HAW HAMBURG

Bachelorthesis

Viktor Airich

Charakterisierung magnetoresistiver Sensor-Arrays mittels eines automatisierten Messsystems

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

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

Viktor Airich

Charakterisierung magnetoresistiver Sensor-Arrays mittels eines automatisierten Messsystems

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung
im Studiengang Informations- und Elektrotechnik
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. Klaus Jünemann

Abgegeben am 10. Januar 2018

Viktor Airich

Thema der Masterarbeit

Charakterisierung magnetoresistiver Sensor-Arrays mittels eines automatisierten Messsystems

Stichworte

AMR-Effekt, TMR-Effekt, Sensor-Array, Hysterese, Sättigung, Störfeld, Messsystem, Steuerungssoftware, Rotation, Translation, Ablagen, Sicherheitssystem, Piezoelement, Darstellungssoftware

Kurzzusammenfassung

In dieser Arbeit wird ein Prüfstand mit automatisierter Positionssteuerung für die Benutzung mit Demonstratoren magnetoresistiver Sensor-Arrays in Betrieb genommen. Dabei wird die Steuerungssoftware modifiziert. Für die Befestigung der Magnetsensoren und Permanentmagneten werden Adaptermodule konstruiert und in 3D-Druck gefertigt. Es wird eine Datenstruktur und eine Vorlage für ein Messprotokoll erstellt. Um das Positioniersystems vor Beschädigungen zu schützen, wird ein Sicherheitssystem für eine Notabschaltung des Messplatzes entworfen.

Viktor Airich

Title of the master thesis

Characterization of magnetoresistive sensor-arrays using an automated measuring system

Keywords

AMR effect, TMR effect, sensor array, hysteresis, saturation, interference field, measurement system, control software, rotation, translation, deposition, safety system, piezo element, presentation software

Abstract

In this work is a test bench with automated position control for demonstrators of Magnetoresistive Array-Sensors. It is put into operation. The control software will be modified. For the fixture of magnetic sensors and permanent magnets adapter modules are constructed and printed in 3D. A datastructure and a template for the measurement protocol is designed. To avoid damage of the positioning system a safety system for an emergency shutdown is developed.

Vorwort

Ich möchte mich an dieser Stelle bei Herrn Prof. Dr.-Ing. Karl-Ragmar Riemschneider, betreuender Professor, dafür bedanken, dass mir das interessante Thema zum Bearbeiten geboten wurde.

Des Weiteren Danke ich Herr Prof. Dr. Klaus Jünemann für die Übernahme der Zweitprüfung.

Weterhin bedanke ich mich bei Herrn M.Sc. Thorben Schüthe für seinen fachlichen Rat und seine wirksame Unterstützung.

Vielen Dank an Herrn Dipl.-Ing. Günter Müller, für die Übernahme der Korrektur meiner Arbeit.

Ebenso bedanke ich mich bei den Forschungsgruppen ISAR und BATSEN für die kollegiale und angenehme Arbeitsklima.

Ein besonderer Dank geht an meine Frau Olga Airich und meine Kinder Roman und Nicole, meine Eltern Alexander und Svetlana Airich, welche mich während des gesamten Studiums unterstützt haben.

Inhaltsverzeichnis

Abbildungsverzeichnis **VIII**

Tabellenverzeichnis **IX**

Abkürzungen **X**

Symbolverzeichnis **XI**

1 Einleitung **1**
1.1 Stand der Technik 1
1.2 Ziel und Aufbau dieser Thesis 2

2 Grundlagen **3**
2.1 Magnetische Sensoren 3
2.1.1 AMR-Sensor 3
2.1.2 TMR-Sensor 5
2.2 Funktionsweise des AMR-Sensor-Arrays 6

3 Software **8**
3.1 Struktur der Software 8
3.2 Dokumentation für Messwerteaufzeichnung 9
3.3 Steuerungsablauf 11
3.3.1 Messsteuerung der Versuchsabläufe im vollautomatischen Modus . 12
3.3.2 Messsteuerung der Versuchsabläufe im halbautomatischen Modus 13
3.4 Ansteuerung bzw. Auslesen der Sensormatrix und der Einzelsensoren . . 13

4 Hardware und Messplatz **14**
4.1 Koordinatensystem des Messplatzes 14
4.2 Adaptermodule für die Permanentmagneten und die Sensorelektronik . . 16
4.3 Sicherheitssystem 16
4.3.1 Sensorauswahl 17
4.3.2 Systementwurf des Sicherheitssystems 18
4.3.3 Test der Kollisionssensoren 20
4.3.4 Auswertung der Kollisionstestergebnisse 21
4.4 Positionierung des Roboterarms 22

5 Datenerfassung und Visualisierung 25
5.1 Erstellung eines Messplanes 25
5.1.1 Messplan für einen vollautomatischen Messablauf 25
5.1.2 Messplan für einen halbautomatischen Messablauf 26
5.1.3 Messplan für die Untersuchung der Hysterese, einer Offset-Spannung und der Sättigung 27
5.2 Visualisierung der Messdaten 28

6 Auswertung der Messdaten 30
6.1 Datenerfassung in einem fehlerfreien und in einem fehlerhaften Betrieb . 30
6.1.1 Der fehlerfreie Betrieb 30
6.1.2 Der fehlerhafte Betrieb 31
6.1.3 Datenerfassung mit einem Störfeld 33
6.2 Charakteristische Effekte der Sensoren 34
6.2.1 Sättigung 34
6.2.2 Hysterese 35
6.2.3 Offset 38
6.3 Toleranz des Messsystems 39

7 Schlussfolgerungen 41
7.1 Zusammenfassung 41
7.2 Ausblick 42

Literatur 44

Anhang

A Messergebnisse und Zeichnungen aus den Kollisionstests 46

B Auswertung 49

C Quellcode 51

D CD 96

Selbstständigkeitserklärung 97

Abbildungsverzeichnis

2.1 AMR-Prinzip 4
2.2 Wheatstone'sche Messbrücken zur Messung elektronischer Widerstände und magnetoresistiver Widerstände beim AMR-Effekt. 5
2.3 Hardware-Aufbau für die Messerfassung 7

3.1 Struktogramm der Steuerung des Messplatzes 9
3.2 Das Koordinatensystem bezogen auf das Sensor-Array. 10

4.1 Koordinatensystem von Messplatz 15
4.2 Der Adapter für einen Magnetsensor und die Befestigung der unterschiedlichen Magnete. 16
4.3 Die getestete Sicherheitssensoren 17
4.4 Konstruktion des Sicherheitssystems für die Fertigung mittels 3D-Druck. 19
4.5 Aufbau für die Fehlpositionierungstests 20
4.6 Verformung der Platine mit der Hängewaage 22
4.7 Positionierung des Roboterarm über der Grundplatte 23
4.8 Die Gegenüberstellung der Extrema der Kollisionstestergebnisse 24

5.1 Schematische Darstellung der Messverlaufes. 26
5.2 Messaufbau für den Versuch mit einem Sensor. 27
5.3 Struktogram der Auswahl eines Darstellungsskriptes. 29

6.1 Der Vergleich von den Spannungquellen für die Versorgung 31
6.2 Die Reduzierung der Rauschen durch RC-Filter mit $\tau = 100\,\mu s$ 31
6.3 Positionierung eines Magneten ausserhalb des Sensor-Array 32
6.4 Messergebnisse ohne Sättigung 32
6.5 Messergebnisse für das angelegte Störfeld 33
6.6 Verformung des Messaufbaus bei dem Versuch mit einem Störfeld 34
6.7 Der Messverlauf mit Vor- und Rückwärtsrotation für die Bestimmung des Hysterese-Effektes. 36
6.8 Der Hysterese-Effekt eines AMR-Sensors (KMZ60) von der Firma NXP . 37
6.9 Der Hysterese-Effekt eines TMR-Sensors (ADT001) von der Firma NVE 37
6.10 Die Messergebnisse der Offset-Spannung 38
6.11 Die Abweichung des eingestellten Winkels φ_z von dem gemessenen Winkel α. 40

A.1 Darstellung der Kollisionserkennungstests 48

B.1 Messaufbau des Versuches mit einem Störfeld. 49

D.1 Ordnerstruktur der beigefügten CD. 96

Tabellenverzeichnis

4.1 Zusammenfassung aus dem Kollisionstest mit dem Piezo-Scheibe-System 21
4.2 Zusammenfassung aus dem Kollisionstest mit dem Mikrotaster-System . 22

A.1 Ergebnisse der Experimente mit dem Piezo-Scheibe-System 46
A.2 Ergebnisse der Experimente mit dem Mikrotaster-System 47

B.1 Die Änderung der magnetischen Feldstärke bei der Sättigungsuntersuchung 50

Abkürzungen

AMR	Anisotroper magnetoresistiver Effekt
CMR	kolossaler magnetoresistiver Effekt
GMR	Riesenmagnetowiderstand
HAW	Hochschule für Angewandte Wissenschaften
ISAR	Signalverarbeitung für **I**ntegrated **S**ensor-**AR**ray basierend auf dem Tunnel-Magnetoresistiven Effekt für den Einsatz in der Automobilelektronik
PLA	Polylactide
PSS	Piezo-Scheibe-System
TMR	Tunnel-Magnetoresistive Effekt
VS	Verbindungsschicht für das Sicherheitssystem

Symbolverzeichnis

Symbol	Einheit	Beschreibung
α	°	Winkel
$\mathbf{H}$	A/m	magnetische Feldstärke
$\mathbf{I}$	A/m^2	Stromdichte
$\mathbf{M}$	A/m	Magnetisierung
U_{off}	V	Offset-Spannung
U_{ref}	V	Referenzspannung des ADC
U_{vcc}	V	Versorgungsspannung
φ_{y}	°	Rotationswinkel der Y-Achse des Roboterarms
φ_{z}	°	Rotationswinkel der Z-Achse des Roboterarms

1 Einleitung

Diese Abschlussarbeit wird als ein Teil des Forschungsprojektes Signalverarbeitung für **I**ntegrated **S**ensor-**AR**ray basierend auf dem Tunnel-Magnetoresistiven Effekt für den Einsatz in der Automobilelektronik (ISAR) geschrieben. Das Projekt wird von der Hochschule für Angewandte Wissenschaften (HAW) Hamburg, einem Partner aus der Wirtschaft sowie der Ostfalia Hochschule für Angewandte Wissenschaften in Wolfenbüttel getragen. Dabei wird den Anwendungen der Tunnel-Magnetoresistive Effekt (TMR)-Technologie als Sensor-Array geforscht.

1.1 Stand der Technik

Heutzutage werden die auf allen Ebenen des Produktionsprozesses generierten Daten zur Verbesserung der Produktqualität, Flexibilität und Produktivität verwendet. Das wäre ohne intelligente Sensoren nicht möglich. Die Sensoren ermöglichen eine Selbstüberwachung, Selbstkonfiguration und eine Zustandsüberwachung der komplexen Prozesse. Im Vergleich zum Projekt Industrie 4.0 ist die Entwicklung von Sensoren auf die unterschiedlichen Entwicklungsphase gestuft. Sensor 4.0 oder die intelligenten Sensoren wird heutige Phase der Entwicklung genannt.

Seit etwa 90 Jahre sind die magnetoresistiven Effekte für die Sensorik von großem Interesse. Der Bereich entwickelt sich weltweit sehr dynamisch. Die magnetischen Sensoren werden bei der berührungslosen auch verschleißfreien Bestimmung einer Position oder Bewegung eingesetzt. Mit ihrer Hilfe wird magnetische, elektrische oder mechanische Information direkt in ein elektrisches Signal umgewandelt und kann dann mit heutigen Elektronik weiter bearbeitet werden. Der Einsatzbereich von XMR-Technologien ist breit gefächert. Der Begriff XMR-Effekt steht für X: alle MR: magnetoresistive-Effekt. MR-Effekte lassen sich in der wichtigen Gebiete unserer Lebens finden. Das sind u.a. Fahrzeugbau, Maschinenbau (Robotik), Informationstechnik, Medizintechnik, zerstörungsfreie Werkstoffprüfung, Mikrosystemtechnik, magnetische Massenspeicher [10].

In den modernen Fahrzeugen wächst der Anteil vom magnetoresistiven (MR)-Sensoren immer weiter, da sie ein berührungsloses und damit verschleißfreies Prinzip, relativ kleine Abmessungen sowie Robustheit und Stabilität anbieten. Knapp eine Hälfte von allen eingesetzten Sensoren in einem Oberklassefahrzeug kann durch MR-Sensoren ersetzt werden [18, S. 282].

1.2 Ziel und Aufbau dieser Thesis

Auf einem Prüfstand werden Magnetfelder von einzelnen Magnetsensoren als auch Sensor-Arrays untersucht und ausgewertet. Zunächst wird der Prüfstand mit automatisierter Positionssteuerung für die Benutzung mit Demonstratoren der Sensor-Array in die Betrieb genommen, wobei Steuerungssoftware, als Matlab-Skript zu modifizieren und zu testen ist. Weiterhin sind Hardware-Aufbauten mit handelsüblichen AMR- bzw. TMR-Magnetsensoren in den Prüfstand einzubringen. Im Rahmen der Arbeit werden folgende Sensoren getestet und gegenüber gestellt:

- AMR Modell KMZ60 von NXP
- TMR Modell ADT001 von NVE Corporation

Für die mechanische Fixierung der Permanentmagneten und der Sensorelektronik sind Adaptermodule zu konstruieren und im 3D-Druck zu fertigen. Bei der Datenerfassung sind Auslesewerte der Sensorik mit der Steuerung der Positionieraufgabe zu koppeln. Da die Messwerte zukünftig für eine Analyse zur Verfügung stehen sollen, ist eine geeignete Datenstruktur zu erstellen. Zusätzlich ist ein Sicherheitssystem zu entwickeln, das eine mechanische Kollision erkennt und eine Notabschaltung bewirkt. Damit sollen Fehlpositionierungen bei der Bedienung oder durch Softwarefehler erkannt werden und eine Beschädigung der wertvollen Elektronik und des Positioniersystems ausgeschlossen werden.

In der Einleitung im Kapitel 1 sind allgemeine Information über die Thematik und Ziele der Arbeit beschrieben.

Die Grundlagen, welche für das Verständnis der Abschlussarbeit notwendig sind, im Kapitel 2 beschrieben. Hierzu werden unter anderem solche Begriffe wie AMR- bzw. TMR-Effekte und Winkelsensoren erklärt.

Zum Kapitel 3 gehört die Beschreibung der Steuerungs- und Auswertungssoftware. Dabei sind eine Steuerungsarchitektur des Messplatzes, die Funktionen einer Messsteuerung und einer Dokumentationserstellung dargestellt und erläutert. Die Erstellung bzw. Modifizierung der Software wird als Matlab-Skript unter dem Linux Betriebssystem stattfinden und entsprechend angepasst.

Der Entwurf einer mechanischen Fixierung der Elektronik bzw. eines Magneten am Messplatz wird in der Kapitel 4 dargestellt und beschrieben. Die Befestigung der Elektronik wird ebenso im gleichen Kapitel erläutert. Hier wird auch die Entwicklung des Sicherheitssystems zum Abschalten des Messplatzes bei einer Berührung zwischen einem Magnet und einem Magnetsensor bzw. Sensor-Array erfasst. Mit dem Sicherheitssystem wird der Messplatz und die Elektronik vor mechanischen Kollision geschützt.

2 Grundlagen

2.1 Magnetische Sensoren

Sensoren, deren Funktion auf dem MR - Effekt beruht, bezeichnet man als Magnetosensoren - oder genauer Magnetowiderstandssensoren, da sich die elektrischen Widerstände des Sensors durch Anlegen eines äußeren Magnetfeldes verändern. Dieser Effekt wurde vor 160 Jahren von dem britischen Physiker William Thomson, später Lord Kelvin, entdeckt. Der industrielle Einsatz hat mit der Evolution der Dünnschichttechnik vor ca. 30 Jahren stattgefunden. Der MR -Effekt lässt sich durch Anordnung und Art der eingesetzten Materialien auf Folgende unterscheiden: Anisotroper magnetoresistiver Effekt (AMR), Riesenmagnetowiderstand (GMR), kolossaler magnetoresistiver Effekt (CMR), TMR sowie der Hall-Effekt. Die Sensoren, die auf dem Effekt basieren sind hochempfindlich und leistungsstark [18, S. 283].

2.1.1 AMR-Sensor

Der AMR-Effekt ist der am längsten bekannte MR-Effekt, der sich besonders gut in einer dünnen Schicht (im Nanometerbereich) aus Permalloy, einer Legierung aus Nickel und Eisen beobachten lässt. Der AMR-Effekt weist eine bis zu 200-fach kleinere Widerstandsänderung als der TMR-Effekt auf, trotzdem sind die AMR-Sensoren die meist eingesetzten Magnetosensoren [3].

Im Abbildung 2.1 ist eine schematische Darstellung des AMR-Effektes zu erkennen, wobei eine Permalloyschicht (NiFe), die von einem äußeren Magnetfeld $\boldsymbol{H}$ beeinflusst ist und die Magnetisierung $\boldsymbol{M}$ von der Stromrichtung $\boldsymbol{I}$ um Winkel α ablenkt. Zu beachten ist, dass bei Sättigung die Richtung von $\boldsymbol{H}$ und $\boldsymbol{M}$ gleich ist. Wenn die Sättigung nicht erreicht wird, wird die Magnetisierung $\boldsymbol{M}$ relativ zu dem äußeren Magnetfeld $\boldsymbol{H}$ nicht mehr vollständig folgen. Das bedeutet, dass die Richtung von $\boldsymbol{H}$ und $\boldsymbol{M}$ nicht mehr gleich ist, was zu den fehlerhaften Messwerten führt.

Der spezifische Widerstand der ferromagnetischen Materialien ρ ist vom Winkel α zwischen Stromrichtung $\boldsymbol{I}$ und den internen Magnetisierung $\boldsymbol{M}$ abhängig. Wenn der Winkel 90° beträgt, ist der Widerstand minimal und wird wie folgt ausgedrückt: $\rho = \rho_{\perp}$. Bei $\alpha = 0°$ ist ρ maximal ($\rho = \rho_{\parallel}$). Daraus ergibt sich:

$$\rho(\alpha) = \rho_{\perp} + (\rho_{\parallel} - \rho_{\perp}) \cdot \cos^2\alpha = \rho_{\perp} + \Delta\rho \cdot \cos^2\alpha \tag{2.1}$$

Der spezifische Widerstand ρ kann als Widerstand R betrachtet werden, wenn ein Strom I in Richtung l fließt, wobei die magnetische Schicht des Sensors aus Länge l, Breite

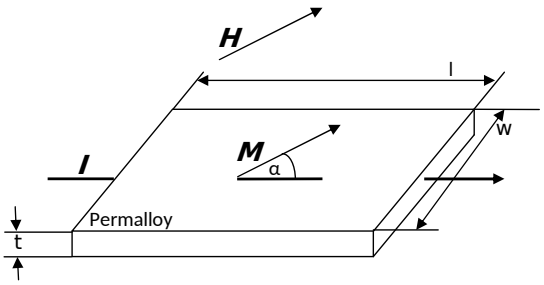


Abbildung 2.1: Die Änderung des Winkels α zwischen Strom $\boldsymbol{I}$ und der internen Magnetisierung $\boldsymbol{M}$, im Abhängigkeit vom äußeren Magnetfeld $\boldsymbol{H}$.

w und Dicke t besteht. Wenn die Richtung des Stromflusses mit der Längsrichtung übereinstimmt, kann der AMR-Effekt durch folgende Gleichung beschrieben werden:

$$R(\alpha) = R + \Delta R \cdot \cos^2\alpha \tag{2.2}$$

Bei der Wheatstone Messbrücke sind die Widerstände um 90° zueinander gedreht. Eine klassische Wheatstone Messbrücke ist in der Abbildung 2.2(a) dargestellt. Das bedeutet, dass der Widerstand ΔR, der vom Winkel α abhängig ist, sich in einem Fall proportional zu $\cos^2\alpha$ und beim anderen proportional zu $\sin^2\alpha$ ändert. Aus der Trigonometrie ist bekannt, dass $\cos 2\alpha = \cos^2\alpha - \sin^2\alpha$, was bedeutet, dass am Ausgang eine Spannung $U \sim \cos 2\alpha$ gemessen wird [6, S. 5].

Wenn zwei um 45° zueinander gedrehte Wheatstone Brücken geschaltet werden (Abbildung 2.2(b)), was auch in der AMR Winkelsensoren der Fall ist, entsteht beim Ausgang einer Brücke ein Signal proportional zu $\sin 2\alpha$ (Gleichung (2.3)). Bei der anderen Brücke entsteht ein Signal proportional zu $\cos 2\alpha$ (Gleichung (2.4)). Um den Winkel α zu berechnen, wird das cos-Signal durch das sin-Signal dividiert, wobei eine Offset-Spannung kompensiert ist (Gleichung (2.5)). Mathematisch lässt sich das wie folgt zusammenstellen:

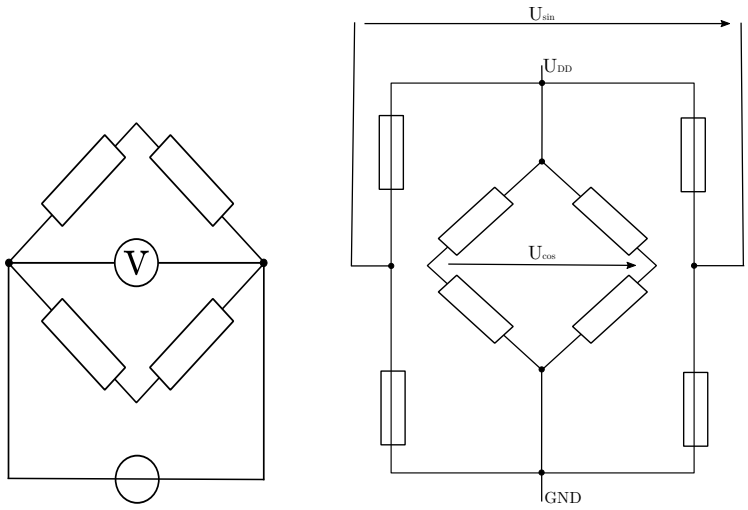
$$U_{sin}(\alpha) = U_{vcc} \cdot \sin 2\alpha + U_{off} \tag{2.3}$$

$$U_{cos}(\alpha) = U_{vcc} \cdot \cos 2\alpha + U_{off} \tag{2.4}$$

$$\tan\alpha = \frac{\Delta U_{sin}}{\Delta U_{cos}} \tag{2.5}$$

Wobei ΔU_{sin}, ΔU_{cos} kompensierte Ausgangssignale sind.

Die Tangens Funktion ist eine periodische Funktion. Die Periode bei $\tan\alpha$ bezüglich α beträgt 180°. Wenn man aber als Argument 2α betrachtet, gilt für die Periode der Tangens Funktion nun 90°. Deswegen ist es möglich, mit Gleichung (2.5) den Winkel α nur für eine ±45° Periode bezüglich des Nullpunktes zu bestimmen. Wenn die Information über das Vorzeichen von den gleichzeitig gemessenen Signalen ΔU_{sin}, ΔU_{cos} beachtet



(a) Klassische Wheatstone'sche Brücke.

(b) Zwei um 45° zueinander gedrehte Wheatstone'sche Messbrücken.

Abbildung 2.2: Wheatstone'sche Messbrücken zur Messung elektronischer Widerständeund (links) und magnetoresistiver Widerstände beim AMR-Effekt (rechts).

wird, ist es möglich, den Messbereich auf ±90° zu vergrößern. Die zwei phasenverschobenen harmonischen Schwingungen werden sich nach einer Periode von 180° wiederholen, was zur Folge hat, dass es nicht möglich ist, mit einem AMR-Winkelsensor einen Winkel α in einer Periode von 360° zu bestimmen [14].

2.1.2 TMR-Sensor

Der TMR-Effekt wurde von dem französischen Physiker M.Jullière an der Universität Rennes in Frankreich vor über 40 Jahren beschrieben. Dieser Effekt wird beobachtet, wenn mindestens drei nanostrukturierte Schichten aneinander gefügt sind, wobei zwei ferromagnetische Schichten einen dünnen Isolator einschließen. Diese Isolationsschicht ist so dünn, dass Strom fließen kann. Da die Elektronen durch das Schichtsystem tunneln, wird der Strom als Tunnelstrom bezeichnet, wobei er von der Orientierung der Magnetfelder der beiden Ferromagneten abhängig ist. Bei einer parallelen Ausrichtung der Dünnschichten ist er maximal und bei dem antiparallelen Verhalten erhält man einen maximalen Widerstand und den geringsten Strom. Die Information kann zur Abspeicherung der unterschiedlichen Widerstandszustände (binär) genutzt werden [19].

Um den spinabhängigen Elektronentransport in Tunnelmagnetowiderständen zu beschreiben, führte Jullière ein einfaches Modell ein. Dieses Modell basiert darauf, dass beim Tunnelprozess der Spin erhalten bleibt. Das bedeutet, dass z.B. die Majoritäts-Spin (spin-up), so werden Elektronen bezeichnet, die zu geringer energiehaltigen Gegenelektrode verschoben werden, nur in freie spin-up Zustände tunneln. Das gilt auch für die

Minoritäts-Spin. Das sind die Elektronen, die zu höheren Energiepotenzialen verschoben werden. Damit ist es möglich, die beiden Spinkanäle getrennt zu betrachten. Nach diesem Modell wird der TMR-Effekt wie folgt berechnet:

$$\mathrm{TMR} = \frac{G_p - G_{ap}}{G_{ap}} = \frac{R_{ap} - R_p}{R_p} = \frac{2P_l P_r}{1 - P_l P_r} \tag{2.6}$$

Wobei G_p — die Leitfähigkeit bei paralleler Magnetisierung ist und G_{ap} — entsprechend bei antiparalleler. Dementsprechend lassen sich die Magnetowiderstände R_p, R_{ap} bezeichnen. Der Zustand der magnetischen Schicht wird als Spinpolarisation P_i bezeichnet, wobei mit dem Index $i = l, r$ der spinabhängige Zustandsdichte an der Fermi-Kante sich unterscheiden lässt [11, S. 13] [9, S. 10].

Die TMR-Sensoren sind durch ihren Flächenwiederstand der Barriere sehr hochohmig, was auch von der Größe der Sensoren abhängig ist: je größer der Widerstand, desto kleiner das Sensorelement. Das führt zur Reduzierung des Leistungsverbrauchs und ermöglicht geringe Leistungsaufnahmen, was bei den AMR-Sensoren nicht der Fall ist. Ein weiterer Vorteil der TMR- gegenüber AMR-Sensoren ist das Verhalten im Drehfeld. Die Winkelsensoren sind nicht doppeldeutig und können die Drehung eines Gebermagneten über 360° detektieren, ohne weitere Berechnungen durchführen zu müssen [18, S. 290].

2.2 Funktionsweise des AMR-Sensor-Arrays

Das Sensor-Array wurde in der ISAR-Gruppe mit KMZ60 Sensoren von der Firma NXP bestückt. Als Verbindungsstelle zwischen den Sensor-Array und PC wird ein „Connected LaunchPad Evalution Kit EK-TM4C1294XL“ von der Firma Texas Instruments eingesetzt. Dabei wird ein TM4C1294 Mikrocontroller genutzt, der folgenden Eigenschaften besitzt [4]:

- 120 MHz 32-bit ARM Cortex-M4 CPU
- 1 MB Flash, 256KB SRAM, 6KB EEPROM
- 12-bit SAR ADC, mit zwei Kanälen

Dieser Mikrocontroller ist in der ISAR-Gruppe verbreitet. Die Verbindung zum Sensor-Array wird über SPI-Schnittstelle realisiert. Mit einem PC wird der Mikrocontroller durch ein USB-Kabel verbunden. Der Aufbau ist schematisch in der Abbildung 2.3 dargestellt.

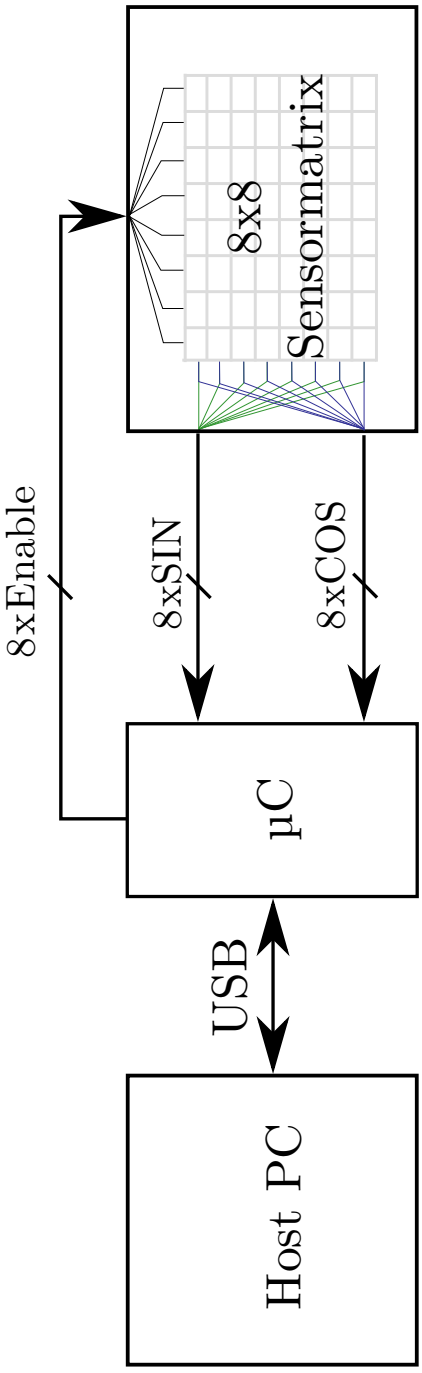


Abbildung 2.3: Die Verbindung zwischen PC über Mikrocontroller und Sensor-Array [16].

3 Software

Die Ansteuerung sowie Datenaufnahme erfolgt mittels Matlab. Für die Messverfahren mit den Sensor-Arrays werden neue Funktionen erstellt, die eine manuelle als auch automatisierte Vermessung der Einzelsensoren bzw. Sensor-Arrays ermöglichen. Die aufgenommenen Messdaten werden für die Darstellung und Analyse in einer festgelegten Struktur und mit einem Messprotokoll gespeichert. Für grundlegende Funktionalität werden die Steuerungsprogramme für den Messplatz aus der Abschlussarbeit [15, S. 106] herangezogen. Die wichtigsten Kriterien sind:

- Bedienerfreundliche Oberfläche der Ein- und Ausgabe-Information
- Variabilität bei der Auswahl der Messverfahren
- Nachvollziehbarkeit der Quellcodes für mögliche Optimierungsvorgänge

3.1 Struktur der Software

Um den Überblick der Struktur der Steuerung zu verschaffen, wurde in der Abbildung 3.1 dargestellte Diagramm entworfen.

Die Idee ist, dass ein Benutzer nur ein Skript starten muss, um einen Zugriff auf alle Einstellungen bzw. Steuerungspunkte haben zu können. Das heißt, nachdem die Verbindung mit dem Messplatz aufgebaut ist und die restlichen Initialisierungen (Referenzfahrt oder Übernahme von gespeicherten Werten für die Definierung aller Motoren) des Messplatzes durchgeführt sind, kann über das Hauptmenü (Ansteuerungsmodus) der Roboterarm entweder Schritt für Schritt oder mit Hilfe der gespeicherten Koordinaten positioniert werden. Es kann über das Messdatenaufnahmemenü Echtzeitverhalten der Sensor-Array für eine Analyse darstellen, eine Datenaufnahme initialisieren und starten oder ein Messprotokoll erstellen.

Die Form der Blöcke, die im Struktogramm (Abbildung 3.1) dargestellt sind, lässt sich nach Funktionalität zusammenfassen. Der sechseckige Block ist das erste Skript, womit eine Verbindung mit dem Messplatz aufgebaut und das Initialisierungsmenü aufgerufen wird. Alle ovalen Blöcke sind Menüs, die einen Zugriff auf die jeweils ausgewählte Funktion ermöglichen. Außerdem ist ein Abbruch des Ablaufs aus dem ovalen Block realisierbar. Die Beschreibungen der Funktionen sind in den Dreiecken als auch in den Rechtecken dargestellt. Der Unterschied zwischen den beiden Formen liegt darin, dass die Rechtecke zu der Initialisierungsphase gehören. Das bedeutet, dass die Ansteuerung des Messplatzes nicht möglich ist, wenn eine der beiden Funktion nicht ausgeführt wird.

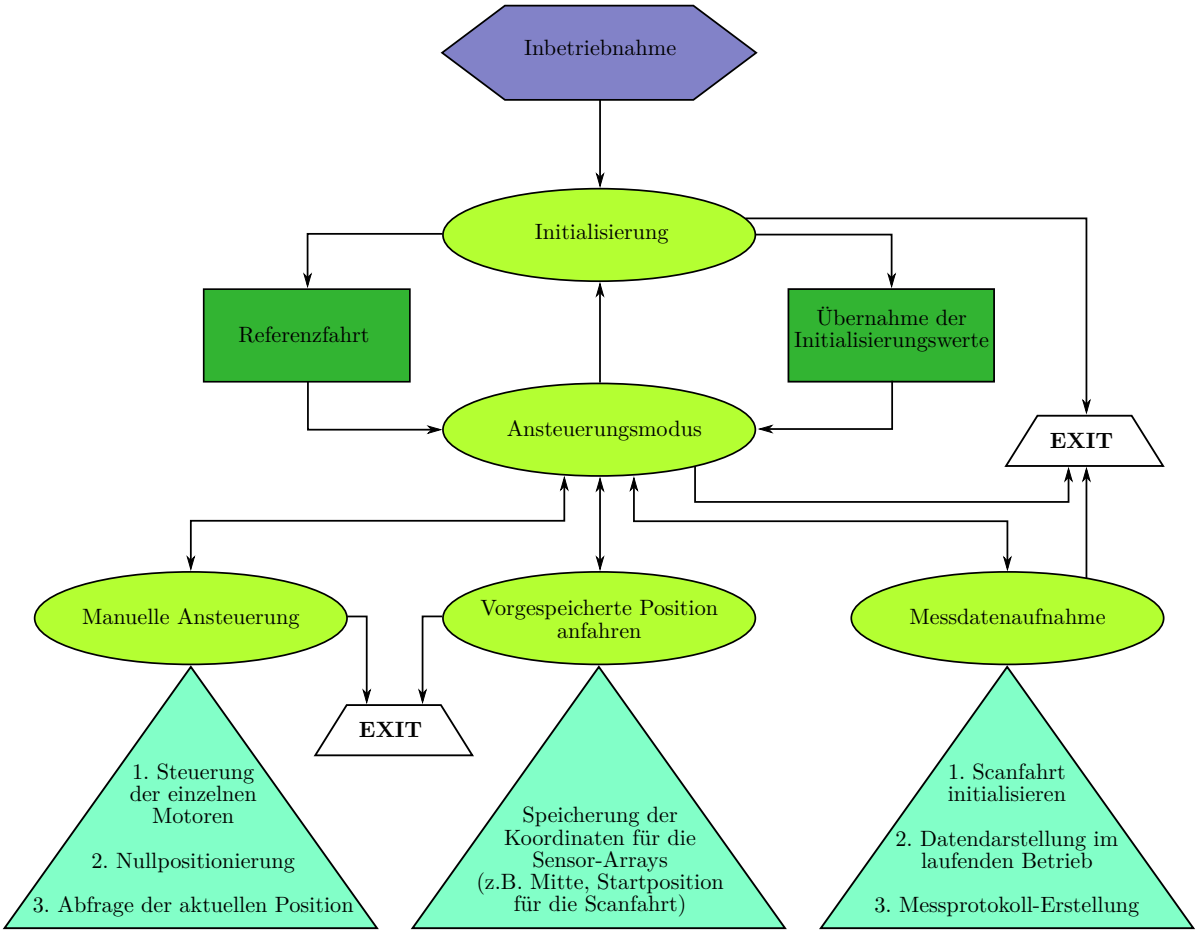


Abbildung 3.1: Struktogramm der Steuerung des Messplatzes.

3.2 Dokumentation für Messwerteaufzeichnung

Alle aufgenommene Messwerte werden analysiert und ausgewertet. Damit das realisierbar ist, wird folgende Struktur zur Abspeicherung der Werte entworfen und die Funktion *rmp_3_datei_erstellung.m* erfasst. Dazu wird für die Lokalisierung der Messposition und das Abspeichern ein Koordinatensystem im Bezug auf das Sensor-Array entworfen und in Abbildung 3.2 dargestellt. Es ist also möglich die Koordinaten nicht nur als ganze Zahl, sondern auch als Dezimalzahl zu bestimmen. Wenn der Roboterarm in der absoluten Mitte positioniert ist, werden ihm die Koordinaten [4.5 4.5] zugeordnet.

Bei dem Dateinamen wird ein Teil vom Benutzer und ein Teil automatisch generiert. Für die individuelle Eingabe wird keine Einschränkung außer allgemein für Matlab geltende vorgegeben. Der automatisch erstellte Teil besteht aus dem aktuellen Datum, Koordinaten bezüglich des Sensor-Array-Koordinatensystem und der eingegebenen Distanzschrittweite in mm. Im Anschluss sieht dann der Dateiname wie folgt aus:

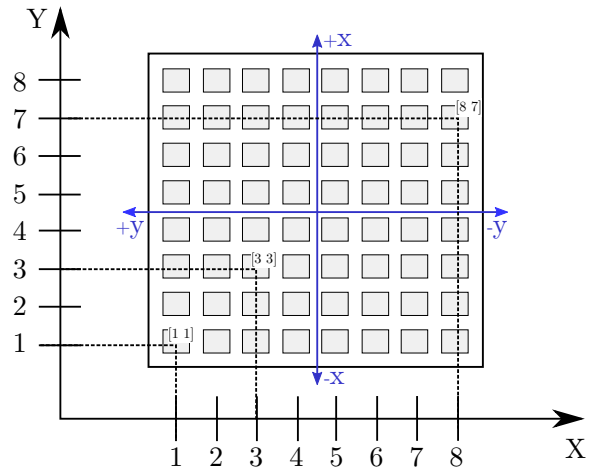


Abbildung 3.2: Das blaue Koordinatensystem bezieht sich auf das interne magnetische Referenzfeld des Array aus AMR-Sensoren. Das rote Koordinatensystem ist für Datenarchivierung festgelegt.

Benutzereingabe_datum_X_Koordinate_Y_Koordinate_Z_Abstand zum Array_in_mm_Schrittweite

In die Datei werden alle auf der Position durchgeführten Messwerte gespeichert. Wenn der Roboterarm bewegt wird, wobei die Koordinaten sich verändern, wird eine neue Datei mit einem entsprechenden Namen automatisch erstellt.

Bei der einzelnen Messung werden die notwendigen Informationen zusammengefasst und in dem dafür erstellten *Mat-file* abgespeichert. Dazu gehören folgenden Messdaten:

- Kosinus Signal von dem Array als Matrix der Dimension 8×8 (Digital)
- Sinus Signal von dem Array als Matrix der Dimension 8×8 (Digital)
- Kosinus Signal von dem Array als Matrix der Dimension 8×8 (Umgerechnet (3.1))
- Sinus Signal von dem Array als Matrix der Dimension 8×8 (Umgerechnet (3.1))
- Koordinate der Messung auf der X-Achse
- Koordinate der Messung auf der Y-Achse
- Winkel der Messung α

Die Umrechnung des Digitalwertes wird mittels folgender Formel durchgeführt:

$$U_{cos/sin} = ADC_{Wert} \frac{U_{ref}}{2^{12\,\mathrm{bit}}} \tag{3.1}$$

ist die Versorgungsspannung des Sensor-Arrays. Das erste *Mat-file* wird mit einer Ordnungszahl der Art *0000000.mat* beschriftet, die für jede Messposition nachgezählt wird. Das bedeutet, dass das letzte *Mat-file* einer Messreihe 0° - 90° eine Beschriftung *0000090.mat* hat.

3.3 Steuerungsablauf

Hier werden einzelne Menüs, die Initialisierungsphase und allgemeine Funktionen für die Steuerung des Messplatzes beschrieben, was auch auf der Struktogramm 3.1 entworfen ist.

rmp_3_inbetriebnahme.m Das Skript wird als erstes aufgerufen, womit die Schnittstelle zum Bussystem initialisiert und das erste Menü aufgerufen wird.

rmp_3_menu_start.m In dem Menü ist es möglich auszuwählen, ob tatsächlich eine Referenzfahrt mit dazugehörigen Achsen- und Motordatenberechnungen durchgeführt wird oder aus einer vorausgeführten Berechnung die notwendigen Initialisierungswerte übernommen werden. Bei der Referenzfahrt wird das Skript so optimiert, dass sich der Roboterarm im Anschluss in der Nullposition (Abbildung 4.1(c)) befindet und alle für die zukünftige Benutzung des Messplatzes notwendigen Werte berechnet werden. Wenn keine Referenzfahrt durchgeführt und die Initialisierungswerte übernommen werden, beeinflusst das keine zukünftige Funktionalität der Steuerung, da die gespeicherten Werte aus einer vorherigen Messplatzteuerung stammen, letzte Positionskoordinaten von dem Roboterarm haben und in einer Datei hinterlegt werden.

rmp_3_menu_manuelle_steuerung.m Für die Erstellung des manuellen Ansteuerungsmenü werden Funktionen aus der Abschlussarbeit [15, S. 112] herangezogen. Für die Erleichterung der Steuerung werden sie dahin gehend modifiziert, dass die Eingabe der Schritte, die gefahren werden, in einem separatem Fenster und in mm erfolgt. Da bei den Linearachsen ein Distanzschritt 32 Motorschritten und 10μm Distanzschrittweite [15, S. 113] entspricht, wird die Eingabe für die Linearachsen mit 100 multipliziert und im Anschluss an die Funktion für die Motorsteuerung übergeben. Zu dem Menü gehört eine Initialisierungsfahrt, die zur Nullposition führt.

rmp_3_menu_koordinaten.m Die wichtigsten Koordinaten werden in der Funktion abgespeichert und beim Auswählen wird der Roboterarm zur gewünschten Position gefahren. Die Definition dieser Koordinaten wird im Kapitel 4.4 beschrieben.

rmp_3_menu_meas_save.m Bei dem Menü handelt es sich um eine Darstellung und/oder Aufnahme von Messungen. Bei der Echtzeitdarstellung werden vom Benutzer einige Information abgefragt, wozu sich die Anzahl der Zyklen, Dauer eines Zyklus und Art Zeichnung (Kapitel 5.2 auf der S. 28) zählen lässt. Außerdem ist es möglich, automatisierte Versuchsabläufe zu starten, wobei man unterschiedliche Kriterien zur Auswahl

hat. Es ist möglich,zwischen eine Drehung, Translationsablage, Vor- und Rückwärtslauf oder einer Kombination von Rotation und Translation auszuwählen. Unter anderem gibt es eine Möglichkeit, ein Messprotokoll zu erstellen.

3.3.1 Messsteuerung der Versuchsabläufe im vollautomatischen Modus

Beim vollautomatischen Modus handelt es sich um ein Messverfahren, bei dem es möglich ist, Translation und Rotation beliebig zu kombinieren. Der Benutzer definiert das Messverfahren einmal und dementsprechend wird es vom Roboterarm abgefahren, Messdaten erfasst und abgespeichert. Die Messversuche werden sich von Fall zu Fall unterscheiden. Um eine Flexibilität zu verschaffen, wird so viel wie möglich der Initialisierungsinformation für den jeweiligen Versuchsablauf vom Benutzer abgefragt. Dazu gehören die folgenden Information:

- Anzahl von Messebenen (Verschiebung in der Z-Achse)
- Abstand zwischen Messebenen
- Dimension des Sensor-Arrays, das abgemessen werden soll
- Distanzschrittweite zwischen Messpositionen
- Anfangskoordinate für X-Achse laut angenommenem Koordinatensystem
- Anfangskoordinate für Y-Achse laut angenommenem Koordinatensystem
- Abstand zwischen den einzelnen Sensoren im Array
- Dateiname für die Speicherung der Messwerte
- Ein Endwinkel, bis zu dem eine Rotation stattfinden soll
- Ein Winkel für einen Schritt, womit rotiert werden soll
- Anzahl von Hin- und Rückwärtsläufen

Dabei ist zu beachten, dass einige Information gleich Null gesetzt werden dürfen. Wenn man bei einer Messaufnahme keine Rotation haben will, muss bei der Anforderung der Information über Rotation in entsprechenden Fenstern alle Werte gleich Null gesetzt werden.

Darauf wird auf das Messaufnahmemenü unter den Punkten: *„Translation und Rotation“ und „Rotation ueber den Array“* zugegriffen. Dabei werden folgende Funktionen aufgerufen: die *rmp_3_translation.m* und *rmp_3_rotation.m*. Unter Beachtung der Regeln aus dem oberen Abschnitt ist es möglich, Translations- oder Rotationsversuche durchzuführen. Wenn die Messdatenaufnahme beendet ist, wird dem Benutzer vorgeschlagen, ein Messprotokoll zu erstellen.

3.3.2 Messsteuerung der Versuchsabläufe im halbautomatischen Modus

Mit dem halbautomatischen Modus wird ein Versuch mit der Drehmatrix bezeichnet. Bei dem Versuch wird ein beliebig großes Sensor-Array mit einem Sensor simuliert. Aus den vom Benutzer eingegebenen Daten werden Koordinaten für ein Sensor-Array berechnet und vom Roboterarm abgefahren, worauf ein Sensor mit dem Sensoradapter (Abbildung 4.2(a)) befestigt wird. Die Realisierung des Versuches mit einem Sensor basiert auf dem Rotationsmatrixprinzip, worauf näher im Kapitel 5.1.2 eingegangen wird. In dem Fall ist es genau so wie beim vollautomatischen Messversuch möglich, den Rotationswinkel gleich null zu setzen, um nur Translationsversuche zu ermöglichen. Außerdem wird überprüft, ob das vom Benutzer dimensionierte Array in der befahrbaren Zone liegt. Wenn die Rotation des Arrays bezüglich eines ausgewählten Rotationspunktes nicht möglich ist, wird ein Warnfenster gegeben und nach der Bestätigung, dass diese Meldung wahrgenommen wurde, wird vom System noch mal vorgeschlagen, der Array neu zu definieren und einen Rotationspunkt neu auszuwählen. Beim Aufruf der *rmp_3_einzelnsensor_drehmatrix.m* werden folgende Angaben abgefragt:

- Rotationswinkel
- Rotationspunkt X-Koordinate
- Rotationspunkt Y-Koordinate
- Dimension des Sensor-Arrays, das abgemessen werden soll
- Distanzschrittweite zwischen Messpositionen
- Abstand zwischen dem Sensor und dem Magnet

3.4 Ansteuerung bzw. Auslesen der Sensormatrix und der Einzelsensoren

Die Messdatenaufnahme erfolgt beim Aufruf der *rmp_3_messaufnahme.m* Funktion. Bei den Messversuchen mit dem Sensor-Array werden über den Mikrocontroller die Daten seriell bzw. spaltenweise eingelesen. Das bedeutet, dass beim Array der Dimension 8×8 eine Spalte mit acht Sensoren, danach die nächste usw. abgelesen wird. Die Archivierung erfolgt sowohl in einem Digitalformat auch in einem nach der Gleichung (3.1) auf Versorgungsspannung umgerechneten Format.

Bei den Versuchen mit einem Sensor wird der Mikrocontroller entsprechend neu programmiert. Das ist notwendig, um entsprechende Eingangsports des Tiva Board für zwei Messwerte zu definieren. Die Messaufnahme sowie Messdatenarchivierung erfolgt genauso wie beim Sensor-Array.

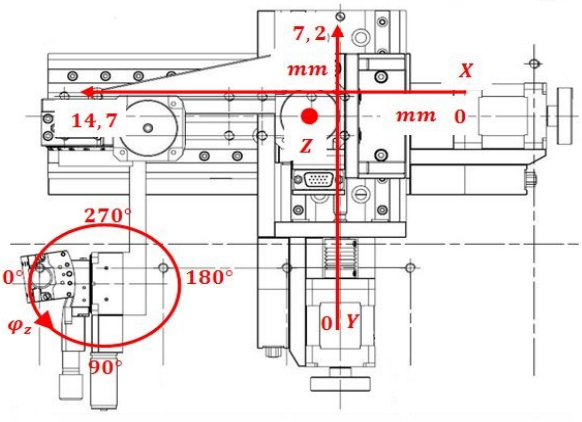
4 Hardware und Messplatz

Damit ein Messversuch gestartet werden kann, muss ein Befestigungsmodell für die Elektronik, als auch für die Magneten entworfen und gefertigt werden. Eine exakte Positionierung des Roboterarmes über dem Sensor-Array ist von größter Bedeutung, da kleine Positionierungsfehler zu Messwerten mit Fehlern führt. Nicht nur genauere, sondern auch sichere Messabläufe werden durchgeführt. Damit die eingesetzte Elektronik, entworfene Hardware und der Messplatz gegen Kollisionen geschützt werden, wird ein Sicherheitssystem entwickelt, getestet und eingesetzt. Im Folgenden wird auf die Vorgehensweise und Umsetzung der genannten Punkte eingegangen.

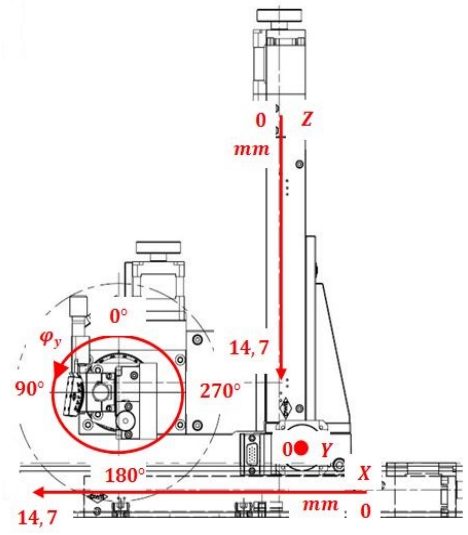
4.1 Koordinatensystem des Messplatzes

Für die Steuerung des Roboterarmes wird ein Koordinatensystem festgelegt. Das wird nach dem Prinzip vorwärts — positiv, rückwärts — negativ gelöst. Da sich der Roboterarm nach der Referenzfahrt in einem Initialisierungspunkt befindet, wird entschieden, den Punkt als Nullpunkt für ein Koordinatensystem des Messplatzes festzulegen (Abbildung 4.1(c)) und bezüglich diesen in folgenden Beschriftungen und Richtungen zu definieren. Im ersten Schritt ist es möglich, den Roboterarm von dem angenommenen Nullpunkt nur in eine Richtung entlang der drei Linearachsen zu bewegen. Genau die Richtung wird als positiv festgelegt. Das festgelegte Koordinatensystem ist in der Abbildung 4.1 dargestellt.

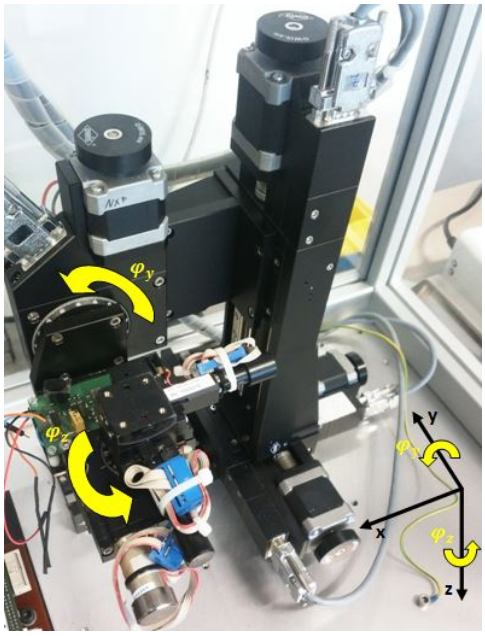
Die Rotationseinheit, die bei den Versuchen im Einsatz sind, sind nach der Referenzfahrt auch im Nullpunkt. Die 360° Drehung um die Y-Achse ist nur in einer Richtung möglich, da werksseitig ein Endschalter für die Referenzfahrt eingebaut wurde, der nicht überfahrbar ist [15, S. 111]. Deswegen wird die entgegengesetzte Richtung, das heißt, entgegen dem Uhrzeigersinn, als positiv festgelegt (Abbildung 4.1(b), φ_y). Der Motor, der eine Umdrehung um die Z-Achse ermöglicht, lässt sich aus dem Nullpunkt in beiden Richtungen um 360° drehen. Trotzdem läuft eine positive Drehung entgegen dem Uhrzeigersinn (Abbildung 4.1(a), φ_z), weil einerseits schon eine Rotationseinheit die gleiche Bezeichnung bekommen hat, andererseits die Grundplatte, worauf ein Magnet bzw. Sensor-Array platziert, links vom Roboterarm liegt und ausschließlich zur Grundplatte gedreht wird.



(a) Achsenbeschriftung der X-Y-Ebene, bei den Linearachsen ist ein Endwert angegeben.

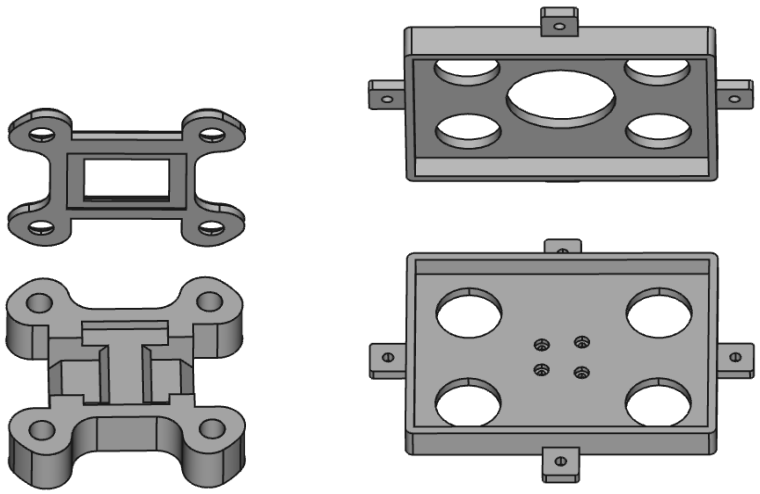


(b) Achsenbeschriftung der X-Z-Ebene, bei den Linearachsen ist ein Endwert angegeben.



(c) Nullposition, an der sich der Roboterarm nach der Referenzfahrt befindet.

Abbildung 4.1: Koordinatensysteme des Messplatzes für die einzelnen Positioniereinheiten. Die Pfeile in (c) stehen für die positive Drehrichtung bei einer Ansteuerung des Messplatzes. Die Pfeile zeigen in die positive Richtung, die der Steuerungseingabe beachtet wird.



(a) Das Befestigungsmodell für die Winkelsensoren.

(b) Das Befestigungsmodell für den Permanentmagneten.

Abbildung 4.2: Der Adapter für einen Magnetsensor und die Befestigung der unterschiedlichen Magnete.

4.2 Adaptermodule für die Permanentmagneten und die Sensorelektronik

Für die Messungen werden sowohl Magnete als auch Sensoren auf dem Roboterarm fixiert. Zunächst wird ein Adapter für die AMR- und TMR-Sensoren entworfen. Die Befestigung für die Sensorelektronik wird in zwei Teilen gefertigt. Auf den unteren Teil wird dann der Sensor gelegt und das obere Teil als Deckel verwendet. Um einen Magneten zu befestigen, werden auch zwei Teile gefertigt. Hier gilt dasselbe Verfahren wie bei dem Sensoradapter. In den unteren Teil wird ein Magnet gelegt und mit dem dazu gefertigten Deckel zusammengeschraubt. Der Austausch der beiden Module wird sehr schnell realisiert, dazu sind nur die vier Schrauben zu lösen. Die Teile werden im 3D-Druck gefertigt und sind in der Abbildung 4.2 dargestellt.

4.3 Sicherheitssystem

Ein Teil der Arbeit ist die Entwicklung der Sicherheitsschaltung, die mechanische Kollisionen erkennt und eine Notabschaltung bewirkt. Damit werden Fehlpositionierungen bei der Bedienung oder durch Softwarefehler erkannt und eine Beschädigung des Messplatzes vermieden. Die Sicherheitsabschaltung dient zum Schutz einerseits der Sensor-Arrays andererseits des Messplatzes, da eine kleine Deformation später zu Ungenauigkeiten oder sogar zu total falschen Werten bei einer Positionierung bzw. Messung führen kann.

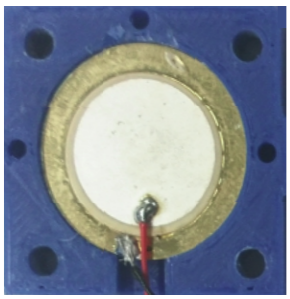
Eine Kollision kann durch fehlerhafte Ansteuerung oder falsche Positionierung der Elemente auf dem Messplatz passieren. Um die Sensoren bzw. die Schrittmotoren des Messstandes zu schützen, muss eine relativ schnelle Abschaltung bei einer minimalen Berührung stattfinden. Weiterhin muss eine präzise Positionierung eines Sensors bzw. Magnetes für einen fehlerfreien Messverlauf gewährleistet sein. Da die Messungen sowohl mit unterschiedlichen Magneten als auch mit verschiedenen Sensoren erfolgen, muss die Notabschaltung möglichst unabhängig vom variierenden Gewicht der angebrachten Bauteile sein.

4.3.1 Sensorauswahl

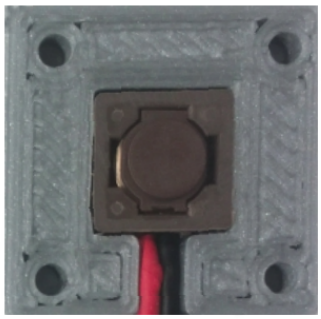
Die Sicherheit ist ein sehr wichtiges Thema, deswegen gibt es in der Industrie eine große Auswahl an Sensoren, mit denen ein sicherer Ablauf von Prozessen beobachtet bzw. kontrolliert wird. Sie zeigen in der Anwendung ihre Vielgestaltigkeit z.B. für Licht, Strahlung, Druck, Durchfluss, Füllstand, Abstand, Kraft, Beschleunigung. Das heißt, Genauigkeit, Schnelligkeit, Größe, Einbaustelle usw. spielen eine entscheidende Rolle bei der Auswahl des Sensortyps. Ein detaillierter Vergleich der Eigenschaften von vielen Sensoren für die Abschaltung wurde in der Abschlussarbeit von Ivanov durchgeführt [7, S. 38].

Im Folgenden werden zwei Sensortypen miteinander verglichen und getestet. Der erste ist eine Piezo-Scheibe (Abbildung 4.3(a)) und der zweite ein Mikrotaster (Abbildung 4.3(b)).

Das Wort Piezo (oder veraltet Piëzo) lässt sich aus der griechischen Sprache mit Druck, drücken, pressen übersetzen. Das Prinzip basiert auf Piezoelektrizität, auch piezoelektrischer Effekt genannt. Wenn Elementarzellen des piezoelektrischen Materials verformt



(a) Das Piezoelement in der Halterung.



(b) Der Mikrotaster in der dazu geeigneten Halterung.

Abbildung 4.3: Piezoelement (a) und Mikrotaster (b) als Sensoren für die Sicherheitsabschaltung.

werden, wird die Außenfläche durch eine Trennung der positiven und negativen Ladungen aufgeladen. Eine Spannung wird erzeugt, wenn Elektroden sich von zwei gegenüberliegende Oberflächen anstoßen [12].

Ein Mikroschalter bzw. Schalter arbeitet nach dem Alles-oder-nichts-Prinzip [2], was bei jeder Berührung zu einem neuen Zustand führt — ein oder aus. Schalter lassen sich in eine Vielzahl Typen unterscheiden, je nach Anwendung, Betätigungseinheit, Schaltfunktion usw. [13]. Für das Sicherheitssystem wird ein Taster untersucht. Das ist ein Bedienelement, das beim Drücken seinen Zustand ändert und beim Loslassen wieder die Ausgangsposition annimmt [17].

4.3.2 Systementwurf des Sicherheitssystems

Für die Fertigung der Abschaltkonstruktion wird Kunststoff Polylactide (PLA) eingesetzt, da er keinen Einfluss auf das magnetische Feld ausübt und störungsfreie Feldmessungen erlaubt. Alle Befestigungsteile werden mit dem Programm FreeCAD [5] entworfen und im Anschluss mit Hilfe eines 3D-Druckers gefertigt.

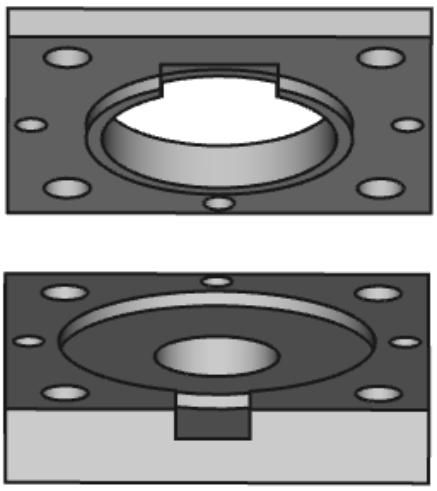
Für die Piezo-Scheibe ist es wichtig, sie fest in einem Adapter unterzubringen, da schon kleinste Bewegungen des Elementes Signale liefern. Um eine stabile Positionierung des Sensors zu erreichen, wird ein Bauteil gefertigt, wie in Abbildung 4.4(a) gezeigt wird. Das Piezo-Scheibe-System Piezo-Scheibe-System (PSS) wird in die dazu gefertigte Tasche verlegt und mit dem Deckel fest zusammengeschraubt, sodass der Sensor keine freie Bewegung mehr hat. Dafür wird auf dem Deckel eine ringförmige Schicht gefertigt, die auf den Piezorand drückt. Die hohlen Stellen auf den beiden Teilen werden für die Verdrahtung des Piezoelements genutzt.

Für einen Mikrotaster ist dies nicht so kritisch, da nur beim Betätigen des Tasters ein Signal entsteht. Deswegen wird auf die Produktion eines Deckels für die Befestigungsschicht verzichtet (vgl. Abbildung 4.4(b)).

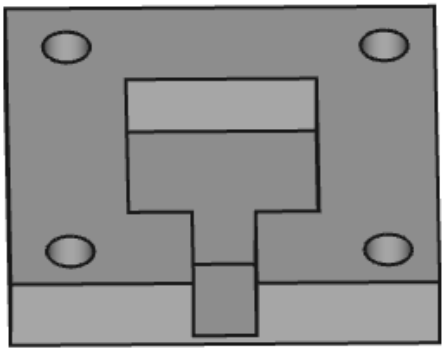
Darauf wird eine Verbindungsschicht für das Sicherheitssystem (VS) gesetzt, die eine Magnet- bzw. Sensorhalterung mit der Piezohalterung verbindet. Die Schicht muss zwei Eigenschaften erfüllen. Einerseits muss sie eine präzise Positionierung des Magneten bzw. des Sensors bieten, andererseits immer sehr leichtlaufend bleiben, damit das Not-Aus-System sehr feine Berührungen detektieren kann. Das Problem wird mit dem in Abbildung 4.4(c) gezeigten Teil realisiert.

Weiterhin wird in der Zeichnung 4.4(d) ein Stift gezeigt, der eine Druckkraft bei einer Kollision auf den Sensor überträgt und das System abschaltet. Das Teil wird auf die Verbindungsschicht gesetzt.

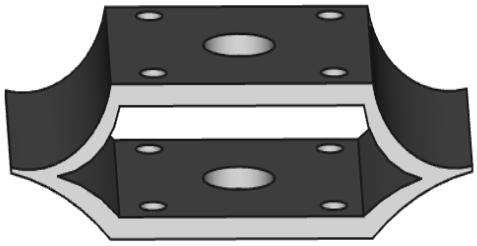
Das Gesamtsystem für die Befestigung des Piezo-Elements bzw. des Mikrotasters und die Detektion einer Kollision bei der Fehlpositionierung ist in der Abbildung 4.4(e) bzw. 4.4(f) dargestellt.



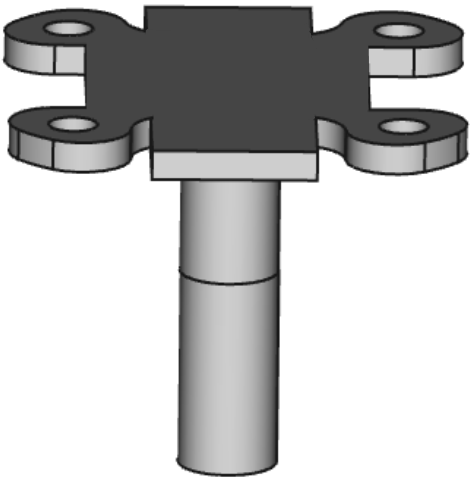
(a) Die Halterung für eine Piezo-Scheibe.



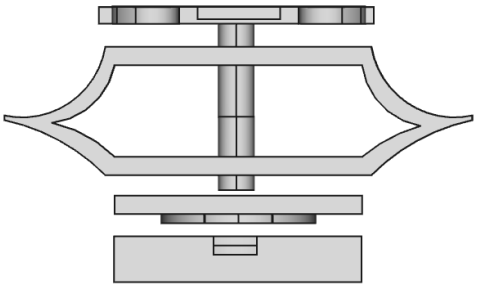
(b) Halterung für die Piezo-Scheibe mit Deckel.



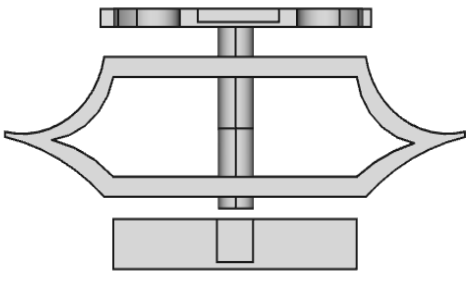
(c) Elastischer Verbinder zwischen der Magnet- bzw. Sensorhalterung und der Halterung der Piezo-Scheibe.



(d) Stift, mit dem die Kraft bei einer Kollision übergeben wird.



(e) Das Sicherheitssystem mit einer Piezo-Scheibe.



(f) Das Sicherheitssystem mit einem Mikrotaster.

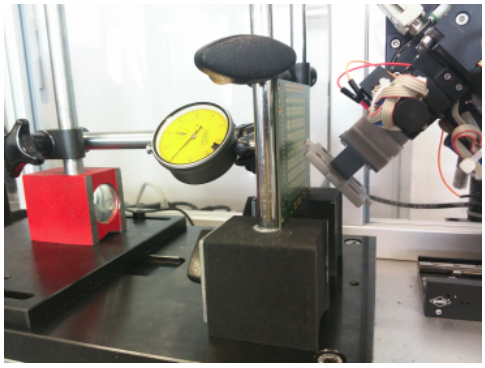
Abbildung 4.4: Konstruktion des Sicherheitssystems für die Fertigung mittels 3D-Druck.

4.3.3 Test der Kollisionssensoren

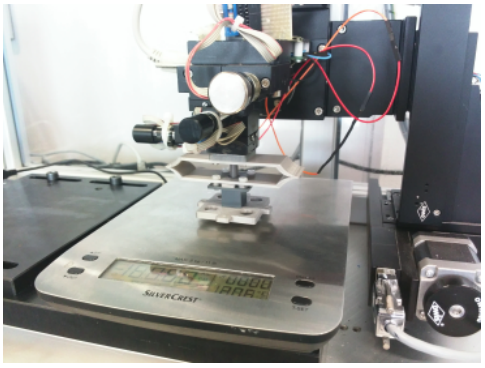
Das Experiment zeigt einerseits, welche Kraft benötigt wird, damit das Sicherheitssystem ein Signal wahrnimmt und den Betrieb des Roboterarmes abschaltet, andererseits, ob die benötigte Kraft klein genug für einen schadenfreien Betrieb ist. Dafür werden drei Messstative, eine Messuhr und eine Platine eingesetzt. Die Platine wurde in der ISAR-Arbeitsgruppe für die AMR-Sensor-Array entworfen, wobei das Array im Rahmen der Arbeit untersucht wird. Da die Platine das größte Biegemoment relativ zur restlich eingesetzten Elektronik hat, wird sie beim Kollisionstests eingesetzt und das Sicherheitssystem darauf eingestellt. Das bedeutet, dass bei anderen Sensortypen, die härter sind und ein kleineres Biegemoment haben, das System noch schneller ausgeschaltet wird. Sie wird zwischen zwei Messstativen befestigt, wobei die Stative auf einer Grundplatte des Messplatzes positioniert werden. Die Messuhr wird mit Hilfe des dritten Stativs befestigt und auf die Mitte der Platine ausgerichtet. Somit kann die Ausdehnung gemessen werden.

Mit dem auf den Roboterarm befestigten Sicherheitssystem und der darauf gesetzten Magnethalterung wird gegen Platine gefahren, wobei die Strecke, vom Kontakt des Abschaltsystem und Platine bis zum Ausschalten gemessen wird. Für jeden Aufbau wird der Versuch zehnmal wiederholt. Dieser Test wird bei unterschiedlichen Stoßwinkeln und verschiedenen Ausrichtungen der Verbindungsschichten (Abbildung 4.4(c)) durchgeführt. Im Anhang A in der Abbildung A.1 sind die Tests für φ_y von 45° bis 135° zusammengestellt.

Bei dem Test aus Abbildung 4.5(b) wird eine Waage eingesetzt, damit die Messwerte von zwei unterschiedlichen Messmethoden verglichen werden können. Die Waage wird auf der Grundplatte des Messplatzes positioniert und φ_y auf 180° eingestellt. Anschließend wird gegen die Waage mit dem Roboterarm bis zum Ausschalten gefahren und ein



(a) Aufbau des der Kollisionstests.



(b) Experiment mit einer Waage bei φ_y = 180°.

Abbildung 4.5: Aufbau für die Fehlpositionierungstests.

Referenzgewicht gemessen. Nach der Notabschaltung wird das Sicherheitssystem weggefahren und auf der Waage wird das für die Abschaltung benötigte Gewicht angezeigt. Das gemessene Gewicht wird in Gleichung (4.1) eingesetzt, um die benötigte Kraft mit ($a = 9{,}81\,\mathrm{m/s^2}$) zu erhalten.

$$F = m \cdot a \tag{4.1}$$

4.3.4 Auswertung der Kollisionstestergebnisse

Alle aufgenommenen Werte sind auf S. 46 in Tabelle A.1 bzw. A.2 eingetragen. Die wichtigsten Werte sind in den Tabellen 4.1 bzw. 4.2 zusammengefasst.

Damit die beide Messmethoden verglichen werden könnten, werden die Messwerte, die in mm dargestellt sind, auch wie folgt in Kraft umgerechnet. Die Platine wird genau positioniert wie bei allen Versuchen. Dann wird mit einem Faden und mit einer Hängewaage die Platine verbogen, wobei die Messungen aufgenommen werden. Nach mehrmaligen Versuchen kann man feststellen, dass die Verformung und die Kraft linear abhängig sind und bei 0.1 mm 200 g beträgt. Die Masse, die für eine Verformung benötigt wird, wird in die Formel (4.1) eingesetzt und die Kraft bestimmt. Der Test ist in der Abbildung 4.6 dargestellt. Die Extrema der Testergebnisse sind in Abbildung 4.8 gegeneinander geplottet.

Beide Sicherheitssensoren weisen stabile und fast gleiche Werte auf. Da Abmessungen keine entscheidende Rolle bei der Sensorauswahl spielt, wobei die Piezo-Scheibe zehnfach dünner als der getestete Mikrotaster und der Preis bei die beiden auch fast identisch ist, wird auf Lebensdauer und Robustheit eingegangen. Beide Kriterien sind von den Einsatzbedingungen abhängig. Das Piezo-Element weist eine lange Lebensdauer und Zuverlässigkeit auf, da es mit speziellen Materialien versiegelt ist. Wenn die Benutzung immer im elastischen Bereich bleibt, wird eine Lebensdauer von mehreren Milliarden Zyklen garantiert [12]. Die Lebensdauer von Mikrotastern beträgt maximal eine halbe Millionen Zyklen, was deutlich weniger ist. Obwohl keine dynamische Benutzung des Sicherheitssystems zu erwarten ist, wird die Piezo-Scheibe ausgewählt und im System integriert.

Tabelle 4.1: Zusammenfassung aus dem Kollisionstest mit dem Piezo-Scheibe-System.

		Verbindungsschicht 4.4(c) ist waagerecht ausgerichtet			Verbindungsschicht 4.4(c) ist senkrecht ausgerichtet		
φ_y	180°	45°	90°	135°	45°	90°	135°
Messeinheiten	(N)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)
Mittelwert	7,080	0,241	0,129	0,222	0,267	0,127	0,229
Maximum	7,210	0,245	0,130	0,225	0,270	0,130	0,230
Minimum	6,288	0,240	0,125	0,220	0,260	0,125	0,225

Tabelle 4.2: Zusammenfassung aus dem Kollisionstest mit dem Mikrotaster-System.

		Verbindungsschicht 4.4(c) ist waagerecht ausgerichtet			Verbindungsschicht 4.4(c) ist senkrecht ausgerichtet		
φ_y	180°	45°	90°	135°	45°	90°	135°
Messeinheiten	(N)	(mm)	(mm)	(mm)	(mm)	(mm)	(mm)
Mittelwert	6,894	0,269	0,132	0,269	0,263	0,119	0,276
Maximum	7,269	0,270	0,135	0,270	0,270	0,130	0,280
Minimum	6,533	0,265	0,125	0,260	0,260	0,110	0,275

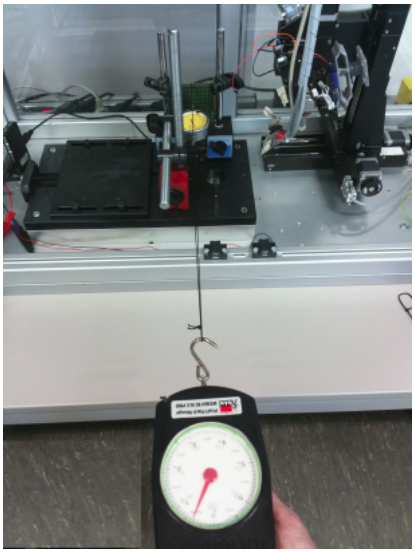


Abbildung 4.6: Verformung der Platine mit der Hängewaage.

4.4 Positionierung des Roboterarms

Da die Sensoren relativ kleine Abmessungen (KMZ60: 5 mm × 4 mm [8], ADT001: 2,5 mm × 2,5 mm [1]) haben und die Positionierung des Magneten über dem Sensor von großer Bedeutung ist, wird im Folgenden die Vorgehensweise bei der Lösung des Problems beschrieben.

Das Ziel ist, den Roboterarm und die Grundplatte in ein verbundenes System zu bringen. Das System ist in der Abbildung 4.7 dargestellt. Auf die Grundplatte wird ein Messstativ mit der Messuhr so positioniert, dass eine senkrechte Kante von der Platte und von dem Stativ in einer Ebene liegen. Dann wird die Positioniereinheit verschoben, wobei eine Berührung der Messuhr detektiert wird. Ab diesen Moment sind die Grundplatte und der Roboterarm miteinander verbunden. Mit Hilfe der Konstruktionszeichnungen, die aus der Abschlussarbeit [15] herangezogen sind, werden Abstände von einem Rand zur Mitte der Positionierung des Magnetes bzw. des Sensor-Arrays sowie auf

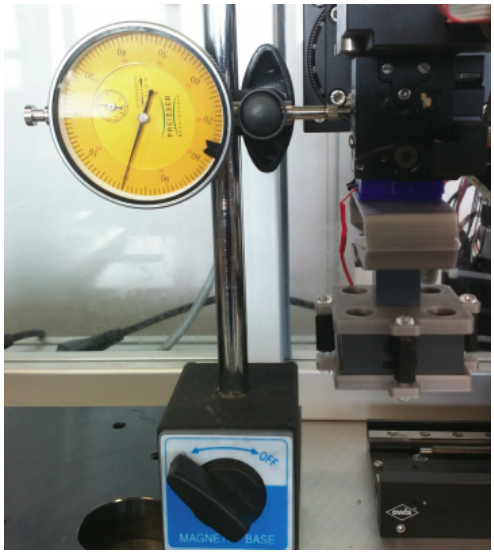


Abbildung 4.7: Positionierung des Roboterarms über der Grundplatte.

der Grundplatte als auch auf dem Roboterarm festgestellt. Die komplette Messstrecke ist in der Zeichnung dargestellt.

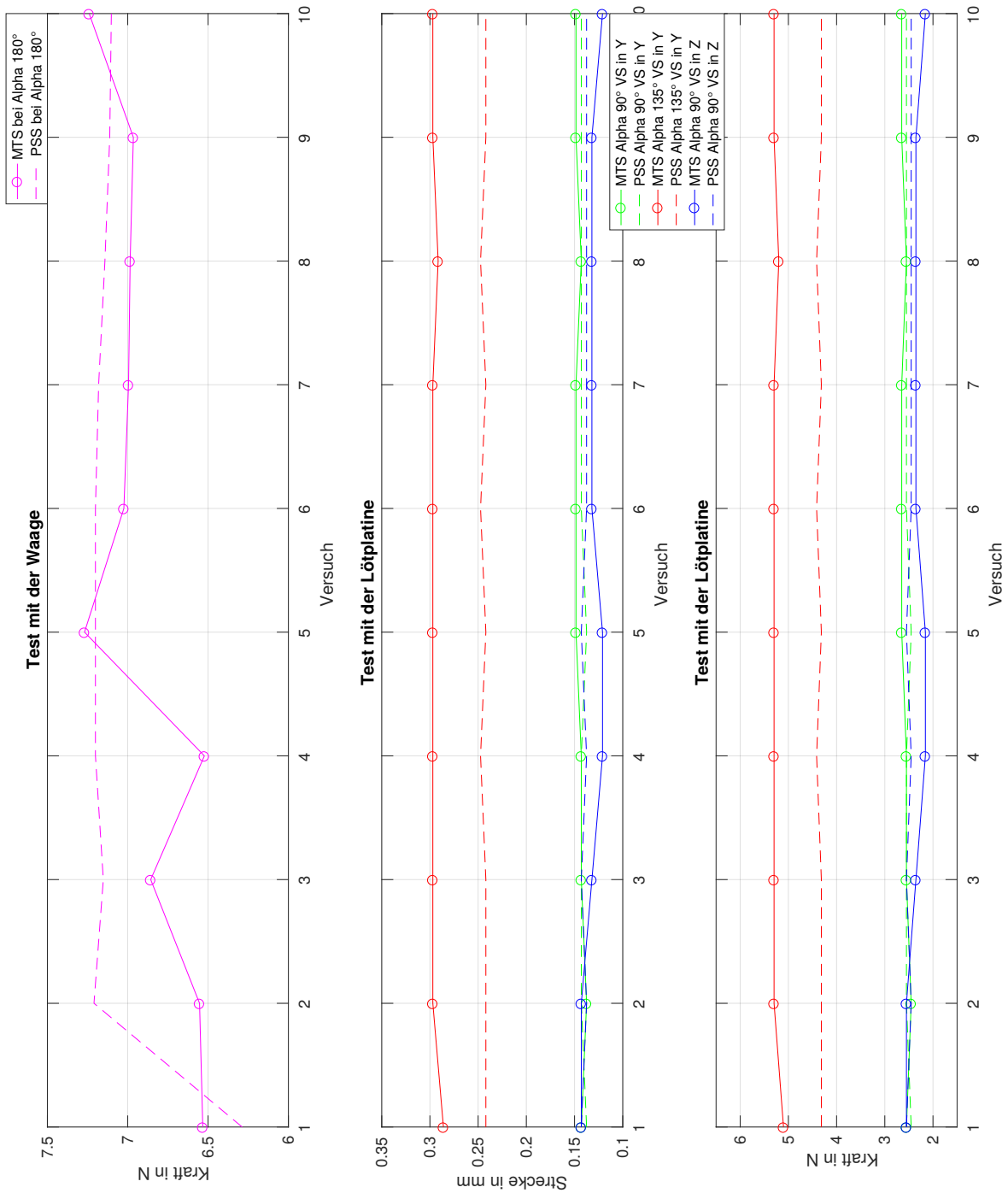


Abbildung 4.8: Die Gegenüberstellung der Extrema der Kollisionstestergebnisse von beiden Sicherheitssystemen.

5 Datenerfassung und Visualisierung

Im Folgenden werden die erstellten Messpläne, die Hardwareaufbauten für die Datenaufnahme und im Anschluss die Darstellung der Messdaten beschrieben. Die Messpläne werden erfasst, um die Messaufnahmen zu systematisieren und die Identität zwischen allen Messversuchen zu verschaffen. Für die Analyse der aufgenommenen Messwerten wird Auswertungssoftware als Matlab-Skript erfasst.

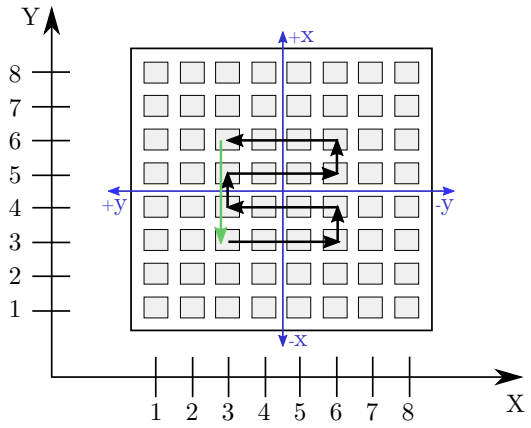
5.1 Erstellung eines Messplanes

Der Messablauf lässt sich grundsätzlich auf die zwei unterschiedlichen Vorgehensweisen einteilen. Einer gilt für die automatische Messaufnahme, wobei es möglich ist, eine Translation, Rotation und eine Änderung der Messhöhe zu definieren. Der andere Messablauf wird mit Hilfe der Rotationsmatrix realisiert und grundsätzlich für die Versuche mit einem Einzelsensor entworfen. Die beiden Methoden haben denselben Messwerteaufzeichnungsverlauf und dieselben Koordinatensysteme, was im Kapitel 3.2 beschrieben wurde.

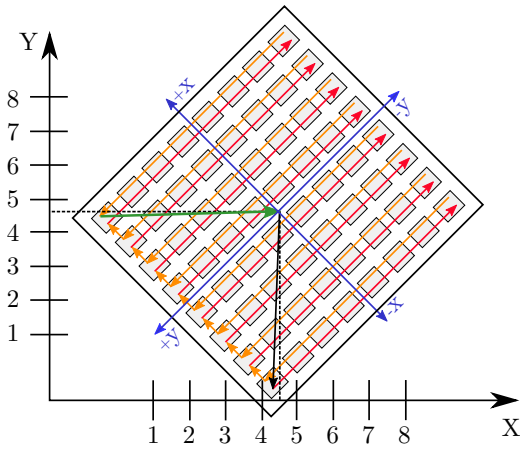
5.1.1 Messplan für einen vollautomatischen Messablauf

Ein vollautomatischer Messablauf wird für jede Fahrt nach demselben Prinzip realisiert. Die schematische Darstellung des Messverlaufs von einer Fläche aus den mittleren 16 Sensoren ist aus der Abbildung 5.1(a) zu entnehmen. Es wird erst immer zum ersten Messpunkt gefahren (in dem Fall hat der Messpunkt die Koordinaten [3 3]), der als Startpunkt definiert ist und am Ende des vom Benutzer definierten Messmoduls wieder angefahren wird, was mit einem grünen Pfeil auf der Abbildung 5.1(a) gekennzeichnet ist. Ob danach eine andere Messung durchgeführt wird oder eine Initialisierungsfahrt durchgeführt wird, ist dem Benutzer freigestellt. Von dem Startpunkt werden alle Messpunkte erst in die positive Richtung der X-Achse und dann ein Distanzschritt in die positive Y-Richtung gefahren. Bei dem Beispiel entspricht der Distanzschritt dem Abstand zwischen zwei nebeneinander zusammen liegenden Sensoren. Danach werden alle Punkte entlang der X-Achse, aber in die gegengesetzte Richtung, abgescannt. Wenn diese Ebene komplett gefahren ist, wird der Roboterarm erst zum Startpunkt bewegt und im Anschluss bei Bedarf in eine andere Scanebene gewechselt. Alle oben beschriebenen Richtungen beziehen sich auf das Koordinatensystem, das im Kapitel 3.2 beschrieben wurde.

Bei der Definition der Messung werden dem Benutzer vorgeschlagen, eine Rotation zu verbinden. Das bedeutet, dass bei jedem Distanzschritt der Magnet oder der Sensor,



(a) Der Messverlauf bei Translation und Rotation.



(b) Messverlauf einer 45° Umdrehung mit der Rotationsmatrix um den Mittelpunkt eines 8 × 8 simulierten Sensor-Arrays.

Abbildung 5.1: Schematische Darstellung des Messverlaufes.

je nach dem was auf dem Roboterarm befestigt ist, rotiert wird, wobei die Messungen aufgenommen werden. Die Rotationswinkel, Endwinkel und die Periodizität wird vom Benutzer angefragt.

5.1.2 Messplan für einen halbautomatischen Messablauf

Das Messverfahren ermöglicht ein Sensor-Array mit einem Sensor darzustellen. Dabei ist zu beachten, dass die Dimension des Arrays beliebig definiert werden kann. Dazu wird noch die Drehung des definierten Arrays bezüglich eines frei wählbaren Punkts realisiert. Als Basis wird die zweidimensionale Drehmatrix gewählt, die eine Drehung im euklidischen Raum beschreibt. Die Determinante der Matrix muss gleich eins sein, um eine Drehmatrix darzustellen. Mit der Drehmatrix wird nach Gleichung (5.1) Koordinaten für die rotierte Matrix (x' y') berechnet.

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{5.1}$$

Der Verlauf ist in der Abbildung 5.1(b) dargestellt, wobei der Roboterarm wie folgt verfährt. Erst muss sich der Arm im Rotationspunkt, worum gedreht wird, befinden. Das ist über das direkte Ansteuerungsmenü oder durch vorher gespeicherte Koordinaten realisierbar. Nach Positionierung des Roboterarms und Eingabe aller notwendigen Werte für den Versuch wird zum rotierten Startpunkt gefahren (vgl. Abbildung 5.1(b)). Es folgt die zeilenweise Ansteuerung der Punkte, die das rotierte Sensor-Array bilden. Nachdem die Daten der ersten Zeile vorliegen, wird der erste Punkt der Zeile angefahren. Anschließend folgt der Sprung auf die nächste Zeile, welche dann wieder spaltenweise

erstellt wird. Nachdem alle Punkte angesteuert und die Messdaten erfasst sind, wird der Roboterarm im Rotationspunkt positioniert.

5.1.3 Messplan für die Untersuchung der Hysterese, einer Offset-Spannung und der Sättigung

Es wird eine gleiche Vorgehensweise für die Bestimmung des Offset und eine Untersuchung des Hysterese-Effektes vorgenommen. Auf die Grundplatte wird eine Positionierungsscheibe angebracht, worauf ein Magnet befestigt wird (Abbildung 5.2). Der Roboterarm wird mit einem von den zwei Winkelsensoren bestückt und über die Mitte des Magneten positioniert. Dabei wird auf die magnetische Feldstärke geachtet, damit die Sensoren sich im Sättigungsbereich befinden. Es wird in eine Richtung mit Messaufnahmen und danach in die Gegenrichtung rotiert, wobei die Messungen bei den gleichen Winkelpositionen aufgenommen werden. Bei dem AMR-Sensor wird bis zu 180° in 2-Grad-Schritten gedreht und die Werte gemessen. Wegen des werksseitig eingebauten Endschalters bei der Rotationseinheit, wird bei den TMR-Sensoren bis 260° in der gleichen Schrittunterteilung gedreht.

Um die Hysterese genauer zu untersuchen, wird ein weiterer Versuch durchgeführt, der sich in der Vorgehensweise von der vorherigen unterscheidet. Beim zweiten Versuch wird kein magnetisches Feld auf dem Sensor in der Ausgangsposition angelegt. Danach wird der Roboterarm zum Magnet nah wie möglich gefahren und um 90° gedreht. Im

Abbildung 5.2: Messaufbau für den Versuch mit einem Sensor.

Anschluss wird der Sensor wieder hochgefahren aber nicht mehr rotiert. Das wird viermal wiederholt, das heißt, am Versuchsende wird ein Winkel $\alpha = 360°$ erreicht.

Für die eine genauere Darstellung des Sättigungsbereiches wird ein Sensor über die Mitte des Magneten positioniert und senkrecht vom Magnet weggefahren. Nun wird die Messaufnahme gestartet, wobei das externe magnetische Feld etwa Null ist. Dann wird der Magnet durch den Roboterarm zum Sensor geführt, wobei der Abstand zwischen den beiden sehr gering ist. Danach wird der Magnet um 90° gedreht, wieder hochgefahren und auf die Null-Grad-Position eingestellt. Der Verlauf wird mehrmals wiederholt, wobei die Messaufnahme im Laufe der Messung immer aktiv bleibt.

5.2 Visualisierung der Messdaten

Die Messdaten werden mittels der Auswertungssoftware als Matlab-Skript dargestellt und analysiert. Für die Darstellung der aufgenommenen Daten werden die folgenden Funktionen aus Matlab genutzt:

- **surf()** — die Funktion erzeugt farbliche 3D-Flächendarstellung
- **quiver3()** — mit der Funktion werden 3D-Vektorfelder geplottet
- **plot()** — die Funktion stellt die parametrisierten Kurven dar

Die Messversuche werden wie mit den Einzelsensoren auch mit dem Sensor-Array durchgeführt. Das bedeutet, dass sich die Menge der aufgenommenen Werte bei einer Messaufnahme zwischen einem Sensor und Sensor-Array unterscheidet. Dadurch werden unterschiedliche Skripte für die Darstellung entworfen. In der Abbildung 5.3 wird ein Struktogramm für die Auswahl des Darstellungsskriptes vorgestellt.

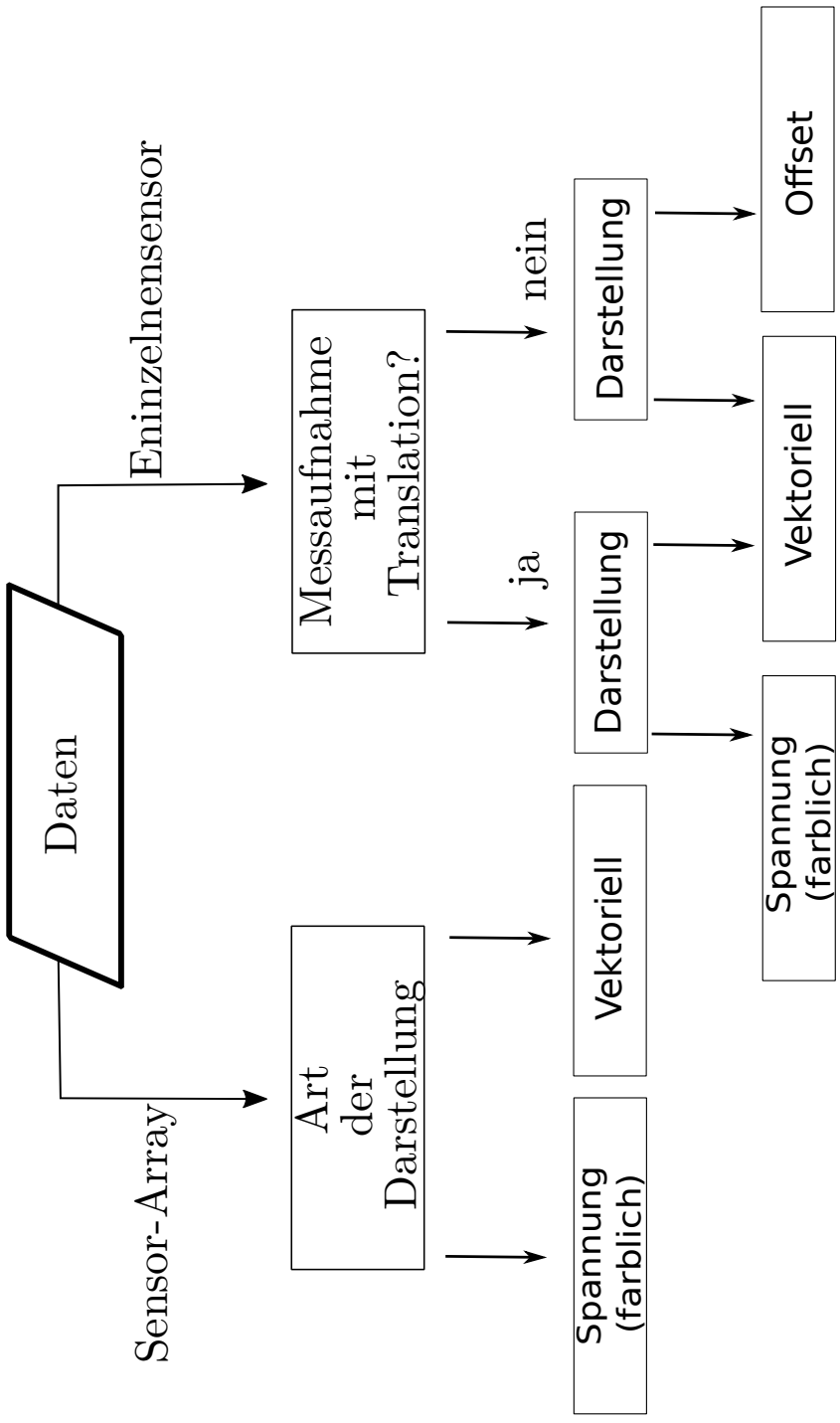


Abbildung 5.3: Stuktogram der Auswahl eines Darstellungsskriptes in Abhängigkeit von ausgewählten Messdaten.

6 Auswertung der Messdaten

Es gibt zwei Messreihen, die sich voneinander unterscheiden. Zum einen ist eine Messung mit dem Sensor-Array und zum anderen die Messung mit einem einzelnen Sensor. Im Folgenden werden Versuchsreihen in einem fehlerfreien und fehlerbehafteten Betrieb beschrieben. Zudem folgt eine Untersuchung der Hystereseversuche.

6.1 Datenerfassung in einem fehlerfreien und in einem fehlerhaften Betrieb

Die messtechnische Datenerfassung wird mit idealen Betriebszuständen und Fehlerpositionen durchgeführt. Dafür werden unterschiedliche Spannungsquellen für die Versorgung des Sensors, Vorschaltfilter für die Minimalisierung des Rauschen bei der Signalerfassung und die Positionierung des Magneten in Bezug auf den Sensor-Array oder Einzelsensor.

6.1.1 Der fehlerfreie Betrieb

Für die Versorgungsspannung werden zwei Spannungsquellen miteinander verglichen. Eine der Quellen ist ein Stromversorgungsgerät vom Typ NGT35. Eine vergleichbare Spannungsquelle wird vom Tiva Board genommen. Bei dem Vergleich wurde festgestellt, dass das Ausgangssignal kaum von den beiden ausgewählten Quellen beeinflusst wird (vgl. Abbildung 6.1).

Für den verwendeten AMR-Sensor ist es nicht notwendig, ein Filter einzusetzen, da das Ausgangssignal ausreichend stabil und rauscharm ist. Bei dem TMR-Sensor ist das Ausgangssignal hingegen verrauscht. Das Rauschen wird mit einem Tiefpassfilter mit einer Zeitkonstanten $\tau = 100\,\mu s$ ($R = 10\,k\Omega$ und $C = 10\,nF$) reduziert. In Abbildung 6.2 ist der Vergleich zwischen gefiltertem und ungefiltertem Signal dargestellt. Die erste 100 Messungen werden ohne Filter aufgenommen. Dabei hat das Ausgangssignal viele Ausreißer und die Amplitude beträgt 150 mV. Ab der 100. Messung wird das RC-Filter eingebaut. Dabei ist festzustellen, dass das Signal stabiler ist und die Amplitude 60 mV beträgt. Mit einem eingebauten Filter ist das Signal für eine Datenauswertung hinreichend stabil.

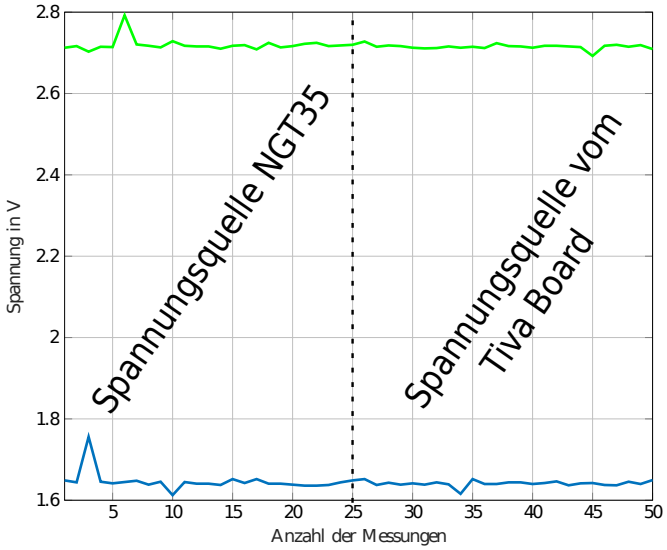


Abbildung 6.1: Der Vergleich von den Spannungquellen für die Versorgung, wobei die erste 25 Messungen eine externe Spannungsquelle und die Messungen 25 – 50 eine Spannungsquelle von Tiva Board darstellt.

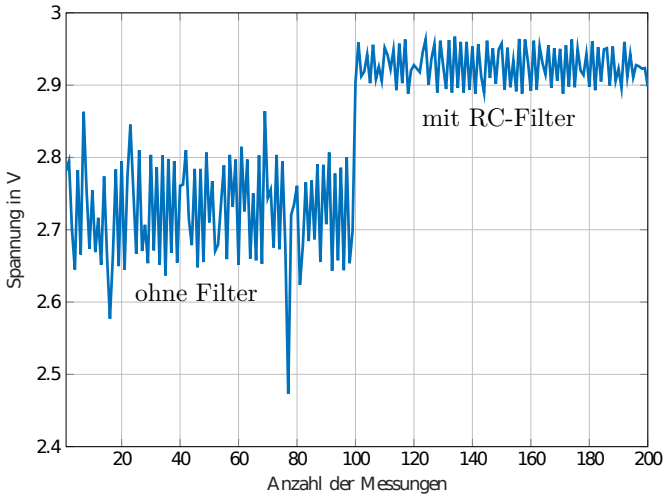


Abbildung 6.2: Die Reduzierung der Rauschen durch RC-Filter mit $\tau = 100\,\mu s$.

6.1.2 Der fehlerhafte Betrieb

Der fehlerhafte Betrieb wird mit zwei unterschiedlichen Vorgehensweisen durchgeführt. Zum einen wird die Messaufnahme bei einer Fehlpositionierung des Magneten erfasst. Das bedeutet, dass der Magnet außerhalb des Zentrums des Sensor-Arrays positioniert ist. In dem Versuch wird der Magnet rotiert. Die Messergebnisse sind in Abbildung 6.3 dargestellt.

Zum anderen wird die Messaufnahme ohne Sättigung erfasst. Das bedeutet, dass die

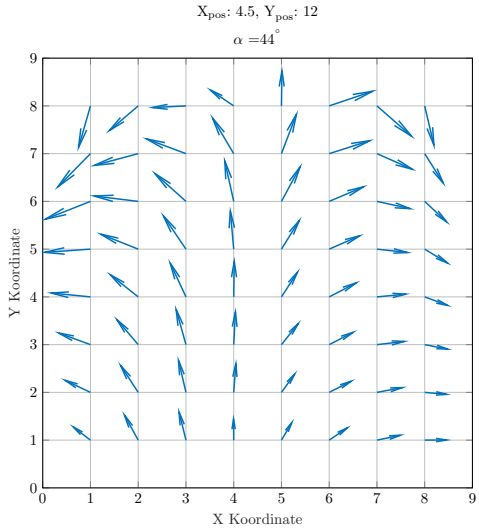


Abbildung 6.3: Positionierung eines Magneten ausserhalb des Sensor-Array.

angelegte Feldstärke maximal 15 kA/m beträgt und somit 10 kA/m geringer ist, als die geforderte Feldstärke aus dem Datenblatt des AMR-Sensors [8]. Der Magnet wird mittig in Bezug auf das Sensor-Array positioniert. Die Messwerte werden während einer Rotation aufgenommen. Das führt dazu, dass die interne Magnetisierung der Sensoren hinsichtlich des äußeren Magnetfeldes nicht mehr folgen kann. Die fehlerhafte Messergebnisse sind in Abbildung 6.4 dargestellt.

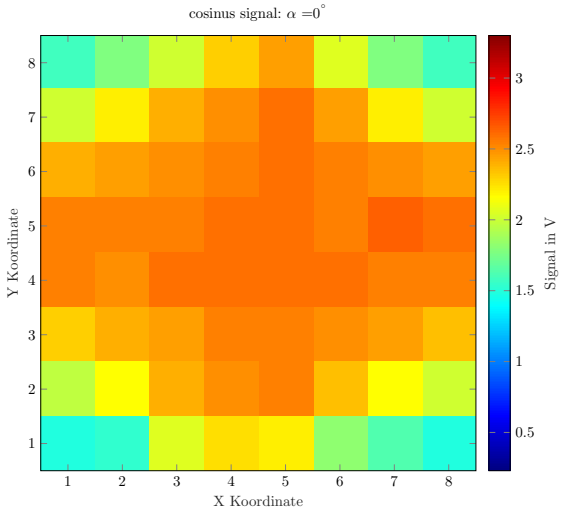


Abbildung 6.4: Bei dem Versuch wird keine Sättigung erreicht.

6.1.3 Datenerfassung mit einem Störfeld

Bei dieser Messreihe wird das Sensor-Array einem Störfeld ausgesetzt. Die Abbildung der Messaufbau befindet sich auf S. 49 in Abbildung B.1. Mehrere Magneten werden in einem Ring platziert. Die Positionierung ist so festgelegt, dass innerhalb des Rings ein homogenes Feld vorliegt [16]. Die Feldstärke in der Mitte des Kreises beträgt etwa 5,2 kA/m. In Abbildung 6.5 ist die Wirkung des Störfeldes auf das Sensor-Array dargestellt. Zudem ist die Homogenität des Feldes zu erkennen.

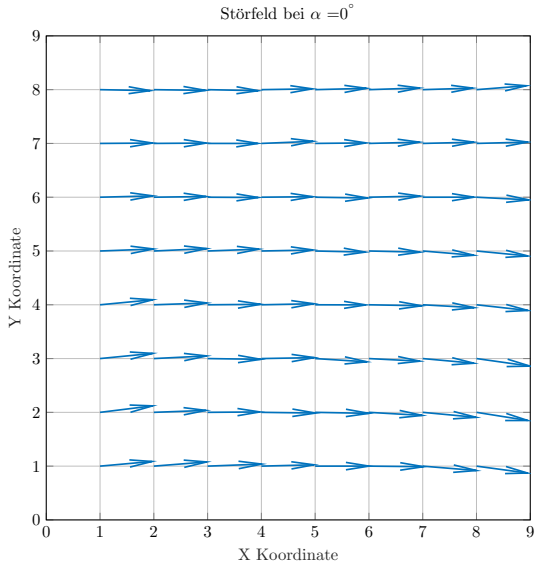


Abbildung 6.5: Messergebnisse für das angelegte Störfeld.

Bei dem Messversuch mit dem Stabmagneten (60 mm × 80 mm × 10 mm) und dem Störfeld wurde festgestellt, dass die abstoßende Kraftwirkung von dem Magneten auf den Roboterarm und dem Störfeld auf der Grundplatte groß genug ist, dass die Schrittmotoren und die Rotationseinheit keinen fehlerfreien Betrieb mehr gewährleisten (vgl. Abbildung 6.6).

Deswegen wird die Messreihe mit dem Störfeld mit einem Stabmagneten in der Abmessung 40 mm × 40 mm × 10 mm durchgeführt. Dabei wird der Magnet hinsichtlich des Arrays mittig positioniert und rotiert. Aus Gründen der Verformung wird in diesem Versuch auf eine Translation des Magneten verzichtet.

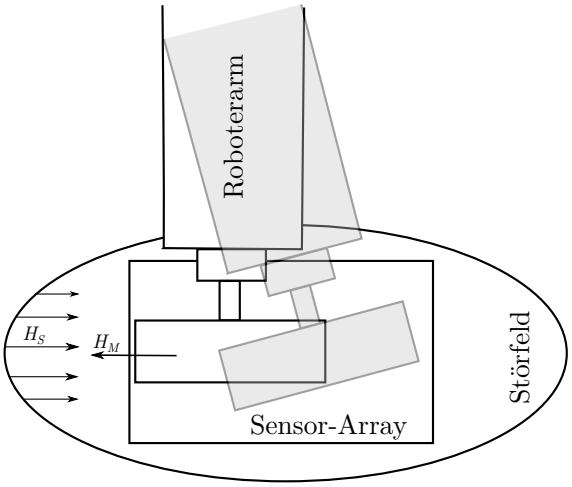


Abbildung 6.6: Verformung des Messaufbaus bei dem Versuch mit einem Störfeld. H_M — magnetisches Feld des Magneten, H_S — magnetisches Feld des Störfeldes.

6.2 Charakteristische Effekte der Sensoren

Die Sättigung, die Hysterese und typische Messabweichungen werden in diesem Kapitel beschrieben. Um die Effekte zu beschreiben, werden Tests mit einem Sensor und mit einem Sensor-Array durchgeführt. Die Vorgehensweise und die Ergebnisse werden erläutert.

6.2.1 Sättigung

Eine der wichtigsten Punkte für eine fehlerfreie Winkelmessung ist die Sättigung, die in Betrieb genommene Sensoren erreichen müssen. Die KMZ60-Sensoren müssen ein externes magnetisches Feld von mindestens 25 kA/m haben [8]. Um den Sättigungsbetrieb der AMR-Sensoren für die vorgesehenen Messversuche beim Sensor-Array und beim Einzelsensor zu erreichen, ist sehr starkes externes Magnetfeld anzulegen. Das ist mit einem Magneten lösbar, der viel größer ist als das Sensor-Array bzw. ein Einzelsensor. Alternativ durch einen kleinen Magneten, der über ein extrem starkes Magnetfeld verfügt. Die beiden Varianten sind bei dem bestehenden Messplatz nicht anwendbar, da sich dadurch die Positioniereinheit verformen würde.

Bei den Versuchsreihen mit dem Einzelsensor KMZ60 wird eine 10mm × 10mm Fläche abgefahren, wobei eine minimale externe Feldstärke von 22,6 kA/m anliegt. Das führt zur Verfälschung der Messwerte. Bei dem Sensor-Array wird ein Stabmagnet eingesetzt, der größer als die Array-Fläche ist. Dennoch kann die Feldstärke durch eine Rotation oder Translation des Magneten deutlich abnehmen. Durch die Bewegung kann es sein, dass Sensoren nicht mehr unterhalb der Magnetfläche liegen. Liegt ein Sensor beispielsweise direkt unter der Kante bzw. Ecke des Magneten, hat die Z-Komponente des magnetischen

Feldes ihr Maximum und die X- und Y-Komponenten gehen gegen null. Daher erfährt die interne Magnetisierung des Sensors nahezu keine Ablenkung und der Ausgangsstrom wird nicht beeinflusst.

Bei dem TMR-Sensor von der Firma NVE ist das minimale und maximales extern angelegte magnetische Feld zu beachten. Das externe Feld muss zwischen ca. 2,39 kA/m und 15,92 kA/m sein, um fehlerfreie Messungen zu ermöglichen [1]. Bei dem Sensor ist es realistischer, die Sättigung bei einer Messung für alle Sensoren in einem Array zu erreichen, da einerseits die benötigte Fläche für ein Sensor-Array aus den TMR Sensoren viermal kleiner als aus den KMZ60-Sensoren sein kann und andererseits das benötigte minimale externe magnetische Feld relativ zum KMZ60 zehnfach kleiner für die fehlerfreien Messungen sein muss.

6.2.2 Hysterese

Hysterese bezeichnet ein Systemverhalten, wobei die Ausgangsgröße von der Eingangsgröße und vor allem von der Vorgeschichte abhängt. Der Effekt wird messtechnisch untersucht. Es werden zwei unterschiedliche Messverfahren realisiert. Es wird ausschließlich mit den Einzelsensoren gemessen und die aufgenommene Werte werden gegenüber gestellt. Dabei wird der Magnet auf der Grundplatte und ein Sensor auf dem Roboterarm positioniert.

Dann werden die Messwerte bei zwei nacheinander durchgeführten Verläufen (vor- und rückwärts) miteinander verglichen. Aus den dargestellten Messwerten für ein Kosinus-Signal bei $\alpha = 36°$ und jeweils Vor- und Rückwärtslauf wird mit Gleichung (6.1) die Differenz mit der Einheit V gebildet. Mit Gleichung (6.2) wird die Differenz prozentual in Bezug auf die Referenzspannung des ADC berechnet. Die gemessenen und berechneten Werte sind in Abbildung 6.7 dargestellt. Der maximale Fehler bei dem Versuch beträgt 1,64 %, was 54 mV entspricht, wobei keine regelmäßigkeit der Fehler festgestellt wurde. Bei dem Vergleich der Messwerte mit dem AMR-Sensor-Array wird festgestellt, dass die Sensoren eine schmale Hysterese haben.

$$U_{cos_d} = U_{cos_{vor}} - U_{cos_{rueck}} \tag{6.1}$$

$$U_{cos_p} = (U_{cos_{vor}} - U_{cos_{rueck}}) \cdot \frac{100}{U_{ref}} \tag{6.2}$$

Wobei $U_{cos_{vor}}$ bzw. $U_{cos_{rueck}}$ — Kosinus-Signal bei Vor- bzw. Rückwärtsfahrt, U_{ref} — eine Referenzspannung des ADC.

Aus dem Versuch, wobei ein externes magnetisches Feld an- und abgeschaltet wird, werden die gemessenen Werte in Abbildung 6.8(b) dargestellt. Darin sind ein Rotationsbereich (Gelb) und der Bereich ohne angelegtes Feld (Grün) farblich gekennzeichnet. Bei der Darstellung ist erkennbar, dass das Sinus-Signal nach jeder Umdrehung seinen Ausgangswert annimmt. Dagegen weist das Kosinus-Signal einen schmalen Hysterese-Effekt

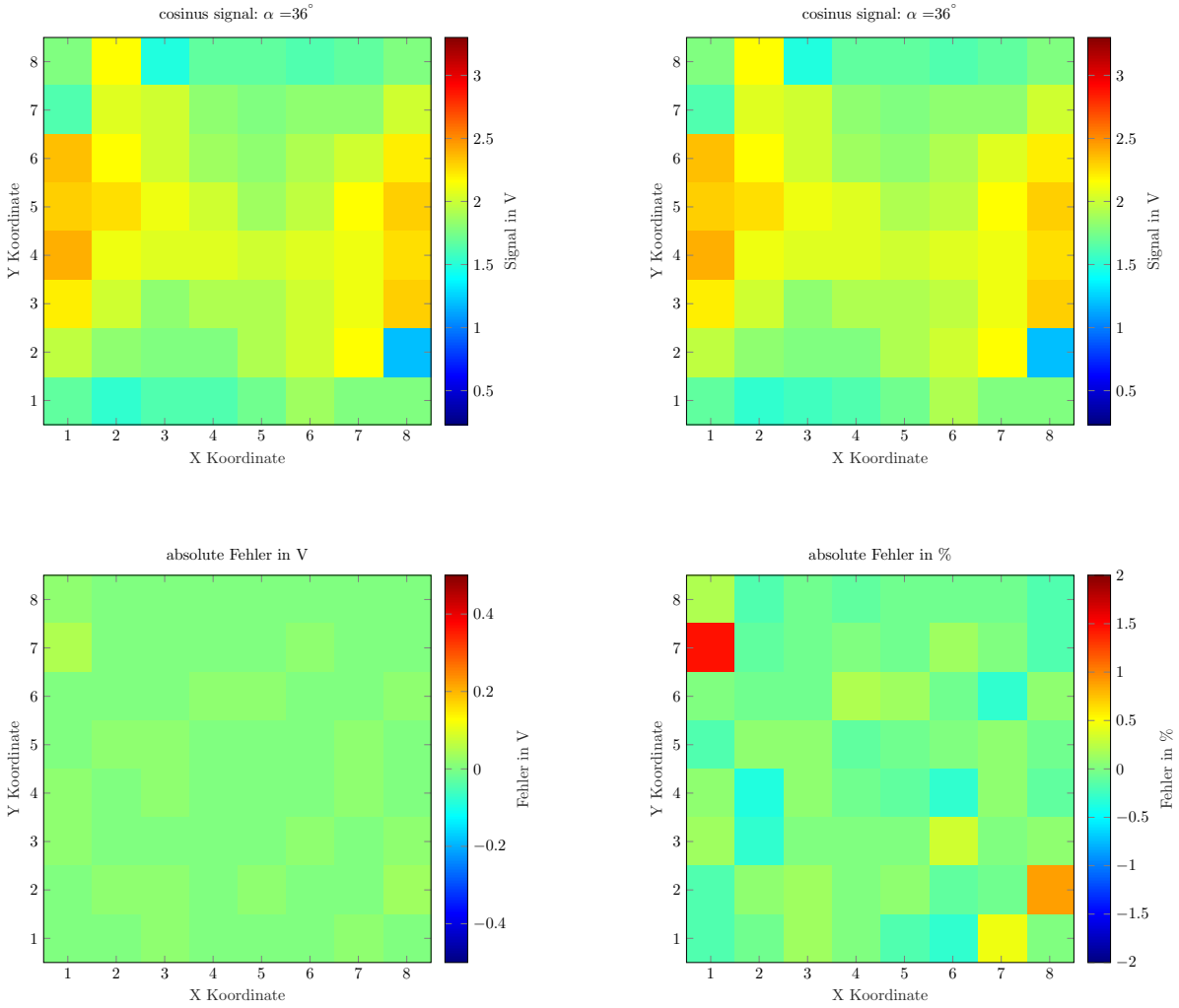
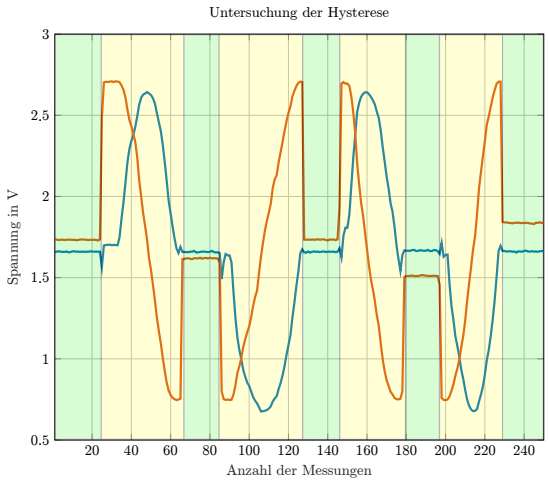


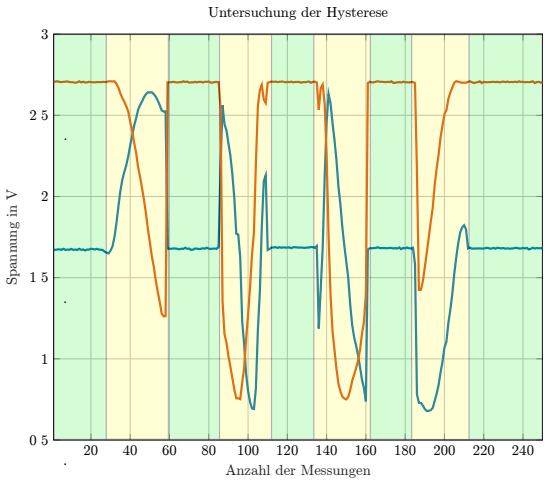
Abbildung 6.7: Der Messverlauf mit Vor- und Rückwärtsrotation für die Bestimmung des Hysterese-Effektes. Oben links — Kosinus-Ausgangssignal beim Vorwärtsrotation, oben rechts — Kosinus-Ausgangssignal beim Rückwärtsrotation, unten links — die Differenz in Einheiten V, unten rechts — die Differenz im Prozent hinsichtlich der Referenzspannung des ADC.

auf. Der Effekt lässt sich mit der Befestigung eines Stützmagnet beseitigen. Die Polarisation des Magneten entspricht der Richtung des internen Referenzfeldes von dem Winkelsensor. Um den Hysterese-Effekt beim KMZ60 zu beseitigen, wird ein etwa 13 kA/m starkes Stützfeld angelegt. In Abbildung 6.8(a) wird der Versuch mit einem Stützmagneten dargestellt. Nach jeder Umdrehung erreicht das Signal seine Ausgangsposition.

Der gleiche Versuch wird mit einem TMR-Sensor durchgeführt. Die Ergebnisse sind



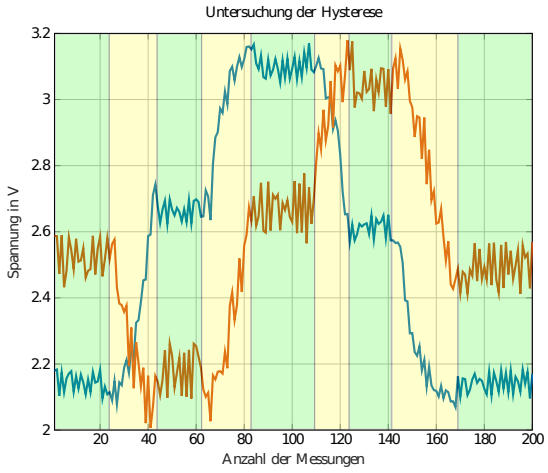
(a) Messverlauf ohne einen Stützmagneten unter dem Sensor.



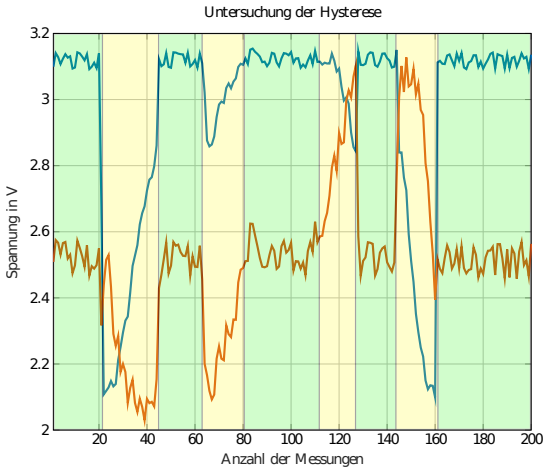
(b) Messverlauf mit einem Stützmagneten unter dem Sensor mit einer Feldstärke von 13 kA/m.

Abbildung 6.8: Der Hysterese-Effekt eines AMR-Sensors (KMZ60) von der Firma NXP.

in der Abbildung 6.9 dargestellt. Der Sensor weist eine sehr breite Hysterese auf. Der Effekt lässt sich genauso wie bei dem AMR-Sensor mit Hilfe des Stützmagneten lösen. Der Unterschied zum AMR-Sensor ist, dass der TMR-Sensor mit einem Stützfeld von 1,5 kA/m beaufschlagt wird.



(a) Messverlauf ohne einen Stützmagneten unter dem Sensor.



(b) Messverlauf mit einem Stützmagneten unter dem Sensor mit einer Feldstärke von 1,5 kA/m.

Abbildung 6.9: Der Hysterese-Effekt eines TMR-Sensors (ADT001) von der Firma NVE.

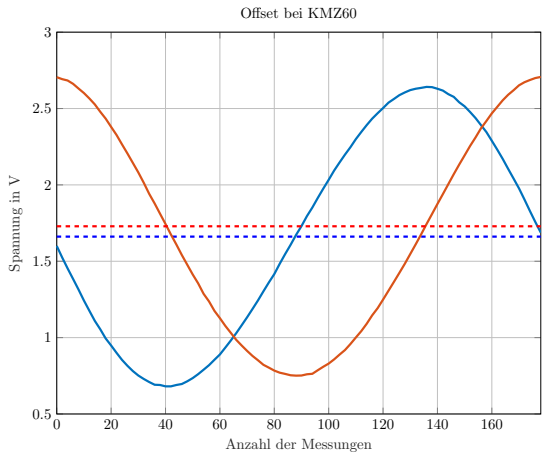
6.2.3 Offset

Im Rahmen dieser Arbeit wird sowohl ein Messplan als auch eine Darstellungssoftware für die Analyse der Offset-Spannung entworfen. Die Betrachtung hinsichtlich der Kalibrierung der Sensoren ist sehr wichtig. Jeder Sensor in einem Sensor-Array kann eine andere Offset-Spannung haben, was zu abweichenden Ausgangssignalen führt. Aus diesem Grund ist es nötig, die Sensoren erst zu kalibrieren.

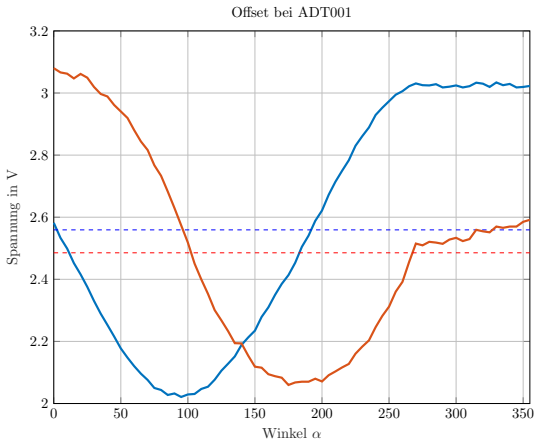
Die Offset-Spannung wird über Mittelwertbildung nach Gleichung (6.3) berechnet. In Abbildung 6.10 werden die aufgenommenen Werte mit den berechneten Offset dargestellt.

$$U_{off} = \frac{1}{N} \sum_{i=1}^{N} U_{cos_i} \tag{6.3}$$

Die Technologie der Winkel-Sensoren basiert auf der Wheatstone-Brücke. Die magnetoresistiven Widerstände haben nach der Herstellung voneinander abweichende Werte, was zur Verfälschung des Ausgangssignales führen kann. Da die Sensoren bei ungleichen Widerständen in den Brückenschaltungen mit einer Gleichspannung versorgt werden, kommt ein Gleichspannungsoffset hinzu. Die Kompensation der Offset-Spannung wird im Rahmen der Arbeit nicht näher behandelt.



(a) Offset-Spannung des KMZ60-Sensors bei einer Drehung bis 180°.



(b) Offset-Spannung des ADT001-Sensors bei einer Drehung bis 360°.

Abbildung 6.10: Die Messergebnisse der Offset-Spannung. Links ist der berechnete Offset bei einem KMZ60-Sensor und rechts bei einem ADT001-Sensor.

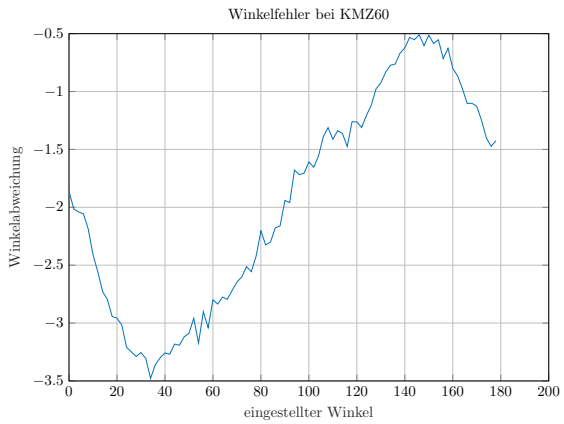
6.3 Toleranz des Messsystems

Für die spätere Analyse und Bewertung der Messergebnisse muss das Messsystem in Bezug auf Toleranz und Ursache der Fehler betrachtet werden. Fehler können bei dem Messplatz und bei der Positionierung der Sensoren auftreten. Da die Messungen durch auftretende Toleranzen verfälscht werden können, ist die Betrachtung der Abweichungen wichtig.

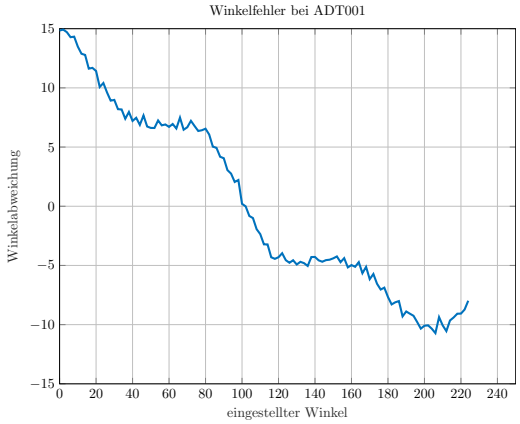
Die wichtigsten Toleranzen werden im Folgenden aufgelistet:

- Die Positionierung eines Sensors auf der Platine. Bei der Fertigung des Sensor-Arrays führt kleine Abweichung der Position eines Sensor zu den anderen zu Ausgangsfehlern. Die Eigenschaften dieser Einflüsse bleibt systematisch und in allen Messungen erhalten.
- Die Befestigung des Sensors und des Magneten auf dem Roboterarm. Eine minimale Verkippung führt zur Verfälschung des Ausgangssignals.
- Die Positionierung des Roboterarmes beim Einstellen, sowohl des Mittelpunktes in der X-Y-Ebene als auch des Rotationswinkels φ_z.
- Zum größten Teil ist der Messplatz aus eisenhaltigen Materialien gefertigt, was einen Einfluss auf das magnetische Feld zur Folge hat.
- Allgemeine Toleranzen der verwendeten Sensoren (Offset-Spannung, Hysterese, Rotation des Chip im Sensorgehäuse).

Bei dem Vergleich des eingestellten Winkels φ_z und dem gemessenen Winkel α wird festgestellt, dass die Winkel voneinander abweichen. Die Berechnung des Winkels α wird nach Gleichung (2.5) durchgeführt. Für den Vergleich werden die beiden Winkel subtrahiert und die Differenz in Abbildung 6.11 dargestellt. Die oben genannten Toleranzen führen bei dem KMZ60-Sensor zur einer Abweichung von −0,5° bis −3,5° und bei dem ADT001-Sensor von etwa ±15°.



(a) Winkelfehler eines KMZ60-Sensors bei einer Drehung bis 180°.



(b) Winkelfehler eines ADT001-Sensors bei einer Drehung bis 225°.

Abbildung 6.11: Die Abweichung des eingestellten Winkels φ_z von dem gemessenen Winkel α.

7 Schlussfolgerungen

Im Folgenden werden die Ergebnisse dieser Abschlussarbeit und die erreichten Ziele beschrieben. Im Anschluss werden Ansätze für Verbesserungen und weitere Arbeiten genannt.

7.1 Zusammenfassung

Die Inbetriebnahme des Prüfstandes mit automatisierter Positionssteuerung, Hardware-Aufbauten für die Positionierung der Sensoren und des Magneten auf dem Prüfstand, eine Entwicklung des Sicherheitssystem für die Vermeidung mechanischer Kollisionen, ein Entwurf und eine Optimierung der Steuerungs- und Auswertungssoftware waren die wichtigsten Aufgaben dieser Arbeit, die als erfüllt angesehen werden.

Bei dem Prüfstand wurden sowohl Linearachsen und Dreheinheiten für die präzise Positionierung neu kalibriert. Umfangreiche Tests bestätigen die Anwendbarkeit des Messplatzes für die vorgesehenen Untersuchungen der Magnetfelder einzelner Magnetsensoren als auch Sensor-Arrays. In der Durchführung wurden einige Begrenzungen der Anwendbarkeit festgestellt, was den allgemeinen Einsatz des Messplatzes für die Versuche nicht beeinflusst.

Die entworfenen und hergestellten Magnet- und Sensorbefestigungen zeigten einerseits eine präzise Befestigung der Elemente am Messplatz, andererseits die Flexibilität bei dem Austausch der Magneten und der Sensors. Für jeden Magneten wurde eine eigene exakt passende Befestigung gefertigt. Bei der Sensorhalterung ist dies nicht der Fall. Die Universalität der Halterung ermöglicht den Einsatz mit handelsüblichen AMR- und TMR-Sensoren.

Das integrierte Sicherheitssystem weist eine stabile und sehr genaue Funktionalität auf. Durch umfangreiche Tests wurde das System an die eingesetzten Elemente angepasst. Die ausreichende Anzahl der Tests und Simulationen der Kollision zeigte eine sichere und regelmäßige Detektion der Kollision und damit die Notabschaltungsfunktion. Die Sicherheit ist dann gegeben, wenn die Druckkraft auf das Sensor- oder Magnetbefestigungselement ausgeübt wird. Das Sicherheitssystem erkennt keine Kollision, die außerhalb der Befestigungsmodule auftritt.

Die Hauptziele beim Entwurf der Software waren eine benutzerfreundliche Oberfläche und eine große Auswahl der Einstellungen für Messungen und Positionierungen. Es steht eine große Auswahl der Einstellungen für die manuelle Steuerung der einzelnen

Schrittmotoren, als auch für die automatisierten Verläufe der Messungen zur Verfügung. Die Archivierung der Messdaten wird auf die gleiche Art durchgeführt, was die spätere Implementierung erleichtert. Es ist möglich, für jeden Messversuch ein Messprotokoll in einer vorgefertigte Vorlage auszufüllen und mit den Messdaten zu speichern.

Die exemplarische Implementierung zeigt die Anwendbarkeit des Messsystems und der Steuerungssoftware für weitere Untersuchungen des Sensor-Arrays. Die unterschiedlichen Konfigurationen und Erweiterungen der Software stehen für weitere Arbeiten zur Verfügung.

7.2 Ausblick

Im Rahmen dieser Arbeit wurde kein Sensor-Array aus TMR-Sensoren untersucht, was für das Forschungsprojektes ISAR sehr wichtig ist. Diese Sensoren weisen ein sehr hohes Einsatzpotenzial auf. Dafür wäre ein Vergleich der TMR-Sensoren unterschiedlicher Hersteller hilfreich, da sie sich in charakteristischen Eigenschaften unterscheiden können.

Die Messaufnahme und deren Auswertung im 3D-Raum wäre eine weitere Vorgehensweise bei der Implementierung der Eigenschaften des Arrays, was auch im Rahmen dieses Forschungsprojektes geplant ist. Der Messplatz ermöglicht eine Verkippung der auf dem Roboterarm befestigten Elemente um bis zu 20°.

Bei dem Prüfstand gibt es sehr viele Kollisionsmöglichkeiten. Es wäre sinnvoll, ein Konzept für den Schutz des Roboterarmes zu entwerfen. Das System könnte den Messplatz unabhängig von den befestigten Element vor einer Kollision schützen. Hierfür könnten zum Beispiel Lichtschranken eingesetzt werden, die allerdings für jeden neuen Messaufbau angepasst werden müssen. Ferner müsste eine Untersuchung hinsichtlich ihrer Genauigkeit stattfinden. Da hier in Bereichen von wenigen Grad und Mikro- bis Millimeter gearbeitet wird, sollte eine Sicherheitsabschaltung ebenfalls in diesen Bereichen messen können.

Literatur

[1] *ADT00X-10E Ultralow Power Rotation Sensors*. NVE Corporation. 2017. URL: https://www.nve.com/Downloads/ADT00x.pdf.

[2] *Alles-oder-nichts-Prinzip*. Zuletzt am 17 Februar 2015 bearbeitet. Wikipedia. URL: https://de.wikipedia.org/wiki/Alles-oder-nichts-Prinzip.

[3] *Anisotroper magnetoresistiver Effekt*. Zuletzt am 8 März 2017 bearbeitet. Wikipedia. URL: https://de.wikipedia.org/wiki/Anisotroper_magnetoresistiver_Effekt.

[4] *ARM® Cortex®-M4F-Based MCU TM4C1294 Connected LaunchPad™ Evaluation Kit*. Texas Instruments. URL: http://www.ti.com/tool/EK-TM4C1294XL.

[5] *FreeCAD*. 30.11.2017. URL: https://www.freecadweb.org/.

[6] Prof. Dr. U. Hartmann. *Magnetfeldsensoren*. 13.06.2000. URL: http://www.uni-saarland.de/fak7/hartmann/files/docs/pdf/teaching/advancedpractical/FoPra_Sensorik.pdf.

[7] K. Ivanov. „Fehlersichere Automatisierung eines Encoder-Messplatzes zur Untersuchung von ABS-Sensoren“. Diplomarbeit. 2011.

[8] *KMZ60 Angle sensor with integrated amplifier*. NXP Semiconductors. 2014. URL: https://www.nxp.com/docs/en/data-sheet/KMZ60.pdf.

[9] Mirco Marahrens. „Untersuchungen zu CoFeB/MgO/CoFeB - Tunnelmagnetowiderstandselementen“. Bachelorarbeit. 2012.

[10] S. Mengel. *Technologieanalyse Magnetismus Band 2 - XMR-Technologien*. VDI Technologiezentrum GmbH im Auftrag des BMBF. August 1997. URL: https://www.vditz.de/publikation/technologieanalyse-magnetismus-band-2-xmr-technologien/.

[11] Jürgen Moser. „TMR- und TAMR-Effekt beim Tunneln durch einkristalline GaAs-Barrieren“. Dissertation. 2007.

[12] *Piezo-Theorie*. piezosystem jena GmbH. URL: https://www.piezosystem.de/piezopedia/piezotheorie/.

[13] *Schalter (Elektrotechnik)*. Zuletzt am 5 September 2017 bearbeitet. Wikipedia. URL: https://de.wikipedia.org/wiki/Schalter_(Elektrotechnik).

[14] S. Schemjakin. *AMR-Winkelsensoren von Honeywell*. 2012. URL: http://www.kit-e.ru/articles/sensor/2012_11_24.php.

[15] Christian Schoermer. „Konstruktion und Automatisierung eines Radmessplatzes für ABS mit Encodern verschiedener Automobil-Hersteller“. Diplomarbeit. 2010.

[16] Thorben Schüthe. *personliche Kommunikation zu Signalverarbeitung vom Sensor-Array*.

[17] *Taste*. Zuletzt am 3 Juli 2017 bearbeitet. Wikipedia. URL: https://de.wikipedia.org/wiki/Taste.

[18] Thomas Tille. *Automobil-Sensorik*. Springer-Verlag, 2016.

[19] *TMR-Effekt: Ein Sandwich mit Gedächtnis*. Forschen in Jülich 2/2000. 23.04.2002. URL: https://www.weltderphysik.de/gebiet/stoffe/magnete/tmr-effekt/.

Anhang

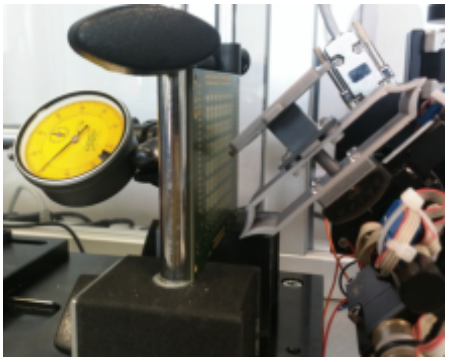
A Messergebnisse und Zeichnungen aus den Kollisionstests

Tabelle A.1: Ergebnisse der Experimente mit dem Piezo-Scheibe-System.

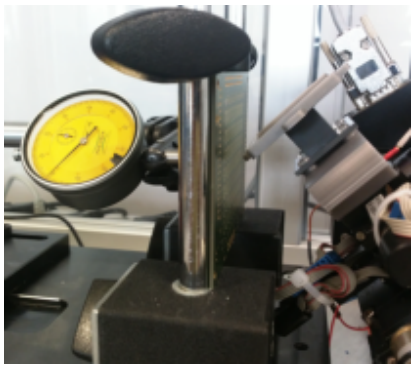
		Verbindungsschicht 4.4(c) ist waagerecht ausgerichtet			Verbindungsschicht 4.4(c) ist senkrecht ausgerichtet		
φ_y Messeinheiten	180° (N)	45° (mm)	90° (mm)	135° (mm)	45° (mm)	90° (mm)	135° (mm)
	6,288	0,240	0,125	0,220	0,270	0,130	0,225
	7,210	0,245	0,130	0,220	0,270	0,125	0,230
	7,152	0,240	0,130	0,220	0,260	0,130	0,230
	7,201	0,240	0,130	0,225	0,265	0,125	0,230
	7,201	0,240	0,125	0,220	0,265	0,130	0,225
	7,201	0,245	0,130	0,225	0,265	0,125	0,230
	7,181	0,240	0,130	0,220	0,265	0,125	0,230
	7,142	0,240	0,130	0,225	0,270	0,125	0,230
	7,112	0,240	0,130	0,220	0,270	0,125	0,230
	7,102	0,240	0,130	0,220	0,265	0,125	0,230

Tabelle A.2: Ergebnisse der Experimente mit dem Mikrotaster-System.

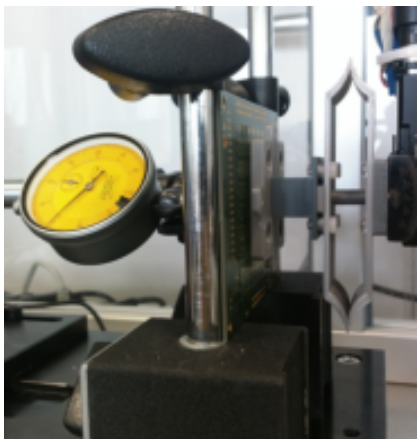
		Verbindungsschicht 4.4(c) ist waagerecht ausgerichtet			Verbindungsschicht 4.4(c) ist senkrecht ausgerichtet		
φ_y Messeinheiten	180° (N)	45° (mm)	90° (mm)	135° (mm)	45° (mm)	90° (mm)	135° (mm)
	6,533	0,270	0,130	0,260	0,270	0,130	0,280
	6,553	0,270	0,125	0,270	0,265	0,130	0,275
	6,857	0,270	0,130	0,270	0,265	0,120	0,275
	6,523	0,270	0,130	0,270	0,260	0,110	0,275
	7,269	0,265	0,135	0,270	0,260	0,110	0,275
	7,024	0,270	0,135	0,270	0,260	0,120	0,275
	6,995	0,270	0,135	0,270	0,265	0,120	0,275
	6,985	0,270	0,130	0,265	0,260	0,120	0,280
	6,965	0,270	0,135	0,270	0,260	0,120	0,275
	7,240	0,265	0,135	0,270	0,260	0,110	0,275



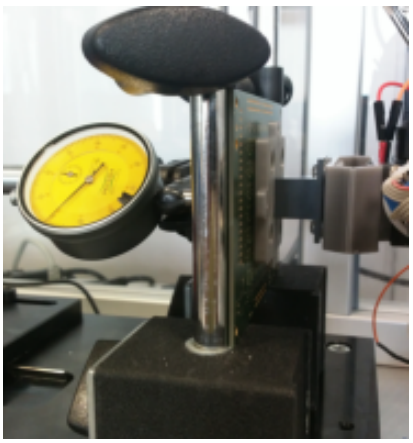
(a) Einstellwinkel φ_y = 45°, Verbindungsschicht ausgerichtet in Z-Richtung.



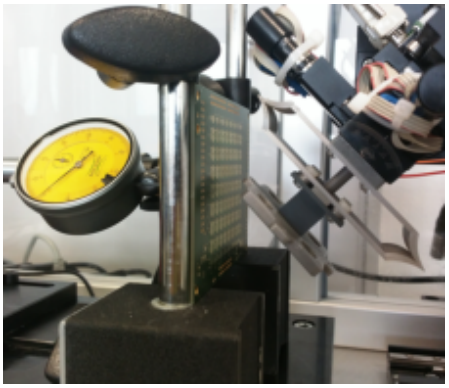
(b) Einstellwinkel φ_y = 45°, Verbindungsschicht ausgerichtet in Y-Richtung.



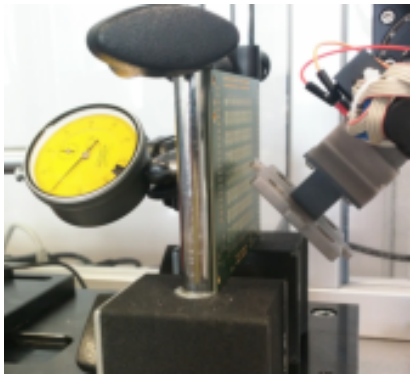
(c) Einstellwinkel φ_y = 90°, Verbindungsschicht ausgerichtet in Z-Richtung.



(d) Einstellwinkel φ_y = 90°, Verbindungsschicht ausgerichtet in Y-Richtung.



(e) Einstellwinkel φ_y = 135°, Verbindungsschicht ausgerichtet in Z-Richtung.



(f) Einstellwinkel φ_y = 135°, Verbindungsschicht ausgerichtet in Y-Richtung.

Abbildung A.1: Kollisionstests mit den unterschiedlichen Stoßwinkeln, die bei dem Roboterarm eingestellt werden.

B Auswertung

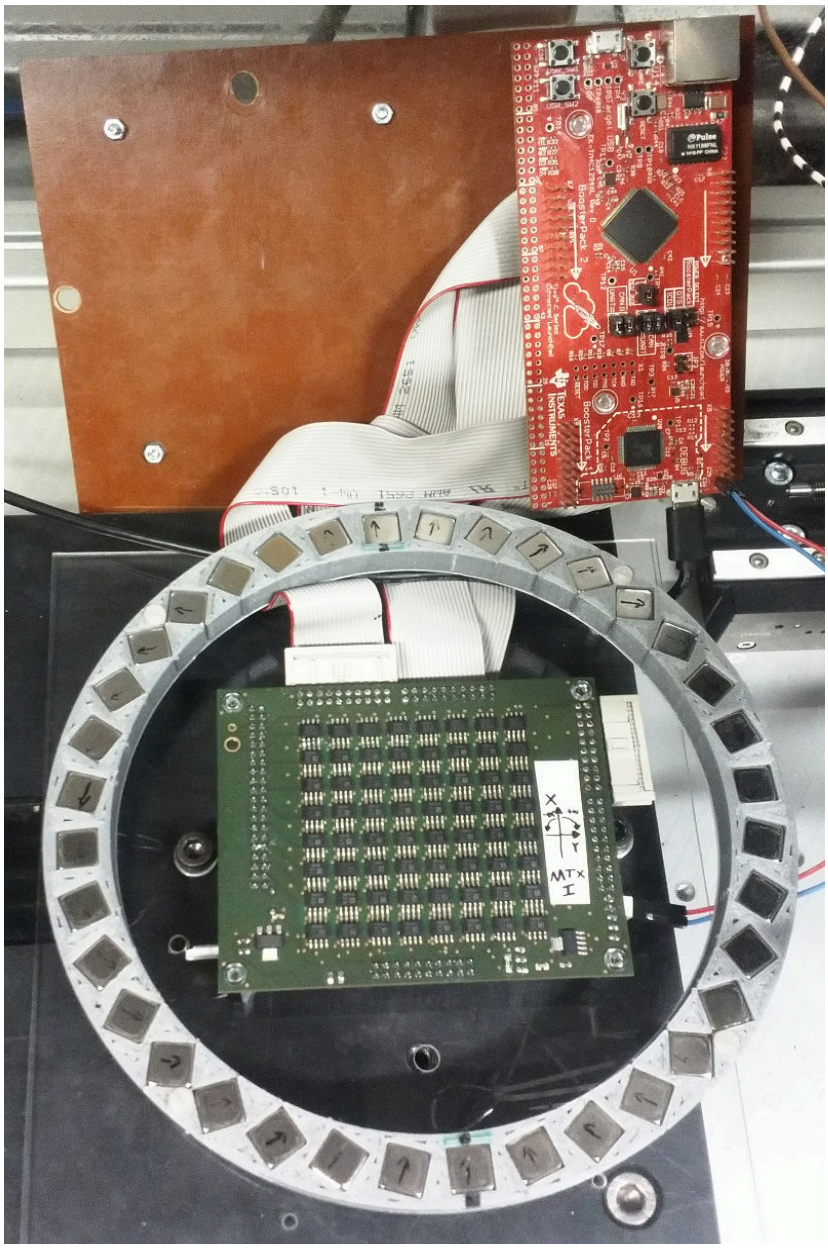


Abbildung B.1: Messaufbau des Versuches mit einem Störfeld.

Tabelle B.1: Die Änderung der magnetischen Feldstärke bei der Sättigungsuntersuchung.

AMR-Sensor mit Magnet 1 (40 mm × 40 mm × 10 mm)		TMR-Sensor mit Magnet 2 (10 mm × 10 mm × 10 mm)	
Abstand (mm)	Feldstärke (kA/m)	Abstand (mm)	Feldstärke (kA/m)
3	35,6	7	38,8
4	34,2	8	35,0
5	33,5	10	31,1
6	32,5	12	27,3
7	31,4	14	25,3
8	30,0	16	23,4
9	28,3	18	19,8
10	26,9	20	17,8
11	25,2	22	15,7
12	23,7	24	13,5
13	22,2	26	12,0
14	20,8	28	10,5
15	19,5	30	9,1
16	18,2	32	7,9
17	17,3	34	7,1
18	16,2	36	6,1
		38	5,4
		40	4,7
		42	4,3
		44	4,1
		46	3,7
		48	3,3
		50	2,9
		52	2,6
		54	2,3
		56	2,1
		58	1,9
		60	1,7

C Quellcode

Referenz C.20 zum Quellcode

Quellcode C.1: Inbetriebnahme.

```
%--------------------------------------------------------------------------
% Inbetriebnahme des Messplatzes
%
% Filename:       rmp_3_inbetriebnahme.m
% Autor:          Viktor Airich
% Datum:          03.11.2017
% Beschreibung:   Inbetriebnahme des Messplatzes,
%                 Schnittstelle zum Bussystem
%
%
%--------------------------------------------------------------------------
if ~exist('work')
    rmp_3_set_open_interface(); %Schnittstelle zum Bussystem
end;
rmp_3_menu_start; %Aufruf des Startmenüs
```

Quellcode C.2: Initialisierung.

```
%--------------------------------------------------------------------------
% Start Menu
%
% Filename:       rmp_3_menu_start.m
% Autor:          Viktor Airich
% Datum:          03.11.2017
% Beschreibung:   Menu fuer die initialisierung des Messplatzes.
%
%
%--------------------------------------------------------------------------

% Hilfsvariable fuer eingabe
input_modus = 0;
while (1)
    %Auswahlmenu
    input_modus = menu('Referenzfahrt durfueren?', ...
        'Referenzfahrt inkl. Initialisierungsfahrt', ...
        'Werte übernehmen', ...
        'EXIT');

    switch(input_modus) % switch axes

            case 1 % Initialisierung inkl. referenz
              clear input_modus; % clear input variable
              rmp_3_init_TMCM_modules; % initialisierungsfunktion
              rmp_3_init_stage_system; % initialisierungsfunktion
              rmp_3_refenrezfahrt; % referenzfahrt
              rmp_3_calc_stage_pos; % initialisierungsfunktion
              rmp_3_calc_wheel_hub_pos; % initialisierungsfunktion
              rmp_3_menu_modusauswahl; % aufruf menu relative positioning
              break;

            case 2
              clear input_modus; % clear input variable
              load('rmp_3_init_stage_pos'); % Initialisierungsdaten uebernahme
              rmp_3_menu_modusauswahl;
              break;

            case 3
              save('rmp_3_init_stage_pos','active_collision_limit',...
                   'active_motor','active_wheel_hub', 'collision_limit',...
                   'global_flags','global_flags','laenge_x','laenge_y',...
                   'laenge_z','motor_setup','specific_parameter',...
                   'stage_positioning','stage_setup','wheel_hub',...
                   'work');
              clear input_modus; % clear input variable
              disp('exit');
              break; % exit while loop
    end;
end;
```

Quellcode C.3: Referenzfahrt.

```
%--------------------------------------------------------------------------
% Referenzfahrt
%
% Filename:       rmp_3_menu_start.
% Autor:          Viktor Airich
% Datum:          03.11.2017
% Beschreibung:   Es wird eine Referenzfahrt und die Berechnung der
%                 befahrbaren Länge der Achsen in Einheiten mm durchgeführt.
%
%
%--------------------------------------------------------------------------
%% Move z-axis to right limit switch (motor side)
set(deviceObj, 'address', stage_setup.z_axis.module_address);
set(deviceObj, 'motor', stage_setup.z_axis.motor_address);
invoke(deviceObj, 'ror',...
    stage_setup.z_axis.maximum_positioning_speed); % rotate right
while(1)
    if( get(deviceObj.Axis,'right_limit_switch_status') == 1 )
        break;
    end
    pause(1);
end
invoke(deviceObj, 'mst'); % stop motor
pause(1);
%% Move x-axis to right limit switch (motor side)
set(deviceObj,'address', stage_setup.x_axis.module_address);
set(deviceObj,'motor', stage_setup.x_axis.motor_address);
invoke(deviceObj,'ror',...
    stage_setup.x_axis.maximum_positioning_speed); % rotate right
while(1)
    if( get(deviceObj.Axis,'right_limit_switch_status') == 1 )
        break;
    end
    pause(1);
end
invoke(deviceObj, 'mst'); % stop motor

pause(1);

%% referencing rotation stage DMT40-D20-HSM: yaw-axis (phi_z)
set(deviceObj, 'address', stage_setup.yaw_axis.module_address);
set(deviceObj, 'motor', stage_setup.yaw_axis.motor_address);
invoke(deviceObj, 'rfs', 'start'); % rotate right
disp('referencing rotation-stage DMT40-D20-HSM...');
while(1)
    if( invoke(deviceObj,'rfs','status') == 0 )
        break;
    end
    pause(1);
end;
invoke(deviceObj, 'mst'); % stop motor
stage_positioning.yaw_axis.initialisation_complete = 1;
disp('finished referencing');
pause(1);

% move yaw-axis to zero position
disp('rotating yaw-axis on zero degree orientation...');
rmp_3_move_abs_unidirect_pos( ...
    stage_positioning.yaw_axis.zero_position, ...
    stage_setup.yaw_axis.module_address, ...
    stage_setup.yaw_axis.motor_address, ...
    deviceObj, ...
    stage_setup, ...
```

```
64      stage_positioning);

66  stage_positioning.yaw_axis.actual_position = 0;

68  %% referencing alignment-axis DMT65-DM4-HSM
    set(deviceObj, 'address', stage_setup.alignment_axis.module_address);
70  set(deviceObj, 'motor', stage_setup.alignment_axis.motor_address);
    invoke(deviceObj, 'rfs', 'start'); % rotate right
72  disp('referencing rotation-stage DMT65-DM4-HSM...');
    while(1)
74      if( invoke(deviceObj,'rfs','status') == 0 )
            break;
76      end
        pause(1);
78  end
    invoke(deviceObj, 'mst'); % stop motor
80  stage_positioning.alignment_axis.initialisation_complete = 1;
    disp('finished referencing');
82  pause(1);
    % move alignment-axis to INITIAL position
84  set(deviceObj,'address', stage_setup.alignment_axis.module_address);
    set(deviceObj,'motor', stage_setup.alignment_axis.motor_address);
86  disp('move alignment-axis in inital position...');
    rmp_3_move_abs_unidirect_pos( ...
88      stage_positioning.alignment_axis.initial_position, ...
        stage_setup.alignment_axis.module_address, ...
90      stage_setup.alignment_axis.motor_address, ...
        deviceObj, ...
92      stage_setup, ...
        stage_positioning);
94
    stage_positioning.alignment_axis.actual_position = 0;
96
    %% referencing z-axis LTM80-150-HSM
98  set(deviceObj, 'address', stage_setup.z_axis.module_address);
    set(deviceObj, 'motor', stage_setup.z_axis.motor_address);
100 invoke(deviceObj, 'rfs', 'start'); % rotate right
    disp('referencing z-axis LTM80-150-HSM...');
102 while(1)
        if( invoke(deviceObj,'rfs','status') == 0 )
104         break;
        end
106     pause(1);
    end
108 invoke(deviceObj, 'mst'); % stop motor
    stage_positioning.z_axis.steps_distance_end_switches...
110     = get(deviceObj.Axis,...
        'steps_distance_end_switches'); % get switch distance
112 stage_positioning.z_axis.initialisation_complete = 1;
    disp('finished referencing z-axis');
114 pause(1);

116
    % Move z-axis to right limit switch (motor side)
118 set(deviceObj, 'address', stage_setup.z_axis.module_address);
    set(deviceObj, 'motor', stage_setup.z_axis.motor_address);
120 invoke(deviceObj, 'ror',...
        stage_setup.z_axis.maximum_positioning_speed); % rotate right
122 while(1)
        if( get(deviceObj.Axis,'right_limit_switch_status') == 1 )
124         break;
        end
126     pause(1);
    end
128 invoke(deviceObj, 'mst'); % stop motor
```

```
    pause(1);
130
    stage_positioning.z_axis.actual_position = 0;
132
    %% referencing y-axis LTM80-75-HSM
134 set(deviceObj, 'address', stage_setup.y_axis.module_address);
    set(deviceObj, 'motor', stage_setup.y_axis.motor_address);
136 invoke(deviceObj, 'rfs', 'start'); % rotate right
    disp('referencing y-axis LTM80-75-HSM...');
138 while(1)
        if( invoke(deviceObj,'rfs','status') == 0 )
140         break;
        end
142     pause(1);
    end
144 invoke(deviceObj, 'mst'); % stop motor
    stage_positioning.y_axis.steps_distance_end_switches...
146     = get(deviceObj.Axis,...
        'steps_distance_end_switches'); % get switch distance
148 stage_positioning.y_axis.initialisation_complete = 1;
    disp('finished referencing y-axis');
150
    pause(1);
152
    % move y-axis to right limit switch (motor side)
154 set(deviceObj,'address', stage_setup.y_axis.module_address);
    set(deviceObj,'motor', stage_setup.y_axis.motor_address);
156 disp('move y-axis in inital position...');
    invoke(deviceObj,'ror',...
158     stage_setup.y_axis.maximum_positioning_speed); % rotate right
    while(1)
160     if( get(deviceObj.Axis,'right_limit_switch_status') == 1 )
            break;
162     end
        pause(1);
164 end
    invoke(deviceObj, 'mst'); % stop motor
166 disp('y-axis in initial position');
    pause(1);
168
    % set actual pos on defined value (no wheel hub -> actual position = -1)
170 stage_positioning.y_axis.actual_position = 0;

172 %% referencing x-axis LTM80-150-HSM
    set(deviceObj, 'address', stage_setup.x_axis.module_address);
174 set(deviceObj, 'motor', stage_setup.x_axis.motor_address);
    invoke(deviceObj, 'rfs', 'start'); % rotate right
176 disp('referencing x-axis LTM80-150-HSM...');
    while(1)
178     if( invoke(deviceObj,'rfs','status') == 0 )
            break;
180     end
        pause(1);
182 end
    invoke(deviceObj, 'mst'); % stop motor
184 stage_positioning.x_axis.steps_distance_end_switches...
        = get(deviceObj.Axis,...
186     'steps_distance_end_switches'); % get switch distance
    stage_positioning.x_axis.initialisation_complete = 1;
188 disp('finished referencing x-axis');
    pause(1);
190
    % Move x-axis to right limit switch (motor side)
192 set(deviceObj,'address', stage_setup.x_axis.module_address);
    set(deviceObj,'motor', stage_setup.x_axis.motor_address);
```

```
194 invoke(deviceObj,'ror',...
        stage_setup.x_axis.maximum_positioning_speed); % rotate right
196 while(1)
        if( get(deviceObj.Axis,'right_limit_switch_status') == 1 )
198         break;
        end
200     pause(1);
    end
202 invoke(deviceObj, 'mst'); % stop motor
    pause(1);
204
    % set actual pos on defined value (no wheel hub -> actual position = -1)
206 stage_positioning.x_axis.actual_position = 0;

208 %% Berechnung der Länge von Achsen
    % Berechnung der x-Achse Laenge
210     laenge_x = (stage_positioning.x_axis.distance_per_step...
            * stage_positioning.x_axis.steps_distance_end_switches);
212     laenge_x = fix(1000*laenge_x)/10; % Die befahrbare Länge in m

214 % Berechnung der y-Achse Laenge
        laenge_y = (stage_positioning.y_axis.distance_per_step...
216         * stage_positioning.y_axis.steps_distance_end_switches);
        laenge_y = fix(1000*laenge_y)/10; % Die befahrbare Länge in m
218
    % Berechnung der z-Achse Laenge
220     laenge_z = (stage_positioning.z_axis.distance_per_step...
            * stage_positioning.z_axis.steps_distance_end_switches);
222     laenge_z = fix(1000*laenge_z)/10; % Die befahrbare Länge in mm

224 %% initial positions of x-, y- and z-axes
    % x-axis
226 stage_positioning.x_axis.initial_position = ...
        stage_positioning.x_axis.steps_distance_end_switches;
228 % y-axis
    stage_positioning.y_axis.initial_position = ...
230     stage_positioning.y_axis.steps_distance_end_switches;
    % z-axis
232 stage_positioning.z_axis.initial_position = ...
        stage_positioning.z_axis.steps_distance_end_switches;
234 %% set global flag if initialisation of measuring station is completed
    if(stage_positioning.x_axis.initialisation_complete && ...
236         stage_positioning.y_axis.initialisation_complete && ...
            stage_positioning.z_axis.initialisation_complete && ...
238         stage_positioning.yaw_axis.initialisation_complete && ...
            stage_positioning.alignment_axis.initialisation_complete)
240     % set global initialisation flag
        global_flags.initialisation_complete_flag = 1;
242
    else
244     % clear global initialisation flag
        global_flags.initialisation_complete_flag = 0;
246 end;
```

Quellcode C.4: Fahrt zur Nullposition.

```
%--------------------------------------------------------------------------
% Fahrt zur Nullposition
%
% Filename:       rmp_3_init_fahrt.m
% Autor:          Viktor Airich
% Datum:          03.11.2017
% Beschreibung:   Es wird zur Nullposition im angenommenen Koordinatensystem
%                 gefahren.
%
%
%--------------------------------------------------------------------------
%% move all stages to INITIAL position
disp('move all stages to INITIAL position');

% change on high positioning speed for x-axis
rmp_3_set_positioning_speed( ...
    stage_positioning.x_axis.high_positioning_speed, ...
    stage_setup.x_axis.motor_address, ...
    deviceObj, ...
    stage_setup, ...
    stage_positioning);

% change on high positioning speed for y-axis
rmp_3_set_positioning_speed( ...
    stage_positioning.y_axis.high_positioning_speed, ...
    stage_setup.y_axis.motor_address, ...
    deviceObj, ...
    stage_setup, ...
    stage_positioning);

% change on high positioning speed for z-axis
rmp_3_set_positioning_speed( ...
    stage_positioning.z_axis.high_positioning_speed, ...
    stage_setup.z_axis.motor_address, ...
    deviceObj, ...
    stage_setup, ...
    stage_positioning);

%% move z-axis to right limit switch (motor side)
set(deviceObj, 'address', stage_setup.z_axis.module_address);
set(deviceObj, 'motor', stage_setup.z_axis.motor_address);
disp('move z-axis in inital position...');
invoke(deviceObj, 'ror',...
    stage_setup.z_axis.maximum_positioning_speed); % rotate right
while(1)
    if( get(deviceObj.Axis,'right_limit_switch_status') == 1 )
        break;
    end
    pause(1);
end
invoke(deviceObj, 'mst'); % stop motor
disp('z-axis in initial position');
pause(1);

% set actual pos on defined value (no wheel hub -> actual position = -1)
stage_positioning.z_axis.actual_position = 0;

%% move x-axis to right limit switch (motor side)
set(deviceObj,'address', stage_setup.x_axis.module_address);
set(deviceObj,'motor', stage_setup.x_axis.motor_address);
disp('move x-axis in inital position...');
invoke(deviceObj,'ror',...
    stage_setup.x_axis.maximum_positioning_speed); % rotate right
```

```
64 while(1)
       if( get(deviceObj.Axis,'right_limit_switch_status') == 1 )
66         break;
       end
68     pause(1);
   end
70 invoke(deviceObj, 'mst'); % stop motor
   disp('x-axis in initial position');
72 pause(1);

74 % set actual pos on defined value (no wheel hub -> actual position = -1)
   stage_positioning.x_axis.actual_position = 0;
76
   %% move y-axis to right limit switch (motor side)
78 set(deviceObj,'address', stage_setup.y_axis.module_address);
   set(deviceObj,'motor', stage_setup.y_axis.motor_address);
80 disp('move y-axis in inital position...');
   invoke(deviceObj,'ror',...
82     stage_setup.y_axis.maximum_positioning_speed); % rotate right
   while(1)
84     if( get(deviceObj.Axis,'right_limit_switch_status') == 1 )
           break;
86     end
       pause(1);
88 end
   invoke(deviceObj, 'mst'); % stop motor
90 disp('y-axis in initial position');
   pause(1);
92
   % set actual pos on defined value (no wheel hub -> actual position = -1)
94 stage_positioning.y_axis.actual_position = 0;

96 %% move alignment-axis to INITIAL position
   if (stage_positioning.alignment_axis.actual_position ~= 0)
98
       set(deviceObj,'address', stage_setup.alignment_axis.module_address);
100    set(deviceObj,'motor', stage_setup.alignment_axis.motor_address);
       disp('move alignment-axis in inital position...');
102    rmp_3_move_abs_unidirect_pos( ...
           stage_positioning.alignment_axis.initial_position, ...
104        stage_setup.alignment_axis.module_address, ...
           stage_setup.alignment_axis.motor_address, ...
106        deviceObj, ...
           stage_setup, ...
108        stage_positioning);

110    stage_positioning.alignment_axis.actual_position = 0;
   end;
112
   %% move yaw-axis to zero position
114 set(deviceObj, 'address', stage_setup.yaw_axis.module_address);
   set(deviceObj, 'motor', stage_setup.yaw_axis.motor_address);
116 invoke(deviceObj, 'rfs', 'start'); % rotate right
   disp('referencing rotation-stage DMT40-D20-HSM...');
118 while(1)
       if( invoke(deviceObj,'rfs','status') == 0 )
120        break;
       end
122    pause(1);
   end;
124 invoke(deviceObj, 'mst'); % stop motor
   stage_positioning.yaw_axis.initialisation_complete = 1;
126 disp('finished referencing');
   pause(1);
128
```

```
    disp('rotating yaw-axis on zero degree orientation...');
130 rmp_3_move_abs_unidirect_pos( ...
        stage_positioning.yaw_axis.zero_position, ...
132     stage_setup.yaw_axis.module_address, ...
        stage_setup.yaw_axis.motor_address, ...
134     deviceObj, ...
        stage_setup, ...
136     stage_positioning);

138 % set actual pos on defined value (no wheel hub -> actual position = -1)
    stage_positioning.yaw_axis.actual_position = 0;
140
    %% initial positions of x-, y- and z-axes
142 % x-axis
    stage_positioning.x_axis.initial_position = ...
144     stage_positioning.x_axis.steps_distance_end_switches;

146 % y-axis
    stage_positioning.y_axis.initial_position = ...
148     stage_positioning.y_axis.steps_distance_end_switches;

150 % z-axis
    stage_positioning.z_axis.initial_position = ...
152     stage_positioning.z_axis.steps_distance_end_switches;
```

Quellcode C.5: Ansteuerungsmodus.

```
%--------------------------------------------------------------------------
% Menu Steuerungsmodus Aswahl
%
% Filename:       rmp_3_menu_modusauswahl.m
% Autor:          Viktor Airich
% Datum:          03.11.2017
% Beschreibung:   Menu fuer die Auswahl der Steuerungsmodi(hauptmenu).
%
%
%--------------------------------------------------------------------------
input_modus = 0; % Hilfsvariable fuer eingabe

while (1)
    %Auswahlmenu
    input_modus = menu('Waehlen Sie Modus aus', ...
            'Dierekte Ansteurung', ...
            'Fertige Positionskoordinaten', ...
            'Initialisierung des Scannverfahrens', ...
            'Start menu', ...
            'EXIT');

     switch(input_modus) % switch axes

         case 1
             clear input_modus; % clear input variable
             rmp_3_menu_manuelle_steuerung; % aufruf menu manuelle steuerung

         case 2
             clear input_modus; % clear input variable
             rmp_3_menu_koordinaten; % aufruf menu Positionsschablone

         case 3
             clear input_modus; % clear input variable
             rmp_3_menu_meas_save; % aufruf menu Messdatenaufnahme

         case 4
             clear input_modus; % clear input variable
             rmp_3_menu_start; % aufruf start menu

         case 5
              save('rmp_3_init_stage_pos','active_collision_limit',...
                  'active_motor','active_wheel_hub', 'collision_limit',...
                  'global_flags','global_flags','laenge_x','laenge_y',...
                  'laenge_z','motor_setup','specific_parameter',...
                  'stage_positioning','stage_setup','wheel_hub',...
                  'work');
             % clear all variables
             clear input_modus; % clear input variable
             disp('exit');
             break; % exit while loop
     end;
end;
```

Quellcode C.6: Manuelle Steuerung.

```
%--------------------------------------------------------------------------
% Manuelle Steuerung Menu
%
% Filename:       rmp_3_menu_manuelle_steuerung.m
% Autor:          Viktor Airich
% Datum:          03.11.2017
% Beschreibung:   Menu fuer die manuelle Steuerung der einzelnen
%                 Achsen, für die Abfrage der aktuellen Position und
%                 für den Initialisierungsfahrt.
%
%
%--------------------------------------------------------------------------
input_axis = 0; % Hilfsvariable fuer eingabe

while(1)
    % Auswahlmenu
    input_axis = menu('RELATIVE POSITIONING', ...
        'EXIT MODE', ...
        'x-axis', ...
        'y-axis', ...
        'z-axis', ...
        'alignment-axis', ...
        'yaw-axis', ...
        'Madus Auswahl',...
        'Aktuelle position vom Roboterarm',...
        'Initialisierungsfahrt');
%         'roll-axis',...
%         'pitch-axis');

    % check up on valid input argument
    switch(input_axis) % switch axes

        case 1 % Menu ausgang
            save('rmp_3_init_stage_pos','active_collision_limit',...
                'active_motor','active_wheel_hub', 'collision_limit',...
                'global_flags','global_flags','laenge_x','laenge_y',...
                'laenge_z','motor_setup','specific_parameter',...
                'stage_positioning','stage_setup','wheel_hub',...
                'work');
            clear input_axis; % clear input variable
            clear input_rel_pos; % clear input variable
            clear eingabe; % clear input variable
            disp('exit - relative positioning mode...');
            break; % exit while loop

        case 2 % Steuerung der X-Achse
           set(deviceObj,'address', ...
               stage_setup.x_axis.module_address); % module address
           set(deviceObj,'motor', ...
               stage_setup.x_axis.motor_address); % motor address

           % Ausgabe der aktuellen Position auf X-Achse
           fprintf(...
           'Aktuelle Position auf der X-Achse:%2.2f[mm] von %2.2f[mm]\n', ...
               stage_positioning.x_axis.actual_position / 100 * -1,...
               laenge_x*10); % actual position

            % Benutzereingabe
            eingabe = inputdlg({'Geben Sie die Laenge in mm (7.62): '},...
                'Eingabe für X-Achse');
            input_rel_pos = str2double(eingabe{1})*100 * -1;

            % check up on valid input argument
```

```
64              if(~(rmp_3_check_input_argument(input_rel_pos)))

66                  % Anlauf der Motor(X-Achse)
                    rmp_3_move_rel_unidirect_pos( ...
68                      input_rel_pos, ...
                        stage_setup.x_axis.module_address, ...
70                      stage_setup.x_axis.motor_address, ...
                        deviceObj, ...
72                      stage_setup,...
                        stage_positioning); % rel positioning
74
                    % Berechnung der aktuellen Position
76                  if (stage_positioning.x_axis.actual_position...
                            + input_rel_pos > 0)
78                      stage_positioning.x_axis.actual_position =  0;
                    elseif (stage_positioning.x_axis.actual_position...
80                          + input_rel_pos < -laenge_x*1000)

82                      stage_positioning.x_axis.actual_position...
                            =  -laenge_x*1000;
84                  else
                        stage_positioning.x_axis.actual_position...
86                          =  stage_positioning.x_axis.actual_position...
                            + input_rel_pos;
88                  end;

90                  % Ausgabe der aktuellen Position auf X-Achse
                    fprintf(...
92                  'Neue Position auf der X-Achse:%2.2f[mm] von %2.2f[mm]\n', ...
                    stage_positioning.x_axis.actual_position / 100 * -1,...
94                  laenge_x*10);
                end;
96
                clear eingabe; % clear input variable
98
            case 3 % Steuerung der Y-Achse
100             set(deviceObj,'address', ...
                    stage_setup.y_axis.module_address); % module address
102             set(deviceObj,'motor', ...
                    stage_setup.y_axis.motor_address); % motor address
104
               % Ausgabe der aktuellen Position auf Y-Achse
106            fprintf(...
               'Aktuelle Position auf der Y-Achse:%2.2f[mm] von %2.2f[mm]\n', ...
108                stage_positioning.y_axis.actual_position / 100,...
                   laenge_y * 10); % actual position
110
               % Benutzereingabe
112            eingabe = inputdlg({'Geben Sie die Laenge in mm (7.62): '},...
                    'Eingabe fÃ¼r Y-Achse');
114            input_rel_pos = str2double(eingabe{1})*100;

116            % check up on valid input argument
               if(~(rmp_3_check_input_argument(input_rel_pos)))
118
                   % Anlauf der Motor(Y-Achse)
120                rmp_3_move_rel_unidirect_pos( ...
                       input_rel_pos, ...
122                    stage_setup.y_axis.module_address, ...
                       stage_setup.y_axis.motor_address, ...
124                    deviceObj, ...
                       stage_setup,...
126                    stage_positioning);

128                % Berechnung der aktuellen Position
```

```
                if (stage_positioning.y_axis.actual_position...
130                     + input_rel_pos < 0)
                    stage_positioning.y_axis.actual_position =  0;
132
                elseif (stage_positioning.y_axis.actual_position...
134                     + input_rel_pos > laenge_y*1000)
                    stage_positioning.y_axis.actual_position...
136                     =  laenge_y * 1000;

138             else
                    stage_positioning.y_axis.actual_position...
140                     =  stage_positioning.y_axis.actual_position...
                        + input_rel_pos;
142             end;
                % Ausgabe der aktuellen Position auf Y-Achse
144             fprintf(...
                'Neue Position auf der Y-Achse:%2.2f [mm] von %2.2f [mm]\n', ...
146                 stage_positioning.y_axis.actual_position / 100,...
                    laenge_y * 10); % actual position
148         end;

150         clear eingabe; % clear input variable

152     case 4 % z-axis
            set(deviceObj,'address', ...
154             stage_setup.z_axis.module_address); % module address
            set(deviceObj,'motor', ...
156             stage_setup.z_axis.motor_address); % motor address

158         % Ausgabe der aktuellen Position auf Z-Achse
            fprintf(...
160         'Aktuelle Position auf der Z-Achse:%2.2f [mm] von %2.2f [mm]\n', ...
                stage_positioning.z_axis.actual_position / 100 * -1,...
162             laenge_z * 10); % actual position

164         % Benutzereingabe
            eingabe = inputdlg({'Geben Sie die Laenge in mm (7.62): '},...
166             'Eingabe fÃ¼r Z-Achse');
            input_rel_pos = str2double(eingabe{1})*100*-1;
168
            % check up on valid input argument
170         if(~(rmp_3_check_input_argument(input_rel_pos)))

172             % Anlauf der Motor(Z-Achse)
                rmp_3_move_rel_unidirect_pos( ...
174                 input_rel_pos, ...
                    stage_setup.z_axis.module_address, ...
176                 stage_setup.z_axis.motor_address, ...
                    deviceObj, ...
178                 stage_setup,...
                    stage_positioning); % rel positioning
180
                % Berechnung der aktuellen Position
182             if (stage_positioning.z_axis.actual_position...
                        + input_rel_pos > 0)
184                 stage_positioning.z_axis.actual_position =  0;

186             elseif (stage_positioning.z_axis.actual_position...
                        + input_rel_pos < -laenge_z*1000)
188                 stage_positioning.z_axis.actual_position...
                        =  -laenge_z * 1000;
190             else
                    stage_positioning.z_axis.actual_position...
192                     =  stage_positioning.z_axis.actual_position...
                        + input_rel_pos;
```

```
194                 end;
196                 % Ausgabe der aktuellen Position auf Z-Achse
                    fprintf(...
198                 'Neue Position auf der Z-Achse:%2.2f [mm] von %2.2f [mm]\n',...
                        stage_positioning.z_axis.actual_position / 100 * -1,...
200                     laenge_z * 10);
                end;
202
                clear eingabe; % clear input variable
204
            case 5 % alignment-axis
206
                set(deviceObj,'address',...
208                 stage_setup.alignment_axis.module_address); %module address
                set(deviceObj,'motor',...
210                 stage_setup.alignment_axis.motor_address); %motor address

212             % Ausgabe der aktuellen winkel Phi_Y
                fprintf('Aktuelle Winkel Phi_Y [mm]: %2.2f \n', ...
214                 stage_positioning.alignment_axis.actual_position / 10);

216             %Benutzereingabe
                eingabe = inputdlg(...
218             {'Geben Sie die Winkel Phi_Y (value x 1Â°):'},...
                    'Umdrehung um die Y-Achse');
220
                % Umrechnung der eingabe in Distanzschritten
222             input_rel_pos = str2double(eingabe{1})/ 0.1;

224             % Anlauf der motor(phi_y)
                rmp_3_move_rel_unidirect_pos(input_rel_pos, ...
226                 stage_setup.alignment_axis.module_address, ...
                    stage_setup.alignment_axis.motor_address, ...
228                 deviceObj, ...
                    stage_setup, ...
230                 stage_positioning);

232             % Berechnung der aktuellen Position
                stage_positioning.alignment_axis.actual_position...
234                 = stage_positioning.alignment_axis.actual_position...
                    + input_rel_pos;
236
                % Ausgabe der aktuellen winkel Phi_Y
238             fprintf('Neue Winkel Phi_Y [mm]: %2.2f \n', ...
                    stage_positioning.alignment_axis.actual_position...
240                 / 10); % actual position
                clear eingabe; % clear input variable
242
            case 6 % yaw-axis
244
                set(deviceObj,'address',...
246                 stage_setup.yaw_axis.module_address); % module address
                set(deviceObj,'motor',...
248                 stage_setup.yaw_axis.motor_address); % motor address

250             % Ausgabe der aktuellen winkel Phi_Z
                fprintf('Aktuelle Winkel Phi_Z [mm]: %2.2f \n', ...
252                 stage_positioning.yaw_axis.actual_position...
                    / 10); % actual position
254
                % Benutzereingabe
256             eingabe = inputdlg(...
                {'Geben Sie die Winkel Phi_Z (value x 1Â°):'},...
258                 'Umdrehung um die Z-Achse');
```

```
260             % Umrechnung der Eingabe in Distanzschritten
                input_rel_pos =str2double(eingabe{1})/ 0.1 * -1;
262
                % Anlauf der Motor(phi_z)
264             [active_collision_limit.add_tilt_dist] = ...
                    rmp_3_move_add_tilt_dist(...
266                 input_rel_pos, ...
                    stage_setup.yaw_axis.motor_address, ...
268                 deviceObj, ...
                    active_wheel_hub, ...
270                 active_collision_limit, ...
                    global_flags, ...
272                 stage_setup, ...
                    stage_positioning);
274
                % Berechnung der aktuellen Position
276             stage_positioning.yaw_axis.actual_position...
                    = stage_positioning.yaw_axis.actual_position...
278                 + input_rel_pos * -1;

280             % Ausgabe der aktuellen winkel Phi_Z
                   fprintf('Neue Winkel Phi_Z [mm]: %2.2f \n', ...
282                    stage_positioning.yaw_axis.actual_position...
                       / 10); % actual position
284             clear eingabe; % clear input variable

286         case 7
                clear input_axis; % clear input variable
288             clear input_rel_pos; % clear input variable
                clear eingabe; % clear input variable
290             rmp_3_menu_modusauswahl;
                break; % exit while loop
292
            case 8 % get actual relative positions of all axes
294
                fprintf('==========\n');
296             % x-axis
                % output relative position (what is the actual position?)
298             fprintf('Aktuelle Position auf der X-Achse [mm]: %2.2f \n', ...
                   stage_positioning.x_axis.actual_position / 100 * -1);
300
                % y-axis
302             % output relative position (what is the actual position?)
               fprintf('Aktuelle Position auf der Y-Achse [mm]: %2.2f \n', ...
304                stage_positioning.y_axis.actual_position / 100);

306             % z-axis
                % output relative position (what is the actual position?)
308             fprintf('Aktuelle Position auf der Z-Achse [mm]: %2.2f \n', ...
                   stage_positioning.z_axis.actual_position / 100 * -1);
310
                % alignment-axis
312             % Ausgabe der aktuellen winkel Phi_Y
                fprintf('Aktuelle Winkel Phi_Y: %2.2f \n', ...
314                     stage_positioning.alignment_axis.actual_position / 10)

316             % yaw-axis
                % Ausgabe der aktuellen winkel Phi_Z
318             fprintf('Aktuelle Winkel Phi_Z: %2.2f \n', ...
                        stage_positioning.yaw_axis.actual_position / 10)
320
                fprintf('==========\n');
322
            case 9
```

```
324             % clear all variables
                clear input_modus; % clear input variable
326             % aufruf des Initialisierungsfahrtes
                rmp3_init_fahrt();
328         otherwise
                disp('no valid argument...')
330         end; % end switch axes
        end;
```

Quellcode C.7: Übernahme der Initialisierungswerte.

```
%--------------------------------------------------------------------------
% Menu der Koordinaten
%
% Filename:       rmp_3_menu_koordinaten.m
% Autor:          Viktor Airich
% Datum:          03.11.2017
% Beschreibung:   Menu fuer die Auswahl der vorgespeicherten Koordinaten.
%
%
%--------------------------------------------------------------------------
input_position = 0; % Hilfsvariable fuer eingabe

while(1)
    % Auswahlmenu
    input_position = menu('Waehlen Sie eine Koordinatenschablone', ...
        'Mitte des Befestigungsbereiches auf der Grundplatte', ...
        'AMR - Array 8x8 (magnet auf dem Arm)', ...
        'Modus-Auswahl Menu',...
        'EXIT');
    switch(input_position)

        case 1
            clear input_position; % clear input variable
            rmp3_init_fahrt(); % aufruf der Initialisierungsfahrt
            %Mitte bei der AMR-ARRAY 8x8
            clear ('x_schritte', 'y_schritte', 'z_schritte', 'phi_y');
            x_schritte = -13805; % Die maximal mögliche Schrittenzahln 14700;
            y_schritte = 1237;   % Die maximal mögliche Schrittenzahln 7200;
            z_schritte = 7520; % Entspricht 2mm Abstand von der MTX
            phi_y = 1800; % Umdrehung um die Y-Achse
            rmp_3_positionierung;% start der Positionierung
            rmp_3_menu_modusauswahl; % aufruf menu modusauswahl

        case 2
            clear input_position; % clear input variable
            rmp3_init_fahrt(); % aufruf der Initialisierungsfahrt
            % Anfangsposition bei der X = 3, Y = 3 Positionierung für MTX 8x8
            clear ('x_schritte', 'y_schritte', 'z_schritte', 'phi_y');
            x_schritte = -14700; % Die maximal mögliche Schrittenzahln 14700;
            y_schritte = 100;   % Die maximal mögliche Schrittenzahln 7200;
            z_schritte = 2820; % Entspricht 2mm Abstand von der MTX
            phi_y = 1800; % Umdrehung um die Y-Achse
            rmp_3_positionierung; % start der Positionierung
            rmp_3_menu_modusauswahl; % aufruf menu modusauswahl

        case 3
            clear input_position; % clear input variable
            %call menu choose modus
            rmp_3_menu_modusauswahl;

        case 4
            save('rmp_3_init_stage_pos','active_collision_limit',...
                'active_motor','active_wheel_hub', 'collision_limit',...
                'global_flags','global_flags','laenge_x','laenge_y',...
                'laenge_z','motor_setup','specific_parameter',...
                'stage_positioning','stage_setup','wheel_hub',...
                'work');
            clear input_position; % clear input variable
            disp('exit');
            break; % exit while loop
    end;
end;
```

Quellcode C.8: Positionierung des Roboterarmes hinsichtlich vorgespeicherten Koordinaten.

```
%--------------------------------------------------------------------------
% Positionierung
%
% Filename:       rmp_3_positionierung.m
% Autor:          Viktor Airich
% Datum:          03.11.2017
% Beschreibung:   Die vorgespeicherte koordinatenschablone koennen
%                 direkt angefahren werden.
%
%
%
%--------------------------------------------------------------------------
%% alignment-axis in scannmodus stellen (phi_y)
rmp_3_move_rel_unidirect_pos(phi_y/0.1, ...
    stage_setup.alignment_axis.module_address, ...
    stage_setup.alignment_axis.motor_address, ...
    deviceObj, ...
    stage_setup, ...
    stage_positioning);
%Abspeichern der aktuallen Position
stage_positioning.alignment_axis.actual_position...
    = stage_positioning.alignment_axis.actual_position + phi_y;
% Ausgabe der aktuellen winkel Phi_Y
fprintf('Aktuelle Winkel Phi_Y: %2.2f \n', ...
        stage_positioning.alignment_axis.actual_position / 10)

%% X-Achse Positionierung
input_rel_pos = x_schritte;
disp('Bewegung des x-Achses in Anfangsposition fuer Scannmodus...');
% uni-directional positioning
rmp_3_move_rel_unidirect_pos( ...
  input_rel_pos, ...
  stage_setup.x_axis.module_address, ...
  stage_setup.x_axis.motor_address, ...
  deviceObj, ...
  stage_setup,...
  stage_positioning);
%Abspeichern der aktuallen Position
stage_positioning.x_axis.actual_position = input_rel_pos;
% Ausgabe der aktuellen Position auf X-Achse
fprintf(...
    'Aktuelle Position auf der X-Achse:%2.2f [mm] von %2.2f [mm]\n', ...
stage_positioning.x_axis.actual_position / 100 * -1,...
laenge_x*1000); % actual position
pause(1);

%% Y-Achse Positionierung
input_rel_pos = y_schritte;
set(deviceObj,'address', stage_setup.y_axis.module_address);
set(deviceObj,'motor', stage_setup.y_axis.motor_address);
disp('Bewegung des y-Achses in Anfangsposition fuer Scannmodus...');
invoke(deviceObj,'ror',...
    stage_setup.y_axis.maximum_positioning_speed); % rotate right
while(1)
    if( get(deviceObj.Axis,'right_limit_switch_status') == 1 )
        break;
    end
    pause(1)
end
invoke(deviceObj, 'mst'); % stop motor
% uni-directional positioning
rmp_3_move_rel_unidirect_pos( ...
```

```
    input_rel_pos, ...  % 3429 Mitte bei 9x9 AMR-Matrix
64  stage_setup.y_axis.module_address, ...
    stage_setup.y_axis.motor_address, ...
66  deviceObj, ...
    stage_setup,...
68  stage_positioning); % rel positioning
%Abspeichern der aktuallen Position
70 stage_positioning.y_axis.actual_position = input_rel_pos;
% Ausgabe der aktuellen Position auf Y-Achse
72 fprintf(...
      'Aktuelle Position auf der Y-Achse:%2.2f [mm] von %2.2f [mm]\n', ...
74   stage_positioning.y_axis.actual_position / 100,...
     laenge_y * 1000); % actual position
76 pause(1);

78 %% Z-Achse Positionierung
input_rel_pos = z_schritte * -1; % Die maximal mögliche Schrittenzahln;
80 disp('Bewegung des z-Achses in Anfangsposition fuer Scannmodus...');
% uni-directional positioning
82 rmp_3_move_rel_unidirect_pos( ...
    input_rel_pos, ...
84  stage_setup.z_axis.module_address, ...
    stage_setup.z_axis.motor_address, ...
86  deviceObj, ...
    stage_setup,...
88  stage_positioning); % rel positioning
%Abspeichern der aktuallen Position
90 stage_positioning.z_axis.actual_position = input_rel_pos;
% Ausgabe der aktuellen Position auf Z-Achse
92 fprintf(...
      'Aktuelle Position auf der Z-Achse:%2.2f [mm] von %2.2f [mm]\n', ...
94   stage_positioning.z_axis.actual_position / 100,...
     laenge_z * 1000); % actual position
```

Quellcode C.9: Messaufnahme durch eine Translation.

```
%--------------------------------------------------------------------------
% Verschiebugn des Roboterarmes
%
% Filename:       rmp_3_mtx_translation.m
% Autor:          Viktor Airich
% Datum:          03.11.2017
% Beschreibung:   automastisiertes Scannverfahren, Translation und Rotation
%                 vorhanden, wobei Rotation gleich null gesetzt werdewn darf
%
%
%--------------------------------------------------------------------------
clear ('rot_inf', 'si'); % clear input variable
%Benutzereingabe fuer scannverfahren
si = inputdlg({'Anzahl von Scanebenen angeben',...
    'Anzahl von Sensorreihen angeben (z.B. 4 = 4x4 Sensormatrix):',...
    'Abstand zwischen Scanpositionen in [mm] (z.B. 7.62):',...
    'Abstand zwischen Scanebenen in [mm] angeben (z.B. 5.5)',...
    'Anfangskoordinate fuer X',...
    'Anfangskoordinate fuer y',...
    'Abstand zwischen Sensoren (AMR=7.62; TMR=2.2; Unbekannt=1)'},...
    'ScanflÃ¤chedeminsion',...
    [1 50; 1 50; 1 50; 1 50; 1 50; 1 50; 1 60]);
limit = str2double(si{1})+1; % Maenge von Ebenen (Wert+1)
matrix = str2double(si{2}); % Groesse des Sensorarrays
time = 0.09; % Zeit des Scanns
h = str2double(si{4}); %Hoehe zwischen Array und Magnet
abstand = str2double(si{3}) * 100; %Abstand zwischen messungspunkten
                                   %(Wert x 10^-3mm = Wert in mm)
input_rel_pos = abstand; %Abstand zwischen sensoren
%Aktuelle Positionskoordinatenübernahme
x_schritte = stage_positioning.x_axis.actual_position;
y_schritte = stage_positioning.y_axis.actual_position;
z_schritte = stage_positioning.z_axis.actual_position;
% hilfsvariablen
x = str2double(si{5}); % Ausgabekoordinate des x-Axes
XX = 1; % hilfszaehler des x-Axes
y = str2double(si{6}); % Ausgabekoordinate des y-Axes
YY = 1; % hilfszaehler des y-Axes
z = 1; % hilfszaehler des z-Axes
scan = 1; %hilfsvariable fuer while-schleife
%% Datei Erstellung und messwerteaufnahme
rmp_3_datei_erstellung; % Datei Erstellung
rmp_3_mtx_rotation(); % Rotation und Messwerteaufnahme

%% Wiederholtes Scannverfahren
while scan == 1
    % Abfrage des Scannmodus
    if z == limit
        scan = 0;
    end;
    %% Bewegung in x-achse richtung
    if(~(rmp_3_check_input_argument(input_rel_pos)) && scan ~= 0)
    % Bewegung in die X-Richtung
    rmp_3_move_rel_unidirect_pos( ...
        input_rel_pos, ...
        stage_setup.x_axis.module_address, ...
        stage_setup.x_axis.motor_address, ...
        deviceObj, ...
        stage_setup,...
        stage_positioning);
        pause(time);

        % Gerade oder Ungerade Zahl
```

```
            if mod(YY,2)  % Anweisung ungerades Zahl
65              x = x + str2double(si{3})/str2double(si{7});
                XX = XX + 1;
67              % Abspeicherung der aktuellen Position
                stage_positioning.x_axis.actual_position...
69                  =  stage_positioning.x_axis.actual_position...
                    + input_rel_pos;
71          else % anweisung gerades Zahl
                x = x - str2double(si{3})/str2double(si{7});
73              XX = XX - 1;
                %Abspeicherung der aktuellen Position
75              stage_positioning.x_axis.actual_position...
                    =  stage_positioning.x_axis.actual_position...
77                  + input_rel_pos;
            end;
79      %% Datei Erstellung und messwerteaufnahme
        rmp_3_datei_erstellung; % Datei Erstellung
81      rmp_3_mtx_rotation(); % Rotation und Messwerteaufnahme

83      end;
        %% Bewegung in y-achse richtung
85      if (XX == 1 && YY ~= matrix || XX == matrix && YY ~= matrix) && scan ~= 0
            %Aenderung der Richtung
87          if input_rel_pos == -abstand
                input_rel_pos = abstand;
89          end
            % Bewegung in die Y-Richtung
91          rmp_3_move_rel_unidirect_pos( ...
                input_rel_pos, ...
93              stage_setup.y_axis.module_address, ...
                stage_setup.y_axis.motor_address, ...
95              deviceObj, ...
                stage_setup,...
97              stage_positioning);
            pause(time);
99          y = y + str2double(si{3})/str2double(si{7}); %Y-Achse Koordinate Aenderung
            YY = YY + 1;
101         %Abspeicherung der aktuellen Position
            stage_positioning.y_axis.actual_position...
103             =  stage_positioning.y_axis.actual_position...
                + input_rel_pos;
105         %% Datei Erstellung und messwerteaufnahme
            rmp_3_datei_erstellung; % Datei Erstellung
107         rmp_3_mtx_rotation(); % Rotation und Messwerteaufnahme

109     end;
        % vorzeichen als Angabe der Richtung
111     if XX == matrix
            input_rel_pos = input_rel_pos * -1;
113     end;
        %% Ebene komplet gescannt?
115     if (((XX == 1 && YY == matrix) && ~mod(matrix,2))...
              || ((XX == matrix && YY == matrix) && mod(matrix,2))) && scan ~= 0
117         disp('Bewegung␣des␣x-Achses␣in␣Anfangsposition␣beim␣Scannmodus...');
            % Bewegung in die X-Richtung
119         rmp_3_move_rel_unidirect_pos( ...
                (x_schritte - stage_positioning.x_axis.actual_position), ...
121             stage_setup.x_axis.module_address, ...
                stage_setup.x_axis.motor_address, ...
123             deviceObj, ...
                stage_setup,...
125             stage_positioning);
            %Abspeicherung der aktuellen Position
127         stage_positioning.x_axis.actual_position = x_schritte;
            pause(time);
```

```
129         % Bewegung in die Y-Richtung bis zu Nullstelle
            set(deviceObj,'address', stage_setup.y_axis.module_address);
131         set(deviceObj,'motor', stage_setup.y_axis.motor_address);
            disp('Bewegung des y-Achses in Anfangsposition beim Scannmodus...');
133         invoke(deviceObj,'ror',...
                stage_setup.y_axis.maximum_positioning_speed); % rotate right
135         while(1)
                if( get(deviceObj.Axis,'right_limit_switch_status') == 1 )
137                 break;
                end
139             pause(1)
            end
141         invoke(deviceObj, 'mst'); % stop motor
            % Bewegung in die Y-Richtung Auf die Scannposition
143         rmp_3_move_rel_unidirect_pos( ...
                y_schritte, ...
145             stage_setup.y_axis.module_address, ...
                stage_setup.y_axis.motor_address, ...
147             deviceObj, ...
                stage_setup,...
149             stage_positioning); % rel positioning
            pause(1);
151         x = str2double(si{5});
            XX = 1;
153         y = str2double(si{6});
            YY = 1;
155         input_rel_pos = abstand;
            %Abspeicherung der aktuellen Position
157         stage_positioning.y_axis.actual_position = y_schritte;
            %Abfrage, ob alles abgescannt wurde
159         if z == limit && scan ~= 0
                scan = 1;
161         end
            %% Abstand zwischen Magnet und Sensormatrix Veränderung
163         if(z ~= limit && scan ~= 0)
                z = z + 1;
165             if z ~= limit
                    disp('Veraenderung der Scannebene (z-Achse)...');
167                 %Verschiebung auf die Z-Achse
                    rmp_3_move_rel_unidirect_pos( ...
169                   h * 100, ...
                      stage_setup.z_axis.module_address, ...
171                   stage_setup.z_axis.motor_address, ...
                      deviceObj, ...
173                   stage_setup,...
                      stage_positioning); % rel positioning
175                 pause(time);
                    %Abspeicherung der aktuellen Position
177                 stage_positioning.z_axis.actual_position...
                        = stage_positioning.z_axis.actual_position...
179                     + (str2double(si{4}) * 100);
                    %% Datei Erstellung und messwerteaufnahme
181                 rmp_3_datei_erstellung; % Datei Erstellung
                    rmp_3_mtx_rotation(); % Rotation und Messwerteaufnahme
183
                end;
185         end;
        end;
187 end;
```

Quellcode C.10: Messaufnahme durch eine Rotation.

```
%--------------------------------------------------------------------------
% Rotation bei Messdatenaufnahme
%
% Filename:       rmp_3_mtx_rotation.m
% Autor:          Viktor Airich
% Datum:          03.11.2017
% Beschreibung:   Rotation bis zum belibigen Winkel, mehrmalige
%                 wiederholung ist möglich(z.B. Hysterese)
%
%
%--------------------------------------------------------------------------

% definition, wenn die Variable nich existiert
if ~exist('rot_inf')
    rot_inf = inputdlg(...
        {'Gewuenschter Endwinkel in Grad Degree eingeben:',...
        'Rotatinswinkel eingeben:',...
        'Anzahl von wiederholungen:'},...
        'Rotationsmetode',...
        [1 50; 1 50; 1 50;]);
end;
COS_SIG = zeros(8,8); % Erzeugung einer 8x8 matrix
SIN_SIG = zeros(8,8); % Erzeugung einer 8x8 matrix
n = 0; %name des mat-Files
% Endwinkel definition
end_alpha = round((str2double(rot_inf{1})/1.01012)*100)/100;
input_alpha = str2double(rot_inf{1});
schritt = 0; % hilfsvariable fuer for-schleife
% Definition eines winkels fuer eine Umdrehung
alpha_degree = str2double(rot_inf{2}) * 1.98 / 2;
input_step = 0;
wiederhol = 0; % hilfsvariable fuer Wiederholvorgang
% benutzereingabe fuer Wiederholvorgang
input_wiederhol = str2double(rot_inf{3});
faktor = 0.1; % Hilfsvariable fuer rotationsfall
alpha = 0; %

%% Aufnahme der Messungen

if input_alpha == 0
    rmp_3_messaufnahme;
else
while ( wiederhol ~= input_wiederhol)

    if ~mod(wiederhol,2)
       %% Referenzfahrt yaw-axis (phi_z)
        set(deviceObj, 'address', stage_setup.yaw_axis.module_address);
        set(deviceObj, 'motor', stage_setup.yaw_axis.motor_address);
        invoke(deviceObj, 'rfs', 'start'); % rotate right
        disp('referencing rotation-stage DMT40-D20-HSM...');
        while(1)
            if( invoke(deviceObj,'rfs','status') == 0 )
                break;
            end
            pause(1);
        end;
        invoke(deviceObj, 'mst'); % stop motor
        stage_positioning.yaw_axis.initialisation_complete = 1;
        disp('finished referencing');
        pause(1);
        % move yaw-axis to zero position
        disp('rotating yaw-axis on zero degree orientation...');
        rmp_3_move_abs_unidirect_pos( ...
```

```
                stage_positioning.yaw_axis.zero_position, ...
65              stage_setup.yaw_axis.module_address, ...
                stage_setup.yaw_axis.motor_address, ...
67              deviceObj, ...
                stage_setup, ...
69              stage_positioning);
            %Aktuelle Positionskoordinatenübernahme
71          stage_positioning.yaw_axis.actual_position = 0;
        end
73      %% die Rotation von Null bis zum Wunschswinkel
        for schritt = 0 : alpha_degree :  end_alpha
75          % move additional tilt distance
            [active_collision_limit.add_tilt_dist] = ...
77              rmp_3_move_add_tilt_dist(...
                input_step, ...
79              stage_setup.yaw_axis.motor_address, ...
                deviceObj, ...
81              active_wheel_hub, ...
                active_collision_limit, ...
83              global_flags, ...
                stage_setup, ...
85              stage_positioning);
            %Aktuelle Positionskoordinatenübernahme
87          stage_positioning.yaw_axis.actual_position ...
                = stage_positioning.yaw_axis.actual_position + input_step;
89          % umrechnung des eingegebenen winkel
            input_step = -alpha_degree./faktor;
91          %-------------------------------------------------
            rmp_3_messaufnahme;
93          n = n + 1;
        end
95  wiederhol = wiederhol + 1;
    faktor = faktor * -1;
97  input_step = 0;
    end
99  end
```

Quellcode C.11: Messaufnahme durch eine zweidimensionale Drehmatrix.

```
%--------------------------------------------------------------------------
% Drehmatrix fuer Enzelsensorversuch
%
% Filename:       rmp_3_mtx_translation.m
% Auto:           Viktor Airich
% Datum:          16.11.2017
% Beschreibung:   Ganze Matrix wird bezueglich ausgewaehlten Drehpunkt mit
%                 Hilfe von Rotationsmatrix verschoben
%
%
%--------------------------------------------------------------------------
clear('abstand','x','y','h', 'input_alpha', 'schritt')
matrix_rot = inputdlg(...
        {'Rotationswinkel:',...
        'Rotationspunkt X-Komponente',...
        'Rotationspunkt Y-Komponente',...
        'Demension des Arrays(z.B. 8 = 8x8):',...
        'Distanzschrittweite zwischen Messpunkten:',...
        'Abstand zwischen Scanebenen in [mm] angeben (z.B. 5.5)'},...
        'Rotationsmetode',...
        [1 50; 1 50; 1 50;1 50; 1 50; 1 50]);

abstand = str2double(matrix_rot{5}) * 100;
M = [str2double(matrix_rot{4}),str2double(matrix_rot{4})];
time = 0.15;
z = str2double(matrix_rot{6});
Pxy = [1
        1];
%Rotationspunkt
Zp = [str2double(matrix_rot{2})
      str2double(matrix_rot{3})];
%Rotationswinkel
input_alpha = str2double(matrix_rot{1});
%Rotationsmatrix
R = [cosd(input_alpha) -sind(input_alpha)
     sind(input_alpha)  cosd(input_alpha)];

x = Pxy(1);
y = Pxy(2);
% Ueberpruefung, ob scanflaeche in befahrbaren Flaeche liegt
for i = 1 : str2double(matrix_rot{4})*str2double(matrix_rot{4})

    IND = [i];
    [I,J] = ind2sub(M,IND);
    Pxy = [I
           J];
    P2xy = (R*(Pxy-Zp))+Zp;
    intel_x = (P2xy(1)-x) * abstand;
    intel_y = (P2xy(2)-y) * abstand;
    x = P2xy(1);
    y = P2xy(2);
    if i == 1
        x_schr = stage_positioning.x_axis.actual_position...
            + ((str2double(matrix_rot{4}) - 1) * abstand)*-1;
        x_schr = x_schr + intel_x;

        y_schr = stage_positioning.y_axis.actual_position...
        + ((str2double(matrix_rot{4}) - 1) * abstand)*-1;
        y_schr = y_schr + intel_y;
        if x_schr < -laenge_x*1000 || y_schr < 0
         disp(...
         'Der ausgewaehlter Bereich liegt ausserhalb der befarbahren Zone');
         z = msgbox(...
```

```
            'Der ausgewaehlter Bereich liegt ausserhalb der befarbahren Zone',...
65          'Error','error');
            uiwait(z);
67          return;
            end
69      else
            x_schr = x_schr + intel_x;
71          y_schr = y_schr + intel_y;
        end
73
        if x_schr < -laenge_x*1000 || y_schr < 0
75          disp(...
            'Der ausgewaehlter Bereich liegt ausserhalb der befarbahren Zone');
77          z = msgbox(...
            'Der ausgewaehlter Bereich liegt ausserhalb der befarbahren Zone',...
79          'Error','error');
            uiwait(z);
81          return;
        end
83  end

85  x_start_position = stage_positioning.x_axis.actual_position;
    y_start_position = stage_positioning.y_axis.actual_position;
87  %% Referenzfahrt yaw-axis (phi_z)
            set(deviceObj, 'address', stage_setup.yaw_axis.module_address);
89          set(deviceObj, 'motor', stage_setup.yaw_axis.motor_address);
            invoke(deviceObj, 'rfs', 'start'); % rotate right
91          disp('referencing rotation-stage DMT40-D20-HSM...');
            while(1)
93              if( invoke(deviceObj,'rfs','status') == 0 )
                    break;
95              end
                pause(1);
97          end;
            invoke(deviceObj, 'mst'); % stop motor
99          stage_positioning.yaw_axis.initialisation_complete = 1;
            disp('finished referencing');
101         pause(1);
            % move yaw-axis to zero position
103         disp('rotating yaw-axis on zero degree orientation...');
            rmp_3_move_abs_unidirect_pos( ...
105             stage_positioning.yaw_axis.zero_position, ...
                stage_setup.yaw_axis.module_address, ...
107             stage_setup.yaw_axis.motor_address, ...
                deviceObj, ...
109             stage_setup, ...
                stage_positioning);
111         %Aktuelle Positionskoordinatenübernahme
            stage_positioning.yaw_axis.actual_position = 0;
113  % Drehung der Drehteller in Scannposition
            [active_collision_limit.add_tilt_dist] = ...
115             rmp_3_move_add_tilt_dist(...
                input_alpha/-0.1, ...
117             stage_setup.yaw_axis.motor_address, ...
                deviceObj, ...
119             active_wheel_hub, ...
                active_collision_limit, ...
121             global_flags, ...
                stage_setup, ...
123             stage_positioning);
            %Aktuelle Positionskoordinatenübernahme
125         stage_positioning.yaw_axis.actual_position ...
                = stage_positioning.yaw_axis.actual_position + input_alpha/0.1;
127
    %% ---------------------------------------------------------------
```

```
% Bewegung in die X-Richtung zur X=1, Y=1 bezueglich Koordinatensystem
    rmp_3_move_rel_unidirect_pos( ...
        ((str2double(matrix_rot{4}) - 1) * abstand)*-1, ...
        stage_setup.x_axis.module_address, ...
        stage_setup.x_axis.motor_address, ...
        deviceObj, ...
        stage_setup,...
        stage_positioning);
    pause(time);
%     Abspeicherung der aktuellen Position
    stage_positioning.x_axis.actual_position...
        = stage_positioning.x_axis.actual_position...
        + ((str2double(matrix_rot{4}) - 1) * abstand)*-1;
    fprintf(...
        'Aktuelle Position auf der X-Achse:%2.2f[mm] von %2.2f[mm]\n', ...
        stage_positioning.x_axis.actual_position / 100 * -1,...
        laenge_x*10); % actual position


    % Bewegung in die Y-Richtung zur X=1, Y=1 bezueglich Koordinatensystem
    rmp_3_move_rel_unidirect_pos( ...
        ((str2double(matrix_rot{4}) - 1) * abstand)*-1, ...
        stage_setup.y_axis.module_address, ...
        stage_setup.y_axis.motor_address, ...
        deviceObj, ...
        stage_setup,...
        stage_positioning);
    pause(time);
    %Abspeicherung der aktuellen Position
    stage_positioning.y_axis.actual_position...
        = stage_positioning.y_axis.actual_position...
        + ((str2double(matrix_rot{4}) - 1) * abstand)*-1;
    fprintf(...
        'Aktuelle Position auf der Y-Achse:%2.2f[mm] von %2.2f[mm]\n', ...
        stage_positioning.y_axis.actual_position / 100,...
        laenge_y * 10); % actual position
%-------------------------------------------------------------------------
x = 1;
y = 1;
n = 0;
rmp_3_datei_erstellung;

for i = 1 : str2double(matrix_rot{4})*str2double(matrix_rot{4})
    IND = [i];
    [I,J] = ind2sub(M,IND);
    Pxy = [I
           J];
    P2xy = (R*(Pxy-Zp))+Zp;
    intel_x = (P2xy(1)-x) * abstand;
    intel_y = (P2xy(2)-y) * abstand;
    x = P2xy(1);
    y = P2xy(2);

    % Bewegung in die X-Richtung
    rmp_3_move_rel_unidirect_pos( ...
        intel_x, ...
        stage_setup.x_axis.module_address, ...
        stage_setup.x_axis.motor_address, ...
        deviceObj, ...
        stage_setup,...
        stage_positioning);
    pause(time);
    %Abspeicherung der aktuellen Position
        stage_positioning.x_axis.actual_position...
            = stage_positioning.x_axis.actual_position...
```

```
            + intel_x ;
195     fprintf(...
            'Aktuelle Position auf der X-Achse:%2.2f [mm] von %2.2f [mm]\n', ...
197         stage_positioning.x_axis.actual_position / 100 * -1,...
            laenge_x*10); % actual position
199     pause(time);
        % Bewegung in die Y-Richtung
201     rmp_3_move_rel_unidirect_pos( ...
            intel_y , ...
203         stage_setup.y_axis.module_address, ...
            stage_setup.y_axis.motor_address, ...
205         deviceObj, ...
            stage_setup,...
207         stage_positioning);
        pause(time);
209     %Abspeicherung der aktuellen Position
        stage_positioning.y_axis.actual_position...
211         = stage_positioning.y_axis.actual_position...
            + intel_y ;
213     fprintf(...
            'Aktuelle Position auf der Y-Achse:%2.2f [mm] von %2.2f [mm]\n', ...
215         stage_positioning.y_axis.actual_position / 100,...
            laenge_y * 10); % actual position
217
        rmp_3_messaufnahme;
219     n=n+1;

221     if Pxy(1) == str2double(matrix_rot{4})...
                && Pxy(2) == str2double(matrix_rot{4})
223         % Bewegung in die X-Richtung
            rmp_3_move_rel_unidirect_pos( ...
225             (stage_positioning.x_axis.actual_position...
                - x_start_position), ...
227             stage_setup.x_axis.module_address, ...
                stage_setup.x_axis.motor_address, ...
229             deviceObj, ...
                stage_setup,...
231             stage_positioning);
            %Abspeicherung der aktuellen Position
233         stage_positioning.x_axis.actual_position = x_start_position;
            pause(time);
235         % Bewegung in die Y-Richtung
            rmp_3_move_rel_unidirect_pos( ...
237             (y_start_position...
                - stage_positioning.y_axis.actual_position), ...
239             stage_setup.y_axis.module_address, ...
                stage_setup.y_axis.motor_address, ...
241             deviceObj, ...
                stage_setup,...
243             stage_positioning);
            %Abspeicherung der aktuellen Position
245         stage_positioning.y_axis.actual_position = y_start_position;
            pause(time);
247         fprintf(...
            'Aktuelle Position auf der X-Achse:%2.2f [mm] von %2.2f [mm]\n', ...
249         stage_positioning.x_axis.actual_position / 100 * -1,...
            laenge_x*10); % actual position
251         fprintf(...
            'Aktuelle Position auf der Y-Achse:%2.2f [mm] von %2.2f [mm]\n', ...
253         stage_positioning.y_axis.actual_position / 100,...
            laenge_y * 10); % actual position
255
        %% Referenzfahrt yaw-axis (phi_z)
257         set(deviceObj, 'address', stage_setup.yaw_axis.module_address);
            set(deviceObj, 'motor', stage_setup.yaw_axis.motor_address);
```

```
259             invoke(deviceObj, 'rfs', 'start'); % rotate right
                disp('referencing rotation-stage DMT40-D20-HSM...');
261             while(1)
                    if( invoke(deviceObj,'rfs','status') == 0 )
263                     break;
                    end
265                 pause(1);
                end;
267             invoke(deviceObj, 'mst'); % stop motor
                stage_positioning.yaw_axis.initialisation_complete = 1;
269             disp('finished referencing');
                pause(1);
271             % move yaw-axis to zero position
                disp('rotating yaw-axis on zero degree orientation...');
273             rmp_3_move_abs_unidirect_pos( ...
                    stage_positioning.yaw_axis.zero_position, ...
275                 stage_setup.yaw_axis.module_address, ...
                    stage_setup.yaw_axis.motor_address, ...
277                 deviceObj, ...
                    stage_setup, ...
279                 stage_positioning);
                %Aktuelle Positionskoordinatenübernahme
281             stage_positioning.yaw_axis.actual_position = 0;

283         end
        end
```

Quellcode C.12: Erstellung einer Datei für Messergebnisse.

```
%--------------------------------------------------------------------------
% Datei Erstellung
%
% Filename:       rmp_3_datei_erstellung.m
% Autor:          Viktor Airich
% Datum:          03.11.2017
% Beschreibung:   Es wird ein Datei fuer die Messwerteaufzeinung erstellt
%                 und Messwerteaufzeichnungsfunktion aufgerufen, dabei wird
%                 aktuelle Position des roboterarmes bekann gegeben
%
%
%--------------------------------------------------------------------------
if ~exist('dat_name')
    dat_name = inputdlg(...
        {'Dateiname angeben (Datum wird dazu geschrieben)'},...
        'Dateiname Definition',...
        [1 60]);
end
% Aktuelle Position als Dateinamensende
datum = date; %Aktuelle datum
%Bindestrich mit Unterstrich ersetzen
ix = strfind(datum,'-');
datum(ix) = '_';
koordinaten = sprintf(...
    '_%s_X_%01.2f_Y_%01.2f_Z_%01.2f_in_%01.2f_mm_Schritten/',...
    datum,x,y,z,abstand/100);
dateiname = [dat_name{1} koordinaten]; % Genirierung der name fuer Datei
mkdir (dateiname); % Erstellung einer Datei
    %Aktuelle Position auf X-Achse
    fprintf(...
      'Aktuelle Position auf der X-Achse: %2.2f [mm] von %2.2f [mm] \n',...
              stage_positioning.x_axis.actual_position / 100 * -1,...
              laenge_x*10);
    %Aktuelle Position auf Y-Achse
   fprintf(...
      'Aktuelle Position auf der Y-Achse: %2.2f [mm] von %2.2f [mm] \n',...
       stage_positioning.y_axis.actual_position / 100,...
       laenge_y * 10); % actual position
  %Aktuelle Position auf Z-Achse
    fprintf(...
      'Aktuelle Position auf der Z-Achse: %2.2f [mm] von %2.2f [mm] \n',...
       stage_positioning.z_axis.actual_position / 100 * -1,...
       laenge_z * 10); % actual position
disp('============================================================');
```

Quellcode C.13: Erstellung einer Messprotokoll.

```
%--------------------------------------------------------------------------
% Messprotokoll
%
% Filename:       rmp_3_messprotokoll.m
% Autor:          Viktor Airich
% Datum:          03.11.2017
% Beschreibung:   Erstellung eines Messprotokolls, wird in einem
%                 extra Fenster ausgeführt
%
%
%--------------------------------------------------------------------------
A{1,1} = 'Datum';
A{2,1} = 'Name';
A{3,1} = 'Beschreibung der Messung';
A{4,1} = 'Rotationsrichtung';
A{5,1} = 'Welche Sensoratrix';
A{6,1} = 'Microkontroller';
A{7,1} = 'Magnetdimensionen';
A{8,1} = 'Minimale Feldstärke bei Nullposition in kA/m';
A{9,1} = 'Maximale Feldstärke bei Nullposition in kA/m';


file_name = inputdlg('Name der Datei eingeben mit .txt','Dateiname',[1 30]);
B = inputdlg({'Datum:',...
    'Name:'...
    'Messung:'...
    'Rotationsrichtung:',...
    'Welche Sensormatrix:'...
    'Welche Mikrocontroller:'...
    'Magnetdimensionen:'...
    'Minimale Feldstärke bei Nullposition in kA/m:'...
    'Maximale Feldstärke bei Nullposition in kA/m:'...
    },...
    'Messprotokollinfo', [1 15; 1 15; 1 50; 1 20;...
                          1 50; 1 50; 1 50; 1 50; 1 50;]);
file = fopen(file_name{1}, 'w');
for n = 1:numel(B)
    fprintf(file, '%s:\t%s\n', A{n},B{n} );
end
fclose(file);
```

Quellcode C.14: Menü für die Messaufnahme.

```
%--------------------------------------------------------------------------
% Menu Messdatenaufnahme
%
% Filename:       rmp_3_menu_meas_save.m
% Autor:          Viktor Airich
% Datum:          03.11.2017
% Beschreibung:   Menu fuer die Darstellung und Speicherung der Messdaten.
%                 Scannfahrt wird hier initialisiert und ein Messprotokoll
%                 erstellt.
%
%
%--------------------------------------------------------------------------
input_meas = 0; % Hilfsvariable fuer eingabe

while (1)
    % Auswahlmenu
    input_meas = menu('Waehlen Sie Modus aus', ...
        'Realtime-Analyse', ...
        'Translation / Translation und Rotation', ...
        'Rotation ueber den Array', ...
        'Rotation mit einem einzelnen Sensor',...
        'Messprotokollerstellung', ...
        'Modus-Auswahl Menu', ...
        'EXIT');
    switch(input_meas) % switch axes
        case 1
            % clear all variables
            clear input_meas; % clear input variable
            rmp_3_darstellung; % Aufruf der Darstellung
        case 2
            % clear all variables
            clear ('input_meas','rot_inf',...
               'si', 'dat_name'); % clear input variable
            rmp_3_mtx_translation; % aufruf Messdatenaufnahme
                                   % bei Translation mit/ohne Rotation
            rmp_3_messprotokoll; % Messprotokollerstellung

        case 3
            clear ('input_meas', 'rot_inf', 'si',...
                'matrix_rot','dat_name'); % clear input variable
            if ~exist('x')||('z')||('y')||('abstand')
                koor_abf = inputdlg({'Geben Sie Koordinate fuer X:',...
                    'Geben Sie Koordinate fuer Y:',...
                    'Geben Sie Koordinate fuer Z:',...
                    'Abstand zwischen Scanpositionen in [mm] (z.B. 7.62):'},...
                    'Koordinaten festlegung',...
                    [1 25;1 25;1 25;1 25]);
                x = str2double(koor_abf{1});
                y = str2double(koor_abf{2});
                z = str2double(koor_abf{3});
                abstand = str2double(koor_abf{4});
            end;
            rmp_3_datei_erstellung;
            rmp_3_mtx_rotation; % aufruf Messdatenaufnahme bei Rotation
            rmp_3_messprotokoll; % Messprotokollerstellung


        case 4
            clear ('input_meas', 'rot_inf', 'si',...
                'matrix_rot','dat_name'); % clear input variable
            rmp_3_einzelnsensor_drehmatrix;
            rmp_3_messprotokoll; % Messprotokollerstellung
            clear('dat_name', 'n');
```

```
64
            case 5
66              clear input_meas; % clear input variable
                rmp_3_messprotokoll; % Messprotokollerstellung
68
            case 6
70              clear input_position; % clear input variable
                rmp_3_menu_modusauswahl; %aufruf menu modusaswahl
72
             case 7
74               save('rmp_3_init_stage_pos','active_collision_limit',...
                      'active_motor','active_wheel_hub', 'collision_limit',...
76                    'global_flags','global_flags','laenge_x','laenge_y',...
                      'laenge_z','motor_setup','specific_parameter',...
78                    'stage_positioning','stage_setup','wheel_hub',...
                      'work');
80              clear input_meas; % clear input variable
                disp('exit');
82              break; % exit while loop
        end;
84  end;
```

Quellcode C.15: Datendarstellung im laufenden Betrieb.

```
%--------------------------------------------------------------------------
% Real-Time Analyse
%
% Filename:       rmp_3_darstellung.m
% Autor:          Thorben Schüthe
% Datum:          05.05.2017
% Modifiziert:    Viktor Airich
% Datum:          03.11.2017
% Beschreibung:   Dabei kann Echtzeitbeobachtungsfunktion ueber menue
%                 initialisiert werden. Die Daten werden im Echtzeitmodus
%                 grafisch dargestellt.
%
% P.s.            Skript wurde auf Linux Ubuntu erstellt
%
%--------------------------------------------------------------------------
clear dar; % LÃ¶schen von Eingfabevariable
colormap jet
COS_SIG = zeros(8,8); % Erzeugung einer 8x8 matrix
SIN_SIG = zeros(8,8); % Erzeugung einer 8x8 matrix
schritt = 0;
% Initialisierung der Darstellung
if (~exist('dar'))
    dar = inputdlg({'Wieviele Zyklen',...
          'Zeitintervall für ein Zyklus(Eingabe in Sekunden)',...
          'Rohdaten darstellen = 1; Umgerechnete Daten(Analog) = 2'},...
          'Zyklusinitialisierung',...
          [1 50;1 50; 1 50]);
end;
% Darstellung der Digitalen Messwerte
if str2double(dar{3}) == 1
    for n = 1 : str2double(dar{1})
        try
          % Serielledatenübertargung
          [s,cmdout] = system('echo -ne "-blk" > /dev/ttyACM0','-echo');
          clear('s','cmdout')
          [s,cmdout] = system('echo -ne "-test" > /dev/ttyACM0','-echo');
          clear('s','cmdout')
          [s,cmdout] = system('timeout 1 cat < /dev/ttyACM0');
          C = strsplit(cmdout,'break\n');
          % Aufnahme der Messdaten in digitaler Form
          SIN_SIG = str2num(C{1,1});
          COS_SIG = str2num(C{1,2});
          clear('C','SIG','cmdout')
        catch
        end
        %Grafische Darstellung der Messwerten
        subplot(1,3,1)
            imagesc(COS_SIG)
            set(gca,'DataAspectRatio',[1 1 1],'YDir','normal')
            title('cosine signal','interpreter','latex')
            colorbar
        subplot(1,3,2)
            imagesc(SIN_SIG)
            title('sine signal','interpreter','latex')
            set(gca,'DataAspectRatio',[1 1 1],'YDir','normal')
            colorbar
        subplot(1,3,3)
            % Kreisdarstellung der Digital/Analog umgerechnete Werte
            scatter(COS_SIG(:)./4096.*3.3,SIN_SIG(:)./4096.*3.3)
            hold on
            xlim([0 3.3])
            ylim([0 3.3])
            axis square
```

```
64              drawnow
                pause(str2double(dar{2}));
66      end
    % Darstellung der Analogen Messwerte
68  elseif str2double(dar{3}) == 2
        for n = 1 : str2double(dar{1})
70          try
              % Serielledatenübertargung
72            [s,cmdout] = system('echo -ne "-test" > /dev/ttyACM0','-echo');
              clear('s','cmdout')
74            [s,cmdout] = system('timeout 0.2 cat < /dev/ttyACM0');
              C = strsplit(cmdout,'break\n');
76            % Aufnahme der Messdaten in digitaler Form
              SIN_SIG = str2num(C{1,1});
78            COS_SIG = str2num(C{1,2});
              SIN_SIG_f   =  SIN_SIG./4095.*3.3;
80            COS_SIG_f   =  COS_SIG./4095.*3.3;
              clear('C','SIG','cmdout')
82          catch
            end
84      % Plot data from measurement
            subplot(1,3,1)
86          imagesc(COS_SIG_f)
            set(gca,'DataAspectRatio',[1 1 1],'YDir','normal')
88          title('cosine signal','interpreter','latex')
            caxis([1 3.3])
90          colorbar
            subplot(1,3,2)
92          imagesc(SIN_SIG_f)
            title('sine signal','interpreter','latex')
94          caxis([1 3.3])
            set(gca,'DataAspectRatio',[1 1 1],'YDir','normal')
96          colorbar
            subplot(1,3,3)
98          % Kreisdarstellung der Digital/Analog umgerechnete Werte
            scatter(COS_SIG(:)./4096.*3.3,SIN_SIG(:)./4096.*3.3)
100         hold on
            xlim([0 3.3])
102         ylim([0 3.3])
            axis square
104         drawnow
            pause(str2double(dar{2}));
106     end
    end;
```

Quellcode C.16: Menü für die Auswahl der vorgespeicherten Koordinaten.

```
%--------------------------------------------------------------------------
% Menu der Koordinaten
%
% Filename:       rmp_3_menu_koordinaten.m
% Autor:          Viktor Airich
% Datum:          03.11.2017
% Beschreibung:   Menu fuer die Auswahl der vorgespeicherten Koordinaten.
%
%
%--------------------------------------------------------------------------
input_position = 0; % Hilfsvariable fuer eingabe

while(1)
    % Auswahlmenu
    input_position = menu('Waehlen Sie eine Koordinatenschablone', ...
        'Mitte des Befestigungsbereiches auf der Grundplatte', ...
        'AMR - Array 8x8 (magnet auf dem Arm)', ...
        'Modus-Auswahl Menu',...
        'EXIT');
    switch(input_position)

        case 1
            clear input_position; % clear input variable
            rmp3_init_fahrt(); % aufruf der Initialisierungsfahrt
            %Mitte bei der AMR-ARRAY 8x8
            clear ('x_schritte', 'y_schritte', 'z_schritte', 'phi_y');
            x_schritte = -13805; % Die maximal mögliche Schrittenzahln 14700;
            y_schritte = 1237;   % Die maximal mögliche Schrittenzahln 7200;
            z_schritte = 7520; % Entspricht 2mm Abstand von der MTX
            phi_y = 1800; % Umdrehung um die Y-Achse
            rmp_3_positionierung;% start der Positionierung
            rmp_3_menu_modusauswahl; % aufruf menu modusauswahl

        case 2
            clear input_position; % clear input variable
            rmp3_init_fahrt(); % aufruf der Initialisierungsfahrt
            % Anfangsposition bei der X = 3, Y = 3 Positionierung für MTX 8x8
            clear ('x_schritte', 'y_schritte', 'z_schritte', 'phi_y');
            x_schritte = -14700; % Die maximal mögliche Schrittenzahln 14700;
            y_schritte = 100;   % Die maximal mögliche Schrittenzahln 7200;
            z_schritte = 2820; % Entspricht 2mm Abstand von der MTX
            phi_y = 1800; % Umdrehung um die Y-Achse
            rmp_3_positionierung; % start der Positionierung
            rmp_3_menu_modusauswahl; % aufruf menu modusauswahl

        case 3
            clear input_position; % clear input variable
            %call menu choose modus
            rmp_3_menu_modusauswahl;

        case 4
            save('rmp_3_init_stage_pos','active_collision_limit',...
                'active_motor','active_wheel_hub', 'collision_limit',...
                'global_flags','global_flags','laenge_x','laenge_y',...
                'laenge_z','motor_setup','specific_parameter',...
                'stage_positioning','stage_setup','wheel_hub',...
                'work');
            clear input_position; % clear input variable
            disp('exit');
            break; % exit while loop
    end;
end;
```

Quellcode C.17: Darstellung der Kollisionstestergebnisse.

```
%-------------------------------------------------------------------------
% Darstellung der Crash-Test Ergebnisse
%
% Filename:      rmp_3_crash_plot.m
%
% Beschreibung:  Es werde die aufgenommenen Kollisionstestergebnisse
%                dargestellt.
%
%
%-------------------------------------------------------------------------
x = 1:1:10;
y1 = [6.533 6.553 6.857 6.523 7.269 7.024 6.995 6.985 6.965 7.240];
y2 = [6.288 7.210 7.152 7.201 7.201 7.201 7.181 7.142 7.112 7.102];

% alpha_45_y_richt = [27 27 27 27 26.5 27 27 27 27 26.5];
% alpha_45_y_richt = (alpha_45_y_richt * 400/20)/1000 * 9.81;
% alpha_45_z_richt = [27 26.5 26.5 26 26 26 26.5 26 26 26];
% alpha_45_z_richt = (alpha_45_z_richt * 400/20)/1000 * 9.81;

alpha_90_y_mts = [13 12.5 13 13 13.5 13.5 13.5 13 13.5 13.5];
alpha_90_y_mts_mm = alpha_90_y_mts * 0.011;
alpha_90_y_mts = (alpha_90_y_mts * 400/20 )/1000 * 9.81;
alpha_90_y_pss = [12.5 13 13 13 12.5 13 13 13 13 13];
alpha_90_y_pss_mm = alpha_90_y_pss * 0.011;
alpha_90_y_pss = (alpha_90_y_pss * 400/20) /1000 * 9.81;

alpha_90_z_mts = [13 13 12 11 11 12 12 12 12 11];
alpha_90_z_mts_mm = alpha_90_z_mts * 0.011;
alpha_90_z_mts = (alpha_90_z_mts * 400/20 )/1000 * 9.81;
alpha_90_z_pss = [13 12.5 13 12.5 13 12.5 12.5 12.5 12.5 12.5];
alpha_90_z_pss_mm = alpha_90_z_pss * 0.011;
alpha_90_z_pss = (alpha_90_z_pss * 400/20) /1000 * 9.81;

alpha_135_y_mts = [26 27 27 27 27 27 27 26.5 27 27];
alpha_135_y_mts_mm = alpha_135_y_mts * 0.011;
alpha_135_y_mts = (alpha_135_y_mts*400/20)/1000 * 9.81;
alpha_135_y_pss = [22 22 22 22.5 22 22.5 22 22.5 22 22];
alpha_135_y_pss_mm = alpha_135_y_pss * 0.011;
alpha_135_y_pss = (alpha_135_y_pss*400/20)/1000 * 9.81;



h = legend('string1', 'string2','string3', 'string4','string5', 'string6')

%%
set(0,'DefaultAxesTitleFontWeight','normal');
hFig = figure(1); % new figure
set(hFig,'Units','centimeters');

ax1 = subplot(3,1,1); % top subplot
plot(x,y1,'m-o', x,y2,'m--','linewidth',1);
title ('Test mit der Waage')
ylabel('Kraft in N')
xlabel('Versuch')
legend('MTS bei Alpha 180°', ' PSS bei Alpha 180°','Location','southEast');
hold on
grid on
save2tikz('crashPlot',0)
%%
ax2 = subplot(3,1,2); % bottom subplot
plot(ax2,x, alpha_135_y_mts_mm, 'r-o',...
        x,alpha_135_y_pss_mm, 'r--',...
        x, alpha_90_z_mts_mm, 'b-o',...
```

```
64          x, alpha_90_z_pss_mm, 'b--',...
            'linewidth',1.5)
66  title (ax2, 'Test mit der Platine','fontweight','normal')
    ylabel(ax2, 'Strecke in mm')
68  xlabel(ax2, 'Versuch')
    grid on
70
    ax3 = subplot(3,1,3); % bottom subplot
72  plot(x, alpha_135_y_mts, 'r-o', x,alpha_135_y_pss, 'r--',...
         x, alpha_90_z_mts, 'b-o', x, alpha_90_z_pss, 'b--',...
74       'linewidth',1.5);
    title (ax3, 'Test mit der Platine','fontweight','normal')
76  ylabel(ax3, 'Kraft in N')
    xlabel(ax3, 'Versuch')
78  legend('MTS \alpha 135° VS in Y',...
           'PSS \alpha 135° VS in Y',...
80         'MTS \alpha 90° VS in Z',...
           'PSS \alpha 90° VS in Z',...
82         'Location','south',...
           'Orientation','vertical');
84  axis(ax1,[1 10 6 7.5])
    axis(ax2,[1 10 0.1 0.35])
86  axis(ax3,[1 10 1.5 6.5])

88  hold on
    grid on
```

Quellcode C.18: Farbliche Darstellung der Spannung des Sensor-Arrays.

```
%-------------------------------------------------------------------------
% Darstellung der Spannung vom Array
%
% Filename:      rmp_3_array_color_plot.m
%
% Beschreibung:  farbliche Darstellung der Messergebnisse vom AMR-Array.
%
%
%-------------------------------------------------------------------------
clc
clear('ideal','a','b','c','d','bb','dd','cc', 's');
colormap jet
COS_SIG = zeros(8,8);
SIN_SIG = zeros(8,8);
schritt = 0;
folder_name =' DATEINAME EINGEBEN'; % Es muss Dateiname eingegeben werden
s = what(folder_name);
s = s.mat;
for a=1:numel(s)
    if ~strcmp(char(s(a)),'init.mat')
        ideal(a) = load([folder_name,'/',char(s(a))]);
    end
end
a = a-1;
r=182;
for n = 1:1:a
    %Berechnung der Fehler beim Vor- und Rückwärtslauf
    COS_SIG = ideal(n).COS_SIG_f - ideal(r).COS_SIG_f;
    SIN_SIG = ideal(n).COS_SIG_f - ideal(r).COS_SIG_f;

% Plot data from measurement
    subplot(2,4,1)
    imagesc(ideal(n).COS_SIG_f)
    set(gca,'DataAspectRatio',[1 1 1],'YDir','normal')
    fh = figure(1);
    % Definierung der Größe des Bildes
    set(fh,'units','centimeters','position',[0 0 15 13])
    s1 = sprintf(['cosinus signal: \\alpha =' ,...
        num2str(ideal(n).alpha),'^°']) ;
    disp(s1);
    title(s1,'fontweight','normal')
    xlabel(['X Koordinate']);
    ylabel(['Y Koordinate']);
    caxis([0.23 3.3])
    h=colorbar;
    ylabel(h,'Signal in V');

    subplot(2,4,2)
    imagesc(ideal(r).COS_SIG_f)
    s2 = sprintf(['cosinus signal: \\alpha =' ,...
        num2str(ideal(r).alpha),'^Â°']) ;
                disp(s2);
                title(s2,'fontweight','normal')
    xlabel(['X Koordinate']);
    ylabel(['Y Koordinate']);
    caxis([0.23 3.3])
    set(gca,'DataAspectRatio',[1 1 1],'YDir','normal')
    h=colorbar;
    ylabel(h,'Signal in V');

    subplot(2,4,3)
    imagesc(COS_SIG)
    xlabel(['X Koordinate']);
```

```
        ylabel(['Y Koordinate']);
65      caxis([-0.5  0.5])
        set(gca,'DataAspectRatio',[1  1  1],'YDir','normal')
67      title('absolute Fehler in V','fontweight','normal')
        h=colorbar;
69      ylabel(h,'Fehler in V');

71       subplot(2,4,4)
        imagesc(COS_SIG*100/3.3)
73      xlabel(['X Koordinate']);
        ylabel(['Y Koordinate']);
75      caxis([-2  2])
        title('absolute Fehler in %','fontweight','normal')
77      set(gca,'DataAspectRatio',[1  1  1],'YDir','normal')
        h=colorbar;
79      ylabel(h,'Fehler in %');
        pause(0.35)
81      n = n+1;
        r = r-1;
83  end
```

Quellcode C.19: Darstellung des Offsets bzw. Winkelfehlers.

```
%-------------------------------------------------------------------------
% Darstellung des Offsets/Winkelfehlers
%
% Filename:      rmp_3_offset_plot.m
%
% Beschreibung:  Es wird die Offset und Winkelfehler bei den
%                Ausgangssignalen berechnet und dargestellt.
%
%
%-------------------------------------------------------------------------

clear('ideal','a','b','c','d','bb','dd','cc');

COS_SIG = 0;
SIN_SIG = 0;
schritt = 0;
folder_name ='DATEINAME EINGEBEN'; % Es muss Dateiname eingegeben werden
s = what(folder_name);
s = s.mat;
for a=1:numel(s)
    if ~strcmp(char(s(a)),'init.mat')
        ideal(a) = load([folder_name,'/',char(s(a))]);
    end
end

for i=1:1:a-3
    winkel(i)=ideal(i).alpha;
    a(i)=ideal(i).COS_SIG_f;
    b(i)= ideal(i).SIN_SIG_f;
    v_off(i) = sqrt(a(i)+b(i))/2;
end

%Offsetberechnung durch Mittelwertbildung
cos_off = mean(a);
sin_off = mean(b);
%Offsetberechnung durch min und max berechnung
%cos_off = (min(a)+max(a))/2;
%sin_off = (min(b)+max(b))/2;
x=[0:1:numel(s)-4]; %Initialisierung der X-Achse
fh = figure(1);
plot(x,a,'linewidth',1.5)
line([x(1) x(end)], [cos_off cos_off],'Color', 'blue','LineStyle', '--')
hold on
plot(x,b,'linewidth',1.5)
line([x(1) x(end)], [sin_off sin_off],'Color', 'red','LineStyle', '--')
% plot(a2,'linewidth',1.5)
xlim([x(1) x(end)])
grid ( 'on');
title('Sinus Signal','fontweight','normal')
xlabel('Anzahl der Messungen');
ylabel('Spannung in V')
hold off
%% Berechnung des Offset

% durch mittelwertbildung
% a2 = a-cos_off;
% b2 = b-sin_off;

% durch min und max berechnung
a2 = a-(min(a)+max(a))/2;
b2 = b-(min(b)+max(b))/2;

%%
```

```
    a2 = a2./max(abs(a2));
65  b2 = b2./max(abs(b2));
    %Berechnung des Winkels alpha
67  phi = atan2(a2,b2);
    phi = unwrap(phi);
69  %Darstellung des berechneten Winkels
    plot(phi.*180./pi),hold off
71  %Darstellung des Winkelfehlers
    plot(x,winkel-((phi).*180./pi+90))
```

Quellcode C.20: Darstellung der Messergebnisse von Einzelsensor.

```
%--------------------------------------------------------------------------
% Darstellung des Messergebnisse von Einzelnsensoren
%
% Filename:      rmp_3_plot_einzeln.m
%
% Beschreibung:  Es wird die Offset und Winkelfehler bei den
%                Ausgangssignalen berechnet und dargestellt.
%
%
%--------------------------------------------------------------------------
%% ========================================AMR/TMR_Enzeln=====================
clear all;
clear('ideal','a','b','c','d','bb','dd','cc');
colormap jet
COS_SIG = 0;
SIN_SIG = 0;
schritt = 0;
folder_name ='DATEINAME EINGEBEN'; % Es muss Dateiname eingegeben werden
s = what(folder_name);
s = s.mat;
for a=1:numel(s)
    if ~strcmp(char(s(a)),'init.mat')
        ideal(a) = load([folder_name,'/',char(s(a))]);
    end
end
ax1 = subplot(2,1,1); % top subplot
ax2 = subplot(2,1,2); % bottom subplot

hold on;
grid on;
xlabel('Alpha in Degree');
ylabel('Spannung in V');

for i=1:1:a
    a(i)=i;
    b(i)= ideal(i).COS_SIG_f;
    bb(i)= ideal(i).SIN_SIG_f;
end

hold(ax1,'on');
grid (ax1, 'on');
plot(ax1,a,b,'g')
title(ax1,'Cosinus Signal')
xlabel(ax1,'Alpha in Degree');
ylabel(ax1,'Spannung in V')

hold(ax2,'on');
grid (ax2, 'on');
plot(ax2,a,bb,'g')
title(ax2,'Sinus Signal')
xlabel(ax2,'Alpha in Degree');
ylabel(ax2,'Spannung in V')

%% ========================================quaver=============================
clear('ideal','a','b','c','d','bb','dd','cc');
colormap jet
COS_SIG = 0;
SIN_SIG = 0;
schritt = 0;
folder_name ='DATEINAME EINGEBEN'; % Es muss Dateiname eingegeben werden
s = what(folder_name);
s = s.mat;
for a=1:numel(s)
```

```
64      if ~strcmp(char(s(a)),'init.mat')
            ideal(a) = load([folder_name,'/',char(s(a))]);
66      end
    end
68  x_off = zeros(8,8);
    y_off = zeros(8,8);
70  for i=1:1:a
        x_off = x_off+ideal(i).COS_SIG_f;
72      y_off = y_off+ideal(i).SIN_SIG_f;
    end
74  x_off = x_off./a;
    y_off = y_off./a;
76
    r = 73;
78  xx = meshgrid([1:8]);
    yy = xx';
80  fh =figure(1);
    set(fh,'units','centimeters','position',[0 0 13 13])
82  LW =  1;

84  for i= 12
        s1 = sprintf(['X_{pos}:␣',num2str(ideal(i).x),...
86                    ',␣Y_{pos}:␣',num2str(ideal(i).y),...
                       '\n\\alpha␣=', num2str(ideal(i).alpha),'^Â°']) ;
88      quiver(xx,yy,ideal(i).COS_SIG_f-x_off,ideal(i).SIN_SIG_f-y_off,...
            'linewidth',LW);
90      disp(s1);
        title(s1,'fontweight','normal')
92      grid on;axis square;xlabel('X␣Koordinate');ylabel('Y␣Koordinate');
        xlim([0 9]);ylim([0 9]);
94      drawnow
    end
96  %=========================================================================
    colormap jet
98  clear('ideal','a','b','c','d','bb','dd','cc');
    COS_SIG = 0;
100 SIN_SIG = 0;
    schritt = 0;
102 folder_name ='DATEINAME␣EINGEBEN'; % Es muss Dateiname eingegeben werden
    s = what(folder_name);
104 s = s.mat;
    for a=1:numel(s)
106     if ~strcmp(char(s(a)),'init.mat')
            ideal(a) = load([folder_name,'/',char(s(a))]);
108     end
    end
110 r = 73;
    xx = meshgrid([1:8]);
112 yy = xx';
    fh =figure(1);
114 set(fh,'units','centimeters','position',[0 0 13 13])
    LW =  1;
116 for i= 23
        quiver(xx,yy,ideal(i).COS_SIG_f-x_off,ideal(i).SIN_SIG_f-y_off,...
118         'linewidth',LW);
        s1 = sprintf(['X_{pos}:␣',num2str(ideal(i).x),...
120                   ',␣Y_{pos}:␣',num2str(ideal(i).y),...
                       '\n\\alpha␣=', num2str(ideal(i).alpha),'^Â°']) ;
122     disp(s1)
        title(s1,'fontweight','normal')
124     grid on;axis square;xlabel('X␣Koordinate');ylabel('Y␣Koordinate');
        xlim([0 9]);ylim([0 9]);
126     drawnow
    end
128
```

```
%% ==========einzeln_hysterese_saettigung==========
% clear all;
clear('ideal','a','b','c','d','bb','dd','cc','x');
colormap jet
COS_SIG = 0;
SIN_SIG = 0;
schritt = 0;
folder_name ='DATEINAME EINGEBEN'; % Es muss Dateiname eingegeben werden
s = what(folder_name);
s = s.mat;
for a=1:numel(s)
    if ~strcmp(char(s(a)),'init.mat')
        ideal(a) = load([folder_name,'/',char(s(a))]);
    end
end
j=1;
for i=2:1:a
    a(i)=ideal(i).COS_SIG_f;
    b(i-1)= ideal(i).SIN_SIG_f;
end

fh = figure(1);
x = [1.7 1.9 2.1 2.3 2.6 2.9 3.3 3.7 4.1 4.3 4.7 5.4 6.1 7.1 7.9...
    9.1 10.5 12 13.5 15.7 17.8 19.8 23.4 25.3 27.3 31.1 35 38];
xlim([x(1) x(end)]);
plot(x,b,'linewidth',1.5)
grid ( 'on');
title('Sinus Signal','fontweight','normal')
xlabel('Abstand in kA\m');
ylabel('Spannung in V')

hold(ax2,'on');
grid (ax2, 'on');
plot(ax2,x,aa,x,bb)
title(ax2,'Sinus Signal')
xlabel(ax2,'Alpha in Degree');
ylabel(ax2,'Spannung in V')
%================================================
```

D CD

Auf der beigefügten CD befinden sich sämtlich Programme, Messdaten und Messprotokolle, die für diese Arbeit erstellt bzw. verwendet worden. Abbildung D.1 zeigt die Ordnerstruktur, wie sie auf der CD vorhanden ist.

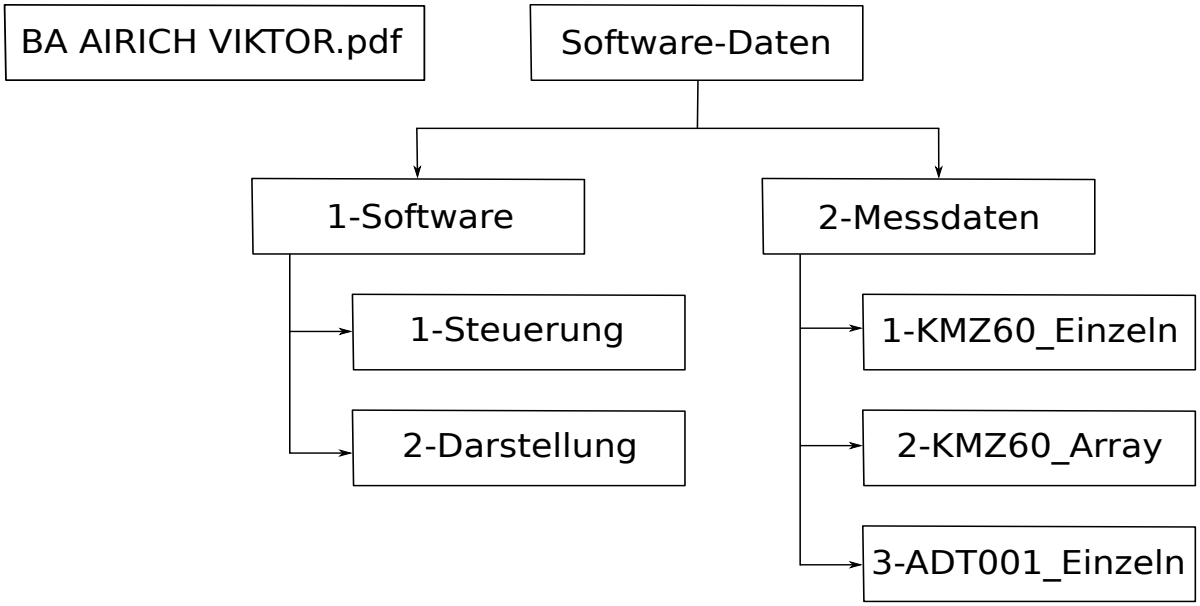


Abbildung D.1: Ordnerstruktur der beigefügten CD.

Selbstständigkeitserklärung

Hiermit versichere ich, Viktor Airich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) APSO-TI-BM ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Hamburg, 10. Januar 2018