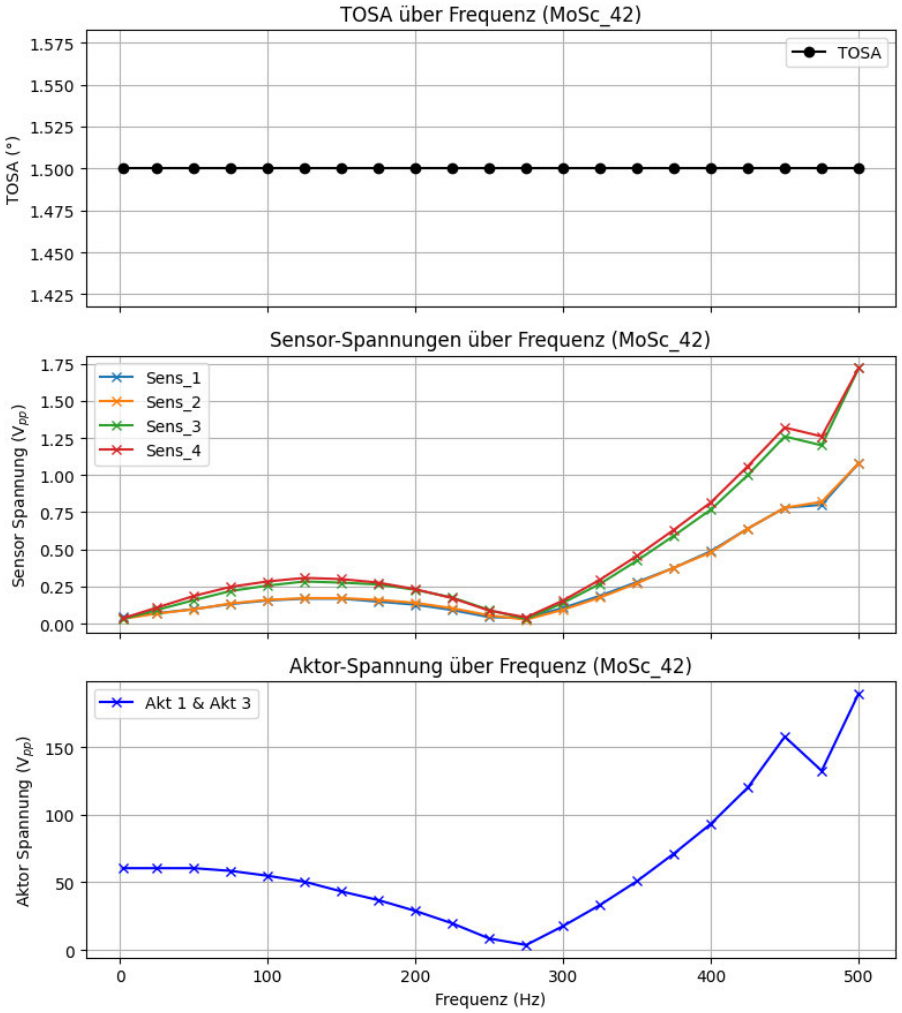

Abbildung 3.12: Konstanter TOSA Messung mit getriebenen Aktuator 1 und 3

Die Verstärkung der Bewegung resultiert aus dem Trägheitsmoment der bewegten Masse. Durch die Hebelwirkung schwingt die Spiegelplatte stärker als die Aktuatoren. Um den gewünschten Auslenkwinkel zu halten, müssen die Aktuatoren daher weniger weit auslenken, was zu geringeren Antriebs- und Sensorspannungen führt.

Messungen mit dem LDV, dargestellt in Abbildung 3.13, zeigen, dass sich die Bewegungsart vor und nach der Resonanz unterscheidet. Um die Bewegungsart von Aktuatoren und Federsystem zu analysieren, wurden LDV-Messpunkte sowohl auf der Säule als auch direkt auf den Aktuatoren platziert. Die Messpunkte auf der Säule repräsentieren die Position der Spiegelplatte, Abweichungen durch das händische verbinden der Spiegelplatte mit der Säule werden vernachlässigt. Für die LDV-Messung wurden keine TOSA-Messungen durchgeführt, da große Verkippungen zu unsauberen Messdaten führen, wie in Kapitel 2.5 beschrieben. Die Aktuatoren 1 und 3 wurden mit einem Sinussignal des im LDV integrierten Funktionsgenerators angesteuert, und zwar mit den Amplituden aus Abbildung 3.12. Die Daten sind in Abbildung 3.13 jeweils in drei Bildern pro Messung visualisiert.

Bei niedrigen Frequenzen, wie 75 Hz, verhält sich das Federsystem quasi starr. Der Messpunkt auf der Säule folgt den Aktuatoren nahezu vollständig, während das Federsystem nur minimal bewegt wird.

Bei etwa 275 Hz zeigt das System eine harmonische Bewegung mit gleichmäßiger Kopplung zwischen Spiegel, Federsystem und Aktuatoren. In der LDV-Animation erscheint es, als würde eine Welle durch den Spiegel laufen. Dabei sind die Aktuatoren trotz Gegentaktansteuerung nicht mehr um 180° phasenverschoben, was im mittleren Bild der Abbildung 3.13 deutlich zu erkennen ist.

Bei höheren Frequenzen, beispielsweise 400 Hz, treten größere Differenzen zwischen der Säule und den Aktuatoren auf, sodass der Aktuator den Spiegel zwar auslenkt, das Federsystem jedoch zu träge reagiert, um die Auslenkung auf Säule und Spiegel zu übertragen. Zudem wird bei dieser Frequenz die Hubmode der Spiegelplatte bereits leicht angeregt.

Eine wichtige Erkenntnis aus den Bewegungsaufnahmen ist die Erklärung für Sensorspannungen an Sensorelementen, die nicht direkt angesteuert werden. In den Messungen aus Abbildung 3.13 sind Torsionsbewegungen bei den Aktuatoren 2 und 4 erkennbar, die selbst nicht getrieben werden. Diese Torsionsbewegungen führen zu Sensorspannungen auf der entsprechenden Achse.

Eine mögliche Optimierung des Sensordesigns besteht darin, das Sensorelement nicht an der Außenkante des Aktuators, sondern als piezoelektrischen Streifen in der Mitte des Aktuators zu positionieren. Dadurch ließen sich Torsionsanteile in den Sensorsignalen verringern und der Messbereich stärker auf die dominierende Biegebewegung fokussieren. Eine konkrete Auslegung und experimentelle Verifikation eines solchen Designs waren nicht Teil dieser Arbeit.

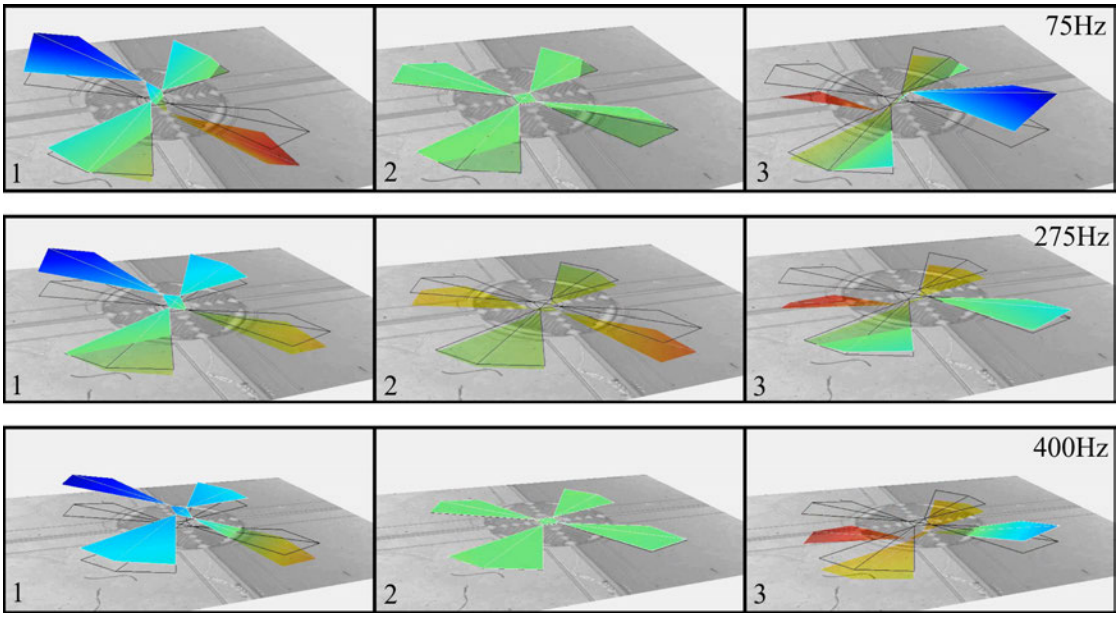

Abbildung 3.13: Aufgenommene Schwingungsbewegung des MEMS-Spiegels mittels LDV bei Resonanzfrequenz sowie bei Frequenzen darunter und darüber. Dabei wird Anregung über Aktuator 1 und 3 eingebracht.

Aus der Eingangsspannung der Aktuatoren und der Ausgangsspannung der Sensorelemente lässt sich die Übertragungsfunktion des Systems ableiten. In Abbildung 3.14 ist diese gemäß

$$G(s) = \frac{Y(s)}{X(s)} \xrightarrow{s=j\omega} \frac{U_{\text{out}}(j\omega)}{U_{\text{in}}(j\omega)} \tag{3.12}$$

dargestellt. Die Funktion wurde auf den Mittelwert der letzten vier Werte auf eins normiert, um später den Hochpass mathematisch isolieren zu können. Der Normierungsfaktor beträgt $\frac{1}{1{,}528 \cdot 10^{-2}}$.

Im Frequenzgang ist das typische Hochpassverhalten der piezoelektrischen Sensoren deutlich erkennbar, ebenso das Feder-Massen-Dämpfersystem (FMD), das durch den signifi-

kanten Resonanzpeak charakterisiert wird. Häufig wird ein Federmasse-Dämpfer-Modell zur Beschreibung von MEMS-Bauteilen verwendet [28].

Die Messpunkte für die Übertragungsfunktion wurden im Abstand von 25 Hz gesetzt. Die Resonanzfrequenz bei 266 Hz wurde nicht direkt gemessen, da dort die Resonanz der nicht getriebenen Achse zu stark angeregt wird. Infolgedessen bewegt sich der Laser, der für die TOSA-Messung verwendet wird, elliptisch anstatt linear entlang der getriebenen Achse, was eine exakte Messung der Übertragungsfunktion in diesem Bereich erschwert. Um das

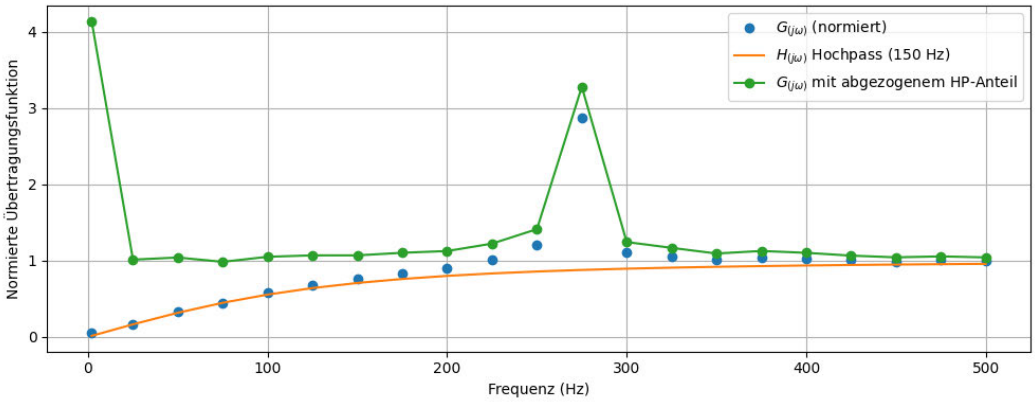

Abbildung 3.14: Extraktion des FMD-Anteils durch Entfernung des Hochpassanteils aus der normierten Übertragungsfunktion der Sensorelemente 2 und 4

System besser analysieren zu können und den Fitting-Algorithmus zu vereinfachen, wurde im ersten Schritt der Systemanalyse der charakteristische Hochpass herausgerechnet. In Abbildung 3.14 sind die Frequenzverläufe dargestellt. Der Frequenzgang des Spiegels erreicht einen Betrag von 0,707 bei etwa 150 Hz. Dies deutet darauf hin, dass ein Hochpass erster Ordnung mit einer Grenzfrequenz von 150 Hz den niederfrequenten Anteil des Frequenzgangs sehr gut abbildet. Anschließend wurde die Hochpassfunktion durch die Messdaten geteilt, sodass die resultierende Funktion nur noch das Feder-Masse-Dämpfer-System darstellt. Der erste Messpunkt bei 2 Hz der Übertragungsfunktion mit abgezogenem Hochpass weicht deutlich von den übrigen Messwerten ab. In diesem Frequenzbereich ist die gemessene Signalamplitude sehr gering. Bereits kleine relative Beiträge von Rauschen und Quantisierung führen zu einem großen Fehler, durch das Multiplizieren mit der Umkehrfunktion des Hochpasses entsteht dann der große Ausreißer bei 2 Hz. Da der Messpunkt den ansonsten glatten Verlauf des Betragsfrequenzgangs nicht widerspiegelt,

wurde er für das Fitten des Modells bewusst ausgeschlossen.

$$
\begin{aligned}
G_{\text{ges}}(s) &= H_{\text{HP}}(s) \cdot G_{\text{ohne HP}}(s) \\
G_{\text{ges}}(j\omega) &= \frac{j\omega}{j\omega + \omega_c} \cdot \frac{1}{k - m(j\omega)^2 + jc\omega} \\
G_{\text{ohne HP}}(j\omega) &= \frac{G_{\text{ges}}(j\omega)}{H_{\text{HP}}(j\omega)}
\end{aligned}
$$

Für das Fitting wurden ausschließlich die Messwerte verwendet, die das Feder-Masse-Dämpfer-System repräsentieren, um die Resonanzfrequenz besser an das physikalische System anzupassen. Da die X-Achse nicht isoliert angeregt werden kann, weil ihre Resonanz sehr nah an der Resonanz der Y-Achse liegt, wurde ein künstlicher Messpunkt bei 268 Hz mit einer Amplitude von 40 gesetzt.

Für das Fitten wurde die gleiche Bibliothek wie in Kapitel 3.4 verwendet. Dem Fitting-Algorithmus wurde die Funktion `G_fdm()` aus Listing 3.2 übergeben.

$$
G_{\text{ohne HP}}(j\omega) = \frac{A}{\sqrt{(k - m\,j\omega^2)^2 + (c\,j\omega)^2}} \tag{3.13}
$$

```
# Modellfunktion fuer das Feder-Masse-Daempfer-System
def G_fdm(omega, m, c, k, A):
    x = A / np.sqrt((k - m * omega**2)**2 + (c * omega)**2)
return x
```

Listing 3.2: Feder Massen Dämpfersystem zur übergabe an den Fitting Algorithmus

Als Input für das Fitting wurde ausschließlich das FMD ausgewählt, wobei der erste Funktionswert bei 2 Hz ausgeblendet wurde, in Abbildung 3.14 grün dargestellt. Damit das Fittingtool physikalisch sinnvolle Werte liefert, muss dem Algorithmus ein plausibles Set an Startwerten übergeben werden. Die Messungen aus Kapitel 3.4 liefern alle erforderlichen Startwerte, bis auf die Federkonstante k, die aus der bekannten Resonanzfrequenz und der Masse berechnet werden kann.

Startwerte:

- $m = 3{,}6 \cdot 10^{-6}\,\mathrm{kg}$
- $c = 39{,}13 \cdot 10^{-6}\,\frac{\mathrm{kg}}{\mathrm{s}}$
- $k = (j\omega)^2 \cdot m = 10{,}31\,\frac{\mathrm{kg}}{\mathrm{s}^2}$
- $a = 10$

Beim Fitten der Messdaten ist es entscheidend, geeignete Grenzen für die Parameter festzulegen. Ohne diese Boundaries können Optimierungsalgorithmen unrealistische oder unphysikalische Werte liefern, insbesondere bei nichtlinearen Modellen oder bei stark verrauschten Daten. Durch das Setzen plausibler Parametergrenzen wird sowohl die Konvergenz des Fit-Prozesses verbessert als auch die physikalische Konsistenz der Ergebnisse gewährleistet.

Tabelle 3.1: Parametergrenzen und gefittete Werte

Parameter	**Minimum**	**Maximum**	**Gefitteter Wert**
Masse m	$3 \cdot 10^{-6}\,\mathrm{kg}$	$6 \cdot 10^{-6}\,\mathrm{kg}$	$3{,}743 \cdot 10^{-6}\,\mathrm{kg}$
Dämpfung c	$20 \cdot 10^{-6}\,\mathrm{kg\,s^{-1}}$	$100 \cdot 10^{-6}\,\mathrm{kg\,s^{-1}}$	$81{,}89 \cdot 10^{-6}\,\mathrm{kg\,s^{-1}}$
Federkonstante k	$9\,\mathrm{kg\,s^{-2}}$	$13\,\mathrm{kg\,s^{-2}}$	$10{,}27\,\mathrm{kg\,s^{-2}}$
Verstärkung A	1	15	9,122
Tiefpassfilter f_{TP}	-	-	150 Hz

Die Ergebnisse aus Tabelle 3.1 sind in Abbildung 3.15 grafisch dargestellt. Im niederfrequenten Bereich bildet das gefittete FMD das physikalische System sehr gut ab. In der Nähe der Resonanzfrequenz überschätzt das Modell die Amplitude leicht, was in der Praxis vorteilhaft ist: Eine Unterschätzung könnte unerwartete Schwingungen oder Stabilitätsprobleme verursachen, während eine moderate Überschätzung konservative Werte liefert und die Systemauslegung absichert. Eine zu starke Überschätzung sollte jedoch vermieden werden, da sie zu übermäßig vorsichtigen Auslegungen oder unnötig hohen Sicherheitsreserven führen könnte. Bei höheren Frequenzen unterschätzt das Modell die Messwerte, da der MEMS-Spiegel zur Erreichung der Normalauslenkung (TOSA 1,5°) über die vorgesehenen Spannungsgrenzen hinaus betrieben wurde. Dies stellt für die vorliegende Messung jedoch kein Problem dar. Insgesamt liefert der Fit physikalisch plausible Systemparameter, die mit Ergebnissen aus anderen Arbeiten, in denen ähnliche oder identische Spiegel verwendet wurden, übereinstimmen. Die hier bestimmten Parameter

dienen dazu, das System in MATLAB abzubilden und bilden gleichzeitig die Grundlage für die spätere Auslegung eines Reglers.

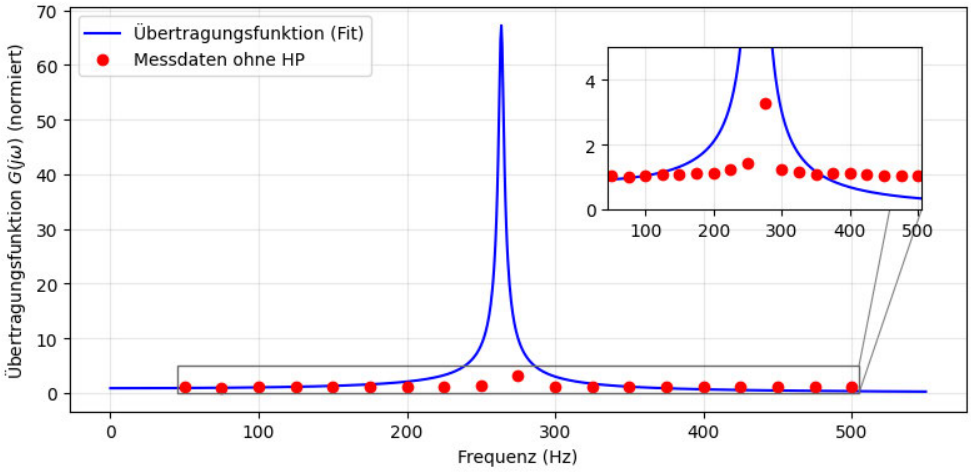

Abbildung 3.15: Fitting Ergebnisse vom Chip MoSc_42 aus Messdaten von 3.14.

3.6 Aktuatorkraft

Damit der Spiegel einer Störeinwirkung auch entgegenwirken kann, müssen die Aktuatoren über genügend Kraft verfügen. Die Kraft der Aktuatoren kann mit der Formel $F = m \cdot a$ abgeschätzt werden. Dafür wird angenommen, dass sich die Aktuatorspitze aus Abbildung 2.5 nur nach oben oder unten bewegt. Weiterhin wird angenommen, dass beide Aktuatoren den Spiegel beschleunigen und der Spiegel als Punktmasse betrachtet werden kann. Die beiden Federsysteme werden als ideale Gelenkpunkte angenommen. Der Weg, den die beiden Aktuatoren zurücklegen, berechnet sich aus dem bekannten maximalen TOSA von 2°, was einer Bewegung der Spiegelplatte von 1° entspricht. Die Aktuatorenden sind 1,2 mm voneinander entfernt, wie in Abbildung 2.6 zu entnehmen ist und in der folgenden Formel mit L beschrieben wird. Die Bewegungsdifferenz Δs kann wie folgt berechnet werden:

$$\Delta s = \tan\left(\frac{\text{TOSA}}{2}\right) \cdot L$$

$$20{,}9\,\mu\text{m} = \tan(1°) \cdot 1{,}2\,\text{mm}$$

Die Zeit, die beide Aktuatoren benötigen, um die quasistatische Vollauslenkung zu erreichen, ist in Abbildung 3.16 dargestellt. Dort ist die Sprungantwort einer Achse gezeigt; dabei wurde die X-Achse vom Mittelpunkt zur maximalen Auslenkung gesteuert. Die Zeit zwischen Beginn des Sprungs und der maximalen Sensorspannung wird als Indikator für die Bewegung verwendet. Daraus ergibt sich eine Bewegungszeit von 562 µs.

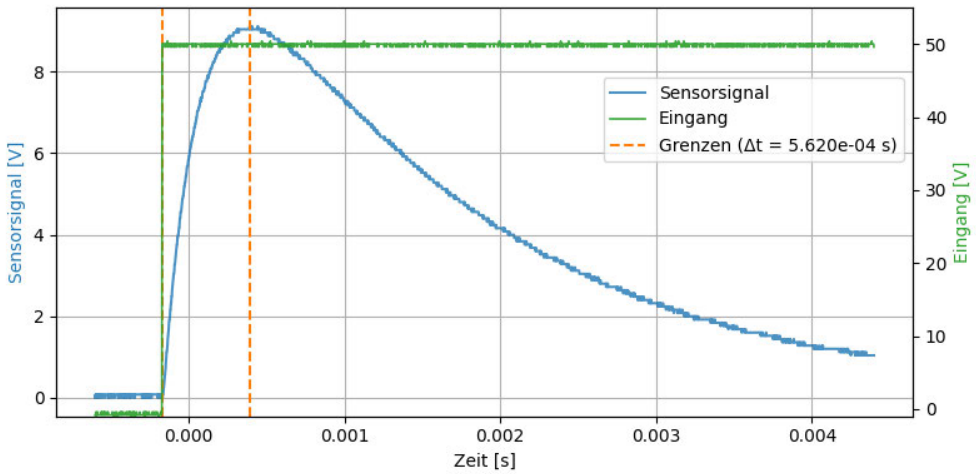

Abbildung 3.16: Sprungantwort des Sensors 1 des MEMS-Spiegels

Die Masse zur Berechnung der Kraft ist in Kapitel 3.4 bestimmt worden und beträgt $3{,}657 \cdot 10^{-6}\,\mathrm{kg}$.

Die Kraft von zwei Aktuatoren berechnet sich damit wie folgt:

$$
\begin{aligned}
F &= m \cdot \frac{l}{t^2} \\
F &= 3{,}657 \cdot 10^{-6}\,\mathrm{kg} \cdot \frac{20{,}9\,\mu\mathrm{m}}{(562\,\mu\mathrm{s})^2} \\
F &= 241{,}9\,\mu\mathrm{N}
\end{aligned}
$$

Die Kraft, die durch den Stoß in das System eingebracht wird, ist in Kapitel 3.3 mit $24\,\mathrm{m\,s^{-2}}$ als Anforderung festgelegt worden. Daraus resultiert eine Kraft von:

$$
\begin{aligned}
F &= m \cdot a \\
F &= 3{,}657 \cdot 10^{-6}\,\mathrm{kg} \cdot 24\,\mathrm{m\,s^{-2}} \\
F &= 87{,}768\,\mu\mathrm{N}
\end{aligned}
$$

Die abgeschätzte Kraft von 241,9 µN ist somit etwa 2,75-mal größer als die Kraft, die durch den Stoß in das System eingebracht wird. Der in Kapitel 3.3 beschriebene Stoß entspricht etwa 2,3 g und ist daher vergleichsweise gering. Um die 87,768 µN aus dem Stoß zu kompensieren, sollte die Kraft der Aktuatoren ausreichend sein. Die verringerte effektive Weglänge der Aktuatorbewegung infolge der Federdehnung wird in dieser Abschätzung der Aktuatorkraft jedoch nicht berücksichtigt.

4 Anforderungsanalyse

4.1 Problemstellung

Die in Kapitel 3 untersuchten MEMS-Spiegel sollen quasistatisch angesteuert werden, um innerhalb des Field of View (FOV) gezielt einzelne Punkte mit variabler Verweildauer adressieren zu können. Eine quasistatische Ansteuerung ermöglicht eine flexible Anpassung des Scanmusters und ist insbesondere für Anwendungen relevant, bei denen bestimmte Bereiche des Sichtfeldes mit höherer Genauigkeit oder Frequenz erfasst werden müssen.

Aus der Charakterisierung in Kapitel 3 ist bekannt, dass die Spiegel bei mechanischen Erschütterungen zum Nachschwingen neigen. Dieses Verhalten ist für den praktischen Einsatz nachteilig, da während der Schwingungsphase die Position des Spiegels von der Sollauslenkung abweicht und die Zielpunktgenauigkeit beeinträchtigt wird.

Ein weiteres Ziel dieser Arbeit ist die Entwicklung eines leicht zu bedienenden Demonstratorboards, das die quasistatische Ansteuerung und die stoßresistente Regelung der Spiegel anschaulich demonstriert. Das Board soll als Experimentierplattform und für Vorführzwecke eingesetzt werden können.

4.2 Ausgangsbedingungen des Systems

Als Ausgangslage für die Ansteuerung steht ein MEMS-Chip zur Verfügung, der auf einer Leiterplatte (Printed circuit board (PCB)) montiert ist. Die Leiterkarte ist so optimiert, dass kein Übersprechen der Leiterbahnen für die Sensorspannungen zu den mit hoher Spannung betriebenen Aktuatorleiterbahnen besteht. Nach Angaben der Entwickler traten in früheren Board-Revisionen Übersprechprobleme auf, die durch das neue Design

„NEUMAQ V2“ behoben wurden. Eine eigenständige Verifikation dieser Aussage war nicht Bestandteil dieser Arbeit.

Weiterhin besitzt das „NEUMAQ V2“ eine elektrische Schnittstelle in Form eines Kantensteckverbinders, über den die Verbindung zur externen Ansteuerelektronik hergestellt wird. Die Kontaktflächen sind hierbei direkt an der Leiterplattenkante ausgeführt, was eine kompakte Bauweise ermöglicht und ein schnelles Austauschen verschiedener Spiegel erlaubt. Diese Bauweise erleichtert den Versuchsaufbau erheblich, da verschiedene MEMS-Chips ohne Lötarbeiten getestet werden können und Kontaktfehler beim wiederholten Ein- und Ausbau minimiert werden.

4.3 Funktionale Anforderungen

Die funktionalen Anforderungen stellen die Mindestanforderungen für die quasistatische Ansteuerung dar. Nach einer detaillierten Beschreibung der Anforderungen werden diese in Tabelle 4.1 zusammengefasst.

Ausgangsstufe

Die in Kapitel 3.1 dargestellten Messungen haben gezeigt, dass die MEMS-Spiegel bis zu einer Spannung von ±50 V stabil und ohne Beschädigung betrieben werden können. Die Ausgangsstufe muss in der Lage sein, eine Spannung von ±2,5 V an den Falkon WMA-02-20-Fach-Verstärker auszugeben. Dabei ist sicherzustellen, dass der Ausgang mit dem Eingang des Falkon WMA-02-Verstärkers kompatibel ist und jede ausgegebene Spannung über einen beliebig langen Zeitraum stabil gehalten werden kann.

Eingangsverstärker

Aus Kapitel 3.3.1 geht hervor, dass bei einem Stoß durch den Schütteltisch eine Sensorspannung von etwa ±30 mV zu erwarten ist. Diese Spannung muss so aufbereitet werden, dass ein ADC (Analog-Digital-Wandler) die Signale erfassen kann. Der Eingangsverstärker muss die Eingangsspannung von ±30 mV im Frequenzbereich von DC bis 500 Hz (Hubmode) rauscharm auf 3,3 V verstärken.

Closed-Loop-Regelung

Beim Einwirken eines Stoßes, wie in Kapitel 3.3.1 beschrieben, soll das anschließende Abklingen der Schwingung durch eine Closed-Loop-Regelung zeitlich verkürzt werden. Dabei muss ein Ereignis geregelt werden, das eine maximale Frequenz von 466 Hz aufweist. Nach der Faustformel aus [27, S. 326], die besagt, dass die Abtastzeit so zu wählen ist, dass das Signal mindestens zehnmal pro Periode abgetastet wird, ergibt sich eine erforderliche Abtastrate von mindestens 4660 Hz.

Zusammenfassung der funktionalen Anforderungen

Tabelle 4.1: Funktionale Anforderungen an das Ansteuer- und Regelungssystem

Nr.	**Anforderung**	**Begründung / Herkunft**	**Zielwert / Bedingung**
F1	Treiberstufe	Kompatibilität mit dem Falkon WMA-02-Verstärker zur Ansteuerung der Aktuatoren	Ausgangsspannung $\pm 2{,}5\,\mathrm{V}$
F2	Sensorverstärker	Sensorspannungen Verstärken und an ADC-Eingang des FPGA-Systems Anpassung	Verstärkungsfaktor von 50 und Ausgangsspannung $0\,\mathrm{V}$ bis $3{,}3\,\mathrm{V}$
F3	Steuerung	Quasistatische Ansteuerung beliebiger Spiegelkoordinaten über Software-Schnittstelle	C++-Funktion zur Positionsvorgabe
F4	Regelung	Geschlossener Regelkreis zur Dämpfung von Resonanzschwingungen	Abklingzeit auf 50 % reduzieren
F5	Stabilitätsanalyse	Vermeiden des Aufschingen des Spiegels durch die Regelung	Stabiles System
F6	Abtastrate	Einhaltung der Nyquist-Bedingung für stabile Regelung der Resonanzen	$f_\mathrm{s} \geq 4{,}66\,\mathrm{kHz}$
F7	FPGA-Implementierung	Vollständige Implementierung der Regelung in synthetisierbarem HDL-Code zur Echtzeitverarbeitung und Vorbereitung einer ASIC-Integration	Hardwarefähige Implementierung, deterministisches Zeitverhalten

5 Auslegung des Reglers

5.1 Zielsetzung der Regelung

Ziel der Regelung ist es, die effektive Dämpfung des MEMS-Spiegels künstlich zu erhöhen, insbesondere um das Aufschwingen nach äußerer Schockeinwirkung zu reduzieren. Die Spiegel sollen überwiegend statisch ausgelenkt werden, sodass sie über längere Zeiträume einen kleinen Bereich halten oder lediglich einen begrenzten Abschnitt des FOV abscannen können. Bei Erschütterungen neigen die Spiegel dazu, stark auszulenken, wodurch sie vorübergehend ihre Zielposition verlieren. Eine zentrale Aufgabe der Regelung besteht darin, die Zeitspanne zu minimieren, in der der Spiegel nicht korrekt ausgerichtet ist. In einem späteren Entwicklungsschritt kann die Regelung zu einer Sollwertregelung erweitert werden, um zusätzlich das Überschwingen des Spiegels beim schnellen Anfahren einer Zielkoordinate zu reduzieren.

5.2 Systemmodell

Auf Grundlage der experimentellen Charakterisierung wurde ein vereinfachtes mathematisches Modell des Systems erstellt. Die gemessene Übertragungsfunktion entspricht dem Verhalten eines unterdämpften Schwingungssystems zweiter Ordnung mit zusätzlicher Hochpasscharakteristik. Dieses Verhalten ist für mikromechanische Strukturen typisch und lässt sich durch ein vereinfachtes FMD-Modell beschreiben. Die Modellparameter wie Resonanzfrequenz, Dämpfungsfaktor und normierende Verstärkung wurden in Kapitel 3.5 identifiziert und bilden die Grundlage für die anschließende Reglerauslegung. Für sehr niedrige Frequenzen wird das gemessene Hochpassverhalten berücksichtigt, indem das Modell um einen zusätzlichen Hochpassfaktor erweitert wird. Das so bestimmte Ersatzmodell ist in Abbildung 5.1 dargestellt und dient als Grundlage für die nachfolgende Reglerauslegung.

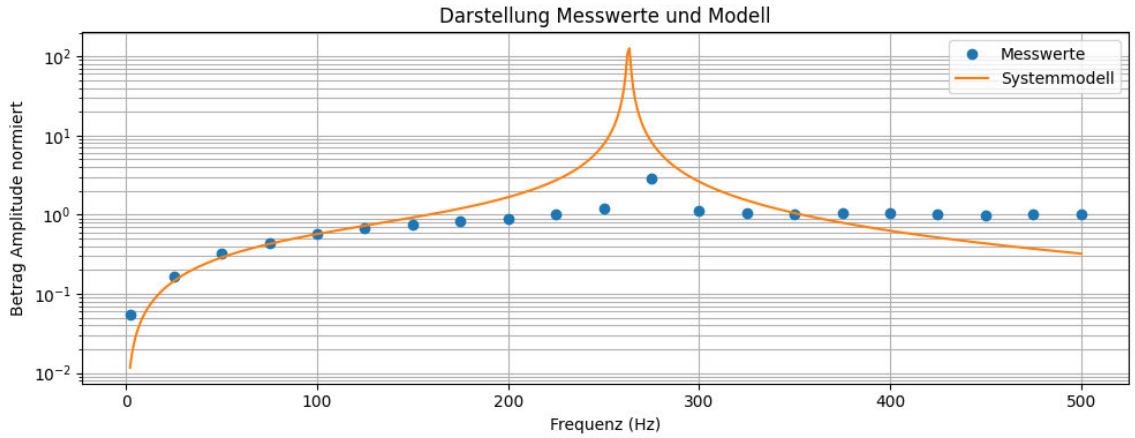

Abbildung 5.1: Systemdarstellung zur Reglerkalibrierung

Das Modell stellt sicher, dass die dominanten dynamischen Eigenschaften, insbesondere die Resonanzfrequenzen und deren Dämpfung, erfasst werden, während parasitäre Effekte und hochfrequente Nebenresonanzen vernachlässigt werden können. Die Resonanzfrequenzen der einzelnen Spiegel liegen bei 268 Hz ±5 Hz. Die Implementierung des Modells erfolgte in MATLAB/Simulink, sodass es direkt für die Simulation der Regelkreise verwendet werden kann. Der Regler kann über den HDL-Coder von MATLAB in eine Hardware Description Language (HDL)-Datei exportiert werden. In Abbildung 5.2 ist das Subsystem, das für die Reglerauslegung verwendet wird, anschaulich dargestellt. Das Subsystem wurde in MATLAB mit einer Maske versehen, in der alle relevanten Spiegelparameter eingegeben werden können, wodurch eine einfache Anpassung für andere Spiegel möglich ist. Die Totzeit der gesamten Regelstrecke setzt sich aus den Verzögerungen des Sensorverstärkers, des Analog digital Umsetzer (ADC), der FPGA-Reglerstruktur, des Digital-Analog-Umsetzer (DAC) sowie der Treiber- und Endstufe zusammen. Die Latenz des Sensorverstärkers äußert sich im Wesentlichen als Tiefpassverhalten im oberen kHz-Bereich und kann daher im Rahmen der Totzeitbetrachtung vernachlässigt werden. Auch die Treuberstufe und Endstufe liefern keinen Signifikanten beitrag zur Totzeit.

Der verwendete ADC (MCP3201) weist eine Totzeit auf, die sich aus der analogen Eingangssamplezeit von 1,5 Serial Peripheral Interface (SPI)-Takten sowie der Konversionszeit von 12 SPI-Takten zusammensetzt. Da der implementierte HDL-Code zur Synchronisation stets ein vollständiges 16-Bit-Übertragungsfenster ausliest, ergibt sich für den ADC eine effektive Totzeit von $t_{\mathrm{ADC}} = 16 \cdot \frac{1}{0{,}8\,\mathrm{MHz}} = 20\,\mu\mathrm{s}$.[18, S.3]

Die Totzeit des DAC (DAC121S101) setzt sich aus der Einschwingzeit der Ausgangsspannung und der Dauer der SPI-Übertragung zusammen. Die Einschwingzeit beträgt laut Datenblatt bei einer Lastkapazität von 500 pF etwa 12 µs. Während der Übertragung werden 16 Bit seriell übertragen, wobei das Chip-Select-Signal nach der vollständigen Übertragung wieder auf High gesetzt wird. Bei einer SPI-Taktfrequenz von 0,8 MHz ergibt sich somit eine Übertragungsdauer von $t_{\mathrm{DAC}} = 16 \cdot \frac{1}{0{,}8\,\mathrm{MHz}} + 12\,\mu\mathrm{s} = 32\,\mu\mathrm{s}$. Damit ergibt sich eine gesamte DAC-Latenz von 32 µs, also die minimale Zeitspanne zwischen Beginn der SPI-Übertragung und Erreichen des stabilen Ausgangswerts. [21, S.5]

Die dominierende Verzögerung wird durch die Abtastrate der Reglerroutine bestimmt. Jeder Ausführungstakt der Reglerberechnung fügt eine zeitdiskrete Verzögerung in Höhe der Abtastperiode hinzu, $t_{\mathrm{Regler}} = \frac{1}{20\,\mathrm{kHz}} = 50\,\mu\mathrm{s}$. Durch Anheben der Reglertaktung auf 40 kHz halbiert sich die Abtastperiode auf 25 µs und damit dieser Verzögerungsanteil.

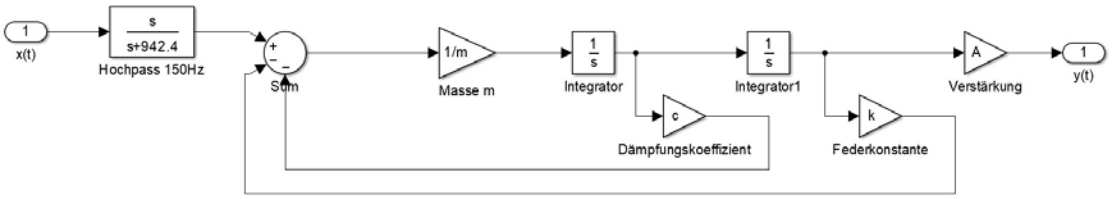

Abbildung 5.2: Subsystem für die Reglerauslegung in Matlab

5.3 Reglerkonzept und Auslegung

5.3.1 Wahl des Reglertyps Störgrößenregelung

Für die Störgrößenregelung wurde eine Proportionalregelung gewählt. Wie in Kapitel 2.7.2 beschrieben, erzeugt das Sensorelement nur dann eine Spannung, wenn der Aktuator in Bewegung ist. Mit zunehmender Geschwindigkeit steigt die abgegebene Spannung an. Dieses Verhalten lässt sich aus dem Experiment in dem das Frequenzverhalten bestimmt wurde ableiten, da dort sichtbar wird, dass bei gleicher Auslenkung des Spiegels und höherer Frequenz die Sensorspannung zunimmt. Dieses Phänomen kann genutzt werden, da die zeitliche Ableitung des Ausgangs eines FMD-Systems der Dämpfung entspricht. Durch Anwendung eines P-Reglers auf das Sensorsignal lässt sich somit die Dämpfung des Systems direkt erhöhen. Eine höhere Dämpfung führt zu einer verkürzten Abklingzeit des

MEMS-Spiegels. Der P-Regler ist außerdem einfach implementierbar und robust gegenüber Modellunsicherheiten. Da ein Stoß ein breitbandiges Anregungsspektrum besitzt, ist eine breit wirkende Dämpfungsregelung am effektivsten. Daher ist vor der Rückführung des Signals keine zusätzliche Filterung erforderlich.

Die Simulation der Proportionalregelung ist in MATLAB/SIMULINK erstellt worden. Der Simulationsaufbau ist in Abbildung 5.3 dargestellt. Dabei wurde mit dem Block „Signal Editor" das Störsignal generiert, das den gleichen Anstiegszeiten aus den Schockmessungen in Kapitel 3.3 entspricht. Bei den Ausgangssignalen der beiden Systeme wurde ein Rauschen überlagert, sodass sich die Systeme wie der reale Aufbau verhalten. Für die Simulation werden zwei Subsysteme mit der Störgröße beaufschlagt: Eines wird nicht geregelt und zeigt das Verhalten der simulierten Sensorspannung, während beim zweiten Subsystem ein Proportionalregler mit einer Zeitverzögerung von 60 μs auf den Eingang des geregelten Subsystems wirkt. Die 60 μs wurden messtechnisch durch eine gezielte Anregung des Reglers erfasst und geben die Totzeit der ADC-, DAC- und Treiberstufen an.

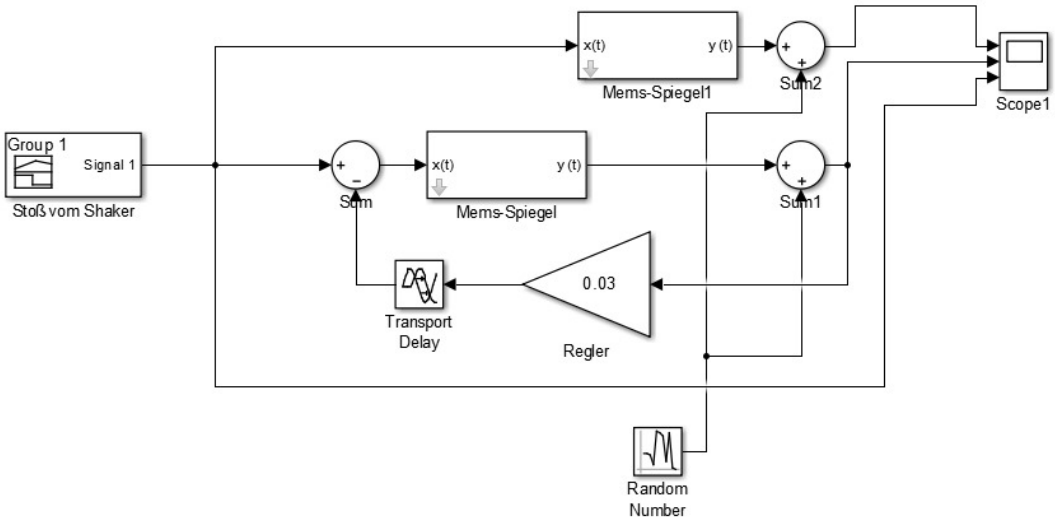

Abbildung 5.3: Simulationsaufbau der Proportionalregelung

In Abbildung 5.4 sind die Simulationsergebnisse der Proportionalregelung dargestellt. Im FPGA sind die digitalen Werte von 1 bis −1 normiert, was die Y-Achse widerspiegelt. Im oberen Graphen ist das simulierte Abklingen des Spiegels dargestellt. In der Mitte ist das Verhalten des Systems mit aktiver Proportionalregelung zu erkennen. Im unte-

ren Graphen ist die simulierte Störgröße, die dem Schocksignal nachempfunden ist, der Vollständigkeit halber dargestellt.

Im Vergleich ist deutlich eine Verbesserung des Abklingverhaltens zu erkennen. Experimente mit größeren Reglerkoeffizienten haben ein noch besseres Dämpfungsverhalten gezeigt. Ab einer Totzeit von 150 μs verschlechtert sich die Abklingzeit, diese Totzeit wurde simulativ bestimmt. Die gemessene Totzeit von 60 μs ist dementsprechend unproblematisch. Erst bei der Implementierung von mehr als drei Taktverzögerungen zeigt das System eine schlechtere Dämpfung.

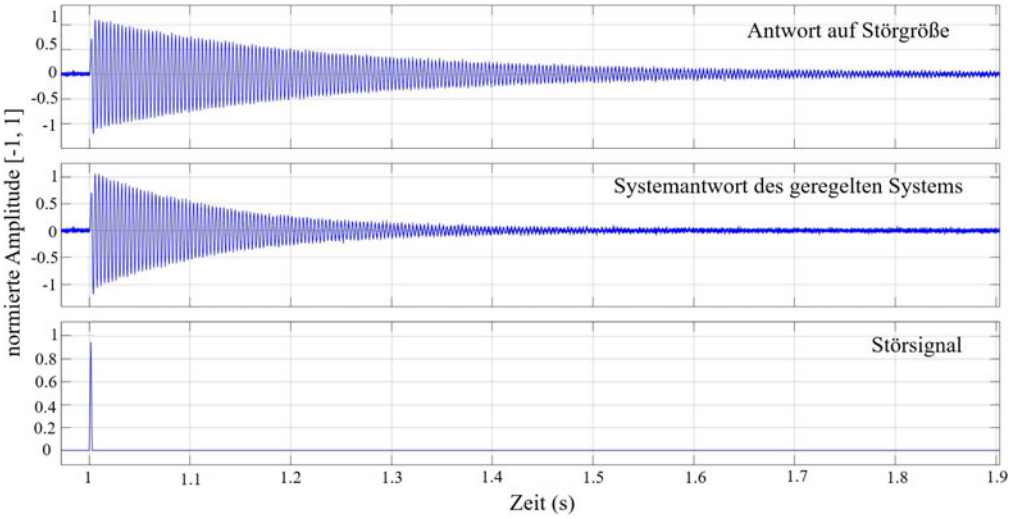

Abbildung 5.4: Ergebnisse der Simulation aus Abbildung 5.3

5.4 Realisierung des Reglers in MATLAB/SIMULINK

5.4.1 Eingangs Normierung

Die Analog-Digital-Wandler liefern 12-Bit-Unsigned-Integer-Werte im Bereich von [0 ... 4095]. Zur weiteren Verarbeitung werden diese Werte im FPGA um den Mittelwert verschoben und in das Festkommaformat Q1.15 überführt. Dadurch steht ein bipolarer Wertebereich von [−1 ... 0,9999] zur Verfügung.

Alle Filter- und Regleroperationen werden im FPGA in Festkommaarithmetik ausgeführt, um Rechenaufwand und Ressourcen zu minimieren. In Abbildung 5.5 ist die Um-

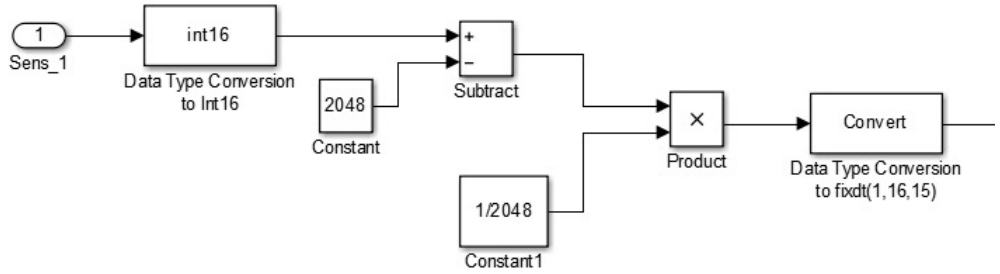

Abbildung 5.5: HDL fähige Simulinkblöcke zur Konvertierung der ADC-Werte in fixedpoint Q1.15

wandlung des ADC-Signals für die Regelung dargestellt. Das Signal `Sens_1` aus Abbildung 5.5 wird zunächst in das Standarddatenformat `Int16` konvertiert. Eine Auflösung von 16 Bit stellt dabei einen guten Kompromiss zwischen Ressourcenverbrauch und numerischer Genauigkeit dar.

Das 12-Bit-ADC-Signal wird anschließend durch Addition eines Offsets von 2048 verschoben, da die Aktuatoren sowohl positive als auch negative Spannungen ansteuern. Der so gewonnene Eingangswert wird mit $\frac{1}{2048}$ multipliziert, um den Wertebereich auf $[-1 \ldots 0{,}9999]$ zu normieren. Um eine deterministische Berechnung der Regelung zu gewährleisten, wird auf Gleitkommaarithmetik verzichtet. Der Wertebereich ist im Q1.15-Festkommaformat definiert, was eine numerisch stabile Umsetzung der Filter- und Regleralgorithmen ermöglicht.

5.4.2 Ausgangs Konvertierung

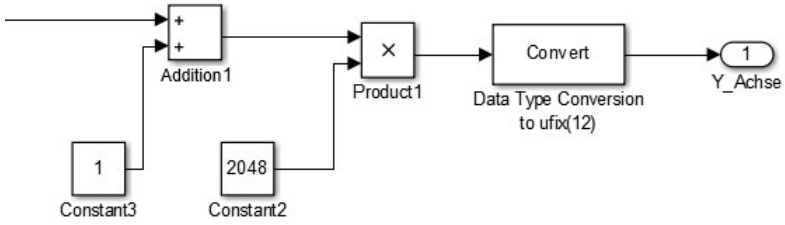

Abbildung 5.6: HDL fähige Simulinkblöcke zur Konvertierung der Ausgangswerte von fixedpoint Q1.15 in uInt16

Damit der berechnete Ausgangswert an den DAC übergeben werden kann, muss eine Rücktransformation in ein 12-Bit-Unsigned-Integer-Format erfolgen. Die hierfür notwendigen Operationen sind in Abbildung 5.6 in Form von HDL-Coder-kompatiblen Blöcken dargestellt.

Zunächst wird das berechnete Ergebnis um den Wert eins erhöht, wodurch der zuvor zur Berechnung benötigte Offset entfernt und ein ausschließlich positiver Wertebereich erzeugt wird. Anschließend wird dieser Wert, der nun im Bereich von 0 bis 1,999 liegt, mit 2048 multipliziert, um den numerischen Bereich auf [0 ... 4096] abzubilden. Der so erhaltene Ausgangswert wird anschließend an den 12-Bit-DAC ausgegeben.

5.4.3 Proportionalrgeler

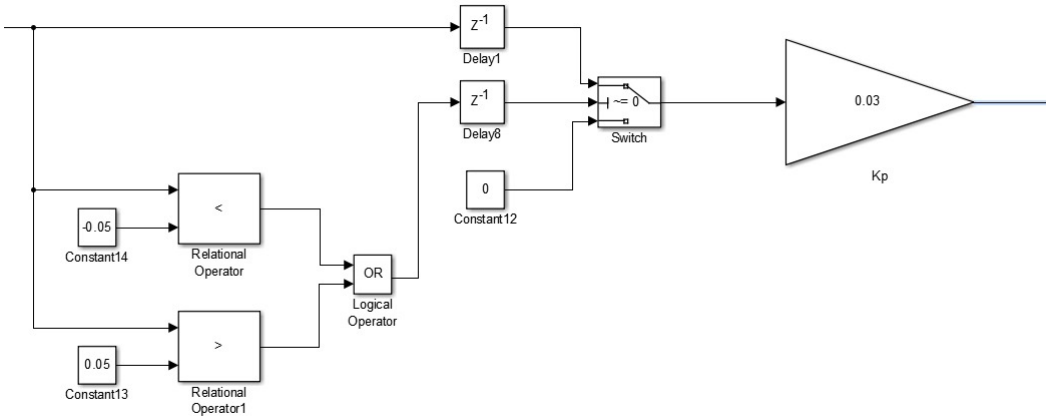

Abbildung 5.7: Proportionalregelung mit Totband in Simulink

Bei der Realisierung eines Proportionalreglers wird der Eingangswert mit einem Verstärkungs- oder Dämpfungsfaktor multipliziert und das Vorzeichen invertiert. Im in Abbildung 5.7 dargestellten Regler ist zusätzlich ein Totband implementiert, das einen Bereich definiert, in dem keine Regelung stattfindet.

Bei der Umsetzung des Totbands ist darauf zu achten, keine arithmetischen Schleifen zu erzeugen. Aus diesem Grund wurden, wie in Abbildung 5.7 dargestellt, die Blöcke *Delay 1* und *Delay 8* eingefügt. Der Block *Delay 1* stellt sicher, dass zeitlich zueinander passende Werte am Schalter verglichen und weitergeleitet werden, während *Delay 8* die Schleifenbildung verhindert.

Jede Verzögerung im Regler erhöht die Totzeit um $\frac{1}{20\,\mathrm{kHz}} = 50\,\mu\mathrm{s}$. Daher sind lange Filterstrukturen für die Regelung ungeeignet. Für die Signalaufbereitung sollten bevorzugt einfache IIR-Filter eingesetzt werden.

Damit der Regler das reale Verhalten des Systems korrekt abbildet, muss der in der Simulation berechnete Ausgangswert um denselben Faktor skaliert werden, der im Hardwarepfad wirksam ist. Aus der Berechnung in Gleichung 5.1 ergibt sich, dass sich die Verstärkungsfaktoren der End-, Treiber- und Sensorverstärker sowie die Skalierungen von ADC und DAC gegenseitig aufheben. Lediglich der in Kapitel 3.5 eingeführte Normierungsfaktor bleibt wirksam:

$$U_{\mathrm{in}} \cdot G(s) \cdot A_{\mathrm{Endstufe}} \cdot A_{\mathrm{Treiberstufe}} \cdot A_{\mathrm{ADC}} = U_{\mathrm{out}} \cdot A_{\mathrm{Normierung}} \cdot A_{\mathrm{Sensorverstärker}} \cdot A_{\mathrm{DAC}} \tag{5.1}$$

$$U_{\mathrm{in}} \cdot G(s) \cdot 20 \cdot 2{,}5 \cdot \frac{1}{3{,}3} = U_{\mathrm{out}} \cdot \frac{1}{1{,}528 \cdot 10^{-2}} \cdot 50 \cdot \frac{1}{3{,}3} \tag{5.2}$$

$$U_{\mathrm{in}} \cdot G(s) = U_{\mathrm{out}} \cdot \frac{1}{1{,}528 \cdot 10^{-2}}$$

$$U_{\mathrm{out}} = U_{\mathrm{in}} \cdot G(s) \cdot 1{,}528 \cdot 10^{-2} \approx \frac{U_{\mathrm{in}} \cdot G(s)}{65{,}45}$$

Um sicherzustellen, dass der Regler im Simulationsmodell die gleiche Verstärkung erfährt wie in der realen Hardware, wird dieser Faktor als Skalierungsfaktor in die Reglerimplementierung aufgenommen. Dadurch werden die Simulationsergebnisse direkt mit den Messwerten der Hardware vergleichbar. Für das Proportionalglied, das mit 0,5 simuliert wurde, ergibt sich somit ein Faktor für den HDL-Coder von $0{,}5 \cdot 1{,}528 \cdot 10^{-2} = 7{,}64 \cdot 10^{-3}$.

6 Entwicklung der Hardware und Software

6.1 Ausgangslage und Vorgehensweise

In dieser Arbeit soll eine hardwarenahe Ansteuerung eines MEMS-Spiegels realisiert werden. Nachdem die Chips gefertigt und vereinzelt wurden, sind sie in der Aufbau und Verbindungstechnik (AVT) mit Epoxidharz auf ein Trägerboard geklebt und anschließend gebondet worden. Der Chip befindet sich auf einem PCB, das mit einem Board-to-Board-Stecker ausgestattet ist. Dies ermöglicht, mehrere Spiegel an derselben Steuerung zu testen. Als Endstufe wird der Falco WMA-02 verwendet, da es sich hierbei um einen gängigen Verstärker für piezoelektrische Aktuatoren handelt. Zunächst wird mit der zur Verfügung stehenden digitalen Hardware eine Steuerung als Demonstrationsobjekt realisiert. Im nächsten Schritt soll eine Closed-Loop-Steuerung auf einem FPGA realisiert werden. Der Einsatz des FPGAs bietet größere Freiräume bei der Entwicklung der Regelung. Der Hauptgrund für den Umstieg auf ein FPGA ist, dass damit der erste Meilenstein in Richtung ASIC gesetzt wird.

6.2 Ansteuerung mit ATmega328P

6.2.1 Der ATmega328P

Zur Ansteuerung des MEMS-Spiegels wurde ein ATmega328P-Mikroprozessor verwendet, der auf einem Arduino-Nano-Board verbaut ist. Der ATmega328P ist ein 8-Bit-Mikrocontroller von Microchip Technology auf Basis der AVR-Architektur. Er verfügt über 32 kB Flash-Speicher, 2 kB SRAM und 1 kB EEPROM. Zu den Peripheriefunktionen zählen Timer, serielle Schnittstellen (UART, SPI, I^2C), analoge Eingänge (ADC) sowie digitale Ein- und Ausgänge (GPIO). Aufgrund seiner geringen Leistungsaufnahme und vielseitigen Einsatzmöglichkeiten eignet sich der ATmega328P besonders für eingebettete Systeme und Mikrocontroller-Anwendungen [17].

6.2.2 Treiberstufe Mikrocontroller

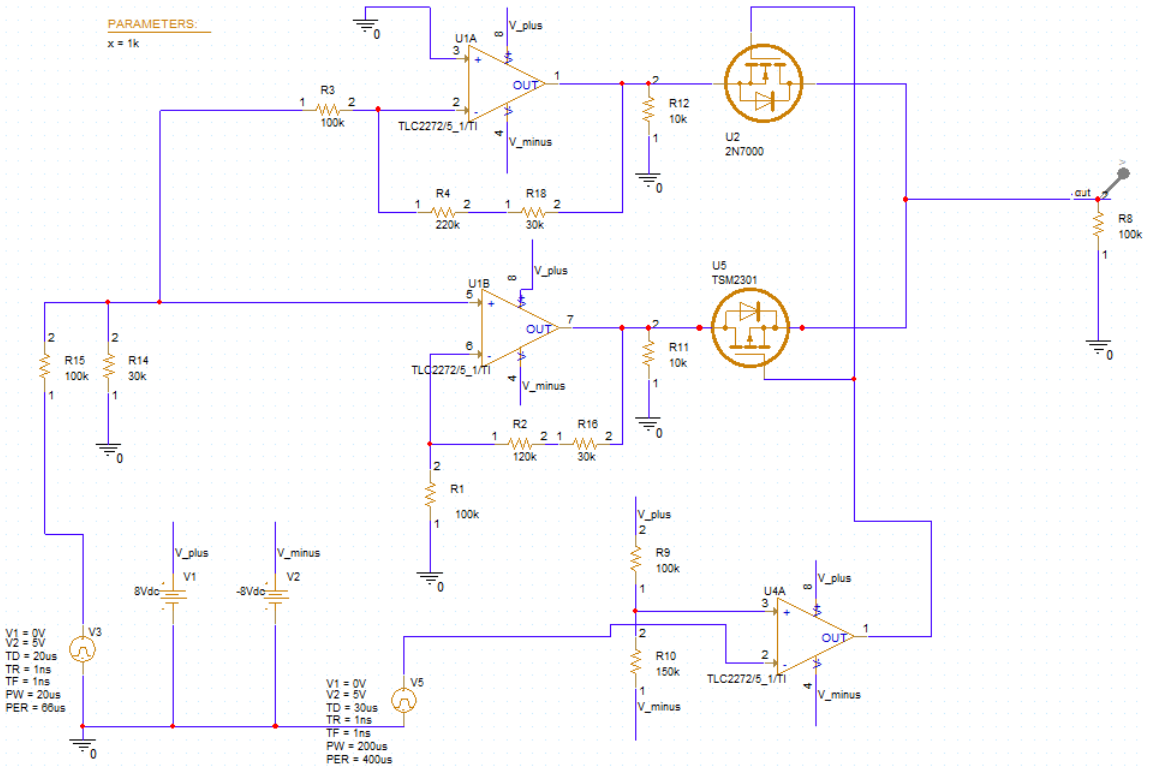

Abbildung 6.1: Treiberstufe für ATmega328P

Die Treiberstufe bildet die Schnittstelle zwischen dem Controller und der Endstufe (Falkon WMA-02) und wird benötigt, um eine Spannung von ±2,5 V bereitzustellen. Digitale Controller liefern digitale Ausgänge mit einer Spannung von 5 V. Die Treiberstufe besteht aus einem invertierenden und einem nichtinvertierenden Verstärker. Je nach benötigter Polarität werden die beiden Verstärkerpfade über einen Feldeffekttransistor geschaltet. Vor der Treiberstufe reduziert ein Spannungsteiler die Ausgangsspannung des Controllers auf unter 1 V, da eine nichtinvertierende Verstärkerschaltung eine Verstärkung größer als 1 aufweist. Die Verstärkungen der beiden Verstärkerschaltungen lassen sich wie folgt berechnen:

Spannungsteiler

$$
\begin{aligned}
U_{in} &= 5\,\mathrm{V} \\
U_{\text{verstärker}} &= U_{in} \cdot \frac{\frac{R_{14} \cdot R_3}{R_{14}+R_3}}{\frac{R_{14} \cdot R_3}{R_{14}+R_3} + R_{15}} \\
U_{\text{verstärker}} &= 5\,\mathrm{V} \cdot \frac{\frac{30\,\mathrm{k\Omega} \cdot 100\,\mathrm{k\Omega}}{30\,\mathrm{k\Omega}+100\,\mathrm{k\Omega}}}{\frac{30\,\mathrm{k\Omega} \cdot 100\,\mathrm{k\Omega}}{30\,\mathrm{k\Omega}+100\,\mathrm{k\Omega}} + 100\,\mathrm{k\Omega}} \\
U_{\text{verstärker}} &= 0{,}9375\,\mathrm{V}
\end{aligned}
$$

nichtinvertierender Verstärker:

$$
V_{\overline{invertierend}} = \frac{R_1 + (R_2 + R_{16})}{R_1} = 1 + \frac{(R_2 + R_{16})}{R_1} = 1 + \frac{(220\,\mathrm{k\Omega} + 30\,\mathrm{k\Omega})}{100\,\mathrm{k\Omega}} = 2{,}5
$$

invertierender Verstärker:

$$
V_{invertierend} = -\frac{R_4 + R_{18}}{R_3} = -\frac{220\,\mathrm{k\Omega} + 30\,\mathrm{k\Omega}}{100\,\mathrm{k\Omega}} = -2{,}5
$$

Die Spiegel zeigen eine hohe Streuung im Verhältnis von Auslenkung zu Spannung, wie in Kapitel 3.1 beschrieben. Um die Schaltung für den Betrieb mehrerer Spiegel anpassen zu können, sind die Widerstände R16 und R18 als Trimmer mit einem Einstellbereich von $0\,\Omega$ bis $100\,\Omega$ ausgelegt. Mit diesen Trimmern lässt sich der Auslenkwinkel auf den maximalen Softwarewert kalibrieren.

Die Auswahl der Polarität der PWM-Signale erfolgt durch den Einsatz zweier Leistungstransistoren, einem N-Kanal-MOSFET (U2, Typ 2N7000) und einem P-Kanal-MOSFET (U5, Typ TSM2301). Diese werden in Abhängigkeit eines Steuersignals selektiv durchgeschaltet oder gesperrt, um den gewünschten Strompfad freizugeben. Zur sauberen Ansteuerung der Gates kommt ein TLC271 Operationsverstärker als Komparator mit Rail-to-Rail-Ausgang zum Einsatz. Liegt am Eingang des Komparators ein logisches High-Pegel (5 V) an, so erzeugt dieser am Ausgang eine Spannung von $+8\,\mathrm{V}$. Diese Spannung wird an die Gates beider MOSFETs angelegt. Der NMOS erreicht dadurch eine Gate-Source-Spannung, die ihn in den leitenden Zustand versetzt, während der PMOS auf-

grund der positiven Gate-Source-Spannung sperrt. In diesem Zustand wird der positive PWM-Zweig aktiviert. Wird der Eingang des Komparators durch den Mikrocontroller auf Masse (0 V) gezogen, so liegt am Ausgang des Komparators eine Spannung von -8 V an. In diesem Fall sperrt der NMOS, während der PMOS durchschaltet, da nun eine negative Gate-Source-Spannung anliegt. Dadurch wird der negative PWM-Zweig freigegeben. Diese Schaltung ermöglicht eine gezielte Auswahl der Stromrichtung durch geeignete Pegelsteuerung und stellt sicher, dass zu jedem Zeitpunkt nur einer der beiden Zweige aktiv ist, um Kurzschlüsse zu vermeiden. Die Simulation in Abbildung 6.2 zeigt das Ausgangssignal der Treiberstufe bei einem PWM-Dutycycle von 30,3 %. Die in Abbildung 6.1 dargestellte Schaltung wird mit einem ATmega328P-Mikrocontroller betrieben. Für die Simulation wurde ein Pulssignal mit einer Amplitude von 5 V und einer Periodendauer von 66 μs, entsprechend einer PWM-Frequenz von 15,15 kHz, angelegt. Der Dutycycle des simulierten PWM-Ausgangs beträgt bei einer eingeschalteten Zeit von 20 μs ebenfalls 30,3 %.

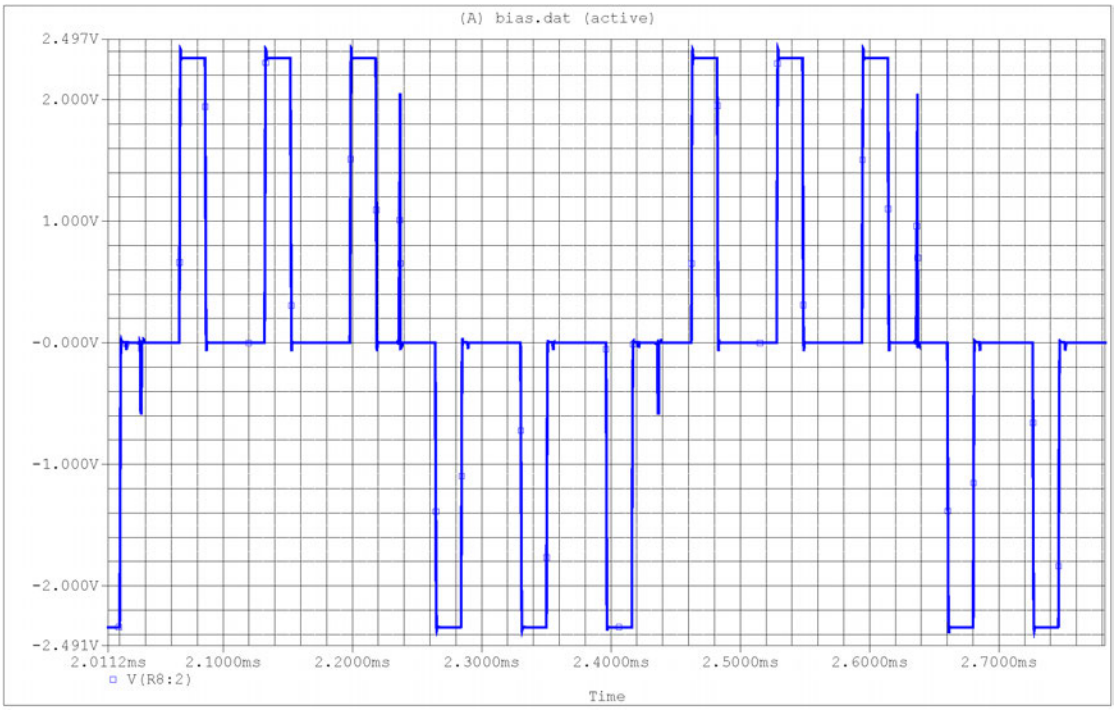

Abbildung 6.2: Simulation Treiberstufe für den ATmega328P

6.2.3 Quellcode ATmega328P

Für die Ansteuerung des MEMS-Spiegels wurde ein Arduino Nano Board verwendet, das mit einem ATmega328P-Mikroprozessor ausgestattet ist. Das Arduino Nano Board arbeitet mit einem Systemtakt von 16 MHz [1].

Zur Steuerung des Spiegels wurden die PWM-Ausgänge der Pins D9 und D10 des Arduino Nano Boards genutzt. Im Anhang B.1 ist der Quellcode für die erste Ansteuerung des MEMS-Spiegels dargestellt. Die PWMs werden in der Funktion `PWM_PBx_init()` initialisiert. Die PWM-Zähler arbeiten ohne Prescaler und besitzen eine Auflösung von 10 Bit. Aus der Auflösung und dem Zähltakt der PWM ergibt sich die PWM-Frequenz von 15,63 kHz, berechnet durch:

$$f_{PWM} = \frac{\text{Zählertakt}}{1 + 2^{\text{Bittiefe}}} = \frac{16\,\text{MHz}}{1 + 2^{10}} \approx 15{,}63\,\text{kHz}, \tag{6.1}$$

wobei der Zählertakt die Frequenz des Systemtakts bezeichnet und die Bittiefe die Auflösung der PWM angibt.

Die Pins D6 und D7 des Boards werden zur Auswahl der Polarität der beiden bipolaren PWM-Signale verwendet. Durch die Treiberstufe aus Kapitel 6.2.2 werden die Signale entsprechend bipolar geschaltet. Durch die Kombination aus Polarititätsumschaltung und 10-Bit-Auflösung der PWMs ergeben sich insgesamt $2^{(1+10)\cdot 2} = 2048 \times 2048$ mögliche Koordinatenpunkte. Damit wird eine ausreichende Auflösung des Laser-Scanbereichs gewährleistet.

Zur Vereinfachung der Programmierung, insbesondere für Demonstrationszwecke, wurde die Funktion `move_to(x_target, y_target)` implementiert. Sie bewegt das System schrittweise von der aktuellen Position (x_{current},y_{current}) zu einem Zielpunkt (x_{target},y_{target}) auf einem zweidimensionalen Gitter. Die Bewegung erfolgt achsenweise: zunächst wird die X-Koordinate angepasst, anschließend die Y-Koordinate. In jeder Achse wird die Position inkrementell oder dekrementell angepasst. Zwischen den Schritten wird eine definierte Zeitverzögerung `step_time` eingehalten. Die Ausgabe an die Register `OCR1A` und `OCR1B` erfolgt zur Ansteuerung der jeweiligen Achse über die PWM-Signale. Über die Funktionen `select_quadrant_x()` und `select_quadrant_y()` wird vorab die Bewegungsrichtung eingestellt.

6.2.4 Vorstellung Demonstratorboard

Das Demonstratorboard wurde auf einer Lochrasterplatine aufgebaut. Die in Kapitel 6.2.2 entwickelte Schaltung wurde dabei zweimal realisiert, sodass jede Achse separat angesteuert werden kann. In Abbildung 6.3 ist der Prototyp zur Ansteuerung dargestellt. Über die vier blauen Trimmer-Potentiometer kann die Verstärkung der einzelnen Verstärkerstufen feinjustiert werden. Das Board wird über eine dreipolige Stiftleiste mit $\pm 8\,\mathrm{V}$ versorgt. Die zweipoligen Stiftleisten dienen dem Debugging, dort kann das PWM-Signal des ATmega328P abgegriffen werden. Die beiden BNC-Ausgänge bilden die Steuersignalausgänge für die beiden Falco WMA-02-Verstärker. Der Mikrocontroller unten links in Abbildung 6.3 ist für die Ansteuerung zuständig; über ihn kann der Spiegel in beliebige Koordinaten ausgelenkt werden. Die Spannungsversorgung des Mikrocontrollers erfolgt über das USB-Kabel.

Die gewählte Lösung ist aus schaltungstechnischer Sicht weniger elegant, erfüllt jedoch die funktionalen Anforderungen zuverlässig. Da der Aufbau als Demonstrator und nicht für den Dauerbetrieb konzipiert wurde, wurde auf eine H-Brücken-Implementierung verzichtet. Für eine zukünftige Optimierung wäre der Einsatz einer H-Brücke mit symmetrischer Treiberlogik empfehlenswert, um den Bauteilaufwand sowie die Leistungsverluste zu reduzieren.

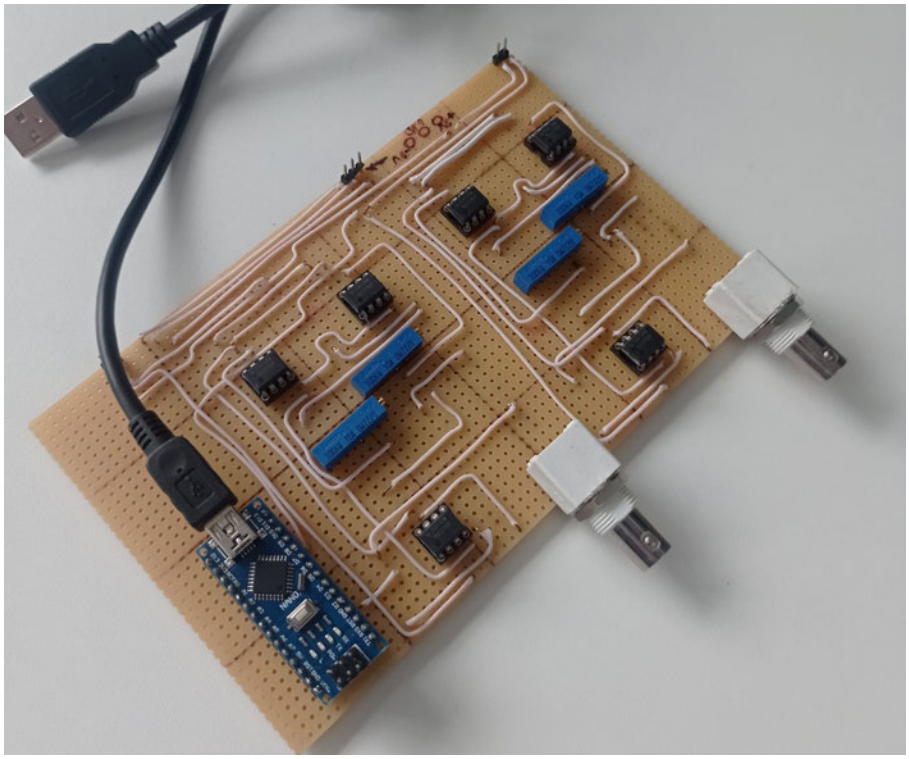

Abbildung 6.3: Demonstratorboard mit ATmeaga328p

6.3 Ansteuerung mittels FPGA

6.3.1 FPGA Auswahl

Für die Umsetzung der Regelung wurde ein FPGA der Artix-7-Familie ausgewählt. Alternativen wie das vorhandene Eclypse Z7 Board erwiesen sich als ungeeignet, da der verfügbare DAC-Bereich nicht passend ist, zu wenige ADC-Kanäle bereitgestellt werden und die Kosten vergleichsweise hoch ausfallen. Zudem ist auf dem Eclypse Z7 ein Zynq-7000 APSoC integriert, dessen eingebundener Softcore die Systemarchitektur zusätzlich verkompliziert, ohne für die geplante Anwendung einen nennenswerten Mehrwert zu bieten.

Der Artix-7 bietet hingegen eine Reihe entscheidender Vorteile. Durch die große Entwicklercommunity und die umfangreiche Dokumentation steht eine breite Wissensbasis für Entwicklung, Implementierung und Fehlersuche zur Verfügung. Mit den integrierten DSP-Modulen lassen sich die benötigten PI- und P-Regler sowie Notch-Filter effizient

in Fixed-Point-Arithmetik umsetzen. Da die Regelung selbst eine vergleichsweise einfache Aufgabe darstellt, sind die verfügbaren Ressourcen und die Taktfrequenz des Artix-7 mehr als ausreichend. Gleichzeitig bleibt ausreichend Puffer für zukünftige Erweiterungen oder komplexere Filterstrukturen.

Praktische Aspekte sprechen ebenfalls für den Einsatz des Artix-7: Das FPGA ist schnell verfügbar, kostengünstig und wird von einer ausgereiften Entwicklungsumgebung (Vivado) unterstützt. Passende DAC-Boards sind verfügbar, sodass die Anbindung der Peripherie unkompliziert erfolgen kann. Insgesamt bietet der Artix-7 damit eine robuste und zukunftssichere Plattform mit hoher Ressourcensicherheit für die vorgesehene Anwendung.

6.3.2 Ausgang

DAC Board

Für das Projekt werden zwei analoge Ausgänge benötigt, die mit derselben Frequenz Werte ausgeben sollen, mit der der ADC Daten aufnimmt. Dadurch wird die Berechnung von Filtern und Reglern übersichtlicher. Die Auflösung des DAC sollte dabei der des Eingangs entsprechen. Es hat sich angeboten, ein Erweiterungsboard der Firma Digilent mit zwei DAC zu verwenden. Das Pmod DA2-Board besitzt zwei analoge Ausgänge, und die Anschlussbelegung ist kompatibel zu den 6-poligen Pmod-Steckern am Basys 3. Auf dem Pmod DA2-Board sind zwei DAC121S101 von Texas Instruments verbaut. Die DACs können mit einem SPI-Takt von bis zu 30 MHz betrieben werden, was eine maximale Abtastrate von

$$f_{s,max} = \frac{30\,\mathrm{MHz}}{16} \approx 1{,}875\,\mathrm{MHz}$$

möglich macht.[21]

Für dieses Projekt wird der SPI-Takt auf 800 kHz begrenzt, sodass dieselbe Clock-Domäne wie beim ADC genutzt werden kann. Der DAC121S101 arbeitet, ebenso wie der verbaute ADC, mit einem 16-Bit-Frame. Dadurch lassen sich beide Bausteine bei gleicher SPI-Taktfrequenz mit identischer Geschwindigkeit betreiben.

Treiberstufe DAC

Die beiden DACs können eine Ausgangsspannung von 0 V bis 3,3 V liefern. Die Endstufe benötigt jedoch eine Spannung von ±2,5 V. Diese wird mithilfe eines Treiber-Boards erzeugt, das aus einer Subtrahiererschaltung mit einer Verstärkung von 1,515 realisiert ist. Die Schaltung wird mit einer Versorgungsspannung von ±8 V betrieben. Die konstante Spannung, die subtrahiert wird, wird durch einen TL431-Shunt-Regler bereitgestellt. In Abbildung 6.4 ist die Schaltung eines Kanals der Treiberstufe dargestellt.

Die konstante Spannung ergibt sich zu:

$$U_{\text{konst. Spannung}} = 1 + \left(\frac{R3}{R2}\right) \cdot U_{\text{REF}} \qquad \text{vgl. [5, S.2]} \tag{6.2}$$

$$U_{\text{konst. Spannung}} = \left(1 + \frac{3{,}3\,\text{k}\Omega}{10\,\text{k}\Omega}\right) \cdot 1{,}24\,\text{V} \tag{6.3}$$

$$U_{\text{konst. Spannung}} = 1{,}649\,\text{V} \tag{6.4}$$

Der zweite Teil der Schaltung ist ein Subtrahierer mit definiertem Verstärkungsfaktor, der den Spannungsbereich für die Endstufe anpasst. Der DAC liefert maximal 3,3 V. Damit bei einer zwanzigfachen Verstärkung eine Ausgangsspannung von ±50 V in der Endstufe erreicht wird, muss das Treibersignal $5\,\text{V}_{\text{pp}}$ liefern. Da der Operationsverstärker in der Schaltung mit einer Versorgungsspannung von ±8 V betrieben wird, bietet sich der TL051 für den Einsatz an. Der TL051 verfügt über einen weiten Versorgungsspannungsbereich bis etwa ±18 V und über Offseteingänge (N1, N2), die sich zum Trimmen der Offsetspannung eignen [23].

Die erforderliche Verstärkung ergibt sich aus den gegebenen Spannungen, nämlich $5\,\text{V}_{\text{pp}}$ am Eingang und 3,3 V am DAC:

$$V = \frac{U_{\text{out}}}{U_{\text{in}}} = \frac{5\,\text{V}}{3{,}3\,\text{V}} = 1{,}515\,.$$

Daraus folgen die Widerstandsverhältnisse:

$$R_8 = 100\,\text{k}\Omega,$$
$$R_5 = (V-1) \cdot R_8 = (1{,}515-1) \cdot 100\,\text{k}\Omega = 51{,}5\,\text{k}\Omega \approx 47\,\text{k}\Omega.$$

Die beiden Offseteingänge N2 und N1 in Abbildung 6.4 sind mit einem 5 kΩ-Potentiometer und je 10 kΩ in Serie realisiert. Mit dieser Anordnung lässt sich der Offset fein einstellen. Der 100 kΩ-Widerstand am Ausgang sorgt für einen definierten Abschluss, sodass der Ausgang stets eine Last sieht.

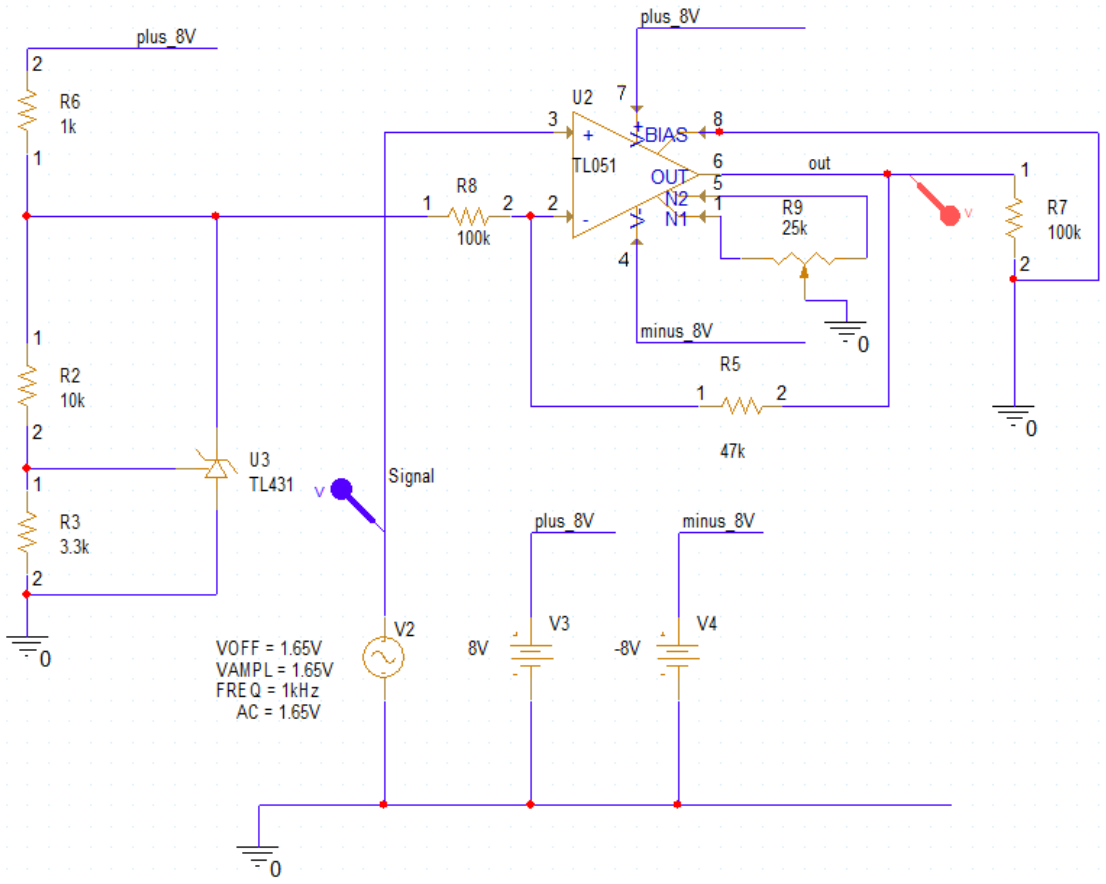

Abbildung 6.4: Schematik von einem Kanal der Treiberstufe

In Abbildung 6.5 ist das Simulationsergebnis der Treiberstufenschaltung aus Schematik 6.4 dargestellt. Die Schaltung wurde mit einem 1 kHz-Sinussignal im Bereich von 0 bis 3,3 V (blau) simuliert. Die resultierende Ausgangsspannung (rot) zeigt ein bipolares Sinussignal. Kleine Offsetungenauigkeiten können später mit dem Potentiometer zwischen N1 und N2 des TL051-Operationsverstärkers ausgeglichen werden. Der Widerstand R_5 wurde etwas kleiner gewählt als es die Berechnung vorgibt, weshalb die Verstärkung leicht reduziert ist. Die erzielte Verstärkung von 1,47 ist für die Ansteuerung ausreichend

und bietet zudem eine Sicherheitsreserve, da die maximale Aktuatorspannung auch bei Toleranzen in der Chipfertigung nicht zur Zerstörung führt.

Im Anhang D.1 ist das Layout des DAC-Treiberboards dargestellt. Da das Board mittels Computerized Numerical Control (CNC)-Fräse hergestellt wurde, befindet sich die Bauteilbeschriftung auf der Kupferlage. Als Anschlussbuchsen wurden BNC-Buchsen gewählt, da sie eine einfache Kontaktierung ermöglichen und für die hier relevanten Frequenzen unter 200 MHz ausreichend sind.

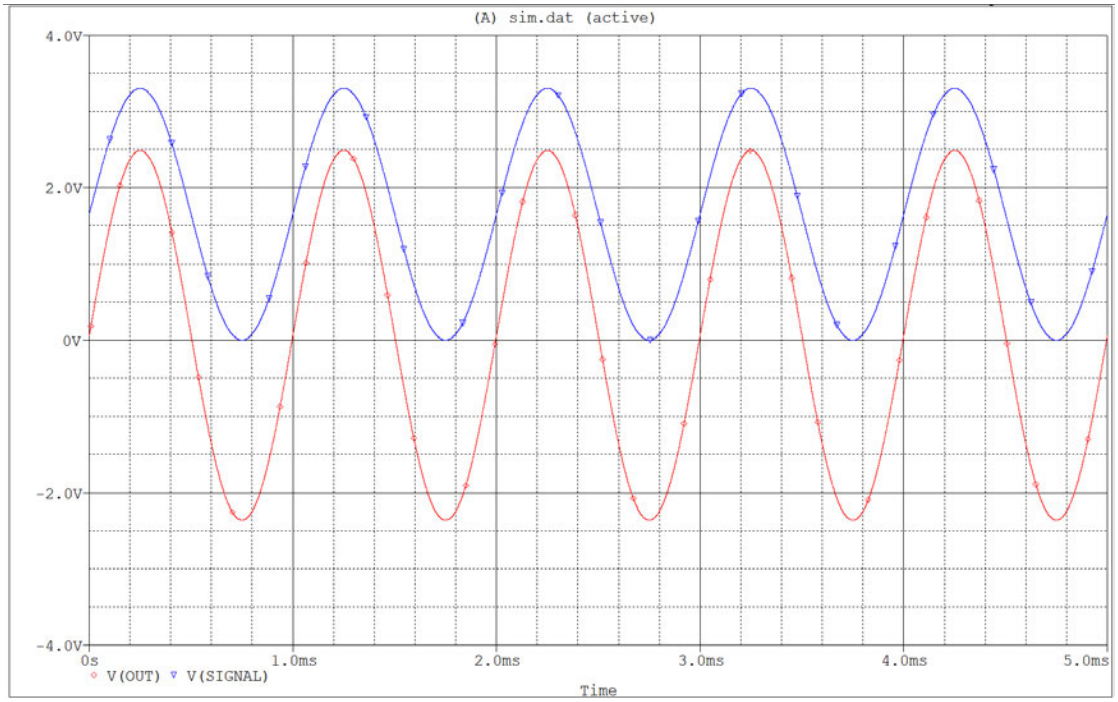

Abbildung 6.5: Simulationsergebnisse eines Kanals der Treiberstufe

Endstufe

Da die Spiegel quasistatisch angesteuert werden sollen, liegt die erforderliche Spannung für die Aktuatoren bei ±50 V. Diese hohe Spannung wird mit einem verfügbaren Falkon WMA-02 erreicht. Der Verstärker besitzt eine feste Verstärkung von 20 und wird häufig für piezoelektrische Aktuatoren eingesetzt. Er verfügt über drei BNC-Buchsen: einen Anschluss für die Versorgung (24 Vdc), einen Eingang und einen Ausgang. Die Eingangsim-

pedanz beträgt 100 kΩ, die Ausgangsimpedanz 50 Ω. Die maximale Ausgangsspannung liegt bei ±175 V. Die Bandbreite reicht von DC bis 13 kHz [6].

Da dieser Verstärker verfügbar war und sowohl die geforderte Ausgangsspannung von ±50 V als auch die notwendige Bandbreite von mehr als 5 kHz erfüllt, wurde er in diesem Projekt eingesetzt.

6.3.3 Eingang

Analog-Digital-Wandler-Platine

In der Regelungstechnik wird häufig eine Faustregel angewandt, nach der die Abtastrate einer Regelung mindestens zehnmal höher sein sollte als die Frequenz des schnellsten zu erfassenden Ereignisses. Das schnellste Ereignis, das von der Regelung erfasst werden muss, ist die Resonanz der Hubmode bei etwa 500 Hz. Daraus ergibt sich eine notwendige Abtastrate von mindestens 5 kHz.

Für die Regelung eignet sich ein Successive Approximation Register (SAR)-ADC besonders gut, da dieser nur eine sehr geringe Totzeit aufweist. Ein SAR-ADC tastet das analoge Eingangssignal einmal ab, hält es mittels einer Sample-&-Hold-Schaltung konstant und ermittelt anschließend das digitale Ergebnis Bit für Bit. Nach dem Aktivieren des Chip-Select-Signals wird die Eingangsspannung festgehalten. Anschließend erzeugt der interne DAC sukzessive Vergleichsspannungen, die über einen Komparator mit der gespeicherten Eingangsspannung verglichen werden. So wird schrittweise entschieden, ob die Eingangsspannung größer oder kleiner als die jeweilige Vergleichsspannung ist, bis das digitale Ergebnis vollständig bestimmt ist. Der Vorteil dieses Verfahrens liegt in der kurzen Wandlungszeit und darin, dass keine nachträgliche Signalfilterung erforderlich ist.

Aus dem Datenblatt des MCP3201 [18] gilt:

$$f_{\mathrm{S,max}}(V_{DD} = 5{,}0\,\mathrm{V}) = 100\,\mathrm{ksps},$$
$$f_{\mathrm{S,max}}(V_{DD} = 2{,}7\,\mathrm{V}) = 50\,\mathrm{ksps}.$$

Da die Versorgungsspannung des Basys 3-Boards $V_{DD} = 3{,}3\,\mathrm{V}$ zwischen diesen beiden Punkten liegt, kann die maximale Abtastrate durch lineare Interpolation abgeschätzt werden, falls für spätere Anwendungen eine höhere Rate notwendig wird. Im weiteren

Verlauf wird eine Abtastrate von 50 kHz zugrunde gelegt. Die Ein- und Ausgänge werden in diesem Projekt durch die Berechnungslogik auf 20 kHz beschränkt, was deutlich oberhalb der Anforderung von 5 kHz liegt. Durch diese konservative Abtastrate besteht zudem die Möglichkeit, ein Zeitmultiplexing zwischen den Samples zu realisieren.

Der MCP3201 benötigt pro Konversion 16 Taktzyklen:

$$f_{\mathrm{SPI}} = 16 \cdot f_{\mathrm{S,max}}$$
$$f_{\mathrm{SPI}} \approx 16 \cdot 50\,\mathrm{ksps} \approx 0{,}8\,\mathrm{MHz}$$

Der MCP3201 erfüllt alle Anforderungen und ist ein kostengünstiger, verfügbarer SAR-ADC. Bei einer Versorgungsspannung von 3,3 V erreicht der MCP3201 eine maximale Abtastrate von 50 kHz sowie einen maximalen SPI-Takt von 0,8 MHz. Damit wird die FSM für die Bitübertragung mit 800 kHz getaktet [18]. In Anhang C.1 ist der VHDL-Code für den MCP3201 dargestellt.

Für das PCB der Regelung werden vier ADC benötigt, jeweils einer pro Aktuator. Das PCB ist so ausgelegt, dass alle vier ADCs mit der zugehörigen Peripherie auf einem Board Platz finden. Es wurden vier separate ADCs eingesetzt, damit die Analog-Digital-Wandlung parallel erfolgen kann. Nahe bei den Chips ist jeweils ein 100 nF-Kondensator als Energiepuffer verbaut, um Spannungseinbrüche zu verhindern. In die Taktleitung wurde ein 47 Ω-Widerstand eingesetzt, um bei einer möglichen Erhöhung der SPI-Frequenz Reflexionen zu vermeiden. Die Eingänge der ADC sind mit BNC-Buchsen ausgeführt, sodass das Vorverstärkerboard nah am MEMS-Spiegel platziert werden kann und für die Signalführung geschirmte BNC-Kabel verwendet werden.

Im Anhang D.2 ist das Layout des im Rahmen dieser Arbeit gefertigten und bestückten PCBs dargestellt. Das Board wurde mit einer CNC-Fräse hergestellt. Zur Anpassung der Steckerkontakte wurde zudem ein kleines Adapterboard eingesetzt, sodass die Signale korrekt mit dem Basys 3-Board verbunden werden können. Das Layout dieses Adapterboards ist ebenfalls im Anhang D.3 gezeigt.

Signalverstärker

Zur Signalverstärkung der piezoelektrischen Sensorelemente wurde der INA333 ausgewählt. Der hochohmige Eingang ist besonders gut geeignet für die Messung der kleinen

Spannungen. Der INA333 wurde gemäß der im Datenblatt beschriebenen Single-Supply-Applikationsschaltung (vgl. Abbildung [22, S.18]) beschaltet. Anstelle der dort angegebenen Werte von 300 Ω und 150 Ω wurden Widerstände mit 2 MΩ bzw. 1 MΩ eingesetzt. In Abbildung 6.6 ist die Messbrücke durch die Widerstände R_{10} bis R_{14} realisiert. Bei der Auswahl der Widerstandsverhältnisse wurde sich am eingestellten Innenwiderstand des Oszilloskops orientiert. Der Eingangskreis wurde so dimensioniert, dass sich aus Sicht des piezoelektrischen Sensorelements ein Eingangswiderstand von 1 MΩ ergibt.

Dies reduziert den Strom durch die Messbrücke erheblich und damit die Verlustleistung. Gleichzeitig wird so sichergestellt, dass die von den piezoelektrischen Sensorelementen bereitgestellte sehr geringe Ladungsmenge ausreicht, um eine messbare Spannung am Verstärkereingang aufzubauen.

Die Versorgungsspannung des INA333 beträgt 3,3 V. Die Referenzspannung wurde auf die halbe Versorgungsspannung (1,65 V) gelegt, damit das Ausgangssignal symmetrisch um diesen Arbeitspunkt schwingen kann. Zur Erzeugung der Referenzspannung wird eine AZ431L-Shuntdiode verwendet. Die Widerstände zur Einstellung der konstanten Referenzspannung von 1,65 V entsprechen denen aus Kapitel 6.3.2. Die Berechnung in Gleichung 6.3 beschreibt die Zusammenhänge der Widerstandsverhältnisse auch für die in Abbildung 6.6 gezeigte Schaltung.

Der INA333 erlaubt eine präzise Einstellung des Verstärkungsfaktors im Bereich von 1 bis 1000 über den externen Widerstand R_8. Der Verstärkungsfaktor wurde auf $G = 51$ eingestellt, sodass die in Abschnitt 3.3.1 bestimmten 30 $\mathrm{mV_{pp}}$ auf etwa 1,53 $\mathrm{V_{pp}}$ verstärkt werden.

Der Gain-Widerstand R_8 berechnet sich nach folgender Formel

$$G = 1 + \left(\frac{100\,\mathrm{k\Omega}}{R_8}\right) \qquad \text{vlg. [22, S.1]}$$

$$R_8 = \frac{G - 1}{100\,\mathrm{k\Omega}}$$

$$R_8 = \frac{51 - 1}{100\,\mathrm{k\Omega}}$$

$$R_8 = 2\,\mathrm{k\Omega}$$

Vor dem Aufbau der Schaltung ist sie mit Cadence OrCad simuliert worden. Dabei wurde eine von der Firma TexasInstrument zur Verfügung gestelltes PSpice Modell ver-

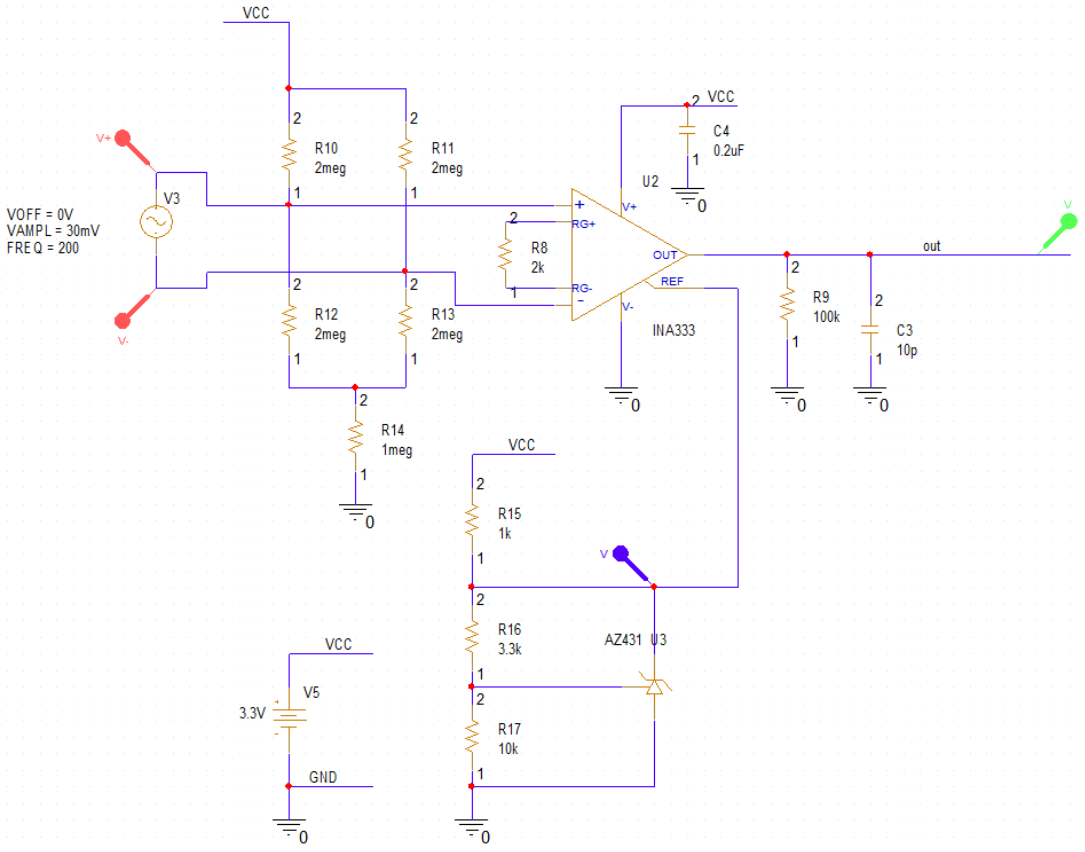

Abbildung 6.6: Schematik der Verstärkerschaltung für piezoelektrische Sensorelemente.

wendet. Die Simulation in Abbildung 6.7 zeigt das erwartete Verhalten im kompletten Dynamikbereich. Zur Simulation wurde die Messbrücke mit einem Sinussignal mit $60\,\mathrm{mV_{pp}}$ ohne Offset versorgt, in Abbildung 6.7 in rot zu sehen. In blau ist die konstante Referenzspannung vom AZ431L zu sehen. In grün der Ausgang des INA333. Die simulierte Spitzenspannung des Verstärkerausgangs deckt sich mit den Berechnungen: $30\,\mathrm{mV} \cdot 51 + 1{,}649\,\mathrm{V} = 3{,}179\,\mathrm{V}$ genau so die Bulldurchgänge bei $1{,}649\,\mathrm{V}$ und die minimalspannung von $1{,}649\,\mathrm{V} - 30\,\mathrm{mV} \cdot 51 = 0{,}119\,\mathrm{V}$.

Die Schaltung wurde für jeden Sensor identisch aufgebaut. Der Schaltungsteil zur Erzeugung der Referenzspannung ist lediglich einmal auf der Leiterplatte realisiert und wird von dort aus allen Verstärkerschaltungen zugeführt. Als Ausgangsanschluss wurde auch hier eine BNC-Buchse verwendet, da diese eine störsichere, abgeschirmte Verbindung ermöglicht und eine einfache Anbindung an gängige Messgeräte erlaubt. Die Verbindung

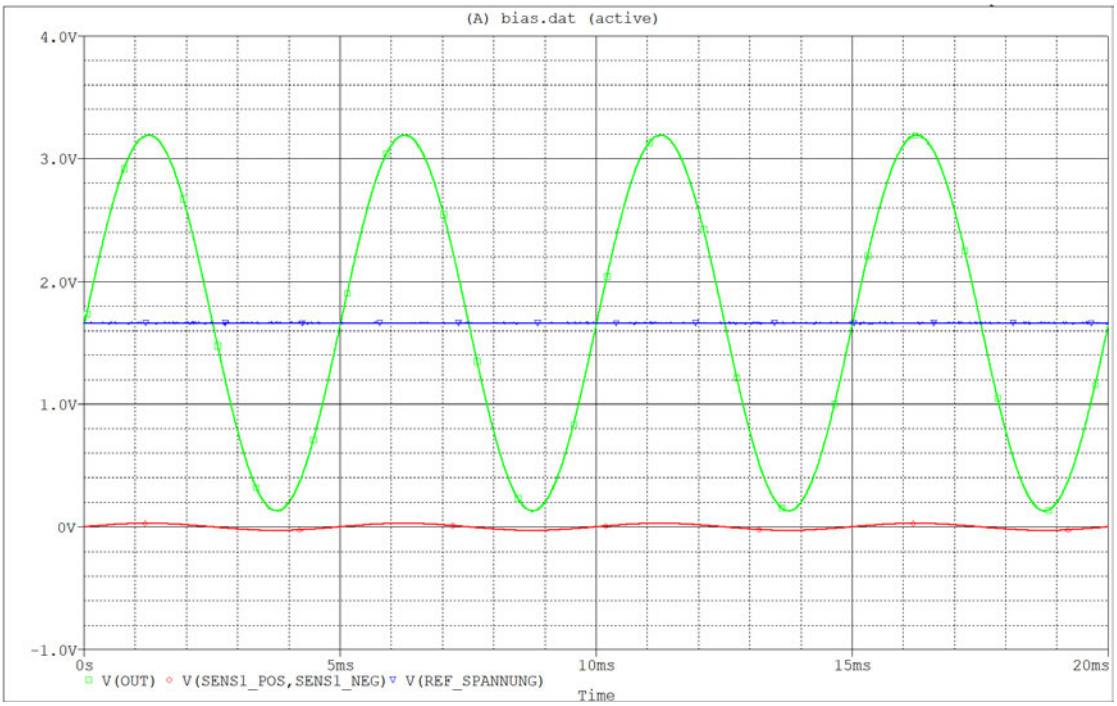

Abbildung 6.7: Simulation der Verstärkerschaltung für piezoelektrische Sensorelemente.

zur Leiterplatte, auf der sich der Chip befindet, wurde über eine zweireihige Stiftleiste im 2,54 mm-Raster realisiert, da diese dort bereits vorgesehen war.

Das PCB wurde mit einer CNC-Fräse gefertigt. Da die CNC-Fräse keine 8-VSSOP-Strukturen herstellen konnte, wurde ein Adapterboard von 8-VSSOP auf DIP-8 eingesetzt, um den INA333 in der Schaltung nutzen zu können. Im Anhang D.2 ist das fertige Layout dargestellt. 1

6.3.4 HDL - Testmodule

UART-Schnittstelle

Zum Testen von Teilsystemen wurde eine UART-Schnittstelle entwickelt. Diese Schnittstelle ist zwar in der Bandbreite limitiert, da die UART-Verbindung bei einer sicheren Baudrate vergleichsweise langsam ist, eignet sich jedoch gut für Debugzwecke. Im Gegensatz zu schnelleren Schnittstellen wie SPI oder USB ist UART einfacher zu implementieren und ermöglicht eine zuverlässige Datenübertragung mit geringem Hardwareaufwand.

Dadurch können Datenströme aus dem FPGA in Echtzeit geplottet, geloggt und gespeichert werden, ohne die Komplexität des Gesamtsystems wesentlich zu erhöhen.

Im FPGA werden die Daten im 12-Bit-Format verarbeitet, was größer ist als das von der UART unterstützte 8-Datenbit-Format. Gängige PCs erlauben maximal 9 Bit, wenn das Paritätsbit als Datenbit genutzt wird [7]. Daher ist es notwendig, die Übertragung auf zwei Übertragungsrahmen aufzuteilen. Das Übertragungsmodul im FPGA besteht aus einem Hauptmodul und einem 8-Bit-Sendermodul, das die eigentliche Übertragung durchführt. Um die Samplerate klar zu definieren und flexibel an den jeweiligen Datenstrom anpassen zu können, verfügt das Modul über einen externen Triggereingang. Für die USB-LED ist zudem ein Ausgang vorgesehen, der den Betrieb signalisiert. Im Anhang C.3 ist der Quellcode des Hauptmoduls dargestellt.

Das Modul ist als Zustandsmaschine realisiert, die das 8-Bit-Sendermodul, dargestellt in Anhang C.4, zweimal aufruft. Im ersten Rahmen werden die LSBs übertragen, im zweiten die vier MSBs. Die übrigen vier Bits im zweiten Rahmen werden auf „0” gesetzt, können jedoch bei Bedarf mit geringem Aufwand aktiviert werden.

Die Limitation des Moduls ergibt sich aus der Baudrate von 115 200 Bd. Daraus ergibt sich die Samplerate:

$$\frac{115200\,\text{Baud}}{2 \cdot 10\,\text{Bit}} \approx 5{,}75\,\text{kHz}.$$

Höhere Baudraten sind möglich, erhöhen jedoch die Anfälligkeit für Bitfehler. Da Signale bis maximal 500 Hz erwartet werden, wird das interne Signal mit 2 kHz abgetastet, was für Debugzwecke ausreichend ist.

Auf der PC-Seite werden die Daten mit einem Python-Skript eingelesen und dargestellt. Die Ausgabe erfolgt sowohl in der Konsole als auch in einer Live-Darstellung. Da die FIFO-Puffer des UART-Controllers nur wenige Zeichen fassen und es bei langsamer Abfrage zu Overrun-Fehlern kommt, wurde ein separater Thread implementiert. Dieser liest die FIFO kontinuierlich aus und schreibt die Daten in einen Ringpuffer im Hauptspeicher. Damit wird sichergestellt, dass keine Daten verloren gehen und die weitere Verarbeitung unabhängig von den Echtzeitanforderungen der UART-Schnittstelle erfolgen kann. Der Python-Quellcode ist im Anhang E.1 dargestellt.

Sinusgenerator

Das zweite Testmodul wurde zur Verifikation der Ausgangsstufe entwickelt. Es besteht aus einer Tabelle mit 256 Werten, die nacheinander an die DAC ausgegeben werden. Die einzelnen Sample-Werte wurden mit einem Python-Skript generiert und fest im Modul gespeichert. Die Frequenz des ausgegebenen Sinussignals ergibt sich aus der Triggerfrequenz des Moduls und der Länge der Tabelle, vorausgesetzt, diese entspricht genau einer Periode. Mit der Länge der Tabelle lässt sich somit die Auflösung des Sinussignals einstellen. Eine Auflösung von 256 Werten pro Periode ist für die Anwendung ausreichend. Die Frequenz des Sinussignals lässt sich mit Gleichung 6.5 berechnen:

$$f_{\text{sin}} = \frac{f_{\text{trig}}}{256} \tag{6.5}$$

Die Tabelle kann bei Bedarf durch beliebige andere Funktionsverläufe ersetzt werden, die ebenfalls aus 256 Werten bestehen. Für das Testmodul ist die Sinusfunktion besonders geeignet, da ihr Mittelwert in der Mitte des DAC-Wertebereichs liegt und das Signal keine Sprünge enthält. Dadurch wird beim testweisen Ansteuern des Spiegels vermieden, dass Resonanzen angeregt werden. Das Testmodul könnte ressourcenschonender implementiert werden, beispielsweise durch die Nutzung von Quarter-Wave-Symmetrie, bei der nur 64 Werte erforderlich wären. Da jedoch ausreichend Ressourcen zur Verfügung stehen und es sich lediglich um ein Testmodul handelt, wurde auf eine Optimierung verzichtet. Das Modul ist im Anhang C.5 dargestellt.

6.4 Gesamtsystem mit Regler

In Abbildung 6.8 ist der Signallaufplan des Gesamtsystems dargestellt. Im Design wird die Proportionalregelung achsweise vorgenommen, wobei die Sensoren 1 und 3 der Y-Achse zugeordnet werden und die Sensoren 4 und 2 der X-Achse entsprechen. Der Sensorverstärker verstärkt das Sensorsignal und fügt einen Offset hinzu, sodass der Dynamikbereich des ADCs gut ausgenutzt wird. Zum Verifizieren und Debuggen ist eine UART-Schnittstelle implementiert, die jedes Signal im FPGA ausgeben kann. In Abbildung 6.8 wird beispielhaft das Eingangssignal des Sensors 1 ausgegeben. Die Addition und der Proportionalregler wurden mit dem HDL-Coder von Simulink erstellt und können durch verschiedene Regelungs- und Steuerungsmethoden ersetzt werden. Durch die achsweise Ansteuerung werden nur zwei Treiber- und Endstufen benötigt. Durch den

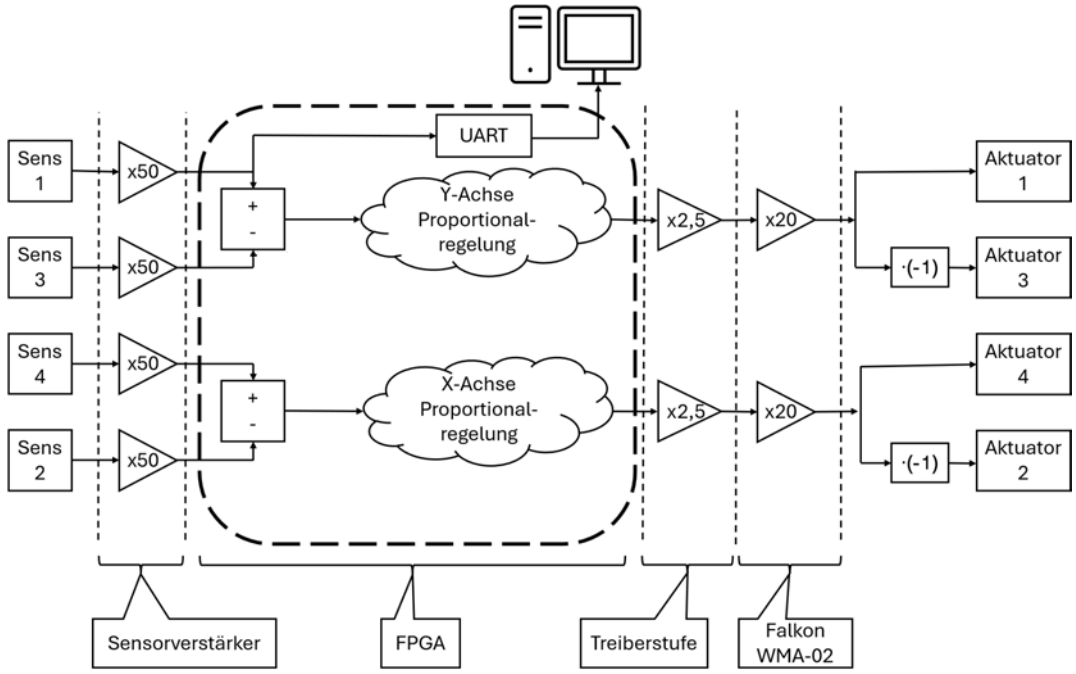

Abbildung 6.8: Signallaufplan des Gesamtsystems

verringerten Hardwareaufwand ist es jedoch nicht möglich, die den Achsen zugeordneten Aktuatoren unabhängig voneinander in die gleiche Richtung anzusteuern. Die Invertierung des Ausgangssignals wird durch Vertauschen der Elektroden des AlScN erreicht. Im Anhang C.1 ist das gesamte HDL-Blockdesign dargestellt, dort sind die einzelnen IP-Blöcke miteinander verbunden und bilden das Gesamtsystem. Jeder Intellectual Property Block (wiederverwendbarer Funktionsbaustein) (IP-Block) in diesem System wurde eigens entwickelt.

7 Verifikation

7.1 Verifikation Ansteuerung mittels ATmega328P

Die Open-Loop-Steuerung, welche mit dem Mikroprozessor realisiert wurde, wurde mit dem in Abbildung 7.1 dargestellten Versuchsaufbau getestet. Dabei kamen verschiedene Scanmuster zum Einsatz, die in der untenstehenden Abbildung dargestellt sind. Mit dem Befehl `move_to(x_target, y_target)` lassen sich, wie in Kapitel 6.2.3 beschrieben, alle Koordinaten anfahren. In der Funktion ist eine Zeitverzögerung implementiert, welche das Zeitintervall zwischen den Inkrementen der Zwischenschritte der Koordinaten definiert. Es hat sich gezeigt, dass die Steuerung mit einer Intervallzeit von 20 µs noch stabil alle dargestellten Scanmuster ausführen kann. Bei einer Intervallzeit von weniger als 15 µs wird der Spiegel instabil, und die Koordinaten werden nicht mehr sauber angefahren. Das quadratische Scanmuster in Abbildung 7.1 zeigt eine TOSA von 2°. Die Verzerrung der Scanmuster resultiert aus der Perspektive, da die Aufnahmen nicht frontal aufgenommen wurden, damit der Laserstrahl nicht unterbrochen wird.

In Abbildung 7.2 ist das Ausgangssignal für das quadratische Scanmuster dargestellt. Während Kanal 2 ein sauberes, trapezförmiges Signal liefert, zeigt Kanal 1 statt der erwarteten konstanten Spannung einen entladeartigen Verlauf. Dieses Verhalten deutet auf einen kleinen Leckstrom oder eine kapazitive Entladung im Ausgangspfad hin. Da das Scanmuster dennoch ein nahezu ideales Quadrat ergibt und die grundlegende Funktion erfüllt wird, wurde auf eine genauere Analyse verzichtet. Beide Kanäle des Demonstrators sind identisch aufgebaut, wie in Abbildung D.5 zu sehen ist. Daher funktioniert das Prinzip grundsätzlich, und die Abweichung in Kanal 1 ist vermutlich auf einen Fertigungsfehler oder eine Unsauberkeit im handgelöteten Aufbau der Treiberstufe zurückzuführen.

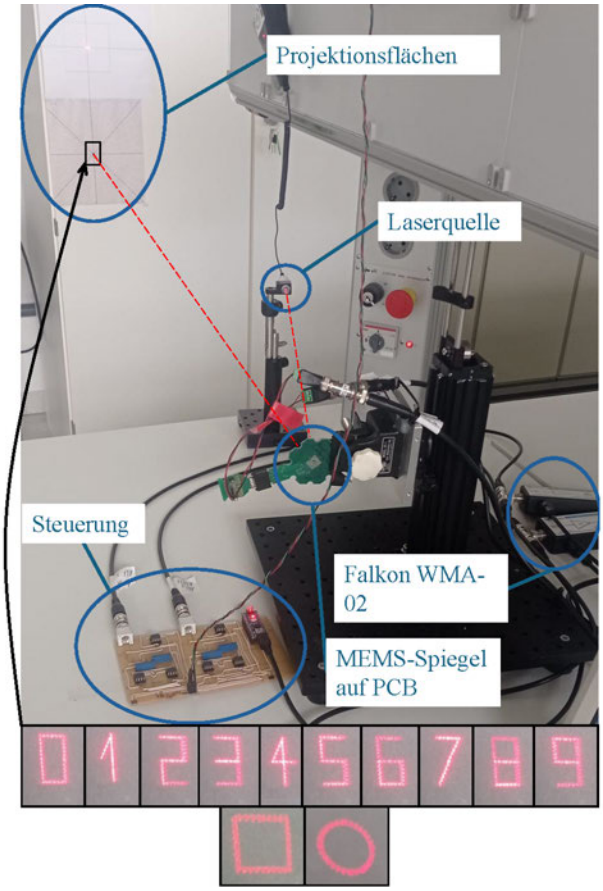

Abbildung 7.1: Testaufbau zur Verifikation der Steuerung mittels ATmega328P, unten im Bild sind Beispielscanmuster gezeigt

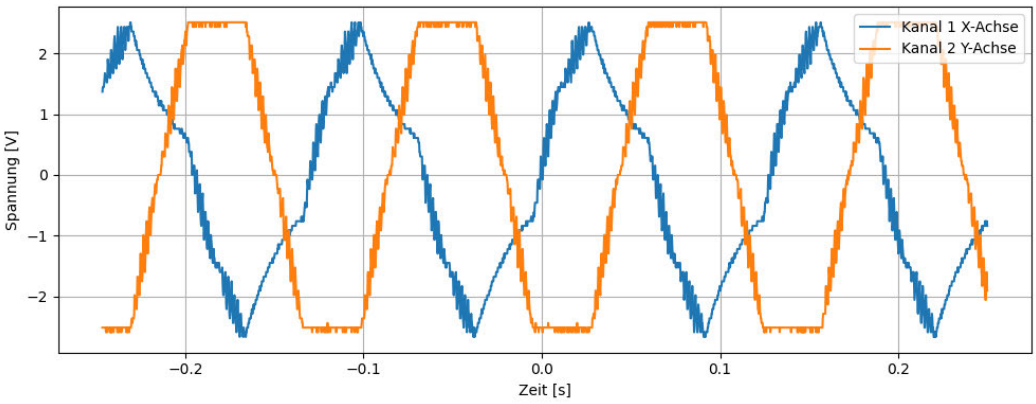

Abbildung 7.2: Ausgangssignal der Steuerung mittels ATmega328P, mit diesem Signal wird das Quadrat aus Abbildung 7.1 erzeugt

7.2 Verifikation Regelung mittels FPGA

7.2.1 Experimentalaufbau

Der Experimentalaufbau der Regelung mittels FPGA ist in Abbildung 7.3 dargestellt. Der Aufbau mit dem Fallhammer dient ersten Vorversuchen, um Parameter und Regelungsstrategien vorab zu validieren. Da die Nutzung des Schütteltisches zeitlich begrenzt ist, wird dieser nur für die finale Verifikation eingesetzt. In Abbildung 7.3 sind die verbundenen Teilkomponenten dargestellt. Dabei wurde darauf geachtet, dass die Sensorverstärker möglichst nahe am Chip positioniert sind, ohne jedoch mechanisch mit diesem verbunden zu sein, um das Einbringen mechanischer Spannungen in das System zu vermeiden. Nicht ersichtlich, aber vorhanden, sind die beiden *Falco WMA-02*, die sich zwischen der Treiberstufe und dem Spiegel befinden und im Aufbau hinter diesen angeordnet sind.

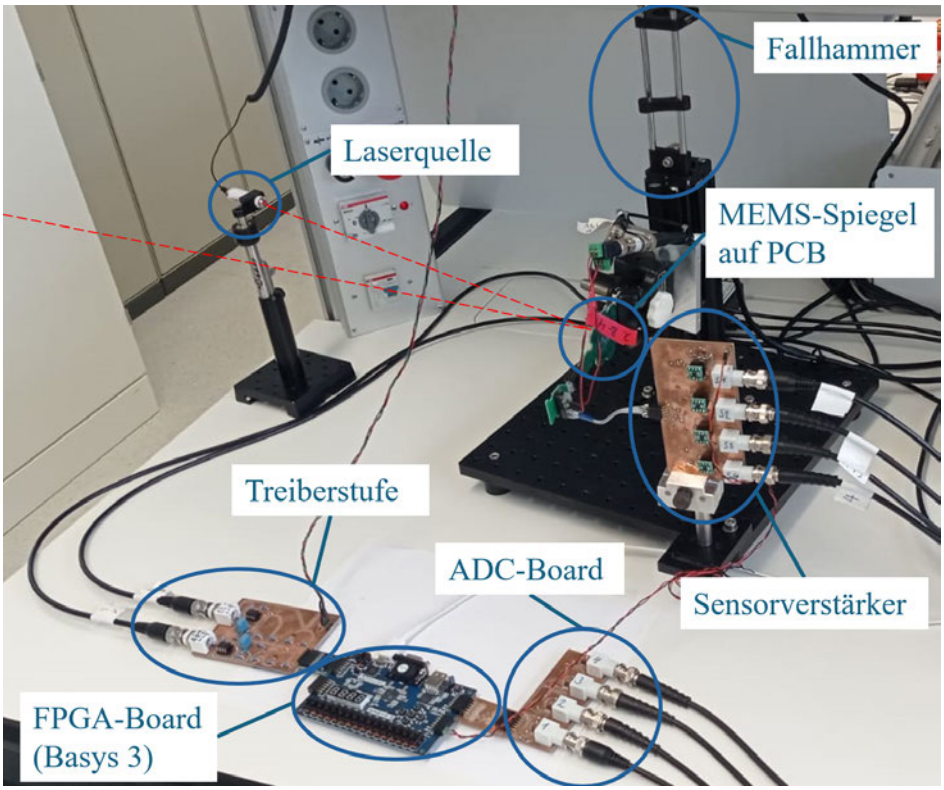

Abbildung 7.3: Experimentalaufbau Regelung mittels FPGA

7.2.2 Verifikation Eingang als Teilsystem

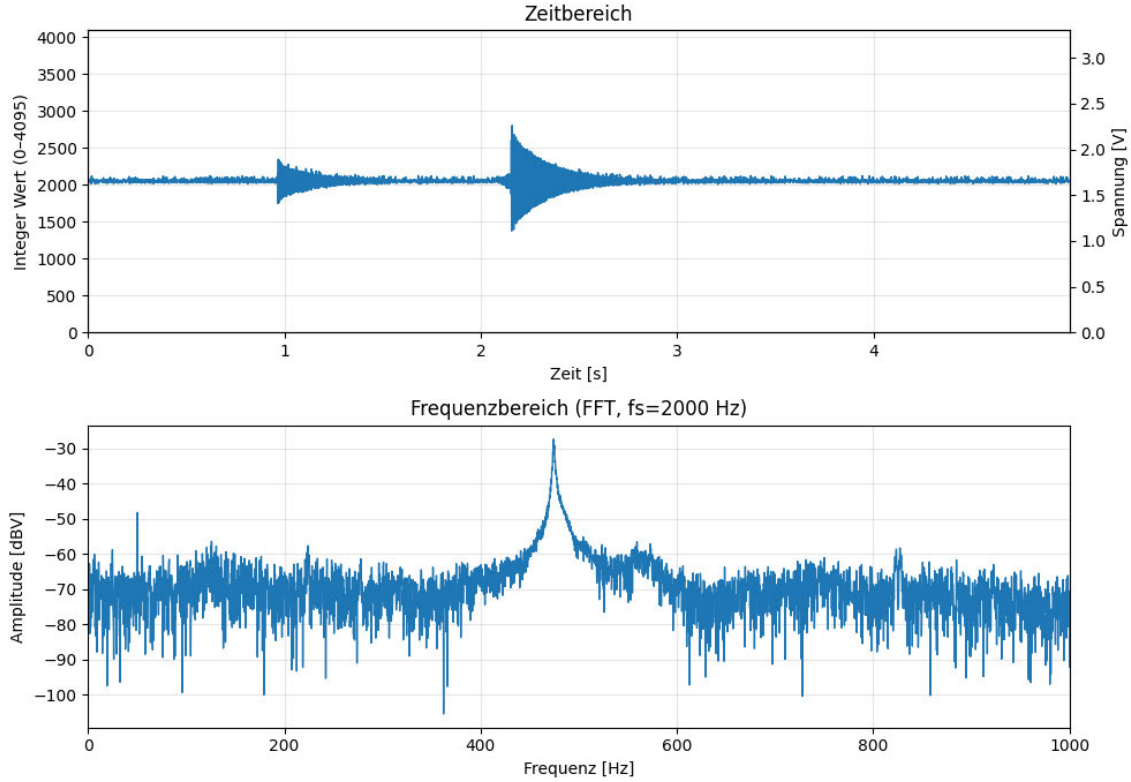

Abbildung 7.4: Ausgabe der 12 Bit ADC-Daten mit $f_s = 2\,\text{kHz}$ zur Verifikation des ADC Boards und Signalverstärker

Zur Verifikation des ADC-Boards und der Sensorverstärkerschaltung wurde das Signal über die in Kapitel 6.3.4 beschriebene UART-Schnittstelle ausgegeben. In Abbildung 7.4 sind die Messdaten eines Stoßereignisses dargestellt, das mithilfe eines Fallhammers erzeugt wurde. Der Fallhammer besteht aus einer Masse, die entlang einer Schiene geführt und anschließend abrupt gestoppt wird. Kurz vor dem Aufprall, der den Stoß verursacht, ist das Gleiten der Masse auf dem Schienensystem erkennbar. In den Messdaten äußert sich dies wie ansteigendes Rauschen.

Auf der oben Seite der Abbildung 7.4 sind die ADC-Samples als Integerwerte dargestellt, während auf der unteren Seite die umgerechneten Spannungswerte gezeigt werden. Diese Spannung entspricht der Ausgangsspannung der Sensorverstärker. Der maximale Spannungswert beträgt 3,3 V und entspricht damit der Versorgungsspannung des Basys 3-Boards.

In der Abbildung ist der Offset von 1,65 V deutlich zu erkennen. Außerdem zeigt sich ein sehr geringes Rauschen. Mit einem Totband von ±0,165 V um den Offsetwert von 1,65 V kann dieses Rauschen in der Nullage vom Regler effektiv ausgeblendet werden. Die Frequenzanalyse des Zeitsignals aus Abbildung 7.4 zeigt, dass das Signal Frequenzanteile von 466 Hz enthält. Dies bedeutet, dass hauptsächlich die Hubmode des Chips angeregt wurde, obwohl die Anregung bei stehendem Chip durchgeführt wurde.

7.2.3 Verifikation Ausgang als Teilsystem

Der Ausgang wurde mithilfe des Sinusgenerator-Testmoduls überprüft und kalibriert. Dazu wurden beide DAC-Ausgänge mit dem Sinusgenerator verbunden. Für die Kalibrierung wurde das Sinusmodul mit einer Triggerfrequenz von 2 kHz betrieben, wodurch eine Sinusfrequenz von 7,8 Hz erzeugt wurde. Mithilfe der Sinusfunktion konnte die Mittenspannung über den Kalibrierwiderstand R_9 aus der Schaltung in Abbildung 6.4 auf 0 V eingestellt werden. Der Sinus nutzt den Spannungsbereich von ±2,5 V optimal aus. Das saubere Ausgangssignal zeigt, dass die Hardwareanbindung an das Basys 3-Board vollständig funktionsfähig ist und die Treiberstufe die richtige Verstärkung aufweist.

7.2.4 Verifikation des proportional geregelten System

In Abbildung 7.5 ist der Betrieb des Gesamtsystems dargestellt. Zum Testen der Closed-Loop-Regelung wurde der Chip mit einem Stoß durch den Fallhammer beaufschlagt. Für die Verifikation des stabilen Proportionalreglers wurden die Eingangswerte eines Sensorelements verwendet, damit die Bewegung des Spiegels über das gegenüberliegende Sensorelement aufgenommen werden kann. In Abbildung 7.5 ist im oberen Plot die Stellgröße dargestellt und im unteren Plot die Stellung des gegenüberliegenden Aktuators. Entgegen den Erwartungen entspricht das Sensorsignal nicht der Stellgröße. Offenbar bewegen sich die Aktuatoren nicht um 180° phasenverschoben, da bei einer solchen Bewegung die gegenüberliegenden Sensorspannungen ein unterschiedliches Vorzeichen hätten. Aus Abbildung 7.5 kann eine Frequenz der Aktuatoren von ca. 476 Hz abgelesen werden. Diese Frequenz deutet auf eine angeregte Hubmode hin. Die Hubmode kann mit der achsenweisen Ansteuerung nicht kompensiert werden, da hierbei die Aktuatoren im Gegentakt angesteuert werden.

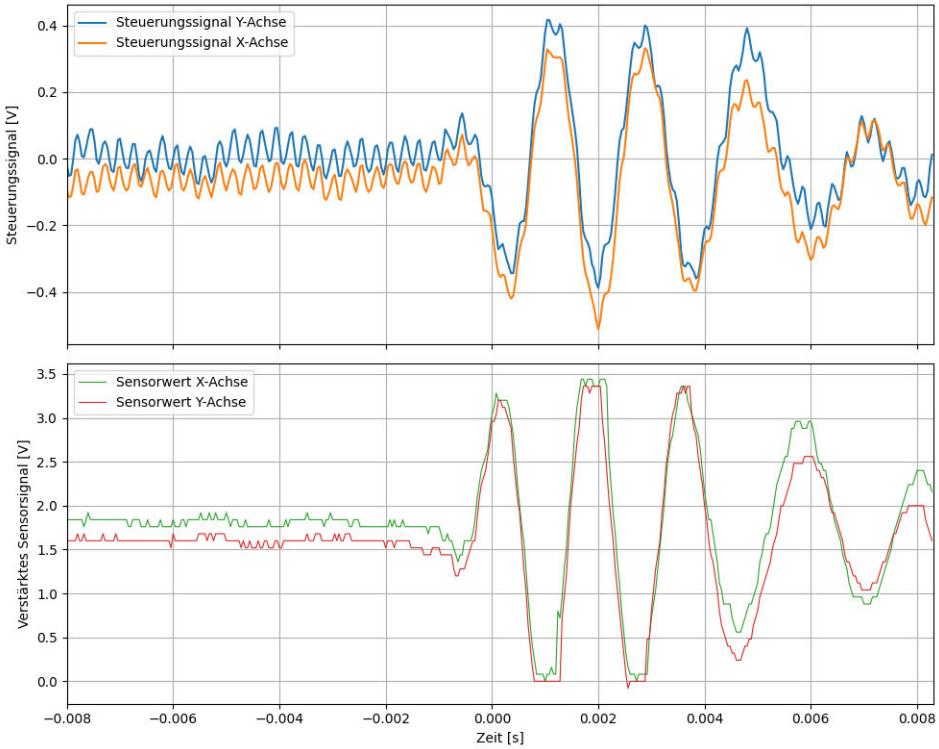

Abbildung 7.5: Signalverlauf des proportionalgeregelten Systems

7.2.5 Verifikation des Abklingverhalten

In Abbildung 7.6 sind zwei identische Schocks dargestellt, die mit dem Schocktisch aus Kapitel 3.3 auf das System eingeprägt wurden. Der eingeprägte Schock hat eine maximale Beschleunigung von 2,4 g. Das Signal wurde vom Sensorverstärker des gegenüberliegenden Sensors aufgenommen, für die X-Achse ist dies Sensor 3. Die Steuerung mit dem Differenzsignal, wie sie in Abbildung 6.8 gezeigt ist, hat zu keinen messbaren Verbesserungen geführt. Das liegt an der Auslegung der Regelung, bei der davon ausgegangen wurde, dass das differentielle Signal eine ähnliche Amplitude wie die einzelnen Sensorsignale besitzt. Die Nutzung nur eines Sensors pro Achse hat hingegen zu messbaren Ergebnissen geführt. Hierbei wurde, wie in Kapitel 7.2.4 beschrieben, pro Achse nur ein Sensorsignal zur Closed-Loop-Regelung verwendet, und mit beiden Aktuatoren wurde der Spiegel angetrieben. Die Daten in Abbildung 7.6 zeigen, dass das Abklingen durch die Proportionalregelung nicht verkürzt wird. Im Zeitbereich zeigt das proportionalgeregelte Signal eine größere Spannung, was darauf hindeutet, dass durch die Regelung mehr

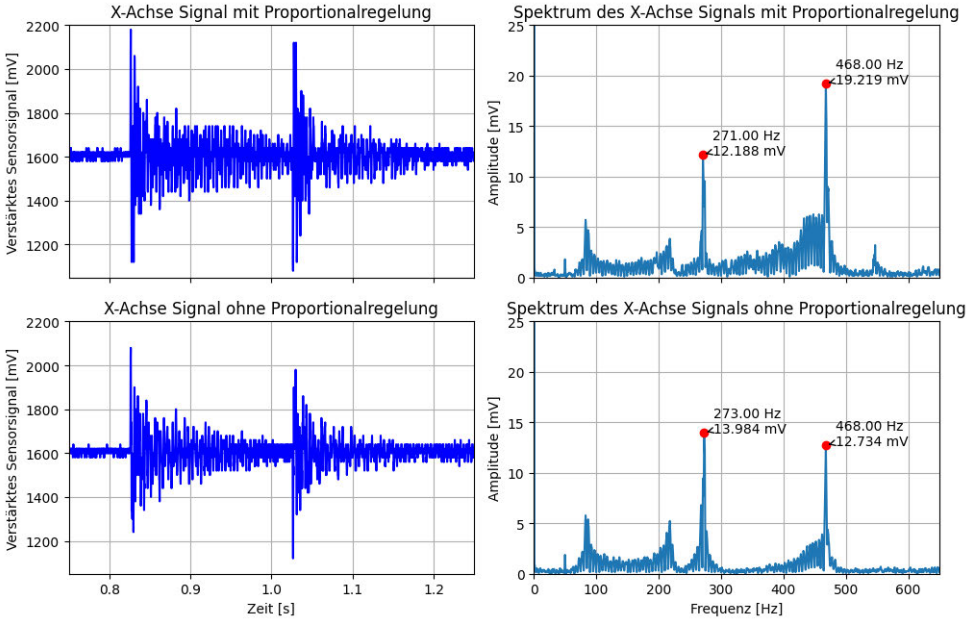

Abbildung 7.6: Vergleich der Schockeinwirkung auf ein proportional geregelten System gegen ein nicht geregelten

Energie in das System eingebracht wird, das System also stärker angeregt statt gedämpft wird. Im Spektrum wird sichtbar, dass die Regelung ein Dämpfungsverhalten bei 266 Hz zeigt, dadurch jedoch die Resonanz bei 466 Hz stärker anregt. Mit der Regelung wird somit die Kippmode bei 266 Hz unterdrückt. Durch die Ansteuerung der Aktuatoren im Gegentakt kann die Hubmode jedoch nicht gegengeregelt werden, da dafür beide Aktuatoren in die gleiche Richtung bewegt werden müssten. Zusammenfassend zeigen die Daten, dass es möglich ist, die Schwingung des Spiegels zu dämpfen. Die erste, nicht optimierte Methode der Schwingungsdämpfung zeigt eine Reduktion der Kippmode um 12,8 %. Die Hubmode wird dagegen durch die Steuerung um etwa 50 % verstärkt.

7.2.6 Zusammenfassung der funktionalen Anforderungen

Tabelle 7.1: Verifikation der funktionalen Anforderungen

Nr.	**Anforderung**	**Zielwert / Bedingung**	**Ergebnis / Bewertung**
F1	Treiberstufe	Ausgangsspannung $\pm 2{,}5\,\mathrm{V}$	Erfüllt. Spannungsbereich erreicht, Mittenspannung feinjustiert.
F2	Sensorverstärker	Ausgangsspannung $0\,\mathrm{V}$ bis $3{,}3\,\mathrm{V}$	Erfüllt. Signalbereich stabil, Schockeinwirkung wird zuverlässig erkannt.
F3	Steuerung	C++-Funktion zur Positionsvorgabe	Erfüllt. Softwaresteuerung funktionsfähig und getestet.
F4	Regelung	Abklingzeit um $50\,\%$ reduzieren	Nicht erfüllt. Abklingzeit unverändert, jedoch $12{,}8\,\%$ Dämpfung der Kippmode erreicht.
F5	Stabilität	Kein Aufschwingen des Spiegels	Erfüllt. System bleibt in allen Betriebszuständen stabil.
F6	Abtastrate	$f_\mathrm{s} \geq 4{,}66\,\mathrm{kHz}$	Erfüllt. Tatsächliche Abtastrate beträgt $20\,\mathrm{kHz}$.
F7	FPGA-Implementierung	Hardwarefähige Implementierung, deterministisches Zeitverhalten	Erfüllt. Synthese erfolgreich, deterministisches Verhalten mit einer Reaktionszeit von $60\,\mu\mathrm{s}$ nachgewiesen.

8 Fazit und Ausblick

In dieser Arbeit wurde eine Open-Loop-Steuerung entworfen, mit der dem quasi-statischen MEMS Spiegel über C/C++-Code Koordinaten vorgegeben werden können, die anschließend vom reflektierten Laser abgefahren werden. Die Funktionalität wurde anhand einiger Probe-Scanmuster gezeigt. Die Steuerung bewegt den MEMS-Spiegel mit vier Aktuatoren im Gegentaktbetrieb, wodurch eine maximale Verkippung des Spiegels von 1° erreicht wird. Die Geschwindigkeit, mit der die Koordinaten angefahren werden können, ist begrenzt. Mit einer optimierten Ansteuerung, die das Anregen der Resonanzfrequenzen verhindert, kann die Geschwindigkeit jedoch verbessert werden.

Im zweiten Teil der Arbeit wurde das Schock- und Frequenzverhalten der Spiegel anhand der Spannungen der neuartigen piezoelektrischen Sensorelemente charakterisiert. Basierend auf diesen Ergebnissen wurde eine Kompensationsstrategie entwickelt. Eine Proportionalregelung hat sich in der Simulation als sinnvoll erwiesen. Die sehr kleinen Sensorspannungen werden mithilfe eines eigens entwickelten PCBs auf einen Pegel gehoben, der eine Verarbeitung der Daten mit einem FPGA ermöglicht. Die Ausgangsspannung wird über eine im Zuge der Arbeit entwickelten Treiberstufe in einen bipolaren Spannungsbereich transformiert. Die Spiegel werden im Gegentakt angesteuert, wodurch lediglich eine Verkippung des Spiegels möglich ist. Mit der Ansteuerung im Gegentakt lässt sich keine Hubbewegung erzeugen, weshalb diese nicht kompensiert werden kann. Die Veränderung der Lichtstrecke bei voller Hubauslenkung ist bei einem LiDAR-System nicht von Bedeutung, da sie weit unter 1 mm liegt und LiDAR-Systeme für Automotivanwendungen eine Tiefenauflösung von 2 cm besitzen [12].

Mit der Steuerung wurde bei Schockeinwirkung eine Dämpfung der Kippmode um 12,8 % erreicht. Damit in Zukunft eine stärkere Dämpfung erzielt werden kann, ist es erforderlich, die Reglerparameter weiter zu optimieren. Zudem bietet die Steuerung über den FPGA die Möglichkeit, andere Kompensationsmethoden zu testen. Für zukünftige Steuerungen ist es empfehlenswert, jeden Aktuator separat anzusteuern. Dadurch kann auch die Hubmode kompensiert werden.

Literaturverzeichnis

[1] Arduino: Arduino_Nano_datenblatt. URL https://docs.arduino.cc/hardware/nano/. – Zugriffsdatum: 2025-07-07, Juli 2025. – Datenblatt

[2] Ballas, Rüdiger G.: The Piezoelectric Effect - an Indispensable Solid State Effect for Contemporary Actuator and Sensor Technologies. In: *Journal of Physics: Conference Series* 1775 (2021), Januar, Nr. 1, S. 012012. – URL https://iopscience.iop.org/article/10.1088/1742-6596/1775/1/012012. – Zugriffsdatum: 2025-09-22. – ISSN 1742-6588, 1742-6596

[3] Ballas, Rüdiger G.: *Piezoelektrische Biegewandler: Zur Physik des statischen und dynamischen Verhaltens*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2025. – URL https://link.springer.com/10.1007/978-3-662-70390-8. – Zugriffsdatum: 2025-04-01. – ISBN 978-3-662-70389-2

[4] Brüel & Kjær: High g Head Type 4811 – Instruction Manual / Brüel & Kjær. Nærum, Dänemark, März 1971. – Forschungsbericht. – URL https://pearl-hifi.com/06_Lit_Archive/15_Mfrs_Publications/10_Bruel_Kjaer/03_Instruction_Manuals/B&K%204811%20Instruction%20Manual.pdf. – Zugriffsdatum: 2025-07-31

[5] Diodes Incorporated: AZ431L / Diodes Incorporated. URL https://www.diodes.com/datasheet/download/AZ431L.pdf. – Zugriffsdatum: 2025-08-27, Juli 2024 (DS36801 Rev. 5 - 2). – Datenblatt. – 19 S

[6] Falco Systems: Falkon_WMA-02_high_voltage_amplifier_manual / Falco Systems. URL https://www.falco-systems.com. – Zugriffsdatum: 2025-09-08, Oktober 2024 (Version 1.12). – Datenblatt

[7] Fern, o Barbero S.: UART 16550 IP Datasheet / SIDSA. 166 Geary Street Suite 1307 San Francisco, CA 94108, Dezember 2000. – Datenblatt. – 18 S. – URL https://caro.su/msx/ocm_de1/16550.pdf. – Zugriffsdatum: 2025-09-29

[8] Horowitz, Paul 1. ; Hill, Winfield (Hrsg.): *The art of electronics*. 3rd ed. Cambridge, New York : Cambridge University Press, 2015. – URL https://external.dandelon.com/download/attachments/dandelon/ids/DE00298DC73D1CBF81E24C1257E810043D9AA.pdf. – Hier auch später erschienene, unveränderte Nachdrucke

[9] Kreutzer, Tom-Niklas ; Fichtner, Simon ; Wagner, Bernhard ; Lofink, Fabian: A double-layer MEMS actuator based on ferroelectric polarization inversion in AlScN. In: *2021 IEEE International Symposium on Applications of Ferroelectrics (ISAF)*. Sydney, Australia : IEEE, Mai 2021, S. 1–3. – URL https://ieeexplore.ieee.org/document/9477382/. – Zugriffsdatum: 2025-09-29. – ISBN 978-1-6654-0444-0

[10] Kurzweil, Peter ; Frenzel, Bernhard ; Gebhard, Florian: *Physik Formelsammlung: Mit Erläuterungen und Beispielen aus der Praxis für Ingenieurberufe und Naturwissenschaften*. Wiesbaden : Springer Fachmedien Wiesbaden, 2024. – URL https://link.springer.com/10.1007/978-3-658-45491-3. – Zugriffsdatum: 2025-08-17. – ISBN 978-3-658-45490-6 978-3-658-45491-3

[11] Kuttner, Thomas: *Praxiswissen Schwingungsmesstechnik*. Wiesbaden : Springer Fachmedien Wiesbaden, 2015. – URL https://link.springer.com/10.1007/978-3-658-04638-5. – Zugriffsdatum: 2025-10-07. – ISBN 978-3-658-04637-8 978-3-658-04638-5

[12] Leishen Intelligent System Co., Ltd.: Leishen LiDAR Product Guide (ENV2.0). URL https://bias.com.tr/storage/media/4308/leishen-lidar-product-guide-801205.pdf?utm_source=chatgpt.com, November 2023. – Forschungsbericht

[13] Li, Yanlu ; Dieussaert, Emiel ; Baets, Roel: Miniaturization of Laser Doppler Vibrometers—A Review. In: *Sensors* 22 (2022), Juni, Nr. 13, S. 4735. – URL https://www.mdpi.com/1424-8220/22/13/4735. – Zugriffsdatum: 2025-07-23. – Publisher: MDPI AG. – ISSN 1424-8220

[14] Libretexts: *6.2: Spring-Mass Problems (with Damping)*. 11 2022. – URL https://math.libretexts.org/Courses/Cosumnes_River_College/Math_420%3A_Differential_Equations_%28Breitenbach%29/06%3A_Applications_of_Linear_Second_Order_Equations/6.02%3A_Spring-Mass_Problems_%28With_Damping%29

[15] Mayrhofer, P.M. ; Riedl, H. ; Euchner, H. ; Stöger-Pollach, M. ; Mayrhofer, P.H. ; Bittner, A. ; Schmid, U.: Microstructure and piezoelectric response of Y Al1-N thin films. In: *Acta Materialia* 100 (2015), November, S. 81–89. – URL https://linkinghub.elsevier.com/retrieve/pii/S1359645415005832. – Zugriffsdatum: 2025-04-02. – ISSN 13596454

[16] Mertins, Alfred: *Signaltheorie: Grundlagen der Signalbeschreibung, Filterbänke, Wavelets, Zeit-Frequenz-Analyse, Parameter- und Signalschätzung*. Wiesbaden : Springer Fachmedien Wiesbaden, 2023. – URL https://link.springer.com/10.1007/978-3-658-41529-7. – Zugriffsdatum: 2025-08-04. – ISBN 978-3-658-41528-0 978-3-658-41529-7

[17] Microchip Technology Inc.: ATmega328/P: 8-bit AVR Microcontrollers. URL https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf. – Zugriffsdatum: 2025-06-05, 2015 (7810D–AVR–01/15). – Datenblatt

[18] {Microchip Technology Inc.}: ADC_mcp3201 / Microchip Technology Inc. Chandler, Arizona, USA, Januar 2007. – Forschungsbericht. – URL https://ww1.microchip.com/downloads/aemDocuments/documents/APID/ProductDocuments/DataSheets/21293C.pdf. – Zugriffsdatum: 2025-09-08

[19] Stoeckel, Chris ; Meinel, Katja ; Melzer, Marcel ; Žukauskaitė, Agnė ; Zimmermann, Sven ; Forke, Roman ; Hiller, Karla ; Kuhn, Harald: Static High Voltage Actuation of Piezoelectric AlN and AlScN Based Scanning Micromirrors. In: *Micromachines* 13 (2022), April, Nr. 4, S. 625. – URL https://www.mdpi.com/2072-666X/13/4/625. – Zugriffsdatum: 2025-04-02. – ISSN 2072-666X

[20] Sundararajan, D.: *Signals and Systems: A Practical Approach*. Cham : Springer Nature Switzerland, 2023. – URL https://link.springer.com/10.1007/978-3-031-19377-4. – Zugriffsdatum: 2025-08-04. – ISBN 978-3-031-19376-7 978-3-031-19377-4

[21] Texas Instruments: dac121s101 / Texas Instruments. 655303, Dallas, Texas 75265, September 2015. – Datenblatt. – URL https://www.ti.com/lit/ds/symlink/dac121s101.pdf?ts=1757390220048. – Zugriffsdatum: 2025-09-05

[22] Texas Instruments: INA333 Micro-Power / Texas Instruments Incorporated. Dallas, TX, USA, 2015. – Datenblatt. – URL https://www.ti.com/lit/ds/

symlink/ina333.pdf?ts=1758742127225&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252Fde-de%252FINA333. – Zugriffsdatum: 2025-09-09

[23] Texas Instruments Incorporated: TL051 / Texas Instruments Incorporated. DALLAS, TEXAS 75265, Februar 2003. – Datenblatt. – 65 S. – URL https://www.ti.com/lit/ds/symlink/tl051.pdf?ts=1757857496892. – Zugriffsdatum: 2025-09-07

[24] Wang, Dingkang ; Watkins, Connor ; Xie, Huikai: MEMS Mirrors for LiDAR: A Review. In: *Micromachines* 11 (2020), April, Nr. 5, S. 456. – URL https://www.mdpi.com/2072-666X/11/5/456. – Zugriffsdatum: 2025-07-04. – ISSN 2072-666X

[25] Wendt, Martin: Piezoelektrische Beschleunigungsaufnehmer. (2021). – URL https://mmf.de/wp-content/uploads/2023/08/aufnehmerman.pdf. – Zugriffsdatum: 2025-08-04

[26] Yoo, Han W. ; Riegler, Rene ; Brunner, David ; Albert, Stephan ; Thurner, Thomas ; Schitter, Georg: Experimental Evaluation of Vibration Influence on a Resonant MEMS Scanning System for Automotive Lidars. In: *IEEE Transactions on Industrial Electronics* 69 (2022), März, Nr. 3, S. 3099–3108. – URL https://ieeexplore.ieee.org/document/9380978/. – Zugriffsdatum: 2025-11-16. – ISSN 0278-0046, 1557-9948

[27] Zacher, Serge ; Reuter, Manfred: *Regelungstechnik für Ingenieure: Analyse, Simulation und Entwurf von Regelkreisen*. Wiesbaden : Springer Fachmedien Wiesbaden, 2024. – URL https://link.springer.com/10.1007/978-3-658-45897-3. – Zugriffsdatum: 2025-08-01. – ISBN 978-3-658-45896-6 978-3-658-45897-3

[28] Zielke, Dirk: *Mikrosysteme: Micro-Electro-Mechanical Systems (MEMS)*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2025. – URL https://link.springer.com/10.1007/978-3-662-71233-7. – Zugriffsdatum: 2025-07-04. – ISBN 978-3-662-71232-0 978-3-662-71233-7

A Anhang

A.1 Verwendete Hilfsmittel

In der Tabelle A.1 sind die im Rahmen der Bearbeitung des Themas der Masterarbeit verwendeten Werkzeuge und Hilfsmittel aufgelistet.

Tabelle A.1: Verwendete Hilfsmittel und Werkzeuge

Tool	Verwendung
LaTeX	Textsatz- und Layout-Werkzeug verwendet zur Erstellung dieses Dokuments
Simulink	Reglerentwurf in HDL
Vivado	IDE für Artix7 FPGA
ChatGPT	Lektor
GitHubCopilot	Generierung von Code-Snippets und Hilfsmittel bei der Fehlersuche im Quellcode
Anaconda	Datenanalyse
KiCad	Erstellung von Layouts
OrCad	Schaltungssimulation
Microchipstudio	IDE für den Atmega238p Microcontroller

B C/C++-Code

B.1 Atmega 328p C-Code

```
/*
 * GccApplication2.cpp
 *
 * Created: 22.07.2025 08:28:19
 * Author : voss
 */

#include <avr/io.h>
#define F_CPU 16000000UL
#include <util/delay.h>
#include <util/twi.h>
#include <avr/interrupt.h>
#include <math.h>
#include <stdlib.h>

void ADC0_init(void){
        PRR0 &= ~(1<<PRADC);     // turn off power saving mode
        DIDR0 &= ~(1<<ADC0D);    // activate input buffer of ADC input
            0
        ADCSRA |= (1<<ADEN);     // enable ADC channel
        ADMUX &= ~(1<<MUX2)|~(1<<MUX1)|~(1<<MUX0);  // activate ADC
           MUX channel 1,
                                                    // in this
                                                       register a
                                                       different
                                                       reference
                                                       source can
                                                       also be set
```

```
22          ADMUX |= (1<<REFS0);                                // VCC as
                reference
23          ADCSRA |= (1<<ADPS2) | (1<<ADPS1) | (1<<ADPS0); // set the
                clock divider to 128 to get maximum resolution, then only
                15 kHz sampling rate
24          ADCSRA &= ~(1<<ADLAR); // auto trigger
25          ADCSRA |= (1<<ADIE); // enable interrupt
26          ADCSRA |= (1<<ADATE); // enable auto trigger
27          ADCSRB &= ~(1<<ADTS2)|~(1<<ADTS1)|~(1<<ADTS0); // set the
                trigger to auto trigger
28  }
29
30  uint16_t ADC0_read(void){
31      static uint8_t ADC_lsb = 0x00,
32                     ADC_msb = 0x00;
33      static uint16_t ADC_val = 0x0000;
34      ADCSRA |= (1<<ADSC); // start the analog-to-digital conversion
35
36      while(1){ // wait for the ADC register until the values are ready
            to read
37          if((ADCSRA & (1<<ADIF))== 0x10){break;}
38          PORTB =0x20;
39      };
40
41      ADC_lsb = ADCL; // read ADC LSB
42      ADC_msb = ADCH; // read ADC MSB
43      ADC_val =0x0000;
44      ADC_val |= (ADC_msb<<8)|(ADC_lsb);
45
46      ADCSRA &= ~(ADIF); // reset the "ready to read" flag
47
48      return ADC_val;
49  }
50
51  void AVR_init(void)
52  {
53      // USART initialization
54      // Communication Parameters: 8 Data, 1 Stop, No Parity
55      // USART Receiver: Off
56      // USART Transmitter: On
```

```
57      // USART Mode: Asynchronous
58      // USART Baud Rate: 9600
59      UCSR0A&=~(1<<RXC0) |// USART Receive Complete
60              ~(1<<TXC0) |// USART Transmit Complete
61              ~(1<<UDRE0)|// USART Data Register Empty
62              ~(1<<FE0)  |// Frame Error
63              ~(1<<DOR0) |// Data OverRun
64              ~(1<<UPE0) |// USART Parity Error
65              ~(1<<U2X0) |// Double the USART Transmission Speed
66              ~(1<<MPCM0);// Multi-processor Communication Mode
67
68      UCSR0B&=~(1<<RXCIE0)|// RX Complete Interrupt Enable
69              ~(1<<TXCIE0)|// TX Complete Interrupt Enable
70              ~(1<<UDRIE0)|// USART Data Register Empty Interrupt
                   Enable
71              ~(1<<RXEN0) |// Receiver Enable
72              ~(1<<UCSZ02)|// Character Size
73              ~(1<<RXB80) |// Receive Data Bit 8
74              ~(1<<TXB80); // Transmit Data Bit 8
75      UCSR0B|=(1<<TXEN0); // Transmitter Enable
76      UCSR0C&=~(1<<UMSEL01)|// USART Mode Selection
77              ~(1<<UMSEL00)|// USART Mode Selection now it is
                   Asynchronous USART
78              ~(1<<UPM01)  |// USART Parity Mode
79              ~(1<<UPM00)  |// USART Parity Mode now it is disabled
80              ~(1<<USBS0)  |// Stop Bit Settings now we have one stop
                   bit
81              ~(1<<UCPOL0);
82      UCSR0C|=(1<<UCSZ01) | (1<<UCSZ00);// Character Size Settings now
           we have 8-bit characters
83
84      UBRR0H=0x00; // MSB for 9600 baud
85      UBRR0L=0x67; // LSB  0x67 = 103 dec    UBRR0 = (f_clock/16*Baud)
           -1
86  }
87
88  void USART0_Transmit( uint8_t data )
89  {
90      /* Wait for empty transmit buffer */
91      while ( !( UCSR0A & (1<<UDRE0)) )
```

```
92      ;
93      /* Put data into buffer, sends the data */
94      UDR0 = data;
95  }
96
97  void PWM_PB1_init(void){
98      DDRB |= (1<<DDRB1); // activate PB1 as output
99
100     OCR1A = 0x0000;
101     //OCR1AL = 0xFF;
102     //OCR1AH = 0x02; // The Output Compare Registers contain a 16-bit
            value that is continuously compared with the counter value (
        TCNT1)
103
104     TCCR1A &= ~(1<<COM1A0); // Compare Output Mode for channel
105     TCCR1A |= (1<<COM1A1);  // Compare Output Mode for channel: Clear
            OC1A/OC1B on Compare Match, set OC1A/OC1B at BOTTOM (non-
        inverting mode)
106     TCCR1A |= (1<<WGM11);   // first two bits for Waveform Generation
            Mode bit description
107     TCCR1A &=~(1<<WGM10);
108
109     TCCR1B |= (1<<WGM12)|(1<<WGM13); // last two bits for Waveform
        Generation Mode description; choose "FAST PWM" with ICR1 as
        TOP
110
111     //TCCR1B |= (1<<ICNC1); // Input Capture Noise Canceler
112     TCCR1B |= (1<<CS10); // clk no prescaling for clock
113     TCCR1B &=~(1<<CS12)|~(1<<CS11); // clk/0 prescaling for clock
114
115     ICR1 = 0x03FF;
116     //ICR1L = 0xFF; // Input Capture 1 - The Input Capture can be
        used for defining the counter TOP value.
117     //ICR1H = 0x03;
118 }
119
120 void PWM_PB2_init(void){
121     DDRB |= (1<<DDRB2); // activate PB2 as output; it is pin D10 on
        the board
122
```

```
123     OCR1B = 0x0000;
124     //OCR1BL = 0xFF;
125     //OCR1BH = 0x02; // The Output Compare Registers contain a 16-bit
            value that is continuously compared with the counter value (
            TCNT1)
126
127     TCCR1A &= ~(1<<COM1B0); // Compare Output Mode for channel
128     TCCR1A |= (1<<COM1B1);  // Compare Output Mode for channel: Clear
            OC1A/OC1B on Compare Match, set OC1A/OC1B at BOTTOM (non-
            inverting mode)
129     TCCR1A |= (1<<WGM11);   // first two bits for Waveform Generation
            Mode bit description
130     TCCR1A &=~(1<<WGM10);
131
132     TCCR1B |= (1<<WGM12)|(1<<WGM13); // last two bits for Waveform
            Generation Mode description; choose "FAST PWM" with ICR1 as
            TOP
133
134     //TCCR1B |= (1<<ICNC1); // Input Capture Noise Canceler
135     TCCR1B |= (1<<CS10); // clk no prescaling for clock
136     TCCR1B &=~(1<<CS12)|~(1<<CS11); // clk/0 prescaling for clock
137
138     ICR1 = 0x03FF;
139     //ICR1L = 0xFF; // Input Capture 1 - The Input Capture can be
            used for defining the counter TOP value.
140     //ICR1H = 0x03;
141 }
142
143 void select_quadrant_y(bool state){
144     if(state == true){
145         PORTD |= (1<<PORTD7);
146     }
147     else {
148         PORTD &= ~(1<<PORTD7);
149     }
150 }
151
152 void select_quadrant_x(bool state){
153     if(state == true){
154         PORTB |= (1<<PORTB0);
```

```
155     }
156     else {
157         PORTB &= ~(1<<PORTB0);
158     }
159 }
160 /*
161 void move_to(int x_target, int y_target) {
162     const int step_time = 2; // ms between steps
163     static int x_current = 0;
164     static int y_current = 0;
165 
166     // ---- Move X axis ----
167     while (x_current != x_target) {
168         if(x_current < x_target){
169             x_current = x_current+1;
170         }
171         else{
172             x_current = x_current-1;
173         }
174         OCR1A = (uint16_t)abs(x_current);
175         if (x_current > 0){
176             select_quadrant_x(true);
177         }
178         else{
179             select_quadrant_x(false);
180         }
181         _delay_ms(step_time);
182     }
183 
184     while (y_current != y_target) {
185         if(y_current < y_target){
186             y_current = y_current+1;
187         }
188         else{
189             y_current = y_current-1;
190         }
191         OCR1B = (uint16_t)abs(y_current);
192         if (y_current > 0){
193             select_quadrant_y(true);
194         }
```

```
195         else{
196             select_quadrant_y(false);
197         }
198         _delay_ms(step_time);
199     }
200 }
201 */
202 
203 void move_to(int x_target, int y_target) {
204     const int step_time = 15; // us between steps
205     static int x_current = 0;
206     static int y_current = 0;
207 
208     // Calculation of differences
209     int dx = abs(x_target - x_current);
210     int dy = abs(y_target - y_current);
211 
212     // Calculation of directions (steps)
213     int sx = (x_current < x_target) ? 1 : -1;
214     int sy = (y_current < y_target) ? 1 : -1;
215 
216     // Error calculation
217     int err = dx - dy;
218 
219     // ---- Move X and Y simultaneously ----
220     while (x_current != x_target || y_current != y_target) {
221         // Update PWM for X and Y
222         OCR1A = (uint16_t)abs(x_current);
223         OCR1B = (uint16_t)abs(y_current);
224 
225         // Quadrant switching
226         if (x_current > 0) {
227             select_quadrant_x(true);
228         } else {
229             select_quadrant_x(false);
230         }
231         if (y_current > 0) {
232             select_quadrant_y(true);
233         } else {
234             select_quadrant_y(false);
```

```
235         }
236
237         // Bresenham error calculation
238         int e2 = 2 * err;
239
240         // Move X if the error in X direction is greater than in Y
                direction
241         if (e2 > -dy) {
242             err -= dy;
243             x_current += sx;
244         }
245
246         // Move Y if the error in Y direction is greater than in X
                direction
247         if (e2 < dx) {
248             err += dx;
249             y_current += sy;
250         }
251
252         _delay_us(step_time);
253     }
254 }
255
256 void move_circle(int radius) {
257     const int step_time = 15;   // step time between coordinates in
            us
258     const float PI = 3.14159265;
259     const int steps = 360; // 360 steps for a nice round movement
260
261     for (int i = 0; i <= steps; i++) {
262         float angle = 2 * PI * i / steps;
263
264         int x_target = (int)(radius * cos(angle));
265         int y_target = (int)(radius * sin(angle));
266
267         // Set quadrant switch
268         select_quadrant_x(x_target >= 0);
269         select_quadrant_y(y_target >= 0);
270
271         OCR1A = (uint16_t)abs(x_target);
```

```
272         OCR1B = (uint16_t)abs(y_target);
273
274         _delay_us(step_time);
275     }
276 }
277
278 int main(void)
279 {
280     DDRB |= (1<<DDRB5);
281     DDRD |= (1<<DDRB7); // activate D6 and D7 as output. It is for
        the amplifier select
282     DDRB |= (1<<DDRB0);
283
284     PORTD |= (1<<PORTD7); // switch D6 and D7 on so they are defined
285     PORTB |= (1<<PORTB0);
286
287     //AVR_init(); // initialize UART0
288     //ADC0_init(); // initialize ADC channel 0
289     PWM_PB1_init(); // initialization PWM pin PD1 (D8 on board)
290     PWM_PB2_init();
291
292     uint8_t out= 0x00;
293     uint8_t i=0x00;
294
295     while(1)
296     {
297         static int counts=0;
298         static int state = 0;
299         if(counts >= 10){
300             state=state+1;
301             counts=0;
302             if(state >= 10){state=0;}
303         }
304         else{
305             counts=counts+1;
306         }
307
308         switch (state) {
309             case 0:
310                 move_to( 512, 1023);
```

```
311                 move_to( 512,-1023);
312                 move_to(-512,-1023);
313                 move_to(-512, 1023);
314             default:
315             break;
316             case 1:
317                 move_to(   512,  0);
318                 move_to(  0,  1023);
319                 move_to(  0, -1023);
320                 move_to(  0,  1023);
321             break;
322             case 2:
323                 move_to( 512, 1023);
324                 move_to(  -512, 1023);
325                 move_to(  -512,  0);
326                 move_to( 512,  0);
327                 move_to( 512,  -1023);
328                 move_to(  -512, -1023);
329                 move_to( 512,  -1023);
330                 move_to( 512,  0);
331                 move_to(  -512,  0);
332                 move_to(  -512, 1023);
333             break;
334             case 3:
335                 move_to(  512, 1023);
336                 move_to(  -512, 1023);
337                 move_to(  -512,  0);
338                 move_to(  512,  0);
339                 move_to(  -512,  0);
340                 move_to(  -512,-1023);
341                 move_to(  512, -1023);
342                 move_to(  -512,-1023);
343                 move_to(  -512,  0);
344                 move_to(  512,  0);
345                 move_to(  -512,  0);
346                 move_to(  -512, 1023);
347             break;
348             case 4:
349                 move_to(  -512, 0);
350                 move_to(  512, 0);
```

```
351                 move_to(  0,  1023);
352                 move_to(  0,  -1023);
353                 move_to(  0,  1023);
354                 move_to(  512, 0);
355             break;
356             case 5:
357                 move_to( -512, 1023);
358                 move_to(  512, 1023);
359                 move_to(  512,  0);
360                 move_to( -512,  0);
361                 move_to( -512,  -1023);
362                 move_to(  512, -1023);
363                 move_to( -512,  -1023);
364                 move_to( -512,  0);
365                 move_to(  512,  0);
366                 move_to(  512, 1023);
367             break;
368
369             case 6:
370                 move_to( -512, 1023);
371                 move_to(  512, 1023);
372                 move_to(  512,  0);
373                 move_to( -512,  0);
374                 move_to( -512,  -1023);
375                 move_to(  512, -1023);
376                 move_to(  512, 0);
377                 move_to(  512, -1023);
378                 move_to( -512,  -1023);
379                 move_to( -512,  0);
380                 move_to(  512,  0);
381                 move_to(  512, 1023);
382             break;
383             case 7:
384                 move_to(  512, 1023);
385                 move_to( -512, 1023);
386                 move_to(  512,-1023);
387                 move_to( -512, 1023);
388             break;
389             case 8:
390                 move_to( 512, 1023);
```

```
391                 move_to(  -512, 1023);
392                 move_to(  -512,  0);
393                 move_to( 512,  0);
394                 move_to( 512,  -1023);
395                 move_to(  -512, -1023);
396                 move_to(  512, -1023);
397                 move_to( -512,  -1023);
398                 move_to( -512,  0);
399                 move_to(  512,  0);
400                 move_to(  512, 1023);
401             break;
402             case 9:
403                 move_to(  -512, 0);
404                 move_to(  512, 0);
405                 move_to(  512, 1023);
406                 move_to(  -512,  1023);
407                 move_to(  -512,  -1023);
408                 move_to(  512, -1023);
409                 move_to(  -512,  -1023);
410                 move_to(  -512,  1023);
411                 move_to(  512, 1023);
412                 move_to(  512, 0);
413             break;
414         }
415
416         // Drive circle
417         // move_circle(1023); // drive a circle (radius)
418
419         /*
420         // Endless loop for continuous circular motion
421         // Drive circle
422         static int radius = 512;
423         static int max_radius = 1024;
424         static int radius_step = 70;
425         move_circle(radius);
426
427         // Adjust radius
428         radius += radius_step;
429
430         // If the radius reaches the minimum, let it grow again
```

```
431         if (radius <= 0) {
432             radius = 0;
433             radius_step = -radius_step; // reverse direction (grow)
434         }
435         // If the radius reaches the maximum, let it shrink again
436         else if (radius >= max_radius) {
437             radius = max_radius;
438             radius_step = -radius_step; // reverse direction (shrink)
439         }
440         */
441     }
442 }
```

Listing B.1: Atmega 328P C-Code

C VHDL Code

C.1 Treiber für 12-Bit-ADC MCP3201

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.NUMERIC_STD.ALL;
4
5  entity mcp3201_reader is
6      Port (
7          clk_800kHz     : in  std_logic;                        -- SPI
               Clock ( 800kHz weil 3,3V Versorgung)
8          trigger_800kHz :  in  std_logic;                       -- SPI
               Trigger fuer Modul
9          clk_100        : in  std_logic;                        --
               Sys_clk 100MHz
10         reset          : in  std_logic;                        --
               reset weil brauchen wir
11         start          : in  std_logic;                        --
               Trigger zum Starten
12         d_out          : in  std_logic;                        --
               Daten vom ADC
13         cs_n           : out std_logic;                        -- Chip
                Select (aktiv low)
14         adc_value      : out std_logic_vector(11 downto 0);    --
               Ergebnis
15         done           : out std_logic;                        -- High
               , wenn fertig
16         spi_clk_out    : out std_logic                         -- SPI
               CLK fuer Chip
17     );
18 end mcp3201_reader;
19
```

```
20 architecture Behavioral of mcp3201_reader is
21
22 type state_type is (state_IDLE, state_WAIT, state_START, state_READ,
       state_DONE);
23     signal state      : state_type := state_IDLE;
24
25     signal bit_count  : integer range 0 to 15 := 0;
26     signal shift_reg  : std_logic_vector(15 downto 0) := (others =>
           '0');
27     signal result     : std_logic_vector(11 downto 0) := (others =>
           '0');
28     signal cs         : std_logic := '1';
29     signal done_int   : std_logic := '0';
30     signal start_delay_counter: integer range 0 to 4 := 0;
31     signal cs_delayed   : std_logic :='1';
32
33 begin
34
35     -- Durchreichen des SPI-Takts
36     cs_n <= cs_delayed;
37     adc_value <= result;
38     done <= done_int;
39     spi_clk_out <= clk_800kHz;
40
41 process(clk_100, reset)
42 begin
43     if reset = '0' then
44         state <= state_IDLE;
45         cs <= '1';
46         cs_delayed <= '1';
47         bit_count <= 0;
48         shift_reg <= (others => '0');
49         result <= (others => '0');
50         done_int <= '0';
51         start_delay_counter <= 0;
52     elsif rising_edge(clk_100) then
53         -- CS verzoegert 1 Takt
54         cs_delayed <= cs;
55         done_int <= '0'; -- done ist nur 1 Takt high
56
```

```
57          if trigger_800kHz = '1' then
58              case state is
59                  when state_IDLE =>
60                      bit_count <= 0;
61                      shift_reg <= (others => '0');
62                      if start = '1' then
63                          cs <= '0'; -- Beginne Uebertragung
64                          state <= state_WAIT;
65                      end if;
66
67                  when state_WAIT =>
68                      if cs_delayed = '0' then
69                          state <= state_START;
70                      end if;
71
72                  when state_START =>
73                       -- kurze Pause fuer die Nullbits
74                      if start_delay_counter < 1 then
75                         start_delay_counter <= start_delay_counter +
                               1;
76                      else
77                          start_delay_counter <= 0;
78                          state <= state_READ;
79                      end if;
80
81                  when state_READ =>
82                      shift_reg <= shift_reg(14 downto 0) & d_out;
83                      bit_count <= bit_count + 1;
84                      if bit_count = 15 then
85                          cs <= '1'; -- Ende der Uebertragung
86                          result <= shift_reg(14 downto 3); -- 12 Bits
87                          state <= state_DONE;
88                      end if;
89
90                  when state_DONE =>
91                      done_int <= '1';
92                      state <= state_IDLE;
93
94                  when others =>
95                      state <= state_IDLE;
```

```
96              end case;
97          end if;
98      end if;
99  end process;
100
101 end Behavioral;
```

Listing C.1: VHDL-Code für den 12 Bit ADC MCP3201

C.2 DAC-Treiber: Treiber für 12-Bit-DAC DAC121S101

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.NUMERIC_STD.ALL;
4
5  entity dac121S101_writer is
6      Port (
7          clk_800kHz      : in  std_logic;                        -- SPI
               Clock (fuer DAC)
8          trigger_800kHz : in  std_logic;                         -- SPI
               Trigger (optional)
9          clk_100         : in  std_logic;                        --
               System Clock 100 MHz
10         reset           : in  std_logic;                        --
               Reset (active low)
11         start           : in  std_logic;                        --
               Trigger zum Starten
12         dac_data        : in  std_logic_vector(11 downto 0);  -- 12
               Bit DAC-Wert
13         cs_n            : out std_logic;                        -- Chip
                Select (aktiv low)
14         din             : out std_logic;                        -- MOSI
                -> DAC
15         done            : out std_logic;                        -- High
               , wenn fertig
16         spi_clk_out     : out std_logic                         -- SPI
               CLK fuer DAC
17     );
18 end dac121S101_writer;
```

```
19
20 architecture Behavioral of dac_writer is
21
22     type state_type is (state_IDLE, state_START, state_WRITE,
           state_DONE);
23     signal state        : state_type := state_IDLE;
24
25     signal bit_count    : integer range 0 to 15 := 0;
26     signal shift_reg    : std_logic_vector(15 downto 0) := (others =>
           '0');
27     signal cs           : std_logic := '1';
28     signal done_int     : std_logic := '0';
29     signal cs_delayed   : std_logic := '1';
30     signal control_bits : std_logic_vector(3 downto 0) := "0000";
31
32 begin
33
34     -- Outputs
35     cs_n       <= cs_delayed;
36     din        <= shift_reg(15);  -- MSB zuerst
37     done       <= done_int;
38     spi_clk_out <= clk_800kHz;
39
40 process(clk_100, reset)
41 begin
42     if reset = '0' then
43         state      <= state_IDLE;
44         cs         <= '1';
45         cs_delayed <= '1';
46         shift_reg  <= (others => '0');
47         bit_count  <= 0;
48         done_int   <= '0';
49     elsif rising_edge(clk_100) then
50         -- CS verzoegert um 1 Takt
51         cs_delayed <= cs;
52         done_int   <= '0';
53
54         case state is
55             when state_IDLE =>
56                 if start = '1' then
```

```
57                      -- Paket vorbereiten: 4 Control-Bits + 12
                           Datenbits
58                      shift_reg <= control_bits & dac_data;
59                      bit_count <= 0;
60                      cs        <= '0';  -- CS low zum Start
61                      state     <= state_START;
62                  end if;
63
64              when state_START =>
65                  -- Warten auf erste SPI Clock Flanke
66                  if trigger_800kHz = '1' then
67                      state <= state_WRITE;
68                  end if;
69
70              when state_WRITE =>
71                  if trigger_800kHz = '1' then
72                      -- Bits nach MSB verschieben
73                      shift_reg <= shift_reg(14 downto 0) & '0';
74                      bit_count <= bit_count + 1;
75
76                      if bit_count = 15 then
77                          cs    <= '1';   -- CS wieder high
78                          state <= state_DONE;
79                      end if;
80                  end if;
81
82              when state_DONE =>
83                  done_int <= '1';
84                  state    <= state_IDLE;
85
86              when others =>
87                  state <= state_IDLE;
88          end case;
89      end if;
90  end process;
91
92  end Behavioral;
```

Listing C.2: VHDL-Code für den 12 Bit DAC DAC121S101

C.3 12 Bit UART-Schnitstelle

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.NUMERIC_STD.ALL;
4
5  entity UART_12Bit_Sender is
6      Port (
7          clk_100       : in  STD_LOGIC;
8          reset         : in  STD_LOGIC;
9          send      : in  STD_LOGIC;  -- Trigger zum Senden
10         data_12bit: in  STD_LOGIC_VECTOR(11 downto 0);
11         tx        : out STD_LOGIC;
12         busy      : out STD_LOGIC
13     );
14 end UART_12Bit_Sender;
15
16 architecture Behavioral of UART_12Bit_Sender is
17
18     component UART_Sender_8bit
19         Port (
20             clk_100   : in  STD_LOGIC;
21             reset       : in  STD_LOGIC;
22             tx_start  : in  STD_LOGIC;
23             data_in   : in  STD_LOGIC_VECTOR(7 downto 0);
24             tx        : out STD_LOGIC;
25             busy      : out STD_LOGIC
26         );
27     end component;
28
29     signal uart_data_8bit     : STD_LOGIC_VECTOR(7 downto 0);
30     signal uart_start_8bit    : STD_LOGIC := '0';
31     signal uart_busy_8bit     : STD_LOGIC;
32     signal send_state    : integer range 0 to 2 := 0;
33     signal busy_int :std_logic := '0';
34     type state_type is (state_idle, state_send_byte_1,
          state_wait_one_cycle_1, state_wait_byte_1,
          state_wait_one_cycle_2, state_send_byte_2, state_wait_byte_2);
35     signal state       : state_type := state_idle;
36
```

```
37 begin
38
39     UART: UART_Sender_8bit
40         port map (
41             clk_100  => clk_100,
42             reset    => reset,
43             tx_start => uart_start_8bit,
44             data_in  => uart_data_8bit,
45             tx       => tx,
46             busy     => uart_busy_8bit
47         );
48
49
50 process(clk_100, reset)
51     begin
52         if reset = '0' then
53             send_state      <= 0;
54             busy_int <= '0';
55             uart_start_8bit <= '0';
56             uart_data_8bit  <= (others => '0');  -- Initialisierung
                   hinzugefuegt
57         elsif rising_edge(clk_100) then
58             uart_start_8bit <= '0';  -- Start-Impuls nur fuer 1 Takt
59
60             case state is
61                 when state_idle =>
62                     if send = '1' and uart_busy_8bit = '0' then
63                         busy_int <= '1';
64                         state      <= state_send_byte_1;
65                     else
66                         busy_int <= '0';
67                     end if;
68                 when state_send_byte_1 =>
69                     uart_data_8bit  <= data_12bit(7 downto 0);  --
                           LSB
70                     uart_start_8bit <= '1';
71                     state <= state_wait_one_cycle_1;
72                 when state_wait_one_cycle_1 =>
73                     state <= state_wait_byte_1;
74                 when state_wait_byte_1 =>
```

```
75                      if uart_busy_8bit = '0' then
76                          state <= state_send_byte_2;
77                      end if;
78
79                  when state_send_byte_2 =>
80                          uart_data_8bit  <= "0000" & data_12bit(11
                                downto 8);  -- MSB (nur unteres Nibble
                                gueltig)
81                          uart_start_8bit <= '1';
82                          state      <= state_wait_one_cycle_2;
83                  when state_wait_one_cycle_2 =>
84                          state <= state_wait_byte_2;
85                  when state_wait_byte_2 =>
86                          if uart_busy_8bit = '0' then
87                              state <= state_idle;  -- Zurueck zum
                                    Anfang
88                          end if;
89                  end case;
90          end if;
91      end process;
92
93      busy <= busy_int;
94
95  end Behavioral;
```

Listing C.3: Hauptmodul 12 Bit UART-Schnitstelle

C.4 8 Bit UART-Sender

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3   use IEEE.NUMERIC_STD.ALL;
4
5   entity UART_Sender_8bit is
6       Port (
7           clk_100       : in  STD_LOGIC;
8           reset         : in  STD_LOGIC;
9           tx_start      : in  STD_LOGIC;
10          data_in       : in  STD_LOGIC_VECTOR(7 downto 0);
```

```
11          tx          : out STD_LOGIC;
12          busy        : out STD_LOGIC
13      );
14  end UART_Sender_8bit;
15
16  architecture Behavioral of UART_Sender_8bit is
17
18      type state_type is (idle, start_bit, data_bits, stop_bit);
19      signal state        : state_type := idle;
20      signal bit_cnt      : integer range 0 to 7 := 0;
21      signal shift_reg    : STD_LOGIC_VECTOR(7 downto 0) := (others =>
            '0');
22      signal tx_reg       : STD_LOGIC := '1';
23      signal clk_cnt      : integer := 0;
24
25      constant CLK_FREQ     : integer := 100000000; -- 50 MHz
26      constant BAUD_RATE    : integer := 115200;
27      constant BAUD_TICKS   : integer := CLK_FREQ / BAUD_RATE;
28
29  begin
30
31      process(clk_100, reset)
32      begin
33          if reset = '0' then
34              state       <= idle;
35              bit_cnt     <= 0;
36              shift_reg   <= (others => '0');
37              tx_reg      <= '1';
38              busy        <= '0';
39              clk_cnt     <= 0;
40          elsif rising_edge(clk_100) then
41              case state is
42                  when idle =>
43                      tx_reg <= '1';
44                      busy <= '0';
45                      if tx_start = '1' then
46                          shift_reg <= data_in;
47                          state <= start_bit;
48                          clk_cnt <= 0;
49                          busy <= '1';
```

```
50                      end if;
51
52                  when start_bit =>
53                      tx_reg <= '0'; -- Startbit
54                      if clk_cnt = BAUD_TICKS - 1 then
55                          clk_cnt <= 0;
56                          state <= data_bits;
57                          bit_cnt <= 0;
58                      else
59                          clk_cnt <= clk_cnt + 1;
60                      end if;
61
62                  when data_bits =>
63                      tx_reg <= shift_reg(bit_cnt);
64                      if clk_cnt = BAUD_TICKS - 1 then
65                          clk_cnt <= 0;
66                          if bit_cnt = 7 then
67                              state <= stop_bit;
68                          else
69                              bit_cnt <= bit_cnt + 1;
70                          end if;
71                      else
72                          clk_cnt <= clk_cnt + 1;
73                      end if;
74
75                  when stop_bit =>
76                      tx_reg <= '1'; -- Stopbit
77                      if clk_cnt = BAUD_TICKS - 1 then
78                          clk_cnt <= 0;
79                          state <= idle;
80                          busy <= '0';
81                      else
82                          clk_cnt <= clk_cnt + 1;
83                      end if;
84              end case;
85          end if;
86      end process;
87
88      tx <= tx_reg;
89
```

```
90 end Behavioral;
```

Listing C.4: 8 Bit UART-Sender

C.5 12 Bit Sinus Generator

```
1
2  library ieee;
3  use ieee.std_logic_1164.all;
4  use ieee.numeric_std.all;
5
6  entity dac_sin_gen is
7      port(
8          clk_100  : in std_logic;  -- 100 MHz Systemtakt
9          reset    : in std_logic;
10         start    : out std_logic;  -- DAC Transfer starten
11         din      : out std_logic_vector(11 downto 0); -- DAC Daten
12         done     : in std_logic;    -- DAC Transfer abgeschlossen
13         trigger_2kHz : in std_logic;
14         cs_n     : in std_logic
15     );
16 end entity;
17
18 architecture rtl of dac_sin_gen is
19
20     -- State Machine
21     type state_type is (IDLE, START_TRANSFER, WAIT_cs, WAIT_done);
22     signal state : state_type := IDLE;
23
24     -- Sinus-LUT (12-Bit DAC)
25     type sin_table_type is array(0 to 255) of unsigned(11 downto 0);
26     constant sin_table : sin_table_type := (
27         to_unsigned(2048,12),
28 to_unsigned(2098,12),
29 to_unsigned(2148,12),
30 to_unsigned(2198,12),
31 --.
32 --.     genau 255 Werte
33 --.     hier nich dargestellt
```

```
--.
to_unsigned(1997,12));

    signal sin_index : integer range 0 to 255 := 0;
    signal dac_value : unsigned(11 downto 0) := (others => '0');

begin

    process(clk_100, reset)
    begin
        if reset = '0' then
            state <= IDLE;
            start <= '0';
            sin_index <= 0;
            dac_value <= (others => '0');
            din <= (others => '0');
        elsif rising_edge(clk_100) then




            -- State Machine
            case state is
                when IDLE =>
                    start <= '0';
                    if trigger_2kHz = '1' then
                        dac_value <= sin_table(sin_index);
                        din <= std_logic_vector(dac_value);
                        state <= START_TRANSFER;
                    end if;

                when START_TRANSFER =>
                    start <= '1';
                    state <= WAIT_cs;

                when WAIT_cs =>
                    if cs_n = '0' then
                        start <= '0';
                        state <= WAIT_done;
                    end if;
```

```
74
75                  when WAIT_done =>
76                      if done = '1' then
77                          sin_index <= (sin_index + 1) mod 256;
78                          state <= IDLE;
79                      end if;
80
81              end case;
82          end if;
83      end process;
84
85  end architecture;
```

Listing C.5: 12 Bit Sinusgenerartor

C.6 Gesamtsystem in VHDL

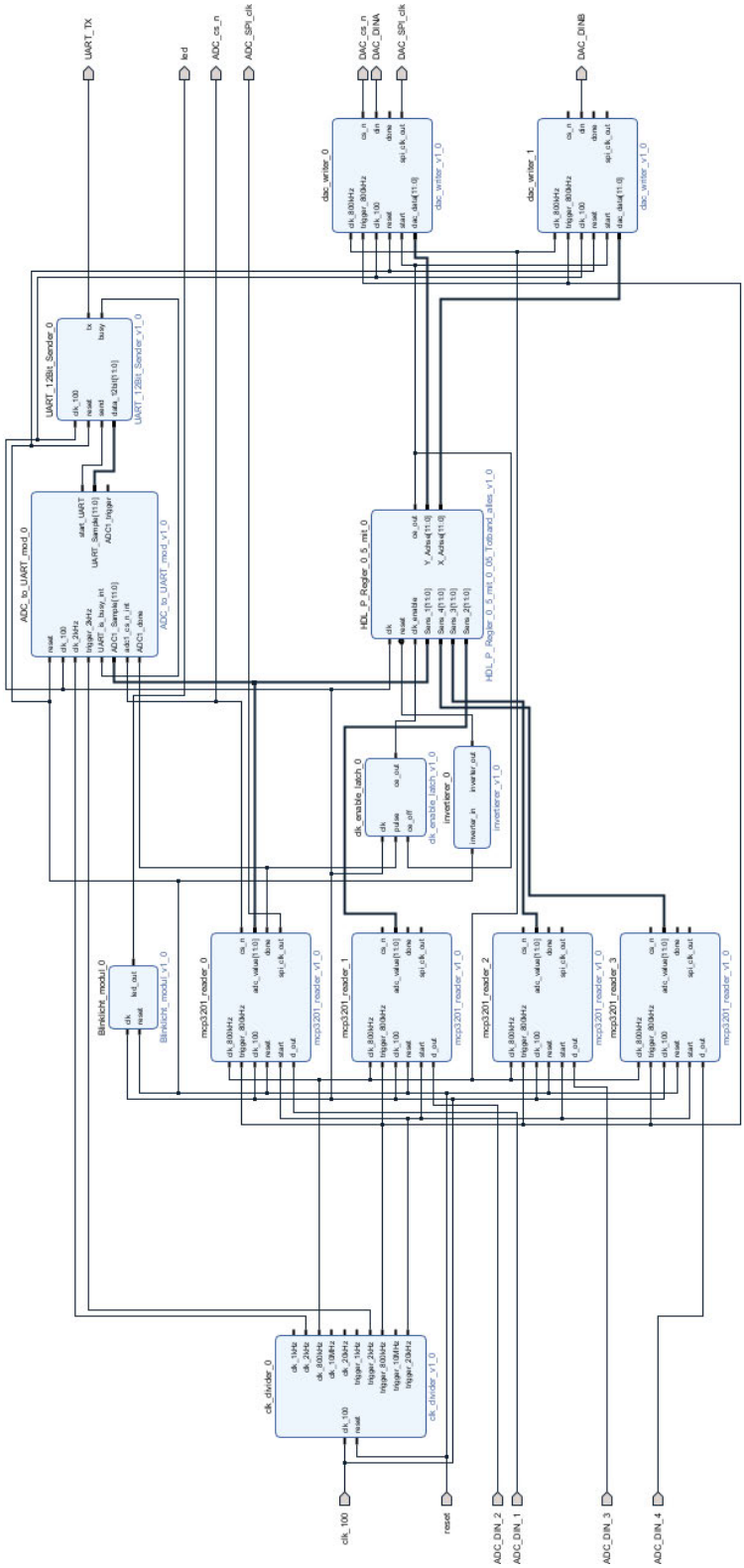

Abbildung C.1: Aufbau des gesamten VHDL-Codes mit IP-Blöcken

D Layout

D.1 DAC Treiber

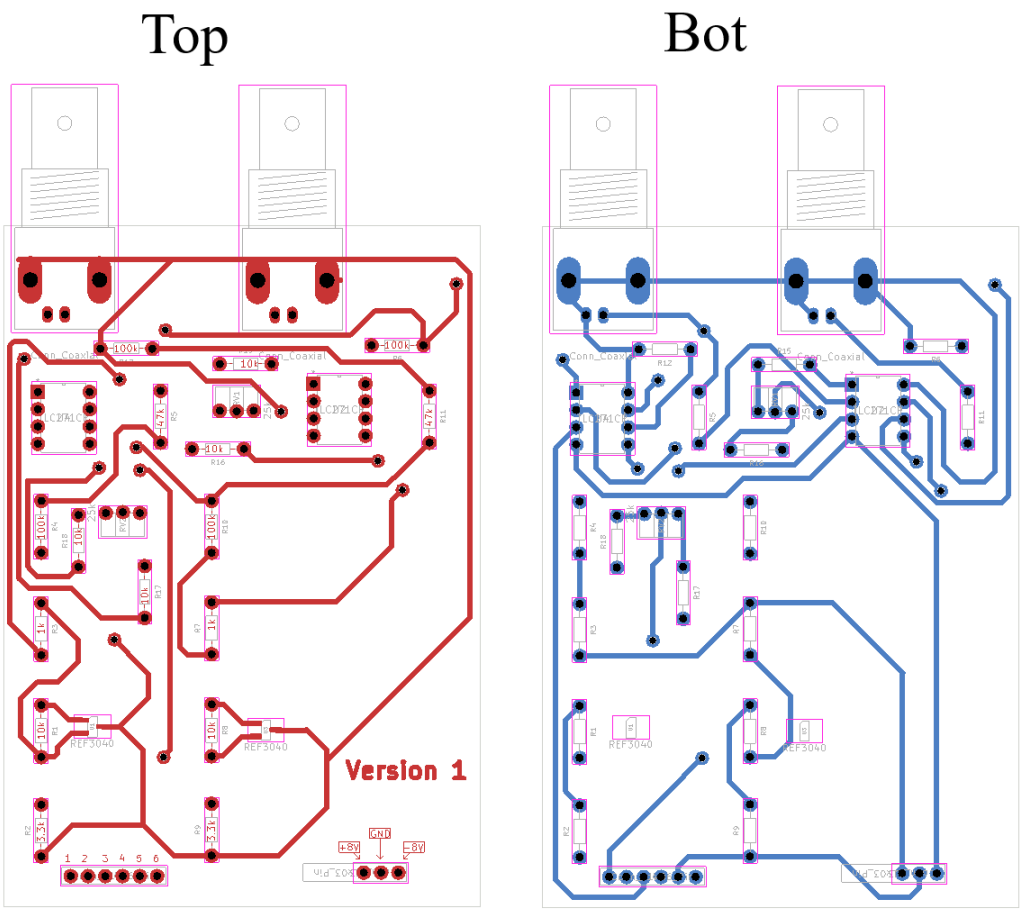

Abbildung D.1: Layout des DAC-Treiberboards

D.2 ADC Board

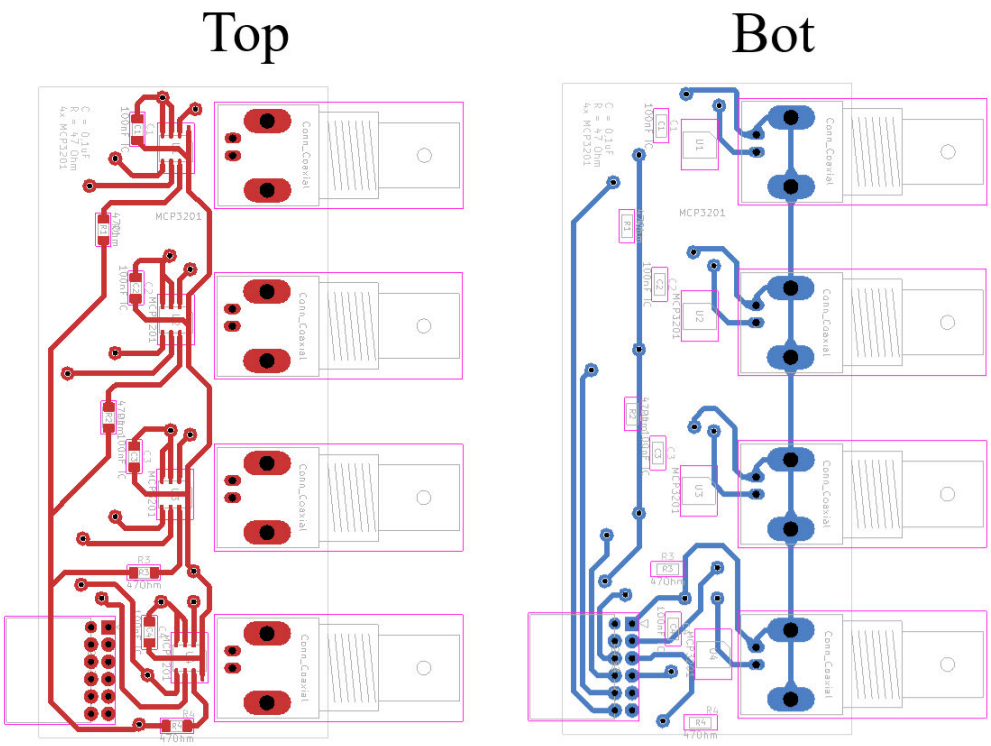

Abbildung D.2: Layout des gefertigten ADC-Boards mit vier ADCs

D.3 Adapterboard

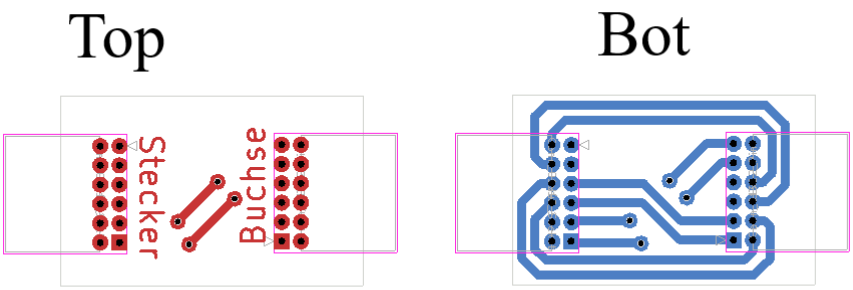

Abbildung D.3: Layout Adapterboard Schnittstelle Basys 3 zu ADC-Board

D.4 Sensorverstärker

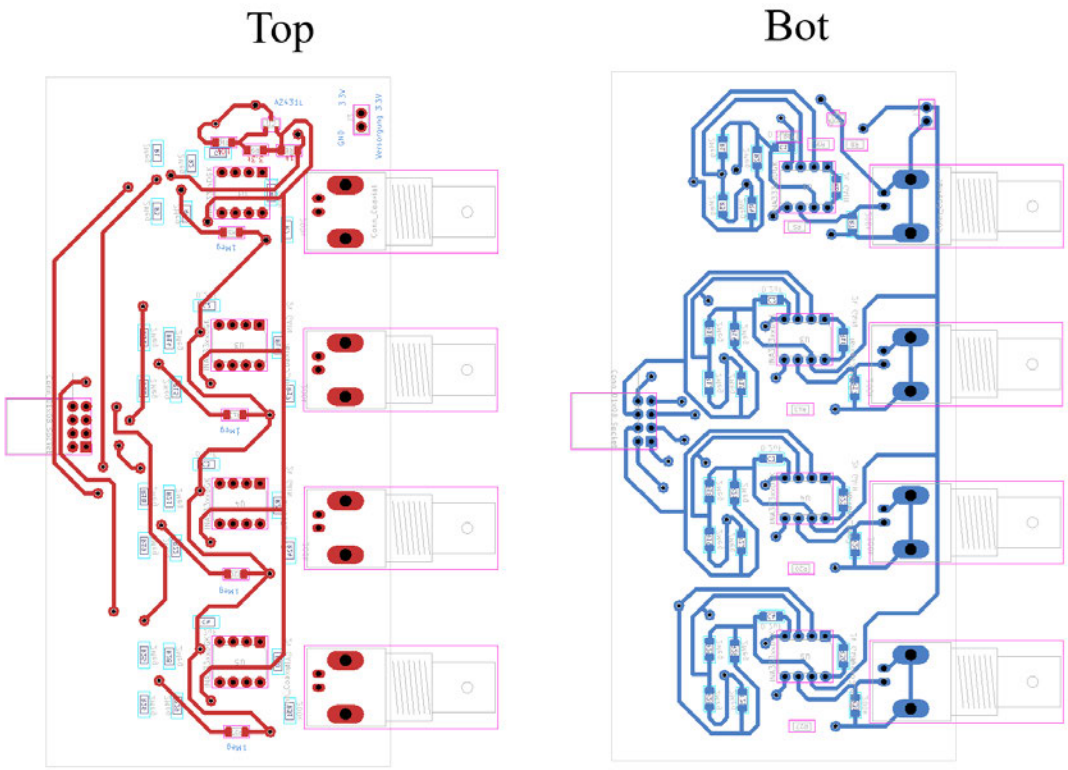

Abbildung D.4: Layout vier Sensorverstärker mit INA333

D.5 Demonstrator ATmega328p

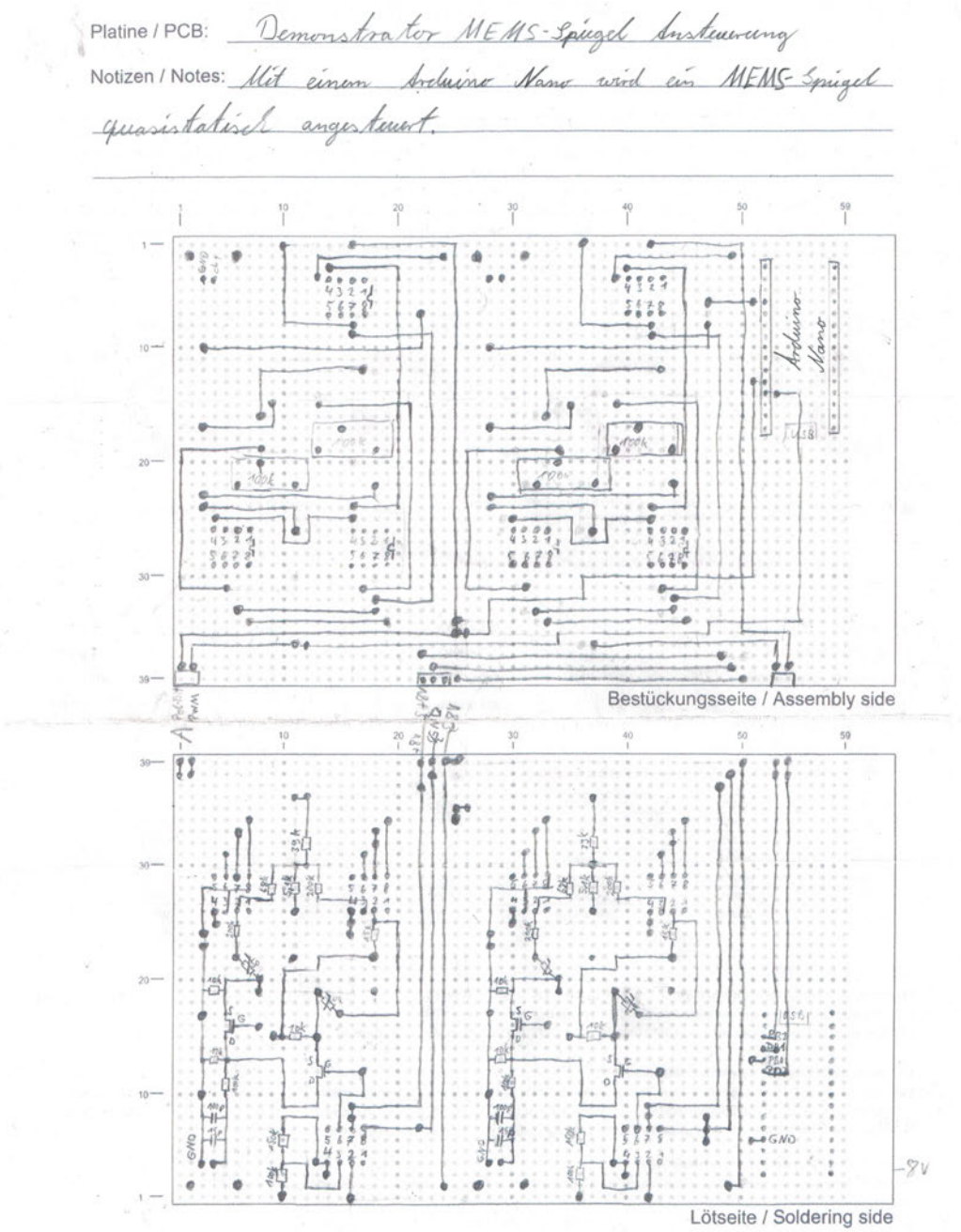

Abbildung D.5: Layout der Lochrasterplatine zur Spiegelansteuerung mittels Mikrocontroller

E Python

E.1 UART Datenlogger

```
import serial
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from collections import deque
import threading
import time
import numpy as np

# Funktion zum Lesen eines 12-Bit-Samples
def read_12bit_sample(ser):
    raw = ser.read(2)
    if len(raw) != 2:
        return None
    lsb = raw[0]
    msb = raw[1] & 0x0F  # Nur untere 4 Bits gueltig
    value = (msb << 8) | lsb
    return value

# Thread-Funktion zum kontinuierlichen Einlesen der Daten
def read_serial_thread(ser, buffer, stop_event):
    while not stop_event.is_set():
        sample = read_12bit_sample(ser)
        if sample is not None:
            buffer.append(sample)
        else:
            time.sleep(0.001)

# Serielle Verbindung konfigurieren
ser = serial.Serial(
```

```
30      port='COM9',        # <-- ggf. anpassen
31      baudrate=115200,
32      timeout=0.1
33  )
34
35  # Plot-Vorbereitung
36  BUFFER_SIZE = 2000
37  data_buffer = deque([0]*BUFFER_SIZE, maxlen=BUFFER_SIZE)  #
        Ringpuffer
38
39  fig, ax = plt.subplots()
40
41  x = np.arange(BUFFER_SIZE)
42  line, = ax.plot(x, data_buffer)
43
44  # Linke x-Achse fuer Spannung (0 bis 3.3 V)
45  ax.set_xlim(0, BUFFER_SIZE-1)
46  ax.set_ylim(0, 4095)
47  ax.set_xlabel("Samples")
48  ax.set_ylabel("Integer Wert (0-4095)")
49
50  # Sekundaere x-Achse (oben) fuer Spannung
51  def int_to_voltage(x_int):
52      return x_int * (3.3 / 4095)
53
54  def voltage_to_int(x_volt):
55      return x_volt * (4095 / 3.3)
56
57  secax = ax.secondary_yaxis('right', functions=(int_to_voltage,
        voltage_to_int))
58  secax.set_ylabel("Spannung [V]")
59  secax.set_ylim(0, 3.3)
60
61  ax.grid(True)
62
63  # Stop-Event fuer sauberes Beenden
64  stop_event = threading.Event()
65
66  # Lese-Thread starten
```

```
67 thread = threading.Thread(target=read_serial_thread, args=(ser,
       data_buffer, stop_event))
68 thread.start()
69
70 # Update-Funktion fuer Animation
71 def update(frame):
72     line.set_ydata(data_buffer)
73     return line,
74
75 ani = animation.FuncAnimation(fig, update, interval=50, blit=True)
76
77 try:
78     print("Starte Live-Plot... (STRG+C zum Beenden)")
79     plt.show()
80 except KeyboardInterrupt:
81     print("Beendet.")
82 finally:
83     stop_event.set()
84     thread.join()
85     ser.close()
```

Listing E.1: Python UART Datenlogger

Erklärung zur selbständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original