HAW HAMBURG

Bachelorthesis

Abraham Begic

Tunnel-Magnetoresistives Sensor-Array - Controllersteuerung, Platinen-Layout und Prüfstands-Erprobung

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

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

Abraham Begic

Tunnel-Magnetoresistives Sensor-Array - Controllersteuerung, Platinen-Layout und Prüfstands-Erprobung

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.-Ing. Luzt Leutelt

Abgegeben am 09. Juli 2018

Abraham Begic

Thema der Bachelorarbeit

Tunnel-Magnetoresistives Sensor-Array - Controllersteuerung, Platinen-Layout und Prüfstands-Erprobung

Stichworte

Magnetische Sensoren, Automobil-Sensoren, Anisotroper magnetorisistiver Sensor, Tunnel magnetorisistiver Sensor, AMR- und TMR-Sensor-Array, Halbbrücken Steuerung, Sinus, Cosinus, Prüfstand-Erprobung, Robotermessplatz

Kurzzusammenfassung

Für zukünftige integrierte Sensoren basierend auf den Tunnel-Magnetoresistiven-Effekt sollen Vorversuche durchgeführt werden. Dazu wird ein vergrößertes Array mit diskreten Sensoren als Platine aufgebaut. Dieses Sensor-Array wird mit einem Mikrocontroller gesteuert. Die anschließende Funktionsüberprüfung erfolgt mit einem Robotermessplatz. Dabei werden Messungen mit und ohne Störfeld aufgenommen, verglichen wird dieses Array mit einem Sensor-Array basierend auf den Anisotropen-Magnetoresistiven-Effekt.

Abraham Begic

Title of the bachlor thesis

Tunnel magnetoresistive Sensor Array - Controler system, board layout and test bench setup

Keywords

Magnetoresistive Sensor, automotive sensors, Anisotropic magnetoresistive Sensor, Tunnel magnetoresistive Sensor, AMR- and TMR-Sensor-Array, half bridge control, sine, cosine, test bench setup, robot measuring setup

Abstract

Preliminary tests will be carried out for future integrated sensors based on the tunnel magnetoresisitve effect. For this purpose, an enlarged array with discrete sensors is constructed as a circuit board. This sensor array is controlled by a microcontroller. The subsequent function test is carried out with a robot measuring station. Measurments with and without interference fields are recorded and compared with a sensor array based on the anisotropic magnetoresistive effect.

Vorwort

An dieser Stelle möchte ich mich bei Herrn Prof. Dr-Ing. Karl-Ragmar Riemschneider für die Möglichkeit und Betreuung bedanken, meine Bachelorarbeit im Rahmen des ISAR-Projektes erstellen zu können. Für die Übernahme des Zweitgutachters möchte ich mich bei Herrn Prof. Dr.-Ing. Lutz Leutelt bedanken.

Bei Herrn M.Sc Thorben Schüthe bedanke ich mich für die fachliche Beratung und Unterstützung während des ganzen Bearbeitungszeitraumes. In diesem Zuge bedanke ich mich bei der Arbeitsgruppe ISAR und BATSEN für das tolle Forschungsklima.

Des Weiteren gilt ein großer Dank an Herrn Dipl.-Ing. Günter Müller für das Korrekturlesen der vorliegenden Arbeit sowie seinem fachlichen Rat.

Ein besonderer Dank gilt, Frau Leonie Herzog, die mich bei der Bestückung des Sensor-Arrays unterstützte.

Meinen Freunden und Kommilitonen, die mich in der Zeit des Studiums und darüber hinaus begleitet haben, spreche ich an dieser Stelle meinen Dank aus.

Vor allem spreche ich meinen großen Dank an meine Eltern aus, die immer an mich geglaubt und mich im gesamten Zeitraum unterstützt haben. Meinen Geschwistern und der Familie, die mir bei mannigfaltigen Gelegenheiten zur Seite standen.

Nicht zuletzt bedanke ich mich bei meiner Freundin, für ihr Verständnis und ihrer Hilfe während der Bearbeitungszeit der vorliegenden Arbeit.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

AMR	Anisotroper-Magnetoresistiver-Effekt
CMR	kolossaler Magnetoresistiver Effekt
GMR	Giant Magnetoresistiver Effekt
GPIO	General Purpose Input/Output
MR	magnetoresistiv
MR-Effekt	magnetoresistiver Effekt
TMR	Tunnel-Magnetoresistiver Effekt

Symbolverzeichnis

Symbol	Einheit	Beschreibung
$COS_$LX	V	Signalleitung der Cosinus-Zeile
$COS_$X	V	Signalleitung COS zwischen Ausgangswiderstand und Pin
$GC_$X	V	Masseleitung der Cosinus-Spalte
$GS_$X	V	Masseleitung der Sinus-Spalte
R_{ADC}	Ω	Widerstand des ADCs
R_{CXX}	Ω	Widerstand der Cosinus-Spalte
R_{GC}	Ω	TMR-Widerstand zwischen dem Kontat GND und COS
R_{GS}	Ω	TMR-Widerstand zwischen dem Kontakt GND und SIN
R_{SXX}	Ω	Widerstand der Sinus-Spalte
R_{VC}	Ω	TMR-Widerstand zwischen dem Kontakt VCC und COS
R_{VS}	Ω	TMR-Widerstand zwischen dem Kontakt VCC und SIN
$SIN_$LX	V	Signalleitung der Sinus-Zeile
$VC_$X	V	Versorgungsspannung der Cosinus-Spalte
$VS_$X	V	Versorgungsspannung der Sinus-Spalte

1 Einleitung

In der Automobilelektronik haben magnetische Sensoren einen hohen Stellenwert. Aufgrund des berührungslosen Erfassens von Winkelinformationen und Drehzahlen. Das Einsatzgebiet umfasst dabei den Antriebsstrang, das Sicherheitssystem sowie den Komfortbereich. Die Hochschule für Angewandte Wissenschaften (HAW) Hamburg und Ihre Partner arbeiten kooperativ, am dem Forschungsprojekt „ISAR“[1]. Dieses Projekt wird vom Bundesministerium für Bildung und Forschung gefördert. Die Ziele dieses Projekts sind zum einen die Zulassung der verschiedenen Einsatzanforderungen und zum anderen die Erkennung und Unterdrückung der Störfelder. Des weiteren wird ein Konzept zur Ein- und Ausgabe von Daten, als auch die Trennung von relevanten und irrelevanten Informationen vorgenommen. Der Beitrag dieser Arbeit ist es ein Sensor-Array zu erstellen basierend auf den Tunnel-Magnetoresistiver Effekt (TMR)-Effekt. Dies ist ein vergrößertes Funktionsmodell des zukünftig zu implementierenden Sensors. Mit dem Modell soll es möglich sein das Magnetfeld eines Permanentmagneten zu erfassen, dessen räumliche Lage die Nutzinformation beinhaltet. Das Sensor-Array ermöglicht die Fehlerkorrektur für diese Nutzinformationen. Ein weiterer Aspekt ist die Kompensation und Detektion von Störfeldern.

1.1 Definition Sensor

Der Begriff Sensor ist auf die lateinischen Wörter *sensus* -„das Beobachten“- bzw. -„das Wahrnehmen“- und *sentire* -„wahrnehmen oder fühlen“- zurückzuführen. Sensoren sind nicht mehr aus der heutigen Zeit, dem Zeitalter der Digitalisierung und Industrie 4.0, wegzudenken. Es wird immer wichtiger schnell, präzise und mit einer hohen Genauigkeit Daten zu erfassen, bearbeiten oder zu interpretieren. Der große technische Fortschritt in der technologischen Welt ist auch den präziseren und zugleich smarten Sensoren zu verdanken. Sie befähigen Maschinen dazu, zu „fühlen“. Die magnetischen Sensoren, die zurzeit in Automobilen verbaut werden beruhen auf den Hall-Effekt oder auf den Anisotroper-Magnetoresistiver-Effekt (AMR)-Effekt. Die letzteren Sensoren sind anfällig gegenüber Störfeldern und Fehllagen des Gebermagneten. Durch ein Sensor-Array soll die Anfälligkeit von Störfeldern und einer Fehllage des Gebermagnetens minimiert werden.

[1] Signalverarbeitung für **I**ntegrated **S**ensor-**AR**rays basierend auf den tunnelmagnetoresistivern-Effekt für den Einsatz in der Automobilelektronik.

Allgemein erfassen Sensoren einen Großteil der physikalischen Größen, welche auf den SI-Einheiten basieren, wie z. B.:

- Zeit
- Länge
- Masse
- Elektrische Stromstärke
- Temperatur
- Stoffmenge
- Lichtstärke

1.2 Sensoren im Automobil

Heutzutage sind Sensoren im Automobil kaum mehr wegzudenken. Aufgrund der technischen Innovationen in den letzten Jahrzehnten haben sich die Sensoren erfolgreich ihren Weg in die Welt der Kraftfahrzeuge gebahnt. Sensoren und die für sie notwendigen elektronischen Bauteile sind Fluch und Segen zugleich. Im modernen Fahrzeug ist keine freie Stelle vorhanden, weil jeder freie Platz so gut wie möglich genutzt und mit Elektronik bestückt wird. Durch die verbaute Technik ist das Automobil gefühlt anfälliger für Fehler und Temperaturextreme im Vergleich von vor 30 Jahren, wo jede technisch begabte Person ihr Fahrzeug selbst reparieren konnte. Die Vorteile der Sensoren überwiegen, alleine im Bereich der Sicherheit und Leistungsoptimierung und damit einhergehend, umweltfreundlicher Nutzung. Die heutzutage innerhalb der Automobiltechnik eingesetzten Sensoren können in folgende drei Kategorien unterteilt werden:

1. Kategorie Antriebsstrang: Dort kommen die Sensoren zum Einsatz, die wichtige Daten verarbeiten und weiterleiten bzw. abfragen, um den technischen Betrieb des Fahrzeugs zu gewährleisten.

2. Kategorie Sicherheit: Sensoren dieser Kategorie sind zum Schutz der Personen im Fahrzeug vorgesehen und leisten einen großen Beitrag, die straßenverkehrsbedingten Unfälle zu reduzieren.

3. Kategorie Komfort: Durch den Einsatz dieser Sensoren ist eine Fahrt auf langen Distanzen erheblich angenehmer geworden. Allein die Navigation hat in den vergangenen zehn Jahren einen großen Sprung, erfahren.

In Tabelle 1.1 sind einige der eingebauten Sensoren in dieser Unterteilung folgend aufgelistet.

Tabelle 1.1: Klassifizierung derzeitiger Sensoren in der Automobilelektronik [10].

Antriebsstrang		
Drucksensor	Ladedrucksensor	Luftmassensensor
Klopfsensor	Umgebungsdrucksensor	Hochdrucksensor
Lambda-Sonde	Drehzahlsensor	Tankdrucksensor
Pedalweggeber	Winkelgeber	Positionsgeber
Ermüdungssensor		
Sicherheit		
Abstandsradar	Neigungssensor	Hochdrucksensor
Drehmomentsensor	Lenkradwinkelsensor	Beschleunigungssensor
Drehratensensor	Sitzbelegungssensor	Drehzahlsensor
Komfort		
Drehratensensor	Luftgütesensor	Feuchtigkeitssensor
Temperatursensor	Drucksensor	Regensensor
Abstandssensor Ultraschall	Lichtsensor	

Das TMR-Array kann in allen drei Bereichen eingesetzt werden. Im Antriebsstrang als Winkelgeber oder Drehzahlsensor. Im Bereich der Sicherheit als Translationssensor im Antiblockiersystem (ABS) und im Komfortbereich als Sensor in der Servolenkung. In der folgenden Tabelle 1.2, wird der Unterschied zwischen dem AMR-Effekt und dem TMR-Effekt aufgezeigt. Der AMR-Effekt hat ein sehr gutes Signal-Rausch-Verhältnis (SNR)[2], sowie Bandbreite und Verhalten bei Hysterese. Der TMR-Effekt besitzt eine sehr gute Widerstandsänderung im ΔR / R Verhältnis zum konstanten Widerstand sowie bei der Leistungsaufnahme, Temperaturstabilität und Miniaturisierbarkeit.

[2]Engl. für signal-to-noise ratio.

Tabelle 1.2: Unterschied in den Attributen des AMR-Effekts und TMR-Effekts (0: neutral; +: gut; ++: sehr gut) [12].

	AMR-Effekt	TMR-Effekt
$\frac{\Delta R}{R}$	0	++
Empfindlichkeit	+	+
Signal/Rauschen	++	0
Bandbreite	++	0
Leistungsaufnahme	0	++
Temperaturstabilität	+	++
Hysterese	++	0
Miniaturisierbarkeit	0	++

1.3 Ziel dieser Arbeit

Basierend auf dem TMR-Effekt soll ein Funktionsmodell in Form eines Sensor-Arrays entworfen und erstellt werden. Es kann auf Vorarbeiten zu einem Funktionsmodell in AMR-Technologie aufgebaut werden. Das Sensor-Array soll kommerzielle TMR-Sensoren so auslesen, dass die Halbbrückenspannung (COS, SIN) getrennt und ohne Kreuzbeeinflussung erfasst werden können. Das Array soll einerseits ein Beispiel für die Beschaltung und das Auslesen der Sensordaten bereitstellten und anderseits Daten für den Entwurf und Test von Auswertealgorithmen liefern. Für die Erprobung wird ein Robotermessplatz verwendet, der in einer vorherigen Abschlussarbeit genutzt wurde [2]. Ferner soll ein Vergleich mit dem AMR-Array durchgeführt werden. Die Zielsetzung wurde in die folgenden Strukturen aufgegliedert, um den Arbeitsablauf in einen festen Rahmen zu definieren.

1. Einarbeitung Grundlagen:
   - Magnetische Sensoren im Automobil
   - Funktionsweise magnetischer Sensoren
   - Einarbeiten in das Platinendesign
2. Sensor-Array:
   - Entwurf eines 8x8 Sensor-Arrays
   - Optimierung der Anschlussführungen zwischen Sensor- und Mikrocontroller
   - Funktionsüberprüfung des Arrays
   - Fertigung und Bestückung
   - Inbetriebnahme und Funktionscheck
3. Software:

- Anpassung der vorhandenen Software
- Neukonzeption des Ausleseverfahrens
- Datenvisualisierung zur Prüfung der Funktionalität

4. Versuchsreihen und Auswertungen mittels automatisiertem Messsystem:
   - Modifikation der Steuer-Erfassungs-Skripte des Robotermessplatzes
   - Aufstellen eines Mess- und Versuchsplans
   - Äquivalente Versuchsreihen mit TMR-Array zu den Vorarbeiten mit dem AMR-Array
   - Störfeldaufschlag mittels Halbach-Ring-Array

2 Grundlagen

Dieses Kapitel befasst sich mit den Grundlagen der Magnetischen Sensoren, dabei stehen der AMR- und TMR-Sensor im Fokus.

2.1 Magnetische Sensoren

Vor ca. 150 Jahren wurde der magnetoresistiver Effekt (MR-Effekt) entdeckt und dient heute als Grundlage für magnetische Sensoren. Einer der ersten Entdecker war der Physiker William Thomson im Jahre 1857. Er bemerkte, dass sich der elektrische Widerstand eines vom Strom durchflossenen Leiters unter dem Einfluss eines Magnetfeldes ändert. Erst gegen Ende des letzten Jahrhunderts, vor ca. 30 Jahren, war es möglich, den Effekt zur sensorischen Nutzung zu industrialisieren, was der Weiterentwicklung der Halbleiterindustrie bzw. Dünnschichttechnik zu verdanken ist [12].

Der MR-Effekt findet seinen größten Anwendungsbereich derzeit in Festplattenleseköpfen. Der am schnellsten wachsende Markt ist heutzutage die Industrieanwendung. Was auf das verschleißfreie Messen bzw. das weite Anwendungsgebiet der Sensoren, wie z. B. in der Medizintechnik, dem Maschinenbau und dem Fahrzeugbau zurückzuführen ist. Die Unterschiede zwischen den Effekten sind gegeben durch die Anordnung und die Art der Materialien. Folgende Unterscheidungen existieren: AMR-Effekt, TMR-Effekt, Giant Magnetoresistiver Effekt (GMR)-Effekt, kolossaler Magnetoresistiver Effekt (CMR)-Effekt und Hall-Effekt. Sensoren auf dieser Grundlage werden auch Magnetowiderstandssensoren genannt, da sich der Widerstandswert der Sensoren beim Anlegen eines äußeren Magnetfeldes ändert. In der Abbildung 2.1 sind die Schichten des AMR, GMR und TMR Aufbaus dargestellt. In dieser Arbeit wird Bezug auf den AMR und TMR genommen ([14, S. 281–315] und [12]).

2.1.1 Anisotroper magnetoresistiver Sensor

Der AMR-Effekt ist der am längsten bekannte Effekt und basiert auf der Forschung von Thomson Mitte des 19. Jahrhunderts. Er beschreibt die Abhängigkeit des spezifischen Widerstandes durch Einflussnahme eines von außen anliegenden Magnetfeldes, wobei der Winkel zwischen der Magnetisierung und Stromrichtung die ausschlaggebende Größe ist. Der Effekt lässt sich am besten in einer Permalloy-Schicht (Nickel-Eisen-Legierung) beobachten. Die Maxima des Widerstandes sind immer dann gegeben, wenn entweder das von außen anliegende Magnetfeld und der Strom gleich groß sind (Maximum) oder

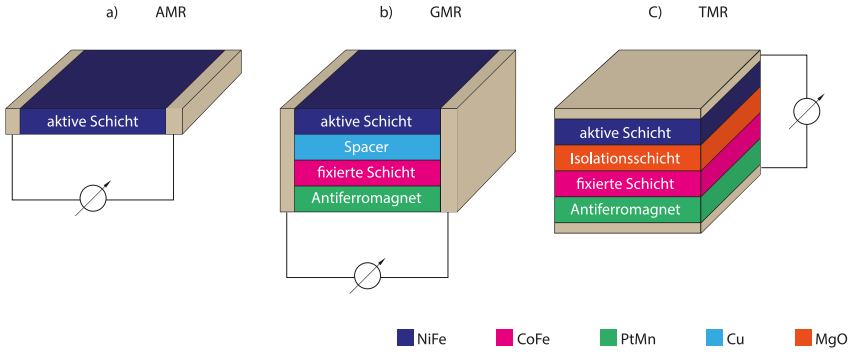


Abbildung 2.1: Gegenüberstellung des Aufbaus von AMR-, GMR- und TMR-Sensoren [12].

das anliegende Feld senkrecht zum Strom steht (Minimum) ([14] und [8]). Die folgende Gleichung 2.1 beschreibt dies:

$$R(\phi) = R_m + \Delta R \cdot cos(2\phi) \tag{2.1}$$

Der Winkel ϕ, ist der Winkel zwischen der Stromrichtung und Magnetisierung. ΔR ist die Widerstandsänderung in Abhängigkeit des Winkels ϕ. R_m ist der konstante Widerstand und $R(\phi)$ ist der Widerstand in Abhängigkeit des Winkels ϕ.

Zu erkennen ist, dass der elektrische Widerstand vom doppelten Winkel abhängt. Dies hat zur Folge, dass lediglich ein Winkelbereich von 180° eindeutig zu bestimmen ist. In den Sensoren befinden sich Wheatstone-Brücken, die 45° verdreht zueinander positioniert sind. Dadurch ergibt sich im Verlauf des Widerstandes ein Phasenversatz von 90°. Durch die Brückenschaltungen und deren Anordnung zueinander ist der Aufbau unempfindlicher gegenüber Temperaturschwankungen [8].

2.1.2 Tunnel magnetoresistiver Sensor

In den 1970-er Jahren entdeckte der Physiker M.Jullière den TMR-Effekt [6]. Dieser beschreibt, dass durch eine Isolationsschicht im Nanobereich, zwei voneinander getrennte ferromagnetische Nanoschichten durch einen quantenmechanischen Effekt, Elektronen tunneln lassen. Jullière führte ein Modell zur Beschreibung des spinabhängigen Elektronentransports in Tunnelmagnetwiderständen ein. Die Voraussetzung dafür ist, dass der Elektronenspin beim Durchtunneln weiter anhält und die zwei unterscheidbaren Spin[1], Majoritäts-Spin (spin-up) und Minoritäts-Spin (spin-down), in den jeweiligen freien Bereich, der zugehörige Gegenelektrode tunneln. Damit können beide Spinkanäle unabhängig voneinander betrachtet werden. Nach dieser Betrachtungsweise lässt sich der

[1]Spin kommt aus dem engl. und bedeutet Drehung oder Drall, hier handelt es sich um den Eigendrehimpuls von Teilchen (Elektronen) [15].

TMR-Effekt anhand der Formel 2.2 berechnen:

$$TMR = \frac{G_p - G_{ap}}{G_{ap}} = \frac{R_{ap} - R_p}{R_p} = \frac{2P_lP_r}{1 - P_lP_r} \tag{2.2}$$

G_p ist der Leitwert, der für die parallele Magnetisierung steht. G_{ap} ist der Leitwert, der für die antiparallele Magnetisierung steht; äquivalent dazu die Magnetowiderstände R_p und R_{ap}. Die Spinpolarisation P_l (Position links) bzw. P_r (Position rechts) bezeichnet die Zustandsdichte an der Fermi-Kante[2]. Der Flächenwiderstand der Barriere und die Fläche der Tunnelverbindungen definieren den Widerstand des TMR-Sensors. Ein elementarer Unterschied zum AMR-Effekt besteht darin, dass sich der Widerstand und die Größe der Sensorelemente proportional zueinander verhalten. Dadurch reduziert sich der Leistungsbedarf bzw. die Leistungsaufnahme. Das Verhalten des TMR-Sensors im Drehfeld unterscheidet sich gegenüber dem des AMR-Sensors. Durch die 90°-Drehung der Magnetisierung im AMR-Sensor vom einen Maxima zum nächsten ist bereits der magnetoresistiv (MR)-Hub erreicht und erst wieder bei 180° gegeben. Beim TMR-Sensor verhält es sich anders. Der volle Bereich ist erst nach einer ganzen Umdrehung ausgeschöpft. Damit hat der TMR-Sensor eine Periode von 360° ([14, S. 281–315] und [9]).

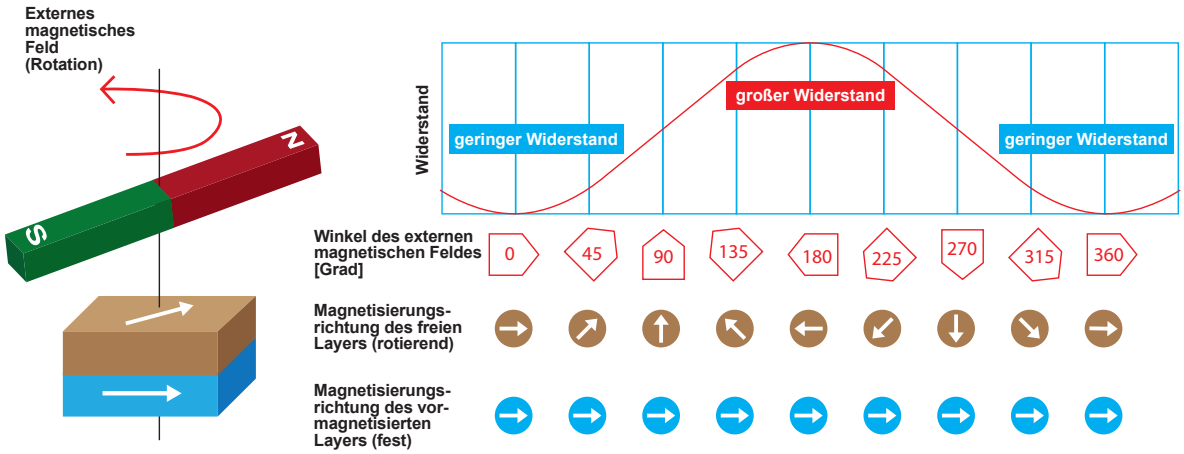


Abbildung 2.2: Funktionsweise von TMR-Sensoren in Abhängigkeit eines externen Magnetfeldes, in braun dargestellt der freie Layer, in blau dargestellt der vormagnetisierter Layer. Der Kurvenverlauf gibt den Widerstandswert in Abhängigkeit zum Winkel an [13].

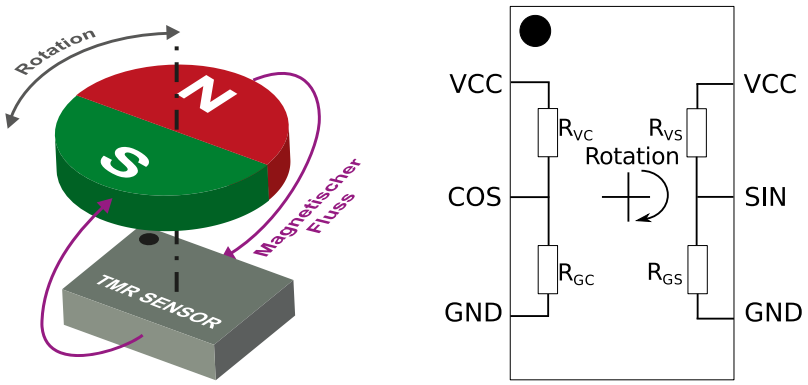
[2]Die Fermi-Kante beschreibt einen steilen Sprung der Fermi-Verteilung bei der Temperatur T = 0K [16].

Abbildung 2.2 stellt die Funktionsweise der TMR-Sensoren in Abhängigkeit von einem äußeren Magnetfelde dar. Die blaue dargestellte Schicht ist der vormagnetisierte Layer, diese dient als Referenzpunkt. Die braune dargestellte Schicht ist der freie Layer, dieser nimmt die Magnetisierung des Gebermagnetens an. Aus dem Winkel zwischen den beiden Schichten ergibt sich der Widerstandswert des Sensors. Dargestellt im Kurvenverlauf. Sind beide Schichten entgegengesetzt magnetisiert, ist der Widerstand groß. Sind sie gleich ausgerichtet, ist der Widerstand dementsprechend klein.

2.2 Verwendeter Sensor zum Aufbau des Arrays

Aus Gründen der Logistik sowie der Verfügbarkeit der Sensoren ist die Entscheidung, bezüglich des zu nutzenden Sensors, auf den TMR-Sensor AAT 001-10E von NVE gefallen. Nach Angaben des Datenblattes sind die TMR-Elemente im Gehäuse um 90° zueinander gedreht und je zwei dieser Elemente sind zu einer Halbbrücke verschaltet. Das am Sensor angelegte Magnetfeld kann durch die Konfiguration der Ausgänge durch Sinus- und Cosinus-Funktionen dargestellt werden. Ein weiter Vorteil ist die geringe Kantenlänge der Sensoren. Diese liegt lediglich bei 2,5 mm, wodurch sich die resultierende Arraygröße von 30,60 mm ergibt.

Die Abbildung 2.3(a) zeigt, wie ein externer Magnet ein Feld durch die Ebene des Sensors generiert. Die freien Layer der Sensorelemente richten sich nach dem Magneten aus. Durch die Rotation des Magneten ändert sich der Winkel zwischen dem freien Layer und dem darunter liegenden, vormagnetisierten Layer. Dies ändert den Widerstand der TMR-Elemente und damit die Ausgangsspannung des Sensors. Die Vorzugsrichtung in Bezug auf die Rotation des Magneten und die Anschlüsse des Sensors sind in der Abbildung 2.3(b) dargestellt. Der Punkt gibt die Ausrichtung des Sensors an. Die relevanten Daten in der Tabelle 2.1 sind dem Datenblatt entnommen.



(a) Gebermagnet und Sensor.

(b) Brückenschaltung und Ausrichtung des TMR-Sensors.

Abbildung 2.3: Darstellungen über die Funktionsweise des Sensors [1].

Tabelle 2.1: Spezifikationen des Sensors AAT 001-10e von NVE [1].

Parameter	Min.	Typ.	max	Einheiten
Betriebstemperatur	-40		125	°C
Widerstandswert	0,6	1,25	2,5	MΩ
Spitze-Spitze Ausgangsspannung	130	200		$\frac{mV}{V}$
Offsetspannung	-10		10	$\frac{mV}{V}$
Betriebsspannung	0		5,5	V
anzulegendes externes Magnetfeld	2,38		15,91	$\frac{kA}{m}$

3 Hardware-Entwurf des Sensor-Arrays

In diesem Kapitel wird erläutert, wie die Umsetzung von der Theorie in die Praxis erfolgt. Angefangen beim theoretischen Ansatz der Verschaltung der Sensoren, bis zur Ansteuerung und der Kommunikation zwischen Hardware, Mikrocontroller und PC.

3.1 Grundkonzept des Arrays

Der Anspruch an die Hardware ist, dass das Sensor-Array kompakt zentriert und frei von jeglichen Störeinflüssen ist. Alle benötigten aktiven oder passiven Bauteile, die nicht zum Sensor-Array gehören, werden aus diesem Bereich ausgeschlossen. So lassen sich mögliche Störeinflüsse reduzieren und im späteren Fertigungsprozess kann das Sensor-

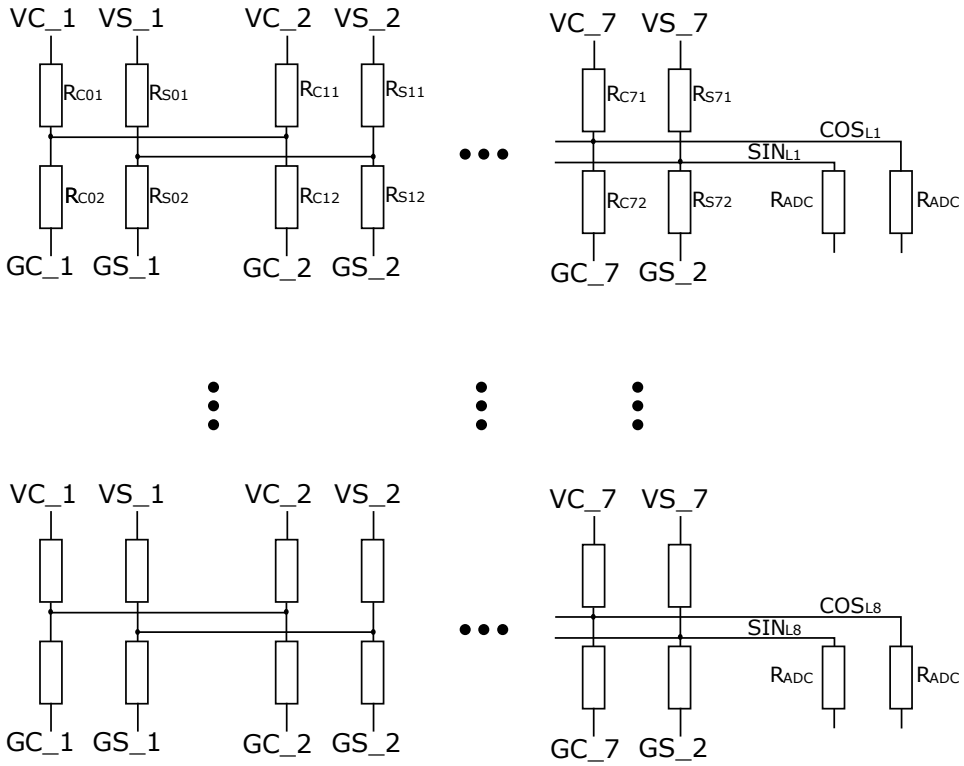


Abbildung 3.1: Beschaltung der Widerstände im 8x8 Array. Die Beschriftung erfolgt spaltenweise und ist durchnummeriert nach einer Array-Indexierung (R_{CXX} Widerstand der COS-Spalte; R_{SXX} Widerstand der SIN-Spalte; R_{ADC} Widerstand am ADC).

Array komplett autark in die Produktion gehen. Um so dicht wie möglich mit einem Gebermagneten an das Sensor-Array heranzufahren, ist es sinnvoll, alle anderen Bauteile auf der Unterseite der Platine zu platzieren.

Das verwendete Mikrocontroller-Board ist das TM4C1294 Connected Launch Pad Evaluation Kit EK-TM4C1294XL der Firma Texas Instruments. Die Verbindung zwischen PC und Mikrocontroller erfolgt mittels einer USB-Schnittstelle. Eine schematische Darstellung kann der Abbildung 3.2 entnommen werden. Dadurch, dass die Sensoren intern zu Halbbrücken verdrahtet sind (wie im Abschnitt 2.2 erläutert), ergibt sich für den Aufbau des Arrays eine bestimmte Struktur. In einer Anordnung in einem 8x8-Array, an dem alle Masse- und Versorgungsleitungen miteinander verbunden sind, ergibt sich ein Widerstandsnetzwerk, das den Betrieb durch die Beschaltung verhindert. Um dies zu umgehen, werden die Sensoren durch die Beschaltung in zwei Seiten aufgeteilt: Links die Cosinus-Seite und rechts die Sinus-Seite. Jede Seite erhält eine eigene Versorgungs- und Masseleitung, welche spaltenweise an acht Sensoren angeschlossen werden. Somit verfügt eine Spalte über vier Versorgungsleitungen, was in der Summe 32 Leitungen ergibt. Die Sinus- bzw. Cosinus-Ausgangssignale sind spaltenweise verbunden. Zwei Leitungen pro Zeile ergeben 16 Datenleitungen. Die beschriebenen Anschlüsse sowie das Konzept zum Betrieb des Sensor-Arrays werden in der Abbildung 3.1 veranschaulicht. Es ist immer nur eine Spalte mit jeweils acht Sensoren durch vier Leitungen mit Spannung versorgt (vertikal). Das Auslesen erfolgt pro Sensor mit jeweils zwei Datenleitungen in einem Zyklus (horizontal).

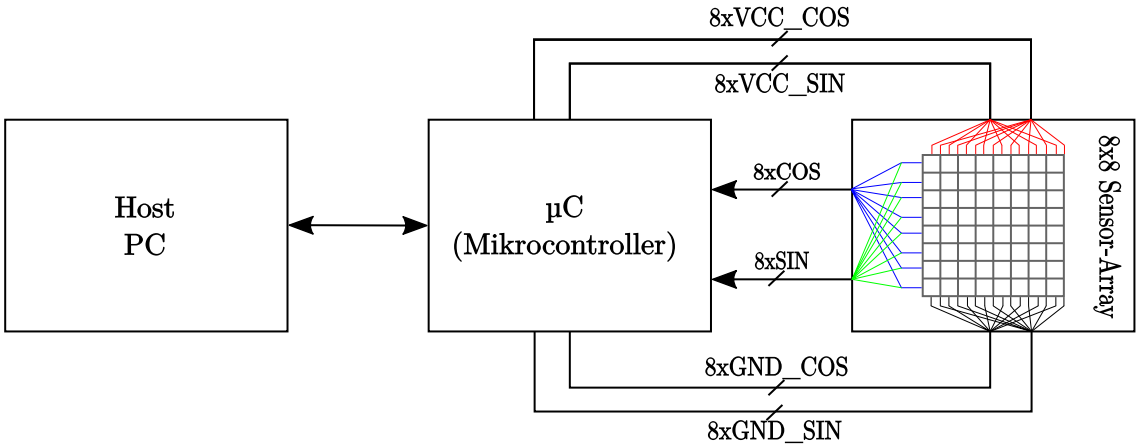


Abbildung 3.2: Schematische Darstellung der Verbindung vom Sensor-Array über den Mikrocontroller zum PC.

3.2 Ausrichtung und Ansteuerung des Sensor-Arrays

Wie im Unterpunkt 2.2 erläutert, haben die Sensoren eine Vorzugsrichtung. Dies bedeutet für das Array, alle Sensoren in vertikaler und horizontaler Richtung gleich auszurichten. Um einen reibungslosen Betrieb zu gewährleisten, sind alle Sensoren auf einer gleichen Höhe anzubringen. Sind sie in der Ebene unterschiedlich ausgerichtet oder unterschiedlich hoch, kommt es zu einer Verschiebung des aufzunehmenden Magnetfeldes. Die Vorzugsrichtung (Koordinatenkreuz für die Drehung) ist anhand eines Achsenkreuzes auf der Oberseite der Platine gekennzeichnet B.3. Dies ergibt sich aufgrund der Ausrichtung der Sensoren, wie in der Abbildung 2.3(b) dargestellt.

Um die Platine anzusteuern und zu versorgen, bedarf es 48 General Purpose Input/Output (GPIO); 16 davon sind für die analogen Signale zuständig. Die 32 restlichen GPIO für die Versorgung und damit ebenfalls für die Ansteuerung des Sensor-Arrays nach dem im Abschnitt 3.1 erläuterten Schema. Die hohe Anzahl der Anschlüsse an der Platine hat zur Folge, dass auf dem Mikrocontroller das Boosterpack X11 genutzt werden muss. Die dafür geeigneten GPIO werden aus dem Datenblatt ermittelt und zugewiesen. Abbildung 3.3 zeigt die Belegung des Boosterpacks X11 [5].

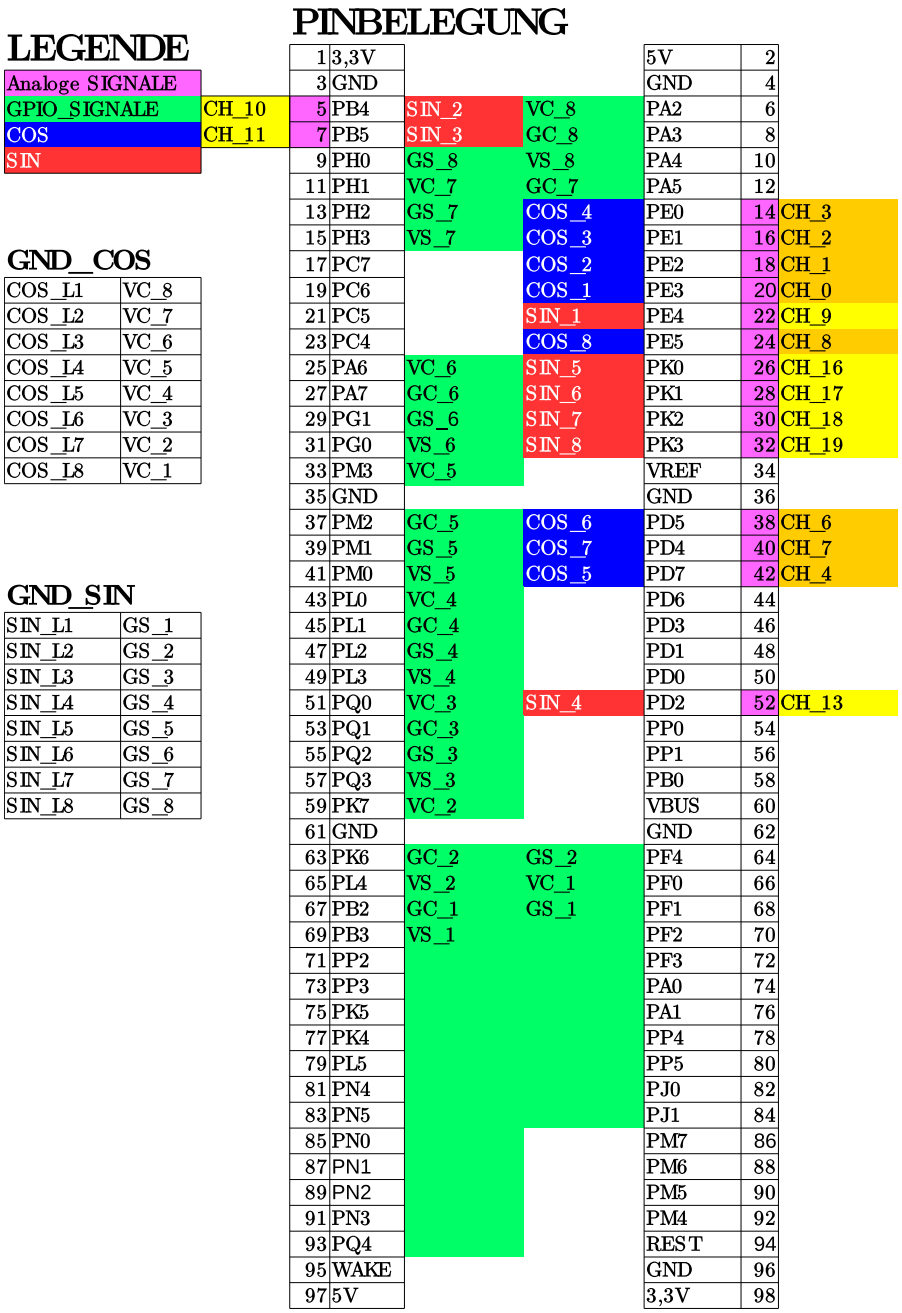


Abbildung 3.3: Darstellung der Pinleiste X11 des Controllers. *VC*_X: Spannungsversorgung der COS-Seite, *VS*_X: Spannungsversorgung der SIN-Seite. *GS*_X: Masseleitung der SIN-Seite, *GC*_X: COS-Seite (grün). *COS*_LX: Ausgangssignal der COS-Seite (blau), *SIN*_LX: SIN-Seite (rot) [3].

3.3 Schaltplan in Eagle

Zu Beginn muss eine Bibliothek für den Sensor und seiner Anschlüsse erstellt und eingebunden werden. Eine Matrixanordnung von 8x8 Elementen wird platziert. Die Versorgungsbusleitungen verlaufen spaltenweise links und rechts von den Sensoren. Das Beschriften der Leitungen erleichtert das Verdrahten und Zuweisen für die Cosinus-Seite VC_X, GC_X und Sinus-Seite VS_X, GS_X. In horizontaler Ausrichtung verlaufen Signalbusleitungen COS_LX und SIN_LX. Das X ist ein Platzhalter für die jeweilige Zeile. Durch die Namenszuweisung der Leitungen lassen sich die Bauteile mit den Buslinien verdrahten. Eine fehlerhafte Verdrahtung wird durch die eindeutige Zuweisung verhindert.

Für die Verbindung zwischen Sensor-Array und Mikrocontroller sind 16 Widerstände und zwei 2x20-Pin-Header platziert. Dadurch, dass die jeweilige Leitung denselben Namen wie die zugehörige Datenleitung hat, entsteht auch hier eine Verbindung zwischen Bauteil und Buslinie. Um eine Brücke bzw. einen Kurzschluss zwischen Pin-Header und Widerstand zu vermeiden, ändert sich der Name der Leitung von COS_LX zu COS_X (exemplarisch; gilt auch für den SIN). Somit ist sichergestellt, dass kein Kontakt zwischen beiden Kontaktpunkten der jeweiligen Widerstände entsteht. Die umbenannten Leitungen werden den Pin-Headern zugewiesen. Die Versorgungslinien erhalten eine direkte Verbindung von den Buslinien zu den Pin-Headern. Für eine eventuelle Spannungsstabilisierung und Filterschaltung sind jeweils 16 Kondensatoren platziert. Die Kondensatoren zur Spannungsstabilisierung sind zwischen der jeweiligen Versorgungsspannung VC_X

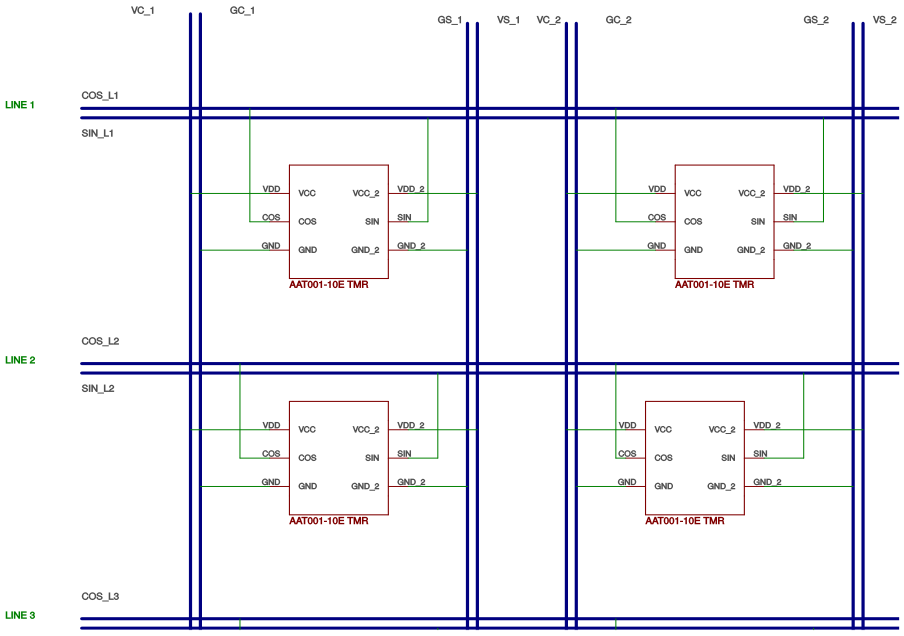


Abbildung 3.4: Auszug aus der schematischen Darstellung der Platine.

und Ground $GC_$X der Sensoren platziert und verdrahtet. Für die Filterschaltung sind die Kondensatoren zwischen Ausgangssignal $SIN_$LX und der jeweiligen Masseleitung $GC_$X platziert und verdrahtet. Zum Messen der Ausgangssignale sind zwei 1x8-Pin-Header platziert, welche direkt mit der Busleitung verbunden sind. Eine schematische Darstellung zeigt die Abbildung 3.4. Die gesamte schematische Darstellung ist abgebildet in B.2 sowie die Pinleisten in Abbildung B.1.

3.4 Erstellung der Platine

In diesem Abschnitt folgt die Beschreibung, wie die Hardware des Sensor-Arrays erstellt wurde. Zum Erstellen der Hardware sind vorab einige Einstellungen nötig. Diese sind zum einen das Einstellen des Rasters in Millimeter, zum anderen die Leitungsbreite sowie der Durchmesser der Kontaktbohrungen zwischen den Ebenen (Via).

3.4.1 Dimensionen der Platine

Die Platine hat eine Größe von 120 mm x 80 mm (vgl. Abbildung B.6). Die Länge wird benötigt, um die Ausgangspinleiste zu platzieren. Zur späteren Orientierung des Robotermessplatzes erhält das Board in der Mitte eine Bohrung von 0,5 mm im Durchmesser. Ausgehend vom Mittelpunkt, im Abstand von 2 mm in Y-Richtung nach oben und von da 2 mm in X-Richtung nach links, befindet sich der Mittelpunkt des ersten Sensors. Die nächsten Sensoren folgen in einem Abstand von 4 mm und es ergibt sich ein Array von 30,60 mm x 30,60 mm, dargestellt in der Abbildung B.6.

3.4.2 Positionierung der Bauteile

Entscheidend für eine kompakte Anordnung des Sensor-Arrays ist die Positionierung der Bauteile. Da das Sensor-Array im Fokus steht, befindet sich dies in der Mitte der Platine. Wie Abschnitt 3.4.1 erwähnt, ist im Zentrum eine Bohrung, die zur Orientierung und Platzierung der ersten Sensoren sowie zur Ausrichtung des Roboterarms am Messplatz dient. Da die Signalausgänge seitlich am Sensor liegen, sind die Pins zum Messen der Signale links und rechts vom Sensor-Array platziert. Die Distanz vom Mittelpunkt eines Sensors zum nächsten beträgt 4 mm. Die 2x20-Pin-Header ergeben zusammen die große Pinleiste oben an der Platine. Diese sind so platziert, dass die Platine direkt auf das Mikrocontroller-Board gesteckt werden kann. Auf der Unterseite der Platine (Bottom_Layer, im Anhang B.4) sind alle Widerstände sowie die Kondensatoren zur Filterung verbaut (links bzw. rechts zwischen dem Sensor-Array und den Messpins). Diese wurden symmetrisch und jeweils versetzt zueinander angeordnet. Die Widerstände befinden sich unterhalb der großen Pinleiste, die Kondensatoren 30 mm darunter. Die Kondensatoren zur Spannungsstabilisierung befinden sich unterhalb des Sensor-Arrays. Nach Positionierung der Bauteile sind im Abstand von 20 mm vom Mittelpunkt waagerecht und senkrecht in beiden Richtungen Bohrlöcher vorhanden. An den Eckpunkten

des Sensor-Arrays befinden sich weitere vier Bohrlöcher, um eine Befestigung in jeder Richtung zu ermöglichen. An den äußeren Ecken sind ebenfalls vier Bohrungen zur Befestigung des Sensor-Arrays an der Grundplatte des Robotermessplatzes vorhanden.

3.4.3 Verdrahtung und Beschriftung der Bauteile

Das Konzept zum Verdrahten der Sensoren ergibt sich durch die Anordnung zueinander. Damit das Sensor-Array kompakt bleibt, sind die Masseleitungen so verdrahtet, dass sie mittig unter den Sensoren verlaufen. Die Ausgangssignalleitungen verlaufen direkt daneben. Die Verdrahtung der Sensoren ist so konzipiert, dass die Spannungsleitungen außen und die Masseleitungen mittig durch die Sensoren führen. Die Dimension des Arrays vergrößert sich dadurch nicht und kann kompakt bleiben. Auf der oberen Seite (TOP_Layer, im Anhang B.5) sind alle Leiterbahnen der Versorgungsspannungen verdrahtet. Diese sind rot gekennzeichnet. Auf der unteren Seite (Bottom_Layer, im Anhang B.4) sind die Signalleitungen geführt. Abbildungen der Layer sind im Anhang unter den Punkten B.4 und B.5 zu finden.

Die beiden Lagen sind durch Vias (Verbindungspunkte) verbunden. Das geschieht überall dort, wo es zu einer Kreuzung der Leitungen kommen könnte. Abbildung 3.5 zeigt, wie die Verbindungen zwischen den Sensoren erfolgen. Zu sehen ist, dass an der Bohrung in der Mitte zwei Versorgungsleitungen sowie zwei Datenleitungen entlang führen. Dies wurde so konzipiert, um das Sensor-Array kompakt zu halten.

Am anspruchsvollsten ist die Verdrahtung des Sensor-Arrays. Es ist darauf zu achten, dass alle Sensoren auf ihrer linken Seite (VC_X GC_X) und rechten Seite (VS_X GS_X) vertikal mit der entsprechenden Versorgungsleitung verdrahtet sind. Die Signalleitungen verlaufen auf der Unterseite der Platine, wobei im Fokus steht, dass alle Sensoren horizontal in einer Zeile miteinander verdrahtet sind (SIN_LX COS_LX). Die große Pinleiste wird nach der Zuordnung in der Abbildung 3.3 verdrahtet. Damit ist es möglich, die Platine direkt mit dem Mikrocontroller zu verbinden. In der Abbildung B.3 kann die Verdrahtung der gesamten Platine nachvollzogen werden.

Die restliche Verdrahtung ist durch die symmetrische Anordnung der Bauteile nicht so kompliziert wie die des Sensor-Arrays. Die Leitungen zur Spannungsstabilisierung sind nach unten herausgeführt. Die jeweilig benötigten Masseleitungen der Filterschaltung werden von der Spannungsstabilisierung abgezweigt. Die Signalleitungen sind mittig aus dem Sensor-Array geführt und verlaufen von dort aus symmetrisch auf die entsprechenden Bauteile. Die Verbindung zwischen Widerständen und der großen Pinleiste ist in Abbildung B.3 ersichtlich.

Die Beschriftung des Boards ist einfach gehalten. Auf der Oberseite ist der Anfang der großen Pinleiste, sowie die beiden Pinleisten an den Seiten beschriftet. Das Sensor-Array sowie die allgemeine Platinenbezeichnung mit Ort, Datum und Namen des Erstellers

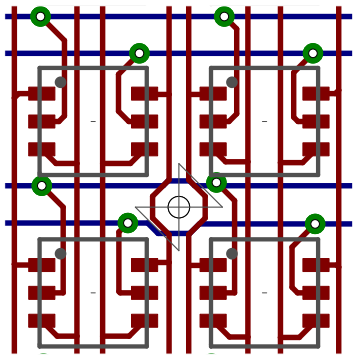


Abbildung 3.5: Ausschnitt aus dem Zentrum des Sensor-Arrays.

befindet sich in der oberen, rechten Ecke. Auf der Unterseite sind die Widerstände und Kondensatoren direkt neben dem Sensor-Array und den Mess-Pins beschriftet.

3.5 Modifikation am verwendeten Mikrocontroller-Board

Aufgrund der hohen Anzahl an Pins ist nicht auszuschließen, dass es zu Konflikten mit dem Mikrocontroller kommt. Mithilfe des Datenblattes und der darin enthaltenen Schaltpläne erschließt sich, dass es zu einem Konflikt mit den in Abbildung 3.6 gekennzeichneten Widerständen kommt. Diese stellen durch ihre 0 Ω-Widerstände eine Verbindung zwischen der Versorgungsleitung 8 und 3 her. Weitere auszubauende Bauteile sind in der Abbildung 3.7 in Rot gekennzeichnet, wobei der Netzwerkanschluss aus Platzgründen entfernt wird.

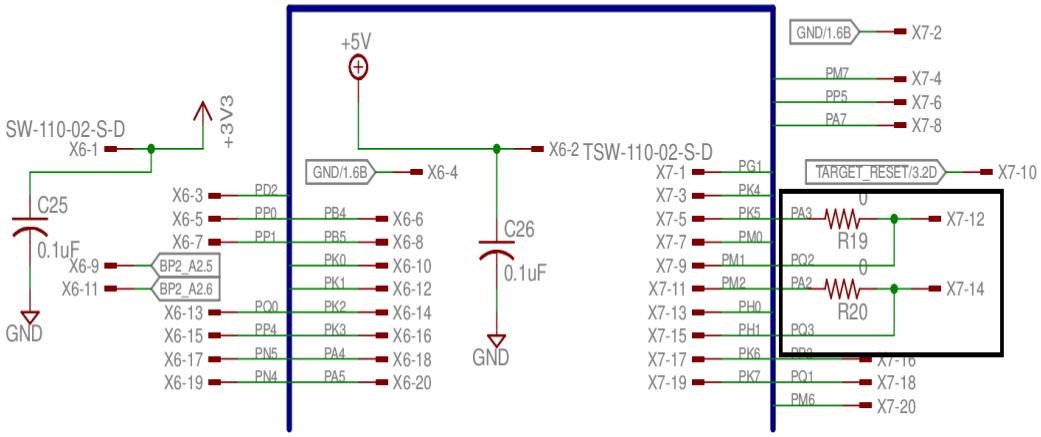


Abbildung 3.6: Im markierten Bereich schwarz sind die 0 Ω-Widerstände R19 und R20 markiert, welche zu einer Verbindung zwischen den Versorgungsleitungen 3 und 8 führen [4].

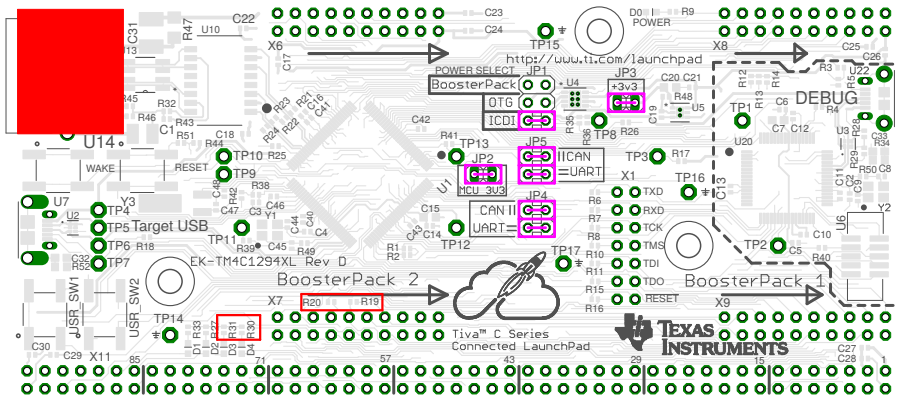


Abbildung 3.7: Modifikationen am Mikrocontroller: In rot sind alle Bauteile markiert, die ausgebaut werden. Die Netzwerkdose wird aus Platzgründen entfernt [4].

3.6 Bestückung der Platine

Die Bestückung der Platine erfolgt sowohl manuell als auch maschinell. Die manuelle Platinenbestückung verläuft folgendermaßen: Als Erstes gilt es, eine gewisse Menge Lötpaste unter Zuhilfenahme der mitgelieferten Schablone auf die Platine zu bringen. Unter einer Lupe bzw. einem Mikroskop werden die Sensoren platziert, somit ist es möglich, die Sensoren akkurat zu setzen. Anschließend kommt die Platine in den smt precision lead-free reflow oven von der Firma elektor, welcher mit einem voreingestellten Temperaturprogramm betrieben wird. Durch das Erhitzen entsteht eine dauerhafte Verbindung zwischen den Sensoren und der Platine. Auf der Rückseite sind lediglich die Widerstände an den Ausgängen, sowie die große Pinleiste bestückt. Beide Verfahren laufen bis auf das Setzen der Sensoren nach dem gleichen Schema ab, mit dem Unterschied, dass diese Sensoren anhand eines Bestückungsroboter platziert werden.

4 Software zum Steuern des Mikrocontrollers und Robotermessplatzes

In diesem Kapitel sind die Erweiterungen und Änderungen an der zur vom ISAR-Projekt zur Verfügung gestellten Software beschrieben. Es wird Bezug auf die Beschaltung und Ansteuerung des Sensor-Arrays genommen sowie die Modifikationen am Robotermessplatz.

4.1 Die Ansteuerung des Mikrocontrollers

Der Mikrocontroller dieser Arbeit dient zur Kommunikation zwischen PC und Sensor-Array. Für diese Schnittstelle wird das TIVA-Board eingesetzt. Dabei ist, ein Grundgerüst zur Kommunikation zwischen TIVA-Board und Sensor-Array bereits vorhanden. Jedoch muss zur besseren Übersicht eine Änderung der Datenstruktur erfolgen. Um dies umzusetzen, werden zwei Header-Dateien erstellt. Die eine zur Einbindung sämtlicher Bibliotheken, und die andere, um Funktionsprototypen einzubinden (vgl. 4.1). Der zentrale Zweck der Softwareerweiterung besteht in der Beschaltung der Versorgungsleitungen und das Auslesen der Datenleitungen. Abbildung 4.2 stellt eine grobe Übersicht der wichtigsten Funktionen dar, die nachfolgend erläutert werden.

Der Programmablauf des Sensor-Arrays Bei einer funktionsfähigen Kommunikation zwischen TIVA-Board und PC kann anschließend die Datenaufnahme sowie die Übertragung starten. Beim Aufruf der Funktion *Get_Values* werden die globalen Arrays zur

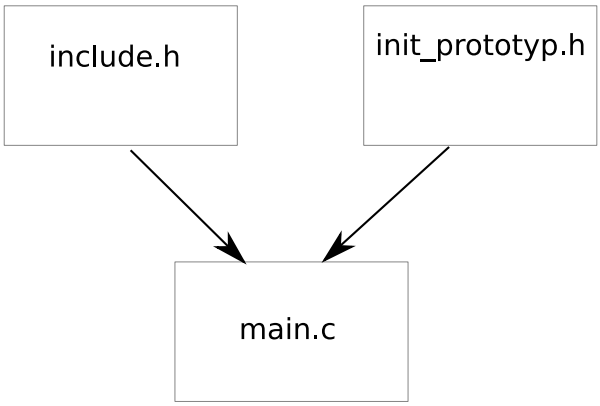


Abbildung 4.1: Aufteilung des Programms nach Bearbeitung und Erweiterung.

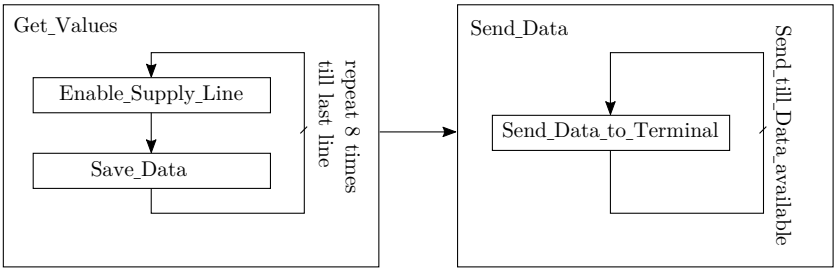


Abbildung 4.2: Übersicht über das Beschalten und Auslesen des Sensor-Arrays.

Abspeicherung der Daten auf null initialisiert. Das Schalten der Spannungen wie im Abschnitt 3.1 beschrieben, startet ab diesem Punkt und durchläuft einen achtfachen Zyklus. Als Erstes wird die Funktion *enable_set_high* aufgerufen und der aktuelle Durchlauf wird der Funktion übergeben. Danach läuft eine empirisch ermittelte Wartezeit von einigen Millisekunden ab. Ist diese durchlaufen, starten die Abtastvorgänge der ADC solange, bis alle Signale anliegen. Nach dem Quittieren des Interrupts werden mit der Funktion *enable_set_low* alle Pins auf Masse geschaltet. Nach einer kurzen Wartezeit werden die ADC-Werte ausgelesen und in den globalen Matrizen abgespeichert.

Am Anfang der Funktion *enable_set_high* werden alle Pins zur Ansteuerung mit der Funktion *enable_set_low* auf Masse geschaltet. Um sicherzustellen, dass das Sensor-Array spannungsfrei ist, wird eine Wartezeit gestartet. Nach Ablauf werden alle Versorgungspins mit der Funktion *gpio_set_input* auf Input (Eingang) geschaltet.

Durch den von *Get_Values* übergebenen Wert wird an dieser Stelle mittels einer Switch-Anweisung in die entsprechende Zeile gesprungen. Dort werden die Pins, welche für das Anschalten der Versorgungsleitung benötigt werden, auf Output (Ausgang) gesetzt und bekommen somit ihr Potenzial zugewiesen. Durch den „break“-Befehl ist die Funktion durchlaufen und wird verlassen.

Ist *Get_Values* durchlaufen, wird die Funktion *vals_to_terminal* ausgeführt. In dieser Funktion werden die gespeicherten Daten mit der Funktion *UART_Send* aus den globalen Matrizen einzeln an das Terminal geschickt. Nach der Übertragung der ersten acht Datensätze erfolgt das abschließende Senden eines Zeilenumbruches zur Trennung der Datensätze. Sind alle 64 Datensätze übermittelt, wird zur Trennung zwischen Cosinus und Sinus ein String gesendet. Dabei werden die Daten der jeweiligen Matrix in einer vorbestimmten Reihenfolge übermittelt, was mithilfe von zwei verschachtelten Schleifen erfolgt. Die innere läuft von 7 abwärts bis einschließlich zur 0, während die äußere entgegengesetzt läuft, was im technischen Aufbau der Platine begründet ist.

4.2 Die Software zum Steuern des Robotermessplatzes

Zur Rotationsmessung einer 360°-Drehung steht ein Robotermessplatz zur im Labor des ISAR-Projektes der HAW zur Verfügung, der im Zuge vorheriger Abschlussarbeiten konstruiert bzw. erweitert wurde ([11] und [2]). Der Messplatz verfügt über einige implementierte und im Folgenden aufgeführten Grundfunktionen:

- Referenzfahrten der drei Achsen-Motoren sowie des Drehtellers zur Rotation
- Eine Benutzeroberfläche zur Ansteuerung des Messplatzes
- Auswahl zwischen automatisierter Messaufnahme oder direkter Ansteuerung der Achsen
- Fertige Positionskoordinaten zur Ausrichtung der Achsen und Anfahren der Messposition
- Es besteht keine Möglichkeit, eine 360°-Messung durchzuführen

Um die 360°-Messung zu ermöglichen, bedarf es Modifikationen an der Programmstruktur, denn der Endschalter des Drehtellers verhindert eine Messung um 360°. Das Deaktivieren des Endschalters ist aufgrund der Referenzfahrt nicht möglich, da diese sich an dem Schalter orientiert. Das Überfahren des Endschalters ist nach Angaben der Vorgängerarbeiten nur in einer Richtung möglich. Durch folgende Modifikationen kann dieses Problem umgangen werden:

- im Unterprogramm *rmp_3_move_relative_position.m* in der Zeile 157 auskommentieren des Endschalters
- in der Datei *rmp_3_init_stage_system.m* Zeile 153 und 160 auf 0 gesetzt
- in der Datei *rmp_3_mtx_rotation.m* Zeile 114 invertieren. Änderung der Drehrichtung in die mathematisch negative Richtung [1]

In der Datei *rmp_3_move_relative_position.m* bewirkt die verknüpfte „Ver-ODER-ung“, in der Zeile 157 durch die Abfrage des Endschalters, das Unterbrechen des Messverfahrens. Diese Abfrage wurde auskommentiert und beeinträchtigt die Grundfunktion dabei nicht. In der Datei *rmp_3_init_stage_system.m* wurden die Grenzen des Drehtellers auf null gesetzt. Um eine volle Umdrehung zu erreichen, sind ggf. mehrere Drehungen durchzuführen. Durch das Invertieren des Vorzeichens in der Datei *rmp_3_mtx_rotation.m*, Zeile 114, ist es möglich, in dieser Drehrichtung den Endschalter zu überfahren. Die Drehrichtung ist damit mathematisch negativ und somit gleichermaßen korrekt für das Sensor-Array ausgerichtet.

[1] Mathematisch positive Richtung ist gegen den Uhrzeigersinn; Mathematisch negative Richtung ist mit dem Uhrzeigersinn.

4.3 Ablauf des Messplans

In diesem Abschnitt wird beschrieben, wie die Messungen mithilfe des Robotermessplatzes durchgeführt werden. Für die folgende Betrachtung werden zwei Koordinatensysteme festgelegt. Das erste System beschreibt die relative Position des Magneten über dem Sensor-Array, wobei der Punkt P_S den Wert $(X_S\ ;\ Y_S)$ beinhaltet. Das zweite Koordinatensystem stellte die absolute Positionierung des Magneten gegenüber der Nullposition des Arrays dar, welche im Mittelpunkt P_0 liegt. Die dazugehörigen Einheiten werden in Millimeter angegeben und eine Darstellung des Prinzips ist in der Abbildung 4.3 zu sehen.

Das Anfahren sowie die Durchführung der Messungen erfolgt mit dem Programm *360_grad Messung.m*. Der automatische Ablauf des Programms sieht wie folgt aus: Es wird ein Vektor mit den Rotationsdaten initialisiert, welche über die maximale Rotation, den Startpunkt und die Schrittweite (Angaben in Grad) entscheiden, ab der die nächste Messung erfolgt. Danach werden die maximalen Grenzen des abzufahrenden Feldes initialisiert. Nach der Initialisierung wird eine Datei mit dem aktuellen Datum erstellt. Anschließend wird sowohl ein Plot des Messfeldes als auch der Messungen erstellt. Die blauen Punkte stellen die zu messenden Punkte dar, während der schwarze Punkt die Position des Mittelpunktes an P_0 kennzeichnet, zu sehne in Abbildung 4.4. Nach der Erstellung der Grafik, wird die erste Position angefahren die zu Messen ist. Das ist der Sensor A1 und der Punkt P_A=(−14 mm; −14 mm). Danach kommt es zu einer Abfrage der Zeile, in welcher sich der Roboter befindet. Diese Information wird benötigt, um die Unterdatei des Messpunktes zu erstellen. Im Namen der Datei wird zum einen die Messbezeichnung und zum anderen die Position des Sensors bzw. dessen Koordinaten angegeben. Ist das erfolgt, werden die Arrays der Sensordaten sowie die benötigten Variablen zur Referenzfahrt und Durchführung der Messung initialisiert. Damit der Magnet

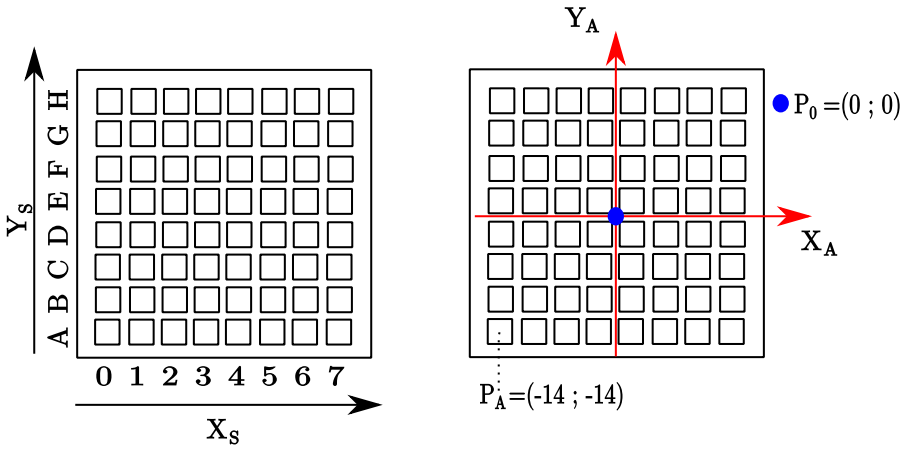


Abbildung 4.3: Gegenüberstellung der Koordinaten des Messschemas (links) zum Sensor-Array (rechts).

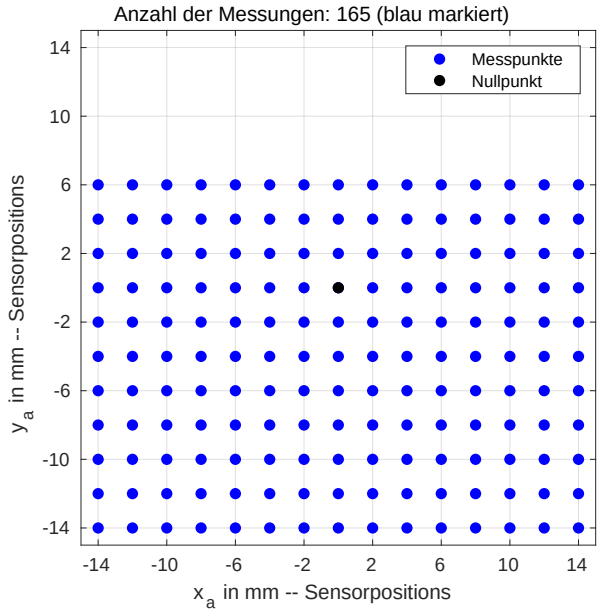


Abbildung 4.4: Darstellung der ab zufahrenden Messpunkte über dem Array.

zur Beginn jeder Messung dieselbe Ausgangsposition hat, wird eine Initialisierungsfahrt durchgeführt und der Magnet anschließend positioniert. Es folgt die Messaufnahme, nach deren Abschluss der Roboter zum nächsten Messpunkt fährt und diesen Vorgang bis zum Erreichen der letzten Koordinate fortfährt.

5 Evaluation und Messerprobung

Die Funktionsüberprüfung des Sensor-Arrays erfolgt in diesem Kapitel. Als Erstes wird auf die Hardware eingegangen und dem folgend auf die Funktion des Arrays. Es werden verschiedene Messungen mit unterschiedlichen Magneten durchgeführt, wobei es zum Vergleich zwischen dem vorhandenen AMR-Array und dem TMR-Array kommt.

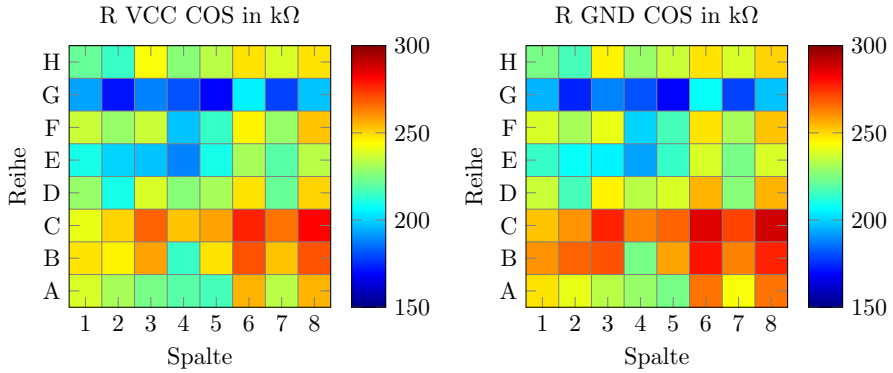
5.1 Sensor-Array

Hier werden alle Messungen dargestellt, welche zur Inbetriebnahme und Funktionsprüfung des Sensor-Arrays nötig sind. Die Halbbrückenwiderstände jedes einzelnen Sensors werden gemessen und ausgewertet. Anschließend wird die Pixelzuweisung in kartesischen Koordinaten durchgeführt und das Zeitverhalten der Ausgangsspannungen analysiert.

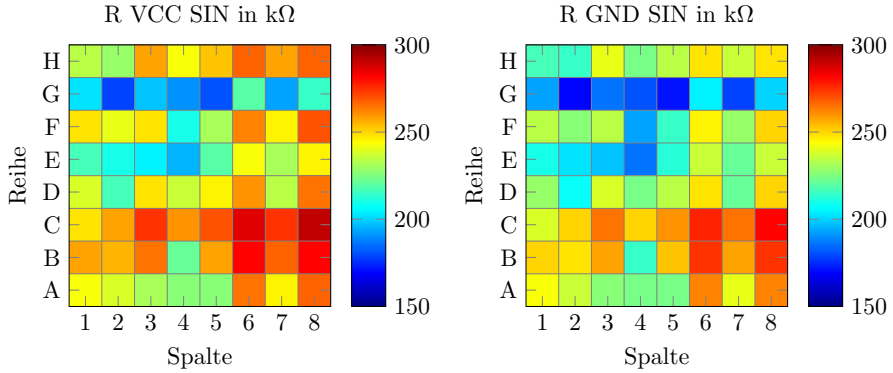
5.1.1 Widerstandsmessung

In diesem Unterpunkt sind die Widerstände der TMR-Elemente im Fokus. Der wesentliche Bestandteil des TMR-Effekts beruht auf den hohen Widerstandsänderungen. Aufgrund dessen, dass die Sensoren, wie in Abbildung 3.1 dargestellt, in Halbbrücken gruppiert sind, erfolgen die Messungen vom Kontakt $VC_$X über den TMR-Widerstand R_{VC} zum jeweiligen Ausgang $COS_$LX. Äquivalent dazu die Messung vom Kontakt $GC_$X über den TMR-Widerstand R_{GC} zum jeweiligen Ausgang $COS_$LX. Die gleiche Prozedur wird anschließend auch für die andere Halbbrücke, über die TMR-Widerstände R_{VS} und R_{GS} durchgeführt, das X steht als Platzhalter für die jeweilige Zahl von 1 bis 8. Die gemessenen Widerstandswerte sind in den, im Anhang befindlichen Tabellen D.1 und D.2 einzusehen.

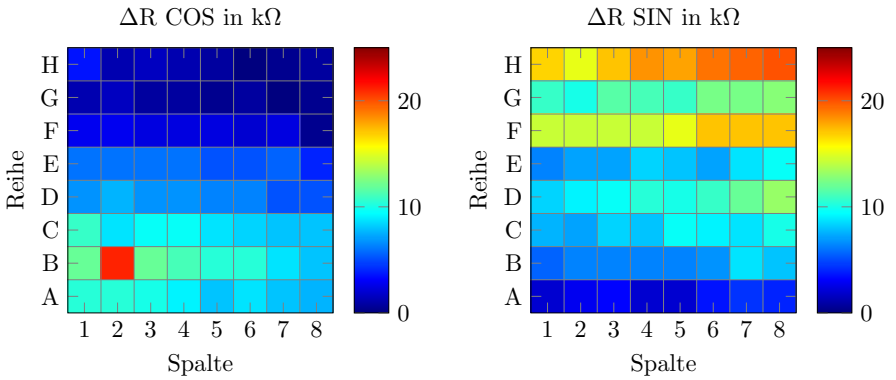
In Abbildung 5.1 ist die Gegenüberstellung der beiden Halbbrücken in kartesischer Darstellung wiedergegeben. Die Abbildung 5.1(a) zeigt die Widerstände der Versorgungs- und Masseleitungen der COS-Seite und Abbildung 5.1(b) die SIN-Seite. Der Wertebereich liegt dabei zwischen 150 kΩ–300 kΩ. Es ist ersichtlich, dass die Widerstandswerte im unteren Bereich höher sind als im oberen. Die Reihe G weist erkennbar die niedrigsten Werte auf. In der Abbildung 5.1(c) ist die Differenz zwischen den jeweiligen Widerständen in einer Halbbrücke dargestellt, was die Symmetrie der jeweiligen Halbbrücke offeriert. Der Wertebereich liegt hier zwischen 0 bis 25 kΩ. Auf der rechten Seite hat der Sensor B2 zwischen den Halbrücken eine höhere Differenz. Auf der linken Seite weisen die Reihen H und F eine höhere Differenz auf, wonach die SIN- im Vergleich zur COS-Seite nicht ausgeglichen sind.



(a) Darstellung der Widerstände der COS-Seite R_{VC} und R_{GC}.

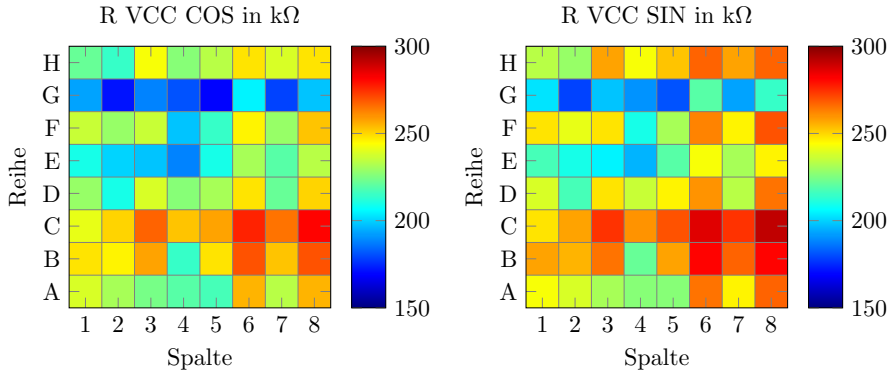


(b) Darstellung der Widerstände der SIN-Seite R_{VS} und R_{GS}.

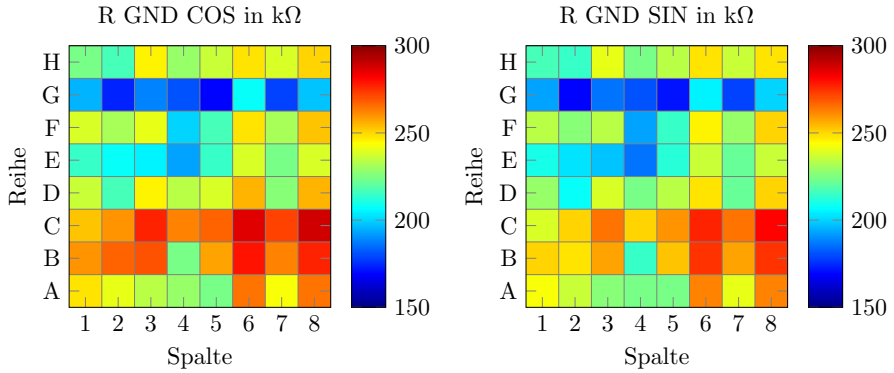


(c) Gegenüberstellung der Differenzen links R_{VC} - R_{GC}, rechts R_{VS} - R_{GS}.

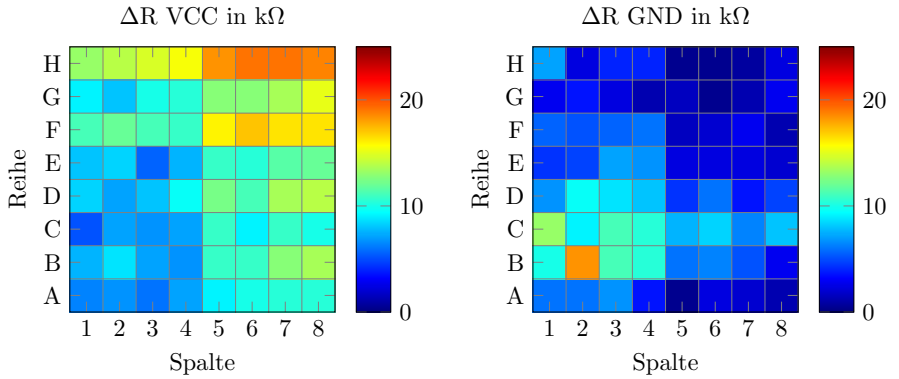
Abbildung 5.1: Links sind die Widerstandswerte der COS-Seite dargestellt; Rechts die der SIN-Seite; Unten sind die Differenzen der linken und rechten Spalte dargestellt.



(a) Gegenüberstellung der Widerstände R_{VC} und R_{VS}, links die COS-Seite rechts die SIN-Seite.



(b) Gegenüberstellung der Widerstände R_{GC} und R_{GS}, links die COS-Seite, rechts die SIN-Seite.

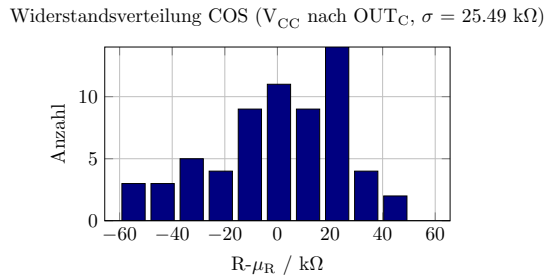


(c) Gegenüberstellung der Differenzen links R_{VC} - R_{VS}, rechts R_{GC} - R_{GS}.

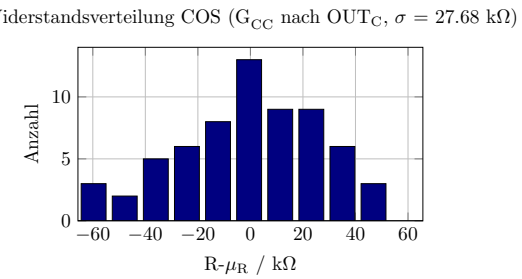
Abbildung 5.2: In dieser Abbildung sind die Widerstände der Halbbrücken so angeordnet wie in 2.3(b) dargestellt. Oben links R_{VC}; oben rechts R_{VS}; unten links R_{GC}; unten rechts R_{GS}.

In Abbildung 5.2 sind die Widerstände der jeweiligen Versorgungsspannung bzw. Masseleitung gegenübergestellt. Es ist zu erkennen, dass die beiden Halbbrücken nicht symmetrisch arbeiten. Auffallend dabei ist, dass die Widerstandswerte der jeweils anderen GND-Seite besser zueinanderpassen, als die Eigenen. Die Abbildung 5.2(c) ist eine Gegenüberstellung der Differenzen zwischen den Versorgungswiderständen und Massewiderständen, wobei die Symmetrie zwischen der Abbildung 5.2(b) und 5.2(c) auffällt.

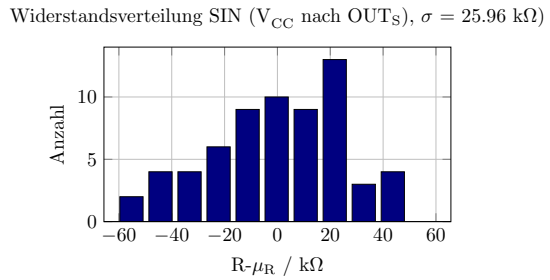
Aus den Daten den Tabellen D.1 und D.2 wurden die Histogramme in Abbildung 5.3 erstellt. Dabei ist bei näherer Betrachtung ersichtlich, dass das Histogramm 5.3(a) rechts schief ist, genau so wie 5.3(c). Das Histogramm 5.3(b) entspricht einer Normalverteilung, wobei 5.3(d) eine Tendenz nach rechts aufweist. Die Standardabweichung σ sowie der Mittelwert μ_R der jeweiligen Sensorelemente können der Tabelle 5.1 entnommen werden.



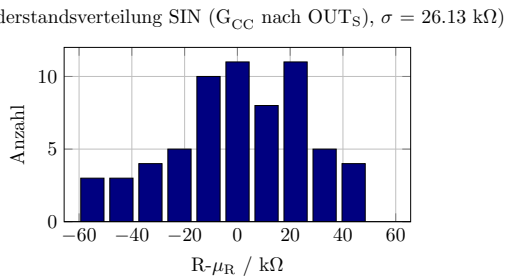
(a) Widerstandsverteilung des Widerstandes R_{VC}.



(b) Widerstandsverteilung des Widerstandes R_{GC}.



(c) Widerstandsverteilung des Widerstandes R_{VS}.



(d) Widerstandsverteilung des Widerstandes R_{GS}.

Abbildung 5.3: Histogramme der TMR-Widerstandselemente R_{VC}, R_{GC}, R_{VS} und R_{GS}.

Tabelle 5.1: Mittelwerte und Standardabweichungen der Widerstände aus den Tabellen D.1 und D.2 für die vier Sensorelemente.

Sensorelement	Mittelwert μ_R kΩ	Standardabweichung σ kΩ
R_{VC}	229,54	25,49
R_{GC}	235,31	27,68
R_{VS}	240,54	25,96
R_{GS}	230,35	26,13

5.1.2 Messungen der Ausgangsspannung an den Signalleitungen

In diesem Kapitel erfolgt die Überprüfung der Signalausgänge, wozu exemplarisch zwei Messungen durchgeführt werden. Die Messungen werden wie folgt durchgeführt: Die Ausgangssignalspannung wird mit einem Oszilloskop des Typs (Tektronix – MSO 3034) aufgezeichnet.

Gemessen wird an den Messpunkten, welche seitlich an der Platine platziert sind. In Abbildung 5.4 ist die Messung an der COS_L1 dargestellt. Es ist zu erkennen, dass der Spitzenwert der Spannung bei ca. 1,66 V liegt, was ca. der halben Betriebsspannung entspricht. Eindeutig zu erkennen sind die acht Schaltvorgänge der Versorgungsleitungen, welche für die Dauer von 4,1 ms aktiv sind. Die Pausenzeiten zwischen den Schaltvorgängen betragen auch 4,1 ms und ein voller Zyklus über die acht Schaltvorgänge dauert 62 ms. Die Abbildung 5.5 die Messung an der SIN_L8. Die Spannung beträgt auch hier 1,66 V, was der halben Betriebsspannung entspricht. Auch hier sind die acht Schaltvorgänge und Pausen zu erkennen, die eine Länge von 4.1 ms haben und deren Zyklus bei 61,5 ms liegt.

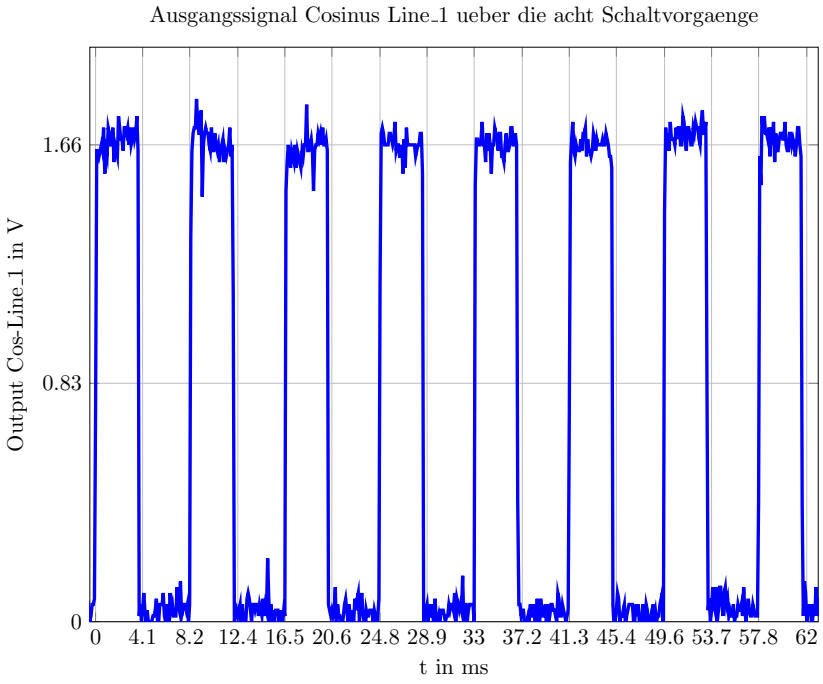


Abbildung 5.4: Ausgangssignal der COS-Leitung 1 über alle Schaltvorgänge.

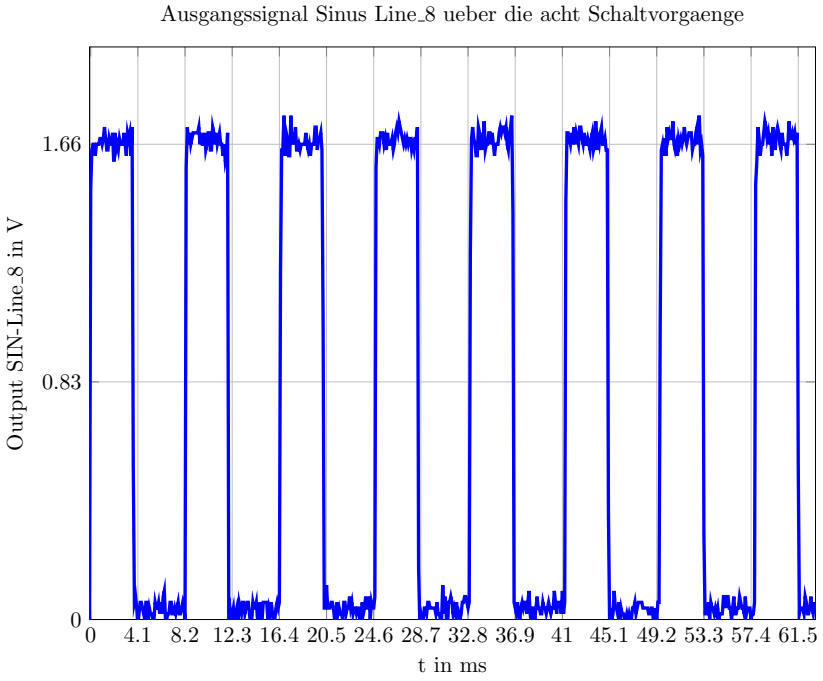


Abbildung 5.5: Ausgangssignal der SIN-Leitung 1 über alle Schaltvorgänge.

5.1.3 Graphische Zuordnung der Sensorelemente

Die einwandfreie Zuordnung der Sensoren in der grafischen Darstellung ist erforderlich, um aussagekräftige Messergebnisse zu erhalten. Die Sensoren müssen auch in der Array-Anordnung eigenständig und unabhängig (verkopplungsfrei) genaue Daten liefern. Um dies zu überprüfen, ist eine Messung in Verbindung mit dem Mikrocontroller-Programm und dem Host-PC notwendig. Zu diesem Zweck wird das Sensor-Array mit dem Mikrocontroller verbunden.

Die aufgenommenen Daten werden dazu in einem Programm erfasst und in Echtzeit am PC dargestellt, was in Abbildung 5.6 aufgezeigt wird. Es ist dabei zu beachten, dass die alphabetischen Zeilen die Versorgungsleitungen darstellen. Der Punkt (A,1) entspricht auf der Platine dem Punkt (1,1) und (H,8) steht für den Punkt (8,8). Das Pixel (A,1) entspricht jedoch aus technischer Betrachtungsweise dem achten Element der ersten Versorgungsleitung. Der Grund hierfür ist, dass die Spalten 1-8 in umgekehrter Reihenfolge die Datenleitungen 8-1 der Hardware darstellen. Spalte 1 entspricht somit der Datenleitung 8 und Spalte 8 der Datenleitung 1. In der Abbildung 5.6 sind alle Versorgungsleitungen sowie Signalleitungen eingeschaltet.

Mithilfe des erstellten Programms lassen sich die einzelnen Spannungsversorgungsleitungen ein- bzw. ausschalten, wodurch als Erstes die entsprechenden Zeilen bestimmt werden können. Zu dieser Identifizierung wird eine Versorgungsleitung eingeschaltet, wonach lediglich die Sensoren der ausgewählten Leitungen mit Spannung versorgt und ausgelesen werden können.

Zum Erstellen der Abbildung 5.7 wurde im Programm die fünfte Versorgungsleitung eingeschaltet und es zeigt sich, dass die Zeile E dieser Leitung entspricht (ausgehend davon, dass, wie vorher erläutert, die Zählung bei A beginnt). Im nächsten Schritt wird eine ausgewählte Signalleitung freigeschaltet, während die restlichen inaktiv bleiben. In der Funktion C.5 *init-ADC.c* werden nur die Zeilen 30 und 43 freigegeben, womit die Datenleitung 5 (SIN/COS) geschaltet sind. Das Pixel (E,4) in der Abbildung 5.8 zeigt somit den fünften Sensor der fünften Datenleitung nach der zuvor erläuterten Zuweisung. Das Zentrum des Arrays wird dementsprechend durch die vierte und fünfte Spannungsversorgungsleitung und der zugehörigen Datenleitungen betrieben. Dies erfolgt nach dem bereits erklärten Schema, durch die Funktionen *enable-set-high.c* und *init.ADC.c*, in denen die entsprechenden Bereiche geschaltet werden, dargestellt in 5.9. Es ist auch möglich, jeweils eine Halbbrücke der Sensoren des Sensor-Arrays zu schalten, siehe Abbildung 5.10. Die Steuerung der Hardware erfolgt in der Entwicklungsumgebung „Code Composer Studio“, wobei jegliche Änderungen vorher auf den Mikrocontroller übertragen werden.

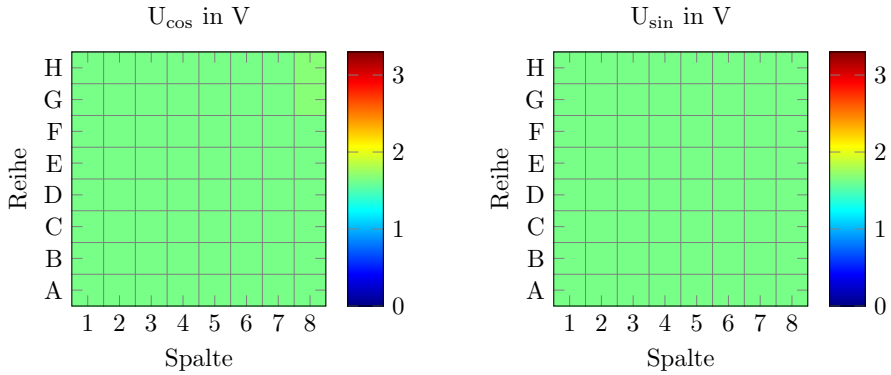


Abbildung 5.6: Pixeldarstellung der beiden Sensorhälften des Arrays; Links die COS-, rechts die SIN-Seite.

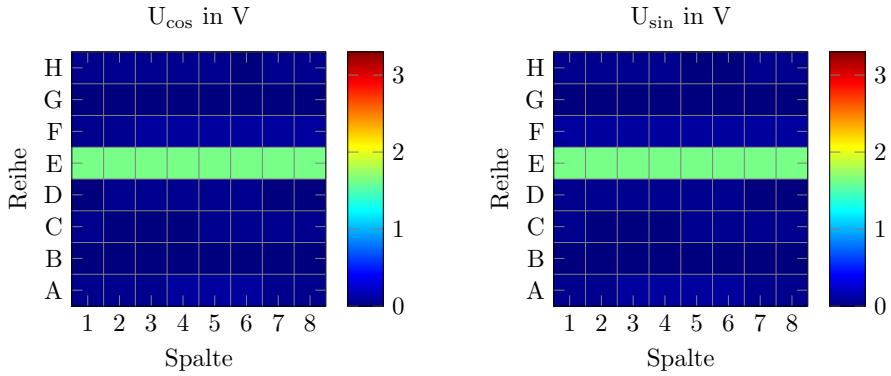


Abbildung 5.7: Inbetriebnahme der Sensoren in der Reihe E; die Ausgänge aller Sensoren sind eingeschaltet.

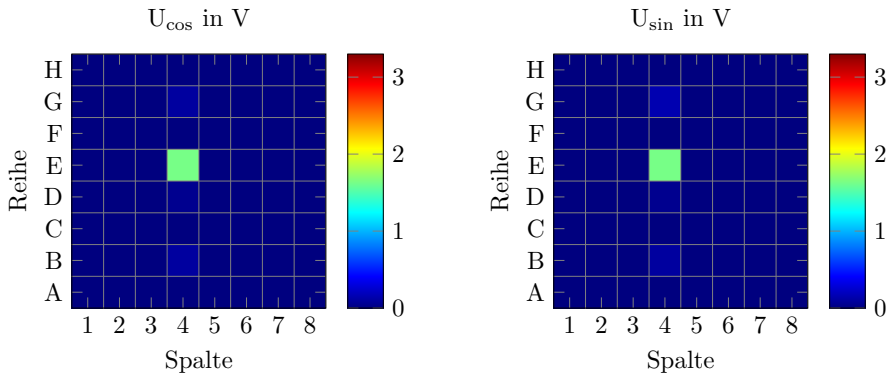


Abbildung 5.8: Einschalten des Sensor E4; die Ausgänge der anderen Sensoren sind ausgeschaltet.

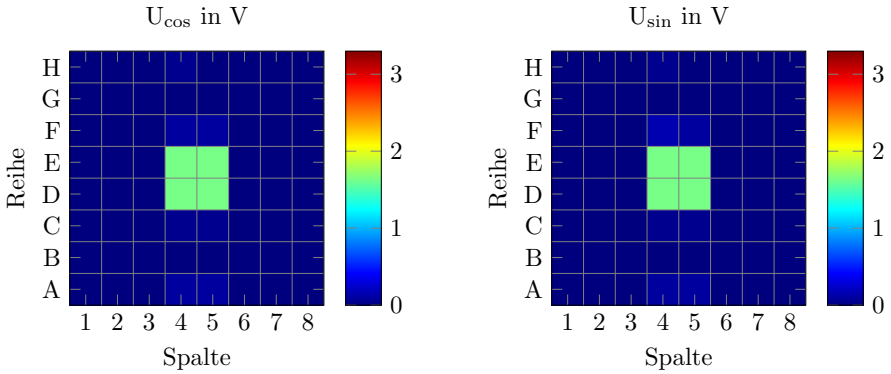


Abbildung 5.9: Inbetriebnahme des Sensor-Arrays Zentrum mit den Sensoren D4 und D5 sowie E4 und E5.

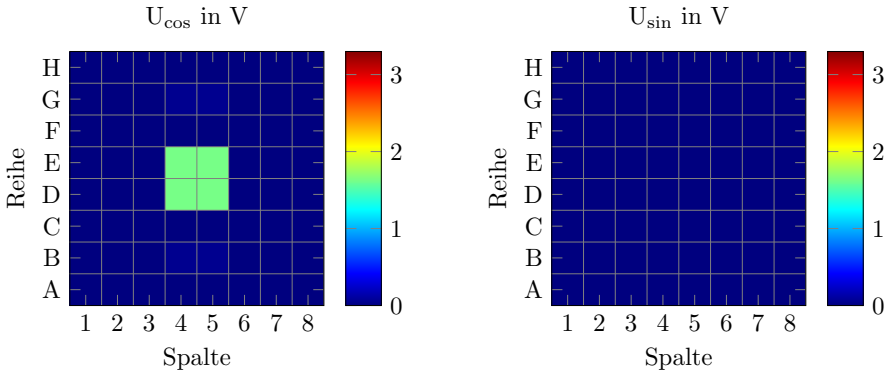


Abbildung 5.10: Einseitiges Einschalten der COS-Seite des Sensor-Arrays.

5.2 Rotationsmessung mittels Robotermessplatz

In diesem Abschnitt werden die Vergleichsmessungen zwischen dem KMZ60 AMR-Array und dem NVE AAT 001-10e-TMR-Array mit dem Robotermessplatz dargestellt und verglichen. Um die Messungen zu beginnen, müssen einige Änderungen am Messplatz vorgenommen werden. Dazu gehört das Auswechseln der Grundplatte des Messplatzes aus Eisen. Diesbezüglich wurde eine Adapterplatte aus Holz zugeschnitten und angepasst. Eine Schablone des TMR-Arrays mit den passenden Bohrungen wurde der Abbildung B.6 entnommen. Die vier äußeren Bohrungen und die Dimension des Arrays wurden auf eine Schablone übertragen. Da das Array auf den Mikrocontroller gesteckt wird, werden die Bohrlöcher zur Befestigung aus Abbildung 3.7 auf die gleiche Schablone übertragen, diese ist in Abbildung B.7 dargestellt. Der Robotermessplatz verfügt über eine voreingestellte Positionsausrichtung, die sich genau auf die Mitte des Sensor-Arrays ausrichtet. Diese Position wurde auf die Adapterplatte übertragen und das Array mittig darauf positioniert.

Aus der vorherigen Abschlussarbeit stehen zwei Magnete, mit entsprechenden Halterungen zur Verfügung. Zur Untersuchung des Verhaltens des Sensor-Arrays auf einen Dipol wurden Kugelmagneten mit verschiedenen Durchmessern bestellt. Für die Kugelmagnete wurde eine neue Halterung angefertigt, abgebildet in 5.11. Erstellt wurden zwei von diesen Halterungen mit einem 3D-Drucker. Die Kugeln werden zwischen zwei dieser Halterungen platziert. In der folgenden Tabelle 5.2 sind die Daten der verschiedenen Magnete sowie die wichtigsten Eigenschaften für die Messungen aufgelistet.

5.2.1 Messungen mit dem KMZ60-AMR-Sensor-Array

Hier werden die Messungen, die mit dem KMZ60-AMR-Array erstellt wurden, dargestellt und beschrieben. Das Array wird mit einer Betriebsspannung von 5 V betrieben. Darüber hinaus verfügt das Array über Filterschaltungen für das Ausgangssignal. Die minimale Feldstärke des Gebermagnetes zum Betreiben des KMZ60 beträgt 25 $\frac{kA}{m}$. Die Größe des

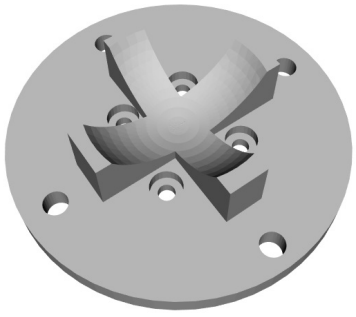


Abbildung 5.11: Erstellte Kugelhalterung für den Messplatz.

Tabelle 5.2: Daten der Magneten zur Messung am Robotermessplatz. Kugel-, Quader- und Rechteckmagneten.

Magnetform	Abmessungen mm	Abstand zum Array mm	Feldstärke min. kA/m	Feldstärke max. kA/m
große Kugel	d=26	19	2,68	31,8
kleine Kugel	d=19	19	2,28	27,8
Quader	10x10x10 (BxHxT)	19	3,5	21,1
Rechteck	40x20x40 (BxHxT)	19	1,3	22,9

Sensor-Arrays beträgt von der Mitte des ersten Sensors zum letzten 5,5 cm [1]. Gemessen wurde von den Punkten (E,5), (A,8) und (F,3). Die dargestellten Messungen werden von einem festen Punkt aus betrachtet. Das ist der Punkt (E,5), das Zentrum des Sensor-Arrays. In der Tabelle 5.3 sind die Auffälligkeiten in den Messungen aufgelistet. Die Ausgangssignalspannung hat einen Wert von 1,6 V (Spitze-Spitze). Folgende Attribute, die betrachtet wurden, sind: Die Auswertungen aus der Tabelle 5.3 zeigen, dass die Vektorverläufe nicht einheitlich in eine Richtung zeigen. Die Signalverläufe der Kugeln, die als Dipol dienen, verhalten sich ähnlich. Dabei ist ersichtlich, dass die Kurvenverläufe, die verhältnismäßig weit entfernt vom Zentrum sind, erhebliche Verzerrungen aufweisen. Jedoch wiederholen sich auch diese Verzerrungen periodisch ab 180°.

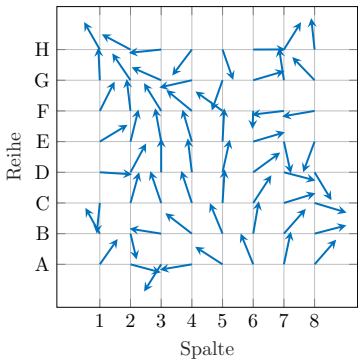
- Der Verlauf der SIN- und COS-Signale
- Die Kurvenform der Signale
- Der Verlauf der Vektoren in Abhängigkeit zum Magnetfeld

Die Messungen erfolgten mit den Magneten aus der Tabelle 5.2. Die Vektorverläufe in den nachfolgenden Abbildungen sind bei 45° Rotation des Gebermagneten aufgenommen worden. Die Periodizität der Sensoren ist deutlich erkennbar bei jeder Kurvenform ab 180°.

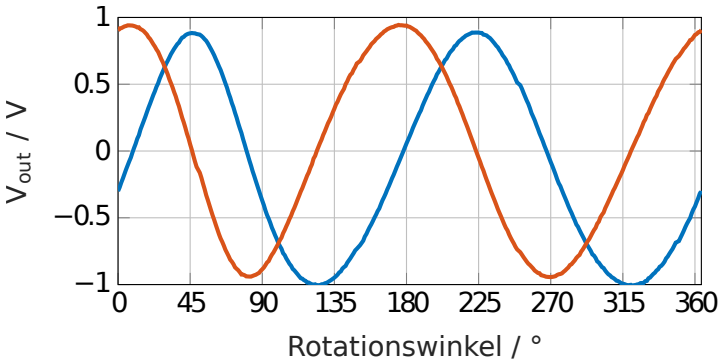
[1] Für weiter Informationen bezüglich des KMZ60 wird auf das Datenblatt verwiesen [7].

Tabelle 5.3: Die Messungen mit den Verschieden Magneten mittels Robotermessplatz werden hier aufgelistet.

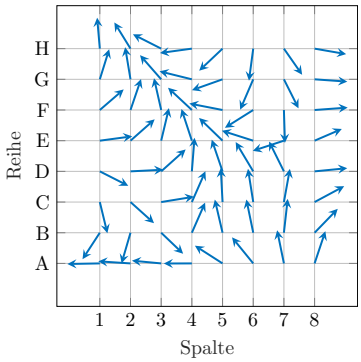
KMZ-AMR-Sensor-Array			
Abbildung 5.12	Ausgangsspannung 1,6 V	Großer Kugelmagnet	
Messpunkt	Verlauf	Kurvenform	Vektorverlauf
(E,5)	Sinusförmig	normal	gehäuft (E,5); Richtung (H,1)
(F,3)	verformter SIN u. COS	invertierter Start	gehäuft (F,3); Richtung (H,1)
(A,8)	Sinusförmig	negativer Start	gehäuft (A,8); Richtung (H,1)
Abbildung E.2	Kleiner Kugelmagnet		
Messpunkt	Verlauf	Kurvenform	Vektorenverlauf
(E,5)	Sinusförmig	normal	gehäuft (E,5); Richtung (H,1)
(F,3)	verformter SIN u. COS	invertierter Start	gehäuft (F,4); Richtung (H,2)
(A,8)	Sinus ist invertiert	negativer Anfang von SIN	gehäuft (A,7); Richtung (H,4)
Abbildung E.3	Quadermagnet		
Messpunkt	Verlauf	Kurvenform	Vektorenverlauf
(E,5)	Sinusförmig	normal	gehäuft (E,5); Richtung (H,1)
(F,3)	SIN dreieckig; COS pulsiert	verformt	gehäuft (F,3); Richtung (H,2)
(A,8)	dreieckig	negativer Anfang bei −0,5 V	gehäuft (B,8); Richtung (E,6)
Abbildung E.4	Rechteckigermagnet		
Messpunkt	Verlauf	Kurvenform	Vektorenverlauf
(E,5)	Sinusförmig	normal	gehäuft(E,5); Richtung (H,1)
(F,3)	gestreckter COS	verformter SIN	gehäuft (E,5); Richtung (H,2)
(A,8)	Sinusförmig	invertierter SIN	gehäuft(E,6); Richtung (H,3)



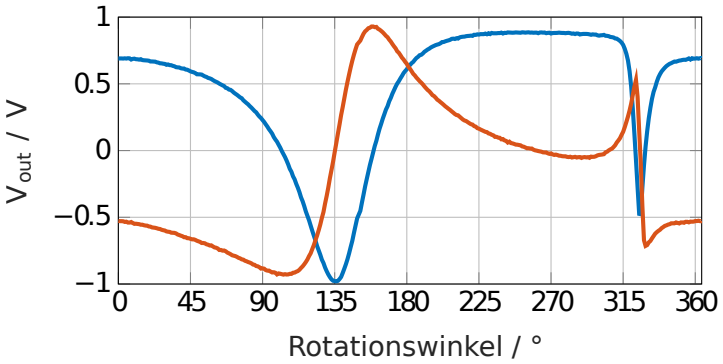
(a) Vektordarstellung der Ausgangssignale bei 45° Rotation des Magneten über der Mitte des Arrays.



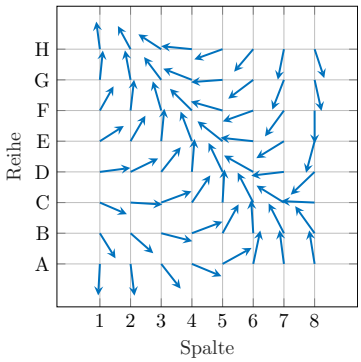
(b) SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5) in Abhängigkeit der Rotation des Magneten über der Mitte des Arrays.



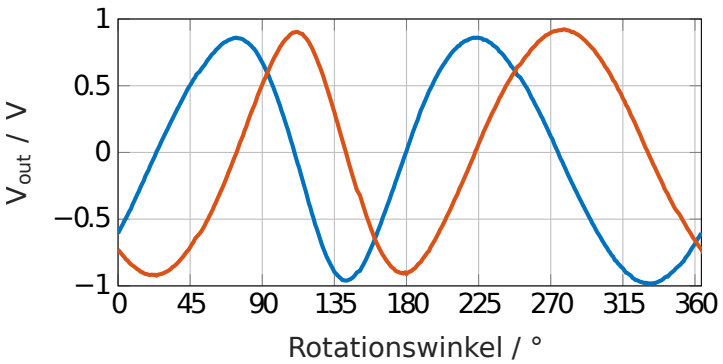
(c) Vektordarstellung der Ausgangssignale bei 45° Rotation des Magneten über dem Sensor (F,3).



(d) SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5) in Abhängigkeit der Rotation des Magneten über dem Sensor (F,3).



(e) Vektordarstellung der Ausgangssignale bei 45° Rotation des Magneten über dem Sensor (A,8).



(f) SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5) in Abhängigkeit der Rotation des Magneten über dem Sensor (A,8).

Abbildung 5.12: Messungen mit dem KMZ-60 Sensor-Array und einem Kugelmagneten (d = 26 mm) im Abstand von 19 mm bei drei verschiedenen Positionen über dem Sensor-Array.

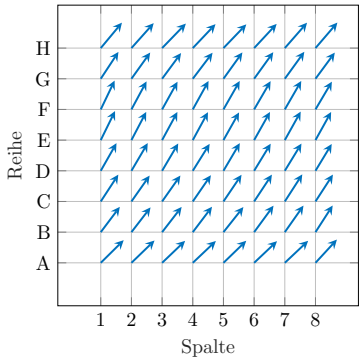
5.2.2 Messungen mit dem NVE-TMR-Sensor-Array

In diesem Anschnitt werden die Messungen des TMR-Sensor-Arrays dargestellt und diskutiert. Das Sensor-Array wird mit einer Betriebsspannung von 3,3 V betrieben. Das Array hat keine Verstärkung, Spannungsstabilisierung oder Filterschaltungen. Es besteht nur aus diskreten Bauteilen. Die benötigte Feldstärke des Arrays beträgt ca. 16 $\frac{\text{kA}}{\text{m}}$. Die Periodendauer der SIN- und COS-Ausgangssignale umfasst eine volle Umdrehung, und somit 360°. Das TMR-Sensor-Array hat die Abmessungen von 2,8 cm, ausgehend vom Zentrum des ersten Sensors bis zum Letzten. Gemessen wurde hier über den Sensoren (E,5), (F,5) und (A,8). Die Betrachtung erfolgt auch hier vom Punkt (E,5), annähernd die Mitte des Sensor-Arrays. Eine Übersicht der Auswertung ist in der folgenden Tabelle 5.4 dargestellt.

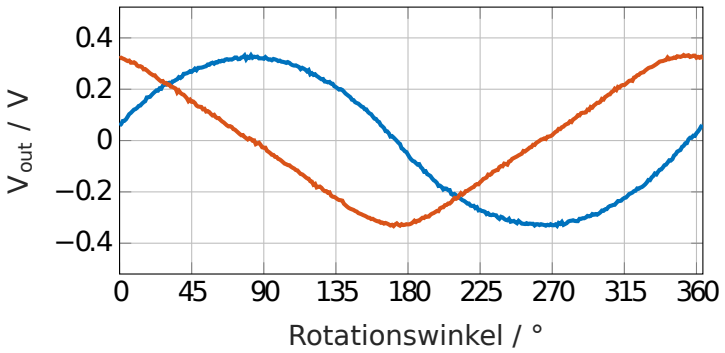
Die Auswertungen der Messungen ergeben, dass die Vektorverläufe der Sensoren bei den TMR-Sensor-Arrays annähernd alle in dieselbe Richtung zeigen mit einem Winkel von 45°. Die Signalverläufe der Kugelmagneten weisen einen gleichen Kurvenverlauf auf.

Tabelle 5.4: Die Messungen mit den Verschieden Magneten mittels Robotermessplatz werden hier aufgelistet.

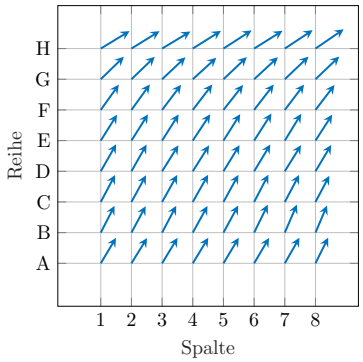
AMR-TMR-Sensor-Array			
Abbildung 5.12	Ausgangsspannung 0,6 V	Großer Kugelmagnet	
Messpunkt	Verlauf	Kurvenform	Vektorverlauf
(E,5)	Sinusförmig	normal	fast alle Sensoren 45°
(F,3)	Sinusförmig	normal	Zeile H verkippt
(A,8)	Sinusförmig	gewellter COS Verlauf verformter SIN	Zeile H und G verkippt
Abbildung E.2	Kleiner Kugelmagnet		
Messpunkt	Verlauf	Kurvenform	Vektorenverlauf
(E,5)	Sinusförmig	Sin-dreieckig	Zeile A u. B verkippt
(F,3)	Verformter SIN u. COS	verformt	Zeile A-D verformt
(A,8)	Sinusförmig	verformt annähernd gleich	verkippte Spalte(1-3) Zeile(F-G)
Abbildung E.3	Quadermagnet		
Messpunkt	Verlauf	Kurvenform	Vektorenverlauf
(E,5)	Sinusförmig	COS gestreckt	Zeile A-B und H verkippt
(F,3)	Sinusförmig	leicht verzerrt	Zeile E 45°
(A,8)	Sinusförmig	beide Signale Verzerrt	Zeile D 45°
Abbildung E.4	Rechteckiger Kugelmagnet		
Messpunkt	Verlauf	Kurvenform	Vektorenverlauf
(E,5)	Sinusförmig	COS getreckt	Zeile C-D und G 45°
(F,3)	Sinusförmig	leicht verzerrt	Zeile D und E 45°
(A,8)	Sinusförmig	verzerrt	Zeile C und D 45°



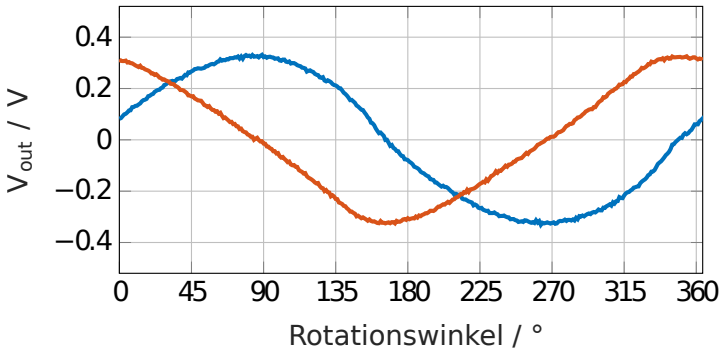
(a) Vektordarstellung der Ausgangssignale bei 45° Rotation des Magneten über der Mitte des Arrays.



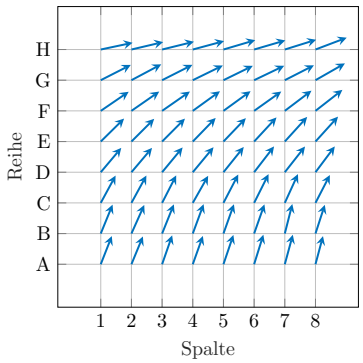
(b) SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5) in Abhängigkeit der Rotation des Magneten über der Mitte des Arrays.



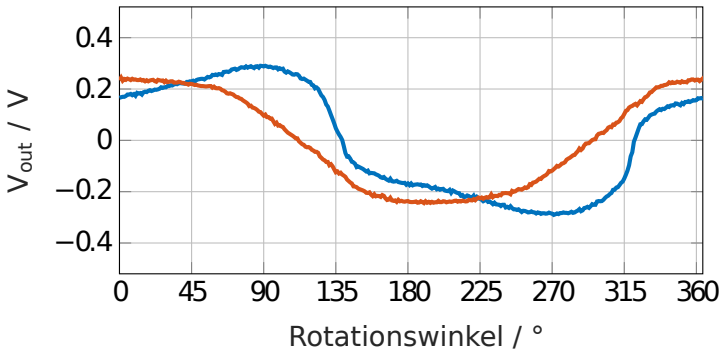
(c) Vektordarstellung der Ausgangssignale bei 45° Rotation des Magneten über dem Sensor (C,3).



(d) SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5) in Abhängigkeit der Rotation des Magneten über dem Sensor (F,3).



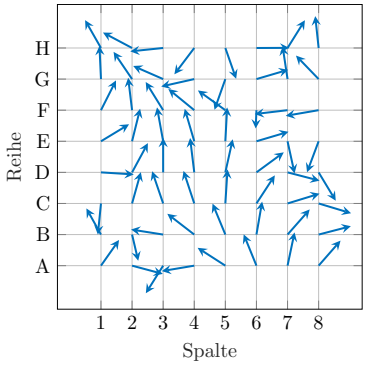
(e) Vektordarstellung der Ausgangssignale bei 45° Rotation des Magneten über dem Sensor (A,1).



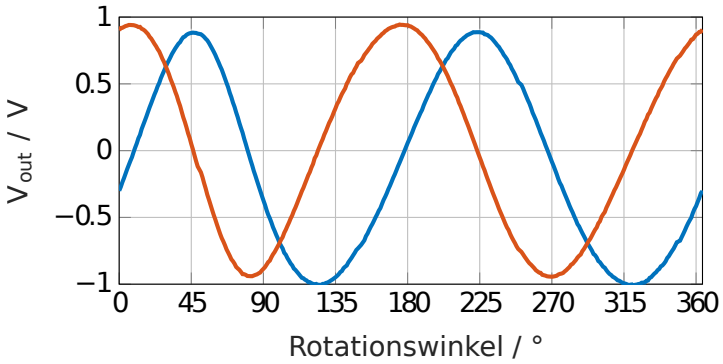
(f) SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5) in Abhängigkeit der Rotation des Magneten über dem Sensor (A,8).

Abbildung 5.13: Messungen mit dem NVE-AAT001-10E Sensor-Array und einem Kugelmagneten (d = 26 mm) im Abstand von 19 mm bei drei verschiedenen Positionen über dem Sensor-Array.

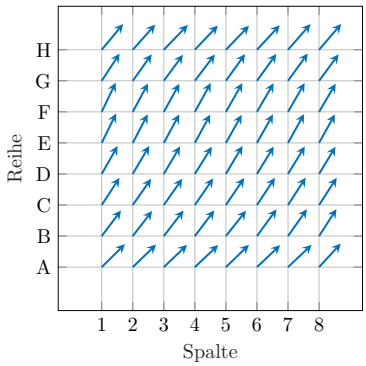
Gegenüberstellung der Sensor-Array Messungen In der Abbildung 5.14 erfolgen die Messungen mit dem großen Kugelmagneten über dem Zentrum des jeweiligen Sensor-Arrays. Die Vektordarstellung der Sensoren des TMR-Sensor-Arrays zeigen alle in dieselbe Richtung in einem Winkel von ca. 45°, die des AMR-Sensor-Arrays sind gestreuter. Werden die Kurvenverläufe gegenübergestellt, so ist die Amplitude des AMR-Sensor-Arrays größer, mit einem Spitze-Spitze Wert von 1,6 V. Die des TMR-Arrays hat einen Wert von 0,6 V. Dabei ist jedoch zu beachten, dass die Auflösung des TMR-Arrays größer ist, wodurch die Periodendauer von 360° des Sensor-Arrays ersichtlich wird. Die Periode des AMR-Arrays wiederholt sich bei 180°.



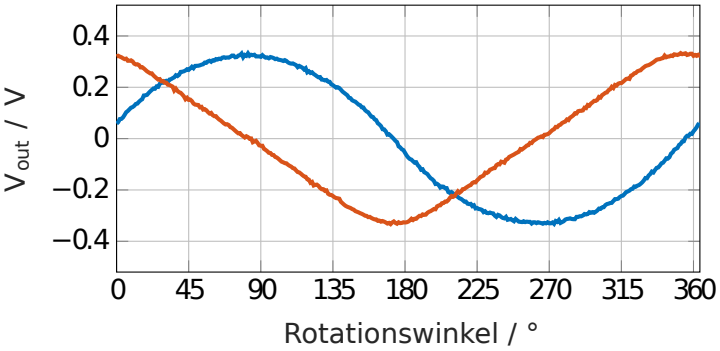
(a) Vektordarstellung der Ausgangssignale bei 45° Rotation des Magneten über der Mitte des Arrays.



(b) SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5) in Abhängigkeit der Rotation des Magneten über der Mitte des Arrays.



(c) Vektordarstellung der Ausgangssignale bei 45° Rotation des Magneten über der Mitte des Arrays.



(d) SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5) in Abhängigkeit der Rotation des Magneten über der Mitte des Arrays.

Abbildung 5.14: Gegenüberstellung der Messungen mit dem KMZ-60 Sensor-Array und dem NVE-AAT001-10E Sensor-Array und einem Kugelmagneten (d = 26 mm) im Abstand von 19 mm mittig über den Sensor-Arrays.

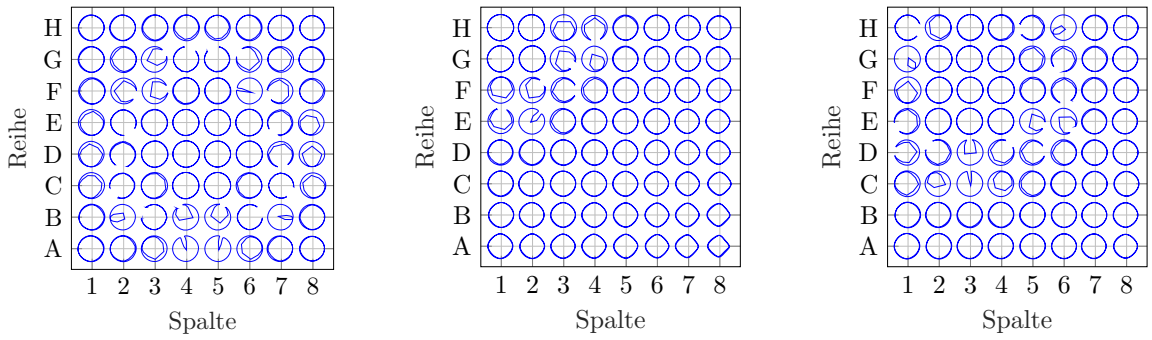
5.2.3 Kreisdarstellung der Messergebnisse beider Sensor-Arrays

Durch die SIN- und COS-Signale eines Sensors lassen sich durch beide Anteile ein Einheitskreis darstellen, falls beide Anteile gleich groß sind. Ist nur ein Anteil kleiner oder größer kommt es zu Verzerrungen und Verformungen des Kreises. In den nachfolgenden dargestellten Abbildungen sind für alle Sensoren diese Kreise erstellt worden.

In der Abbildung 5.15 wurde über den Punkten (H,1), (F,3) und der Mitte des Sensor-Arrays gemessen. Die Punkte sind in dieser Abbildung gut ersichtlich, zu erkennen an den Kreisdarstellungen. Dabei fällt auf, dass die Kreise groß und in den gesättigten Bereichen einheitlich sind.

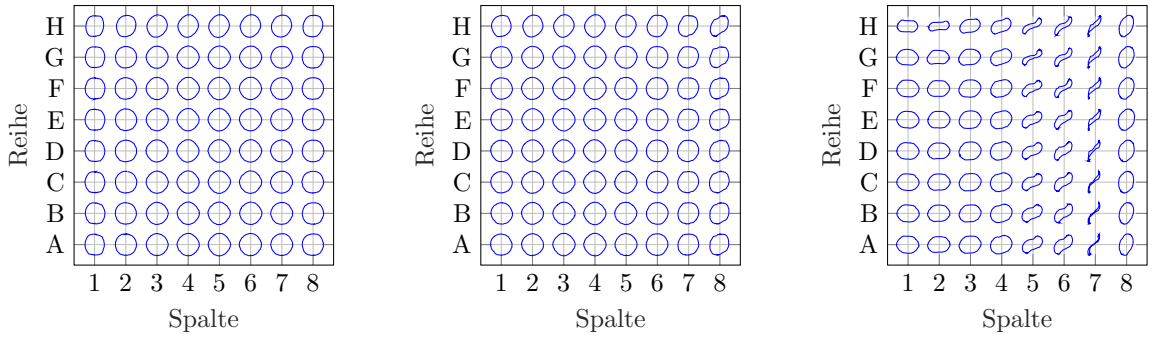
In der Abbildung 5.16, wurde über den Punkten (C,3), (A,1) und dem Zentrum des Sensor-Arrays gemessen. Bei der Messung im Mittelpunkt ergeben sich einheitliche Kreise über allen Sensoren. Die Messung um den Punkt (C,3) ergeben, dass am äußeren Rand des Sensor-Arrays sich die Form der Kreise ändert, das ist noch deutlicher über dem Messpunkt (A,1) zu erkennen. Die Sensoren sind zu weit entfernt und verzerren.

In der Abbildung E.1 sind die Kreisdarstellungen der Sensor-Arrays gegenübergestellt. Gemessen wurde mit dem großen Kugelmagneten, dem kleinen Kugelmagneten und dem quaderförmigen Magneten. Bei dem KMZ-Sensor-Array, ist erkennbar, dass mit Abnahme der Größe der Magneten, die Sensoren den Sättigungsbereich und die deutlichen Kreisdarstellungen verlassen. Das NVE-Sensor-Array weist denselben Effekt auf, jedoch ist die Abnahme nur in den äußeren Bereichen vorhanden.



(a) Mitte. (b) H,1 (c) F,3

Abbildung 5.15: Messungen mit dem KMZ-60 Sensor-Array und einem Kugelmagneten (d = 26 mm) im Abstand von 19 mm bei drei verschiedenen Positionen über dem Sensor-Array.

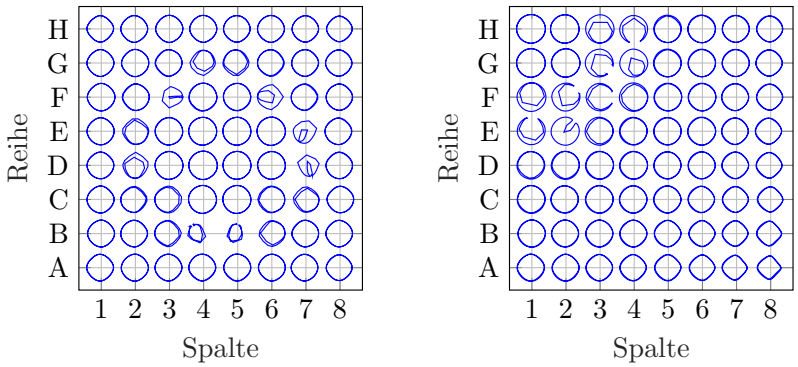


(a) Mitte. (b) C,3 (c) A,1

Abbildung 5.16: Messungen mit dem NVE-Sensor-Array und einem Kugelmagneten (d = 26 mm) im Abstand von 19 mm bei drei verschiedenen Positionen über dem Sensor-Array.

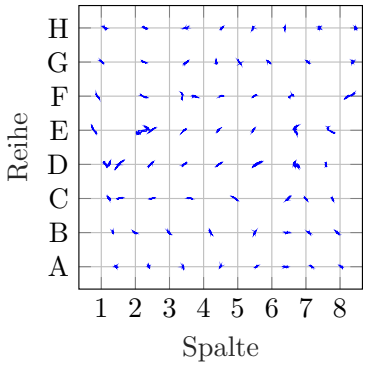
5.2.4 Störfeldmessungen

Das Störfeld wird mit zwei unterschiedlichen Halbachringen erzeugt, das eine hat ein homogenes Magnetfeld von 4 $\frac{\text{kA}}{\text{m}}$, das andere 6 $\frac{\text{kA}}{\text{m}}$. Der stärkere Ring wurde um das TMR-Sensor-Array gelegt, das schwächere um das AMR-Sensor-Array. Dies ist aus Gründen der Geometrie der jeweiligen Arrays erfolgt. Die Abbildung 5.17 ist mit dem kleinen Kugelmagneten gemessen worden. Gegenübergestellt sind die Messpunkte (A,1) und der Mittelpunkt des KMZ-Sensor-Arrays. Deutlich zu erkennen ist, dass durch das Störfeld die Kreise verschwinden. In der Abbildung 5.18 ist mit dem kleinen Kugelmagneten gemessen worden. Die gemessenen Punkte sind zum einen das Zentrum des TMR-Arrays und zum anderen der Punkt (A,1). Es ist ersichtlich, dass trotz Störfeld Messergebnisse vorhanden sind. Bei der Messung um den Mittelpunkt sind die Spalten 1-3 und die Spalte 8 vom Störfeld beeinflusst. Bei der Messung um den Punkt (A,1) sind die Spalten 5-8 beeinträchtigt sowie die oberen Zeilen.

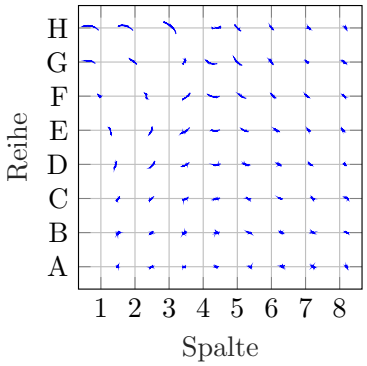


(a) Mitte ohne stoerfeld.

(b) (A,1) ohne stoerfeld

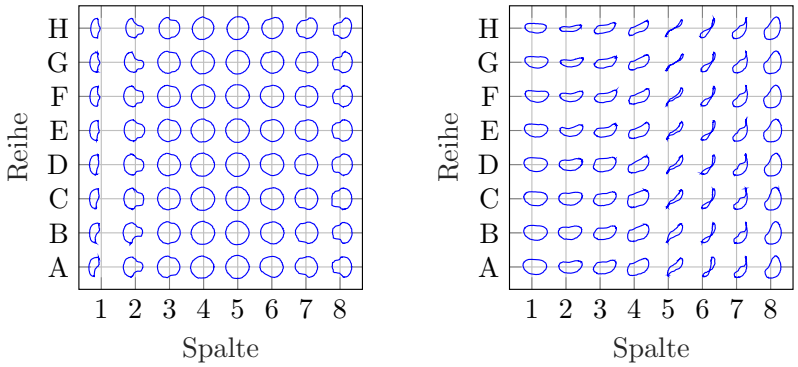


(c) Mitte mit stoerfeld.



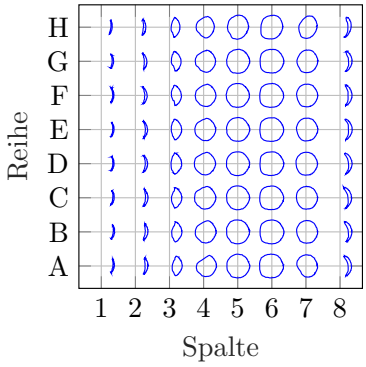
(d) (A,1) mit stoerfeld

Abbildung 5.17: Messungen mit dem KMZ-60 Sensor-Array und einen kleinen Kugelmagneten mittig über dem Array und an der Position (A,1) ohne Störfeld und mit Störfeld.

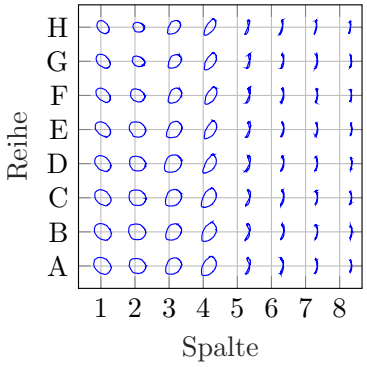


(a) Mitte ohne Störfeld.

(b) (A,1) ohne Störfeld



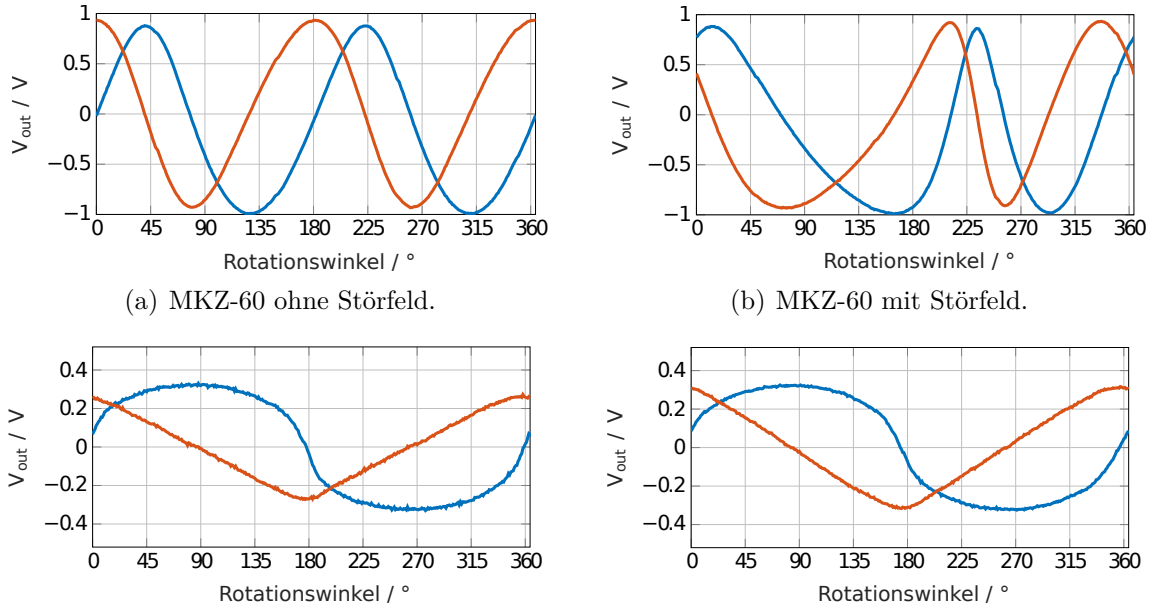
(c) Mitte mit Störfeld.



(d) (A,1) mit Störfeld

Abbildung 5.18: Messungen mit dem NVE-Sensor-Array und einer kleinen Kugel mittig über dem Array und an der Position (A,1) ohne und mit Störfeld.

Gegenüberstellung der Kurvenverläufe der beiden Sensor-Arrays In der Abbildung 5.19 sind die Kurvenverläufe beider Sensoren gegenübergestellt. Auf der linken Seite ohne Störfeld auf der rechten mit Störfeld. Gemessen wurde über den Zentren der beiden Sensor-Arrays mit dem quaderförmigen Magneten. Bei der Betrachtung der oberen Verläufe des AMR-Senor-Arrays zeigt sich, dass es durch das Störfeld es zu einer Verschiebung des Ausgangssignals kommt. Eine Periodizität ab 180° lässt sich auch nicht erkennen. Die Messungen unten am TMR-Sensor-Array verformt sich nicht. Die Amplitude des SIN-Ausgangssignal wird dabei minimal verstärkt und das Rauschen beider Signale nimmt ab.



(a) MKZ-60 ohne Störfeld.

(b) MKZ-60 mit Störfeld.

(c) NVE ohne Störfeld.

(d) NVE mit Störfeld.

Abbildung 5.19: SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5). Es sind jeweils die Signale ohne (links) und mit (rechts) Stöerfeld für das KMZ-60 und NVE-Array dargestellt. Der Gebermagnet ist ein Würfelmagnet mit einer Kantenlänge von 10 mm und einem Abstand von 19 mm zum jeweiligen Sensor-Array.

5.3 Résumé der Messungen

Bei der Betrachtung der Messergebnisse, der Widerstandsmessungen, im ersten Teil des Kapitels fällt auf, dass es eine Häufung von relativ hohen Widerstandswerten in den Zeilen A-C besteht. Die Häufung der Widerstandswerte wird nochmals in den Histogrammen 5.3 ersichtlich. Die Ursache der Häufung könnte an dem Bestückungsverfahren liegen, da die Sensoren direkt aus der Verpackung nacheinander auf die Platine gesetzt wurden.

Eine Besonderheit fällt zwischen den Abbildungen 5.1(c) und 5.2(c) auf. Die erste Abbildung stellt die Differenz zwischen den jeweiligen Halbbrücken dar, die Zweite die Differenz zwischen den Widerständen der Versorgungsspannung und des Massenpotenzials. Dabei sind die Symmetrien deutlich zwischen ΔR COS und Δ GND zu erkennen. Dies ist vermutlich auf eine interne Verschaltung der Sensorelemente zurückzuführen.

In den Abbildungen 5.4 und 5.5 sind exemplarisch die Ausgangssignalspannungen der COS-Leitung 1 und SIN-Leitung 8 dargestellt. Deutlich zu erkennen ist, dass der Maximalwert der Ausgangsspannung von 1,66 V der Hälfte der Versorgungsspannung entspricht. Klar Unterscheidbar sind die Pausenintervalle zwischen den acht Schaltvorgängen. Damit ist bewiesen, dass es zu keiner Beeinflussung der anderen Versorgungsleitungen kommen kann. Beim Betrachten der Abbildungen 5.9 bis 5.10 ist ersichtlich, dass die Beschaltung und Pixelzuweisung sich so wie in der Software programmiert verhält.

Die wesentlichen Unterschiede zwischen den zwei Sensor-Array Typen, sind deutlich in den Abbildungen 5.14, E.1 und 5.19 zu erkennen. In Abbildung 5.14 ist eindeutig ersichtlich, dass die Vektordarstellungen des AMR-Arrays nicht so ausgerichtet sind wie die des TMR-Arrays. Das kann zum einen an der Sättigung und zum anderen an der benötigten Feldstärke des Gebermagneten liegen. Der Kurvenverlauf des AMR-Arrays weist eine Doppelperiodizität bei 180° auf. Ersichtlich ist auch, dass das AMR-Array über Filterschaltungen sowie eine höhere Betriebsspannung von 5 V verfügt.

Abbildung E.1 stellt die Kreisdarstellung der beiden Sensor-Array-Typen gegenüber. Die doppelte Periodizität ist auch hier zu erkennen, die doppelten Kreise bzw. die Muster entstehen durch den zweiten Verlauf ab 180°. Anhand dessen, dass das AMR-Sensor-Array Filter an den Ausgangssignalen hat, ähneln die Kreisbahnen mehr einem Kreisverlauf. Der größere Durchmesser resultiert aus der höheren Betriebs- und Ausgangssignalspannung.

Der direkte Vergleich zwischen den Arrays untere Störfeldeinfluss ist dargestellt in Abbildung 5.19. Die Störfelder haben unterschiedliche Feldstärken. Das des AMR-Sensor-Array beträgt 4 kA/m und des TMR-Sensor-Array 6 kA/m. Der Kurvenverlauf des AMR-Sensor-Arrays ist verzerrt und der periodische Verlauf ist nicht mehr symmetrisch. Dahingegen hat das Störfeld kaum Einfluss auf das TMR-Sensor-Array. Im Gegensatz dazu ist das Sinussignal leicht verstärkt und das Rauschen beider Signale hat abgenommen, möglicherweise dient das Störfeld in diesem Fall auch als Stützfeld.

6 Schlussfolgerungen

In diesem Kapitel werden die Ergebnisse dieser Arbeit zusammengefasst und die erreichten Ziele dargestellt. Darauf folgend wird beschrieben, in welchen Aspekten diese Arbeit fortgeführt werden kann. Zum Schluss folgen Anmerkungen und Anregungen zur Verbesserung beim Fortführen dieser Arbeit.

6.1 Zusammenfassung

Der wesentliche Teil dieser Arbeit war die Erstellung eines funktionsfähigen 8x8 TMR-Sensor-Arrays, die Modifikationen zur Ansteuerung des Sensor-Arrays sowie die Messerprobung an einem Robotermessplatz. Das Sensor-Array wird zur zweidimensionalen Magnetfelderfassung eingesetzt.

Zu Anfang wurden die Grundlagen der MR-Prinzipien sowie deren Funktionsweisen erläutert. Dabei sind die unterschiedlichen MR-Effekte, explizit der AMR- und der TMR-Effekt, erläutert. Darüber hinaus erfolgte eine ausgiebige Darstellung des genutzten Sensors, der zum Erstellen des Arrays genutzt wurde.

Nachdem das Grundschema des Sensor-Arrays dargestellt und erarbeitet wurde, erfolgte der schematische Entwurf der Platine in einem kurzen Zeitraum. Das Sensor-Array wurde nach den Vorgaben erstellt und dementsprechend bestückt. Zum Betreiben der Hardware mussten einige Modifikationen am Mikrocontroller vorgenommen werden.

Für die Hardware wurde ein Steuerungsprogramm implementiert, dass das Auslesen der Zeilen und Spalten ermöglicht. Der Robotermessplatz wurde nach einigen Modifikationen in Betrieb genommen. Für die Messungen des Sensor-Arrays, wurde ein Programm erstellt, dass es ermöglicht eine, 360° Aufnahme über mehrere Messpositionenen zu erstellen und abzuspeichern.

Die TMR-Sensorelemente wurden ausführlich betrachtet und ausgewertet. Zur Inbetriebnahme des Arrays war als Erstes die Zuweisung der Sensorelemente grafisch darzustellen und einwandfrei zuzuordnen. Die Ausgangssignalspannungen wurden gemessen und haben wie erwartet den halben Wert der Betriebsspannung. Anschließend wurden ausführliche Messungen mit dem AMR- und TMR-Sensor-Array durchgeführt. Dies erfolgte mit vier verschiedenen Magneten. Darüber hinaus wurden Messungen mit Störeinflüssen, anhand eines Halbachrings, der ein homogenes Magnetfeld erzeugt, gemessen.

6.2 Herausforderung

Der Ausgangspunkt dieser Arbeit sah wie folgt aus: Eine erste Version eines TMR-Sensor-Arrays wurde im Projekt erstellt. Dabei kam es zu Fehlern mit der Beschaltung der Sensoren in einer Array-Struktur. Der grundlegende Fehler war, dass aufgrund der internen Beschaltung der Sensorelemente Kurzschlüsse mit der Masse auftraten. Das hatte zur Folge, dass das Sensor-Array so nicht betrieben werden konnte. Es wurde dann entschieden, die Beschaltung der Sensoren, spaltenweise mit je zwei separaten Spannungsversorgungen zu betreiben. Das ergab in Summe 48 Leitungen für das Betreiben des Sensor-Arrays, dazu kamen weitere 16 für die Datenleitungen.

Das Erstellen der Platine erwies sich als schwierig. Eine der Herausforderung war die manuelle Bestückung der Platine. Aufgrund des Aufbaus der Sensoren, die über einen Massekontakt auf fast der gesamten Fläche der Unterseite verfügen, kam es zu Kurzschlüssen beim Bestücken. Eine andere Schwierigkeit ergab sich durch das Anordnen und Platzieren der Sensoren. Einige der Sensoren hatten ein unterschiedliches Höhenprofil. Wiederum andere Sensoren waren nach dem SMD-Löten verkippt. Erst im vierten Anlauf ergab sich ein funktionsfähiges und fehlerfreies Sensor-Array.

Die Umsetzung zur Beschaltung und Steuerung der Platine war eine Herausforderung angesichts der Menge an benötigten GPIOs. Die Umsetzung war nur möglich, in dem das Boosterpack X11 des Mikrocontrollers genutzt wurde. Dazu wurden mittels Datenblatt, versucht geeignete GPIOs auszuwählen. Dies war aber aufgrund der hohen Anzahl nicht ganz möglich. Während des Testbetriebs fiel auf, dass es zu Rückkopplungen und Kurzschlüssen auf der Platine kam. Bedingt war dies durch die $0\,\Omega$ Widerstands-Brücken auf dem Mikrocontroller, die zuvor beseitigt werden mussten. Es konnten nicht alle Einflüsse des Mikrocontrollers entfernt werden, da sonst die Funktionsfähigkeit des Debuggmodus nicht zu gewährleisten wäre. Das Entfernen der Netzwerkdose war zwingend erforderlich, da die Platine sonst nicht montierbar gewesen wäre.

Das zur Verfügung gestellte Programm wurde zur Übersichtlichkeit zunächst umstrukturiert. Es musste aufgrund der vielen GPIO umgeschrieben werden. Einige Grundfunktionen konnten dabei erhalten bleiben. Zu Komplikationen kam es mit den ADCs und dem Senden der Daten. Die Signale weisen einen minimalen Offset auf, dies ist auf die Widerstandselemente der Sensoren sowie vermutlich des nicht vorhandenen gemeinsamen Groundpotenzials zurückzuführen. Es wurde deutlich beim Zuweisen der Analogeingänge der ADC, bei abwechselnden Folgen von SIN- und COS-Signalen in der Implementierung. Behoben wurde das, in dem jeweils die SIN-Signale auf einen ADC gelegt wurden und die COS-Signale auf den anderen ADC. Das Senden der Daten war in dem Fall erschwert, da die Signale zeilenweise in umgekehrter Reihenfolge erfolgen mussten, um am PC richtig angezeigt zu werden. Diese Herausforderungen haben ein Großteil der Zeit in Anspruch genommen.

Mit dem zur Verfügung gestellten Messplatz sollten die notwendigen Messungen der

voreingestellten Funktionen aufgenommen und dargestellt werden. Jedoch war das nicht möglich, da das Sensor-Array die Aufnahmen mittels einer 360° Drehung eines Gebermagneten durchführen soll. Der Messplatz war durch den Betrieb eines Endschalters nicht in der Lage, diese Messung durchzuführen. Bekannt war das am Anfang der Arbeit nicht. Die Lösung erfolgte durch ein Auskommentieren einer Abfrage des Endschalters. Da aber niemanden bekannt war, wie das Programm tatsächlich funktioniert, war es nötig, sich mit dem Messplatz auseinanderzusetzen. Daraus resultierte, dass sich die Drehrichtung des Drehtellers ändern musste. Die Folge war, dass alle voreingestellten Messabläufe und deren Darstellungen nicht genutzt werden konnten. Ein Messprogramm zur Aufnahme der Daten über mehrere Positionen wurde daraufhin erstellt. Bei den Testmessungen ist aufgefallen, dass zwei unterschiedliche Initialisierungsfahrten existieren, die beide eine andere Nullposition aufweisen. Bei den Messungen und dem Wechseln der Magnete wurde ersichtlich, dass es nicht immer möglich ist, diese gleich auszurichten. Das kann zu einem verschobenen Messanfang führen, je nach Art der Verkippung und Fehlstellung der Anfangsposition der Magnete. Bei den ersten Messungen über das Array ist aufgefallen, dass es zu Welligkeiten im Kurvenverlauf der Sensoren kommt. Dies hat vermutlich damit zu tun, dass entweder die Sättigung der betroffenen Sensoren nicht erreicht ist und somit ein Hysterese Effekt eintritt oder durch anderweitige Störeinflüsse vorhanden sind.

Bei den Messungen zur Inbetriebnahme des Sensor-Arrays wurden die Sensorelemente und deren Widerstandswerte genauer betrachtet. Dabei wurde ersichtlich, dass vermutlich eine interne Verkopplung der Sensorelemente über die Halbbrücken besteht. Darüber hinaus war zu sehen, dass es zu einer Häufung von hohen Widerstandswerten in der unteren rechten Ecke des Arrays kam. Höchstwahrscheinlich hängt es damit zusammen, dass beim Bestücken der Platine die Sensoren nicht durchmischt wurden. So sind Sensoren aus derselben Produktionsreihe nebeneinandergesetzt worden.

Bei den Versuchsreihen zwischen dem TMR- und AMR-Sensor-Array konnten nicht genaue Vergleichsmessungen durchgeführt werden. Es fehlte an geeigneten Magneten, da das AMR-Sensor-Array ein stärkeres Magnetfeld benötigt als das TMR-Sensor-Array. Um vergleichbare Messungen zu erstellen, musste deutlich näher an das Sensor-Array herangefahren werden. Darüber hinaus war das Anfahren der Messpositionen des AMR-Arrays nicht exakt möglich, da es zu Kommunikationsunterbrechungen mit der Ansteuerung des Messplatzes kam. Dieser hat im Betrieb mit dem Array erhöhte Fehlermeldungen. Bei den Messungen mit dem Störfeld war die Größe des AMR-Arrays ein Hindernis. Die äußeren Messpunkte konnten nicht angesteuert werden, da sonst der Gebermagnet vom Halbachring durch seine Anziehungskraft festgehalten wurde. Aus diesem Grund musste beim AMR-Sensor-Array mit einem schwächeren Magneten gemessen werden.

Abschließende Zusammenfassung

Abschließend lässt sich zusammenfassen, dass das erstellte TMR-Sensor-Array den Erwartungen des ISAR-Projektes entspricht. Die Messwerte und die Funktionsprüfung ha-

ben bewiesen, dass die Funktionalität gegeben ist. Im Vergleich zum AMR-Sensor-Array hat das TMR-Sensor-Array weder Filterschaltungen noch Spannungsstabilisierung, jedoch ist die Funktionalität bei Störfeldeinfluss immer noch gegeben.

6.3 Ausblick

In Bezug auf das erstellte Sensor-Array, könnte erwägt werden, Verstärkerschaltungen sowie Filter für die Ausgangssignalspannungen einzubauen. Es sollten weitere Messungen erfolgen, beispielsweise die Messung der Hysterese, Verkippung des Gebermagneten oder des Sensor-Arrays. Die Halbachringe für das homogene Störfeld sollten im Durchmesser und in der Höhe vergrößert werden. Damit könnte das Problem der Anziehung des Gebermagneten und des damit einhergehenden Blockierens des Schrittmotors behoben werden. Durch Erhöhung des Ringes sollte die Einwirkung des Störfelds mittig auf das Sensor-Array und den Magneten wirken. Für weitere Messungen ist zu empfehlen, eine neue Universalhalterung für die Magneten zu erstellen, die einfach auszutauschen ist, ggf. über eine Steckverbindung. In diesem Zuge könnte damit auch die Ausrichtung des Magneten präzisiert werden, sodass es zu keiner Fehlstellung kommt. Ein weiteres Array, basierend auf dem TMR-Effekt mit den Sensoren der Firma TDK könnten neue Informationen liefern. Außerdem könnte ein ADC verbaut werden der alle Signale gleichzeitig auflösen kann.

Der Messplatz sollte überarbeitet werden, dabei sind die Verbindungen der Leitungen zu überprüfen und gegebenenfalls auszutauschen. Die Programmierung sollte überdacht und an die 360° angepasst werden. Da Langzeitmessungen von ca. drei Tagen nun möglich sind, sollte ein Konzept zur Verhinderung von Stromausfällen erdacht werden.

Das Programm zur Steuerung des Sensor-Arrays kann noch weiter optimiert werden. Auch der Gedanke zum Schalten der Halbbrücken sollte überdacht werden. Möglicherweise kann die Schaltung nur mit zwei unterschiedlichen Spannungsversorgungen und Masseleitungen betrieben werden.

Literatur

[1] *AAT00x Ultralow Power TMR Angle Sensors*. NVE Corporation. Version: 02.0. Apr. 2017.

[2] Viktor Airich. „Charakterisierung magnetoresistiver Sensor-Arrays mittels eines automatisierten Messsystem“. Bachelorarbeit. 2018.

[3] Energia. *Guide to the TM4C129 Connected LaunchPad*. URL: `http://energia.nu/pin-maps/guide_tm4c129launchpad/`.

[4] TEXAS INSTRUMENTS. *Tiva C Series TM4C1294 Connected LaunchPad Evaluation Kit*. `http://www.ti.com/lit/ug/spmu365c/spmu365c.pdf`; Zugriffsdatum: 19.06.2018. October 2016.

[5] TEXAS INSTRUMENTS. *Tiva TM4C1294NCPDT Microcontroller*. `http://www.ti.com/lit/ds/spms433b/spms433b.pdf`; Zugriffsdatum: 19.06.2018. Juni 2014.

[6] Michel Julliere. „Tunneling between ferromagnetic films“. In: *Physics letters A* 54.3 (1975), S. 225–226.

[7] *KMZ60*. NXP B.V 2014. Rev. 2-7 February 2014. Feb. 2014.

[8] Uwe Loreit. *Magnetoresistive Winkelsensorenfür extreme Einsatzbedingungen*. Haus der Technik. `http://www.mr-sensor.de/Vortraege/99LK28Essen.pdf`; Zugriffsdatum: 19.06.2018. Februar 1999.

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

[10] Konrad Reif. „Sensoren im Kraftfahrzeug“. In: *Sensoren im Kraftfahrzeug*. Springer, 2010, S. 10–33.

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

[12] ROLF SLATTER. „Tunnelmagnetoresistive Sensoren für die Antriebstechnik“. In: *ELEKTRONIKPRAXIS* 14 (2017), S. 28–30.

[13] TDK. *Principle of angle sensor using TMR element*. URL: `https://product.tdk.com/info/en/products/sensor/angle/tmr_angle/technote/tpo/index.html`.

[14] Thomas Tille. *Automobil-Sensorik - Ausgewählte Sensorprinzipien und deren automobile Anwendung*. Berlin Heidelberg New York: Springer-Verlag, 2016. ISBN: 978-3-662-48944-4.

[15] Wikipedia. *Elektronenspin*. URL: `https://de.wikipedia.org/wiki/Elektronenspin`.

[16] Wikipedia. *Fermi-Kante*. URL: `https://de.wikipedia.org/wiki/Fermi-Dirac-Statistik`.

Anhang

A PAP

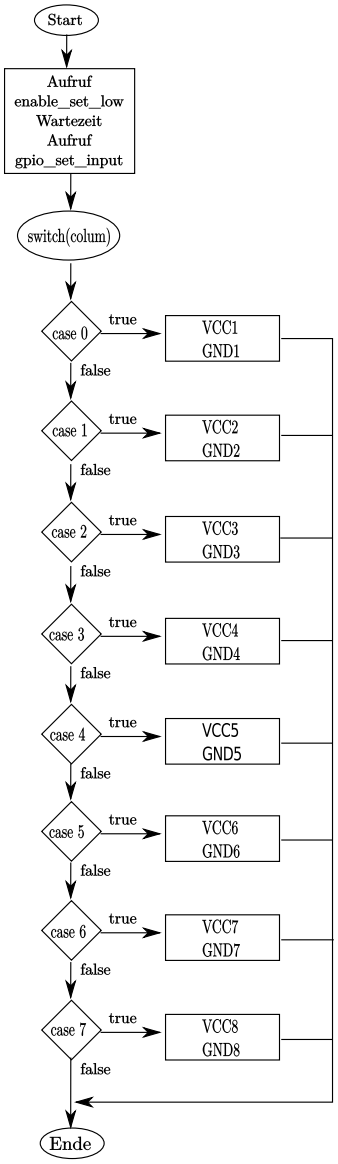


Abbildung A.1: Programmablaufplan: enable_set_high

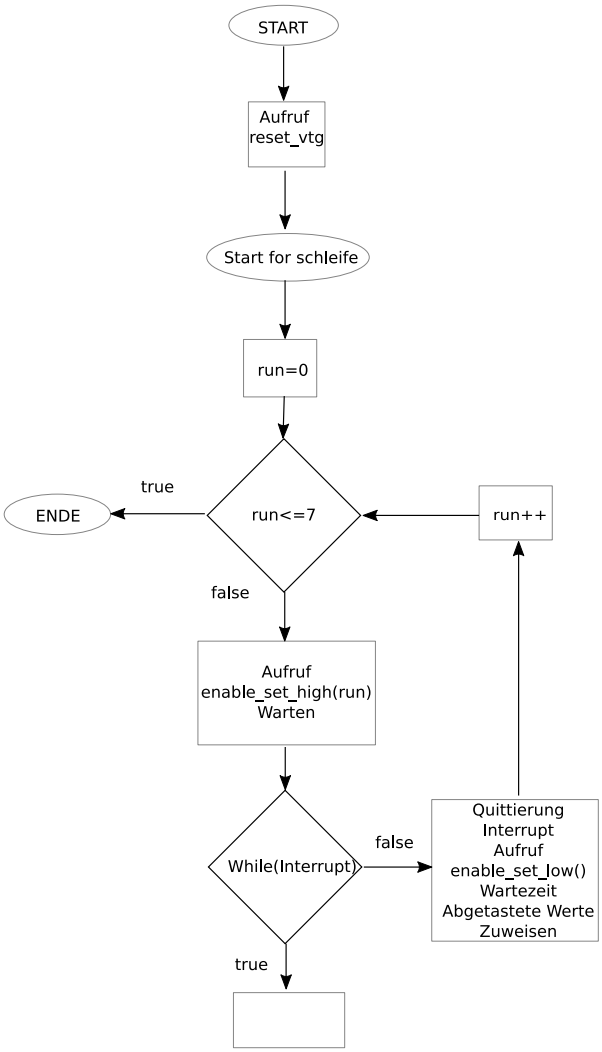


Abbildung A.2: Programmablaufplan: Get_Values

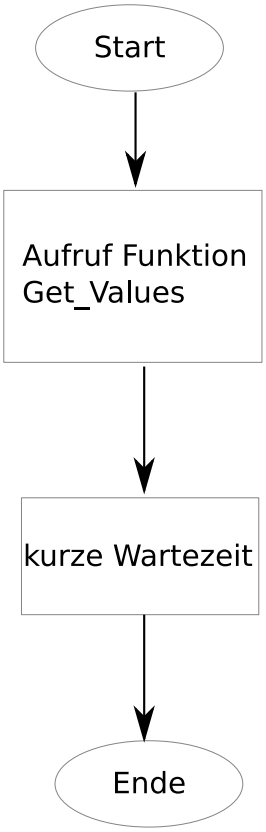


Abbildung A.3: Programmablaufplan: Init_Systick_Interrupt_Handler

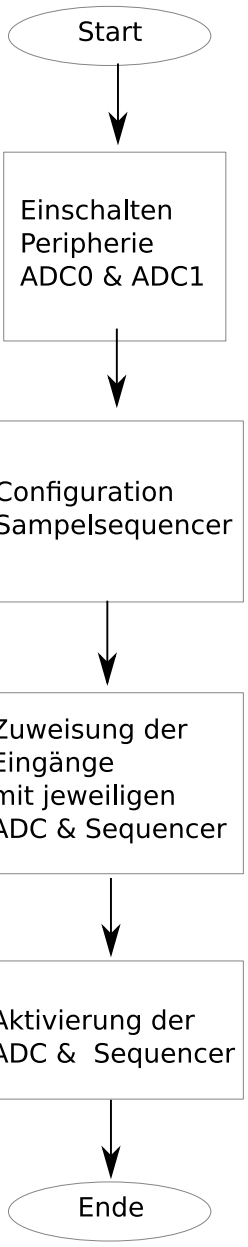


Abbildung A.4: Programmablaufplan: Init_ADC

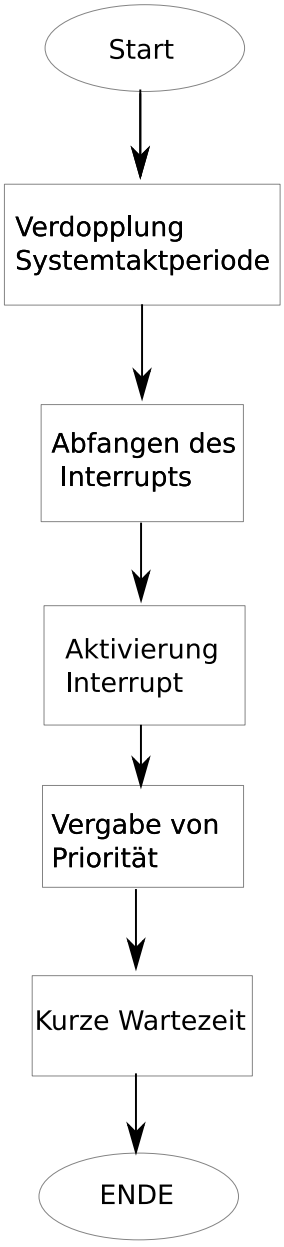


Abbildung A.5: Programmablaufplan: Init_Systick_Interrupt

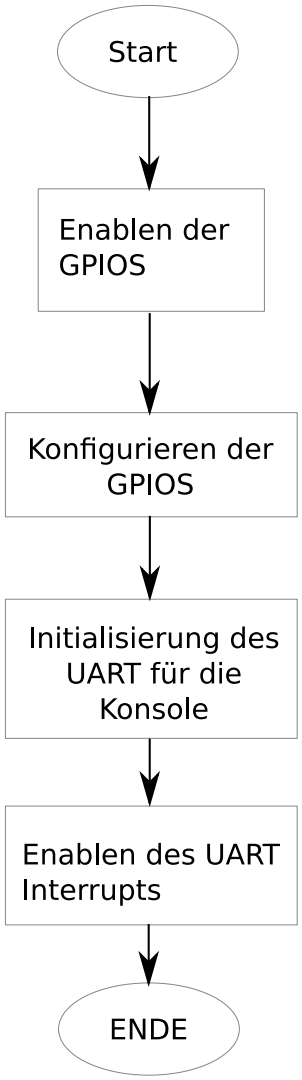


Abbildung A.6: Programmablaufplan: INIT_UART

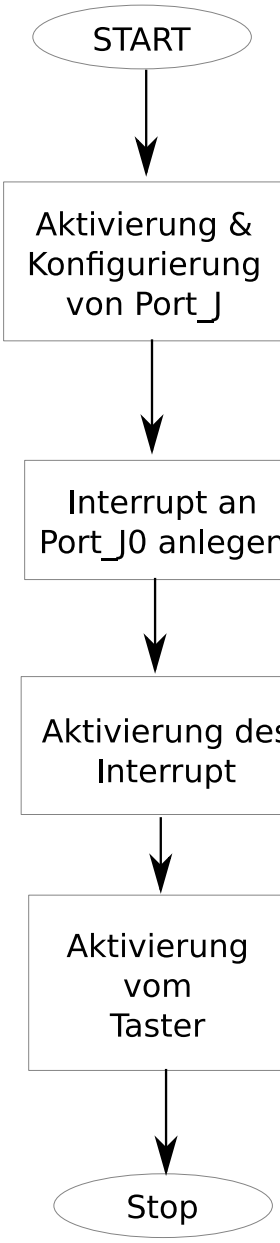


Abbildung A.7: Programmablaufplan: Interrupt_Port_J

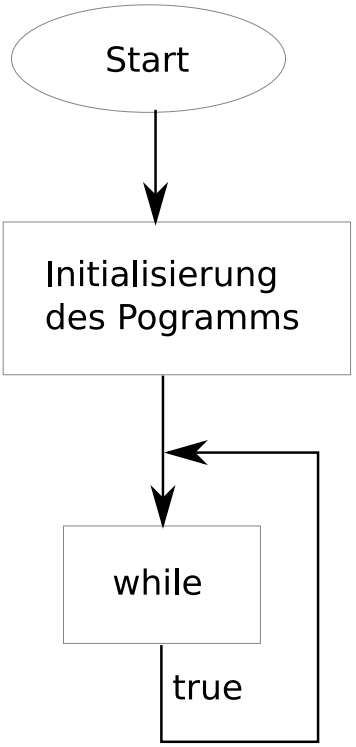


Abbildung A.8: Programmablaufplan: Main

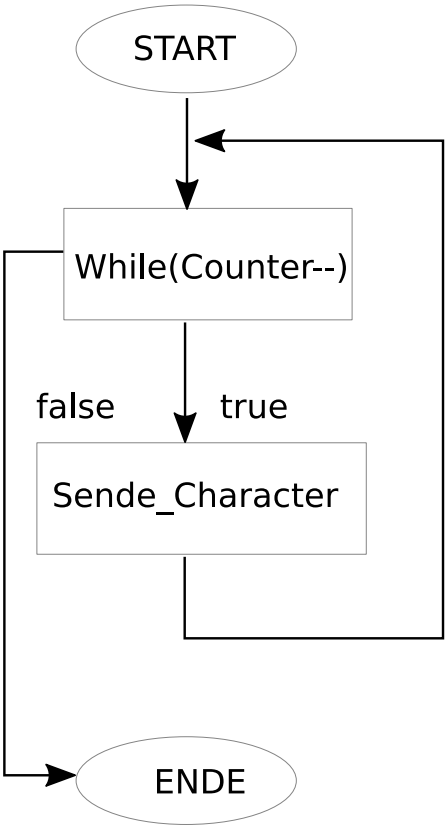


Abbildung A.9: Programmablaufplan: UART_SEND

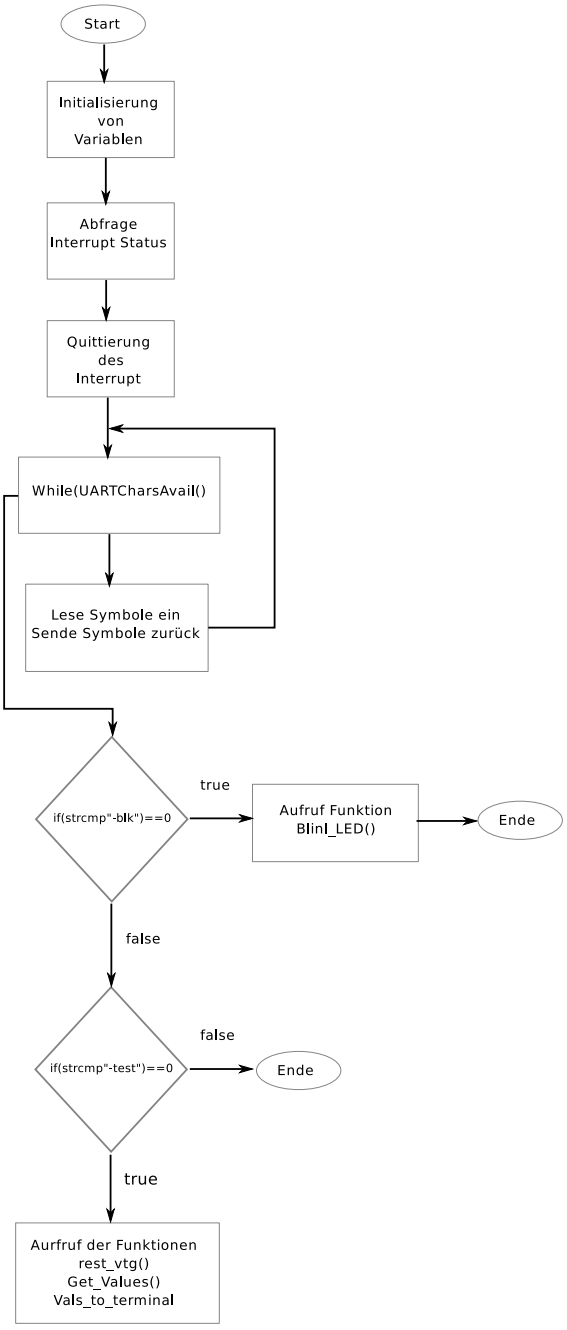


Abbildung A.10: Programmablaufplan: UARTIntHandler

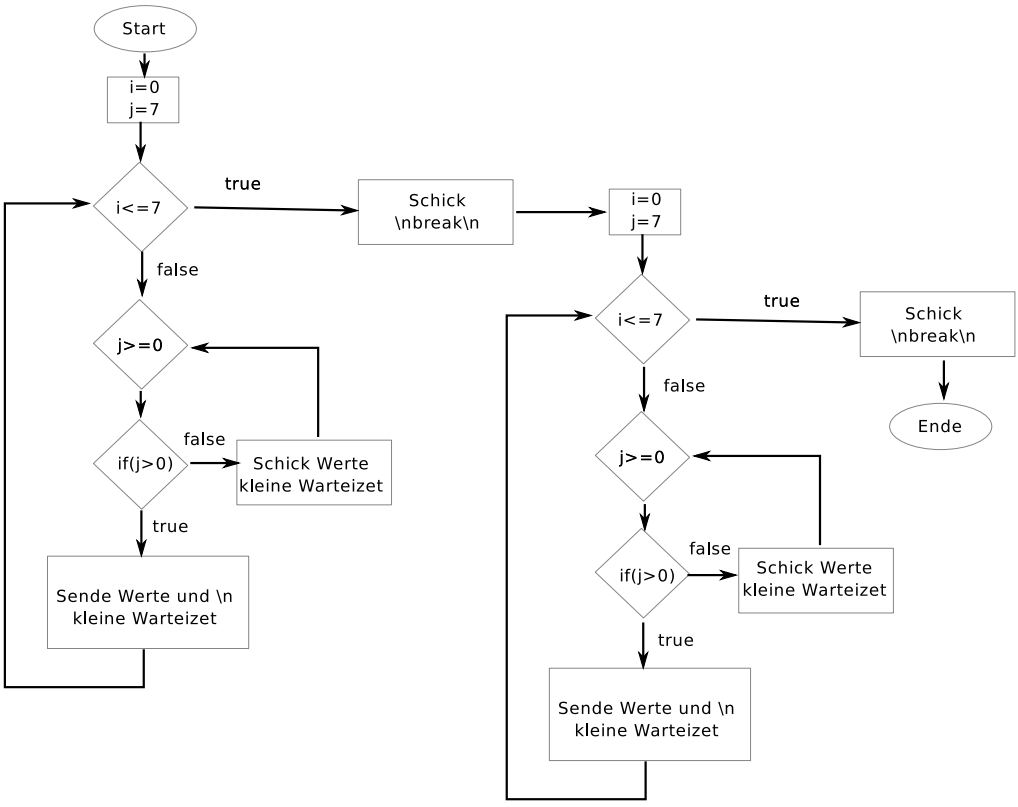


Abbildung A.11: Programmablaufplan: vals_to_terminal

B Technische Abbildungen

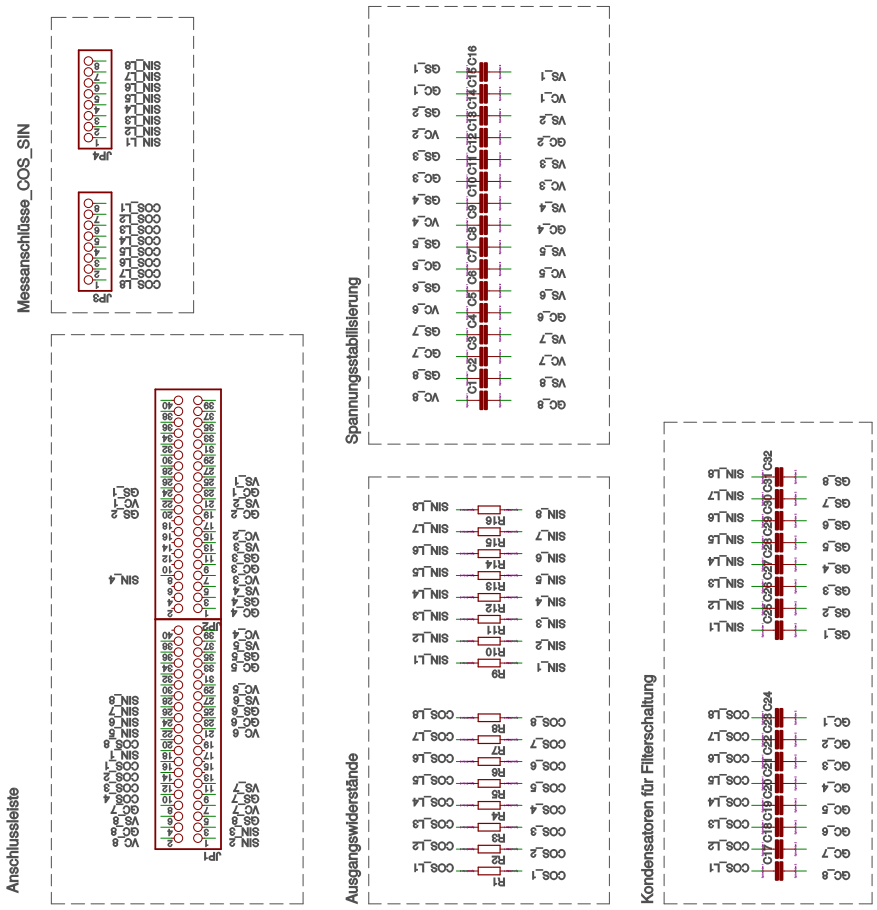


Abbildung B.1: Schematische Darstellung der Anschlüsse.

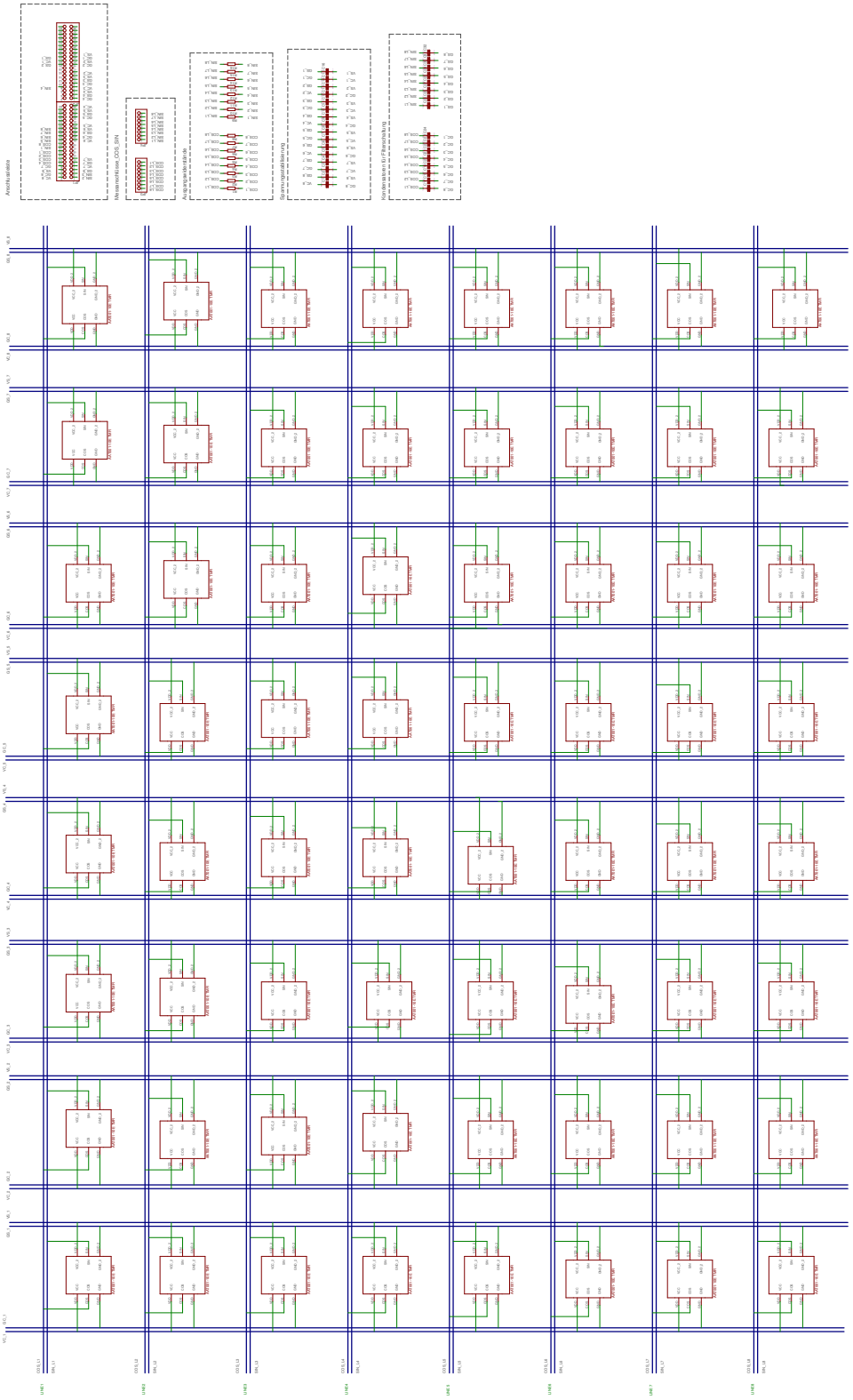


Abbildung B.2: Abbildung über die gesamte schematische Darstellung in EAGLE.

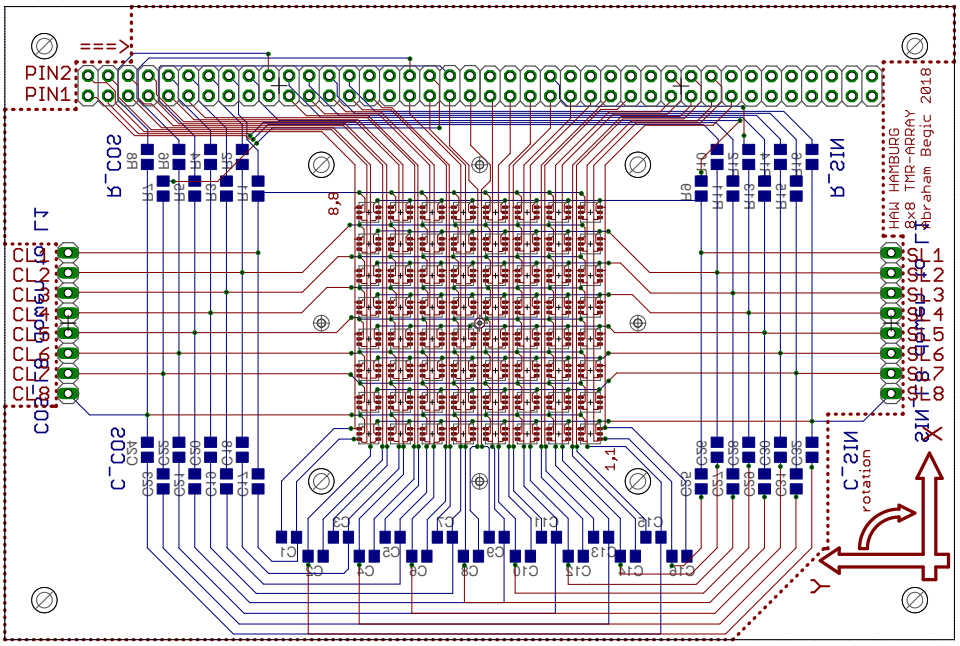


Abbildung B.3: Vollständiges Platinen-Layout.

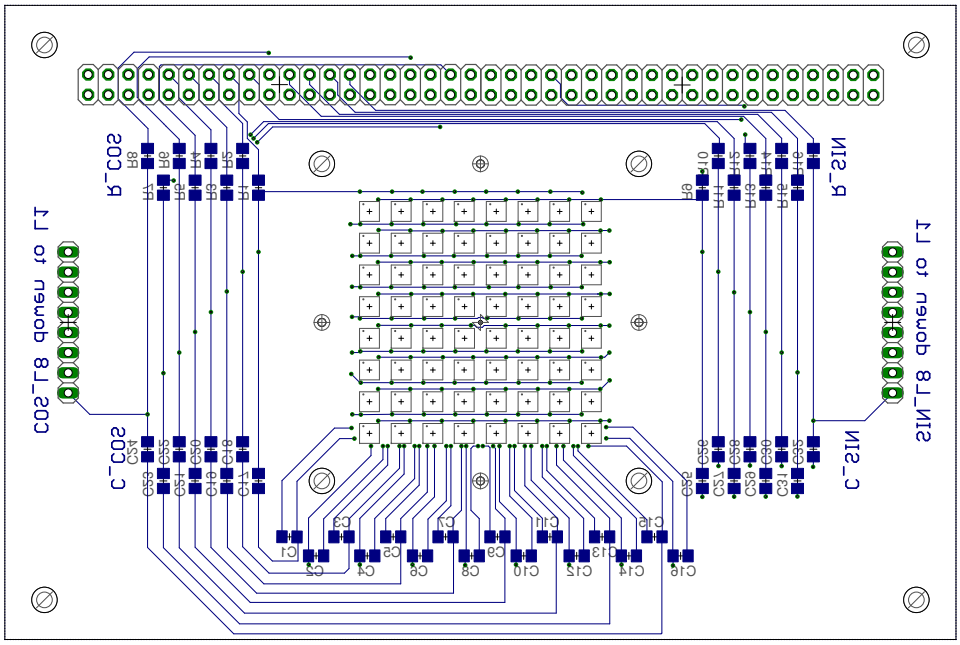


Abbildung B.4: Layout der Unterseite der Platine.

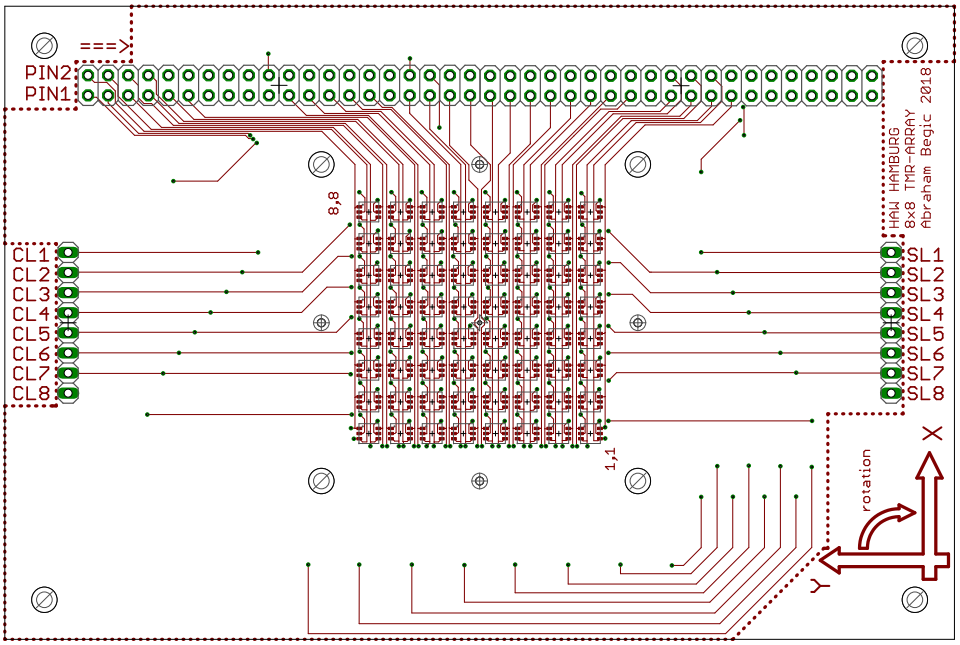


Abbildung B.5: Layout der Oberseite der Platine.

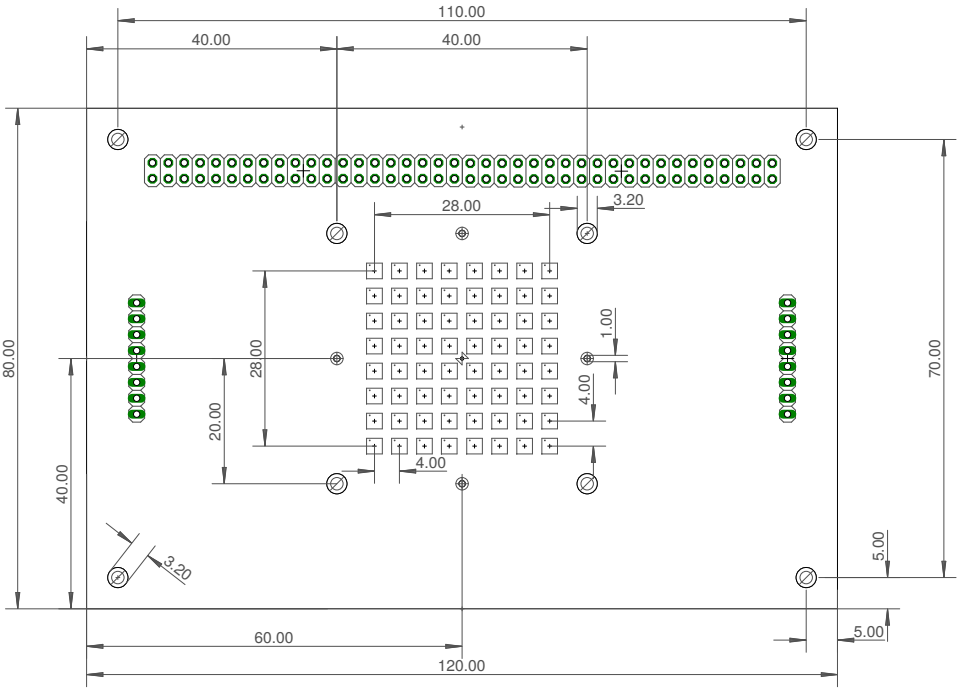


Abbildung B.6: Dimensionen und technische Daten der Platine.

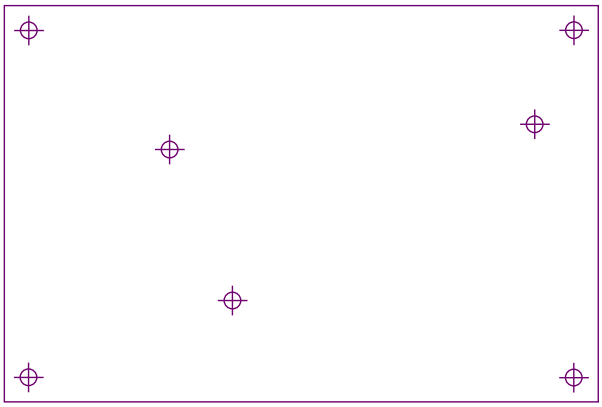


Abbildung B.7: Schablone für die Holzplatte.

C Quellcode

C.1 C-Quellcode

Quellcode C.1: include.h

```
/*
 * include.h
 *
 *  Created on: 03.04.2018
 *      Author: abegic
 */

#ifndef INCLUDE_H_
#define INCLUDE_H_

#include <stdint.h>
#include <stdbool.h>
#include "stdlib.h"
#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"
#include "inc/hw_uart.h"
#include "inc/hw_timer.h"
#include "inc/hw_gpio.h"
#include "inc/hw_pwm.h"
#include "inc/hw_types.h"
#include "driverlib/adc.h"
#include "driverlib/timer.h"
#include "driverlib/gpio.h"
#include "driverlib/timer.h"
#include "driverlib/interrupt.h"
#include "driverlib/pin_map.h"
#include "driverlib/rom.h"
#include "driverlib/rom_map.h"
#include "driverlib/sysctl.h"
#include "driverlib/uart.h"
#include "driverlib/udma.h"
#include "driverlib/pwm.h"
#include "driverlib/ssi.h"
#include "driverlib/systick.h"
#include "driverlib/adc.h"
//#include "utils/uartstdio.h"
//#include "utils/uartstdio.c"
#include <string.h>
#include "driverlib/fpu.h"
#include "inc/hw_memmap.h"
#include "driverlib/debug.h"
#include "driverlib/adc.h"
#include "inc/hw_gpio.h"
#include "inc/hw_ints.h"
#include "inc/hw_memmap.h"
#include "inc/hw_sysctl.h"
#include "math.h"
#include "string.h"
#include "stdio.h"
```

```
50
#define PI 3.14159265
52 #define VDD 3.3
#define NBITS 1
54 #define AMOUNT_SENSORS 8
#define RAD_TO_ANGLE 57.29577950560105 //  =>  180.0 / PI
56
#define SIN_1 ADC_CTL_CH9
58 #define SIN_2 ADC_CTL_CH10
#define SIN_3 ADC_CTL_CH11
60 #define SIN_4 ADC_CTL_CH13
#define SIN_5 ADC_CTL_CH16
62 #define SIN_6 ADC_CTL_CH17
#define SIN_7 ADC_CTL_CH18
64 #define SIN_8 ADC_CTL_CH19

66 #define COS_1 ADC_CTL_CH0
#define COS_2 ADC_CTL_CH1
68 #define COS_3 ADC_CTL_CH2
#define COS_4 ADC_CTL_CH3
70 #define COS_5 ADC_CTL_CH4
#define COS_6 ADC_CTL_CH6
72 #define COS_7 ADC_CTL_CH7
#define COS_8 ADC_CTL_CH8
74
#define HIGH 0xFF
76 #define LOW  0x00

78 // Arrays fuer die ADC-Werte
unsigned int ADC_Values_SS0[8];
80 unsigned int ADC_Values_SS1[4];
unsigned int ADC_Values_SS2[4];
82
// Variable zum Auslesen des Systemtaktes
84
// Interne Variablen zur Kalibierung des einzelnen Sensors
86 //int Calibration_State = 0;
//int Set_State = 0;
88
// Zwischenspeicher fuer printf-Funktionen
90 char buffer[100];

92 // entsprechende Sin- und Cos-Spannungen
double SINUS_Voltages[AMOUNT_SENSORS][AMOUNT_SENSORS];
94 double COSINUS_Voltages[AMOUNT_SENSORS][AMOUNT_SENSORS];

96

#endif /* INCLUDE_H_ */
```

Quellcode C.2: init-prototyp.h

```
/*
 * init_prototyp.h
 *
 *  Created on: 04.04.2018
 *      Author: abegic
 */

#ifndef INIT_PROTOTYP_H_
#define INIT_PROTOTYP_H_

void reset_vtg(void);
void UARTSend(const uint8_t *pui8Buffer, uint32_t ui32Count);
void Init_LED_GPIOs(void);
void enable_init_gpios(void);
void gpio_set_input(void);
void enable_set_low(void);
void Init_UART(uint32_t takt);
void Init_ADCs(void);
void Switch_LED_on(void);
void Switch_LED_off(void);
void enable_set_high(int column);
void Get_Values(void);
void vals_to_terminal(void);
void Blink_LED(void);
void UARTIntHandler(void);
void PortJIntHandler();
void Systick_Interrupt_Handler(void);
void Init_GPIO_and_Interrupt_Port_J(void);
void Init_SysTick_Interrupt(uint32_t takt);


#endif /* INIT_PROTOTYP_H_ */
```

Quellcode C.3: main.c

```
/*
 *
 *
 *
 *
 */
#include "include.h"
#include "init_prototyp.h"


 int main(void) {

     uint32_t sysClock = 0;

    // set processor to 80 MHz (intern PLL with 320 MHz)
    sysClock = SysCtlClockFreqSet(  SYSCTL_OSC_INT |
                                    SYSCTL_USE_PLL |
                                    SYSCTL_CFG_VCO_320,
                                    80000000-1);

    IntMasterDisable(); // turn off master
    Init_LED_GPIOs();
    enable_init_gpios();
    gpio_set_input();
    Init_ADCs();
    Init_SysTick_Interrupt(sysClock);
    Init_GPIO_and_Interrupt_Port_J();
    Init_UART(sysClock);
    UARTSend((uint8_t *)"\n-Hello World",strlen("\n-Hello World"));

    //enable_set_low();
    IntMasterEnable();  // turn on master


    while(1)
    {
    };
}
```

Quellcode C.4: Enable.c

```
/*
 * Enable.c
 *
 *  Created on: 24.03.2018
 *      Author: abegic
 */

#include "include.h"


void Init_LED_GPIOs()
{
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPION);
    GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_1);
    GPIOPinTypeGPIOOutput(GPIO_PORTN_BASE, GPIO_PIN_0);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_4);
    GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_0);

}
 /*
  * Initialize the gpios for the enable lines. All these lines
  * need to be set to high after initialization
  */
void enable_init_gpios(void)
{



    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
        GPIOPinTypeGPIOInput(GPIO_PORTA_BASE, GPIO_PIN_2);          // VC_8
        GPIOPinTypeGPIOInput(GPIO_PORTA_BASE, GPIO_PIN_3);          // GC_8
        GPIOPinTypeGPIOInput(GPIO_PORTA_BASE, GPIO_PIN_4);          // VS_8
        GPIOPinTypeGPIOInput(GPIO_PORTA_BASE, GPIO_PIN_5);          // GC_7
        GPIOPinTypeGPIOInput(GPIO_PORTA_BASE, GPIO_PIN_6);          // VC_6
        GPIOPinTypeGPIOInput(GPIO_PORTA_BASE, GPIO_PIN_7);          // GC_6

        GPIOPadConfigSet(GPIO_PORTA_BASE, GPIO_PIN_2, GPIO_STRENGTH_2MA,
                                            GPIO_PIN_TYPE_STD_WPU);    // VC_8
        GPIOPadConfigSet(GPIO_PORTA_BASE, GPIO_PIN_3, GPIO_STRENGTH_2MA,
                                            GPIO_PIN_TYPE_STD_WPU);    // GC_8
        GPIOPadConfigSet(GPIO_PORTA_BASE, GPIO_PIN_4, GPIO_STRENGTH_2MA,
                                            GPIO_PIN_TYPE_STD_WPU);    // VS_8
        GPIOPadConfigSet(GPIO_PORTA_BASE, GPIO_PIN_5, GPIO_STRENGTH_2MA,
                                            GPIO_PIN_TYPE_STD_WPU);    // GC_7
        GPIOPadConfigSet(GPIO_PORTA_BASE, GPIO_PIN_6, GPIO_STRENGTH_2MA,
                                            GPIO_PIN_TYPE_STD_WPU);    // VC_6
        GPIOPadConfigSet(GPIO_PORTA_BASE, GPIO_PIN_7, GPIO_STRENGTH_2MA,
                                            GPIO_PIN_TYPE_STD_WPU);    // GC_6


    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
        GPIOPinTypeGPIOInput(GPIO_PORTB_BASE, GPIO_PIN_2);          // GC_1
        GPIOPinTypeGPIOInput(GPIO_PORTB_BASE, GPIO_PIN_3);          // VS_1


        GPIOPadConfigSet(GPIO_PORTB_BASE, GPIO_PIN_2, GPIO_STRENGTH_2MA,
                                            GPIO_PIN_TYPE_STD_WPU);    // GC_1
        GPIOPadConfigSet(GPIO_PORTB_BASE, GPIO_PIN_3, GPIO_STRENGTH_2MA,
                                            GPIO_PIN_TYPE_STD_WPU);    // VS_1


```

```
64      SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
            GPIOPinTypeGPIOInput(GPIO_PORTF_BASE, GPIO_PIN_0);          // VC_1
66          GPIOPinTypeGPIOInput(GPIO_PORTF_BASE, GPIO_PIN_1);          // GS_1
            GPIOPinTypeGPIOInput(GPIO_PORTF_BASE, GPIO_PIN_4);          // GS_2
68
            GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_STRENGTH_2MA,
70                                                  GPIO_PIN_TYPE_STD_WPU);   // VC_1
            GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_1, GPIO_STRENGTH_2MA,
72                                                  GPIO_PIN_TYPE_STD_WPU);   // GS_1
            GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_4, GPIO_STRENGTH_2MA,
74                                                  GPIO_PIN_TYPE_STD_WPU);   // GS_1
76
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOG);
78          GPIOPinTypeGPIOInput(GPIO_PORTG_BASE, GPIO_PIN_0);          // VS_6
            GPIOPinTypeGPIOInput(GPIO_PORTG_BASE, GPIO_PIN_1);          // GS_6
80
82          GPIOPadConfigSet(GPIO_PORTG_BASE, GPIO_PIN_0, GPIO_STRENGTH_2MA,
                                                    GPIO_PIN_TYPE_STD_WPU);   // VS_6
84          GPIOPadConfigSet(GPIO_PORTG_BASE, GPIO_PIN_1, GPIO_STRENGTH_2MA,
                                                    GPIO_PIN_TYPE_STD_WPU);   // GS_6
86
88      SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOH);
            GPIOPinTypeGPIOInput(GPIO_PORTH_BASE, GPIO_PIN_0);          // GS_8
90          GPIOPinTypeGPIOInput(GPIO_PORTH_BASE, GPIO_PIN_1);          // VC_7
            GPIOPinTypeGPIOInput(GPIO_PORTH_BASE, GPIO_PIN_2);          // GS_7
92          GPIOPinTypeGPIOInput(GPIO_PORTH_BASE, GPIO_PIN_3);          // VS_7
94          GPIOPadConfigSet(GPIO_PORTH_BASE, GPIO_PIN_0, GPIO_STRENGTH_2MA,
                                                    GPIO_PIN_TYPE_STD_WPU);   // GS_8
96          GPIOPadConfigSet(GPIO_PORTH_BASE, GPIO_PIN_1, GPIO_STRENGTH_2MA,
                                                    GPIO_PIN_TYPE_STD_WPU);   // VC_7
98          GPIOPadConfigSet(GPIO_PORTH_BASE, GPIO_PIN_2, GPIO_STRENGTH_2MA,
                                                    GPIO_PIN_TYPE_STD_WPU);   // GS_7
100         GPIOPadConfigSet(GPIO_PORTH_BASE, GPIO_PIN_3, GPIO_STRENGTH_2MA,
                                                    GPIO_PIN_TYPE_STD_WPU);   // VS_7
102
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOK);
104         GPIOPinTypeGPIOInput(GPIO_PORTK_BASE, GPIO_PIN_6);          // GC_2
            GPIOPinTypeGPIOInput(GPIO_PORTK_BASE, GPIO_PIN_7);          // VC_2
106
            GPIOPadConfigSet(GPIO_PORTK_BASE, GPIO_PIN_6, GPIO_STRENGTH_2MA,
108                                                 GPIO_PIN_TYPE_STD_WPU);   // GC_2
            GPIOPadConfigSet(GPIO_PORTK_BASE, GPIO_PIN_7, GPIO_STRENGTH_2MA,
110                                                 GPIO_PIN_TYPE_STD_WPU);   // VC_2
112
114     SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOL);
            GPIOPinTypeGPIOInput(GPIO_PORTL_BASE, GPIO_PIN_0);          // VC_4
116         GPIOPinTypeGPIOInput(GPIO_PORTL_BASE, GPIO_PIN_1);          // GC_4
            GPIOPinTypeGPIOInput(GPIO_PORTL_BASE, GPIO_PIN_2);          // GS_4
118         GPIOPinTypeGPIOInput(GPIO_PORTL_BASE, GPIO_PIN_3);          // VS_4
            GPIOPinTypeGPIOInput(GPIO_PORTL_BASE, GPIO_PIN_4);          // VS_2
120
            GPIOPadConfigSet(GPIO_PORTL_BASE, GPIO_PIN_0, GPIO_STRENGTH_2MA,
122                                                 GPIO_PIN_TYPE_STD_WPU);   // VC_4
            GPIOPadConfigSet(GPIO_PORTL_BASE, GPIO_PIN_1, GPIO_STRENGTH_2MA,
124                                                 GPIO_PIN_TYPE_STD_WPU);   // GC_4
            GPIOPadConfigSet(GPIO_PORTL_BASE, GPIO_PIN_2, GPIO_STRENGTH_2MA,
126                                                 GPIO_PIN_TYPE_STD_WPU);   // GS_4
            GPIOPadConfigSet(GPIO_PORTL_BASE, GPIO_PIN_3, GPIO_STRENGTH_2MA,
128                                                 GPIO_PIN_TYPE_STD_WPU);   // VS_4
```

```
        GPIOPadConfigSet(GPIO_PORTL_BASE, GPIO_PIN_4, GPIO_STRENGTH_2MA,
                                                GPIO_PIN_TYPE_STD_WPU);   // VS_2

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOM);
        GPIOPinTypeGPIOInput(GPIO_PORTM_BASE, GPIO_PIN_0);          // VS_5
        GPIOPinTypeGPIOInput(GPIO_PORTM_BASE, GPIO_PIN_1);          // GS_5
        GPIOPinTypeGPIOInput(GPIO_PORTM_BASE, GPIO_PIN_2);          // GC_5
        GPIOPinTypeGPIOInput(GPIO_PORTM_BASE, GPIO_PIN_3);          // VC_5

        GPIOPadConfigSet(GPIO_PORTM_BASE, GPIO_PIN_0, GPIO_STRENGTH_2MA,
                                                GPIO_PIN_TYPE_STD_WPU);   // VS_5
        GPIOPadConfigSet(GPIO_PORTM_BASE, GPIO_PIN_1, GPIO_STRENGTH_2MA,
                                                GPIO_PIN_TYPE_STD_WPU);   // GS_5
        GPIOPadConfigSet(GPIO_PORTM_BASE, GPIO_PIN_2, GPIO_STRENGTH_2MA,
                                                GPIO_PIN_TYPE_STD_WPU);   // GC_5
        GPIOPadConfigSet(GPIO_PORTM_BASE, GPIO_PIN_3, GPIO_STRENGTH_2MA,
                                                GPIO_PIN_TYPE_STD_WPU);   // VC_5

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOQ);
        GPIOPinTypeGPIOInput(GPIO_PORTQ_BASE, GPIO_PIN_0);          // VC_3
        GPIOPinTypeGPIOInput(GPIO_PORTQ_BASE, GPIO_PIN_1);          // GC_3
        GPIOPinTypeGPIOInput(GPIO_PORTQ_BASE, GPIO_PIN_2);          // GS_3
        GPIOPinTypeGPIOInput(GPIO_PORTQ_BASE, GPIO_PIN_3);          // VS_3

        GPIOPadConfigSet(GPIO_PORTQ_BASE, GPIO_PIN_0, GPIO_STRENGTH_2MA,
                                                GPIO_PIN_TYPE_STD_WPU);   // VC_3
        GPIOPadConfigSet(GPIO_PORTQ_BASE, GPIO_PIN_1, GPIO_STRENGTH_2MA,
                                                GPIO_PIN_TYPE_STD_WPU);   // GC_3
        GPIOPadConfigSet(GPIO_PORTQ_BASE, GPIO_PIN_2, GPIO_STRENGTH_2MA,
                                                GPIO_PIN_TYPE_STD_WPU);   // GS_3
        GPIOPadConfigSet(GPIO_PORTQ_BASE, GPIO_PIN_3, GPIO_STRENGTH_2MA,
                                                GPIO_PIN_TYPE_STD_WPU);   // VS_3

}

/*
 * Initialize the gpios for the enable lines.
 */
void gpio_set_input(void)
{
        GPIOPinTypeGPIOInput(GPIO_PORTA_BASE, GPIO_PIN_2);          // VC_8
        GPIOPinTypeGPIOInput(GPIO_PORTA_BASE, GPIO_PIN_3);          // GC_8
        GPIOPinTypeGPIOInput(GPIO_PORTA_BASE, GPIO_PIN_4);          // VS_8
        GPIOPinTypeGPIOInput(GPIO_PORTA_BASE, GPIO_PIN_5);          // GC_7
        GPIOPinTypeGPIOInput(GPIO_PORTA_BASE, GPIO_PIN_6);          // VC_6
        GPIOPinTypeGPIOInput(GPIO_PORTA_BASE, GPIO_PIN_7);          // GC_6

        GPIOPinTypeGPIOInput(GPIO_PORTB_BASE, GPIO_PIN_2);          // GC_1
        GPIOPinTypeGPIOInput(GPIO_PORTB_BASE, GPIO_PIN_3);          // VS_1

        GPIOPinTypeGPIOInput(GPIO_PORTF_BASE, GPIO_PIN_0);          // VC_1
        GPIOPinTypeGPIOInput(GPIO_PORTF_BASE, GPIO_PIN_1);          // GS_1
        GPIOPinTypeGPIOInput(GPIO_PORTF_BASE, GPIO_PIN_4);          // GS_2

        GPIOPinTypeGPIOInput(GPIO_PORTG_BASE, GPIO_PIN_0);          // VS_6
        GPIOPinTypeGPIOInput(GPIO_PORTG_BASE, GPIO_PIN_1);          // GS_6

        GPIOPinTypeGPIOInput(GPIO_PORTH_BASE, GPIO_PIN_0);          // GS_8
        GPIOPinTypeGPIOInput(GPIO_PORTH_BASE, GPIO_PIN_1);          // VC_7
        GPIOPinTypeGPIOInput(GPIO_PORTH_BASE, GPIO_PIN_2);          // GS_7
        GPIOPinTypeGPIOInput(GPIO_PORTH_BASE, GPIO_PIN_3);          // VS_7

        GPIOPinTypeGPIOInput(GPIO_PORTK_BASE, GPIO_PIN_6);          // GC_2
```

```
194         GPIOPinTypeGPIOInput(GPIO_PORTK_BASE, GPIO_PIN_7);          // VC_2
196
            GPIOPinTypeGPIOInput(GPIO_PORTL_BASE, GPIO_PIN_0);          // VC_4
198         GPIOPinTypeGPIOInput(GPIO_PORTL_BASE, GPIO_PIN_1);          // GC_4
            GPIOPinTypeGPIOInput(GPIO_PORTL_BASE, GPIO_PIN_2);          // GS_4
200         GPIOPinTypeGPIOInput(GPIO_PORTL_BASE, GPIO_PIN_3);          // VS_4
            GPIOPinTypeGPIOInput(GPIO_PORTL_BASE, GPIO_PIN_4);          // VS_2
202
            GPIOPinTypeGPIOInput(GPIO_PORTM_BASE, GPIO_PIN_0);          // VS_5
204         GPIOPinTypeGPIOInput(GPIO_PORTM_BASE, GPIO_PIN_1);          // GS_5
            GPIOPinTypeGPIOInput(GPIO_PORTM_BASE, GPIO_PIN_2);          // GC_5
206         GPIOPinTypeGPIOInput(GPIO_PORTM_BASE, GPIO_PIN_3);          // VC_5
208
            GPIOPinTypeGPIOInput(GPIO_PORTQ_BASE, GPIO_PIN_0);          // VC_3
210         GPIOPinTypeGPIOInput(GPIO_PORTQ_BASE, GPIO_PIN_1);          // GC_3
            GPIOPinTypeGPIOInput(GPIO_PORTQ_BASE, GPIO_PIN_2);          // GS_3
212         GPIOPinTypeGPIOInput(GPIO_PORTQ_BASE, GPIO_PIN_3);          // VS_3
214
216 }
218
    void enable_set_low(void)
220 {
            GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_0);         // VC_1
222         GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_3);         // VS_1
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0,LOW);
224         GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3,LOW);
226         GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_2);         // GC_1
            GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1);         // GS_1
228         GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2,LOW);
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1,LOW);
230
232         GPIOPinTypeGPIOOutput(GPIO_PORTK_BASE, GPIO_PIN_7);         // VC_2
            GPIOPinTypeGPIOOutput(GPIO_PORTL_BASE, GPIO_PIN_4);         // VS_2
234         GPIOPinWrite(GPIO_PORTK_BASE, GPIO_PIN_7,LOW);
            GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_4,LOW);
236
            GPIOPinTypeGPIOOutput(GPIO_PORTK_BASE, GPIO_PIN_6);         // GC_2
238         GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_4);         // GS_2
            GPIOPinWrite(GPIO_PORTK_BASE, GPIO_PIN_6,LOW);
240         GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_4,LOW);
242
244         GPIOPinTypeGPIOOutput(GPIO_PORTQ_BASE, GPIO_PIN_0);         // VC_3
            GPIOPinTypeGPIOOutput(GPIO_PORTQ_BASE, GPIO_PIN_3);         // VS_3
246         GPIOPinWrite(GPIO_PORTQ_BASE, GPIO_PIN_0,LOW);
            GPIOPinWrite(GPIO_PORTQ_BASE, GPIO_PIN_3,LOW);
248
            GPIOPinTypeGPIOOutput(GPIO_PORTQ_BASE, GPIO_PIN_1);         // GC_3
250         GPIOPinTypeGPIOOutput(GPIO_PORTQ_BASE, GPIO_PIN_2);         // GS_3
            GPIOPinWrite(GPIO_PORTQ_BASE, GPIO_PIN_1,LOW);
252         GPIOPinWrite(GPIO_PORTQ_BASE, GPIO_PIN_2,LOW);
254
            GPIOPinTypeGPIOOutput(GPIO_PORTL_BASE, GPIO_PIN_0);         // VC_4
256         GPIOPinTypeGPIOOutput(GPIO_PORTL_BASE, GPIO_PIN_3);         // VS_4
            GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_0,LOW);
258         GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_3,LOW);
```

```
260         GPIOPinTypeGPIOOutput(GPIO_PORTL_BASE, GPIO_PIN_1);          // GC_4
            GPIOPinTypeGPIOOutput(GPIO_PORTL_BASE, GPIO_PIN_2);          // GS_4
262         GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_1,LOW);
            GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_2,LOW);
264

266


268
            GPIOPinTypeGPIOOutput(GPIO_PORTM_BASE, GPIO_PIN_3);          // VC_5
270         GPIOPinTypeGPIOOutput(GPIO_PORTM_BASE, GPIO_PIN_0);          // VS_5
            GPIOPinWrite(GPIO_PORTM_BASE, GPIO_PIN_3,LOW);
272         GPIOPinWrite(GPIO_PORTM_BASE, GPIO_PIN_0,LOW);

274         GPIOPinTypeGPIOOutput(GPIO_PORTM_BASE, GPIO_PIN_1);          // GS_5
            GPIOPinTypeGPIOOutput(GPIO_PORTM_BASE, GPIO_PIN_2);          // GC_5
276         GPIOPinWrite(GPIO_PORTM_BASE, GPIO_PIN_1,LOW);
            GPIOPinWrite(GPIO_PORTM_BASE, GPIO_PIN_2,LOW);
278
            GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_6);          // VC_6
280         GPIOPinTypeGPIOOutput(GPIO_PORTG_BASE, GPIO_PIN_0);          // VS_6
            GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_6,LOW);
282         GPIOPinWrite(GPIO_PORTG_BASE, GPIO_PIN_0,LOW);

284         GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_7);          // GC_6
            GPIOPinTypeGPIOOutput(GPIO_PORTG_BASE, GPIO_PIN_1);          // GS_6
286         GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_7,LOW);
            GPIOPinWrite(GPIO_PORTG_BASE, GPIO_PIN_1,LOW);
288
            GPIOPinTypeGPIOOutput(GPIO_PORTH_BASE, GPIO_PIN_1);          // VC_7
290         GPIOPinTypeGPIOOutput(GPIO_PORTH_BASE, GPIO_PIN_3);          // VS_7
            GPIOPinWrite(GPIO_PORTH_BASE, GPIO_PIN_1,LOW);
292         GPIOPinWrite(GPIO_PORTH_BASE, GPIO_PIN_3,LOW);

294         GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_5);          // GC_7
            GPIOPinTypeGPIOOutput(GPIO_PORTH_BASE, GPIO_PIN_2);          // GS_7
296         GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_5,LOW);
            GPIOPinWrite(GPIO_PORTH_BASE, GPIO_PIN_2,LOW);
298


300
            GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_2);          // VC_8
302         GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_4);          // VS_8
            GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_2,LOW);
304         GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_4,LOW);

306         GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_3);          // GC_8
            GPIOPinTypeGPIOOutput(GPIO_PORTH_BASE, GPIO_PIN_0);          // GS_8
308         GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_3,LOW);
            GPIOPinWrite(GPIO_PORTH_BASE, GPIO_PIN_0,LOW);
310
}
```

Quellcode C.5: init-ADC.c

```
/*
 * Init_ADCs.c
 *
 *  Created on: 04.04.2018
 *      Author: abegic
 */

#include "include.h"

void Init_ADCs(void)
{   // ADC0-Peripherie Einschalten (SIN-Signale)
    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
    // ADC1-Peripherie Einschalten (COS-Signale)
    SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC1);

    // Sample Sequencer 0 fuer ADC0, hoechste Prio
    ADCSequenceConfigure(ADC0_BASE, 0, ADC_TRIGGER_PROCESSOR, 0);
    // Sample Sequencer 1 fuer ADC1, zweithoechste Prio
    ADCSequenceConfigure(ADC1_BASE, 1, ADC_TRIGGER_PROCESSOR, 1);
    // Sample Sequencer 2 fuer ADC1, dritthoechste Prio
    ADCSequenceConfigure(ADC1_BASE, 2, ADC_TRIGGER_PROCESSOR, 2);


    // Verknuepfung der Eingaenge mit dem ADC (+ Sample Sequenzer)
    // Verknuefung mit dem Sample Sequenzer 0
    ADCSequenceStepConfigure(ADC0_BASE, 0, 0, SIN_1);
    ADCSequenceStepConfigure(ADC0_BASE, 0, 1, SIN_2);
    ADCSequenceStepConfigure(ADC0_BASE, 0, 2, SIN_3);
    ADCSequenceStepConfigure(ADC0_BASE, 0, 3, SIN_4);
    ADCSequenceStepConfigure(ADC0_BASE, 0, 4, SIN_5);
    ADCSequenceStepConfigure(ADC0_BASE, 0, 5, SIN_6);
    ADCSequenceStepConfigure(ADC0_BASE, 0, 6, SIN_7);
    ADCSequenceStepConfigure(ADC0_BASE, 0, 7, SIN_8);

    // Verknuepfung mit dem Sample Sequenzer 1
    ADCSequenceStepConfigure(ADC1_BASE, 1, 0, COS_1);
    ADCSequenceStepConfigure(ADC1_BASE, 1, 1, COS_2);
    ADCSequenceStepConfigure(ADC1_BASE, 1, 2, COS_3);
    ADCSequenceStepConfigure(ADC1_BASE, 1, 3, COS_4);


    // Verknuepfung mit dem Sample Sequenzer 2
    ADCSequenceStepConfigure(ADC1_BASE, 2, 0, COS_5);
    ADCSequenceStepConfigure(ADC1_BASE, 2, 1, COS_6);
    ADCSequenceStepConfigure(ADC1_BASE, 2, 2, COS_7);

    ADCSequenceStepConfigure(ADC1_BASE, 2, 3, COS_8| ADC_CTL_IE | ADC_CTL_END);
    //  Definition der letzten Abtastung  inkl. Interrupt Erzeugung

    // Hardwareoversampling einschalten (64-fach)
    //ADCHardwareOversampleConfigure(ADC0_BASE,64);
    //ADCHardwareOversampleConfigure(ADC1_BASE,64);

    // ADCs mit den Sample Sequenzern aktivieren
    ADCSequenceEnable(ADC0_BASE, 0); // ADC0 for sample sequenzer 0
    ADCSequenceEnable(ADC1_BASE, 1); // ADC1 for sample sequenzer 1
    ADCSequenceEnable(ADC1_BASE, 2); // ADC1 for sample sequenzer 2
}
```

Quellcode C.6: enable-set-high.c

```
/*
 * enable_set_high.c
 *
 *  Created on: 04.04.2018
 *      Author: abegic
 */

#include "include.h"
#include "init_prototyp.h"

// Anschalten der jeweiligen Line
void enable_set_high(int column)
{
    enable_set_low();           // Alle Pins werden erstmal auf Low gesetzt
    SysCtlDelay(80000);         // Wartezeit
    gpio_set_input();           // Alle Gpios werden auf Input Gesetzt

    switch(column)
    {
    // Einschalten der LINE 1
    case 0:
        // Die notwendigen GPIOS werden auf Outputgesetzt
        GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_2);          // GC_1
        GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1);          // GS_1
        GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_0);          // VC_1
        GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_3);          // VS_1

        // Die entsprechenden GPIOS Werden Auf High und LOW gesetzt
        GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_2,LOW);               // GC_1
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1,LOW);               // GS_1
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_PIN_0);       // VC_1
        GPIOPinWrite(GPIO_PORTB_BASE, GPIO_PIN_3, GPIO_PIN_3);       // VS_1

        break;

        // Einschalten der LINE 2

    case 1:
        // Die notwendigen GPIOS werden auf Outputgesetzt
        GPIOPinTypeGPIOOutput(GPIO_PORTK_BASE, GPIO_PIN_6);          // GC_2
        GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_4);          // GS_2
        GPIOPinTypeGPIOOutput(GPIO_PORTK_BASE, GPIO_PIN_7);          // VC_2
        GPIOPinTypeGPIOOutput(GPIO_PORTL_BASE, GPIO_PIN_4);          // VS_2

        // Die entsprechenden GPIOS Werden Auf High und LOW gesetzt
        GPIOPinWrite(GPIO_PORTK_BASE, GPIO_PIN_6,LOW);               // GC_2
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_4,LOW);               // GS_2
        GPIOPinWrite(GPIO_PORTK_BASE, GPIO_PIN_7, GPIO_PIN_7);       // VC_2
        GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_4, GPIO_PIN_4);       // VS_2

        break;

        // Einschalten der LINE 3
    case 2:
        // Die notwendigen GPIOS werden auf Outputgesetzt
        GPIOPinTypeGPIOOutput(GPIO_PORTQ_BASE, GPIO_PIN_1);          // GC_3
        GPIOPinTypeGPIOOutput(GPIO_PORTQ_BASE, GPIO_PIN_2);          // GS_3
        GPIOPinTypeGPIOOutput(GPIO_PORTQ_BASE, GPIO_PIN_0);          // VC_3
        GPIOPinTypeGPIOOutput(GPIO_PORTQ_BASE, GPIO_PIN_3);          // VS_3

        // Die entsprechenden GPIOS Werden Auf High und LOW gesetzt
        GPIOPinWrite(GPIO_PORTQ_BASE, GPIO_PIN_1,LOW);               // GC_3
        GPIOPinWrite(GPIO_PORTQ_BASE, GPIO_PIN_2,LOW);               // GS_3
```

```
64          GPIOPinWrite(GPIO_PORTQ_BASE, GPIO_PIN_0, GPIO_PIN_0);       // VC_3
            GPIOPinWrite(GPIO_PORTQ_BASE, GPIO_PIN_3, GPIO_PIN_3);       // VS_3
66
            break;
68
            // Einschalten der LINE 4
70      case 3:
            // Die notwendigen GPIOS werden auf Outputgesetzt
72          GPIOPinTypeGPIOOutput(GPIO_PORTL_BASE, GPIO_PIN_1);          // GC_4
            GPIOPinTypeGPIOOutput(GPIO_PORTL_BASE, GPIO_PIN_2);          // GS_4
74          GPIOPinTypeGPIOOutput(GPIO_PORTL_BASE, GPIO_PIN_0);          // VC_4
            GPIOPinTypeGPIOOutput(GPIO_PORTL_BASE, GPIO_PIN_3);          // VS_4
76
            // Die entsprechenden GPIOS Werden Auf High und LOW gesetzt
78          GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_1,LOW);               // GC_4
            GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_2,LOW);               // GS_4
80          GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_0, GPIO_PIN_0);       // VC_4
            GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_3, GPIO_PIN_3);       // VS_4
82          break;
            // Einschalten der LINE 5
84      case 4:
            // Die notwendigen GPIOS werden auf Outputgesetzt
86          GPIOPinTypeGPIOOutput(GPIO_PORTM_BASE, GPIO_PIN_1);          // GS_5
            GPIOPinTypeGPIOOutput(GPIO_PORTM_BASE, GPIO_PIN_2);          // GC_5
88          GPIOPinTypeGPIOOutput(GPIO_PORTM_BASE, GPIO_PIN_3);          // VC_5
            GPIOPinTypeGPIOOutput(GPIO_PORTM_BASE, GPIO_PIN_0);          // VS_5
90
            // Die entsprechenden GPIOS Werden Auf High und LOW gesetzt
92          GPIOPinWrite(GPIO_PORTM_BASE, GPIO_PIN_1,LOW);               // GS_5
            GPIOPinWrite(GPIO_PORTM_BASE, GPIO_PIN_2,LOW);               // GC_5
94          GPIOPinWrite(GPIO_PORTM_BASE, GPIO_PIN_3, GPIO_PIN_3);       // VC_5
            GPIOPinWrite(GPIO_PORTM_BASE, GPIO_PIN_0, GPIO_PIN_0);       // VS_5
96
            break;
98
            // Einschalten der LINE 6
100     case 5:
            // Die notwendigen GPIOS werden auf Outputgesetzt
102         GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_7);          // GC_6
            GPIOPinTypeGPIOOutput(GPIO_PORTG_BASE, GPIO_PIN_1);          // GS_6
104         GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_6);          // VC_6
            GPIOPinTypeGPIOOutput(GPIO_PORTG_BASE, GPIO_PIN_0);          // VS_6
106
            // Die entsprechenden GPIOS Werden Auf High und LOW gesetzt
108         GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_7,LOW);               // GC_6
            GPIOPinWrite(GPIO_PORTG_BASE, GPIO_PIN_1,LOW);               // GS_6
110         GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_6, GPIO_PIN_6);       // VC_6
            GPIOPinWrite(GPIO_PORTG_BASE, GPIO_PIN_0, GPIO_PIN_0);       // VS_6
112
            break;
114
            // Einschalten der LINE 7
116     case 6:
            // Die notwendigen GPIOS werden auf Outputgesetzt
118         GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_5);          // GC_7
            GPIOPinTypeGPIOOutput(GPIO_PORTH_BASE, GPIO_PIN_2);          // GS_7
120         GPIOPinTypeGPIOOutput(GPIO_PORTH_BASE, GPIO_PIN_1);          // VC_7
            GPIOPinTypeGPIOOutput(GPIO_PORTH_BASE, GPIO_PIN_3);          // VS_7
122
            // Die entsprechenden GPIOS Werden Auf High und LOW gesetzt
124         GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_5,LOW);               // GC_7
            GPIOPinWrite(GPIO_PORTH_BASE, GPIO_PIN_2,LOW);               // GS_7
126         GPIOPinWrite(GPIO_PORTH_BASE, GPIO_PIN_1, GPIO_PIN_1);       // VC_7
            GPIOPinWrite(GPIO_PORTH_BASE, GPIO_PIN_3, GPIO_PIN_3);       // VS_7
128
```

```
            break;
130
            // Einschalten der LINE 8
132     case 7:
            // Die notwendigen GPIOS werden auf Outputgesetzt
134         GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_3);          // GC_8
            GPIOPinTypeGPIOOutput(GPIO_PORTH_BASE, GPIO_PIN_0);          // GS_8
136         GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_2);          // VC_8
            GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_4);          // VS_8
138
            // Die entsprechenden GPIOS Werden Auf High und LOW gesetzt
140         GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_3,LOW);               // GC_8
            GPIOPinWrite(GPIO_PORTH_BASE, GPIO_PIN_0,LOW);               // GS_8
142         GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_2, GPIO_PIN_2);       // VC_8
            GPIOPinWrite(GPIO_PORTA_BASE, GPIO_PIN_4, GPIO_PIN_4);       // VS_8
144
            break;
146
    }
148
}
```

Quellcode C.7: Get-VAlues.c

```
/*
 *
 *
 * Get_VAlues.c
 *
 *  Created on: 04.04.2018
 *      Author: abegic
 */

/*
* Triggering of ADC process
*/

#include "include.h"
#include "init_prototyp.h"

void Get_Values(void)
{

    int run = 0;
    int i=0;
    int j=0;
     for (i=0;i<=7;i++)
     {
         for(j=0;j<=7;j++)
         {
         SINUS_Voltages[i][j]=0;
         }
         SINUS_Voltages[i][j]=0;
     }


    for(run=0;run<=7;run++)
    {


        enable_set_high(run);

        SysCtlDelay(80000); // Wartezeit
        // Triggern des ADC-Processors => Starten des Abtastvorgangs
        ADCProcessorTrigger(ADC0_BASE, 0);
        ADCProcessorTrigger(ADC1_BASE, 1);
        ADCProcessorTrigger(ADC1_BASE, 2);

        while(!ADCIntStatus(ADC1_BASE, 2, false))
        {
            // Warten bis alle Sample_Sequencer fertig sind und ADC-Werte vorliegen
        }
        ADCIntClear(ADC1_BASE, 2); // Quittierung des Interrupts

        enable_set_low();
        SysCtlDelay(8000);// Wartezeit
        // Auslesen der ADC-Werte und Speicherug in den globalen Variablen
        ADCSequenceDataGet(ADC0_BASE, 0, ADC_Values_SS0);
        ADCSequenceDataGet(ADC1_BASE, 1, ADC_Values_SS1);
        ADCSequenceDataGet(ADC1_BASE, 2, ADC_Values_SS2);

        SINUS_Voltages[run][0]      =(double) ADC_Values_SS0[0];
        SINUS_Voltages[run][1]      =(double) ADC_Values_SS0[1];
        SINUS_Voltages[run][2]      =(double) ADC_Values_SS0[2];
        SINUS_Voltages[run][3]      =(double) ADC_Values_SS0[3];
        SINUS_Voltages[run][4]      =(double) ADC_Values_SS0[4];
        SINUS_Voltages[run][5]      =(double) ADC_Values_SS0[5];
```

```
64          SINUS_Voltages[run][6]      =(double) ADC_Values_SS0[6];
            SINUS_Voltages[run][7]      =(double) ADC_Values_SS0[7];
66
            COSINUS_Voltages[run][0]    =(double) ADC_Values_SS1[0];
68          COSINUS_Voltages[run][1]    =(double) ADC_Values_SS1[1];
            COSINUS_Voltages[run][2]    =(double) ADC_Values_SS1[2];
70          COSINUS_Voltages[run][3]    =(double) ADC_Values_SS1[3];
            COSINUS_Voltages[run][4]    =(double) ADC_Values_SS2[0];
72          COSINUS_Voltages[run][5]    =(double) ADC_Values_SS2[1];
            COSINUS_Voltages[run][6]    =(double) ADC_Values_SS2[2];
74          COSINUS_Voltages[run][7]    =(double) ADC_Values_SS2[3];

76      }

78  }
```

Quellcode C.8: Interrupt-Handler.c

```
/*
 * Interrupt_Handler.c
 *
 *  Created on: 04.04.2018
 *      Author: abegic
 */

#include "include.h"
#include "init_prototyp.h"

/*****************************************************
 * UART-interrupt-handler
 *****************************************************/

 void UARTIntHandler(void)
 {
    uint32_t ui32Status;
    int32_t rcv_char;
    char rcv_char2[8] = {};
    int a = 0;

    //
    // Get the interrrupt status.
    //
    ui32Status = UARTIntStatus(UART0_BASE, true);

    //
    // Clear the asserted interrupts.
    //
    UARTIntClear(UART0_BASE, ui32Status);
    //
    // Loop while there are characters in the receive FIFO.
    //
    while(UARTCharsAvail(UART0_BASE))
    {
        // Read the next character from the UART and write it back to the UART.
        #if 0
            ROM_UARTCharPutNonBlocking(UART0_BASE,
            ROM_UARTCharGetNonBlocking(UART0_BASE));
        #else
            rcv_char = UARTCharGetNonBlocking(UART0_BASE);
        #endif
         if(a<= 8)
         {
         rcv_char2[a] = (unsigned char) rcv_char;
         }
         a +=1;
    }
    if (strcmp (rcv_char2,"-blk") == 0)
    {
        Blink_LED();
    }
    if(strcmp (rcv_char2,"-test") == 0)
    {
        IntDisable(INT_UART0);
        reset_vtg();
        Get_Values();
        vals_to_terminal();
        IntEnable(INT_UART0);
    }
 }

 /*************************************************************
```

```
64   * GPIO-interrupt handler
     ************************************************************/
66
     void PortJIntHandler()
68   {
        SysTickDisable();
70      reset_vtg();
        Get_Values();
72      vals_to_terminal();
        GPIOIntClear(GPIO_PORTJ_BASE, GPIO_PIN_0);
74   }

76   /************************************************************
      * Systick_Interrupt_Handler
78    ************************************************************/

80   void Systick_Interrupt_Handler(void)
     {
82       Get_Values();
         SysCtlDelay(100);
84   }

86   /************************************************************
      * Initialization of GPIO-Peripherie
88    ************************************************************/

90   void Init_GPIO_and_Interrupt_Port_J(void)
     {
92      // Port J aktivieren
        SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOJ);
94
        // J0 konfigurieren
96      GPIOPinTypeGPIOInput(GPIO_PORTJ_BASE, GPIO_PIN_0);
        /* Weak Pull-Up: Einstellen als Pull-Up, weak = schwaches Pull-Up,
98         bedeutet der Pull-Up Widerstand ist gross.
           Der SW1 zieht den Ausgang and GND!
100     */
        GPIOPadConfigSet(GPIO_PORTJ_BASE, GPIO_PIN_0, GPIO_STRENGTH_2MA,
102                                                   GPIO_PIN_TYPE_STD_WPU);

104     // Interrupt an J0 legen (SW1)
        // Interrupt fuer Port J0 registrieren
106     GPIOIntRegister(GPIO_PORTJ_BASE, PortJIntHandler);
        // Interrupt bei fallender Flanke auf J0
108     GPIOIntTypeSet(GPIO_PORTJ_BASE, GPIO_PIN_0, GPIO_RISING_EDGE);
        IntPrioritySet(47, 0x00); // hoehste Prio
110
        // Interruptfaeigkeit aktivieren
112     GPIOIntEnable(GPIO_PORTJ_BASE, GPIO_PIN_0); // Interrupt fuer J0 aktivieren

114     // SW2 als zusaezlicher Taster an J1
        GPIOPinTypeGPIOInput(GPIO_PORTJ_BASE, GPIO_PIN_1);
116     GPIOPadConfigSet(GPIO_PORTJ_BASE, GPIO_PIN_1, GPIO_STRENGTH_2MA,
                                                      GPIO_PIN_TYPE_STD_WPU);
118  }

120  /************************************************************
      * Initialization of systick-interrupt
122   ************************************************************/

124   void Init_SysTick_Interrupt(uint32_t takt)
      {
126     // Interrupt Aufruf auf 0.5Hz setzen (2-fache der Systemtaktperiode)
        SysTickPeriodSet(takt/2);
128     // Interrupt registrieren und in dem definierten Handler abarbeiten
```

```
    SysTickIntRegister(Systick_Interrupt_Handler);
130 SysTickIntEnable(); // Interrupt aktivieren
    IntPrioritySet(15, 0x20); // zweithoehste Prioritae vergeben
132 SysCtlDelay(5); // kurzes Delay
    //SysTickEnable();  // SysTick einschalten
134 }

136 /************************************************************
     * Initialization of the UART and turn on after initialization
138  ************************************************************/

140 void Init_UART(uint32_t takt)
    {
142
    // Enable the GPIO Peripheral used by the UART.
144 SysCtlPeripheralEnable(SYSCTL_PERIPH_UART0);
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
146
    // Configure GPIO Pins for UART mode.
148 GPIOPinConfigure(GPIO_PA0_U0RX);
    GPIOPinConfigure(GPIO_PA1_U0TX);
150 GPIOPinTypeUART(GPIO_PORTA_BASE, GPIO_PIN_0 | GPIO_PIN_1);

152 // Initialize the UART for console I/O.
    UARTConfigSetExpClk(UART0_BASE, takt, 115200,
154                     (UART_CONFIG_WLEN_8 | UART_CONFIG_STOP_ONE |
                         UART_CONFIG_PAR_NONE));
156
    // Enable the UART interrupt.
158 IntEnable(INT_UART0);
    UARTIntEnable(UART0_BASE, UART_INT_RX | UART_INT_RT);
160 UARTIntRegister(UART0_BASE, UARTIntHandler);
    }
```

Quellcode C.9: reset-vtg.c

```
/*
 * test.c
 *
 *  Created on: 24.03.2018
 *      Author: abegic
 */

/*
 * Reset sine and cosine values for all sensors.
 * This function sets all sine and cosine voltages to zero.
 */

#include "include.h"


void reset_vtg(void)
{
    int i=0;
    int k = 0;
    for(i=0;i<=0;i++)
    {
        for(k=0;k<=0;k++)
        {
            SINUS_Voltages[i][k] = 0;
            COSINUS_Voltages[i][k] = 0;
        }
    }
}


void Switch_LED_on(void)
{
     GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_1, 0x02);
}

 /*
  * Switch on the LED
  */
 void Switch_LED_off(void)
{
     GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_1, 0x00);
}

 /*
  * Short blink of the first LED
  */
 void Blink_LED(void)
{
     int i =0;
     Switch_LED_on();
     for(i = 0; i <= 50000; i++);
     Switch_LED_off();
     for(i = 0; i <= 50000; i++);
}
```

Quellcode C.10: UARTSend.c

```
/*
 * UARTSend.c
 *
 *  Created on: 24.03.2018
 *      Author: abegic
 */

#include "include.h"

  void UARTSend(const uint8_t *pui8Buffer, uint32_t ui32Count)
{
    //
    // Loop while there are more characters to send.
    //
     while(ui32Count--)
    {
        //
        // Write the next character to the UART.
        //
        //UARTCharPutNonBlocking(UART0_BASE, *pui8Buffer++);
        UARTCharPut(UART0_BASE, *pui8Buffer++);
    }
}
```

Quellcode C.11: vals-to-terminal.c

```
/*
 * vals_to_terminal.c
 *
 *  Created on: 04.04.2018
 *      Author: abegic
 */

#include "include.h"
#include "init_prototyp.h"
 /*
  * send the data to the uart terminal
  */
void vals_to_terminal(void)
{

    int i = 0;
    int j = 7;
    for(i = 0 ; i<=7; i++)
    {
        for(j = 7 ; j>=0; j--)
        {
            if(j>0)
            {
                sprintf(buffer,"%4.0f␣␣␣", COSINUS_Voltages[i][j]);
                UARTSend((uint8_t*)buffer,strlen(buffer));
                SysCtlDelay(10);
            }
            else
            {
                sprintf(buffer,"%4.0f␣\n", COSINUS_Voltages[i][j]);
                UARTSend((uint8_t*)buffer,strlen(buffer));
                SysCtlDelay(10);
            }
        }

    }
    sprintf(buffer,"\nbreak\n");
    UARTSend((uint8_t*)buffer,strlen(buffer));

    for(i = 0 ; i<=7; i++)
    {
        for(j = 7 ; j>=0; j--)
        {
            if(j>0)
            {
                sprintf(buffer,"%4.0f␣␣␣", SINUS_Voltages[i][j]);
                UARTSend((uint8_t*)buffer,strlen(buffer));
                SysCtlDelay(10);
            }
            else
            {
                sprintf(buffer,"%4.0f␣\n", SINUS_Voltages[i][j]);
                UARTSend((uint8_t*)buffer,strlen(buffer));
                SysCtlDelay(10);
            }
        }

    }
    sprintf(buffer,"\nbreak\n");
    UARTSend((uint8_t*)buffer,strlen(buffer));

}
```

C.2 Matlab-Quellcode

Quellcode C.12: 360_grad Messung.m

```
%--------------------------------------------------------------------------
% Rotation bei Messdatenaufnahme
%
% Filename:       360 Grad Messung.m
% Autor:          Abraham Begic
% Datum:          30. Mai 2018
% Beschreibung:   Rotation bis zum belibigen Winkel, mehrmalige
%                 wiederholung ist moelich(z.B. Hysterese)
%
%
%--------------------------------------------------------------------------

% definition, wenn die Variable nich existiert
% if ~exist('rot_inf')
%       rot_inf = inputdlg(...
%           {'Gewuenschter Abstand zwischen den Sensoren angeben:',...
%           'Distanz zwischen den Sensoren in x-Richtung:',...
%           'Distanz zwischen den Sensoren in x-Richtung:'},...
%           [1 50; 1 50;]);
%  end;


rot     =   [0,...
              1,...
              360];

Ps00 = -14;
Ps70 =  14;
Ps05 =  6;

Pdist_maxX = abs(Ps00)+abs(Ps70);
Pdist_maxY = abs(Ps00)+abs(Ps05);


%%
datum = date; %Aktuelle datum
%Bindestrich mit Unterstrich ersetzen
ix = strfind(datum,'-');
datum(ix) = '_';
fname1=sprintf( '$360\circ$_Messung_vom_%s ',datum);
fname = fname1;
if isfolder(fname) == 0
    mkdir(fname)
end

% dxIN        = str2double(rot_inf{1});    % distance from sensor to sensor
% dyIN        = str2double(rot_inf{2});    % distance from sensor to sensor

dxIN        = 2;    % distance from sensor to sensor
dyIN        = 2;    % distance from sensor to sensor
dx          = dxIN * 100;
dy          = dyIN * 100;
wiederhol= 0;
pos_dir = 1;    % 1: x-> in positive direction step to next sensor
                %-1: y-> in negative direction step to next sensor


clear('step_x','step_y')
step_x = round(Pdist_maxX/dxIN);
```

```
step_y = round(Pdist_maxY/dyIN);


[xpoints,ypoints] = meshgrid([Ps00:Pdist_maxX/(step_x):Ps70],...
                             [Ps00:Pdist_maxY/(step_y):Ps05]);
% ypoints = meshgrid([Ps00:Pdist_maxY/(step_y):Ps05]);
% ypoints(ypoints>Ps05) = nan;

scatter(xpoints(:),ypoints(:),'b','fill'),hold on
scatter(0,0,'k','fill'),hold off
xlim([-15 15])
ylim([-15 15])
set(gca,'Xtick',[-14:4:14],...
        'Ytick',[-14:4:14])
grid on
box on
axis square
xlabel('x_a in mm -- Sensorpositions')
ylabel('y_a in mm -- Sensorpositions')

title(['Anzahl der Messungen: ',mat2str((step_x+1)*(step_y+1)),...
                    ' (blau markiert)'],'FontWeight','normal')
legend('Messpunkte','Nullpunkt')


%% Positioning to the Sensor-Point 1,1

DX = -Ps00*100;
% Bewegung in die X-Richtung zur X=1, Y=1 bezueglich Koordinatensystem
rmp_3_move_rel_unidirect_pos(   DX, ...
                                stage_setup.x_axis.module_address, ...
                                stage_setup.x_axis.motor_address, ...
                                deviceObj, ...
                                stage_setup,...
                                stage_positioning);

DY =  Ps00*100;
% Bewegung in die Y-Richtung zur X=1, Y=1 bezueglich Koordinatensystem
rmp_3_move_rel_unidirect_pos(   DY, ...
                                stage_setup.y_axis.module_address, ...
                                stage_setup.y_axis.motor_address, ...
                                deviceObj, ...
                                stage_setup,...
                                stage_positioning);
%%
q=0;
for m = 0 : step_y

    % ask for the line
    if mod(m,2) == 1
        pos_dir = -1;
    else
        pos_dir =  1;
    end

    for n = 0 : step_x
        x       = xpoints(m+1,n+1);
        y       = ypoints(m+1,n+1);

        fname2  = sprintf('%03.0fmeas_posX_%02.0f_posY_%02.0f_xa_...
                                        %02.0f_ya_%02.0f',q,n,m,x,y);
        q       = q+1;

        if isfolder(fname2) == 0
            mkdir([fname,'/',fname2])
```

```
        end
126
        dateiname = [fname,'/',fname2,'/'];
128

130
        %% set all initial variables for each measurement
132     COS_SIG = zeros(8,8); % Erzeugung einer 8x8 matrix
        SIN_SIG = zeros(8,8); % Erzeugung einer 8x8 matrix
134

136     n = 0;

138     input_alpha     = rot(1);
        alpha_degree    = rot(2) * 1.98 / 2;
140     end_alpha       = round((rot(3)/1.01012)*100)/100;
        schritt         = 0; % hilfsvariable fuer for-schleife
142     input_step      = 0;
        faktor          = 0.1; % Hilfsvariable fuer rotationsfall
144     alpha           = 0;

146     %% initializing to zero position z rotation
        %  Referenzfahrt yaw-axis (phi_z)
148        set(deviceObj, 'address' , stage_setup.yaw_axis.module_address);
           set(deviceObj, 'motor'   , stage_setup.yaw_axis.motor_address);
150     invoke(deviceObj, 'rfs'     , 'start'); % rotate right
        disp('referencing rotation-stage DMT40-D20-HSM...');
152     while(1)
            if( invoke(deviceObj,'rfs','status') == 0 )
154             break;
            end
156         pause(1);
        end
158     invoke(deviceObj, 'mst'); % stop motor
        stage_positioning.yaw_axis.initialisation_complete = 1;
160     disp('finished referencing');
        pause(1);
162     % move yaw-axis to zero position
        disp('rotating yaw-axis on zero degree orientation...');
164     rmp_3_move_abs_unidirect_pos(   stage_positioning.yaw_axis.zero_position, ...
                                        stage_setup.yaw_axis.module_address, ...
166                                     stage_setup.yaw_axis.motor_address, ...
                                        deviceObj, ...
168                                     stage_setup, ...
                                        stage_positioning);
170     %Aktuelle Positionskoordinaten uebernahme
        stage_positioning.yaw_axis.actual_position = 0;
172

        %% Auf Nullposition fahren
174

        input_step =  277*...
176                    1.98 / 2./faktor;

178     rmp_3_move_add_tilt_dist(   input_step, ...
                                    stage_setup.yaw_axis.motor_address, ...
180                                 deviceObj, ...
                                    active_wheel_hub, ...
182                                 active_collision_limit, ...
                                    global_flags, ...
184                                 stage_setup, ...
                                    stage_positioning);
186

188     %% start measurement
        if input_alpha == 0
```

```
190             rmp_3_messaufnahme;
            end
192         % die Rotation von Null bis zum Wunschswinkel
            for schritt = 0 : rot(2) :  rot(3)
194             rmp_3_move_add_tilt_dist(...
                                input_step, ...
196                             stage_setup.yaw_axis.motor_address, ...
                                deviceObj, ...
198                             active_wheel_hub, ...
                                active_collision_limit, ...
200                             global_flags, ...
                                stage_setup, ...
202                             stage_positioning);

204             %Aktuelle Positionskoordinatenuebernahme
                stage_positioning.yaw_axis.actual_position ...
206                 = stage_positioning.yaw_axis.actual_position + input_step;
                % umrechnung des eingegebenen winkel
208             input_step = alpha_degree./faktor;
                %alpha invertiert vorzeichen positiv
210             rmp_3_messaufnahme;
                n = n + 1;
212         end
            rmp_3_messaufnahme
214         wiederhol = wiederhol + 1;
            faktor = faktor * -1;
216         input_step = 0;

218         %                   step to the next column
            %       => with positive or negative dir -> pos_dir <=
220         rmp_3_move_rel_unidirect_pos(   dy*pos_dir, ...
                                            stage_setup.y_axis.module_address, ...
222                                         stage_setup.y_axis.motor_address, ...
                                            deviceObj, ...
224                                         stage_setup,...
                                            stage_positioning);
226     end
        % step to the next line
228     rmp_3_move_rel_unidirect_pos(  -dx, ...
                                       stage_setup.x_axis.module_address, ...
230                                    stage_setup.x_axis.motor_address, ...
                                       deviceObj, ...
232                                    stage_setup,...
                                       stage_positioning);
234

236 end
```

Quellcode C.13: rmp_3_mtx_rotation.m

```
%-------------------------------------------------------------------------
% 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 moeglich(z.B. Hysterese)
% Editiert:      Abraham Begic
%                Um die volle Funktionalitaet zu nutezn muessen
%                modifikationen am Quellcode vorgenommen werden
%-------------------------------------------------------------------------

% 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...');
```

```
64          rmp_3_move_abs_unidirect_pos( ...
                stage_positioning.yaw_axis.zero_position, ...
66              stage_setup.yaw_axis.module_address, ...
                stage_setup.yaw_axis.motor_address, ...
68              deviceObj, ...
                stage_setup, ...
70              stage_positioning);
            %Aktuelle Positionskoordinatenuebernahme
72          stage_positioning.yaw_axis.actual_position = 0;
        end
74
%       move=-270./faktor;
76      %   rmp_3_move_abs_unidirect_pos( ...
%               stage_positioning.yaw_axis.zero_position, ...
78      %       stage_setup.yaw_axis.module_address, ...
%               stage_setup.yaw_axis.motor_address, ...
80      %       deviceObj, ...
%               stage_setup, ...
82      %       stage_positioning);
%       disp('rotating yaw-axis to Abraham...');
84
86
88      %% die Rotation von Null bis zum Wunschswinkel
        for schritt = 0 : alpha_degree :  end_alpha
90          % move additional tilt distance
%           [active_collision_limit.add_tilt_dist] = ...
92      %       rmp_3_move_add_tilt_dist(...
%               input_step, ...
94      %       stage_setup.yaw_axis.motor_address, ...
%               deviceObj, ...
96      %       active_wheel_hub, ...
%               active_collision_limit, ...
98      %       global_flags, ...
%               stage_setup, ...
100     %       stage_positioning);
        rmp_3_move_add_tilt_dist(...
102                 input_step, ...
                    stage_setup.yaw_axis.motor_address, ...
104                 deviceObj, ...
                    active_wheel_hub, ...
106                 active_collision_limit, ...
                    global_flags, ...
108                 stage_setup, ...
                    stage_positioning);
110
        %Aktuelle Positionskoordinatenuebernahme
112         stage_positioning.yaw_axis.actual_position ...
                = stage_positioning.yaw_axis.actual_position + input_step;
114         % umrechnung des eingegebenen winkel
            input_step = alpha_degree./faktor;
116         %alpha invertiert vorzeichen war positiv
            %-----------------
118         rmp_3_messaufnahme;
            n = n + 1;
120     end
        rmp_3_messaufnahme
122 wiederhol = wiederhol + 1;
    faktor = faktor * -1;
124 input_step = 0;
    end
126 end
```

Quellcode C.14: rmp_3_menu_meas_save.m

```
%--------------------------------------------------------------------------
% 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 das Array', ...% korrektur
        'Rotation mit einem einzelnen Sensor',...
        'Messprotokollerstellung', ...
        'Modus-Auswahl Menu', ...
        '$360\circ$ Rotationsmessung',...
        '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
```

```
64              clear('dat_name', 'n');

66          case 5
                clear input_meas; % clear input variable
68              rmp_3_messprotokoll; % Messprotokollerstellung

70          case 6
                clear input_position; % clear input variable
72              rmp_3_menu_modusauswahl; %aufruf menu modusaswahl

74          case 7
%                clear ('input_meas', 'rot_inf', 'si',...
76  %                   'matrix_rot','dat_name'); % clear input variable

78              %rmp_3_datei_erstellung;
               rmp_360_grad_Messung; % aufruf Messdatenaufnahme
80                                      % bei Translation mit/ohne Rotation
                %rmp_3_messprotokoll; % Messprotokollerstellung
82
             case 8
84               save('rmp_3_init_stage_pos','active_collision_limit',...
                      'active_motor','active_wheel_hub', 'collision_limit',...
86                    'global_flags','global_flags','laenge_x','laenge_y',...
                      'laenge_z','motor_setup','specific_parameter',...
88                    'stage_positioning','stage_setup','wheel_hub',...
                      'work');
90              clear input_meas; % clear input variable
                disp('exit');
92              break; % exit while loop

94

96

98

100

102     end;
    end;
```

Quellcode C.15: rmp_3_init_stage_system.m

```
%-------------------------------------------------------------------------
% BASIC INITIALISATION OF THE STAGE SYSTEM
%
% version 0.1
% filename:     rmp_3_init_stage_system.m
%-------------------------------------------------------------------------
%
% author :      Christian Schoermer
% date   :      21.04.2010
% changes:      Viktor Airich
%               03.11.2017
%               Abraham Begic, Thorben Schuethe
%               17.05.18
% description:  This script initializes the structures "stage_positioning",
%               "wheel_hub", "active_wheel_hub", "collision_limit",
%               "active_collision_limit", "active_motor" and
%               "specific_parameter".
%               The parameters in "stage_positioning" are used for
%               several positioning. Furthermore informations about
%               the wheel hubs, like reference positions on
%               encoder wheel and active flags are deposited in the
%               structure "wheel_hub".
%               To avoid collisions between positioning stage and
%               wheel hub, a structure called "collision_limit" is
%               built up. In active mode (measuring station is in use)
%               parameters of one choosen wheel hub are transfered into
%               an active structure, called "active_wheel_hub". The
%               collision information about the choosen wheel hub is
%               transfered into the structure "active_collision_limit".
%               Specific positions structure provides specific stage
%               positions(e.g. absolute position for sensor change)
%               measured by a meter.
%
% precondition: initialisation of the structure "stage_setup".
%
%
%-------------------------------------------------------------------------



%% positioning structure
% positioning structure for every positioning stage
% the positioning structure includes information about:
%
%           * stage identification
%           * bi-repeatibility error in dependency of fullstep divisor
%           * stepsize resolution of hardware [m]
%           * spindle pitch (only linear axes)
%           * fullsteps per motor revolution (hardware)
%           * reduction of motor and gear ratio
%           * motor steps per revolution in dependency of fullstep divisor
%           * difference between limit switches (not DMT65 and DMT40)
%           * initial (linear axes )and zero position (rot. stages: no tilt)
%           * different positioning speed values
%           * information flag: initialisation complete
%           * positioning limits, to avoid collisions with the wheel hub
%           * acutal position: difference between reference position on
%             encoder wheel and sensor
%           * reversal error compensation factor for additional numbers
%             of stepsize to guarantee reversal error compensation
%           * finsal reversal error compensation factor
%           * absolute orientation of alignment rotation stage (DMT65)
%             for sensor alignment before encoder (absolute position)
```

```
64  %           * actual tilt distance to avoid collisions between sensor
    %             or rather sensor socket and wheel hub [m]
66  %           * initialisation completed flag

68  stage_positioning =  struct( ...
        'x_axis',          struct( ...
70                         'stage_identification',      'LTM80-150-HSM' , ...
                           'bi_repeatibility_error',         15*10^(-6) , ...
72                         'stepsize',                       10*10^(-6) , ...
                           'spindle_pitch',                   1*10^(-3) , ...
74                         'mot_fullsteps_per_rev',                200 , ...
                           'low_positioning_speed',                 40 , ...
76                         'high_positioning_speed',               400 , ...
                           'mot_fullstep_divisor',                   0 , ...
78                         'mot_steps_per_rev',                      0 , ...
                           'distance_per_step',                      0 , ...
80                         'steps_per_stepsize',                     0 , ...
                           'steps_bi_repeatibility_error',           0 , ...
82                         'steps_distance_end_switches',            0 , ...
                           'inital_position',                        0 , ...
84                         'collision_limit',                        0 , ...
                           'actual_position',                       -1 , ...
86                         'reversal_err_comp_factor',               2 , ...
                           'reversal_err_comp',                      0 , ...
88                         'initialisation_complete',                0), ...
        'y_axis',          struct( ...
90                         'stage_identification',      'LTM80- 75-HSM' , ...
                           'bi_repeatibility_error',         15*10^(-6) , ...
92                         'stepsize',                       10*10^(-6) , ...
                           'spindle_pitch',                   1*10^(-3) , ...
94                         'mot_fullsteps_per_rev',                200 , ...
                           'low_positioning_speed',                 40 , ...
96                         'high_positioning_speed',               400 , ...
                           'mot_fullstep_divisor',                   0 , ...
98                         'mot_steps_per_rev',                      0 , ...
                           'distance_per_step',                      0 , ...
100                        'steps_per_stepsize',                     0 , ...
                           'steps_bi_repeatibility_error',           0 , ...
102                        'steps_distance_end_switches',            0 , ...
                           'initial_position',                       0 , ...
104                        'collision_limit',                        0 , ...
                           'actual_position',                       -1 , ...
106                        'reversal_err_comp_factor',               2 , ...
                           'reversal_err_comp',                      0 , ...
108                        'initialisation_complete',                0) , ...
        'z_axis',          struct( ...
110                        'stage_identification',      'LTM80-150-HSM' , ...
                           'bi_repeatibility_error',         15*10^(-6) , ...
112                        'stepsize',                       10*10^(-6) , ...
                           'spindle_pitch',                   1*10^(-3) , ...
114                        'mot_fullsteps_per_rev',                200 , ...
                           'low_positioning_speed',                 40 , ...
116                        'high_positioning_speed',               400 , ...
                           'mot_fullstep_divisor',                   0 , ...
118                        'mot_steps_per_rev',                      0 , ...
                           'distance_per_step',                      0 , ...
120                        'steps_per_stepsize',                     0 , ...
                           'steps_bi_repeatibility_error',           0 , ...
122                        'steps_distance_end_switches',            0 , ...
                           'initial_position',                       0 , ...
124                        'actual_position',                       -1 , ...
                           'reversal_err_comp_factor',               2 , ...
126                        'reversal_err_comp',                      0 , ...
                           'initialisation_complete',                0), ...
128     'alignment_axis',  struct( ...
```

```
                    'stage_identification',      'DMT65-DM4-HSM' , ...
130                 'bi_repeatibility_error',              0.02 , ...
                    'stepsize',                             0.1 , ...
132                 'reduction',                            180 , ...
                    'mot_fullsteps_per_rev',                200 , ...
134                 'mot_fullstep_divisor',                   0 , ...
                    'mot_steps_per_rev',                      0 , ...
136                 'degree_per_step',                        0 , ...
                    'steps_per_stepsize',                     0 , ...
138                 'steps_bi_repeatibility_error',           0 , ...
                    'initial_position',                       0 , ...
140                 'actual_position',                       -1 , ...
                    'reversal_err_comp_factor',               2 , ...
142                 'reversal_err_comp',                      0 , ...
                    'axial_alignment',                        0 , ...
144                 'radial_alignment',                       0 , ...
                    'initialisation_complete',                0), ...
146     'yaw_axis',     struct( ...
                    'stage_identification',      'DMT40-D20-HSM' , ...
148                 'bi_repeatibility_error',             0.07 , ...
                    'stepsize',                             0.1 , ...
150                 'reduction',                       6917.7669 , ...
                    'mot_fullsteps_per_rev',                 24 , ...
152                 'degree_tilt_limit',                      0, ...
                   %changed vom 8 to zero
154                 'mot_fullstep_divisor',                   0 , ...
                    'mot_steps_per_rev',                      0 , ...
156                 'degree_per_step',                        0 , ...
                    'steps_per_stepsize',                     0 , ...
158                 'steps_bi_repeatibility_error',           0 , ...
                    'steps_distance_end_switches',            0 , ...
160                 'tilt_limit',                             0 , ...
                   % changed 2018-05-17 -> default 8 to 0
162                 'zero_position',                          0 , ...
                    'actual_position',                       -1 , ...
164                 'reversal_err_comp_factor',               2 , ...
                    'reversal_err_comp',                      0 , ...
166                 'initialisation_complete',                0), ...
    'roll_axis',    struct( ...
168                 'stage_identification',     'MOGO65-W16-HSM' , ...
                    'bi_repeatibility_error',               0.2 , ...
170                 'stepsize',                             0.2 , ...
                    'reduction',                          55040 , ...
172                 'mot_fullsteps_per_rev',                 20 , ...
                    'mot_fullstep_divisor',                   0 , ...
174                 'mot_steps_per_rev',                      0 , ...
                    'degree_per_step',                        0 , ...
176                 'steps_per_stepsize',                     0 , ...
                    'steps_bi_repeatibility_error',           0 , ...
178                 'steps_distance_end_switches',            0 , ...
                    'tilt_limit',                             0 , ...
180                 'zero_position',                          0 , ...
                    'actual_position',                       -1 , ...
182                 'reversal_err_comp_factor',               2 , ...
                    'reversal_err_comp',                      0 , ...
184                 'initialisation_complete',                0), ...
    'pitch_axis',     struct( ...
186                 'stage_identification',     'MOGO40-W16-HSM' , ...
                    'bi_repeatibility_error',               0.2 , ...
188                 'stepsize',                             0.2 , ...
                    'reduction',                          32000 , ...
190                 'mot_fullsteps_per_rev',                 20 , ...
                    'mot_fullstep_divisor',                   0 , ...
192                 'mot_steps_per_rev',                      0 , ...
                    'degree_per_step',                        0 , ...
```

```
                        'steps_per_stepsize',                   0 , ...
                        'steps_bi_repeatibility_error',         0 , ...
                        'steps_distance_end_switches',          0 , ...
                        'tilt_limit',                           8 , ...
                        'zero_position',                        0 , ...
                        'actual_position',                     -1 , ...
                        'reversal_err_comp_factor',             2 , ...
                        'reversal_err_comp',                    0 , ...
                        'initialisation_complete',              0) ...
                        );

%% global flag structure
% wheel hub structure provides information about the states of the
% measuring station

global_flags = struct( ...
                        'initialisation_complete_flag',           0 , ...
                        'active_wheel_hub_flag',                  0);

%% wheel hub structure
% wheel hub structure provides information about:
%
%           * coded value for active wheel hub
%             no active wheel hub = 0
%             Golf IV             = 1
%             Golf  V             = 2
%             BMW 3er             = 3
%           * encoder orientation:
%             axial               = 1
%             radial              = 2
%             ATTENTION: wrong encoder orientation causes
%                        collisions!
%           * tooth sum of encoder wheel
%           * reference offset [m] above encoder wheel
%             and required steps
%           * absolute position of x- ,y- and z-linear stage
%             calculated in the function "ref_pos_encoder".
%             Initial values are "steps_distance_end_switches"
%             to avoid collision

wheel_hub = struct( ...
    'Golf_IV',          struct( ...
                        'identification',                 'GOLF_IV' , ...
                        'encoder_orientation',                   1 , ...
                        'tooth_sum',                            43 , ...
                        'reference_offset',              0.2*10^(-3), ...
                        'steps_reference_offset',                0 , ...
                        'threshold_speed_change',         5*10^(-3), ...
                        'steps_threshold_speed_change',          0 , ...
                        'x_ref_pos',                             0 , ...
                        'y_ref_pos',                             0 , ...
                        'z_ref_pos',                             0), ...
    'Golf_V',           struct( ...
                        'identification',                 'GOLF_V' , ...
                        'encoder_orientation',                   2 , ...
                        'tooth_sum',                            43 , ...
                        'reference_offset',              0.2*10^-3 , ...
                        'steps_reference_offset',                0 , ...
                        'threshold_speed_change',         5*10^(-3), ...
                        'steps_threshold_speed_change',          0 , ...
                        'x_ref_pos',                             0 , ...
                        'y_ref_pos',                             0 , ...
                        'z_ref_pos',                             0), ...
    'BMW_3er',          struct( ...
                        'identification',                 'BMW_3er' , ...
```

```
                        'encoder_orientation',                  1 , ...
260                     'tooth_sum',                           48 , ...
                        'reference_offset',              0.2*10^-3 , ...
262                     'steps_reference_offset',               0 , ...
                        'threshold_speed_change',        5*10^(-3), ...
264                     'steps_threshold_speed_change',         0 , ...
                        'x_ref_pos',                            0 , ...
266                     'y_ref_pos',                            0 , ...
                        'z_ref_pos',                            0) ...
268                     );

270 %% collision limit structure
    % collision limit structure provides information about:
272 %
    %           * active flag, when collision protection is activated
274 %           * radius of the collsion sphere [m]
    %           * absolute position of the collision sphere center
276 %             in x- ,y- and z-coordinates (absolute positions of
    %             linear stages)
278
    collision_limit = struct( ...
280     'Golf_IV',          struct( ...
                        'identification',                'GOLF_IV' , ...
282                     'collision_flag',                       0 , ...
                        'add_tilt_dist',                        0 , ...
284                     'steps_add_tilt_dist',                  0 , ...
                        'collision_radius',              8*10^-3 , ...
286                     'x_pos_sphere_center',                  0 , ...
                        'y_pos_sphere_center',                  0 , ...
288                     'z_pos_sphere_center',                  0), ...
        'Golf_V',           struct( ...
290                     'identification',                'GOLF_V' , ...
                        'collision_flag',                       0 , ...
292                     'add_tilt_dist',                        0 , ...
                        'steps_add_tilt_dist',                  0 , ...
294                     'collision_radius',              25*10^-3 , ...
                        'x_pos_sphere_center',                  0 , ...
296                     'y_pos_sphere_center',                  0 , ...
                        'z_pos_sphere_center',                  0), ...
298     'BMW_3er',          struct( ...
                        'identification',                'BMW_3er' , ...
300                     'collision_flag',                       0 , ...
                        'add_tilt_dist',                        0 , ...
302                     'steps_add_tilt_dist',                  0 , ...
                        'collision_radius',              20*10^-3 , ...
304                     'x_pos_sphere_center',                  0 , ...
                        'y_pos_sphere_center',                  0 , ...
306                     'z_pos_sphere_center',                  0) ...
                        );
308
    %% active wheel hub structure
310 % wheel hub structure provides information about the active wheel hub

312 active_wheel_hub = struct( ...
                        'identification',                     '-' , ...
314                     'encoder_orientation',                  0 , ...
                        'tooth_sum',                            0 , ...
316                     'reference_offset',                     0 , ...
                        'steps_reference_offset',               0 , ...
318                     'threshold_speed_change',               0 , ...
                        'steps_threshold_speed_change',         0 , ...
320                     'x_ref_pos',                            0 , ...
                        'y_ref_pos',                            0 , ...
322                     'z_ref_pos',                            0);
```

```
324
%% active collision limit structure
326 % collision limit structure provides information about:
% colllision limits of the active wheel hub
328
active_collision_limit = struct( ...
330                         'identification',                       '-' , ...
                            'collision_flag',                         0 , ...
332                         'add_tilt_dist',                          0 , ...
                            'steps_add_tilt_dist',                    0 , ...
334                         'collision_radius',                       0 , ...
                            'x_pos_sphere_center',                    0 , ...
336                         'y_pos_sphere_center',                    0 , ...
                            'z_pos_sphere_center',                    0);
338
 %% active motor structure
340 % active motor structure provides information about the active motor

342 active_motor = struct( ...
                            'identification',                       '-' , ...
344                         'module_address',                         0 , ...
                            'microstep_resolution',                   0 , ...
346                         'maximum_acceleration',                   0 , ...
                            'absolute_max_current',                   0 , ...
348                         'gear_ratio',                             0 , ...
                            'actual_speed',                           0);
350


352
%% specific parameter structure
354 % specific parameter structure provides information about:
%
356 %           * specific absolute positions (e.g. DMT65:sensor change
%             position measured by meter and an adjusted fullstep
358 %             divisor. The ending of the variables contain
%             information about the fullstep divisor:
360 %             e.g. div_4 -> absolute position measured with a fullstep
%             divisor of 4.
362

364 specific_parameter =  struct( ...
    'x_axis',            struct( ...
366                         'stage_identification',       'LTM80-150-HSM'), ...
    'y_axis',            struct( ...
368                         'stage_identification',       'LTM80- 75-HSM'), ...
    'z_axis',            struct( ...
370                         'stage_identification',       'LTM80-150-HSM'), ...
    'alignment_axis',    struct( ...
372                         'stage_identification',       'DMT65-DM4-HSM' , ...
                            'steps_measured_initial_div_4',         72200 , ...
374                         'steps_measured_radial_div_4',           -440 , ...
                            'steps_measured_axial_div_4',           35400), ...
376     'yaw_axis',         struct( ...
                            'stage_identification',       'DMT40-D20-HSM' , ...
378                         'steps_measured_zero_div_2',            74600), ...
                           % wieso nicht bei 75000
380     'roll_axis',        struct( ...
                            'stage_identification',      'MOGO65-W16-HSM' , ...
382                         'steps_measured_zero_div_4',           119600), ...
    'pitch_axis',             struct( ...
384                         'stage_identification',      'MOGO40-W16-HSM' , ...
                            'steps_measured_zero_div_4',            60421) ...
386                         );

388
```

```
390
392 %% set motor parameters
    % calculate fullstep divisor
394
    % x-axis
396 stage_positioning.x_axis.mot_fullstep_divisor = ...
        2^(stage_setup.x_axis.microstep_resolution);
398
    % y-axis
400 stage_positioning.y_axis.mot_fullstep_divisor = ...
        2^(stage_setup.y_axis.microstep_resolution);
402
    % z-axis
404 stage_positioning.z_axis.mot_fullstep_divisor = ...
        2^(stage_setup.z_axis.microstep_resolution);
406
    % alignment-axis
408 stage_positioning.alignment_axis.mot_fullstep_divisor = ...
        2^(stage_setup.alignment_axis.microstep_resolution);
410
    % yaw-axis
412 stage_positioning.yaw_axis.mot_fullstep_divisor = ...
        2^(stage_setup.yaw_axis.microstep_resolution);
414
    % roll-axis
416 stage_positioning.roll_axis.mot_fullstep_divisor = ...
        2^(stage_setup.roll_axis.microstep_resolution);
418
    % pitch-axis
420 stage_positioning.pitch_axis.mot_fullstep_divisor = ...
        2^(stage_setup.pitch_axis.microstep_resolution);
422
424 %% calculate steps per revolution in dependency on number of fullsteps and
    % fullstep divisor
426
    % x-axis
428 stage_positioning.x_axis.mot_steps_per_rev = ...
        stage_positioning.x_axis.mot_fullsteps_per_rev * ...
430     stage_positioning.x_axis.mot_fullstep_divisor;

432 % y-axis
    stage_positioning.y_axis.mot_steps_per_rev = ...
434     stage_positioning.y_axis.mot_fullsteps_per_rev * ...
        stage_positioning.y_axis.mot_fullstep_divisor;
436
    % z-axis
438 stage_positioning.z_axis.mot_steps_per_rev = ...
        stage_positioning.z_axis.mot_fullsteps_per_rev * ...
440     stage_positioning.z_axis.mot_fullstep_divisor;

442 % alignment-axis
    stage_positioning.alignment_axis.mot_steps_per_rev = ...
444     stage_positioning.alignment_axis.mot_fullsteps_per_rev * ...
        stage_positioning.alignment_axis.mot_fullstep_divisor;
446
    % roll-axis
448 stage_positioning.yaw_axis.mot_steps_per_rev = ...
        stage_positioning.yaw_axis.mot_fullsteps_per_rev * ...
450     stage_positioning.yaw_axis.mot_fullstep_divisor;

452 % pitch-axis
    stage_positioning.roll_axis.mot_steps_per_rev = ...
```

```
454     stage_positioning.roll_axis.mot_fullsteps_per_rev * ...
        stage_positioning.roll_axis.mot_fullstep_divisor;
456
    % yaw-axis
458 stage_positioning.pitch_axis.mot_steps_per_rev = ...
        stage_positioning.pitch_axis.mot_fullsteps_per_rev * ...
460     stage_positioning.pitch_axis.mot_fullstep_divisor;

462
    %% calculate distance per step in dependency on reduction
464
    % x-axis
466 stage_positioning.x_axis.distance_per_step = ...
        stage_positioning.x_axis.spindle_pitch / ...
468     stage_positioning.x_axis.mot_steps_per_rev;

470 % y-axis
    stage_positioning.y_axis.distance_per_step = ...
472     stage_positioning.y_axis.spindle_pitch / ...
        stage_positioning.y_axis.mot_steps_per_rev;
474
    % z-axis
476 stage_positioning.z_axis.distance_per_step = ...
        stage_positioning.z_axis.spindle_pitch / ...
478     stage_positioning.z_axis.mot_steps_per_rev;

480 % alignment-axis
    stage_positioning.alignment_axis.degree_per_step = ...
482     360 / (stage_positioning.alignment_axis.mot_steps_per_rev * ...
        stage_positioning.alignment_axis.reduction);
484
    % roll-axis
486 stage_positioning.yaw_axis.degree_per_step = ...
        360 / (stage_positioning.yaw_axis.mot_steps_per_rev * ...
488     stage_positioning.yaw_axis.reduction);

490 % pitch-axis
    stage_positioning.roll_axis.degree_per_step = ...
492     360 / (stage_positioning.roll_axis.mot_steps_per_rev * ...
        stage_positioning.roll_axis.reduction);
494
    % yaw-axis
496 stage_positioning.pitch_axis.degree_per_step = ...
        360 / (stage_positioning.pitch_axis.mot_steps_per_rev * ...
498     stage_positioning.pitch_axis.reduction);

500
    %% calculate bi-directional error in steps
502
    % x-axis
504 stage_positioning.x_axis.steps_bi_repeatibility_error = ...
        stage_positioning.x_axis.bi_repeatibility_error / ...
506     stage_positioning.x_axis.distance_per_step;

508 stage_positioning.x_axis.steps_bi_repeatibility_error = round(...
        stage_positioning.x_axis.steps_bi_repeatibility_error); % round
510
    % y-axis
512 stage_positioning.y_axis.steps_bi_repeatibility_error = ...
        stage_positioning.y_axis.bi_repeatibility_error / ...
514     stage_positioning.y_axis.distance_per_step;

516 stage_positioning.y_axis.steps_bi_repeatibility_error = round(...
        stage_positioning.y_axis.steps_bi_repeatibility_error); % round
518
```

```
% z-axis
stage_positioning.z_axis.steps_bi_repeatibility_error = ...
    stage_positioning.z_axis.bi_repeatibility_error / ...
    stage_positioning.z_axis.distance_per_step;

stage_positioning.z_axis.steps_bi_repeatibility_error = round(...
    stage_positioning.z_axis.steps_bi_repeatibility_error); % round

% alignment-axis
stage_positioning.alignment_axis.steps_bi_repeatibility_error = ...
    stage_positioning.alignment_axis.bi_repeatibility_error / ...
    stage_positioning.alignment_axis.degree_per_step;

stage_positioning.alignment_axis.steps_bi_repeatibility_error = round(...
    stage_positioning.alignment_axis.steps_bi_repeatibility_error); % round

% roll-axis
stage_positioning.yaw_axis.steps_bi_repeatibility_error = ...
    stage_positioning.yaw_axis.bi_repeatibility_error / ...
    stage_positioning.yaw_axis.degree_per_step;

stage_positioning.yaw_axis.steps_bi_repeatibility_error = round(...
    stage_positioning.yaw_axis.steps_bi_repeatibility_error); % round

% pitch-axis
stage_positioning.roll_axis.steps_bi_repeatibility_error = ...
    stage_positioning.roll_axis.bi_repeatibility_error / ...
    stage_positioning.roll_axis.degree_per_step;

stage_positioning.roll_axis.steps_bi_repeatibility_error = round(...
    stage_positioning.roll_axis.steps_bi_repeatibility_error); % round

% yaw-axis
stage_positioning.pitch_axis.steps_bi_repeatibility_error = ...
    stage_positioning.pitch_axis.bi_repeatibility_error / ...
    stage_positioning.pitch_axis.degree_per_step;

stage_positioning.pitch_axis.steps_bi_repeatibility_error = round(...
    stage_positioning.pitch_axis.steps_bi_repeatibility_error); % round


%% calculate step size in "steps" for all positioning stages

% x-axis
stage_positioning.x_axis.steps_per_stepsize = ...
    stage_positioning.x_axis.stepsize / ...
    stage_positioning.x_axis.distance_per_step;

stage_positioning.x_axis.steps_per_stepsize = round(...
    stage_positioning.x_axis.steps_per_stepsize); % round

% y-axis
stage_positioning.y_axis.steps_per_stepsize = ...
    stage_positioning.y_axis.stepsize / ...
    stage_positioning.y_axis.distance_per_step;

stage_positioning.y_axis.steps_per_stepsize = round(...
    stage_positioning.y_axis.steps_per_stepsize); % round

% z-axis
stage_positioning.z_axis.steps_per_stepsize = ...
    stage_positioning.z_axis.stepsize / ...
    stage_positioning.z_axis.distance_per_step;

stage_positioning.z_axis.steps_per_stepsize = round(...
```

```
584     stage_positioning.z_axis.steps_per_stepsize); % round

586 % alignment-axis
    stage_positioning.alignment_axis.steps_per_stepsize = ...
588     stage_positioning.alignment_axis.stepsize / ...
        stage_positioning.alignment_axis.degree_per_step;
590
    stage_positioning.alignment_axis.steps_per_stepsize = round(...
592     stage_positioning.alignment_axis.steps_per_stepsize); % round

594 % roll-axis
    stage_positioning.yaw_axis.steps_per_stepsize = ...
596     stage_positioning.yaw_axis.stepsize / ...
        stage_positioning.yaw_axis.degree_per_step;
598
    stage_positioning.yaw_axis.steps_per_stepsize = round(...
600     stage_positioning.yaw_axis.steps_per_stepsize); % round

602 % pitch-axis
    stage_positioning.roll_axis.steps_per_stepsize = ...
604     stage_positioning.roll_axis.stepsize / ...
        stage_positioning.roll_axis.degree_per_step;
606
    stage_positioning.roll_axis.steps_per_stepsize = round(...
608     stage_positioning.roll_axis.steps_per_stepsize); % round

610 % yaw-axis
    stage_positioning.pitch_axis.steps_per_stepsize = ...
612     stage_positioning.pitch_axis.stepsize / ...
        stage_positioning.pitch_axis.degree_per_step;
614
    stage_positioning.pitch_axis.steps_per_stepsize = round(...
616     stage_positioning.pitch_axis.steps_per_stepsize); % round

618
    %% calculate bi-directional error compensation steps
620
    % x-axis
622 stage_positioning.x_axis.reversal_err_comp = ...
        stage_positioning.x_axis.steps_bi_repeatibility_error /...
624     stage_positioning.x_axis.steps_per_stepsize + ...
        stage_positioning.x_axis.reversal_err_comp_factor;
626
    stage_positioning.x_axis.reversal_err_comp = round(...
628     stage_positioning.x_axis.reversal_err_comp); % round

630 % y-axis
    stage_positioning.y_axis.reversal_err_comp = ...
632     stage_positioning.y_axis.steps_bi_repeatibility_error /...
        stage_positioning.y_axis.steps_per_stepsize + ...
634     stage_positioning.y_axis.reversal_err_comp_factor;

636 stage_positioning.y_axis.reversal_err_comp = round(...
        stage_positioning.y_axis.reversal_err_comp); % round
638
    % z-axis
640 stage_positioning.z_axis.reversal_err_comp = ...
        stage_positioning.z_axis.steps_bi_repeatibility_error /...
642     stage_positioning.z_axis.steps_per_stepsize + ...
        stage_positioning.z_axis.reversal_err_comp_factor;
644
    stage_positioning.z_axis.reversal_err_comp = round(...
646     stage_positioning.z_axis.reversal_err_comp); % round

648 % alignment-axis
```

```
stage_positioning.alignment_axis.reversal_err_comp = ...
    stage_positioning.alignment_axis.steps_bi_repeatibility_error /...
    stage_positioning.alignment_axis.steps_per_stepsize + ...
    stage_positioning.alignment_axis.reversal_err_comp_factor;

stage_positioning.alignment_axis.reversal_err_comp = round(...
    stage_positioning.alignment_axis.reversal_err_comp); % round

% roll-axis
stage_positioning.yaw_axis.reversal_err_comp = ...
    stage_positioning.yaw_axis.steps_bi_repeatibility_error /...
    stage_positioning.yaw_axis.steps_per_stepsize + ...
    stage_positioning.yaw_axis.reversal_err_comp_factor;

stage_positioning.yaw_axis.reversal_err_comp = round(...
    stage_positioning.yaw_axis.reversal_err_comp); % round

% pitch-axis
stage_positioning.roll_axis.reversal_err_comp = ...
    stage_positioning.roll_axis.steps_bi_repeatibility_error /...
    stage_positioning.roll_axis.steps_per_stepsize + ...
    stage_positioning.roll_axis.reversal_err_comp_factor;

stage_positioning.roll_axis.reversal_err_comp = round(...
    stage_positioning.roll_axis.reversal_err_comp); % round

% yaw-axis
stage_positioning.pitch_axis.reversal_err_comp = ...
    stage_positioning.pitch_axis.steps_bi_repeatibility_error /...
    stage_positioning.pitch_axis.steps_per_stepsize + ...
    stage_positioning.pitch_axis.reversal_err_comp_factor;

stage_positioning.pitch_axis.reversal_err_comp = round(...
    stage_positioning.pitch_axis.reversal_err_comp); % round


%% initial position of alignment axis (rotation stage DMT65-DM4-HSM)

% rotate stage to initial for sensor change
if(stage_setup.alignment_axis.microstep_resolution == 2)
    stage_positioning.alignment_axis.initial_position = ...
        specific_parameter.alignment_axis.steps_measured_initial_div_4;
else
    % throw exception
    ME = MException('VerifyOutput:OutOfBounds', ...
        'Microstep resolution does not match with positioning steps');
    throw(ME);
end;


%% alignment orientation on different encoder wheels

% radial sensor alignment
if(stage_setup.alignment_axis.microstep_resolution == 2) % fullstep div = 4
    stage_positioning.alignment_axis.radial_alignment = ...
        specific_parameter.alignment_axis.steps_measured_radial_div_4;
else
    % throw exception
    ME = MException('VerifyOutput:OutOfBounds', ...
        'Microstep resolution does not match with positioning steps');
    throw(ME);
end;

% axial sensor alignment
if(stage_setup.alignment_axis.microstep_resolution == 2) % fullstep div = 4
```

```
714     stage_positioning.alignment_axis.axial_alignment = ...
            specific_parameter.alignment_axis.steps_measured_axial_div_4;
716 else
    % throw exception
718     ME = MException('VerifyOutput:OutOfBounds', ...
            'Microstep resolution does not match with positioning steps');
720     throw(ME);
    end;
722
    %% limit positions in tilt operation to avoid collision with wheel hub
724
    % DMT40 limit tilt position
726 stage_positioning.yaw_axis.steps_tilt_limit = ...
        stage_positioning.yaw_axis.degree_tilt_limit/ ...
728     stage_positioning.yaw_axis.degree_per_step;

730 %% zero positions of rotation stages

732 % yaw-axis DMT40
    if(stage_setup.yaw_axis.microstep_resolution == 1)
734     stage_positioning.yaw_axis.zero_position = ...
            specific_parameter.yaw_axis.steps_measured_zero_div_2;
736 else
    % throw exception
738     ME = MException('VerifyOutput:OutOfBounds', ...
            'Microstep resolution does not match with positioning steps');
740     throw(ME);
    end;
```

Quellcode C.16: rmp_3_move_relative_position.m

```
function [] = rmp_3_move_relative_position( ...
    input_rel_pos,...
    deviceAddr, ...
    motorAddr, ...
    deviceObj, ...
    stage_setup, ...
    stage_positioning)
%--------------------------------------------------------------------------
% MOVE POSITIONING STAGE TO RELATIVE POSITION
%
% version 0.1
% filename:     rmp_3_move_relative_position.m
%--------------------------------------------------------------------------
%
% author :      Christian Schoermer
% date   :      29.04.2010
% changes:      Abraham Begic
% date:         23.05.18
%
% input:        input_rel_pos                   : relative position in
%                                                 number of stepsize
%               moduleAddr                      : module address
%               motorAddr                       : motor address
%               deviceObj                       : actual device parameter
%               stage_setup                     : stage setup structure
%               stage_positioning               : stage positioning
%                                                 structure
%
% output:       -
%
% description:  function moves stage to a user defined relative
%               position. Step increment is definded in the structure
%               "stage_positioning".
%
%
%--------------------------------------------------------------------------

% allocate variable

switch deviceAddr % switch modules

    case 1 % linear axes

        switch motorAddr % switch motor

            case stage_setup.x_axis.motor_address % x-axis

                set(deviceObj,'address',...
                    stage_setup.x_axis.module_address); % module address
                set(deviceObj,'motor', ...
                    stage_setup.x_axis.motor_address); % motor address

                %                 disp('positioning x-axis...');

                % relative positioning
                invoke(deviceObj, 'mvp','relative',...
                    stage_positioning.x_axis.steps_per_stepsize *...
                    input_rel_pos);

                % wait till stage reached position
                while(1)
                    if((get(deviceObj.Axis,'target_pos_reached') == 1) ||...
                            (get(deviceObj.Axis,'left_limit_switch_status') == 1)...
```

```
64                      || (get(deviceObj.Axis,'right_limit_switch_status') == 1))
                        break;
66                  end;
                    pause(1);
68              end

70              invoke(deviceObj, 'mst'); % stop motor

72              %               disp('positioning finished');

74          case stage_setup.y_axis.motor_address % y-axis

76              set(deviceObj,'address',...
                    stage_setup.y_axis.module_address); % module address
78              set(deviceObj,'motor', ...
                    stage_setup.y_axis.motor_address); % motor address
80
                %               disp('positioning y-axis...');
82
                % relative positioning
84              invoke(deviceObj, 'mvp','relative',...
                    stage_positioning.y_axis.steps_per_stepsize *...
86                  input_rel_pos)

88              % wait till stage reached position
                while(1)
90                  if((get(deviceObj.Axis,'target_pos_reached') == 1) ||...
                            (get(deviceObj.Axis,'left_limit_switch_status') == 1)...
92                  || (get(deviceObj.Axis,'right_limit_switch_status') == 1))
                        break;
94                  end;
                    pause(1);
96              end

98              invoke(deviceObj, 'mst'); % stop motor

100             %               disp('positioning finished');

102         case stage_setup.z_axis.motor_address % z-axis

104             set(deviceObj,'address', ...
                    stage_setup.z_axis.module_address); % module address
106             set(deviceObj,'motor', ...
                    stage_setup.z_axis.motor_address); % motor address
108
                %               disp('positioning z-axis...');
110
                % relative positioning
112             invoke(deviceObj, 'mvp','relative',...
                    stage_positioning.z_axis.steps_per_stepsize *...
114                 input_rel_pos)

116             % wait till stage reached position
                while(1)
118                 if((get(deviceObj.Axis,'target_pos_reached') == 1) ||...
                            (get(deviceObj.Axis,'left_limit_switch_status') == 1)...
120                 || (get(deviceObj.Axis,'right_limit_switch_status') == 1))
                        break;
122                 end;
                    pause(1);
124             end

126             invoke(deviceObj, 'mst'); % stop motor

128             %               disp('positioning finished');
```

```
130             otherwise

132                 % throw exception
                    ME = MException('VerifyOutput:OutOfBounds', ...
134                      'No valid motor address');
                    throw(ME);
136         end;

138     case 2 % tilt axes

140         switch motorAddr

142             case stage_setup.yaw_axis.motor_address % yaw-axis

144                 set(deviceObj,'address', ...
                        stage_setup.yaw_axis.module_address); % module address
146                 set(deviceObj,'motor', ...
                        stage_setup.yaw_axis.motor_address); % motor address
148
                    %                   disp('positioning yaw-axis...');
150
                    % relative positioning
152                 invoke(deviceObj, 'mvp','relative',...
                        stage_positioning.yaw_axis.steps_per_stepsize *...
154                     (-input_rel_pos));

156                 % wait till stage reached position
                    while(1)
158                     % zweite bedingung fuert zur unterbrechung der messung
                        if((get(deviceObj.Axis,'target_pos_reached') == 1))%||...
160                       %(get(deviceObj.Axis,'left_limit_switch_status') == 1))
                            break;
162                     end;
                        pause(1);
164                 end

166                 invoke(deviceObj, 'mst'); % stop motor

168                 %                   disp('positioning finished');

170             case stage_setup.roll_axis.motor_address % roll-axis

172                 set(deviceObj,'address', ...
                        stage_setup.roll_axis.module_address); % module address
174                 set(deviceObj,'motor', ...
                        stage_setup.roll_axis.motor_address); % motor address
176
                    %                   disp('positioning roll-axis...');
178
                    % relative positioning
180                 invoke(deviceObj, 'mvp','relative',...
                        stage_positioning.roll_axis.steps_per_stepsize *...
182                     input_rel_pos);

184                 % wait till stage reached position
                    while(1)
186                     if((get(deviceObj.Axis,'target_pos_reached') == 1) ||...
                                (get(deviceObj.Axis,'left_limit_switch_status')== 1)...
188                     ||(get(deviceObj.Axis,'right_limit_switch_status') == 1))
                            break;
190                     end;
                        pause(1);
192                 end
```

```
194                 invoke(deviceObj, 'mst'); % stop motor
196                 %                   disp('positioning finished');
198             case stage_setup.pitch_axis.motor_address % pitch-axis
200                 set(deviceObj,'address', ...
                        stage_setup.pitch_axis.module_address); % module address
202                 set(deviceObj,'motor', ...
                        stage_setup.pitch_axis.motor_address); % motor address
204
                    %                   disp('positioning pitch-axis...');
206
                    % relative positioning
208                 invoke(deviceObj, 'mvp','relative',...
                        stage_positioning.pitch_axis.steps_per_stepsize *...
210                     input_rel_pos);
212                 % wait till stage reached position
                    while(1)
214                     if((get(deviceObj.Axis,'target_pos_reached') == 1) ||...
                                (get(deviceObj.Axis,'left_limit_switch_status') == 1)...
216                     ||(get(deviceObj.Axis,'right_limit_switch_status') == 1))
                            break;
218                     end;
                        pause(1);
220                 end
222                 invoke(deviceObj, 'mst'); % stop motor
224                 %                   disp('positioning finished');
226             otherwise
228                 % throw exception
                    ME = MException('VerifyOutput:OutOfBounds', ...
230                     'No valid motor address');
                    throw(ME);
232
        end
234
    case 3 % alignment axis
236
        switch motorAddr
238
            case stage_setup.alignment_axis.motor_address % alignment-axis
240             set(deviceObj,'address', ...
                        stage_setup.alignment_axis.module_address); % module address
242             set(deviceObj,'motor', ...
                        stage_setup.alignment_axis.motor_address); % motor address
244
                %                   disp('positioning alignment-axis...');
246
                % relative positioning
248             invoke(deviceObj, 'mvp','relative',...
                        stage_positioning.alignment_axis.steps_per_stepsize *...
250                     (-input_rel_pos))
252             % wait till stage reached position
                while(1)
254                 if((get(deviceObj.Axis,'target_pos_reached') == 1) ||...
                            (get(deviceObj.Axis,'left_limit_switch_status') == 1))
256                     break;
                    end;
258                 pause(1);
```

```
                end
260
                invoke(deviceObj, 'mst'); % stop motor
262
                %                 disp('positioning finished');
264
            case 1 % no device
266
                % throw exception
268             ME = MException('VerifyOutput:OutOfBounds', ...
                    'No valid motor address');
270             throw(ME);

272         case 2 % no device

274             % throw exception
                ME = MException('VerifyOutput:OutOfBounds', ...
276                 'No valid motor address');
                throw(ME);
278
            otherwise
280
                % throw exception
282             ME = MException('VerifyOutput:OutOfBounds', ...
                    'No valid motor address');
284             throw(ME);

286     end % end switch motor address

288 otherwise

290     % throw exception
        ME = MException('VerifyOutput:OutOfBounds', ...
292         'No valid module address');
        throw(ME);
294
end; % end switch module address
```

D Tabellen

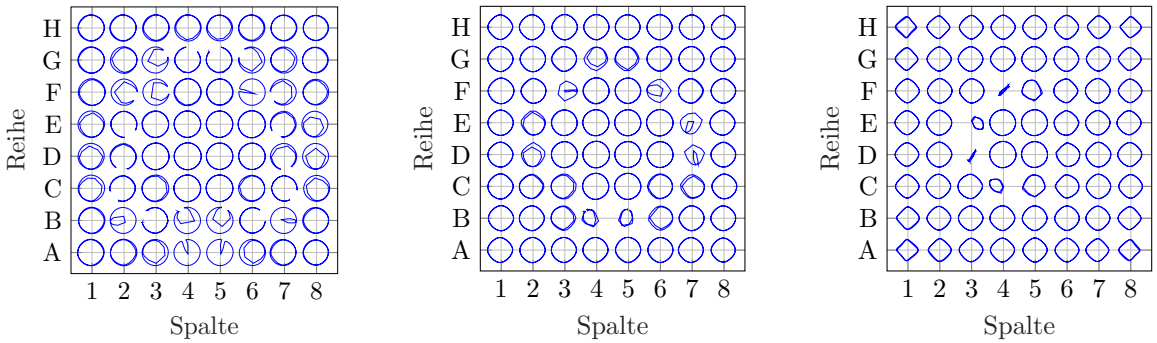
Tabelle D.1: Widerstandswerte der COS-Seite angegeben in kΩ.

0 Zeile			1	2	3	4	5	6	7	8
Brückenwiderstand in $k\Omega$	R_{C1}	+	247,6	236,9	248,0	233,4	226,4	242,6	215,2	220,5
		–	248,6	237,5	248,3	234,5	227,9	244,3	216,6	224,1
		$\lvert\Delta\rvert$	1,0	0,6	0,3	1,1	1,5	1,7	1,4	3,6
	R_{C2}	+	198,5	179,3	205,8	169,3	181,7	188,9	171,8	194,5
		–	197,9	179	206,8	169,7	182,5	189,7	173,6	196,0
		$\lvert\Delta\rvert$	0,6	0,3	1,0	0,4	0,8	0,8	1,8	1,5
	R_{C3}	+	251,1	228,1	245,5	214,9	198,1	236,6	228,0	235,9
		–	251,8	230,6	247,7	217,4	200,5	239,2	231,0	238,8
		$\lvert\Delta\rvert$	0,7	2,5	2,2	2,5	2,4	2,6	3,0	2,9
	R_{C4}	+	233,1	218,8	231,6	209,3	187,8	198,9	201,5	208,6
		–	237,3	224,3	236,8	214,5	193,8	204,8	207,6	214,6
		$\lvert\Delta\rvert$	4,2	5,5	5,2	5,2	6,0	5,9	6,1	6,0
	R_{C5}	+	248,8	220,7	247,7	231,8	225,2	238,8	209,5	229,4
		–	254,2	226	254,3	238,3	232,1	245,8	217,0	236,4
		$\lvert\Delta\rvert$	5,4	5,3	6,6	6,5	6,9	7,0	7,5	7,0
	R_{C6}	+	279,9	262,5	276,0	257,3	250,8	265,4	249,5	241,1
		–	288,1	270,6	284,5	266,2	260,3	275,0	258,3	251,7
		$\lvert\Delta\rvert$	8,2	8,1	8,5	8,9	9,5	9,6	8,8	10,6
	R_{C7}	+	267,7	252,9	268,3	246,9	213,9	255,9	244,9	248,0
		–	275,8	261,7	278,6	257,2	224,9	267,9	265,7	260,0
		$\lvert\Delta\rvert$	8,1	8,8	10,3	10,3	11,0	12,0	20,8	12,0
	R_{C8}	+	254,9	233,5	254,1	216,2	219,2	224,1	230,4	237,1
		–	262,5	241,6	263,0	224,1	228,3	233,9	240,6	247,5
		$\lvert\Delta\rvert$	7,6	8,1	8,9	7,9	9,1	9,8	10,2	10,4

Tabelle D.2: Widerstandswerte der SIN-Seite angegeben in kΩ.

Zeile			1	2	3	4	5	6	7	8
Brückenwiderstand in $k\Omega$	R_{S1}	+	266,0	256,0	266,9	251,7	242,0	257,3	229,1	233,5
		–	246,2	236,7	247,9	233,9	223,9	240,3	214,2	216,9
		$\lvert\Delta\rvert$	19,8	19,3	19 ,0	17,8	18,1	17,0	14,9	16,6
	R_{S2}	+	213,5	192,7	218,5	182,1	192,1	198,8	179,8	203,8
		–	200,7	180,3	206,1	171,5	181,0	187,1	169,9	192,9
		$\lvert\Delta\rvert$	12,8	12,4	12,4	10,6	11,1	11,7	9,9	10,9
	R_{S3}	+	267,2	244,5	262,4	230,8	208,7	247,9	240,1	247,2
		–	250,3	227,7	245,4	215,6	194,5	233,7	225,9	233,0
		$\lvert\Delta\rvert$	16,9	16,8	17,0	15,2	14,2	14,2	14,2	14,2
	R_{S4}	+	245,0	230,2	241,8	220,0	195,4	204,7	210„0	216,6
		–	235,3	222,1	234,4	211,8	187,1	197,6	202,9	210,1
		$\lvert\Delta\rvert$	9,7	8,6	7,4	8,2	8,3	7,1	7,1	6,5
	R_{S5}	+	262,6	234,1	258,9	244,0	234,6	246,7	216,7	237,8
		–	249,5	222,1	248,2	233,9	224,2	237,1	207,5	229,5
		$\lvert\Delta\rvert$	13,1	12,0	10,7	10,1	10,4	9,6	9,2	8,3
	R_{S6}	+	289,7	273,1	285,2	268,2	258,2	272,3	256,7	246,5
		–	279,9	264,3	276,2	258,5	250,0	264,0	249,3	238,8
		$\lvert\Delta\rvert$	9,8	8,8	9,0	9,7	8,2	8,3	7,4	7,7
	R_{S7}	+	281,0	265,4	279,1	257,5	220,8	263,0	253,8	255,8
		–	272,8	256,6	272,3	251,0	214,5	256,7	247,4	250,1
		$\lvert\Delta\rvert$	8,2	8,8	6,8	6,5	6,3	6,3	6,4	5,7
	R_{S8}	+	265,3	243,9	264,1	225,5	226,4	230,3	237,2	243,7
		–	261,1	239,4	260,4	223,5	224,4	227,1	234,4	241,5
		$\lvert\Delta\rvert$	4,2	4,5	3,7	2,0	2,0	3,2	2,8	2,2

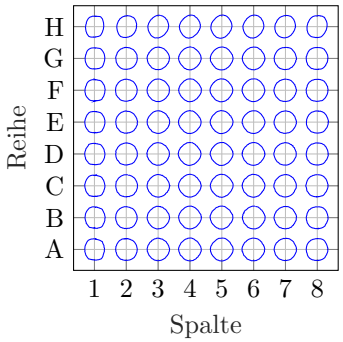
E Messungen



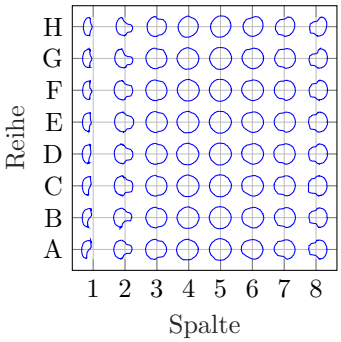
(a) Mitte KMZ große Kugel.

(b) Mitte KMZ kleine Kugel

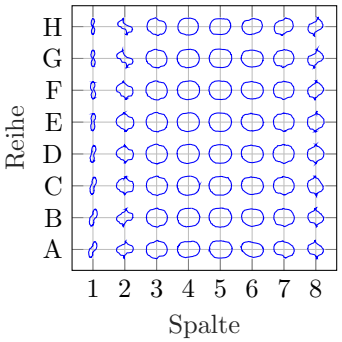
(c) Mitte KMZ Würfelmagnet



(d) Mitte NVE große Kugel

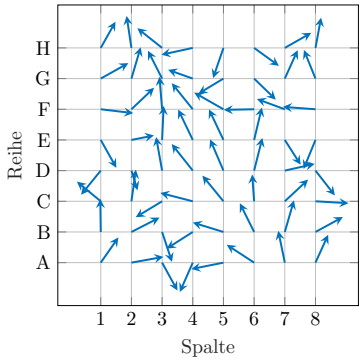


(e) Mitte NVE kleine Kugel

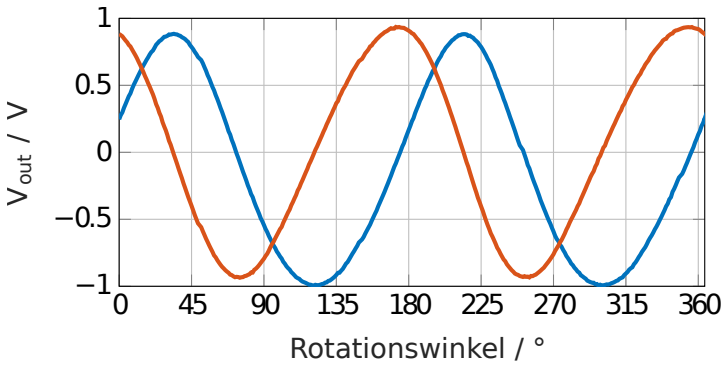


(f) Mitte NVE Würfelmagnet

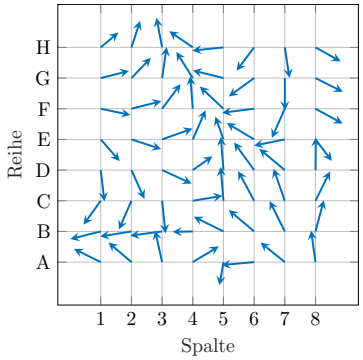
Abbildung E.1: Gegenüberstellungen der Kreisdarstellungen von KMZ und NVE mit verschiedenen Magneten mittig über beide Sensor-Arrays.



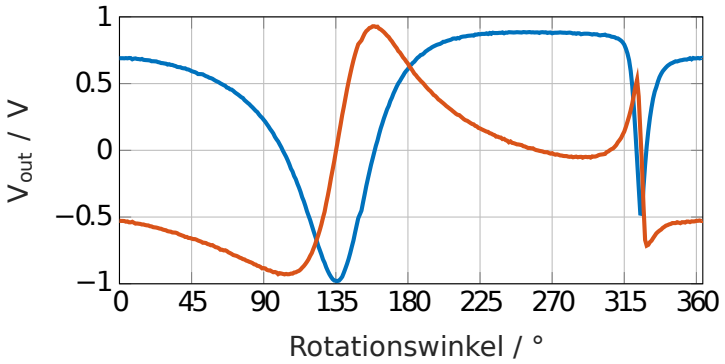
(a) Vektordarstellung der Ausgangssignale bei 45° Rotation des Magneten über der Mitte des Arrays.



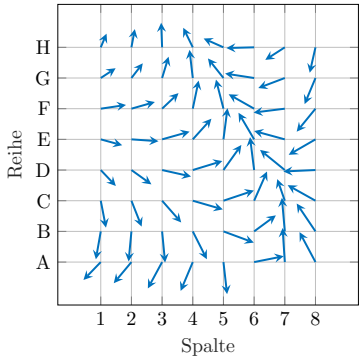
(b) SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5) in Abhängigkeit der Rotation des Magneten über der Mitte des Arrays.



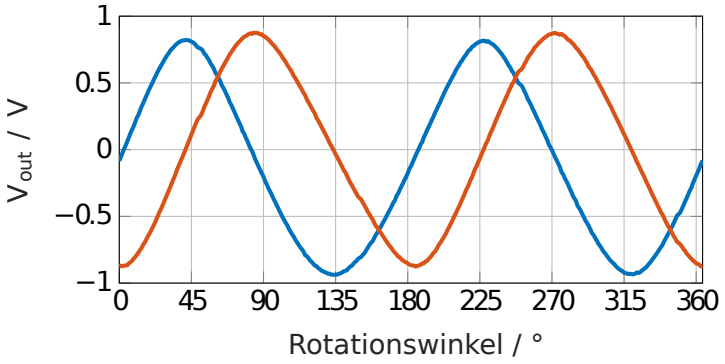
(c) Vektordarstellung der Ausgangssignale bei 45° Rotation des Magneten über dem Sensor (F,3).



(d) SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5) in Abhängigkeit der Rotation des Magneten über dem Sensor (F,3).

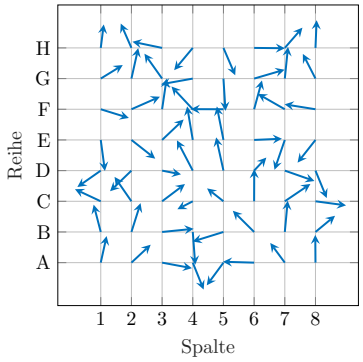


(e) Vektordarstellung der Ausgangssignale bei 45° Rotation des Magneten über dem Sensor (A,8).

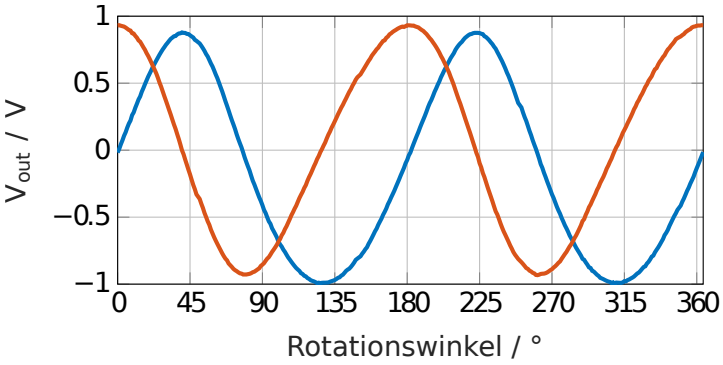


(f) SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5) in Abhängigkeit der Rotation des Magneten über dem Sensor (A,8).

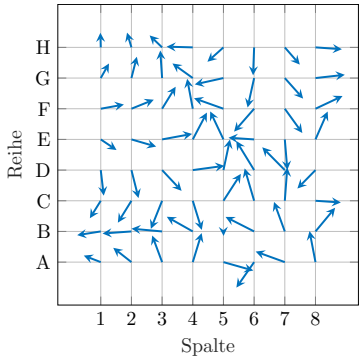
Abbildung E.2: Messungen mit dem KMZ-60 Sensor-Array und einem Kugelmagneten (d = 19 mm) im Abstand von 19 mm bei drei verschiedenen Positionen über dem Sensor-Array.



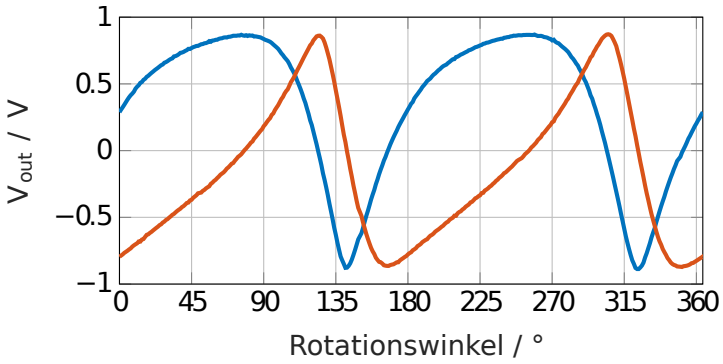
(a) Vektordarstellung der Ausgangssignale bei 45° Rotation des Magneten über der Mitte des Arrays.



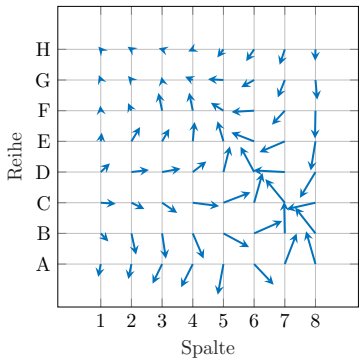
(b) SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5) in Abhängigkeit der Rotation des Magneten über der Mitte des Arrays.



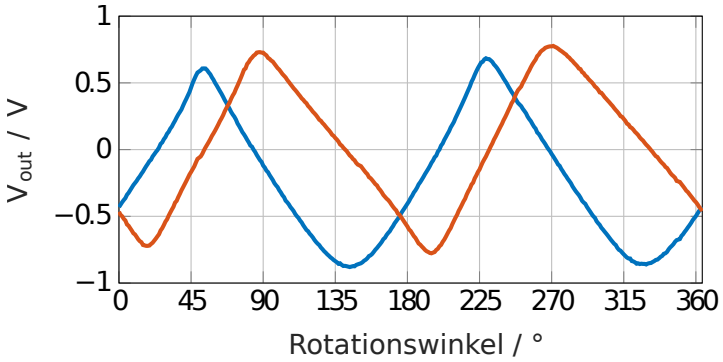
(c) Vektordarstellung der Ausgangssignale bei 45° Rotation des Magneten über dem Sensor (F,3).



(d) SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5) in Abhängigkeit der Rotation des Magneten über dem Sensor (F,3).

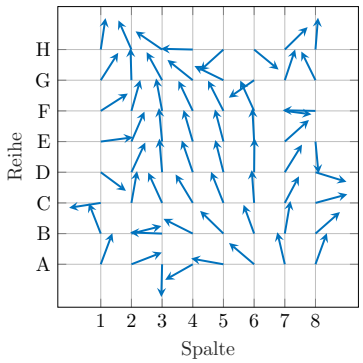


(e) Vektordarstellung der Ausgangssignale bei 45° Rotation des Magneten über dem Sensor (A,8).

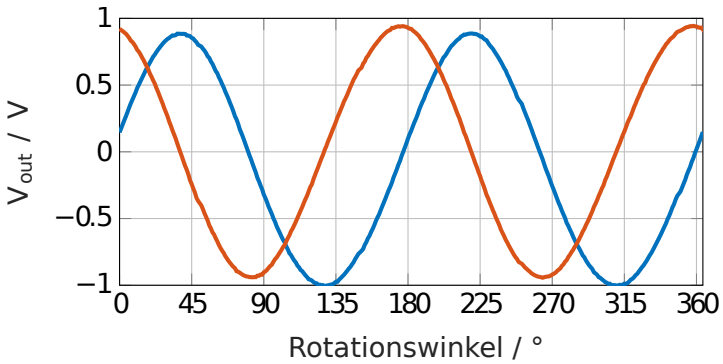


(f) SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5) in Abhängigkeit der Rotation des Magneten über dem Sensor (A,8).

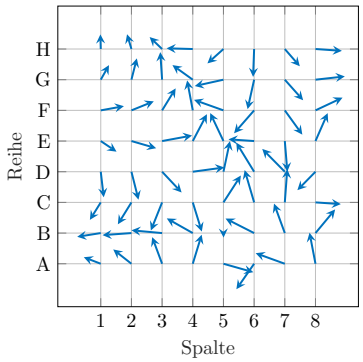
Abbildung E.3: Messungen mit dem KMZ-60 Sensor-Array und einem Würfelmagneten der Kantenlänge 10 mm im Abstand von 19 mm bei drei verschiedenen Positionen über dem Sensor-Array.



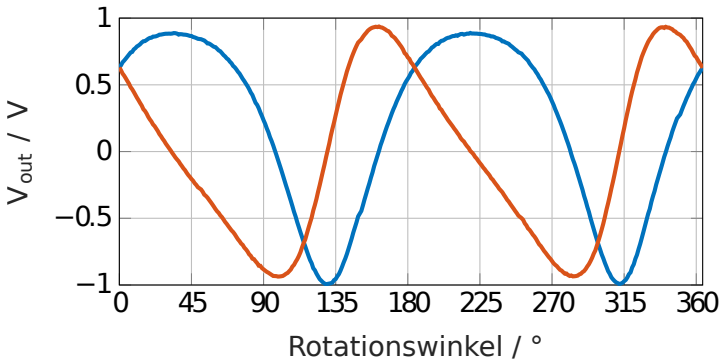
(a) Vektordarstellung der Ausgangssignale bei 45° Rotation des Magneten über der Mitte des Arrays.



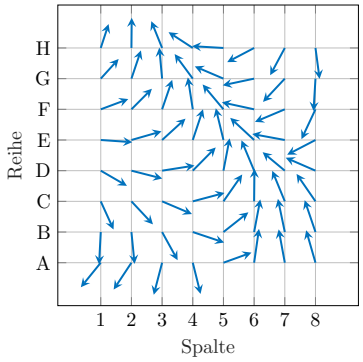
(b) SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5) in Abhängigkeit der Rotation des Magneten über der Mitte des Arrays.



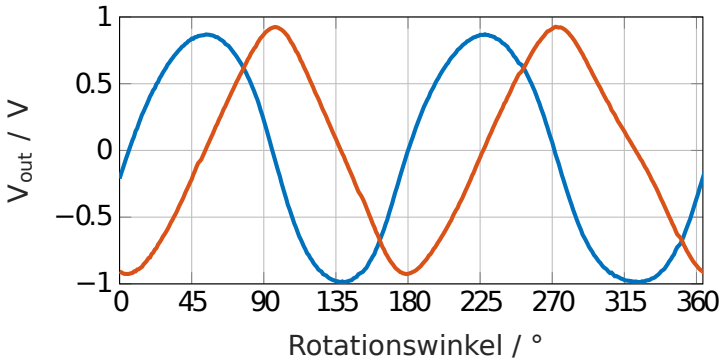
(c) Vektordarstellung der Ausgangssignale bei 45° Rotation des Magneten über dem Sensor (F,3).



(d) SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5) in Abhängigkeit der Rotation des Magneten über dem Sensor (F,3).

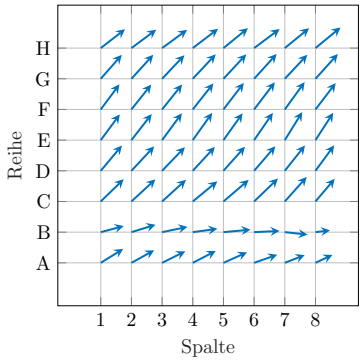


(e) Vektordarstellung der Ausgangssignale bei 45° Rotation des Magneten über dem Sensor (A,8).

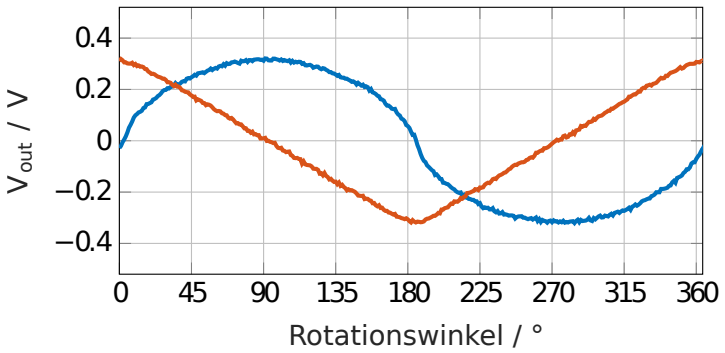


(f) SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5) in Abhängigkeit der Rotation des Magneten über dem Sensor (A,8).

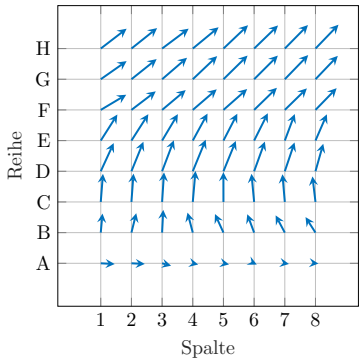
Abbildung E.4: Messungen mit dem KMZ-60 Sensor-Array und einem Quadermagneten (l = 40 mm, w = 40 mm, h = 20 mm) im Abstand von 19 mm bei drei verschiedenen Positionen über dem Sensor-Array.



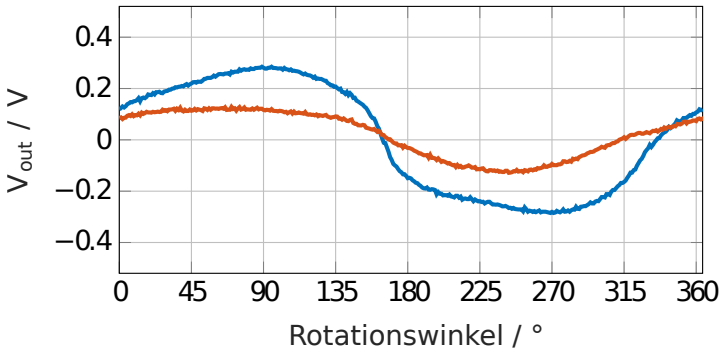
(a) Vektordarstellung der Ausgangssignale bei 45° Rotation des Magneten über der Mitte des Arrays.



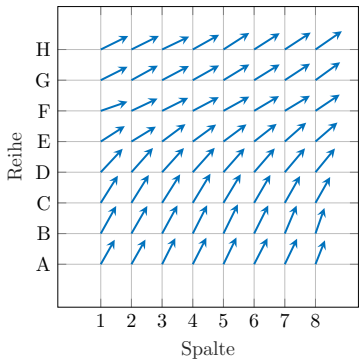
(b) SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5) in Abhängigkeit der Rotation des Magneten über der Mitte des Arrasy.



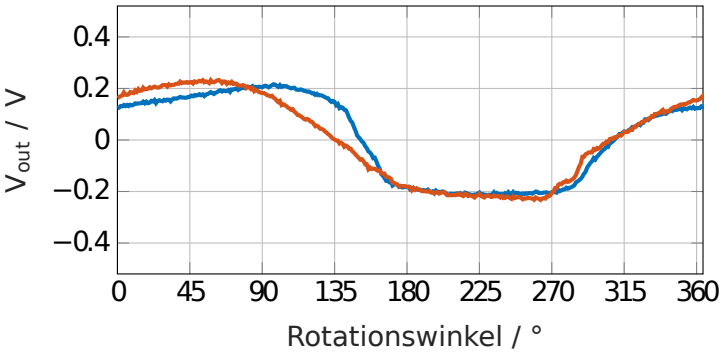
(c) Vektordarstellung der Ausgangssignale bei 45° Rotation des Magneten über dem Sensor (C,3).



(d) SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5) in Abhängigkeit der Rotation des Magneten über dem Sensor (C,3).

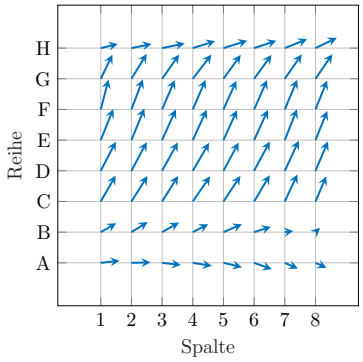


(e) Vektordarstellung der Ausgangssignale bei 45° Rotation des Magneten über dem Sensor (A,1).

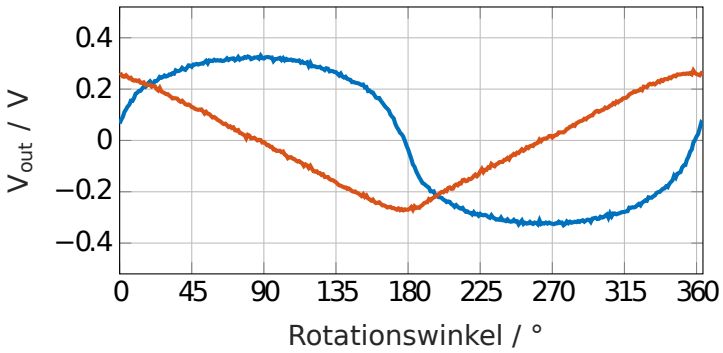


(f) SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5) in Abhängigkeit der Rotation des Magneten über dem Sensor (A,1).

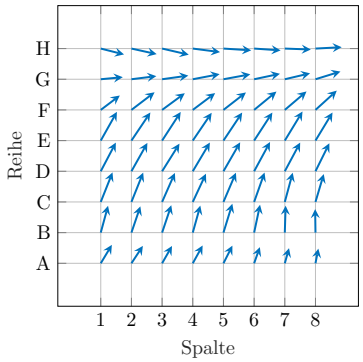
Abbildung E.5: Messungen mit dem NVE-AAT001-10E Sensor-Array und einem Kugelmagneten (d = 19 mm) im Abstand von 19 mm bei drei verschiedenen Positionen über dem Sensor-Array.



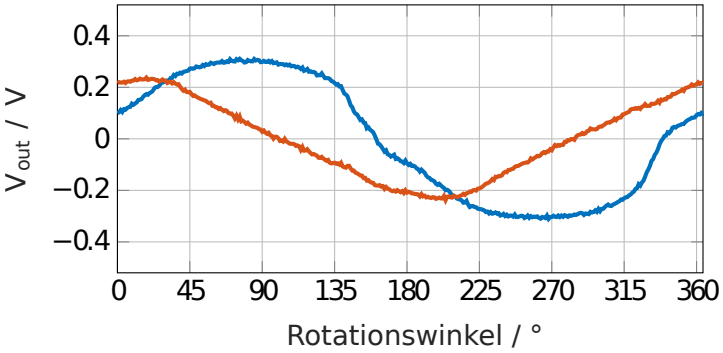
(a) Vektordarstellung der Ausgangssignale bei 45° Rotation des Magneten über der Mitte des Arrays.



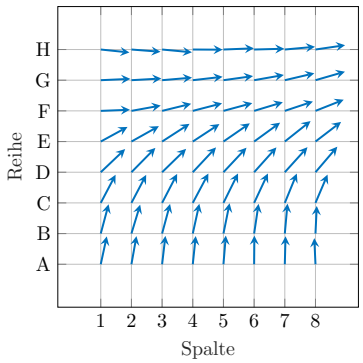
(b) SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5) in Abhängigkeit der Rotation des Magneten über der Mitte des Arrays.



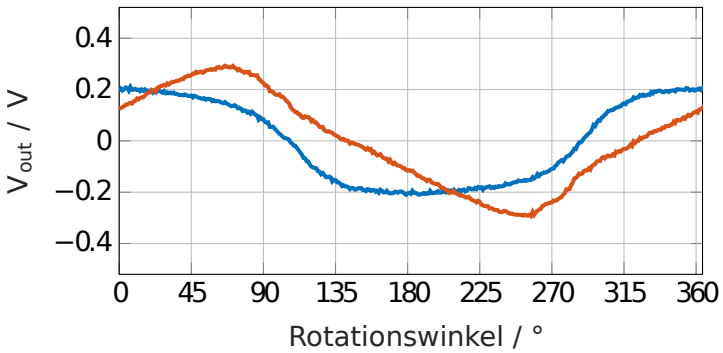
(c) Vektordarstellung der Ausgangssignale bei 45° Rotation des Magneten über dem Sensor (C,3).



(d) SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5) in Abhängigkeit der Rotation des Magneten über dem Sensor (C,3).

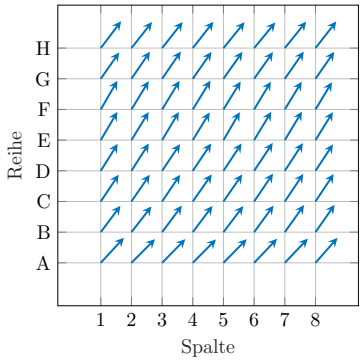


(e) Vektordarstellung der Ausgangssignale bei 45° Rotation des Magneten über dem Sensor (A,1).

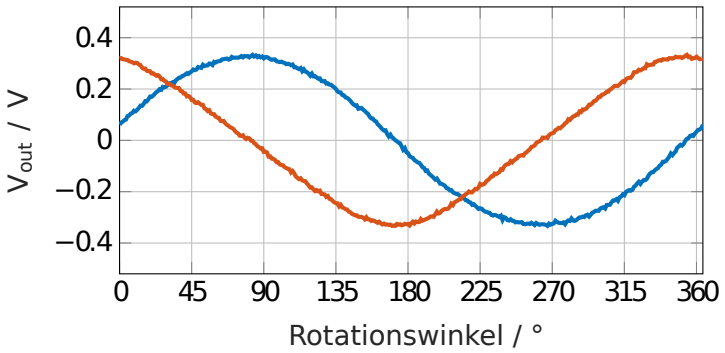


(f) SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5) in Abhängigkeit der Rotation des Magneten über dem Sensor (A,1).

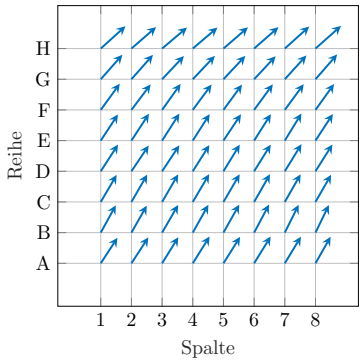
Abbildung E.6: Messungen mit dem NVE-AAT001-10E Sensor-Array und einem Würfelmagneten der Kantenlänge 10 mm im Abstand von 19 mm bei drei verschiedenen Positionen über dem Sensor-Array.



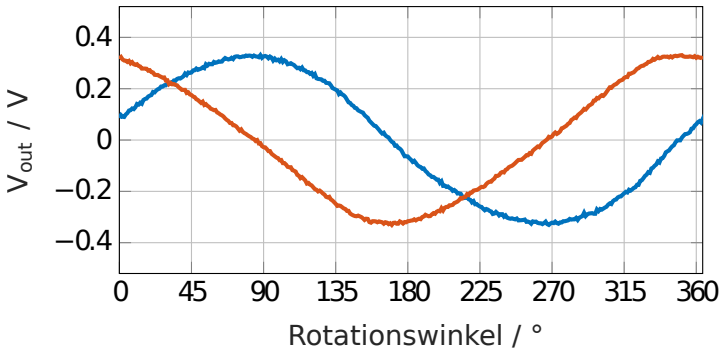
(a) Vektordarstellung der Ausgangssignale bei 45° Rotation des Magneten über der Mitte des Arrays.



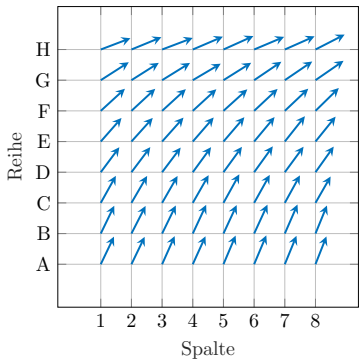
(b) SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5) in Abhängigkeit der Rotation des Magneten über der Mitte des Arrays.



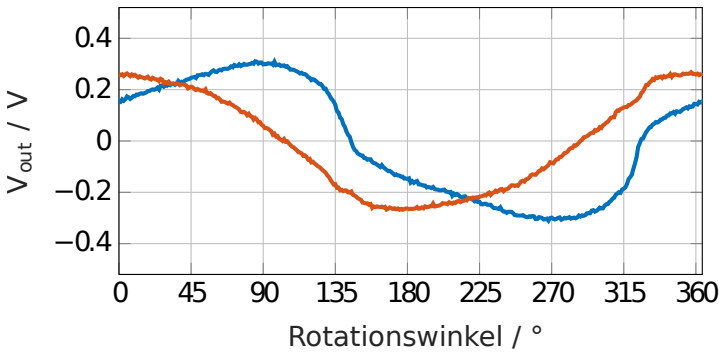
(c) Vektordarstellung der Ausgangssignale bei 45° Rotation des Magneten über dem Sensor (C,3).



(d) SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5) in Abhängigkeit der Rotation des Magneten über dem Sensor (C,3).



(e) Vektordarstellung der Ausgangssignale bei 45° Rotation des Magneten über dem Sensor (A,1).



(f) SIN- (rot) und COS-Ausgangssignale (blau) des Sensors (E,5) in Abhängigkeit der Rotation des Magneten über dem Sensor (A,1).

Abbildung E.7: Messungen mit dem NVE-AAT001-10E Sensor-Array und einem Quadermagneten (l = 40 mm, w = 40 mm, h = 20 mm) im Abstand von 19 mm bei drei verschiedenen Positionen über dem Sensor-Array.

F CD

Auf der beigefügten CD befinden sich sämtliche Programme, Messdaten, Darstellungen die im Rahmen dieser Arbeit erstellt wurden.

-

Selbstständigkeitserklärung

Hiermit versichere ich, Abraham Begic, 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, 09. Juli 2018