HAW HAMBURG

Bachelorarbeit

Jakob Häge

Hard- und Softwareentwicklung eines magnetischen Sensor-Arrays für ein Schraubwerkzeug

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Jakob Häge

Hard- und Softwareentwicklung eines magnetischen Sensor-Arrays für ein Schraubwerkzeug

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Elektro- und Informationstechnik*
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr.-Ing. Karl-Ragmar Riemschneider
Zweitgutachter: Prof. Dr.-Ing. Lutz Leutelt

Eingereicht am: 15. Juni 2023

Jakob Häge

Thema der Arbeit

Hard- und Softwareentwicklung eines magnetischen Sensor-Arrays für ein Schraubwerkzeug

Stichwörter

Magnetsensor, TMR-Sensor, Sensorarray, rund, Schraubwerkzeug, Vorspannkraft, Drehmoment

Kurzzusammenfassung

Um eine präzise Schätzung der in einer Schraubverbindung wirkenden Vorspannkraft zu erreichen, ist es von Interesse, den Winkel der Torsion einer Schraube mit hoher Auflösung zu ermitteln. Für die Entwicklung eines Schraubwerkzeugs mit integrierter kontakloser Torsionswinkelmessung wird im Rahmen dieser Arbeit ein rundes magnetisches Sensorarray bestehend aus 16 kreisförmig angeordneten zweidimensionalen TMR-Sensoren entwickelt und überprüft.

Jakob Häge

Title of Thesis

Hardware and Software Development of a Magnetic Sensor Array for a Screw Fastening Tool

Keywords

Magnetic Sensor, TMR-Sensor, Sensorarray, Circular, Wrench, Tool, Preload Force, Torque

Abstract

In order to achieve a precise estimate of the preload force acting in a bolted joint, it is of interest to determine the angle of torsion of a bolt with high resolution. For the development of a screw fastening tool with integrated contactless torsion angle measurement, a circular magnetic sensor array consisting of 16 circularly arranged two-dimensional TMR sensors is developed and tested within the scope of this work.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

ADC Analog Digital Converter.

CSV Comma Seperated Value.

GND Masse.

GPIO General Purpose Input-Output.

IC Integrated Circuit.

ISAR Integrated Sensor-Arrays.

LSB Least Significant Bit.

LVTTL Low-Voltage Transistor-Transistor Logic.

MOSFET Metall-Oxid-Halbleiter-Feldeffekttransistor.

MUX Multiplexer.

TMR Tunnel Magnetoresistance.

TSSOP Thin Film Small Outline Package.

UART Universal Asynchronous Receiver / Transmitter.

USB Universal Serial Bus.

VCC Betriebsspannung.

1 Einleitung

1.1 Einführung

Dieser Arbeit voran geht die Planung eines Schraubwerkzeuges mit integrierter Messung des Torsionswinkels einer Schraube kontaktlos und ohne mechanische Komponenten. Ziel der Entwicklung dieses Werkzeuges ist die präzise Bestimmung der auf eine Schraubverbindung wirkende Vorspannkraft um genauere Auslegungen von Schrauben zu ermöglichen. Um dies zu realisieren, soll ein kleiner Drehwinkel durch magnetische Winkelsensoren gemessen werden. Aufgrund der Konstruktionsweise einer Schraube oder der Nuss des Werkzeugs bietet sich die Messung über ein rundes Sensorarray an. Ein Vorschlag für das Design eines runden Sensorarrays ist in Abbildung 1.1 gezeigt. Ein rundes Sensorarray bietet des Weiteren mehrere theoretische Vorteile gegenüber einer rechteckigen Anordnung. Durch Fouriertransformation der Sensorwerte kann die Messung von zwei, sich überlagernden Feldern ermöglicht werden. Dies könnte die Messung eines Referenzdrehwinkels erleichtern. Auch kann durch die Fouriertransformation das Nutzsignal von Störsignalen, ausgehend von überlagernden Streufeldern, getrennt werden.

Inhaltlich wird in dieser Arbeit wird die Planung, Entwicklung und Überprüfung eines runden magnetischen Sensorarrays beschrieben. Es werden zuerst die Grundlagen der Funktionsweise des Arrays genauer beschrieben. Anschließend wird die Planung zur Entwicklung der Hard- und Software des Arrays erklärt. Es wird genau auf die Auslegung der Bauteile und die Schaltungsentwürfe eingegangen. Die Auswahl und Programmierung der logischen Elemente werden beschrieben, welche es ermöglichen, die Platine in einem Versuchsaufbau auszulesen. Im letzten Teil der Entwicklung wird der Messaufbau dargestellt. Es wird in diesem Teil genauer auf die Erprobung der Platine und die Planung der Tests eingegangen. Im nächsten Kapitel werden die Messergebnisse interpretiert. Die Erkenntnisse werden außerdem im Kontext der Anwendung als Schraubwerkzeug bewertet. Abschließend wird die Arbeit zusammengefasst und ein Fazit gezogen.

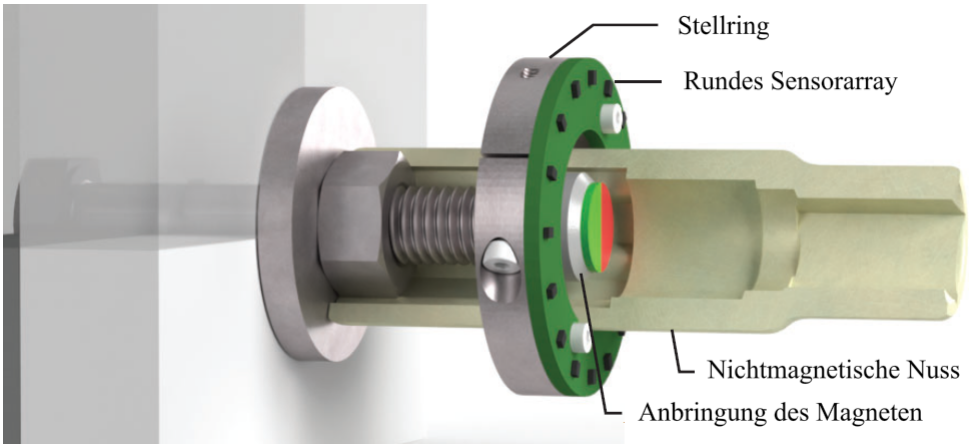

Abbildung 1.1: Vorschlag für Design des Schraubwerkzeuges aus "Magnetic Sensor Array for Determining the Assembly Torsion and Preload of a Bolted Joint"[8]

1.2 Grundlagen

In diesem Kapitel werden die, dieser Arbeit zu Grunde liegenden Sachverhalte erklärt. Es wird auf das in dieser Arbeit behandelte Problem der Vorspannkraftbestimmung eingegangen sowie Bestimmungsmethoden erklärt. Anschließend wird genauer in die Methode eingeführt, auf welcher die entwickelte Platine basiert.

1.2.1 Ermittelung der Vorspannkraft einer Schraubverbindung

Allgemein

Das Anziehdrehmoment einer Schraubverbindung M_A lässt sich durch Formel 1.1 bestimmen. Es setzt sich zusammen aus dem Moment des Schraubengewindes M_G (Formel 1.2) und dem Moment des Schraubenkopfes M_K (Formel 1.3). Diese sind abhängig von den Schraubendimensionen und ihrer Schmierung. Hierbei ist d_2 der Flankendurchmesser des Gewindes, μ_K der Reibungskoeffizient des Schraubenkopfes und μ_G der des Schraubengewindes. Das Moment der Kopfreibung ist außerdem abhängig vom Reibungsdurchmesser, welcher sich aus dem äußeren Durchmesser der Mutter d_w und ihrem inneren Durchmesser d_{ha} zusammensetzt. Alle Längen sind der Abbildung 1.2 zu entnehmen.
Beide Formeln der Momente sind des Weiteren abhängig von der Vorspannkraft F_M. Aus diesen Formeln wird ersichtlich, dass die Vorspannkraft bei Ermittelung durch das

Anziehdrehmoment eine Abhängigkeit von den Schraubendimensionen und Reibungskoeffizienten aufweist.

$$M_A = M_G + M_K \tag{1.1}$$

$$M_G = F_M \frac{d_2}{2} \left(\frac{P}{\pi d_2} + \mu_G \frac{1}{\cos(\alpha/2)} \right) \tag{1.2}$$

$$M_K = F_M \mu_K \frac{D_{Km}}{2} \tag{1.3}$$

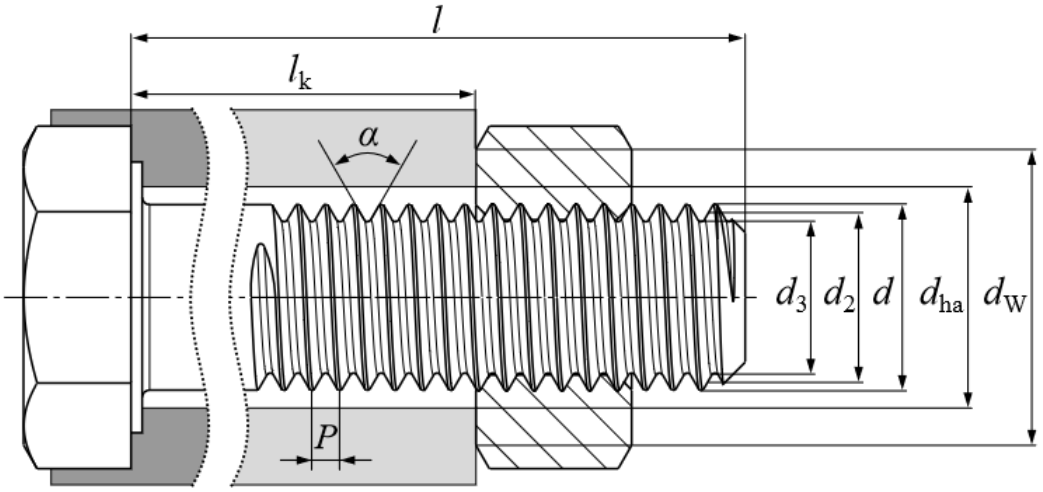

Abbildung 1.2: Darstellung einer Schraubverbindung mit geometrischen Parametern [8]

Konventionelle Methoden

Da es sich beim Anziehdrehmoment um das Moment handelt, welches vom Schraubwerkzeug ausgeht und auf die Schraubverbindung wirkt, kann eine Messung dieses Moments sich einfach mit mechanischen Methoden umsetzen lassen und ein Überschreiten des Moments haptisch oder akustisch signalisiert werden [9].
Die drehmomentbasierte Ermittelung der Vorspannkraft einer Schraubverbindung kann jedoch nur mit eingeschränkter Genauigkeit erfolgen da, wie im letzten Absatz gezeigt, Reibung des Kopfes und Gewindes der Schraube ebenso wie die Vorspannkraft einen

Einfluss auf das am Schraubwerkzeug gemessene Drehmoment haben.
Des Weiteren kann die auf die Schraubverbindung wirkende Vorspannkraft über die Längung der Schraube ermittelt werden. Bei dieser Methode können, im Gegensatz zur Drehmomentbestimmung, Reibungswerte weitestgehend zur Bestimmung der Vorspannkraft vernachlässigt werden. Hierfür müssen jedoch speziell gefertigte Messchrauben mit einem Stifteinsatz genutzt werden. Dies erfordert einen großen Aufwand und erhöht die Kosten. Außerdem wird die Schraube durch den Stifteinsatz strukturell geschwächt und eine Bestimmung der Längendifferenz der Schraube gestaltet sich häufig schwierig und ist meist nur mit hochpräzisem Werkzeug möglich.

Torsionsmethode

Die Methode, auf deren Grundlage das in dieser Arbeit betrachtete Schraubwerkzeug basiert, betrachtet die Vorspannkraft in Abhängigkeit des Torsionswinkels der Schraube. Zu diesem Zweck wird die Schraube als Torsionsfeder angenommen. Über den Torsionswinkel der Schraube kann dann mit der Kenntnis über die Schraubengrößen und Schraubenlängen auf die Vorspannkraft geschlossen werden.
Die Messung des Torsionswinkels gestaltet sich jedoch häufig schwierig. Zum einen ist die Winkeländerung selbst bei großer Vorspannkraft sehr klein und setzt dafür eine hohe Präzision an das Messwerkzeug vorraus. Diese Präzision setzt des weiteren meist eine Referenzmessung des Drehwinkels der Schraube in Referenz zum Schraubwerkzeug bzw. dem Werkstück voraus. In Abbildung 1.3 sind eine Auswahl an Torsionswinkel tabellarisch für verschiedene Schrauben und Vorspannkräfte gezeigt.
Die magnetische Messung des Torsionswinkel weist gegenüber mechanischen und optischen Methoden durch die kontaktlose Anwendung eine hohe Robustheit auf [8]. Es ist jedoch für viele Anwendungsfälle nötig die Präzision der magnetischen Winkelmessung zu erhöhen, um damit auch eine hohe Auflösung des Torsionswinkels zu erreichen.

Größe	F_M (kN)	l_k (mm)				
		60.00	80.00	100.00	120.00	140.00
M6	9.90	3.30°	4.40°	5.50°	6.60°	7.70°
M8	18.10	2.43°	3.24°	4.05°	4.86°	5.67°
M10	28.80	1.92°	2.57°	3.21°	3.85°	4.49°
M12	41.90	1.59°	2.12°	2.65°	3.18°	3.71°
M16	78.80	1.15°	1.53°	1.92°	2.30°	2.68°

Abbildung 1.3: Torsion einer Schraube in Grad(°) bei auf sie wirkender Vorspannkraft F_M. Tabelle für unterschiedliche metrische ISO-Regel-Gewinde und Schraubenlänge l_k

[8]

1.2.2 Drehwinkelbestimmung durch Magnetsensorik (TMR)

Spezifika des Sensortyps

Auf der entwickelten Sensorplatine werden Sensoren des Typs CT310 genutzt. Bei diesen handelt es sich um zweidimensionale, differentielle Tunnel Magnetoresistance (TMR)-Sensoren, welche den magnetischen Tunnelwiderstand ausnutzen, um ein magnetisches Feld in zwei Achsen zu messen und in zwei differentielle Ausgangsspannungen zu wandeln. TMR-Sensoren bieten gegenüber Magnetfeldsensoren mit anderen Technologien (Hall, AMR) einige Vorteile. Sie können etwa Felder bei kleinen Feldstärken mit einer hohen Empfindlichkeit ($\Delta R/R$ über 100%) messen [13]. Durch ihre hohen Ausgangsspannungen kann deshalb auf zusätzliche Verstärkerschaltungen vor einer Analog-Digital-Wandlung verzichtet werden.

Zur Darstellung der Kennlinien des Sensors ist eine Charakterisierung nötig. Diese wird an einem im Rahmen des Projektes Integrated Sensor-Arrays (ISAR) entwickelten Kreuzspulenmessplatz durchgeführt [1]. Die Ergebnisse liegen vor. Durch die Charakterisierung des Sensors lässt sich sein Verhalten in zwei Bereiche einordnen.

1. lineares Verhalten - [-2.7 kA/m $<$ H $<$ 2.7 kA/m]

2. gesättigtes Verhalten - [|H| $>$ $\pm$ 2.7 kA/m]

Im Bereich der Feldstärke von ca. -2.7 kA/m bis 2,7 kA/m kann die Ausgangsspannung in ein lineares Verhältnis zur Feldstärke gesetzt werden. Dieses Verhalten ist in den Abbildungen 1.4 und 1.5 gezeigt. Wird der Betrag des Feldes einer Dimension (z.B X-Richtung) größer als 2.7 kA/m, geht der Sensor in die Sättigung und verliert seine Linearität. Querfelder in der zweiten Dimension (z.B Y-Richtung) des Sensors führen jedoch zur Verzerrung der Kennlinie. Wirkt also bei einer Messung auf der X-Achse ein Feld in Y-Richtung, so sinkt die Empfindlichkeit des Sensors in X-Richtung. Der lineare Bereich des Sensors wird weiter, während die Ausgangsspannung sinkt. Die Sättigung tritt erst bei einer größeren Feldstärke ein.

Der Hersteller des Sensors empfiehlt, den Sensor in einem Feld mit Flussdichte von mindestens 25 mT bis maximal 90 mT zu betreiben [2]. Dies entspricht bei Ausbreitung in Luft einer Feldstärke von etwa 17 kA/m. Die empfohlene Ausrichtung des Sensors ist in Abbildung 1.6 gezeigt. Hierbei wird der Sensor in Sättigung betrieben. Die Stärke des Feldes an der Position des Sensors bei 90°-Drehung des Magneten verändert sich nicht, lediglich die Richtung des Feldes. Dies ist nicht der Fall, wird der Magnet auf gleicher Höhe wie der Sensor neben ihm platziert. Dies wird im weiteren Verlauf der Arbeit betrachtet.

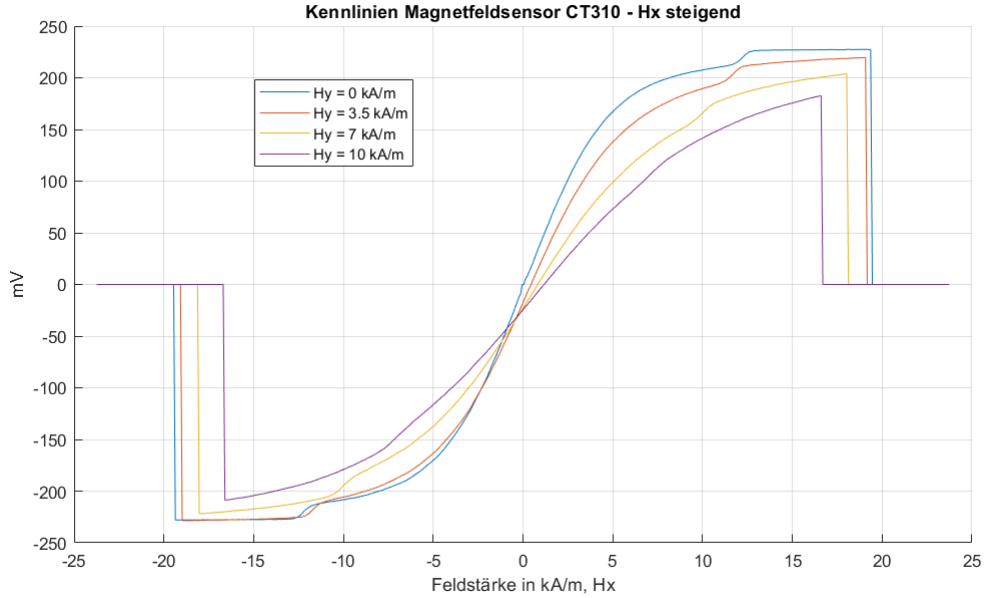

Abbildung 1.4: Kennlinienschar des Sensors CT310 bei steigender Feldstärke in X-Richtung für unterschiedliche Querfelder in Y-Richtung, aufgenommen am Kreuzspulenmessplatz des Labors.

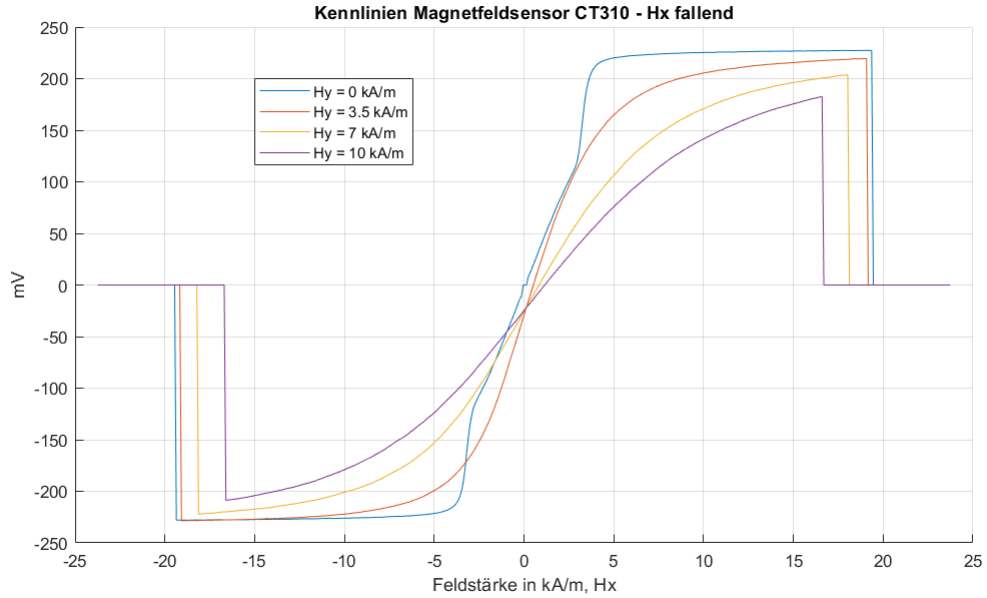

Abbildung 1.5: Kennlinienschar des Sensors CT310 bei fallender Feldstärke in X-Richtung für unterschiedliche Querfelder in Y-Richtung, aufgenommen am Kreuzspulenmessplatz des Labors.

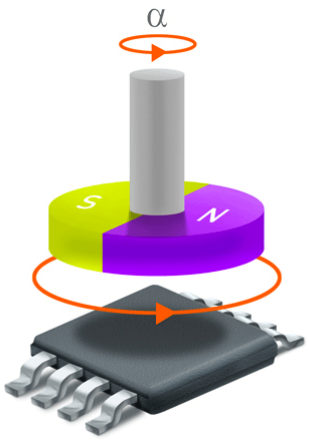

Abbildung 1.6: Darstellung von Drehwinkelmessung mit Sensor des Typen CT310 nach Hersteller [3]

1.2.3 Konzept eines Schraubwerkzeugs mit rundem Sensorarray

Um den Torsionswinkel einer Schraube in einer mit einer Mutter gekonterten Schraubverbindung zweier Werkstücke zu ermitteln, müssen folgende Drehwinkel bekannt sein.

1. Drehwinkel des Schraubenkopfes W_1
2. Drehwinkel der Mutter W_2
3. Drehwinkel des Schraubenendes W_3

In Abb. 1.7 sind diese Drehwinkel in einer Patentzeichnung gezeigt. Außerdem wird in der Zeichnung eine mögliche Methode zur Messung dieser Winkel durch zwei runde Sensorarrays [2],[5] mit Magnetsensoren, sowie zwei magnetische Halbacharrays [3],[4] und einem radial magnetisierten Magneten [7] auf dem Schraubenende gezeigt.

Im weiteren Verlauf dieser Arbeit wird die Entwicklung eines runden Sensorarrays beschrieben und Methoden zur Bestimmung des Drehwinkels eines rotierenden magnetischen Dipols im Zentrum eines Arrays betrachtet. Dies ist vergleichbar mit der Bestimmung des Drehwinkels δ in Abbildung 1.7.

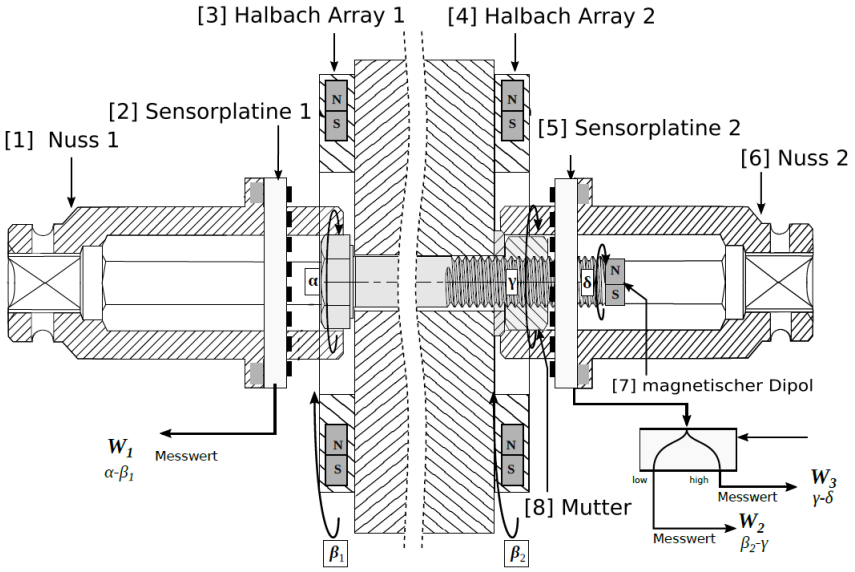

Abbildung 1.7: Patentzeichnung für eine theoretische Umsetzung eines magnetischen Schraubwerkzeugs zur Bestimmung eines Torsionswinkels einer Schraube mit Drehwinkelmessung an Schraubenkopf und Schraubenende. [9]

1.2.4 Ermittelung des Drehwinkels durch Fouriertransformation

Für die Betrachtung des Drehwinkels δ des auf dem Schraubenende angebrachten radial magnetisierten Magneten kann dieser als magnetischer Dipol angenommen werden. Voraussetzung hierfür ist, dass der Radius des Sensorarrays deutlich größer ist als der Radius des Magneten. Es gilt also $R_{Array} >> R_{Magnet}$. Das magnetische Feld eines Dipols kann durch folgende Formel angenähert werden [6]:

$$\vec{H}(\vec{r}) = \frac{1}{4\pi}\left[\frac{3(\vec{y}\cdot\vec{r})\vec{r}}{r^5} - \frac{\vec{y}}{r^3}\right] \tag{1.4}$$

Wird nun ein 2-dimensionaler Magnetfeldsensor(X,Y) mit einem X-Achsenabstand von a von einem magnetischen Dipol mit magnetischem Moment b in Y-Richtung im Ursprung des Koordinatensystems angenommen (vergleichbar mit Abbildung 1.9), so ergeben sich folgende Vektoren

$$(\vec{r}) = \begin{pmatrix} r_x = a \\ r_y = 0 \\ r_z = 0 \end{pmatrix} (\vec{y}) = \begin{pmatrix} y_x = b \\ y_y = 0 \\ y_z = 0 \end{pmatrix} \tag{1.5}$$

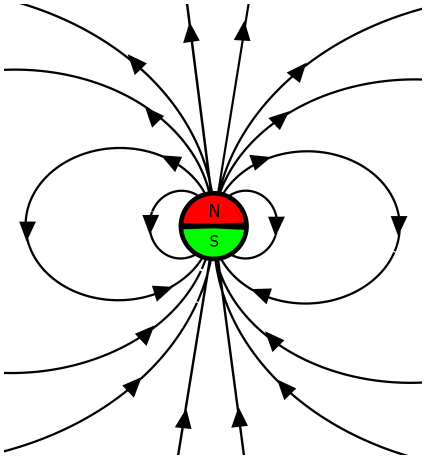

Abbildung 1.8: Ausbreitung des magnetischen Felds eines Dipols

An der Position des Sensors lässt sich die Formel zur Berechnung des Feldes in X-Richtung folgendermaßen vereinfachen.

$$\vec{H}_x(\vec{r}) = \frac{1}{4\pi}\left[\frac{2b}{a^3}\right] \tag{1.6}$$

Wird der magnetische Dipol nun um 90 ° gegen den Uhrzeigersinn gedreht und der Sensor bleibt an der gleichen Position wie zuvor, ergeben sich folgende Vektoren

$$(\vec{r}) = \begin{pmatrix} r_x = a \\ r_y = 0 \\ r_z = 0 \end{pmatrix} (\vec{y}) = \begin{pmatrix} y_x = 0 \\ y_y = b \\ y_z = 0 \end{pmatrix} \tag{1.7}$$

Es lässt sich die Formel zur Berechnung des Feldes nun in Y-Richtung erneut vereinfachen.

$$\vec{H}_y(\vec{r}) = \frac{1}{4\pi}\left[-\frac{b}{a^3}\right] \tag{1.8}$$

Das Verhältnis von H_x zu H_y beträgt also

$$\vec{H}_x(\vec{r}) = -2\vec{H}_y(\vec{r}) \tag{1.9}$$

Das Feld eines magnetischen Dipols weist also auf der Achse seiner Magnetisierung, bei gleichem Abstand, ein Feld mit der doppelten Feldstärke im Vergleich zur orthogonalen Achse auf. Dieses Verhalten und die daraus resultierenden Sensorwerte eines runden Sensorarrays sind in den Abbildungen 1.10 und 1.9 gezeigt. Der magnetische Dipol wird für die folgende Betrachtung nun um einen Drehwinkel ϕ in der X/Y-Ebene verdreht.

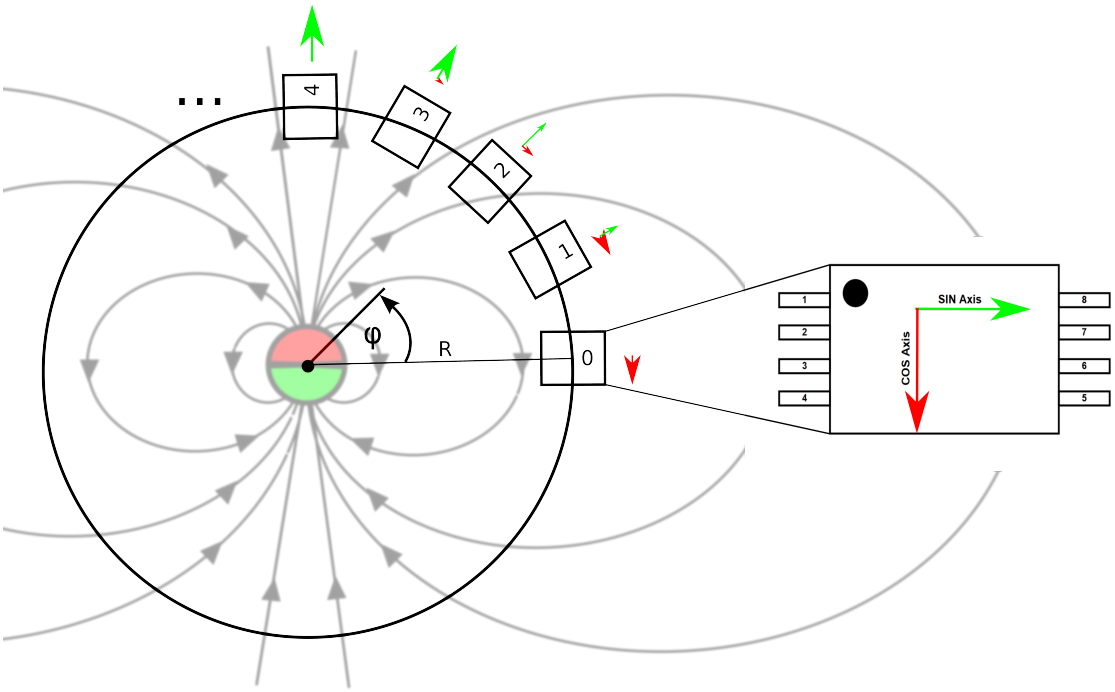

Abbildung 1.9: Skizze der Abtastwerte eines runden Sensorarrays aus Winkelsensoren des Typs CT310 im Feld eines magnetischen Dipols

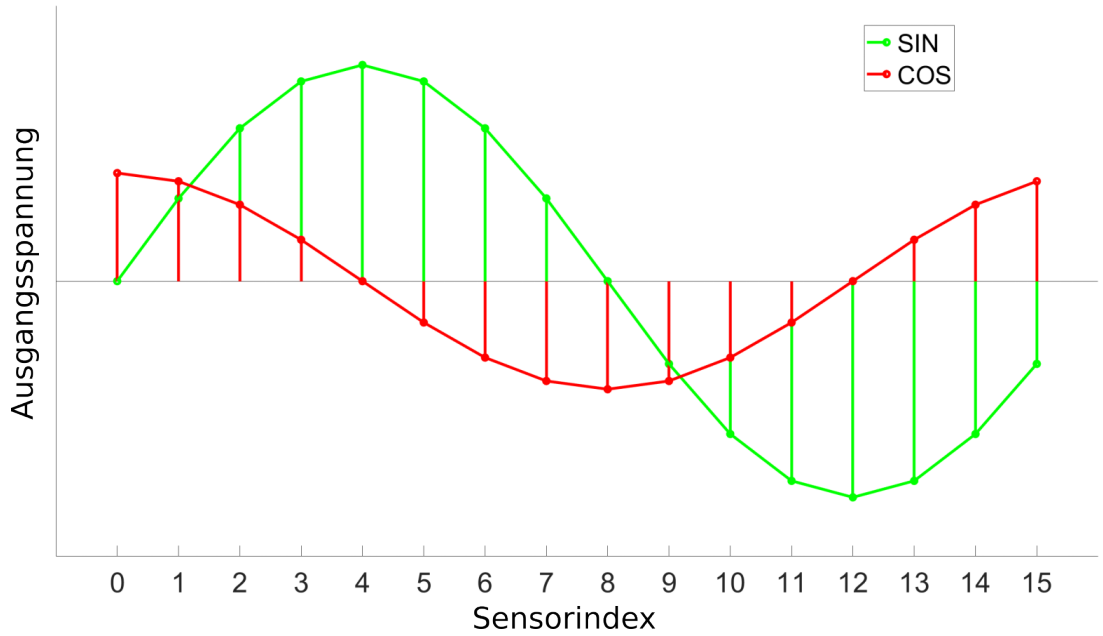

Abbildung 1.10: Darstellung der Ausgangsspannungen der Sensoren aus Abbildung 1.9

Nach einer idealen Abtastung ergeben sich folgende diskrete Abtastwerte (Formel 1.10):

$$x[n] = U_{Max} \cdot \sum_{n=0}^{15} \cos(\frac{2\pi nf}{16} + \phi) + i \cdot sin(\frac{2\pi nf}{16} + \phi) \tag{1.10}$$

In Schreibweise mit komplexer e-Funktion ergibt sich also folgende Summe:

$$x[n] = U_{Max} \cdot \sum_{n=0}^{N-1} e^{+*i\cdot(\frac{2\pi nf}{N}+\phi)} \tag{1.11}$$

Es ergibt sich nach einer diskreten Fouriertransformation von Formel 1.11 folgender Ausdruck [5]:

$$X(k) = \mathcal{F}x[n] = U_{Max\_ADC} \cdot e^{-i\phi} N\delta[k-f] \tag{1.12}$$

Für f = 1 und k = 1 ergibt sich folgender Winkel θ:

$$\theta = \arctan\left(\frac{imag(X(1))}{real(X(1))}\right) = \arctan\left(\frac{sin(\phi)}{cos(\phi)}\right) = \phi \tag{1.13}$$

θ entspricht dem Drehwinkel unseres Magneten ϕ, welcher Für den realen Fall, der in Abbildung 1.9 gezeigt ist, ergeben sich folgende Abtastwerte.

$$x[n] = U_{Max} \cdot \sum_{n=0}^{N-1} \frac{1}{2} \cos(\frac{2\pi nf}{N} + \phi) + i \cdot sin(\frac{2\pi nf}{N} + \phi) \tag{1.14}$$

Wird diese Folge an Abtastwerten der diskreten Fouriertransformation unterzogen ergeben sich folgende Werte im Frequenzbereich:

$$\begin{aligned} &X(k) = \mathcal{F}x[n] \\ &= U_{Max\_ADC} \cdot e^{-i\phi} \left(\frac{N}{4}\left(\delta[k-f] + \delta[k+f-N]\right) + \frac{N}{2}\left(\delta[k-f] - \delta[k+f-N]\right)\right) \end{aligned} \tag{1.15}$$

Wird nun angenommen, dass f = 1, $N \neq 0$ und das Spektrum an der Frequenz k = 1 betrachtet wird entfallen die Terme mit $\delta[k+f-N]$. Diese Annahmen kann getroffen werden, da die durch einen magnetischen Dipol zu erwartenden Signale aus einer komplexen Sinusschwingung und einer reellen Kosinusschwingung mit Frequenz 1 bestehen

(Siehe Abb.1.10). Es bleibt folgender Term:

$$X(k) = \mathcal{F}x[n] = U_{Max\_ADC} \cdot e^{i\phi}\delta[k-f]\left(\frac{N}{4} + \frac{N}{2}\right) \tag{1.16}$$

$$X(k) = \mathcal{F}x[n] = U_{Max\_ADC} \cdot \frac{3}{4}N \cdot e^{i\phi}\delta[k-f] \tag{1.17}$$

Im Vergleich zu Formel 1.12 ändert sich lediglich der Betrag der Funktion an der diskreten Frequenz k=1 bei Annahme von f=1. Der über die Formel 1.13 bestimme Winkel bleibt jedoch gleich. In der Theorie ist es also möglich, den Drehwinkel des Magneten in der X/Y-Ebene durch eine komplexe Fouriertransformation der Signale eines runden Sensorarrays zu bestimmen, obwohl die Ausbreitung des Feldes sich nicht in alle Richtungen gleich verhält.

2 Hardware-Entwurf des Sensor-Arrays

In diesem Kapitel wird die Entwicklung der Hardwareplatine näher behandelt. Es werden Informationen über die Anforderungen an die Form und Funktion erklärt, die Schaltungsentwicklung und die Auslegung der Bauteile werden beschrieben. Abschließend wird auf die Bestückung und die Kompatibilität der Platine mit dem Mikrocontroller des Messaufbaus eingegangen.

2.1 Entwicklung der Sensorplatine

Um die nötigen Bauteile auf der Platine unterzubringen, ist sie kreisförmig mit einem Durchmesser von 63.3 mm. Außerdem befinden sich in einem Abstand von 90° 6 mm lange, abgerundete Nasen mit einer Öffnung zum Befestigen mit Schrauben der Größe M3. auf einer Seite der Platine ist eine 9 mm breite und 32 mm lange rechteckige Erweiterung für die Integration des Pin-Headers der Platine. In der Mitte der Sensorplatine befindet sich eine runde Öffnung mit 25 mm Durchmesser um die Platine auf die Nuss des Schraubwerkzeugs zu setzen, welche für Testzwecke für das metrische ISO-Regel-Gewinde M8 konzipiert ist. Die Platine beinhaltet in ihrer Grundform ein Sensorarray aus 16 kreisförmig angeordneten TMR-Magnetfeldsensoren im Thin Film Small Outline Package (TSSOP)-Gehäuse mit einem Radius von 35 mm. Die Anzahl an Sensoren kann in der Theorie beliebig erweitert werden, um eine höhere Auflösung zu erreichen und ist auf 16 begrenzt worden, um den Aufwand für die Entwicklung der Platine nicht unnötig zu erhöhen. Die Unterbringung von weiteren Sensoren und Höhere Formfaktoren können zum Beispiel durch mehrlagige Platinen, kleinere Packages, oder effizientere Anordnung der Bauteile erreicht werden.

Um die verschiedenen Sensoren auf der Platine über einen bzw. zwei analoge differentielle Busse auszulesen, müssen sie im Multiplexverfahren auf die Busse geschaltet werden. In den bisherigen Realisierungen von magnetischen Sensorarrays im Rahmen des Projektes

ISAR werden die Ausgänge der Sensoren hierfür über Integrated Circuit (IC)s mit analogen Schaltern auf die Busse geschaltet [4]. Diese werden von einem zentralen Multiplexer (MUX) geschaltet, der binär über Auswahlleitungen mit General Purpose Input-Output (GPIO) Pins eines Mikrocontrollers angesteuert wird.
Da für die Anwendung in einem Schraubwerkzeug ein hoher Formfaktor von Interesse ist, wurde zusätzlich zu dieser Funktionsweise für diese Arbeit eine zweite Platine entwickelt, bei welcher die Schalter an den Ausgängen der Sensoren durch Dioden ersetzt werden. Der zentrale Multiplexer schaltet nun statt den Schaltern an den Ausgängen der Sensoren die Stromversorgungen der Sensoren. Durch die Dioden kann verhindert werden, dass die Messbrücken der Sensoren über die Busleitungen parallel auf Ground geschaltet werden, was das Messergebnis verfälscht.

2.2 Schaltungsprinzip des Arrays

2.2.1 Multiplexvariante

In dieser Variante der Platine werden die Ausgänge der Sensoren über Schalter in ICs auf den jeweiligen Bus geschaltet. Hierfür wird für jeden Ausgang der Sensoren ein Schalter benötigt. Bei zwei differentiellen Ausgängen werden also vier Schalter pro Sensor benötigt. Da es nicht möglich war, ICs mit vier gekoppelten Schaltern zu finden, welche über eine einzige Enable-Leitung geschaltet werden, wurden ICs mit vier Schaltern und individuellen Enable-Eingängen gewählt und anschließend im Platinenlayout alle Enable-Eingänge mit einer Busleitung verbunden. Diese Busleitungen werden von einem analogen 16-Kanal-Multiplexer geschaltet.
Da der digitale Multiplexer seine geschalteten Ausgänge lediglich mit einem Common-Input verbindet, ist das Potential an den nicht geschalteten Ausgängen undefiniert. Um zu vermeiden, dass bei der Steuerung der Schalter der Sensorausgänge unerwartete Fehler auftreten ist deshalb an jede Enable-Leitung ein Weak-Pulldown Widerstand von 10 kΩ auf Masse (GND) geschaltet.
Bei den Schaltern der Sensorausgänge handelt es sich um den IC CD74HCT4066 der Firma Texas Instruments. Für den zentralen Multiplexer wird der IC CD74HCT4067 des selben Herstellers verwendet. Nur mit Multiplexerchips des HCT- Typs ist es möglich die Platine mit 5V und zu gleich die Selectleitungen mit $< 5V$ (Low-Voltage Transistor-Transistor Logic (LVTTL) 3.3V) zu betreiben. Werden die Chips des HC-Typs genutzt, welche sich durch eine höhere Störfestigkeit auszeichnen, muss die an den Selectleitungen

angelegte Spannung mindestens 0.7· Betriebsspannung (VCC) betragen[10][11].
Zur Unterdrückung von hochfrequenten Störsignalen auf der Betriebsspannung der Sensoren ist in der Nähe des Betriebsspannungspins jedes Sensors ein Bypass-Kondensator von 100 nF vorgesehen. Um die Betriebsspannung der gesamten Platine zu glätten, ist ebenfalls in der Nähe des Betriebsspannungspins der Platine ein Glättungskondensator von 10 uF vorgesehen.

In den Abbildungen 2.1 ist die Schaltung der Multiplexerplatine in einem Schaltungsdiagramm dargestellt. Das Schaltungsprinzip ist dabei zur Vereinfachung für zwei Sensoren ohne Pull-Up-Widerständen an den Enableleitungen und ohne Betriebsspannungsversorgung dargestellt.

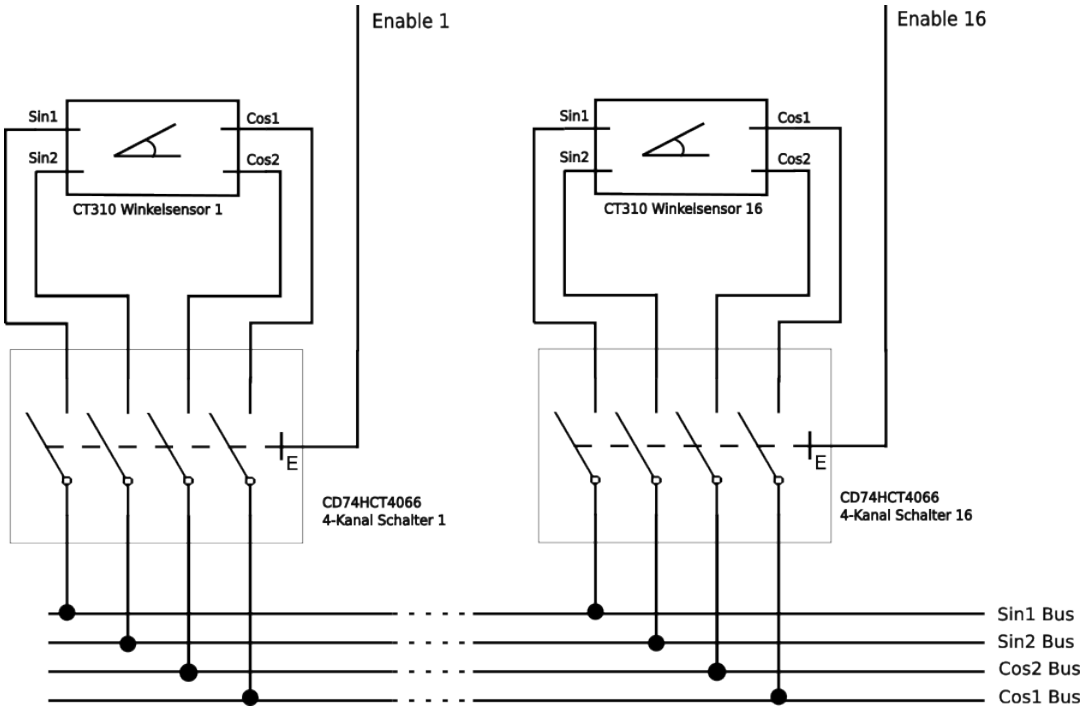

Abbildung 2.1: Schaltbild der Funktionsweise der Multiplexerplatine mit 4 Kanal Schalter und gemultiplexter Enableleitungen

Das Layout der Multiplexerplatine kann in 16, sich für jeden Sensor wiederholende, gleich aufgebaute Segmente unterteilt werden. Diese Segmente bestehen aus dem Sensor, dem Sensorschalter sowie einem Pull-Up-Widerstand für die Enable-Leitung des Sensorschalters und einem Bypasskondensator vor den Betriebsspannungspins des Sensors. Eines dieser Segmente ist in Detailansicht in Abbildung 2.5 gezeigt. Die einzelnen Teile dieser Segmente werden axial vom Mittelpunkt wegführend über Leiterbahnen miteinander

verbunden. Diese axialen Verbindungen werden auf der Oberseite der Platine gelegt (Abbildung A.1 links). Die radialen Verbindungen für die Busleitungen der Sensorausgänge sowie die Enableleitungen und der Betriebsspannung werden auf der Unterseite der Platine gelegt (Abbildung A.1 rechts).

2.2.2 Diodenvariante

In der Diodenvariante der Platine werden die Ausgänge der Sensoren über Dioden auf die Busse geschaltet. Der zentrale Multiplexer schaltet nun die Betriebsspannungen der Sensoren und nicht mehr die Schalter-ICs. Dies ist möglich, da an den Ausgängen der Sensoren nie ein negatives Signal abfällt und die Diode damit im Betrieb nur in Flussrichtung geschaltet wird.
Bei den Messbrücken der nicht geschalteten Sensoren wird der Ausgang über die Brücke auf GND gezogen. An den Dioden dieser Sensoren liegt also eine Spannung entgegen der Flussrichtung an. Sie leiten also nicht und der Brückenwiderstand des geschalteten Sensors wird nicht beeinflusst.
Für den zentralen Multiplexer der Diodenplatine wird wie auch in der Multiplexerplatine ein Texas Instruments CD74HCT4067 verwendet. Als Dioden an den Ausgängen der Sensoren werden 1N5817WS in SMD-Bauweise des Herstellers Diodes Incorporated genutzt.
Auch auf dieser Platine sind an den Betriebsspannungen der Sensoren 100 nF und an der Betriebsspannung der Platine 10 uF Kondensatoren zur Glättung der Betriebsspannung vorgesehen

Eine vereinfachtes Schaltungsprinzip der Diodenplatine ist in ähnlicher Weise wie bei der Multiplexerplatine in der Abbildung 2.2 gezeigt.

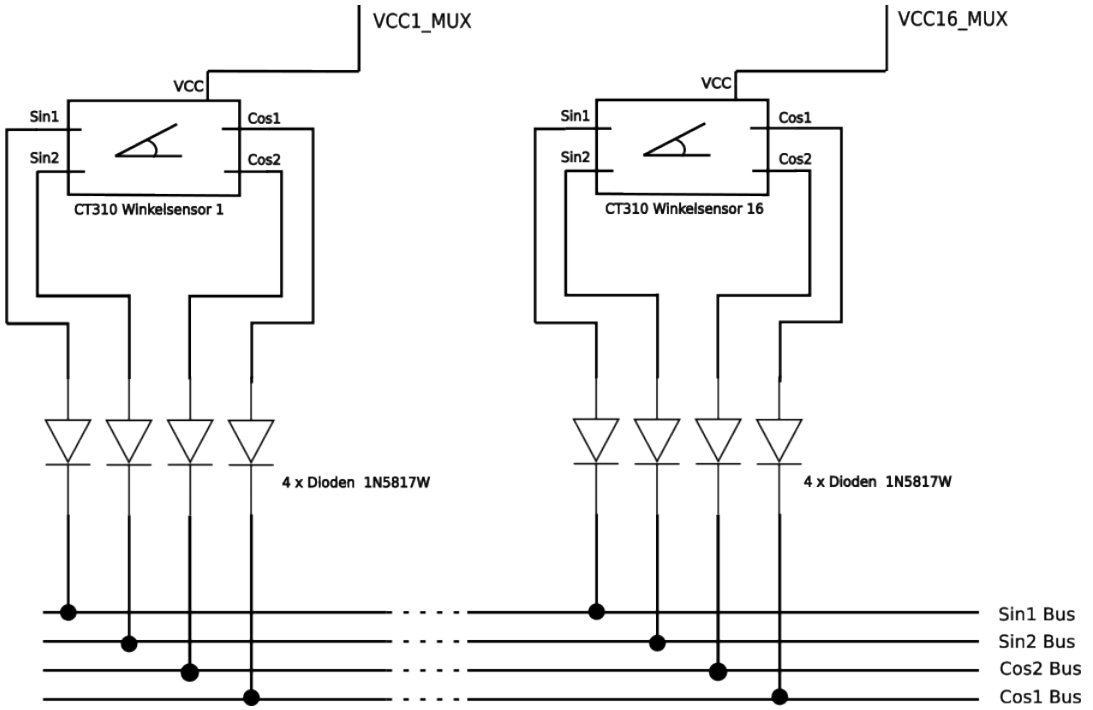

Abbildung 2.2: Schaltbild der Funktionsweise der Diodenplatine mit gemultiplexter Stromversorgung der Sensoren

2.2.3 Auswahl und Auslegung der Bauteile

Als Magnetfeldsensoren wurden TMR-Sensoren des Typs CT310 des Herstellers Crocus Technology ausgewählt. Wie schon genauer in Kapitel 1.2.2, beschrieben, zeichnen sich TMR-Sensoren durch ihre hohe Empfindlichkeit und dadurch hohe Ausgangsspannungsdifferenz aus. Die Sensoren können somit ohne Verstärkerschaltung betrieben und direkt auf einen Analog Digital Converter (ADC) geschaltet werden.
Für den zentralen Multiplexer wird ein analoger 16-Kanal-Multiplexer des Typs CD74HCT4057 gewählt. Im Fall der Diodenplatine kann der Chip problemlos betrieben werden, da er mit der Betriebsspannung der Sensoren ein analoges Signal schaltet. Für den Betrieb in der Multiplexerschaltung müssen Pullupwiderstände an den digitalen Enable-Eingängen der Sensoren vorgesehen werden. Diese Widerstände sind in SMD Bauform in der Größe 0602 vorgesehen.
Auch die Bypasskondensatoren an den Sensoren sind in der Größe 0602 vorgesehen. Der Glättungskondensator an der Betriebsspannung der Platine ist auf beiden Platinen in der Größe 1206 vorgesehen.
Als Schalter für die Sensorausgänge werden ebenfalls analoge Schalter des Typs CD74HCT4066 genutzt. Diese beinhalten 4 analoge parallele Halbleiterschalter in einem TSSOP14-Package.
Wie bereits zuvor erwähnt kann die Platine sowohl mit Multiplexerbauteilen für Metall-Oxid-Halbleiter-Feldeffekttransistor (MOSFET)-Pegel (HC-Reihe) als auch LVTTL (HCT-Reihe) betrieben werden. Bei einem Betrieb mit Bauteilen der HC-Reihe muss auf einen High-Pegel der Selectleitungen von mindestens $0,7 \cdot VCC$ geachtet werden.

2.3 Entwicklung der Schaltung und des Platinenlayouts

An die Platine werden folgende grundlegende Anforderungen gestellt. Sie darf einen Innendurchmesser von 25 mm nicht unterschreiten und muss ein kreisförmiges Array von magnetischen Winkelsensoren beinhalten. Die Platine muss außerdem am Schaubwerkzeug zu befestigen sein.
In Abbildung 2.4 ist die Platine in ihrer einfachsten Form dargestellt. In einem kreisrunden Abstand von 18.75 mm befindet sich ein Array von 16 Winkelsensoren. Auf ihrer rechten Seite ist eine rechteckige Erweiterung vorgesehen, um einen 2.54 mm Pinheader für 12 Pins in 2 Reihen mit je 6 Pins zu integrieren. In einem Distanz von 35 mm vom Mittelpunkt der Platine sind mit 90°Versatz Schraublöcher für M3-Schrauben an runden

Ausbuchtungen vorgesehen.
Um die Platine auszulesen wird ein Mikrocontroller benötigt, welcher folgende Bedingungen erfüllt:
1. 5 x GPIO als Selectleitungen für MUX und Enable
2. 2 x Differentielle ADC
3. 2 x Universal Asynchronous Receiver / Transmitter (UART)-Schnittstelle (bzw. UART over Universal Serial Bus (USB)) für Winkelgeber und Kommunikation mit dem PC

Diesen Anforderungen entpricht das Mikrocontroller Evaluation-Board EK-TM4C1294XL der Firma Texas Instruments. Dieses wird bereits in vielen Bereichen der Hochschule genutzt. Dies vereinfacht es schnell und ohne viel Aufwand externe Hilfe für die Arbeit mit dem Mikrocontroller zu erhalten.

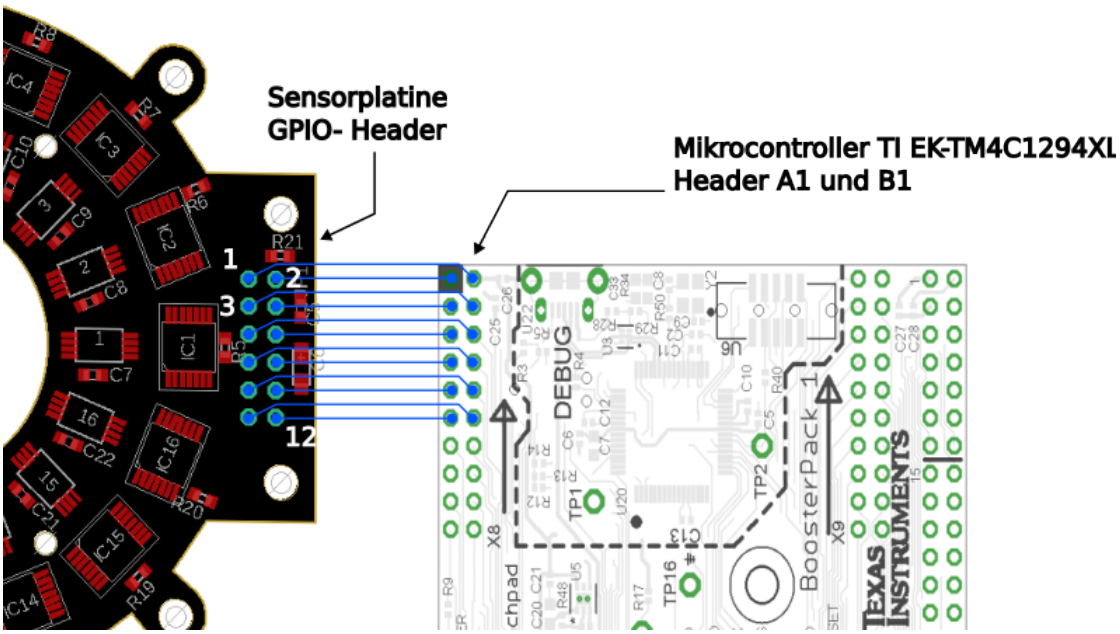

Abbildung 2.3: Belegung der GPIO Pins der Sensorplatine. Exakte Bezeichnungen der GPIO Pins in Tabelle 2.1

Pin-Nmr (PCB)	Pin-Name (PCB)	GPIO (EK)	Funktion
1	+5V	+5V	VCC (+5V / +3.3V)
2	n.C	n.C	n.C
3	GND	GND	Referenzpotential
4	S0	PE4	Multiplexer Select 0
5	COS1_BUS	PE0	Cos Differential Output 1 (Cos+)
6	S1	PC4	Multiplexer Select 1
7	COS2_BUS	PE1	Cos Differential Output 2 (Cos-)
8	S2	PC5	Multiplexer Select 2
9	SIN1_BUS (Platine 2022), aktueller Stand in Eagle: SIN2_BUS	PE2	Sin Differential Output 1 (Sin+) (Platine 2022), aktueller Stand in Eagle: Sin Differential Output 2 (Sin-)
10	S3	PC6	Multiplexer Select 2
11	SIN2_BUS (Platine 2022), aktueller Stand in Eagle: SIN1_BUS	PE3	Sin Differential Output 2 (Sin-) (Platine 2022), aktueller Stand in Eagle: Sin Differential Output 1 (Sin+)
12	E	PE6	Multiplexer Enable (low-activ)

Tabelle 2.1: Pinbelegung der Platine mit entsprechendem GPIO auf dem Mikrocontroller Evaluation-Kit EK-TM4C1294XL. Aufgrund eines Fehler im Routings der Platine waren die Signale SIN1_BUS_TP und SIN2_BUS_TP für die erste Lieferung an Platinen vertauscht. Dies ist in den EAGLE-Files verbessert worden. Stand: 13.06.2023 [12]

2.4 Umsetzung, Fertigung und Bestückung

Die Platinen werden bei einem Leiterplattenhersteller bestellt und anschließend im Labor per Hand bestückt. Bei der Bestückung ist auf eine exakte Platzierung der Sensoren zu achten, da jede Verkippung oder Schrägstellung zu einem Winkelfehler in der Messung beitragen kann. Um diesen Fehler zu verringern, sollten bei zukünftigen Bestellungen der Platinen Stencils für eine Bestückung im Reflow-Ofen bestellt werden oder eine maschinelle Bestückung beim Hersteller in Betracht gezogen werden. Dadurch können die Bauteile sehr präzise platziert werden.
Bei der händischen Bestückung der Platine wird der zentrale Multiplexer zuerst eingelötet und seine Funktionalität überprüft. Nachdem diese bestätigt werden kann, wird die

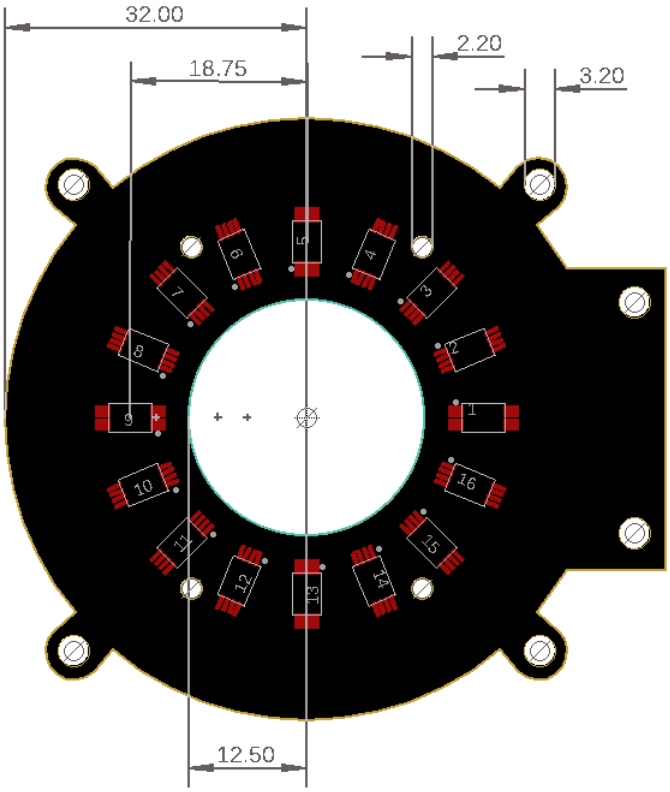

Abbildung 2.4: Abmessungen der grundlegenden Platine in mm mit TMR-Sensoren

erste Sensorschaltung, bestehend aus Sensor und Schalter bzw. Dioden eingelötet und anschließend deren Funktionalität überprüft.
Nach und nach werden die weiteren Sensor-ICs bestückt und überprüft, bis die Platine schließlich fertig bestückt ist.
Mit der Bestückung der Bypass-Kondensatoren der Sensorbetriebsspannung wird gewartet, bis eine Überprüfung der Platine mit einem Oszilloskop erfolgt.

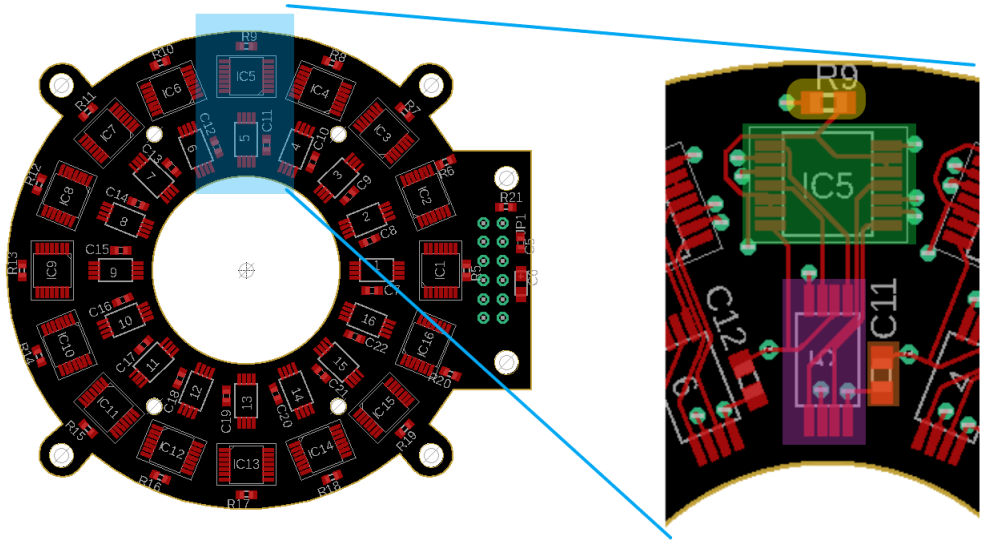

Abbildung 2.5: Aufbau der Multiplexerplatine. Detaildarstellung einer Sensorschaltung. Gelb: Pull-Up-Widerstand für Enable, Grün: Sensorschalter, Lila: Magnetfeldsensor, Orange: Bypasskondensator

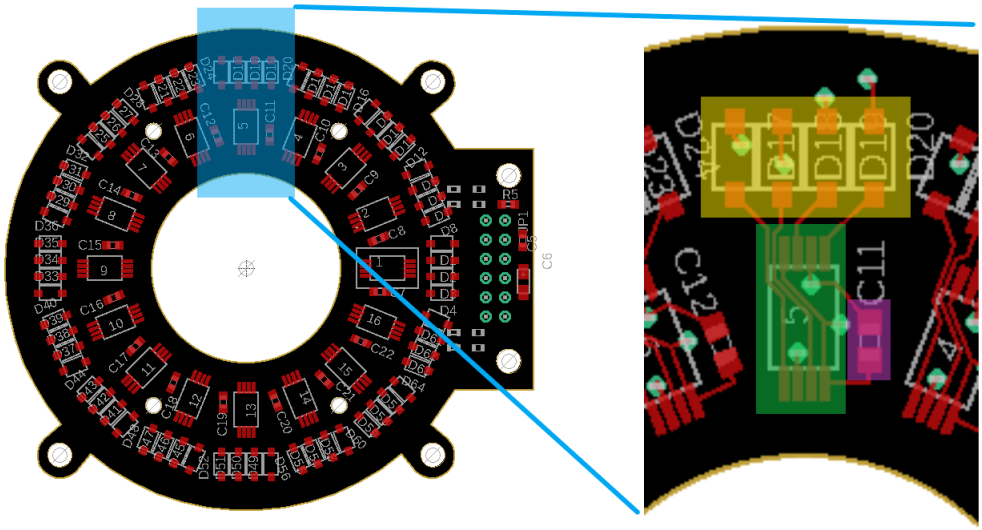

Abbildung 2.6: Aufbau der Diodenplatine. Detaildarstellung einer Sensorschaltung. Gelb: Dioden, Grün: Magnetfeldsensor, Lila: Bypasskondensator

3 Software-Entwicklung

In diesem Kapitel wird die Entwicklung der zum Auslesen der Platine nötigen Software beschrieben und die Inbetriebnahme sowie die Beobachtung des Schaltverhalten beschrieben.
Im Beginn wird ein Testplan über die Vorgehensweise zur Überprüfung der Funktion der Platine erläutert. Im Anschluss wird die Entwicklung der Software beschrieben. Zum Schluss dieses Kapitels, werden die Ergebnisse der Erprobung dargestellt und erklärt.

3.1 Planung der Inbetriebnahme

Um die Funktionalität der Platine zu überprüfen, wird eine Platine mit den für das Auslesen eines Sensors nötigen Bauteilen bestückt (zentraler Mux, Strombegrenzungswiderstand, Pinheader). Außerdem wird die Schaltung des ersten Sensorsegments bestückt (Sensor, Sensorschalter, Pull-Up-Widerstand). Die übrigen Sensorsegmente bleiben unbestückt, um Fehlerquellen zu reduzieren und die Grundfunktion der Platine - das Auslesen eines Sensors - zu überprüfen.
Der gesamte Prozess der Inbetriebnahme ist in 3 Testfälle aufgeteilt. Diese Testfälle werden in der Tabelle 3.1. In dieser Tabelle werden die Testfälle beschrieben, ein erwartetes zufriedenstellendes Ergebnis erklärt und das tatsächlich Beobachtete Ergebnis verzeichnet.

3.2 Aufbau und Test

Die Analog-Digital-Wandlung und das Abspeichern der Sensorwerte in einem Array wird mit einem Mikrocontroller vom Typ TM4C1294ncpt des Herstellers Texas Instruments realisiert. Der Mikrocontroller wird auf einem Evaluation-Board des Typs EK-TM4C1294XL genutzt.

Im Messaufbau werden die Referenzwerte für die Bestimmung des absoluten Winkels einer Motorwelle durch einen Winkelgeber des Typen AMT212B der Firma CUI-Devices generiert. Dieser besitzt eine RS-485 Schnittstelle und wird über eine Wandlerplatine von UART-TTL auf RS485 durch den gleichen Mikrocontroller über UART ausgelesen.
Die Übertragung der Daten an den PC zur Abspeicherung in einer Comma Seperated Value (CSV)-Tabelle und zur weiteren Verarbeitung findet über den USB-Debug-Port durch eine emulierte UART Verbindung statt.

Testbezeichnung	Beschreibung	Erwartung	Ergebnis
Test einer Sensorschaltung	Ein Sensor und die zur Auswertung notwendige Schaltung (Dioden,Multiplexer) werden bestückt. Die Schaltung wird elektrisch Überprüft und Betriebsspannung über eine Strombegrenzung angelegt und langsam erhöht. anschließend wird sie über den Mikrocontroller ausgelesen.	Keine Kurzschlüsse oder Unterbrechungen. Bei Auswertung variierende Ausgangswerte bei verschiedenen Magnetstellungen	erfolgreich
Test der voll bestückten Platine	Die Platine wird voll bestückt und elektrisch überprüft. Die Betriebsspannung wird über eine Strombegrenzung angelegt und langsam erhöht. Nach Anliegen der vollen Betriebsspannung wird die Platine über den Mikrocontroller ausgelesen	Keine Kurzschlüsse oder Unterbrechungen. Bei Auswertung erkennbares Sinus und Kosinussignal wie in Abbildung 1.10	erfolgreich
Test des Schaltverhaltens	Eine Sensorschaltung wird auf den Bus geschaltet und während des Schaltvorgangs werden die wichtigsten Spannungen und ihr Zeitverhalten auf dem Oszilloskop betrachtet. Diodenplatine: VCC_{Sensor}, Sensorausgang, Zeitbereich der Abtastung - Muxplatine: $Enable_{SensorMux}$, Sensorausgang, Zeitbereich der Abtastung	Erkenntnisse über den Schaltvorgang und das Einschwingverhalten der Sensorausgänge. Abschätzen des Abtastbereiches	Diodenplatine: Einschwingzeit ca. $< 2\mu$ sec, Muxplatine: Einschwingzeit ca. $< 500n$ sec

Tabelle 3.1: Testplan zur Inbetriebnahme der Platine

3.3 Entwicklung von Software

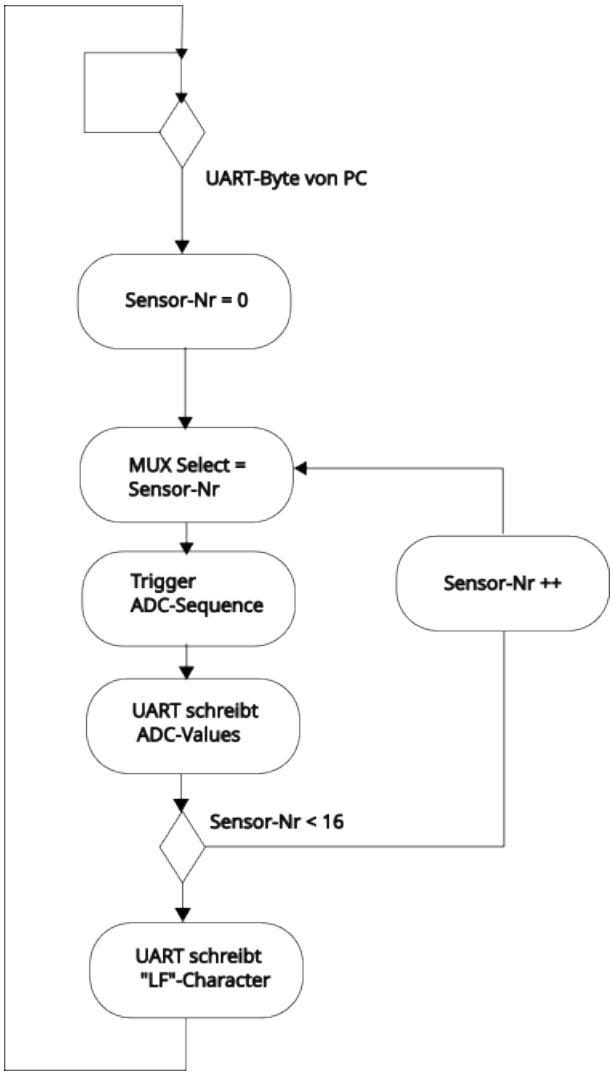

Abbildung 3.1: Vereinfachtes Ablaufdiagramm der Mikrocontroller-Routine zur Aufnahme und Übertragung der Sensorwerte.

Die Routine des Mikrocontrollers ist ausschließlich mit Blocking-Code geschrieben, da das System keine Anforderungen an Parallelität hat. Die Routine wird also in der Main-Schleife ausgeführt, ohne in eine Interrupt-Routine zu springen.
Um zu verhindern, dass die Routine bei einem Übertragungsfehler komplett blockiert oder falsche Daten überträgt, sind einige Sicherheitsmechanismen in den Code eingebaut. Ein Software-Watchdogtimer überwacht die Routine und löst im Falle eines länge-

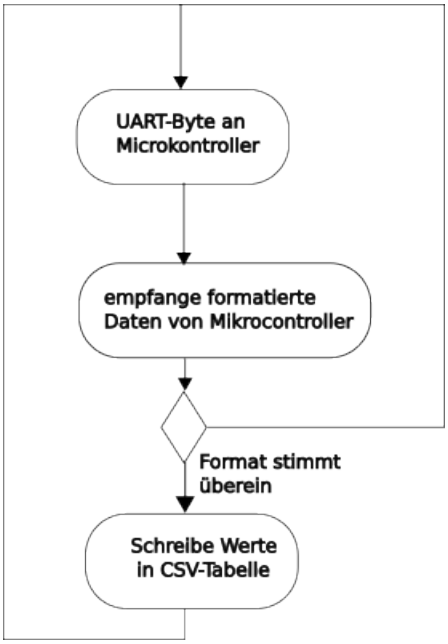

Abbildung 3.2: Vereinfachtes Ablaufdiagramm der Matlab-Routine am Laborrechner zur Aufnahme und Abspeicherung der Sensorwerte. Bei Mittelung finden mehrere Übertragung und eine Mittelwertbildung statt, bevor die Werte abgespeichert werden.

ren Timeouts einen Software-Reset aus. Außerdem werden die zu übertragenden Daten formatiert und mit einem Line-Feed-Character abgeschlossen. Am PC wird die Anzahl der übertragenen Bits überprüft und bei fehlender Übereinstimmung mit dem Format wird eine erneute Datenübertragung angefordert. Die Routine des Mikrocontrollers ist vereinfacht in Abbildung 3.1 dargestellt. Der Ablauf des Datenempfangs und der Abspeicherung in Matlab ist vereinfacht in Abbildung 3.2 gezeigt.

3.4 Darstellung der Ergebnisse

Nachdem die Platine auf Fehler überprüft worden ist und Kurzschlüsse oder Unterbrechungen ausgeschlossen werden konnten, wird das Schaltverhalten der Platine beobachtet. Hierbei müssen die Diodenplatine und die Multiplexerplatine getrennt von einander betrachtet werden. Bei der Multiplexerplatine werden die Sensorausgänge über analoge Multiplexer, welche als Schalter genutzt werden, auf die Signalbusse geschaltet. In diesem Fall ist es von Interesse, das Schaltverhalten von folgenden Signalen zu beobachten.

1. Die Spannung am Eingang des Sensormultiplexer-/schalters. Erst wenn diese Spannung einen Schwellwert von etwa $> 2V$ (HCT-Reihe der Multiplexer) überschreitet kann angenommen werden, dass das Signal auf den Signalbus geschaltet wird.
2. Die Spannung am Sensorausgang. Durch das Umschalten von Signalen muss mit Ladevorgängen gerechnet werden, welche zu Einschwingvorgängen führen. Die Effekte dieser Ladevorgänge auf der Multiplexerplatine sollten vergleichsweise gering ausfallen, da keine hohen Spannungs- oder Stromdifferenzen zwischen den geschalteten Signalen bestehen.
3. Ein Signal zur Abschätzung des Abtastbereiches der ADCs des Mikrocontrollers. Durch die Kenntnis über diesen Zeitbereich kann die Verzögerungszeit zwischen dem Schalten der Sensoren und dem Abtasten der ADCs geschätzt werden. Dieses Signal wird durch das Umschalten eines GPIO des Mikrocontrollers erzeugt.
Bei der Multiplexerplatine kann, statt des Schaltsignals am Eingang des Sensormultiplexers, die Betriebsspannung des Sensors betrachtet werden. Die Betriebsspannung ist in diesem Fall das Signal, welches zu einer Beschaltung des Sensors mit dem Signalbus führt.
Die Ergebnisse dieser Tests sind in den Abbildungen 3.3 bis 3.6 gezeigt. Für die Multiplexerplatine ist nach Umschalten der Sensorausgänge ein kurzer Einschwingvorgang erkennbar (Abbildung 3.3). Nach etwa 500 ns kann das Signal als eingeschwungen angenommen werden und eine Abtastung erfolgen. In Abbildung 3.4 ist eine Abtastung ohne Verzögerung zwischen dem Setzen der Selectsignale des Multiplexers und dem Triggern der ADC-Sequenz gezeigt. Diese führt zu einer Verzögerung von etwa 900 ns. Im Fall der Multiplexerplatine wäre eine Abtastung ohne Verzögerung theoretisch möglich. Da zu diesem Zeitpunkt in der Betrachtung der Schaltung keine Zeitanforderungen an das Auslesen der Platine gestellt werden, wird jedoch trotzdem eine Verzögerung von 200 μs vor der Abtastung in Software implementiert.
Im Fall der Diodenplatine ist das Zeitverhalten für eine Abtastung ohne Zeitverzögerung in Abbildung 3.6 gezeigt. Durch das Schalten der Betriebsspannungen der Sensoren müssen sich bei jedem Schaltvorgang die Kapazitäten in den Brücken der Sensoren, sowie der Dioden an den Ausgängen laden. In diesem Fall beträgt der Einschwingvorgang deshalb ca 2 μs. Eine Abtastung ist nun nicht mehr ohne Verzögerung möglich. Als Verzögerungszeit wird deshalb ebenfalls 200 μs in Software implementiert.

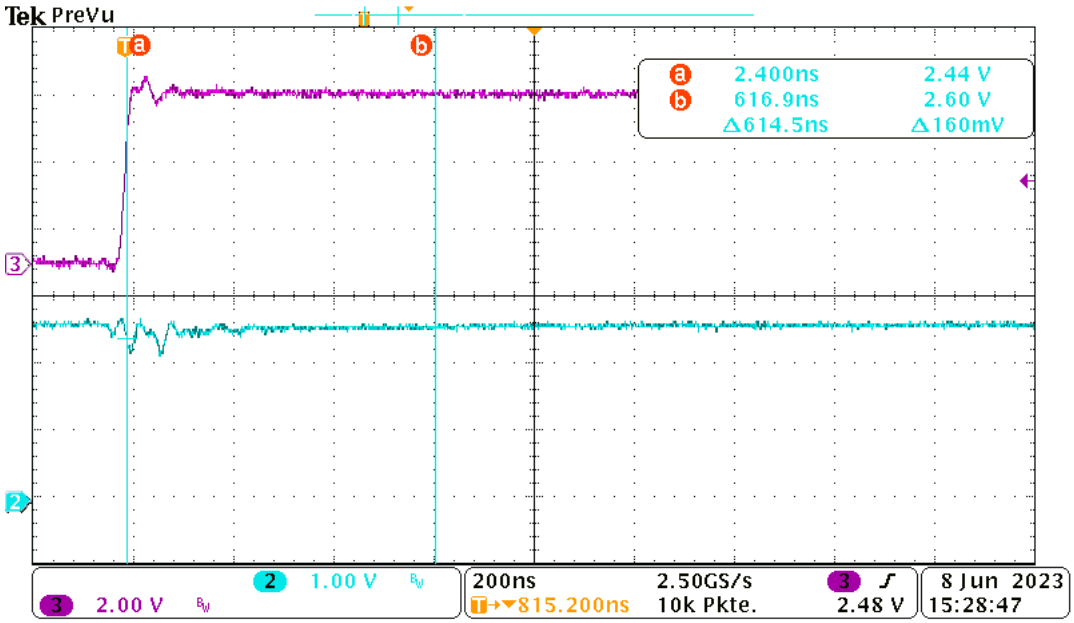

Abbildung 3.3: Beispielhafter Schaltvorgang einer Multiplexerschaltung. Oben(Magenta): Geschaltetes Selectsignal eines Sensor-Multiplexers. Unten(Cyan): Sin1-Ausgang des Sensors

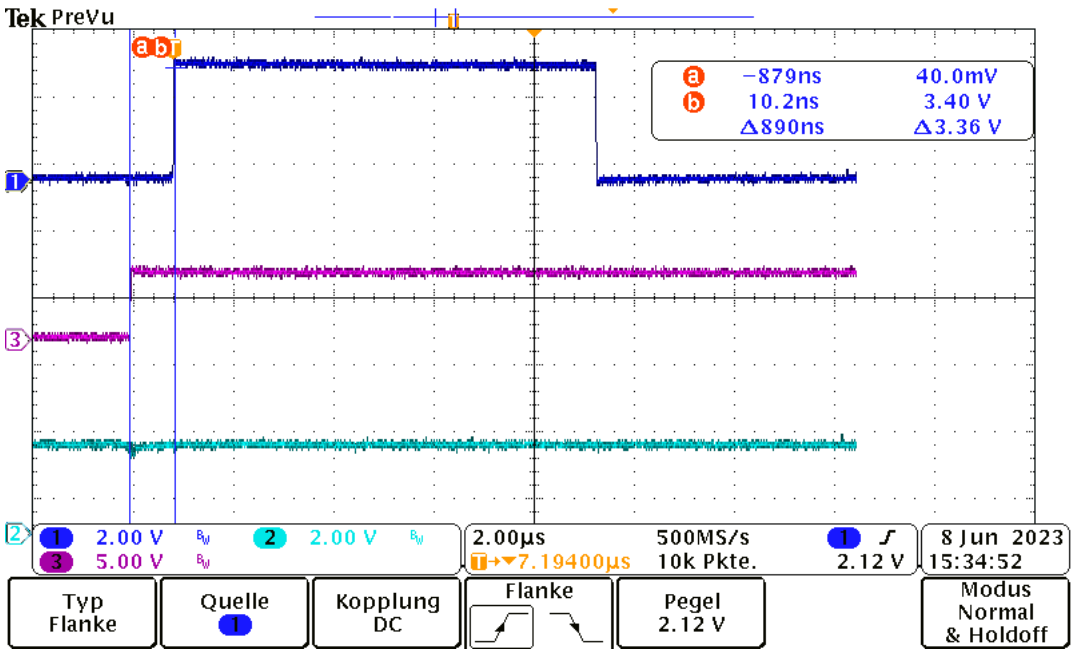

Abbildung 3.4: Beispielhafter Schaltvorgang einer Multiplexerschaltung. Oben(Blau): Zeitraum der ADC-Umwandlung (High-Pegel). Mitte(Magenta): Geschaltetes Selectsignal eines Sensor-Multiplexers. Unten(Cyan): Sin1-Ausgang des Sensors

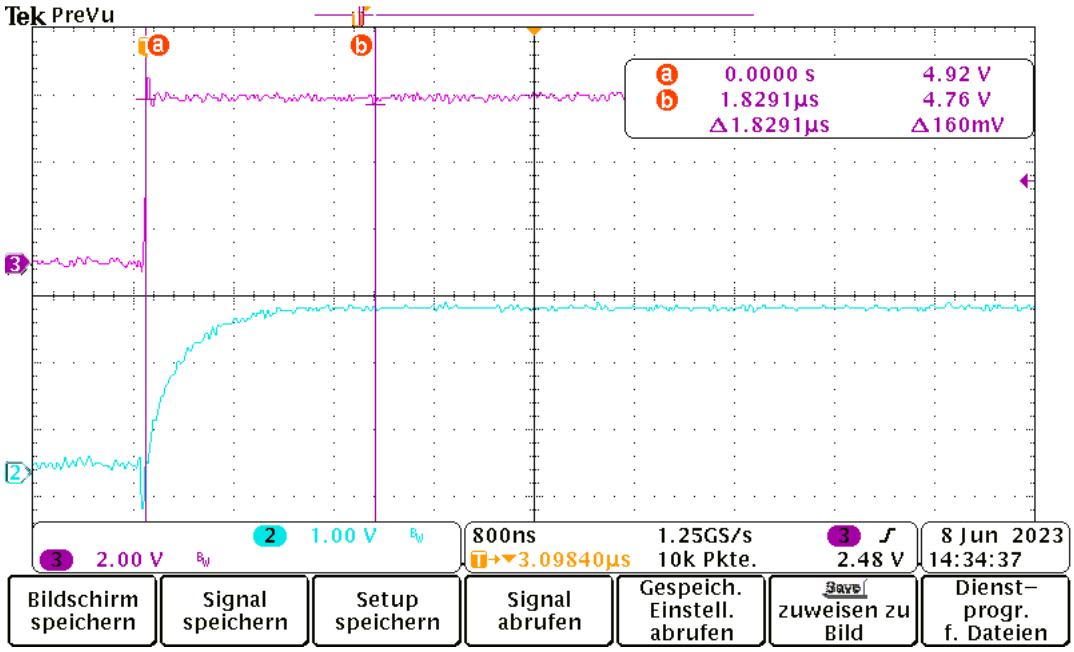

Abbildung 3.5: Beispielhafter Schaltvorgang einer Diodenschaltung. Oben(Magenta): Geschaltete Versorgungsspannung des Sensors. Unten(Cyan): Sin1-Ausgang des Sensors

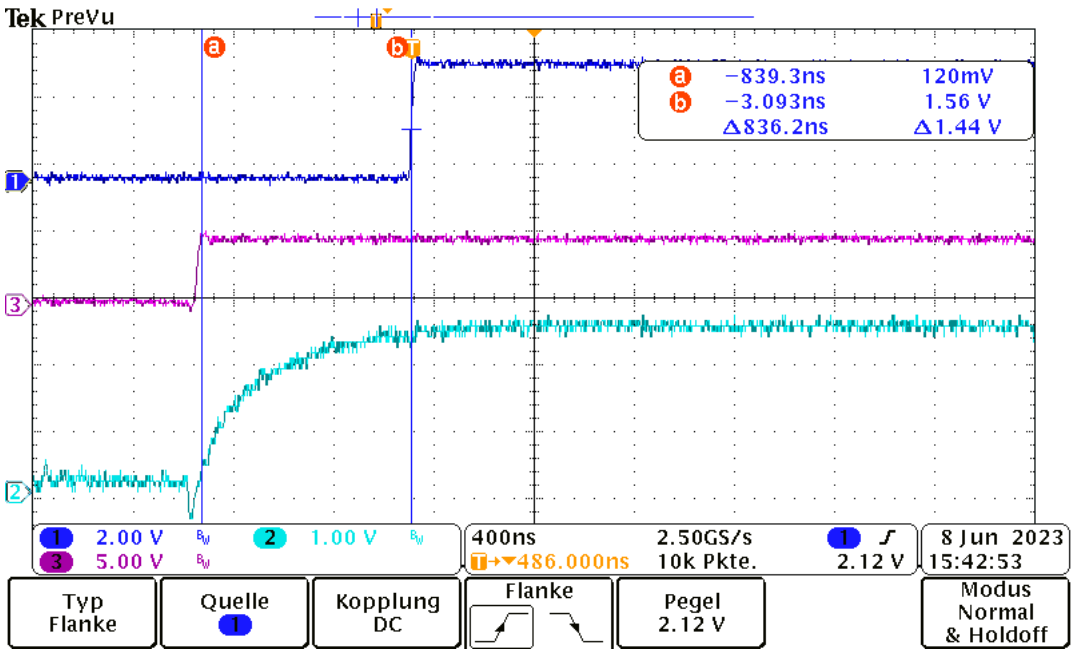

Abbildung 3.6: Beispielhafter Schaltvorgang einer Diodenschaltung. Oben(Blau): Zeitraum der ADC-Umwandlung (High-Pegel). Mitte(Magenta): Geschaltete Versorgungsspannung des Sensors. Unten(Cyan): Sin1-Ausgang des Sensors

4 Entwicklung des Messaufbaus

In diesem Kapitel wird die Entwicklung des Messaufbaus zur Auswertung der Platine beschrieben. Zu Beginn werden die Anforderungen an den Messaufbau erklärt. Anschließend wird der Messaufbau vorgestellt. Es wird hierbei auch näher auf die Konstruktion eingegangen.

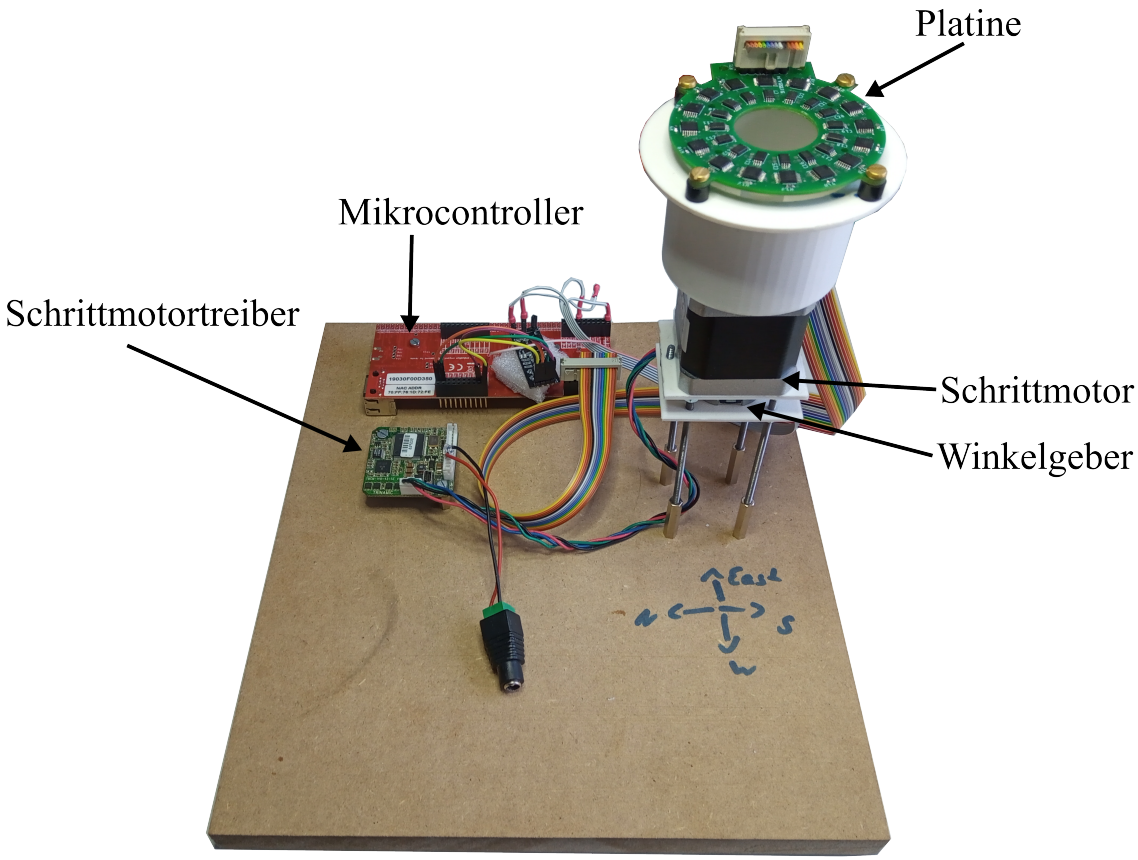

Abbildung 4.1: Bild des Versuchsaufbaus

4.1 Anforderung an die Messung

Der Messaufbau zum Testen der Platine muss ermöglichen, einen kleinen Magneten kontrolliert im Mittelpunkt der fixierten Platine zu drehen und den durch die Platine ermittelten Drehwinkel mit einem Referenzdrehwinkel hoher Genauigkeit zu vergleichen.
Hierzu werden auf einem Schrittmotor mit Doppelwelle über einen Adapter auf einer Seite die Platine und auf der anderen ein hochauflösender Winkelgeber installiert. Auf der Seite der Platine wird auf der Motorwelle der Magnet installiert, der beim Schraubwerkzeug auf dem Schraubenende sitzt. Dadurch, dass die Platine über starre Bauteile auf dem Schrittmotor installiert wird kann ein starkes Verkippen der Sensorplatine in Referenz zum Magneten vermieden werden.
In der Nähe des befestigten Schrittmotors werden der Mikrocontroller sowie der Schrittmotortreiber installiert. Über Flachbandkabel sind diese mit der Sensorplatine bzw. dem Schrittmotor verbunden.

4.2 Aufbau mit Schrittmotor und CUI-Encoder

Es ist erforderlich, die Platine und den Magneten über den Platinenadapter in ausreichender Entfernung zum Schrittmotor zu installieren, um Einflüsse auf die Ausbreitung des Feldes des Magneten durch die ferromagnetischen Bauteile des Schrittmotors zu vermeiden. Außerdem, kann so eine Einwirkung durch das Fremdfeld des stromdurchflossenen Schrittmotors verringert werden.
Die Bauteile des Aufbaus werden durch die vier Gewindekanäle an den Kanten des Schrittmotors mit langen Gewindestangen am Schrittmotor befestigt.
Der Winkelgeber wird an der Welle auf der Unterseite des Schrittmotors befestigt. Er wird über eine auf dem Motor aufliegenden Platte durch zwei Schrauben an den Flügeln des Winkelgebers zentriert. Diese Schrauben dienen nicht der Befestigung des Winkelgebers. Über eine zweite Platte wird der Winkelgeber über vier Muttern an den Gewindestangen am Schrittmotor befestigt. Die Befestigung des Winkelgebers ist in der Abbildung 4.4 gezeigt

Der zylinderförmige Adapter der Sensorplatine wird an der Oberseite des Schrittmotors durch Muttern über die vier Gewindestangen auf dem Schrittmotor befestigt. Die Befestigung des Platinenadapters ist in der Abbildung 4.3 gezeigt.

4.3 Entwicklung und 3D-Druck der Verbindungsteile

Um eine schnelle Entwicklung des Messaufbaus zu ermöglichen, werden die Verbindungsteile für den Messaufbau per CAD-Software entwickelt und im Anschluss mit einem 3D-Drucker gedruckt. Die Entwicklung mit 3D-Druck ermöglichst es auch, Bauteile leicht anzupassen und schnelle Änderungen am Messaufbau vorzunehmen. So konnten Magnetadapter schnell hergestellt werden, mit welchen es möglich war, eine Verschiebung des Magneten in der Z-Achse zu simulieren. Ein weiterer Vorteil von mit Kunststoffen 3D-gedruckten Teilen ist, dass sie weder magnetisch noch elektrisch leitfähig sind. Dies verhindert eine Ablenkung des magnetischen Feldes, und die Wahrscheinlichkeit eines Kurzschlusses des Schrittmotors mit der Platine.

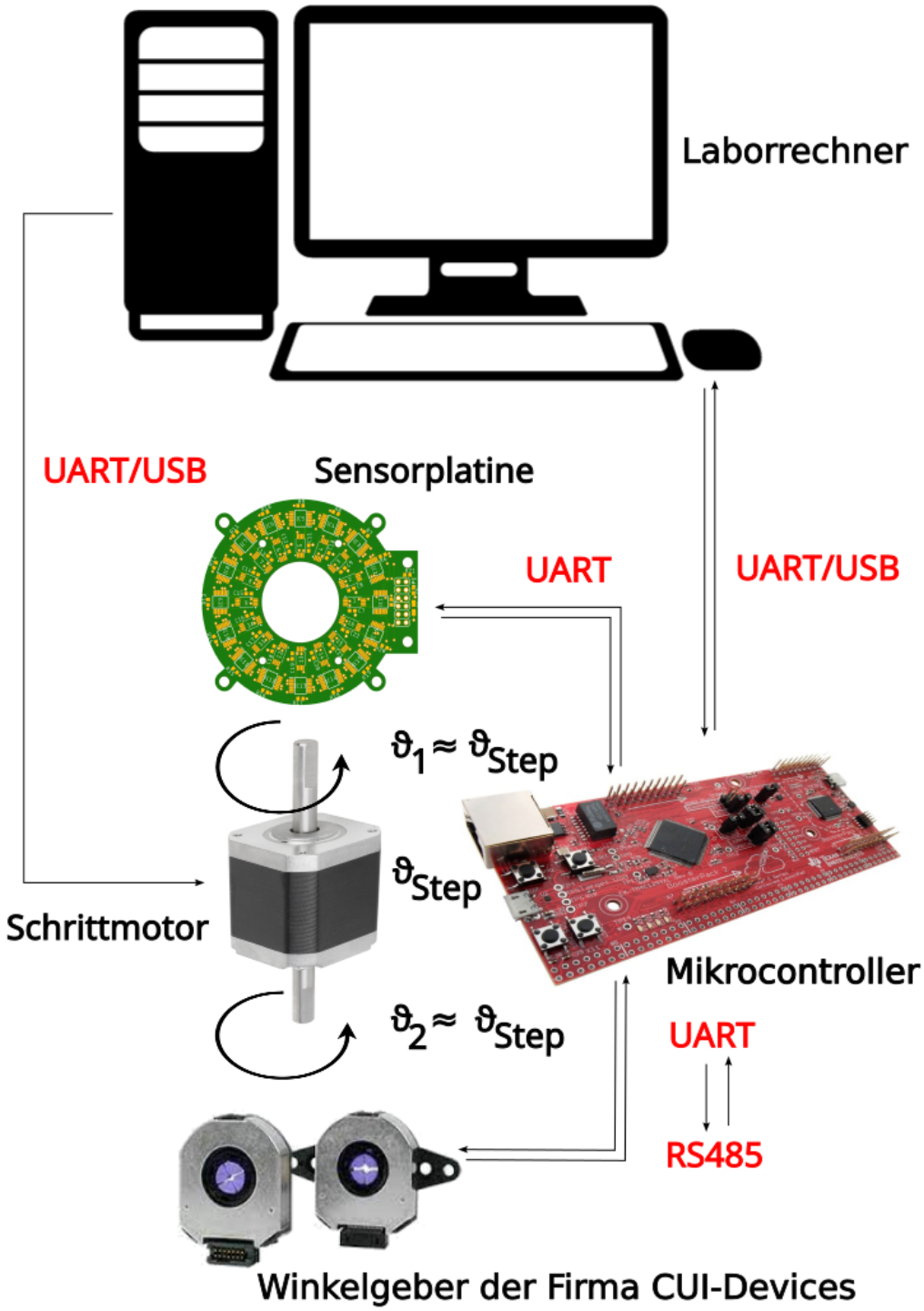

Abbildung 4.2: Skizze des Versuchsaufbaus und der Kommunikationsschnittstellen der einzelnen Elemente

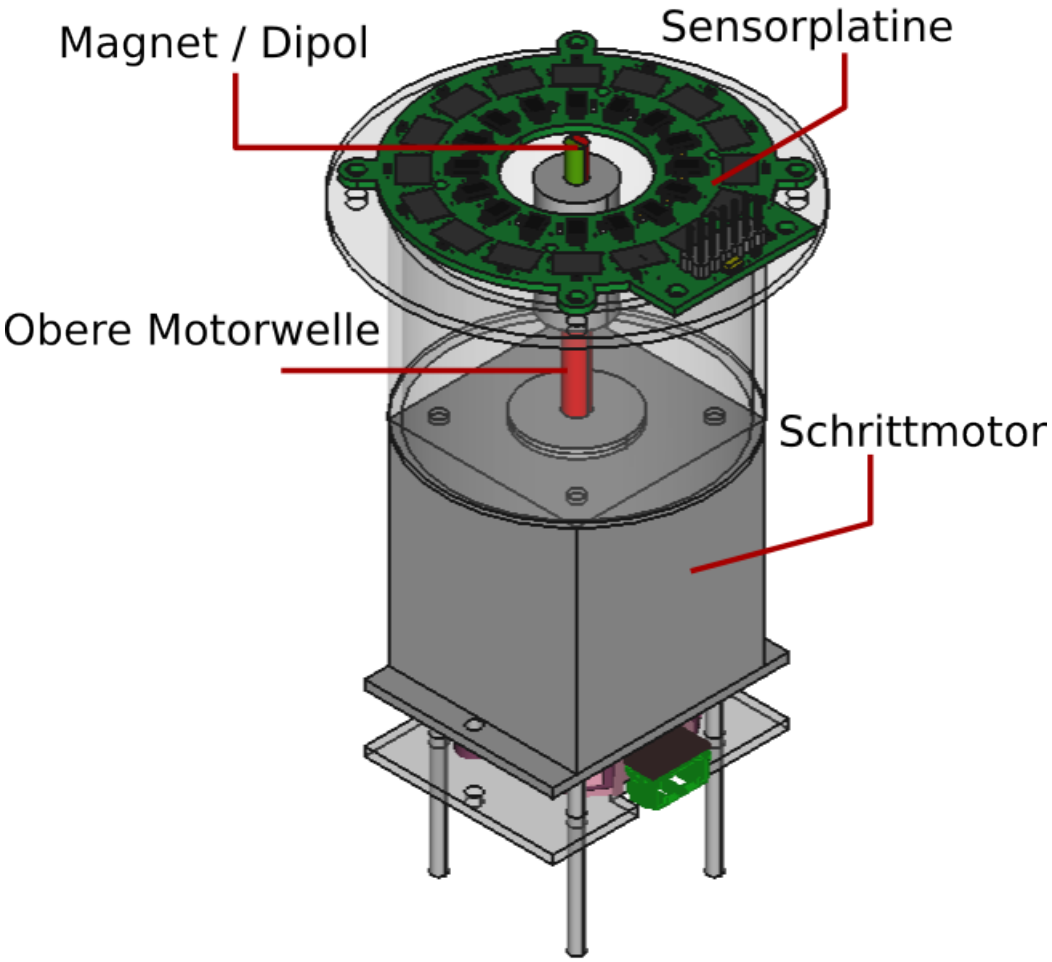

Abbildung 4.3: CAD-Modell des Versuchsaufbaus. Ansicht von oben. Die Platine ist über 2 mm hohe Abstandshalter mit 4 Schrauben am transparenten Adapter befestigt, welcher ebenso mit 4 Schrauben am Schrittmotor befestigt ist.

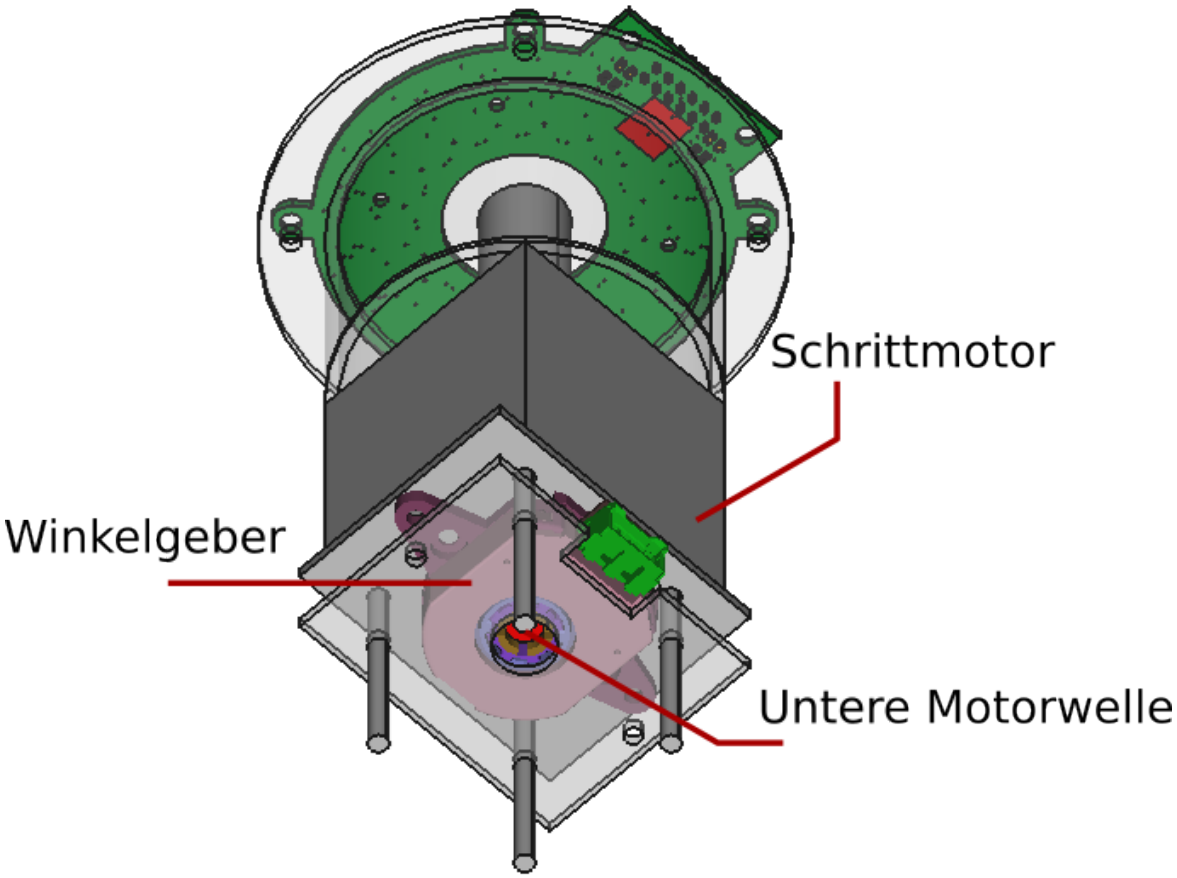

Abbildung 4.4: CAD-Modell des Versuchsaufbaus. Ansicht von unten. Die transparente Platte unter dem Winkelgeber befestigt ihn über 4 Muttern an den durch sie führenden Gewindestangen und damit am Schrittmotor.

5 Bewertung der Messergebnisse

Dieses Kapitel beinhaltet die Auswertung, der mit dem Messaufbau gemachten Messungen. Die Messungen werden beschrieben und graphisch dargestellt. Im Anschluss werden die Messungen interpretiert. Hierbei wird auch auf die Nutzung der Sensorplatine in einem Schraubwerkzeug genauer eingegangen. Zum Schluss werden mögliche Fehlerquellen behandelt.

5.1 Messwerte und graphische Auswertung/Darstellung

Nachdem die Funktionalität der Platine bestätigt wurde, kann sie am Platinenadapter des Messaufbaus befestigt werden. Nun wird ein radial magnetisierter Stabmagnet über einen Magnetadapter auf der Motorwelle angebracht. Der Magnet befindet sich auf gleicher Höhe im Zentrum der Platine.
Nach einer Abfrage aller Sensoren werden die Messwerte geplottet. Wird der Magnet nun auf dem Schrittmotor so lange gedreht, bis der Nordpol in Richtung des vierten Sensors zeigt, ergeben sich die Messkurven aus Abbildung 5.1. Aus den gemessenen Werten sind klar die in Abbildung 1.10 dargestellten zu erwartenden Kurven mit Sinus- und Cosinusverlauf zu erkennen.

5.2 Vergleich mit Messwerten des Winkelgebers AMT212B-V der Firma CUI-Devices

Um die Präzision des mit der Platine gemessenen Winkels zu schätzen ist es notwendig, ihn mit einer Referenz von höherer Präzision zu vergleichen. Hierzu wird der kapazitiver Winkelgeber AMT212B-V der Firma CUI-Devices genutzt. Dieser kann seriell einen absoluten Drehwinkel ausgeben und verspricht dabei eine Auflösung des Winkels von 14 Bit. Die Winkeldifferenz bei Umschalten des Least Significant Bit (LSB) beträgt also

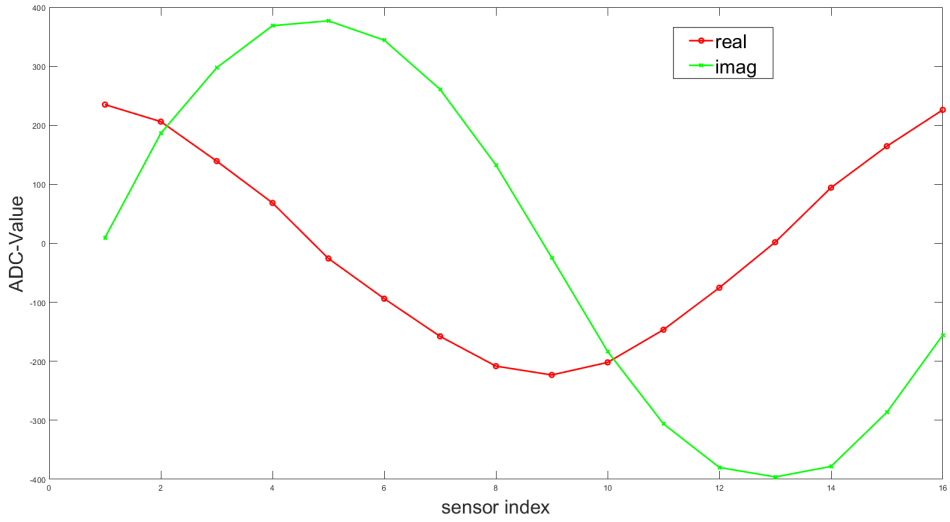

Abbildung 5.1: ADC-Abtastwerte der Sensoren bei Drehwinkel ca. 0°

0.02197°. In Abbildung 5.2 ist der Fehler des Winkelgebers zwischen zwei Messungen über 360° dargestellt. Ein regelmäßiges Kippen zwischen -LSB und +LSB ist zu beobachten. Schrittmotor und Winkelgeber ermöglichen also, mit guter Reproduzierbarkeit diskrete Referenzwerte über eine Drehung von 360° im Abstand von ca 1.8° mit einer Präzision von +-0.02197° zu erreichen.

5.2.1 Multiplexerplatine

An allen 16 Sensoren der Multiplexerplatine kann das zu erwartende Signal mit guter Reproduzierbarkeit ausgelesen werden. Durch die in den Grundlagen beschriebene Theorie lässt sich nach Abspeichern der Werte an einem Laborrechner aus ihnen ein Drehwinkel bestimmen. Bei statischer Betrachtung des Magneten schwankt der über 20 Mittelungen berechnete Winkel ϕ mit ca. $-0.1° < \phi < 0.1°$ um einen Wert.
Nun wird der Magnet in 200 Schritten von $1,8°$ um 360° gedreht. An jeder Position wird wie zuvor ein Winkel über 20 Mittelungen berechnet und abgespeichert. Es wird anschließend ein beliebiger Referenzwinkel des Winkelgebers gewählt. An diesem Referenzwinkel werden die aus der Sensorplatine und dem Winkel ermittelten Werte gleich dem Drehwinkel 0 gesetzt. Werden nun die Winkel der Sensorplatine über eine Drehung mit den Referenzwinkeln des Winkelgebers verglichen ergibt sich ein charakteristischer Fehler. Dieser absolute Fehler ist beispielhaft links oben in Abbildung 5.3 gezeigt. Dieser

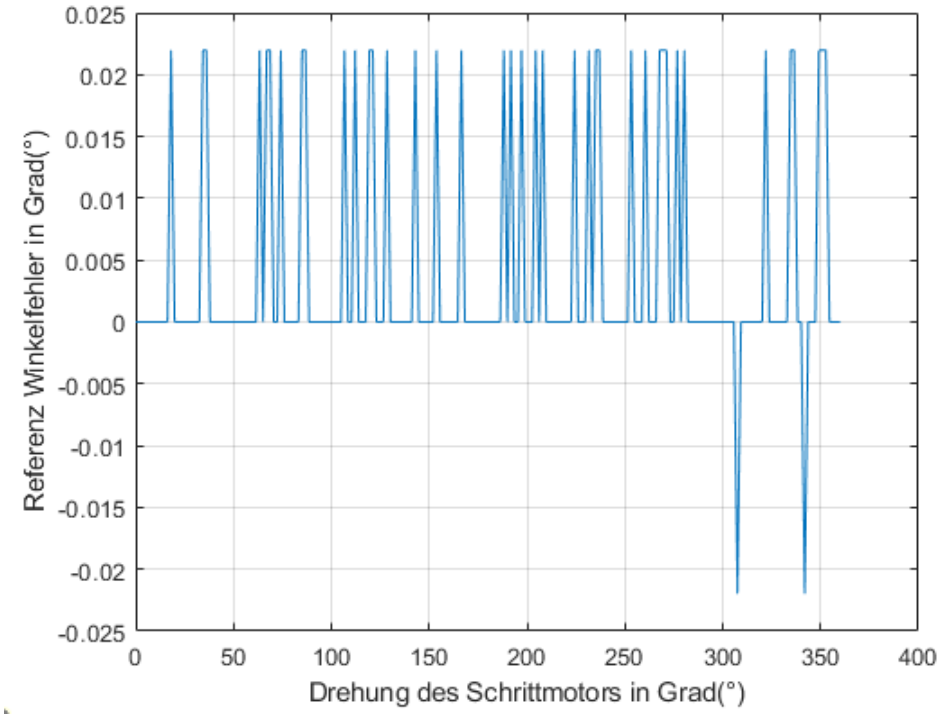

Abbildung 5.2: Winkelfehler des Winkelgeber der Firma CUI-Devices über 360 ° Drehung. Diese Werte werden als Referenzwinkel verwendet.

absolute Fehler schwankt stark zwischen Messungen. Es wurden Kurven des absoluten Fehlers mit Maximalwerten zwischen 0.4 und 2 gemessen Er weist über eine Periode von 360° eine stark ausgeprägte Modulation mit der doppelten Frequenz auf. des Weiteren sind auch hochfrequentere Schwankungen mit niedrigeren Amplituden erkennbar.
Der absolute Fehler über eine gesamte Rotation ist gut reproduzierbar. Wird er erneut über eine zweite Rotation aufgenommen und immer an denselben Winkelstellungen betrachtet, weichen die Werte mit ca. $-0.1° < \phi < 0.1°$ von einander ab. Diese Betrachtung entspricht also in etwa einer statischen Betrachtung. Werden nun die beiden Fehler von einander subtrahiert, kann der absolute Fehler kompensiert werden. Dies ist schon mit dieser einfachen Methode mit einem guten Resultat möglich. Der kompensierte Winkelfehler ist für die Multiplexerplatine beispielhaft links unten in Abbildung 5.3 gezeigt. Durch Mittelung der Messwerte kann die Präzision der Winkelmessung deutlich erhöht werden. In Abbildung 5.4 ist die maximale Differenz zwischen den absoluten Winkelfehlern bei der gleichen Stellung für 50 volle Rotationen des Schrittmotors gezeigt. Es

ist zu erkennen, dass der Winkelfehler negativ exponentiell abfällt und sich ab ca. 20 Mittelungen im Bereich 0.1 befindet.

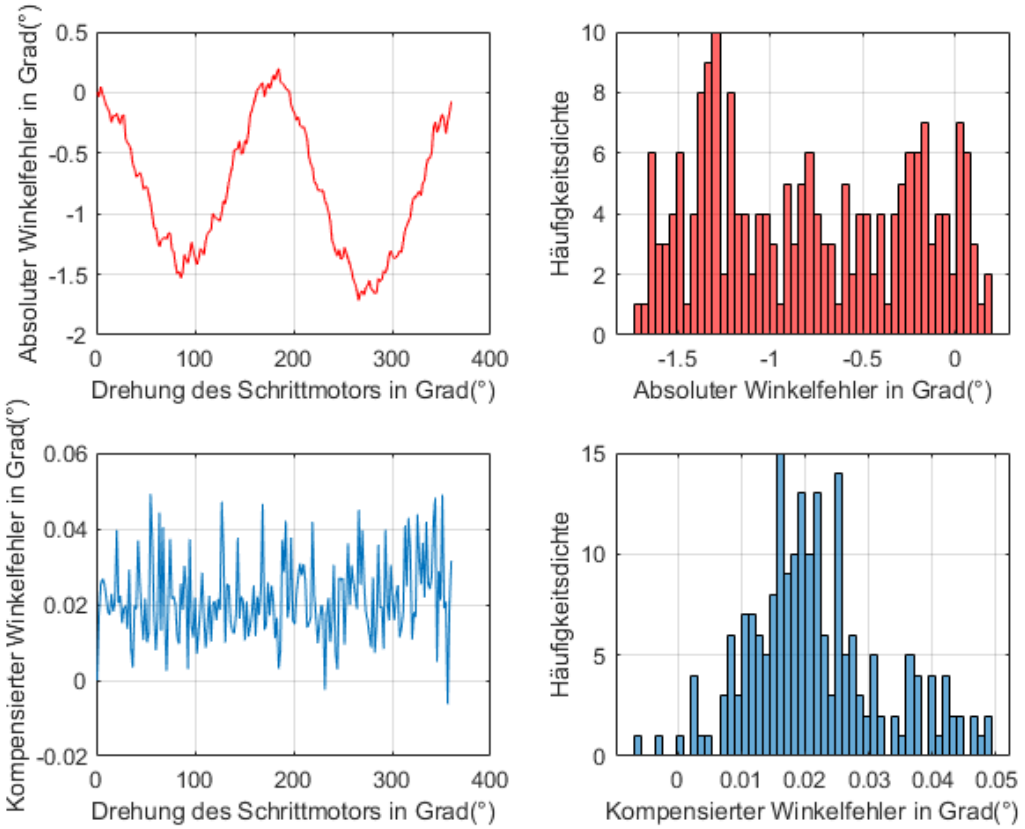

Abbildung 5.3: Multiplexerplatine. Winkelfehler über 360° Drehung. Ringmagnet in ca. -2 cm Entfernung zur Platine. Kompensierung durch Subtraktion des Fehlers zwischen 2 Messungen

5.2.2 Diodenplatine

Die Diodenplatine lässt sich im Grunde genommen auf dieselbe Weise wie die im letzten Abschnitt behandelte Multiplexerplatine auslesen und auswerten. Auch das Verhalten der Platine ist ähnlich wie das der Multiplexerplatine. Dies ist in Abbildung 5.5 gezeigt. Deutlich hebt sich allerdings ein auf den Winkelfehler moduliertes Signal ab. Dies ist in der Abbildung links oben zu erkennen. Über eine Periode von 360° ist ein Signal von der 16-fachen Frequenz mit in etwa einer Amplitude von ±1° zu erkennen. Es ist zu vermer-

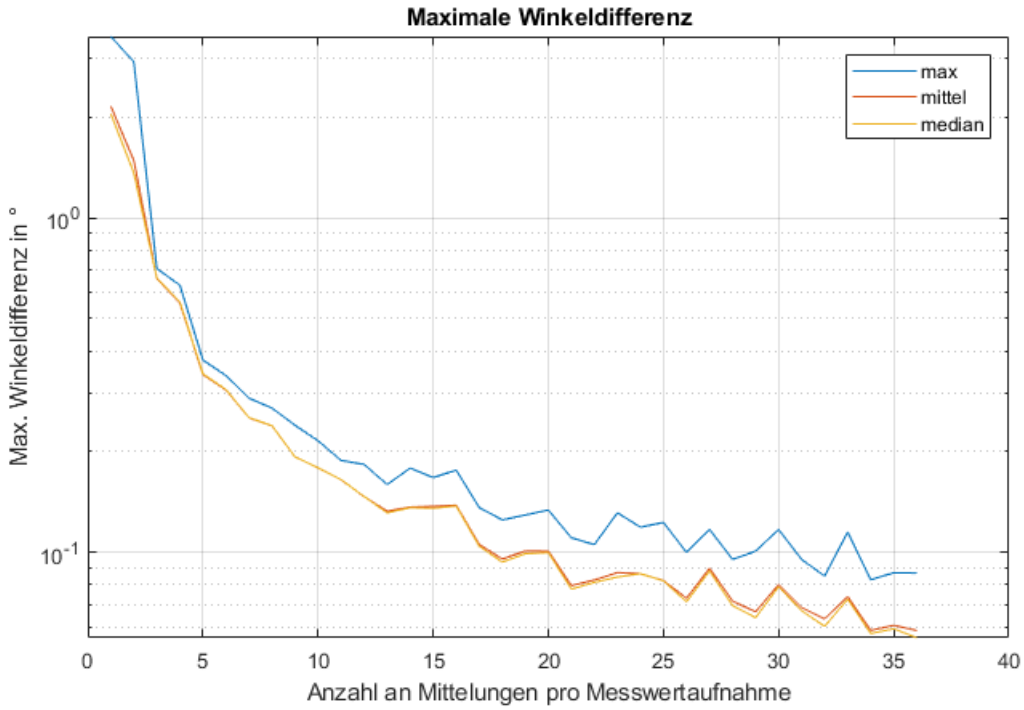

Abbildung 5.4: Maximale Differenz zwischen Winkelfehlern bei gleichen Drehwinkeln über 50 volle Rotationen. Multiplexerplatine und Stabmagent ohne Verschiebung auf der Z-Achse.

ken, dass die Frequenz dieser Modulation mit der Anzahl der Sensoren auf der Platine übereinstimmt.
Auch diese Modulation ist allerdings charakteristisch und lässt sich wie im letzten Abschnitt gut durch eine einfache Subtraktion der Fehler mehrerer Messwertaufnahmen kompensieren. Der hierbei berechnete kompensierte Fehler scheint jedoch etwas mehr zu streuen, als der kompensierte Fehler der Multiplexerplatine.

5.2.3 Betrachtung der Fehlerquellen

Die Funktionsweise der Platine wird in der Theorie unter idealen Umständen betrachtet. In der Realität gibt es jedoch bei der Erprobung im Labor und für die spätere Nutzung als Werkzeug in Werkstätten einige Ursachen, welche die Präzision der Winkelbestimmung beeinflussen.
Bei einer idealen Betrachtung der Platine wird angenommen, dass sich die Sensoren in ei-

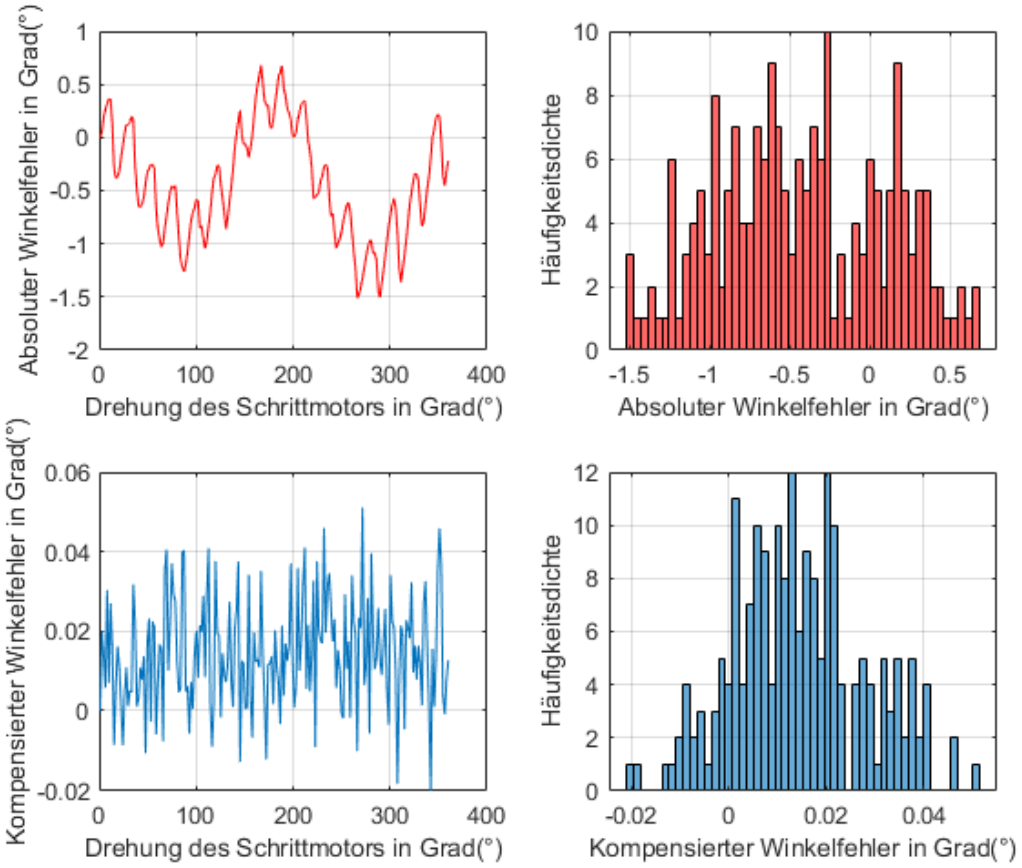

Abbildung 5.5: Diodenplatine. Winkelfehler über 360 ° Drehung. Ringmagnet in ca. -2 cm Entfernung zur Platine. Kompensierung durch Subtraktion des Fehlers zwischen 2 Messungen

nem Kreis mit exakt gleichem Radius und auf der exakt gleichen Höhe zum magnetischen Dipol befinden. Durch maschinelle Bestückung ist es möglich, eine hohe Präzision bei der Bestückung der Platine zu erreichen. Jedoch auch bei guter Ausrichtung der Sensoren auf der Platine sind Fehlerquellen häufig nicht zu vermeiden. So ist auch eine gewisse Toleranz bei der Platzierung der Chips im IC-Package der Sensoren zu erwarten. Auch führen Unebenheiten auf der Platine oder den Packages der Sensoren und Verformungen der Platine zu Verkippungen der Sensoren zu einander. Diese Fehlerquellen bei der Bestückung und Ausrichtung der Sensoren sind in Abbildung 5.6 gezeigt.
Einen Einfluss auf die Präzision der Winkelbestimmung hat auch eine Verschiebung des Magneten bzw. der Platine auf der Z-Achse. Die genaue Platzierung in dieser Ebene

dürfte sich in der Benutzung als Schraubwerkzeug vermutlich als sehr schwierig herausstellen. Um diesen Fehlerfall zu betrachten sind per 3D-Druck Magnetadapter in unterschiedlichen Höhen in Referenz zur Platine gedruckt worden. Anschließend wurde der Winkelfehler für 6 unterschiedliche Positionen des Magneten gemessen. Die Ergebnisse sind in Abbildung 5.7 gezeigt. Es ist zu erkennen, dass sowohl der absolute als auch der kompensierte Fehler abnehmen, je weiter der Magnet in das Zentrum der Platine verschoben wird. Bei der Messung des absoluten Winkels ist jedoch eine signifikante Steigerung des maximalen Fehlers bei einer Verschiebung von -5 mm zu erkennen. Dies liegt an der zufälligen Wahl des Referenzwinkels. Dieser wird an der Stelle gewählt an der der Winkelgeber den Referenzwinkel 0° ausgibt. Dies is unabhängig von der Ausrichtung des Magneten, welcher bei allen Messungen unterschiedlich ausgerichtet war.
Der Kompensierte Fehler ist entgegen dem absoluten Fehler nicht von der Ausrichtung des Magneten Abhängig, da der charakteristische Fehler durch die Kompensation durch Subtraktion des Fehlers herausgerechnet ist. Nach der Kompensation ist ein Minimum des maximalen Fehlers bei -5 mm, also leicht unter der idealen Platzierung im Zentrum der Platine zu erkennen. Der kompensierte Fehler nimmt mit zunehmendem Abstand stetig zu.

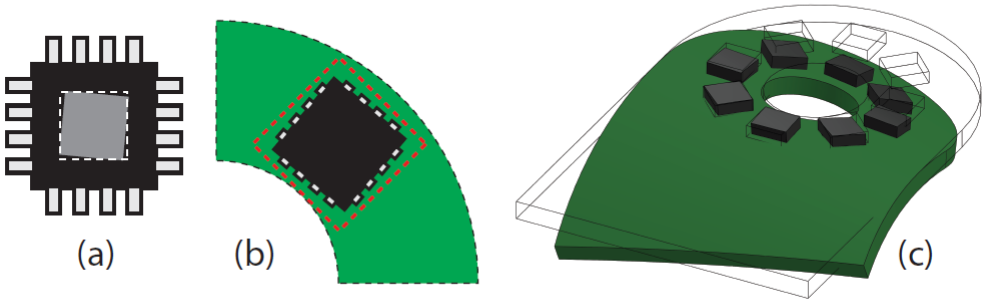

Abbildung 5.6: Verschiedene Fehlerquellen durch Fehlausrichtung der Sensoren: Ausrichtung der Brücken auf dem Chip (a), Ausrichtung auf der Platine (b), Verformung der Platine (c) [7]

5.3 Interpretation im Kontext der Anwendung als Schraubwerkzeug

Wird die Tabelle aus Abbildung 1.3 betrachtet und ein lineares Verhalten des Torsionswinkels einer Schraube zur auf das Werkstück wirkenden Vorspannkraft angenommen ergibt sich für eine 100 mm lange M8-Schraube ein Verhältnis von 0.224 °/kN. Dieses Verhältnis nimmt für kürzer- und im Durchmesser größer-werdende Schrauben ab. Für eine 60 mm lange M16 Schraube beträgt es nur noch ca. 0.015 °/kN. Eine Auflösung des Torsionswinkels in der Größenordnung 0.01° könnte demnach eine Voraussetzung für viele Anwendungen sein.
Eine realistische Schätzung des absoluten Drehwinkels eines Dipols ist mit dem in dieser Arbeit betrachteten Messaufbau in der Größenordnung 10° möglich. Bei unveränderten Bedingungen ist eine Kompensierung des Fehlers durch Subtraktion möglich und ermöglicht dadurch eine Schätzung der Größenordnung 1°.
Durch weitere Signalverarbeitung mit Kalibrierungsalgorithmen könnte es möglich sein, den Drehwinkel unter kontrollierten Bedingungen in der Größenordnung 0.1° verlässlich zu schätzen.
Ein nicht in dieser Arbeit behandelter Aspekt der Funktionsweise des Schraubwerkzeugs ist die Betrachtung der Resistenz gegenüber magnetischen Streufeldern. Obwohl die runde Bauweise und die Auswertung der Sensorwerte durch die Fouriertransformation theoretisch eine Unterdrückung der Einflüsse von Streufeldern auf die Winkelbestimmung haben, ist diese in dieser Arbeit nicht betrachtet und ihre Effektivität dadurch nicht beobachtet worden.
des Weiteren wurde auch keine Auftrennung mehrerer Felder erprobt. Somit kann keine Aussage darüber getroffen werden, ob es möglich ist, die Winkelinformation eines von einem äußeren Halbacharray ausgehenden homogenen Feld vom den Informationen des Felds des in der Mitte liegenden Dipols zu trennen. Auch über die Winkelgenauigkeit eines äußeren homogenen Feldes kann keine Aussage getroffen werden. Winkelfehler durch die Überlagerung der Felder können nicht ausgeschlossen werden, welche die Genauigkeit beider Winkelmessung beeinträchtigen können.

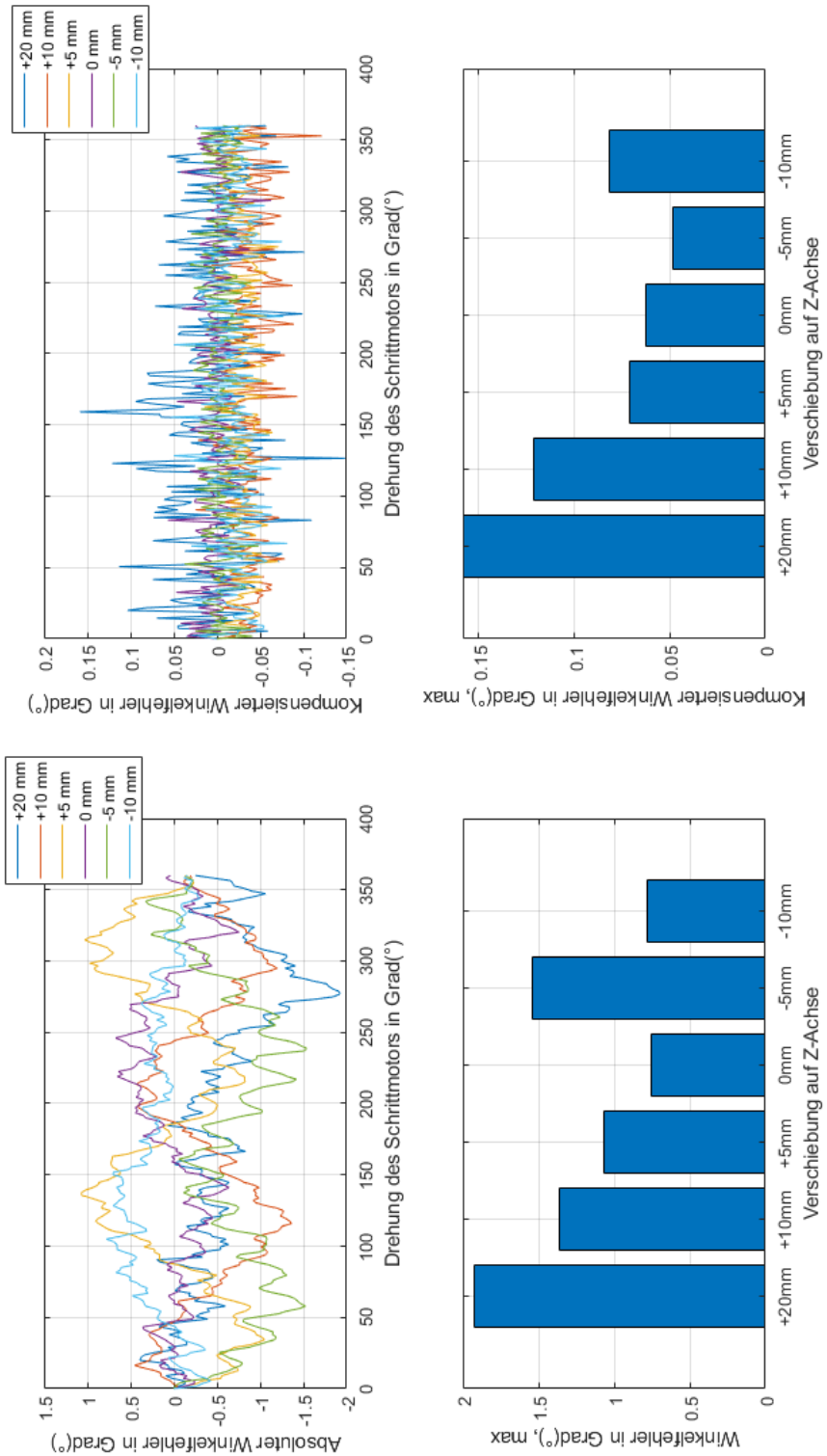

Abbildung 5.7: Winkelfehler bei Verschiebung des Magneten in der Z Ebene mittels Stabmagnet

6 Fazit

6.1 Zusammenfassung

Zusammenfassend kann aus dieser Arbeit Folgendes geschlossen werden. Die Drehwinkelmessung eines magnetischen Dipols mittels eines runden TMR-Sensorarrays auf einer Sensorplatine ist möglich und kann mit weiterem Forschungsaufwand aussichtsreiche Präzisionen erreichen. Im Rahmen dieser Arbeit wurde mit einem runden Array, bestehend aus 16 TMR-Sensoren mit einem Radius von 17.5 mm um einen Magneten, ein absoluter Winkelfehler von $<2°$ in Referenz zu einem 14-Bit Winkelgeber des Typs AMT212B erreicht. Dieser absolute Fehler variiert jedoch stark. Er wurden zwischen verschiedenen Messungen Änderungen des maximalen absoluten Fehlers von bis zu 500 % gemessen. Mittels einfacher Kompensation des charakteristischen absoluten Fehlers war es möglich, den Winkelfehler auf $<0.1°$ zu reduzieren. Es wurde eine charakteristischer Fehlerverlauf mit einer Modulation mit der Frequenz $f = 2$ in Referenz zur Periode 360° beobachtet. Bei der Diodenplatine konnte zusätzlich hierzu noch eine Modulation des charakteristischen Fehlers mit der Frequenz $f = 16$ in Referenz zur Periode 360° beobachtet werden.

6.2 Ausblick

Im Rahmen dieser Arbeit konnten einige Aspekte nicht behandelt werden. Es ist deshalb nötig, in weiteren Versuchen Erkenntnisse über diese zu erlangen. Ein wichtiger Teil ist die Erforschung der Fehlerquellen und ihrer Auswirkung. Viele dieser Quellen haben komplexe Abhängigkeiten voneinander oder weisen kein lineares Verhalten auf, was es schwieriger macht, sie in Simulationen zu erfassen.
Über weitere Experimente können etwa genauere Informationen über kontrollierte Verkippungen des Magneten oder der Platine gesammelt werden. Auch eine Verschiebung

des Magneten oder der Platine in der X/Y-Ebene könnte aufschlussreiche Daten zur Fehlerermittlung beitragen.
Die Störfeldunterdrückung ist eine Vermutung, welche noch experimentell zu beweisen ist. Ebenso ist die Vermutung über die Trennbarkeit sich überlagernder homogener Felder und Felder magnetischer Dipole zu beweisen.
Für all diese Betrachtungen bietet sich die Konstruktion bzw. Nutzung eines Platinenmessplatzes an mit welchem Verkippung, lineare Verschiebung und Rotation möglich ist.
Für die Überprüfung der Trennbarkeit der Felder empfiehlt sich ein Test mit magnetischen Halbacharrays. Mit diesen können homogene Störfelder simuliert werden und des Weiteren können sie auch als Referenzwinkelquelle genutzt werden.

Literaturverzeichnis

[1] Albounyan, Ahmed: *Charakterisierung magnetischer Winkelsensoren mittels Kreuzspulenmessplatz.* HAW-Hamburg, 2018

[2] Crocus Technologies: *CT310 XtremeSense® 2D TMR Angular Sensor Datasheet.* 2021

[3] Crocus Technologies: *Sensor Illustration.* 2023. – URL https://crocus-technology.com/products/ct310/

[4] Mehm, Thorbjörn: *Schaltungsentwurf und Mikrocontrollersteuerung für ein Tunnel-Magnetoresistives Sensor-Array.* HAW-Hamburg, 2019

[5] Neubauer, André: *DFT - Diskrete Fourier-Transformation.* Springer Vieweg, 2012

[6] Reis, Mario: *Fundamentals of Magnetism.* Elsevir Inc., 2013

[7] Ripka, Pavel ; Fischer, Jan ; Grim, Václav: *Characterization of circular array current transducers.* IEEE, 2019

[8] Schüthe, Thorben ; Riemschneider, Karl-Ragmar ; Meyer-Eschenbach, Andreas: *Magnetic Sensor Array for Determining the Assembly Torsion and Preload of a Bolted Joint.* HAW-Hamburg, 2022

[9] Schüthe, Thorben ; Riemschneider, Karl-Ragmar ; Meyer-Eschenbach, Andreas ; Weithoff, Finn ; Richter, Sarah ; Brodersen, Jannik: *Patent - Werkzeug für die Schraubenmontage mit Magnetsensor-Array zur Torsionsmessung.* -, 2020

[10] Texas Instruments: *CD74HCT4067 High-Speed CMOS Logic 16-Channel Analog Multiplexer/Demultiplexer Datasheet.* 2003

[11] Texas Instruments: *CD74HCT4066-Q1 High-Speed CMOS Logic Quad Bilateral Switch Datasheet.* 2008

[12] Texas Instruments: *EK-TM4C1294XL Tiva™ C Series TM4C1294 Connected LaunchPad Evaluation Kit - User's Guide*. 2016

[13] Tumański, Sławomir: *Handbook of Magnetic Measurements*. CRC Press, 2011

A Anhang

A.1 Platinenlayout

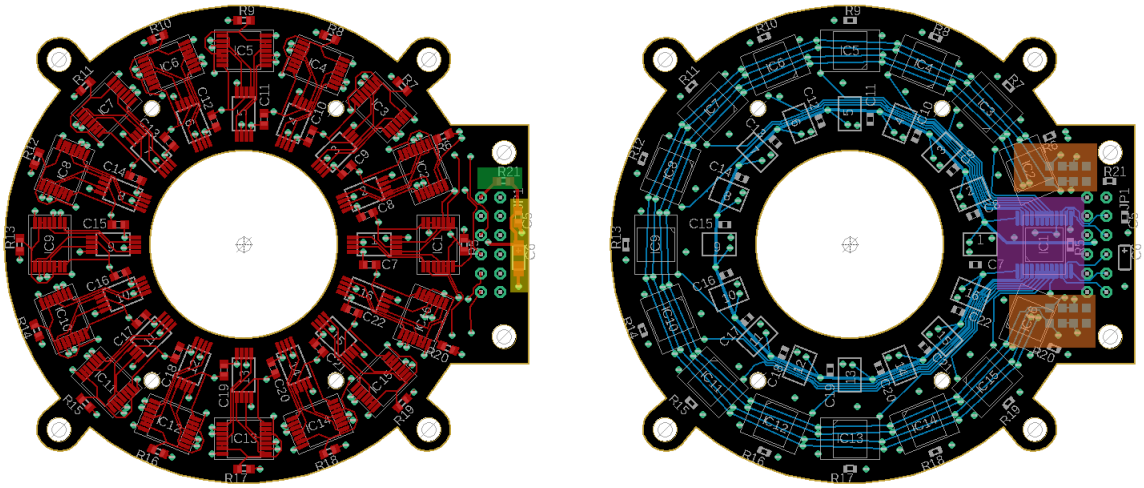

Abbildung A.1: Aufbau der Multiplexerplatine. Links(rot): Leiterbahnen Oberseite: Rechts(rot), Leiterbahnen Unterseite(blau) . Gelb: Glättungs- und Filterkondensatoren für Betriebsspannung, Grün: Widerstand zur Strombegrenzung, Lila: Zentraler Multiplexer, Orange: RC-Filterschaltung für Sensorbusse

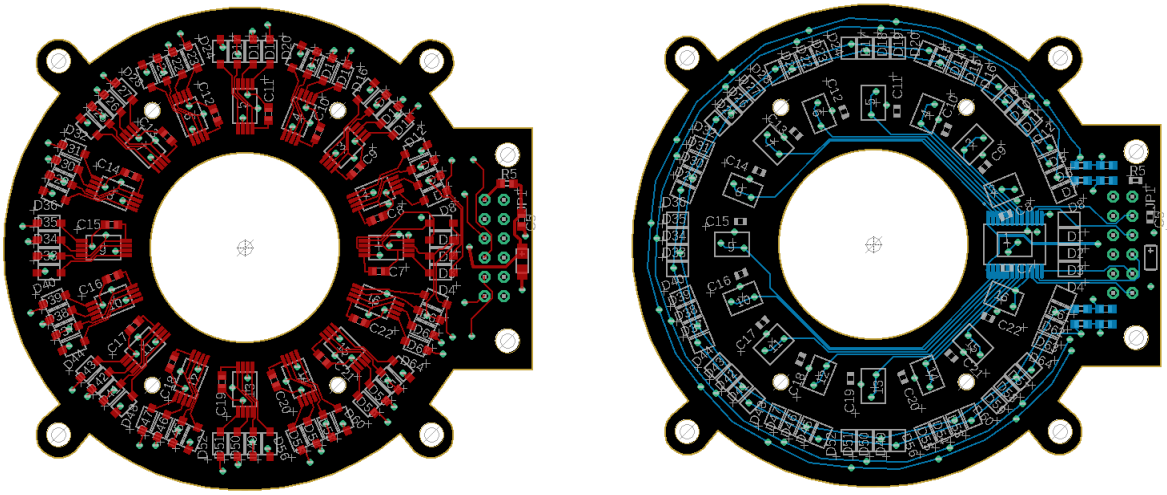

Abbildung A.2: Aufbau der Diodenplatine. Links(rot): Leiterbahnen Oberseite, Rechts(blau): Leiterbahnen Unterseite. Glättungs und Filterkondensatoren, Widerstände zur Strombegrenzung, Multiplexer und RC-Filterschaltungen wie in Abbildung A.1

A.2 Eagle Schematics

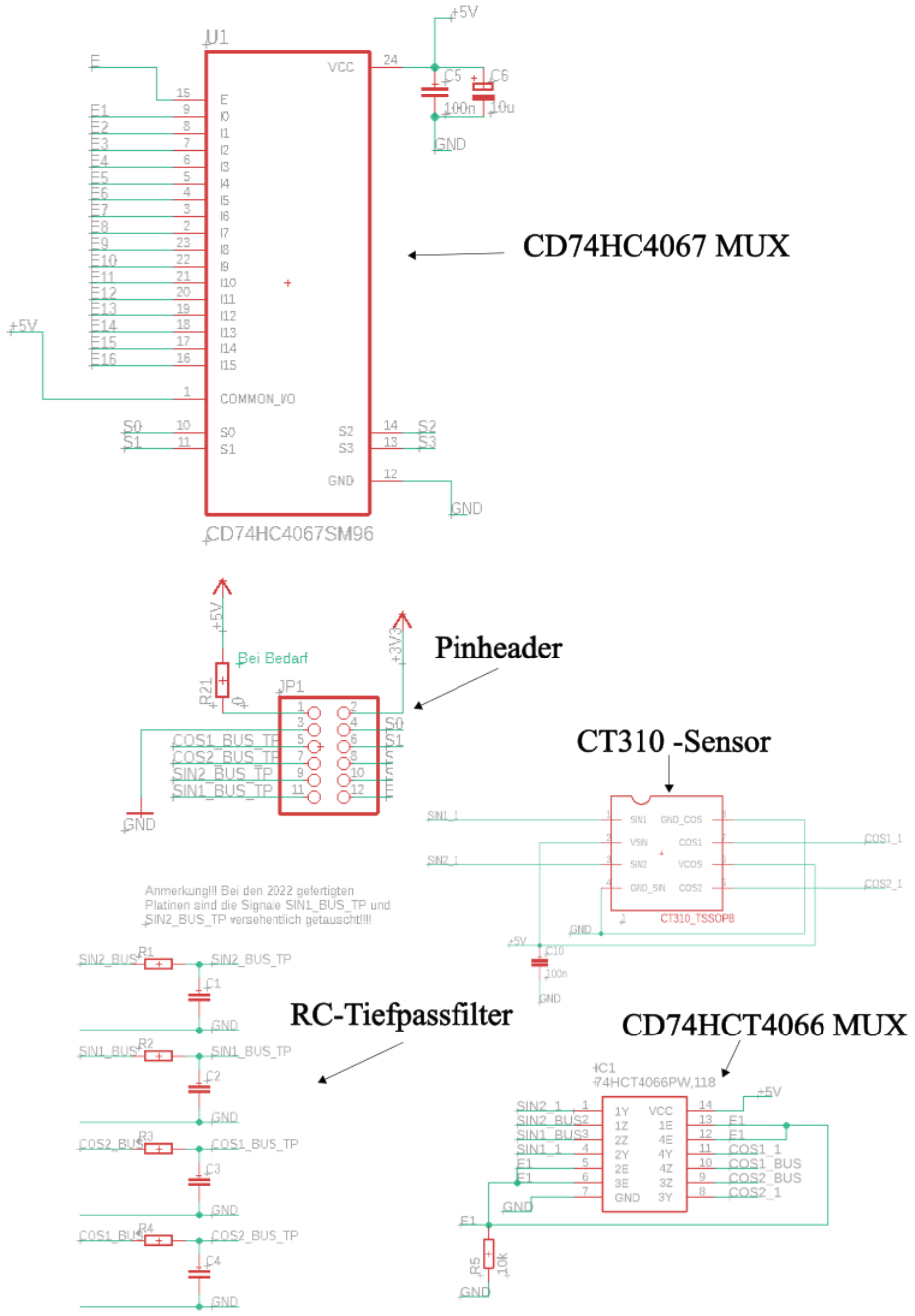

Abbildung A.3: Schaltung der Multiplexerplatine in Eagle für einen Sensor

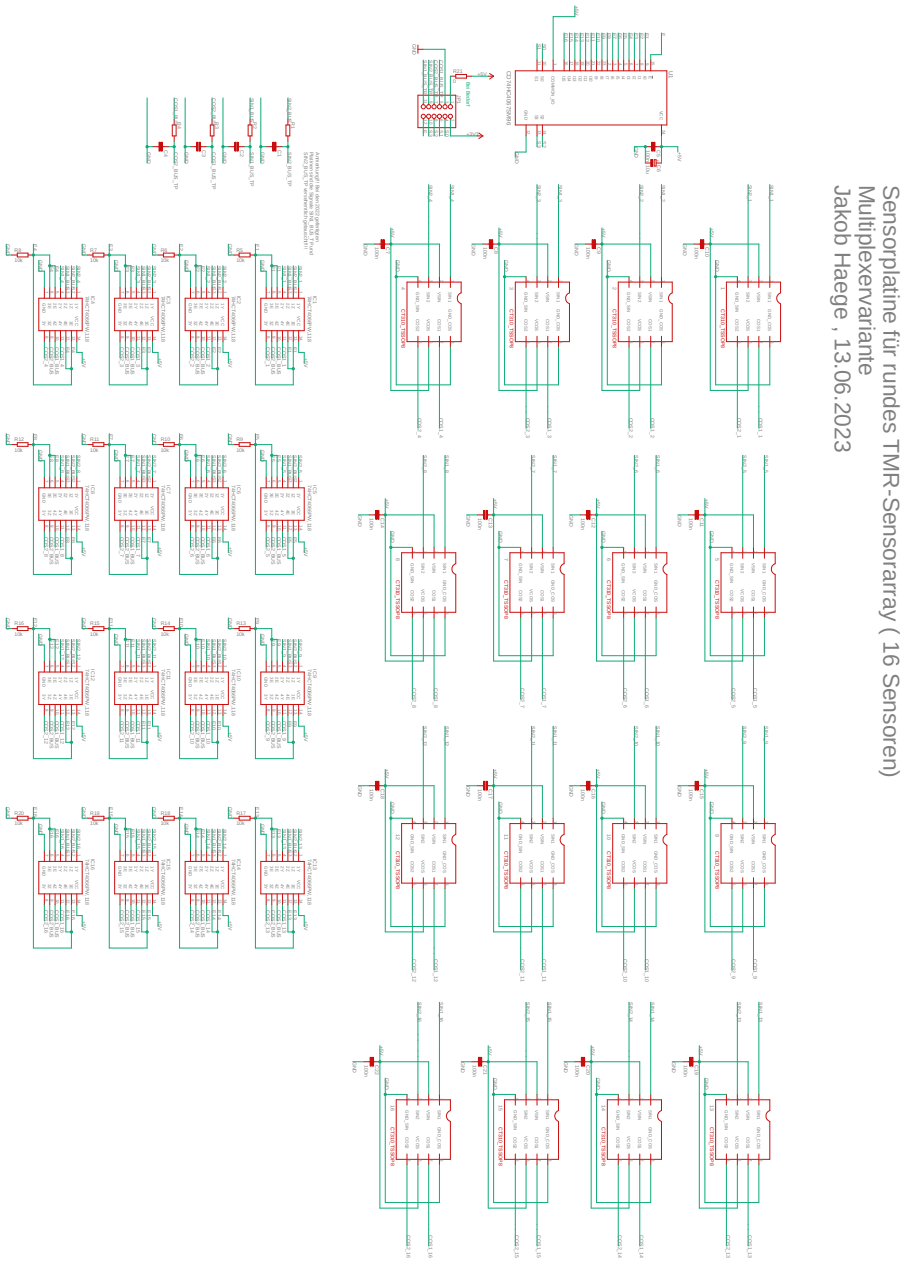

Abbildung A.4: Komplette Schaltung der Multiplexerplatine in Eagle

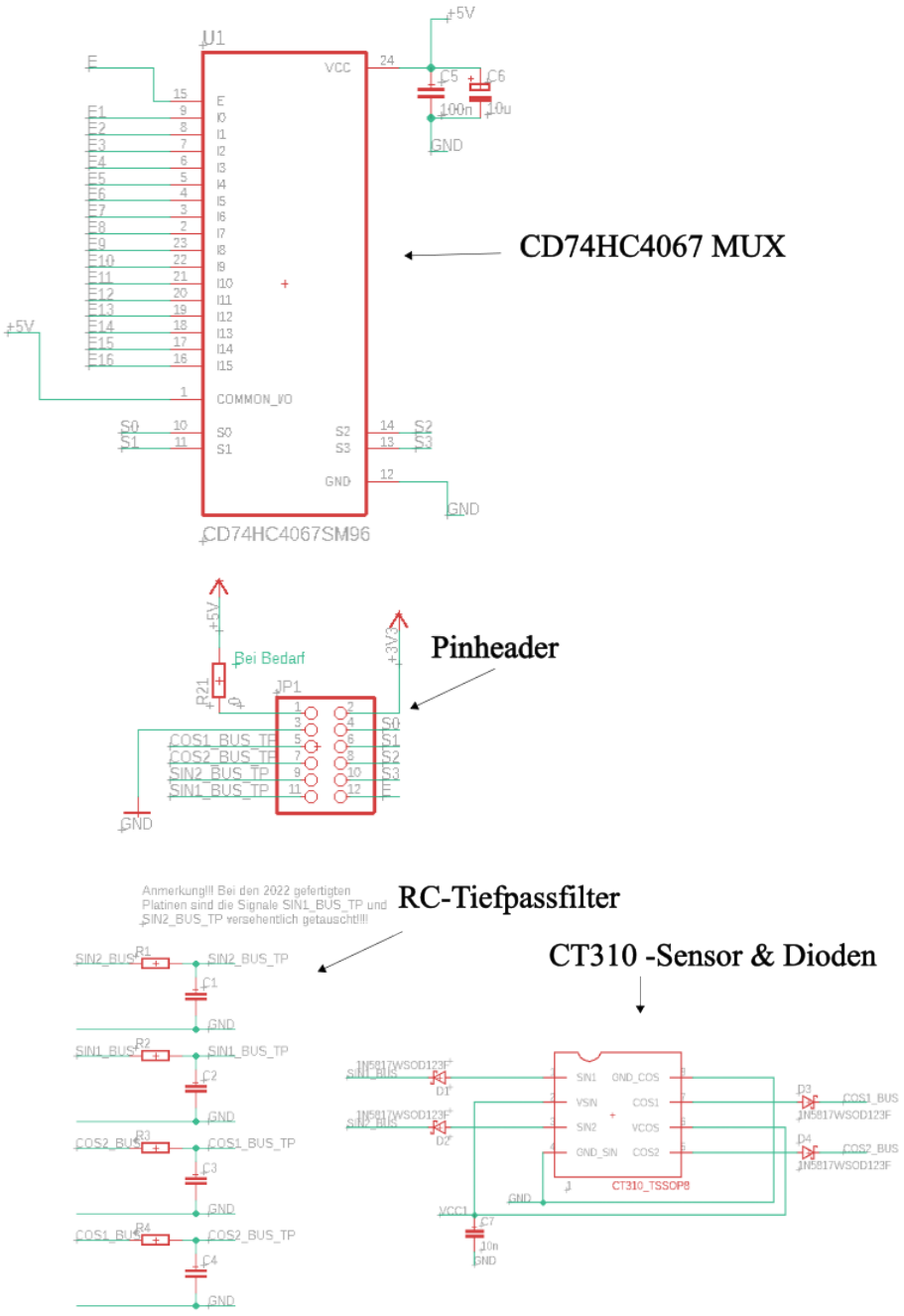

Abbildung A.5: Schaltung der Diodenplatine in Eagle für einen Sensor

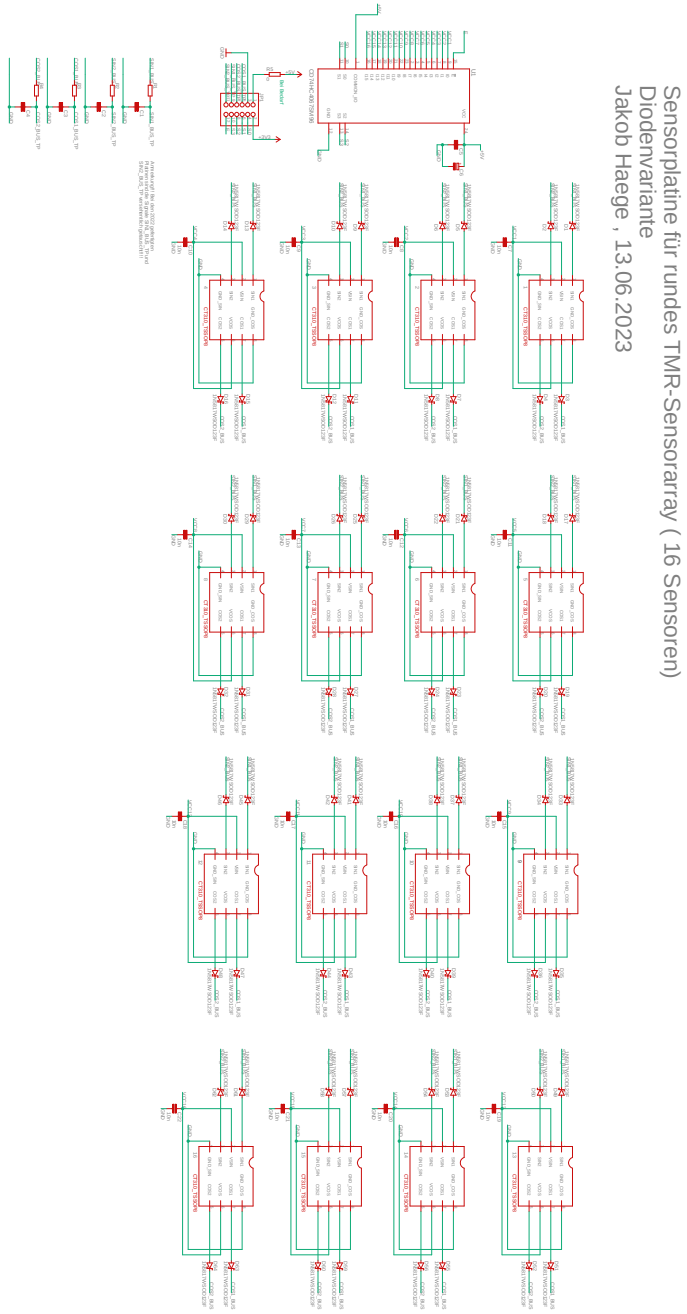

Abbildung A.6: Komplette Schaltung der Diodenplatine in Eagle

A.3 Quellcodes

A.3.1 Mikrocontroller

```
1  //*****************************************************************************
2  //
3  // main.c - Treiber fuer den Versuchsaufbau der Bachelorarbeit von Jakob Haege, 09.06.2023
4  //
5  // Mit diesem Programm ist es moeglich die Multiplexer und Diodenplatine auszulesen
6  // und die formatierten Datan an einen Laborrechner zu uebertragen. Die Platinen wurden
7  // im Rahmen der Bachelorarbeit entwickelt.
8  //
9  // Dieses Programm wird zusammen mit dem Matlab-File "Sensor_reader.m" genutzt.
10 // Nach flashen des Programms auf das Evaluationsboard EK-TM4C1294XL koennen die
11 // Werte der Sensoren ueber ein "Sensor_reader"-Objekt roh ausgelesen werden oder
12 // ueber die entsprechenden Methoden interpretiert werden.
13 //
14 // Daten werden als ASCII-String ausgegeben. Die Formattierung der ausgegebenen Daten ist folgende:
15 //
16 // "cui_winkel,sensor1_sin,sensor1_cos,sensor2_sin,sensor2_cos,...sensor16_sin,sensor16_cos\n"
17 //
18 // Sensor-Werte sind hierbei mit %04d formattiert. Cui-Winkel mit %05d.
19 //
20 //
*****************************************************************************

21
22 #include "stdint.h"
23 #include "stdio.h"
24 #include "stdlib.h"
25 #include "stdbool.h"
26 #include "time.h"
27 #include "string.h"
```

```
28 #include "driverlib/sysctl.c"
29 #include "driverlib/watchdog.c"
30 #include "driverlib/gpio.c"
31 #include "driverlib/uart.c"
32 #include "driverlib/adc.c"
33 #include "driverlib/pin_map.h"
34
35 // Funktionen zur Initialisierung
36 int init_uC(void);
37 int setup_adc(void);
38 int setup_uart(void);
39 int setup_switch_gpio(void);
40 //int setup_stepper(void);
41
42 // Funktionen fuer Mux-Steuerung und Kommunikation
43 int enable_mux(void);
44 int disable_mux(void);
45 int get_ADC0(void);
46 int get_ADC1(void);
47 int uart_print_string(char* string);
48 int switch_select(uint8_t select);
49 int delay(int micros);
50 int cui_rx(char* cui_msg);
51 int trigger_ADC_conversions_and_read_adcs(void);
52
53 // Funktionen fuer Schrittmotor
54 //int stepper_rx(char* msg);
55 //int motor_step(int steps, int rot);
56
57 //Hilfsvariablen und Flags
58 int var = 0;
59 int rx = 0;
60 int z = 7;
61 bool position_reached = 0;
62
63 //Systemvariablen
64 int sysfreq;
65
66 // serielle Puffer
67 char ausgabe[30] = "";
68 char msg[9] = "";
69 char cui_msg[2] = "";
70
71 // Variablen fuer Werte
```

```
72 uint16_t cui_msg_tot = 0;
73 uint16_t cui_angle = 0;
74 uint8_t cui_chksum = 0;
75 uint8_t select = 0;
76 uint32_t adc_value = 0;
77 uint32_t adc_value1 = 0;
78
79 int main(void)
80 {
81     // Ausfuehren der Init-Funktionen zu Beginn des Programmablaufs
82     init_uC();
83     setup_adc();
84     setup_uart();
85     //setup_stepper();
86     setup_switch_gpio();
87
88     // Einschalten des zentralen Multiplexers
89     enable_mux();
90
91     //------------------- !!!!!!!!!!!! Hauptschleife !!!!!!!!!!!!!!
92     while (true){
93         // Schleife der Hauptroutine
94         rx = UARTCharGet(UART0_BASE); // Empfangenes Byte vom Laborrechner
    startet Routine (Blocking)
95
96         UARTCharPut(UART3_BASE, 0x54); // Modul-Adresse des CUI-
    Winkelgebers als TX. Startet Umsetzung in Winkelgeber -> RX 16 Bit
    absoluter Winkel und Checksumme
97         cui_rx(cui_msg); // Empfange Winkel und speichere ihn in globale
    variable "cui_angle"
98
99         // Ausgabe des CUI Wertes an Laborrechner als 1 Wert. Komma
    getrennt.
100        sprintf(ausgabe, "%05d,", cui_angle);
101        uart_print_string(ausgabe);
102
103        for(select = 0; select < 16; select++) {
104            // Auswahl des auszulesenden Sensors
105            switch_select(select);
106            delay(200); // Delay in Mikrosekunden zwischen Sensor-Select
    und ADC-Trigger
107
108            trigger_ADC_conversions_and_read_adcs(); // ADC Umsetzung
109
```

```
110             if(select != 15){
111             // Bei Sensor < 16 -> Ausgabe der ADC-Werte mit Komma am
    Schluss
112             sprintf(ausgabe, "%04d,%04d,", adc_value, adc_value1);
113             uart_print_string(ausgabe);
114             }else{
115             // Bei letztem Sensor (Sensor 16) -> Ausgabe der ADC-Werte ohne
     Komma am Schluss
116             sprintf(ausgabe, "%04d,%04d", adc_value, adc_value1);
117             uart_print_string(ausgabe);
118             }
119         }
120         // Line Feed Character als Abschluss der uebertragung
121         uart_print_string("\n");
122 
123         ////////////////////////////////////////////////////////////
124         //  Es wurden nun alle 16 Sensoren der Platine ausgelesen.
125         //  folgendes, mit den der Bezeichnung entsprechenden Werte, wurde
    an den Laborrechner ausgegeben:
126         //
127         //  "cui_winkel,sensor1_sin,sensor1_cos,sensor2_sin,sensor2_cos,...
    sensor16_sin,sensor16_cos\n"
128         ////////////////////////////////////////////////////////////
129 
130         // Bevor die Routine erneut gestartet wird, wird der Watchdog auf
    60 Sekunden zurueckgesetzt
131         WatchdogReloadSet(WATCHDOG0_BASE,(uint32_t) 60 * 40000000);
132 
133         // Delay als Puffer zwischen Auslesungen
134         delay(500); // delay in micros
135     }
136 
137   return 0;
138 }
139 
140 
141 
142 
143 // ********************************************** Funktionen
144 
145 int init_uC(void)
146 {
147     // Clock Setup
```

```
148     sysfreq = SysCtlClockFreqSet( SYSCTL_USE_PLL | SYSCTL_OSC_INT |
        SYSCTL_CFG_VCO_320, 40000000); // 40 MHz Clock Setup
149
150
151     // Port A fuer UART -> genaueres in den jeweiligen Init-Funktionen
152     SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
153     while (!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOA))
154     {
155         ;
156     }
157     // Port C fuer GPIO -> genaueres in den jeweiligen Init-Funktionen
158     SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC);
159     while (!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOC))
160    {
161        ;
162    }
163     // Port E fuer GPIO und ADC Funktionalitaet -> genaueres in den
        jeweiligen Init-Funktionen
164     SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
165     while (!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOE))
166     {
167         ;
168     }
169     // For Testing
170     SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
171     while (!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOB))
172     {
173         ;
174     }
175     GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_4);
176
177
178     //
        ------------------------------------------------------------------------


179     // Enable Watchdog Timer
180     SysCtlPeripheralEnable(SYSCTL_PERIPH_WDOG0);
181    while (!SysCtlPeripheralReady(SYSCTL_PERIPH_WDOG0))
182    {
183        ;
184    }
185    // Erstes setzen des Watchdog mit Timer Countdown von 10 sec ( 400 000
       000 Clockcycles)
186    // Wird anschliessend am Ende jeder Routine zum Auslesen zurueckgesetzt
```

```
187     WatchdogResetEnable(WATCHDOG0_BASE);
188     WatchdogReloadSet(WATCHDOG0_BASE, 400000000);
189     WatchdogEnable(WATCHDOG0_BASE);
190
191     return 0;
192 }
193
194 int setup_switch_gpio(void) {
195     // Initalisierung der GPIOs. Port C Pin 4,5,6 und Port E Pin 4 fuer
      Selectsignal. Port E Pin 5 fuer Enablesignal
196     GPIOPinTypeGPIOOutput(GPIO_PORTC_BASE, GPIO_PIN_4 | GPIO_PIN_5 |
      GPIO_PIN_6);
197     GPIOPinTypeGPIOOutput(GPIO_PORTE_BASE, GPIO_PIN_4);
198     GPIOPinTypeGPIOOutput(GPIO_PORTE_BASE, GPIO_PIN_5);
199     // Test GPIO
200     GPIOPinTypeGPIOOutput(GPIO_PORTC_BASE, GPIO_PIN_7);
201     return 0;
202 }
203
204 int setup_adc(void) {
205     // Funktion zur Initialisierung der ADC-Module
206     GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_3 | GPIO_PIN_2);
207     GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_0 | GPIO_PIN_1);
208     // ..
209     SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
210     // ..
211     while (!SysCtlPeripheralReady(SYSCTL_PERIPH_ADC0))
212     {
213     }
214     SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC1);
215     while (!SysCtlPeripheralReady(SYSCTL_PERIPH_ADC1))
216     {
217     }
218     // ADCs werden differentiel Konfiguriert mit 64x Oversampling (
      Oversampling erhoehte in Tests die Genauigkeit deutlich messbar)
219     ADCSequenceConfigure(ADC0_BASE, 0, ADC_TRIGGER_PROCESSOR, 0);
220     ADCSequenceStepConfigure(ADC0_BASE, 0, 0, ADC_CTL_D | ADC_CTL_END |
      ADC_CTL_IE | ADC_CTL_CH0);
221     ADCSequenceEnable(ADC0_BASE, 0);
222     ADCHardwareOversampleConfigure(ADC0_BASE, 64);
223
224     ADCSequenceConfigure(ADC1_BASE, 0, ADC_TRIGGER_PROCESSOR, 0);
225     ADCSequenceStepConfigure(ADC1_BASE, 0, 0, ADC_CTL_D | ADC_CTL_END |
      ADC_CTL_IE | ADC_CTL_CH1);
```

```
226     ADCSequenceEnable(ADC1_BASE, 0);
227     ADCHardwareOversampleConfigure(ADC1_BASE, 64);
228     return 0;
229 }
230
231 int setup_uart(void)
232 {
233     // Funktion zur Initialisierung der UART-Module zur Kommunikation mit
        Laborrechner und CUI-Winkelgeber
234
235     // Reset der Uart-Module
236     SysCtlPeripheralReset(SYSCTL_PERIPH_UART0);
237     SysCtlPeripheralReset(SYSCTL_PERIPH_UART3);
238     // Initialisierung des UART-Modul UART0 zur Kommunikation mit
        Laborrechner
239     SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
240     while (!SysCtlPeripheralReady(SYSCTL_PERIPH_UART0))
241     {
242         ;
243     }
244     // Initialisierung des UART-Modul UART3 zur Kommunikation mit CUI-
        Winkelgeber
245     SysCtlPeripheralEnable(SYSCTL_PERIPH_UART3);
246     while (!SysCtlPeripheralReady(SYSCTL_PERIPH_UART3))
247     {
248         ;
249     }
250     // ...
251     GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);
252     GPIOPinConfigure(GPIO_PA0_U0RX);
253     GPIOPinConfigure(GPIO_PA1_U0TX);
254     // ...
255     GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_4 | GPIO_PIN_5);
256     GPIOPinConfigure(GPIO_PA4_U3RX);
257     GPIOPinConfigure(GPIO_PA5_U3TX);
258
259     // UART-Setup UART0 (Laborrechner), BAUD-Rate 1 MHz, 8-Bit kein Parity,
        1 StopBit
260     UARTConfigSetExpClk(UART0_BASE, sysfreq, 1000000, (UART_CONFIG_PAR_NONE
        | UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE));
261
262     // UART-Setup UART0 (Laborrechner), BAUD-Rate 2 MHz, 8-Bit kein Parity,
        1 StopBit
```

```
263     UARTConfigSetExpClk(UART3_BASE, sysfreq, 2000000, (UART_CONFIG_PAR_NONE
        | UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE));
264
265     UARTEnable(UART0_BASE);
266     UARTEnable(UART3_BASE);
267     return 0;
268 }
269
270 int uart_print_string(char* string) {
271     // TX-Funktion zur Kommunikation mit dem Laborrechner
272     for (var = 0; var < strlen(string); ++var) {
273         while(UARTBusy(UART0_BASE));
274         UARTCharPut(UART0_BASE, string[var]);
275     }
276     // UARTCharPut(UART0_BASE, '\0');
277     return 0;
278 }
279
280 int trigger_ADC_conversions_and_read_adcs(void) {
281     // Diese Funktion triggert gleichzeitig die ADC-Sequencen fuer beide
        differentiellen ADCs ADC0 und ADC1
282     // und liest diese anschliessend nacheinander in zwei globale Variablen
        aus
283     ADCProcessorTrigger(ADC0_BASE, 0 | ADC_TRIGGER_WAIT);
284     ADCProcessorTrigger(ADC1_BASE, 0 | ADC_TRIGGER_SIGNAL);
285     while(!ADCIntStatus(ADC0_BASE, 0, false))
286     {
287         ;
288     }
289     while(!ADCIntStatus(ADC1_BASE, 0, false))
290     {
291         ;
292     }
293     ADCSequenceDataGet(ADC0_BASE, 0, &adc_value);
294     ADCSequenceDataGet(ADC1_BASE, 0, &adc_value1);
295     return 0;
296 }
297
298 int get_ADC0(void) {
299     // Funktion zum einzelnen Auslesen des differentiellen ADC0 (Sinus-
        Signal)
300     ADCProcessorTrigger(ADC0_BASE, 0);
301     while(!ADCIntStatus(ADC0_BASE, 0, false))
302     {
```

```
303         ;
304     }
305     ADCSequenceDataGet(ADC0_BASE, 0, &adc_value);
306     return 0;
307 }
308
309 int get_ADC1(void) {
310     // Funktion zum einzelnen Auslesen des differentiellen ADC0 (Cosinus-
        Signal)
311     ADCProcessorTrigger(ADC1_BASE, 0);
312     while(!ADCIntStatus(ADC1_BASE, 0, false))
313     {
314         ;
315     }
316     ADCSequenceDataGet(ADC1_BASE, 0, &adc_value1);
317     return 0;
318 }
319
320 int switch_select(uint8_t select) {
321     // Funktion setzt Selectleitungen des zentralen Mux (Auswahl des
        auszulesenden Sensors mit dieser Funktion moeglich)
322     GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_4, select<<4);
323     GPIOPinWrite(GPIO_PORTC_BASE, GPIO_PIN_4 | GPIO_PIN_5 | GPIO_PIN_6 ,
        select<<3);
324     return 0;
325 }
326
327 int enable_mux(void){
328     // Funktion schaltet zentralen Mux an
329     GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_5, 0x0);
330     return 0;
331 }
332
333 int disable_mux(void){
334     // Funktion schaltet zentralen Mux aus ( Auslesen der Platine nicht
        mehr moeglich )
335     GPIOPinWrite(GPIO_PORTE_BASE, GPIO_PIN_5, 0x05);
336     return 0;
337 }
338
339 int cui_rx(char* cui_msg) {
340     // Diese Funktion empfaengt die Antwort des CUI-Winkelgebers auf die
        Aufforderung einen absoluten Winkel auszugeben (TX = Moduladresse)
```

```
341     // 16 Bit Wert wird empfangen -> unterste 14 Bit sind der absolute
        Winkel & obere 2 Bit sind eine Checksumme
342     for(var = 0; var < 2; var++){
343     cui_msg[var] = UARTCharGet(UART3_BASE); // 16 Bit empfangen
344     }
345     cui_msg_tot = (uint16_t)cui_msg[0] | ((uint16_t)cui_msg[1] << 8); //
        Empfangene 2 Byte zusammenfuegen
346     cui_angle = cui_msg_tot & 0b0011111111111111; // Winkel durch Maske
        einlesen
347     cui_chksum =  (cui_msg_tot & 0b1100000000000000)>>14; // Checksumme
        durch Maske einlesen und an die Stelle LSB schieben
348
349    return 0;
350 }
351
352
353 int delay(int micros) {
354     // Funktion zur Verzoegerung des Programmablaufs in Mikrosekunden
355     SysCtlDelay((sysfreq/1000000)*(micros/3)); // Fuehrt fuer die
        uebergebene Anzahl (int) jeweils 3 Dummy-Rechenschritte aus
356     return 0;
357 }
358
359 ////////////////////////////////////////////////////
360 // Theoretisch Steuerung des TRINAMIC-Schrittmotortreibers (TMCM-140-42-SE)
       ueber UART -> RS485 moeglich
361
362
363 /*
364 int setup_stepper(void){
365     // current limiting max
366         UARTCharPut(UART3_BASE, 1); // module adress
367         UARTCharPut(UART3_BASE, 5); // command (setup axis)
368         UARTCharPut(UART3_BASE, 6); // type number (absolute maximum current
        )
369         UARTCharPut(UART3_BASE, 0); // motor or bank number
370         UARTCharPut(UART3_BASE, 0); // value byte 1 ( MSB first )
371         UARTCharPut(UART3_BASE, 0); //  value byte 2
372         UARTCharPut(UART3_BASE, 0); // value byte 3
373         UARTCharPut(UART3_BASE, 35); // value byte 4 (35 ~ 380 mA)
374         UARTCharPut(UART3_BASE, 47); // checksum
375     //
376         stepper_rx(msg);
377     // power down delay
```

```
378         UARTCharPut(UART3_BASE, 1); // module adress
379         UARTCharPut(UART3_BASE, 5); // command (setup axis)
380         UARTCharPut(UART3_BASE, 214); // type number (power down delay)
381         UARTCharPut(UART3_BASE, 0); // motor or bank number
382         UARTCharPut(UART3_BASE, 0); // value byte 1 ( MSB first )
383         UARTCharPut(UART3_BASE, 0); //  value byte 2
384         UARTCharPut(UART3_BASE, 0); // value byte 3
385         UARTCharPut(UART3_BASE, 10); // value byte 4 (10 , 10 ms)
386         UARTCharPut(UART3_BASE, 230); // checksum
387      //
388
389         stepper_rx(msg);
390    // current limiting standby
391       UARTCharPut(UART3_BASE, 1); // module adress
392       UARTCharPut(UART3_BASE, 5); // command (setup axis)
393       UARTCharPut(UART3_BASE, 7); // type number (standby current)
394       UARTCharPut(UART3_BASE, 0); // motor or bank number
395       UARTCharPut(UART3_BASE, 0); // value byte 1 ( MSB first )
396       UARTCharPut(UART3_BASE, 0); //  value byte 2
397       UARTCharPut(UART3_BASE, 0); // value byte 3
398       UARTCharPut(UART3_BASE, 0); // value byte 4 (0, no current)
399       UARTCharPut(UART3_BASE, 13); // checksum
400   //
401       stepper_rx(msg);
402     // microstep resolution 0 ( Fullstep )
403        UARTCharPut(UART3_BASE, 1); // module adress
404        UARTCharPut(UART3_BASE, 5); // command (setup axis)
405        UARTCharPut(UART3_BASE, 140); // type number (standby current)
406        UARTCharPut(UART3_BASE, 0); // motor or bank number
407        UARTCharPut(UART3_BASE, 0); // value byte 1 ( MSB first )
408        UARTCharPut(UART3_BASE, 0); //  value byte 2
409        UARTCharPut(UART3_BASE, 0); // value byte 3
410        UARTCharPut(UART3_BASE, 0); // value byte 4 (0, no current)
411        UARTCharPut(UART3_BASE, 146); // checksum
412     //
413
414       stepper_rx(msg);
415
416 }
417
418 int stepper_rx(char* msg) {
419     for(var = 0; var < 9; var++){
420     msg[var] = UARTCharGet(UART3_BASE);
421     }
```

```
422     return 0;
423 }
424
425 int motor_step(int steps, int rot){
426     if(rot == 0){
427         steps = -steps;
428     }
429     uint8_t steps1 = 0xFF & steps;
430     uint8_t steps2 = 0xFF & steps>>8;
431     uint8_t steps3 = 0xFF & steps>>16;
432     uint8_t steps4 = 0xFF & steps>>24;
433     int sum_steps = steps1 + steps2 + steps3 + steps4;
434     // Motor dreht "steps" Schritte
435         UARTCharPut(UART3_BASE, 1); // module adress
436         UARTCharPut(UART3_BASE, 4); // command
437         UARTCharPut(UART3_BASE, 1); // type number
438         UARTCharPut(UART3_BASE, 0); // motor or bank number
439         UARTCharPut(UART3_BASE, 0xFF & steps>>24); // value byte 1 ( MSB
    first )
440         UARTCharPut(UART3_BASE, 0xFF & steps>>16); //  value byte 2
441         UARTCharPut(UART3_BASE, 0xFF & steps>>8); // value byte 3
442         UARTCharPut(UART3_BASE, 0xFF & steps); // value byte 4
443         UARTCharPut(UART3_BASE, sum_steps+6); // checksum
444     //
445         stepper_rx(msg);
446     while(position_reached == 0) {
447     //
448             UARTCharPut(UART3_BASE, 1); // module adress
449             UARTCharPut(UART3_BASE, 6); // command
450             UARTCharPut(UART3_BASE, 8); // type number
451             UARTCharPut(UART3_BASE, 0); // motor or bank number
452             UARTCharPut(UART3_BASE, 0); // value byte 1 ( MSB first )
453             UARTCharPut(UART3_BASE, 0); //  value byte 2
454             UARTCharPut(UART3_BASE, 0); // value byte 3
455             UARTCharPut(UART3_BASE, 0); // value byte 4
456             UARTCharPut(UART3_BASE, 15); // checksum
457       //
458             stepper_rx(msg);
459             if(msg[7] == 1){
460                 position_reached = 1;
461             }
462     }
463     position_reached = 0;
464
```

```
465 }
466
467 */
```

A.3.2 Matlab-Code

Listing A.1: Sample code from Matlab

```
classdef Sensor_reader
    properties
        % create dictionaries to save angle data for faster access
        dic_fft_angles = containers.Map;
        dic_cui_angles = containers.Map;
        %array to store calibration error
        arr_calibration_error = zeros(2,200)
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        % create variables for serial port objects
        ser_pcb
        ser_step
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    end
    methods
        function obj = Sensor_reader(ComSTEP,ComPCB)
            % Constructor method. Opens serial Ports for stepper and uc if
            % 2 arguments are passed.
            % Initializes the stepper motor and sends first msg to uC.
            switch nargin
                case 2
                    obj.ser_step = serialport(ComSTEP,9600);
                    obj.ser_pcb = serialport(ComPCB,1000000);
                    obj.ser_pcb.Timeout = 10;

                    %current limit
                    elements = [1,5,6,0,0,0,0,150,47];
                    write(obj.ser_step,elements,uint8);
                    test = read(obj.ser_step,9,uint8);

```

```
30                  %power down delay
31                  elements = [1,5,214,0,0,0,0,10,230];
32                  write(obj.ser_step,elements,uint8);
33                  test = read(obj.ser_step,9,uint8);
34
35                  %power down delay
36                  elements = [1,5,7,0,0,0,0,0,13];
37                  write(obj.ser_step,elements,uint8);
38                  test = read(obj.ser_step,9,uint8);
39
40                  %microstep resolution 0
41                  elements = [1,5,140,0,0,0,0,0,146];
42                  write(obj.ser_step,elements,uint8);
43                  test = read(obj.ser_step,9,uint8);
44
45              case 1
46
47                  obj.ser_pcb = serialport(ComSTEP,1000000);
48                  obj.ser_pcb.Timeout = 10;
49
50              case 0
51              otherwise
52          end
53      end
54
55      function step(obj,dir)
56          % steps the stepper motor one fullstep in the direction
57          % specified by dir (left,right)
58          if dir == right
59              elements = [1,4,1,0,0,0,0,1,7];
60              write(obj.ser_step,elements,uint8)
61              read(obj.ser_step,9,uint8);
62              write(obj.ser_step,elements,uint8)
63              read(obj.ser_step,9,uint8);
64          elseif dir == left
65              elements = [1,4,1,0,255,255,255,255,2];
```

```
66              write(obj.ser_step,elements,uint8)
67              read(obj.ser_step,9,uint8);
68              write(obj.ser_step,elements,uint8)
69              read(obj.ser_step,9,uint8);
70           end
71       end
72
73       function [sens_data,cui_angle] = read_sensdata_pcb_and_cui_angle(
             obj)
74           % Reads sin and cos data for all 16 sensors and the cui angle
75           % as ASCII Data. returns 1x16 complex array and cui angle in
                 Degree.
76           % Format: cui_angle,sin1, cos1, sin2, cos2 .... sin15, cos15
                 n
77           for cnt  = 1:2
78               flush(obj.ser_pcb,input) % flush input buffer
79               pause(0.001)
80               write(obj.ser_pcb,0,uint8); % command  to uC to prepare
                     next conversion
81               while(obj.ser_pcb.NumBytesAvailable<166)
82               end
83               data = readline(obj.ser_pcb);
84           end
85           data = str2num(strrep(data,,, ));
86           cui_angle = data(1) * (360.0/16384.0);
87           data(1) = []; % remove first element that was cui angle
88           % even index values = imag data
89           real_val = data(2:2:end) -2048.0;
90           % odd index values = real data
91           imag_val = data(1:2:end) -2048.0;
92           sens_data = (real_val + 1i*imag_val);
93       end
94
95       function [pcb_angles,cui_angle] = read_angle_pcb_and_cui_angle(obj
             , mean_cnt)
96           % Reads sin and cos data for all 16 sensors and the cui angle
```

```
97            % as ASCII Data. returns pcb_angle calculated by complex fft
                  in Degree and cui angle in Degree.
98            pcb_angles_array = []
99            for n = 1:mean_cnt
100               [sens_data,cui_angle] = read_sensdata_pcb_and_cui_angle(
                      obj);
101               pcb_angles = angle(fft(sens_data')) .* 180/pi;
102               pcb_angles_array(n,:) = pcb_angles(2);
103           end
104           pcb_angles = mean(pcb_angles_array);
105       end
106
107       function save_sensdata_pcb_and_cui_angle(obj,index,csv_name,
              mean_cnt)
108           % Saves mean Sensor Data and cui angle in a new line of a csv
                  table
109           % specified by csv_name. averages sensor data over mean_cnt
110           % measurements. indexes new line with index
111           % csv line format: index,[cos1+sin1i cos2+sin2i ...
                  cos15+sin15i],pcb_angle,cui_angle
112           sens_data_array_imag = [];
113           sens_data_array_real = [];
114           sens_data = [];
115           %cui_angle_sum = 0;
116           %cui_angle_last = 0;
117           %[sens_data,cui_angle] = obj.read_sensdata_pcb_and_cui_angle()
                  ; %
118           for n = 1:mean_cnt
119               [sens_data,cui_angle] = obj.
                      read_sensdata_pcb_and_cui_angle();
120               sens_data_array_imag(n,:) = imag(sens_data);
121               sens_data_array_real(n,:) = real(sens_data);
122               flush(obj.ser_pcb)
123           end
124           sens_data_imag = mean(sens_data_array_imag,1);
125           sens_data_real = mean(sens_data_array_real,1);
```

```
126             sens_data = sens_data_real + 1i* sens_data_imag;
127             %cui_angle = cui_angle_sum / mean;
128             pcb_angles = angle(fft(sens_data));
129             pcb_angles = pcb_angles(2) * 180/pi;
130             sens_data_cell = strcat([, num2str(sens_data,10),]);
131             write_data ={index, sens_data_cell, pcb_angles , cui_angle};
132             writecell(write_data,csv_name,'WriteMode','append','Delimiter'
                    ,',')
133         end
134
135         function  zero_cui(obj)
136             % steps the stepper motor until the cui angle overflows from
137             % 360 to 0
138             cui_angle_last = 0;
139             cui_angle = 0;
140             while(not(cui_angle_last - cui_angle > 300))
141                 cui_angle_last = cui_angle;
142                 obj.step(right)
143                 pause(200/1000) % Pause 200 ms
144                 [sens_data,cui_angle] = obj.
                        read_sensdata_pcb_and_cui_angle();
145             end
146         end
147
148         function start_measurement(obj,csv_root, rotations, mean)
149             % zeros the cui angle and then saves measurements for
150             % rotations amount of rotations in csv files while averaging
151             % the sensor data mean amount of times.
152             % csv format: csv_rootrotation.csv
153
154             %obj.zero_cui()
155             for n = 0:rotations
156                 csv_name = strcat(csv_root,num2str(n),.csv);
157                 if isfile(csv_name)
158                 else
159                     for i = 0:199
```

```
160                     pause(0.1) % Pause 100 ms
161                     obj.save_sensdata_pcb_and_cui_angle(i,csv_name,
                            mean)
162                     obj.step(right)
163                 end
164             end
165         end
166     end
167
168     function sens_data = read_sensdata_csv(~,csv_name)
169         % reads sensor data from csv and returns 200x16 complex vector
                .
170         % (200 measurements for 16 sensors)
171         values = readtable(csv_name,'Delimiter',',');
172         values_array = table2array(values(:,2));
173         sens_data_str = regexprep(values_array,'[[] ]','');
174         sens_data_str = strrep(sens_data_str,j,j );
175         sens_data_str = strrep(sens_data_str,i,j );
176         sens_data = zeros(length(sens_data_str),16);
177         for num = 1:length(sens_data_str)
178             sens_data(num,:) = str2num(sens_data_str(num)) .* -1j;
179         end
180     end
181
182     function angles_cui = read_cuidata_csv(obj,csv_name)
183         if obj.dic_cui_angles.isKey(csv_name)
184             angles_cui = obj.dic_cui_angles(csv_name);
185         else
186             values = readtable(csv_name,'Delimiter',',');
187             angles_cui = table2array(values(:,4));
188             obj.dic_cui_angles(csv_name) = angles_cui;
189         end
190     end
191
192     function fft_angles = calc_angles_fft_complex(obj,csv_name)
193         % reads sensor data from csv and returns 16x200 angle data for
```

```
194             % Frequenzies k= 0 to 15
195             if obj.dic_fft_angles.isKey(csv_name)
196                 fft_angles = obj.dic_fft_angles(csv_name);
197             else
198                 values = read_sensdata_csv(obj,csv_name);
199                 fft_angles = angle(fft(values')) .* 180/pi;
200                 obj.dic_fft_angles(csv_name) = fft_angles;
201                 %fft_angles = atan2(imag(fftshift(fft(values'))),real(
                        fftshift(fft(values')))) .* 180/pi;
202             end
203         end
204
205         function err = calc_array_err_fft_complex(obj,csv_name)
206             angles_fft = calc_angles_fft_complex(obj,csv_name);
207             for freq = [2 16]
208                 angles_fft = angles_fft(freq,:)'; % index 2 = Frequenz 1
                        ( k=1 )
209                 angles_cui = read_cuidata_csv(obj,csv_name);
210                 %angles_cui = unwrap(angles\_cui,180);
211                 %angles_fft = unwrap(angles\_fft,180);
212                 for x = 2:length(angles_fft)
213                     if (angles_cui(x-1) -angles_cui(x)) < -300
214                         angles_cui(x) = angles_cui(x) - 360.0;
215                     elseif (angles_cui(x-1) -angles_cui(x)) > 300
216                         angles_cui(x) = angles_cui(x) + 360.0;
217                     end
218                     if (angles_fft(x-1) - angles_fft(x)) < -300
219                         angles_fft(x) = angles_fft(x) - 360.0;
220                     elseif (angles_fft(x-1) - angles_fft(x)) > 300
221                         angles_fft(x) = angles_fft(x) + 360.0;
222                     end
223                 end
224
225                 diff = angles_fft(1) - angles_cui(1);
226                 angles_fft = angles_fft - diff;
227                 if freq == 2
```

```
228                     err_temp(1,:) = angles_fft - angles_cui;
229                 else
230                     err_temp(2,:) = angles_fft - angles_cui;
231                 end
232             end
233 %             if (abs(max(err_temp(1,:))))>(abs(max(err_temp(2,:))))
234 %                 err = err_temp(2,:);
235 %             else
236 %                 err = err_temp(1,:);
237 %             end
238                 err = err_temp(1,:);
239         end
240 
241         function max_diffs = calc_error_diffs_between_measurements(obj,
                csv_root,n)
242             % return the highest difference between angle errors for full
243             % rotations.
244             % takes root path and concats x.csv where x is an incremented
                    number
245             % from 0 to n.
246             for cnt = 1:n+1
247                 filename = strcat(csv_root,int2str(cnt-1),.csv);
248                 read_data(cnt,:) = calc_array_err_fft_complex(obj,filename
                        );
249             end
250             diffs = read_data';
251             for x = 1:length(diffs)
252                 max_diffs(x) = max(diffs(x,:)) - min(diffs(x,:));
253             end
254         end
255 
256     end
257 end
```

Erklärung zur selbstständigen 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