

# DIPLOMARBEIT

## Softwareentwicklung und Simulation für einen kapazitiven Drehmomentsensor

ausgeführt am Institut für  
Industrielle Elektrotechnik und Materialwissenschaften  
der Technischen Universität Wien

unter Anleitung von

**Univ.Prof. Dipl.-Ing. Dr.techn. Georg Brasseur**  
**Dipl.-Ing. Dr.techn. Paul Fulmek**

durch

**Stefan Cermak**  
Matrikelnummer: 9426086  
1170 Wien, Czartoryskigasse 220/10

Wien, im August 2000

.....

# DANKE !

Es ist mir ein besonderes Anliegen, mich bei all jenen zu bedanken, die mich bei meiner Ausbildung und der Erstellung dieser Diplomarbeit unterstützt haben:

Mein erster Dank geht an meinen betreuenden Professor, Herrn Dr. Georg Brasseur, der meinen Studienkollegen Wolfgang Zdiarsky, Florian Wandling und mir die Gelegenheit gab, unter seiner Leitung im Bereich der Automobilelektronik an einem Projekt mitzuarbeiten. Auch hat er es uns ermöglicht, dieses Projekt bei internationalen Konferenzen zu präsentieren, wodurch wir weitere Erfahrungen sammeln konnten. Ein Teil des Projektes war dann Grundlage für meine Diplomarbeit.

Mein Dank gilt Herrn Professor Brasseur auch dafür, daß ich durch ihn an Hand industrieller Projekte bei der Arbeitsgemeinschaft Automobilelektronik praktische Erkenntnisse erlangen konnte. Bei den Herren Edmund Bartl, Dipl.-Ing. Tibor Fabian, Dr. Paul Fulmek und Dipl.-Ing. Dimitri Lawroff bedanke ich mich für die herzliche Aufnahme in die Arbeitsgemeinschaft.

Meinen Studienkollegen kann ich nicht genug für ihre treue Freundschaft und die verlässliche Unterstützung danken. Die zahlreich abgehaltenen Lernsessions werden mir immer in Erinnerung bleiben.

Ein ganz besonderer Dank gebührt aber meinen Eltern, die mir mein Studium überhaupt ermöglicht haben. Meine lange Schulzeit (Volksschule, Gymnasium, HTL und UNI), in welcher ich jede Art von Unterstützung bekommen habe, muß sie wohl zahlreiche Nerven gekostet haben.



# Inhaltsverzeichnis

<b>1</b>	<b>Vorwort</b>	<b>1</b>
<b>2</b>	<b>Sensoren zur Momentenmessung</b>	<b>2</b>
2.1	Anforderungen an einen Momentensensor . . . . .	2
2.2	Verschiedene Konzepte zur Messung von Drehmomenten . . . . .	3
2.2.1	Momentenmessung durch Rückführung auf einfache Kraftmessung an einem Hebel . . . . .	3
2.2.2	Momentensensor mit Dehnungsmeßstreifen . . . . .	3
2.2.3	Magnetoelastischer Momentensensor . . . . .	4
2.2.4	Piezoelektrischer Momentensensor . . . . .	4
2.2.5	Messung des Moments mit Hilfe eines Schwingsaiten- frequenzumsetzers . . . . .	5
<b>3</b>	<b>Prinzip des kapazitiven Momentensensors</b>	<b>6</b>
3.1	Einleitung in die kapazitive Meßtechnik . . . . .	6
3.1.1	Idee des kapazitiven Sensors . . . . .	6
3.1.2	Ratiometrisches Prinzip . . . . .	7
3.1.3	Erklärung des Sensorprinzipes anhand des Platten- modelles . . . . .	9
3.1.4	Anwendung auf den Winkelsensor . . . . .	11
3.2	Konzept der Momentenmessung . . . . .	12
3.2.1	Torsion einer mechanischen Welle . . . . .	12
3.2.2	Mechanischer Aufbau . . . . .	13
3.2.3	Elektrodenstruktur und Auswertung . . . . .	14
<b>4</b>	<b>Projektübersicht</b>	<b>20</b>

<b>5</b>	<b>Feldsimulation</b>	<b>24</b>
5.1	Die Struktur der Feldsimulation . . . . .	25
5.2	Das Feld einer Streifenladung . . . . .	27
5.3	Beschreibung des Feldproblems . . . . .	30
5.3.1	Aufstellen des beschreibenden Gleichungssystem . . . . .	31
5.4	Der Segmentverlauf . . . . .	37
5.5	Simulationsanwendung . . . . .	39
5.5.1	Die optimale Sensorhöhe . . . . .	41
5.5.2	Die optimale Rotorhöhe . . . . .	43
5.5.3	Einfluß der Rotorverkipfung . . . . .	45
5.6	Resumé . . . . .	46
<b>6</b>	<b>Die Sensor-Software</b>	<b>47</b>
6.1	Die Hauptdatei . . . . .	48
6.1.1	main.h . . . . .	48
6.1.2	main.c . . . . .	52
6.2	Der periodische Interrupt . . . . .	53
6.2.1	per_int.h . . . . .	53
6.2.2	per_int.c . . . . .	54
6.3	Die Meßroutine . . . . .	55
6.3.1	measure.h . . . . .	55
6.3.2	measure.c . . . . .	55
6.4	Der Algorithmus . . . . .	59
6.4.1	Algorith.h . . . . .	59
6.4.2	Algorith.c . . . . .	59
6.5	Die PWM-Ausgabe . . . . .	67
6.5.1	PWM.h . . . . .	67
6.5.2	pwm.c . . . . .	67
6.6	Die serielle Schnittstelle . . . . .	68
6.6.1	ser.h . . . . .	68
6.6.2	ser.c . . . . .	69

<b>7</b>	<b>Windows-Software zur Meßplatz Automatisierung</b>	<b>73</b>
7.1	Bedienungsanleitung für die Meßplatz Software . . . . .	74
7.1.1	Meßplatzsteuerung . . . . .	74
7.1.2	Kommunikations Konfiguration . . . . .	75
7.1.3	Phi definieren . . . . .	77
7.2	Die DDE Schnittstelle . . . . .	77
7.3	Spezifikation der DDE-Schnittstelle . . . . .	77
<b>8</b>	<b>Spezifikation des Prototypes</b>	<b>79</b>
<b>9</b>	<b>Ausblick</b>	<b>84</b>
	<b>Literaturverzeichnis</b>	<b>85</b>
<b>A</b>	<b>Capacitive Sensor for Relative Angle Measurement</b>	<b>87</b>

# Abbildungsverzeichnis

2.1	<i>Positionierung von DMS bei Momentenmessung . . . . .</i>	3
2.2	<i>Prinzip des magnetoelastischen Sensors (Darstellung lt. Firma Magna-lastic Devices, Inc.) . . . . .</i>	4
2.3	<i>Anordnung von Schwingssaiten zur Momentenmessung . . . . .</i>	5
3.1	<i>Beispiel für eine Elektrodenanordnung (Abb. lt. [FAB97]) . . . . .</i>	7
3.2	<i>Blockschaltbild eines kapazitiven Sensors (Meßkette und Auswerteeinheit) .</i>	7
3.3	<i>Prinzipielle Plattenanordnung bestehend aus Sendekondensatorplatten <math>S_n</math>, Empfangskondensatorplatte <math>E</math> und in x-Richtung verschiebbarem Rotor <math>R</math>.</i>	10
3.4	<i>Kapazitätsverlauf von <math>C_{S1E}(x_H)</math> ideal: durchgezogene Linie, Einfluß der Streufelder: strichlierte Linie. . . . .</i>	11
3.5	<i>Formänderung bei Torsionsspannung lt. [BOEG85] . . . . .</i>	13
3.6	<i>Mechanische "Rückführung" des Verdrehwinkels <math>\varphi_{rel}</math> . . . . .</i>	14
3.7	<i>Elektrodenstruktur des Relativwinkelsensors (oben: 3D-Ansicht, unten von links nach rechts: Senderelektrode, Rotor 1, Rotor 2, Empfängerelektrode)</i>	15
3.8	<i>"Abwicklungsdarstellung" der Elektrodenstruktur aus Abbildung 3.7 in Nulllage (<math>x_{rel} = 0</math> bzw. <math>\varphi_{rel} = 0</math>) . . . . .</i>	15
4.1	<i>Blockschaltbild zur Projektübersicht - Die fettumrandeten Blöcke sind Gegenstand des Gesamtprojektes "Kapazitiver Momentensensor" . . . . .</i>	22
4.2	<i>Photo der Meßaufbauten - links-mitte: Prüfstand mit (von links nach rechts): Momentensensor in Positioniereinheit, Schrittmotor, Referenzwinkelsensor (BALDWIN), darüber: Verstärker für Schrittmotoransteuerung, rechts: Auswertemonitor . . . . .</i>	23
5.1	<i>Symbolische Plattenanordnung, mit eingezeichneten Ersatzkapazitäten. Da Rotor und Empfänger gemeinsam auf Massepotential liegen, verschwindet der Einfluß von <math>C_{RE}</math>. . . . .</i>	26

5.2	<i>Graphische Darstellung einer simulierten Ladungsverteilung. Die Polarität der Ladungen ist durch die Farben Rot und Blau dargestellt. Die Intensität der Farben ist ein Maß für die Ladungsdichte. Am Sender ist die Verdrängung der Ladungen (Ladungen gleicher Polarität stoßen sich ab) an die Plattenränder zu erkennen. Beim Rotor hingegen bewirken die Ladungen am Sender, daß sich die Ladungen zum Großteil in der Mitte der Platte, wo der Abstand zum Sender minimal wird, konzentrieren.</i>	26
5.3	<i>Die Abbildung zeigt eine Linienladung im Punkt <math>(x_{lin}, 0)</math> mit eingezeichneten Radiusvektor zum Punkt <math>(x_p, y_p)</math>.</i>	27
5.4	<i>Die Abbildung zeigt die zur Formel 5.7 gehörende geometrische Anordnung.</i>	28
5.5	<i>Segmentierte Plattenanordnung für das zu lösende Feldproblem bei unabgedecktem Rotor (maximaler Kapazität).</i>	31
5.6	<i>Vergleich des Segmentverlaufes bei verschiedenen Rotorhöhen. Man erkennt, daß hauptsächlich der minimale Kapazitätswert beeinflußt wird.</i>	39
5.7	<i>Simulierter Winkelfehlerverlauf bedingt durch den Streufeldverlauf bei zwei unterschiedlichen Auswertalgorithmen.</i>	40
5.8	<i>Vergleich eines berechneten und eines gemessenen Kapazitätsverlaufes.</i>	40
5.9	<i>Simulierter Einfluß der Sensorhöhe auf den maximalen Winkelfehler. Es zeigt sich, daß beim Algorithmus 2 (b) der Fehler durchgehend kleiner ist als beim Basisalgorithmus (a), und sogar bei größeren Abständen <math>d</math> kleinere Werte zeigt. Algorithmus 2 ist daher bevorzugt einzusetzen.</i>	43
5.10	<i>Simulierter Einfluß der Rotorhöhe auf den Winkelfehler bei Verwendung des Basisalgorithmus (a) und des Algorithmus 2 (b).</i>	45
5.11	<i>Einfluß der Rotorverkipfung auf die Relativwinkelabweichung (Vergleich Simulation und Messung)</i>	46
6.1	<i>Flußdiagramm der Hauptdatei - "main"</i>	48
6.2	<i>Flußdiagramm des periodischen Interrupts - "per_int"</i>	49
6.3	<i>Flußdiagramm des seriellen Interrupts - "ser"</i>	50
6.4	<i>Planare Abwicklungsdarstellung des Sensors. Die beiden Rotoren sind in ihrer neutralen Lage (keine Relativverschiebung) eingezeichnet. Über den Segmenten ist die ohne Streufelder wirkende, auf die Segmentkapazität bezogene, Sender-Empfänger Kapazität eingetragen</i>	60
7.1	<i>Das Hauptfenster der Prüfstandssoftware</i>	74
8.1	<i>Gemessene Relativwinkelabweichung über Referenzrelativwinkel und Referenzabsolutwinkel - Inverse Vierflügelstruktur</i>	80
8.2	<i>Mittlere Relativwinkelabweichung über Referenzabsolutwinkel - Inverse Vierflügelstruktur. Die Mittlere Abweichung geht ab einem Absolutwinkel von <math>180^\circ</math> von <math>\pm 0.2^\circ</math> auf <math>\pm 0.08^\circ</math> zurück, weil dann alle Segmentwerte bereits einmal kalibriert werden konnten.</i>	81

8.3	<i>Mittlere Relativwinkelabweichung über Referenzrelativwinkel - Inverse Vierflügelstruktur . . . . .</i>	81
8.4	<i>Gemessene Absolutwinkelabweichung über Referenzrelativwinkel und Referenzabsolutwinkel - Inverse Vierflügelstruktur . . . . .</i>	82
8.5	<i>Mittlere Absolutwinkelabweichung über Referenzabsolutwinkel - Inverse Vierflügelstruktur. Die Mittlere Abweichung geht ab einem Absolutwinkel von 180° von <math>\pm 0.12^\circ</math> auf <math>\pm 0.08^\circ</math> zurück, weil dann alle Segmentwerte bereits einmal kalibriert werden konnten. . . . .</i>	82
8.6	<i>Mittlere Absolutwinkelabweichung über Referenzrelativwinkel - Inverse Vierflügelstruktur . . . . .</i>	83



# Abkürzungsverzeichnis

ADC	Analog Digital Converter
ACAD	Automatic Computer Aided Design
ANSI	American National Standardization Institute
CMOS	Complementary Metal Oxide Semiconductor
CNC	Computer Numeric Controlled
DDE	Dynamic Data Exchange
DMS	DehnungsMeßStreifen
EMV	ElektroMagnetische Verträglichkeit
HC12	Microcomputer MC68HC912D60 von Motorola
IAEE	Institut für Allgemeine Elektrotechnik und Elektronik
IAEQ	Institut für Angewandte Elektronik und Quantenelektronik
IEEE	Institute of Electrical and Electronics Engineers
IEMW	Institut für Industrielle Elektronik und Materialwissenschaften
IMEKO	International Measurement Confederation
IMTC	Conference on Instrumentation and Measurement Technology
M-Files	MatLab Programmdateien
$\mu C$	Microcontroller
PC	Personal Computer
PWM	Puls Width Modulation
TU	Technische Universität

# Verwendete Zeichen und Symbole

## für mechanische Größen:

$f_0$	mechanische Eigenfrequenz einer gespannten Saite
$\varepsilon_{mech}$	Dehnung
$E_{mech}$	Elastizitätsmodul
$\varrho_{Saite}$	Dichte
$l_S$	Länge der gespannten Saite
$l$	Länge der Welle
$M$	Drehmoment
$\tau_t$	Torsionsspannung
$v$	Formänderung durch Verdrehung
$\varphi$	Verdrehwinkel
$G$	Schubmodul
$W_P$	polares Widerstandsmoment
$r$	Radius der Welle
$(x_P, y_P)$	Punkt im 2 dimensionalen Raum

## für elektrische Größen:

$\tau_{el}$	elektrische Zeitkonstante
$C_{piezo}$	elektrische Ersatzkapazität eines Piezoelementes
$R_{piezo}$	elektrischer Ersatzwiderstand eines Piezoelementes
$\varepsilon$	Permittivität
$\varepsilon_0$	elektrische Feldkonstante ( $\varepsilon_0 = 8.8541878 \cdot 10^{-12} F/m$ )
$\varepsilon_R$	relative Permittivität
$C$	elektrische Kapazität
$C'$	längenbezogene Kapazität, Kapazitätsbelag
$C_{SiE}$	Kapazität zwischen Sendesegment Si und Empfangselektrode E
$C_{SiR}$	Kapazität zwischen einem allgemeinen Sendesegment Si und dem Rotor R
$C_{RE}$	Kapazität zwischen dem Rotor R und der Empfangselektrode E
$C_{SiEmax}$	maximale Kapazität von $C_{SiE}$
$C_{sim}$	der von der Simulation ermittelte Kapazitätswert
$C_{Plat}$	Kapazität eines homogen geladenen Plattenkondensators
$C_{tat}$	tatsächlich erwarteter Kapazitätswert, Ermittlung durch

	Verbesserung des Simulationswertes.
$\tau$	Linienladungsdichte
$\sigma$	Flächenladungsdichte
$\varrho$	Abstand zwischen Linienladung und Betrachtungspunkt
$\vec{e}_\varrho$	Einheitsvektor, zeigt von der Linienladung zum Betrachtungspunkt
$E_{lin}$	Feldstärke einer Linienladung im Betrachtungspunkt P.
$E_y$	Feldstärke in y-Richtung
$T_{max}$	Anzahl der simulierten Ersatzlinienladungen in der Simulation
$T_{Sender}$	Anzahl der simulierten Ersatzlinienladungen für den Sender
$T_{Rotor}$	Anzahl der simulierten Ersatzlinienladungen für den Rotor
$\mathbf{U}$	Spannungsvektor, enthält die Potentiale der einzelnen Ersatzlinienladungen
$\mathbf{Q}$	Ladungsvektor, enthält die Ladungen der einzelnen Ersatzlinienladungen
$\mathbf{C}$	Kapazitätsmatrix, drückt den Zusammenhang zwischen $\mathbf{U}$ und $\mathbf{Q}$ aus.
$\mathbf{K}$	Konstantenmatrix, beschreibt den Zusammenhang zwischen $\mathbf{Q}$ und $\mathbf{U}$ .

#### für Messsignale:

$k$	Verstärkung
$y_{offset}$	Signaloffset
$y_{isti}$	Istwert der zu messenden Größe ( $i = 1, 2, \dots$ )
$y_{messi}$	Messwert ( $i = 1, 2, \dots$ )
$y_{istmax}$	Maximalwert der zu messenden Größe
$y_{istmin}$	Minimalwert der zu messenden Größe
$y_{messmax}$	gemessener Maximalwert der zu messenden Größe
$y_{messmin}$	gemessener Minimalwert der zu messenden Größe
$y_{plus}$	Meßwert bei positiver Steigung
$y_{minus}$	Meßwert bei negativer Steigung
$\xi_{interpol}$	Positionsinterpolationswert (auf Intervall $[-1,1]$ beschränkt)
$y_{relmax}$	Maximalwert bei Relativwinkelmessung
$y_{relmin}$	Minimalwert bei Relativwinkelmessung
$y_{relplus}$	Meßwert bei positiver Steigung bei Relativwinkelmessung
$y_{relminus}$	Meßwert bei negativer Steigung bei Relativwinkelmessung
$\xi_{rel}$	Relativwinkelinterpolationswert (auf Intervall $[-1,1]$ beschränkt)
$y_{absmax}$	Maximalwert bei Absolutwinkelmessung
$y_{absmin}$	Minimalwert bei Absolutwinkelmessung
$y_{absplus}$	Meßwert bei positiver Steigung bei Absolutwinkelmessung
$y_{absminus}$	Meßwert bei negativer Steigung bei Absolutwinkelmessung
$\xi_{abs}$	Absolutwinkelinterpolationswert (auf Intervall $[-1,1]$ beschränkt)
$SW_i$	i-ter Segmentwert
$H(i)$	Liste der Segmentwerte in Nullage (auf ganzzahlige Werte skaliert)
$R_{xy}(\tau)$	Korrelation zweier um $\tau$ verschobener Messsignale $x(t)$ , $y(t)$
$x(t)$	Messsignal
$y(t)$	Messsignal
$t$	Zeit

$\tau$	zeitliche Verschiebung
$T$	Periodendauer
$R_{SW,H}(j)$	Korrelation der um $j$ verschobener diskreter Signale $SW(i)$ , $H(i)$
$j$	Segementoffset
$n$	Anzahl der Segemente

#### für Elektrodenstrukturen:

$R$	Rotor
$S_1, E_1$	Elektroden des einfachen Plattenkondensators
$S_i$	i-tes Sendesegment ( $i = 1 \dots$ Anzahl der Segmente)
$E$	Empfangselektrode
$RS$	Breite des Rotorschattens (lt. Abbildung 3.8)
$K_i$	relevante Rotorkante ( $i = 1 \dots 4$ )(lt. Abbildung 3.8)
$x_S$	Segmentbreite
$x_{PER}$	räumliche Periodizität der Segmente
$x_H$	Rotorposition (lt. Abbildung 3.3)
$x_{abs}$	Absolute Rotorposition (lt. Abbildung 3.8)
$x_{rel}$	Relative Rotorverschiebung (lt. Abbildung 3.8)
$x_{Ki}$	Lage der Rotorkanten ( $i = 1 \dots 4$ ) (lt. Abbildung 3.8)
$\varphi_{abs}$	Absolutwinkel des resultierenden Rotors
$\varphi_{rel}$	Relativwinkel zwischen den beiden Rotoren
$A$	wirksame Elektrodenfläche
$a$	Tiefe der Elektrode
$x_S$	Breite der Elektrode
$d$	Elektrodenabstand
$d_{tol}$	Toleranz der Rotorhöhe
$d_{r1}, d_{r2}$	Dicke der Rotorflügel
$d_{min}$	minimal zulässiger Sender-Empfänger Abstand
$r_P$	Rotorposition
$h_R$	Rotorhöhe

# Kapitel 1

## Vorwort

Im Rahmen der Arbeiten am Institut für Angewandte Elektrotechnik und Quantenelektronik (IAEQ) in der Arbeitsgruppe Automobilelektronik wurde die Frage nach einem Momentensensor zur Lenkkraftbestimmung gestellt. Die Lenkkraft ist neben Lenkwinkel- und Lenkwinkelgeschwindigkeit eine weitere Größe für die optimale Auslegung von Servolenksystemen.

Ausgehend von den am IAEQ bereits entwickelten und erfolgreich getesteten kapazitiven Winkel- und Winkelgeschwindigkeitssensoren sollte das Meßprinzip dieser Sensoren auf eine mögliche Verwendbarkeit bei der Momentenmessung untersucht werden. Ziel war es, aufbauend auf bereits vorhandenem Know How und bestehenden Soft- und Hardwarekomponenten einen Prototypen zu entwickeln.

Die mit dem Projekt Momentenmessung beschäftigten Diplomanden Stefan Cermak, Florian Wandling und Wolfgang Zdiarsky hatten folgende Teilaufgaben zu realisieren:

- [i] Entwicklung eines Meßkonzeptes
- [ii] Mechanische Anpassung der vorhandenen Meßaufbauten
- [iii] Anpassung der Hard- und Softwarekomponenten
- [iv] Testen verschiedener Rotorstrukturen
- [v] Auswertung der Meßergebnisse
- [vi] Prüfung der Einsetzbarkeit

Ziel der vorliegenden Arbeit ist die Anpassung der Hard- und Softwarekomponenten ([iii], abgehandelt in Kapitel 6 und 7). Während der Arbeit stellte sich heraus, daß eine theoretische Behandlung des Themas in Form einer Feldsimulation ebenfalls nützliche Erkenntnisse bringen kann. In Kapitel 5 ist diese Simulation vorgestellt.

In den folgenden Kapiteln werden beispielhaft verschiedene Methoden der Momentenmessung kurz vorgestellt und die Funktion des kapazitiven Momentensensors erläutert.

# Kapitel 2

## Sensoren zur Momentenmessung

Zur Messung von Momenten gibt es verschiedenste Ansätze, beginnend von der einfachen Kraftmessung an einem Hebel bestimmter Länge, bis hin zu magnetoelastischen Sensoren. Einige bisher gebräuchliche Methoden sollen hier beispielhaft erwähnt werden. Zuvor sollen aber noch die wesentlichsten Anforderungen an einen Sensor zur Momentenmessung aufgezeigt werden, wobei diese natürlich vom Einsatzgebiet abhängen. So stehen bei Sensoren für die Automobilindustrie mehr Robustheit, Zuverlässigkeit, Temperaturstabilität usw. im Vordergrund, während die Meßgenauigkeit nicht besonders hoch sein muß, sondern nur die geforderten Spezifikationen erfüllen soll.

### 2.1 Anforderungen an einen Momentensensor

Will man verschiedene Sensoren auf ihre Verwendbarkeit untersuchen, muß man zuerst eine Reihe von Merkmalen und Eigenschaften definieren, die zur objektiven Bewertung herangezogen werden können. Im Hinblick auf die gestellte Aufgabe werden folgende Anforderungen an einen Momentensensor gestellt:

- Neben der Messung von statischen Momenten, muß die Messung dynamischer Lastwechsel an einer sich drehenden Welle möglich sein.
- Arbeitsbereich, Auflösung und Linearität sind abhängig von der Aufgabenstellung.
- Temperaturbereich (Beispiel Automobilindustrie  $-40^{\circ}\text{C}$  bis  $120^{\circ}\text{C}$ ).
- Lange Lebensdauer.
- Robustheit gegen Umwelteinflüsse wie Feuchtigkeit und Verschmutzung, sowie mechanischer Beanspruchung.
- Vorgaben bezüglich der elektromagnetischen Verträglichkeit (EMV).
- Geringe aufgenommene Leistung der gesamten Sensoreinheit.

## 2.2 Verschiedene Konzepte zur Messung von Drehmomenten

### 2.2.1 Momentenmessung durch Rückführung auf einfache Kraftmessung an einem Hebel

Das Moment wird durch einen Hebel mit bestimmter Länge in eine zu messende Kraft gewandelt. Zur Kraftmessung können einfache Kraftmeßsensoren (z.B.: Dehnungsmeßstreifen (DMS)) verwendet werden. Über Längenvariation des Hebelarmes ist eine zusätzliche Meßbereichseinstellung leicht möglich. Da mit dieser Methode aber keine zu übertragenden Momente an rotierenden Wellen gemessen werden können, liegt ihr Einsatzgebiet vor allem in der Werkstoffprüfung, zum Beispiel für die Aufnahme von Spannungs/Dehnungsdiagrammen zur Materialbeurteilung bis hin zu Bruchtests.

### 2.2.2 Momentensensor mit Dehnungsmeßstreifen

Die zur Kraftübertragung benutzte Welle erfährt durch das angreifende Moment eine Verdrehung, die sich durch eine Verlängerung bzw. Verkürzung der mit der Oberfläche fest verbundenen Dehnungsmeßstreifen DMS erfassen läßt. Da die Dehnung unter einem Winkel von  $45^\circ$  zu Wellenachse maximal ist, werden die DMS unter diesem Winkel aufgeklebt, siehe Abbildung 2.1. Durch Verwendung von vier DMS in Brückenschaltung ergibt sich eine temperaturkompensierte Brückendiagonalspannung proportional zum angreifenden Moment (siehe [SRUF92]).

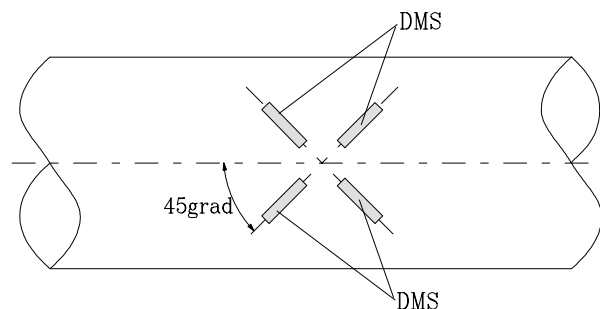


Abbildung 2.1: Positionierung von DMS bei Momentenmessung

Ein Nachteil dieser Methode besteht darin, daß es sich hier nicht um eine berührungslose Messung handelt, da die Anschlüsse des DMS-Sensors zur Auswerteschaltung geführt werden müssen. Bei Messung an rotierenden Wellen kann die Meßwertübertragung mittels Schleifkontakten realisiert werden, was wiederum eine Einschränkung in Lebensdauer (Verschleiß) und Robustheit gegen Umwelteinflüsse bedeutet. Eine andere Möglichkeit ist es, das Meßsignal über eine transformatorische Kopplung mit einem Trägerfrequenzverfahren zu übertragen. Nachteile dieser Methode sind der größere Aufwand für Schaltungsentwurf und ein erhöhter Platzbedarf.

### 2.2.3 Magnetoelastischer Momentensensor

Analog zur Messung mit Dehnungsmeßstreifen wird auch hier das Moment indirekt - durch Torsion einer Welle - gemessen.

Die Welle wird mit einem, in radialer Richtung vormagnetisierten, magnetoelastischen Ring starr verbunden. Eine Torsion der Welle bewirkt nun auch eine Formänderung im magnetoelastischen Material, welche einen axialen Magnetfeldanteil hervorruft. Die axiale Feldkomponente ist direkt proportional zum anliegenden Moment und kann berührungslos von einer Auswerteeinheit erfaßt werden (vgl. Abbildung 2.2). Die Firma Magna-lastic Devices, Inc.<sup>1</sup> hat unter Verwendung dieses Prinzips einen Sensor mit einem breiten industriellen Anwendungsspektrum entwickelt. Dieses System ist, da berührungslos, verschleißfrei. Diese Sensoren werden bereits für Lenkkraftmessungen eingesetzt.

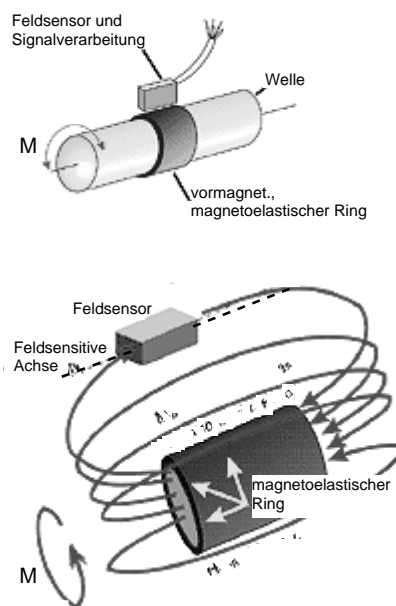


Abbildung 2.2: Prinzip des magnetoelastischen Sensors (Darstellung lt. Firma Magna-lastic Devices, Inc.)

### 2.2.4 Piezoelektrischer Momentensensor

Eine weitere Möglichkeit zur Momentenmessung ist die Erfassung der Verdrehkräfte mit Hilfe von Piezo-Elementen, vergleichbar mit der Messung der Verdrehung in Abschnitt 2.2.2. Durch die Deformation des Piezo-Elementes ändert sich seine Polarisierung, und an der Oberfläche werden Ladungen frei. Das Piezoelement läßt sich als Kondensator auffassen, der sich unter dem Einfluß der Schubspannung infolge der freigesetzten Ladung auf eine Spannung auflädt. Die Entladung erfolgt über eine Zeitkonstante  $\tau_{el}$ , die sich aus dem Produkt der Kapazität  $C_{piezo}$  und des Widerstandes  $R_{piezo}$  ergibt. Die Kapazität  $C_{piezo}$

<sup>1</sup>Magna-lastic Devices, Inc. A subsidiary of Methode Electronics, Inc. 111 W. Buchanan Street Carthage, IL 62321, Internet: [www.mdi-sensor.com](http://www.mdi-sensor.com)



und der Widerstand  $R_{piezo}$  hängen von der Elementgeometrie ab. Da das Piezoelement nur Änderungen im Moment erfassen kann, ist also eine statische Momentenmessung mit dieser Methode nicht möglich. Eine genauere Beschreibung des Piezoeffektes ist in [FAS94], Anwendungen sind beispielsweise in [SRUF92] zu finden.

### 2.2.5 Messung des Moments mit Hilfe eines Schwingsaiten-frequenzumsetzers

Die Eigenfrequenz  $f_0$  einer gespannten Saite ist laut Gleichung 2.1 (siehe [SRUF92]) abhängig von der Dehnung  $\varepsilon_{mech}$ .

$$f_0 = \frac{1}{2l_S} \sqrt{\frac{\varepsilon_{mech} E_{mech}}{\varrho_{Saite}}} \quad (2.1)$$

Außerdem gehen in diese Gleichung das Elastizitätsmodul  $E_{mech}$ , die Dichte  $\varrho_{Saite}$  und die Länge  $l_S$  der Saite ein. Mit Hilfe einer in Abbildung 2.3 gezeigten Anordnung ist es möglich, die Dehnung der beiden Meßsaiten proportional zum angreifenden Moment zu verändern, wodurch sich die Eigenfrequenzen der Meßsaiten ändern. Für die Messung der veränderten Eigenfrequenzen können die in [SRUF92] beschriebenen Aufnehmer verwendet werden. Die vormagnetisierten Saiten werden mit einem Elektromagneten durch einen Stromimpuls angeregt. Die ungedämpft ausschwingende Saite induziert eine Spannung in einer Meßspule. Die Frequenz der induzierten Spannung ist ein Maß für die Dehnungsbeanspruchung der Saiten und somit für das angreifende Moment.

Nachteil dieser Methode ist der relativ hohe Aufwand bei der mechanischen Ausführung und bei der Auswertung.

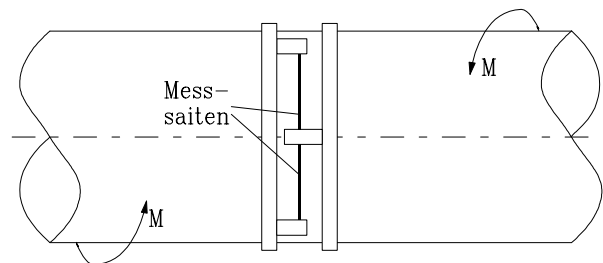


Abbildung 2.3: Anordnung von Schwingsaiten zur Momentenmessung

# Kapitel 3

## Prinzip des kapazitiven Momentensensors

### 3.1 Einleitung in die kapazitive Meßtechnik

Die Entwicklung des hier behandelten Momentensensors baut auf den bereits am IEMW entwickelten Winkelsensor [BRA92], [BRA93], [BRA96], [BRA97] sowie [FAB97] auf. In [KOEL97] und [FAB98] finden sich die zugrundeliegenden Prinzipien zusammengefaßt. Eine ausführliche theoretische Abhandlung über kapazitive Winkelsensoren ist in [JNG94] zu finden.

Für das bessere Verständnis und die Lesbarkeit dieser Arbeit werden nochmals die wesentlichsten Grundbegriffe und Konzepte vorgestellt.

#### 3.1.1 Idee des kapazitiven Sensors

Aufgabe eines jeden elektronischen Sensors ist es, die Meßgröße, beispielsweise eine mechanische Größe, in eine entsprechende elektrisch meßbare Größe umzuwandeln. In welcher Form die elektrisch meßbare Größe dann in ein weiterverarbeitbares Signal umgewandelt wird, hängt von der jeweiligen Anwendung ab, und muß an dieser Stelle nicht weiter behandelt werden.

Bei den hier besprochenen kapazitiven Sensoren erfolgt die Umwandlung der mechanischen Größe, beispielsweise des Winkels, in eine elektrisch meßbare Größe durch Veränderung der kapazitiven Kopplung zwischen einer Sender- und einer Empfangselektrode. Dabei wird zwischen Sende- und Empfangselektrode eine Mittelelektrode in Abhängigkeit von der mechanischen Größe verändert. In Abbildung 3.1 ist ein Beispiel für eine mögliche Anordnung der Elektroden, wie sie auch beim Winkelsensor laut [FAB98] verwendet wurde, dargestellt, wobei die Rotorstellung vom zu messenden Winkel abhängig ist. Die Erfassung der kapazitiven Kopplung in Abhängigkeit von der Rotorstellung ist eine meßtechnisch anspruchsvolle Aufgabe, da die relevanten Kapazitäten in der Größenordnung der vorhandenen parasitären Kapazitäten mit weit weniger als 1 pF liegen. Zur Bewältigung dieser Aufgabe kann die folgende Meßkette (Abbildung 3.2) verwendet werden. Die Einzelsegmente der Sendelektrode

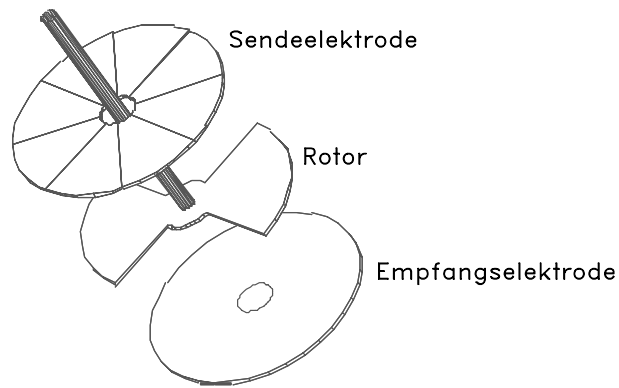


Abbildung 3.1: Beispiel für eine Elektrodenanordnung (Abb. lt. [FAB97])

werden über die Senderansteuerung mit einer Impulsfolge angesteuert, je nach Koppelkapazität wird ein unterschiedlich großes Signal an die Empfängerelektrode übertragen und über einen Empfangsschwingkreis abgenommen. Das so empfangene Signal wird dann im Analogteil verstärkt, gefiltert, gleichgerichtet und anschließend AD-gewandelt. Im Digitalteil wird mit Hilfe eines Algorithmus nach dem ratiometrischen Prinzip aus den Empfangssignalen und den dazugehörigen Segmentwerten auf die zu messende Größe, in diesem Beispiel der Winkel, zurückgeschlossen.

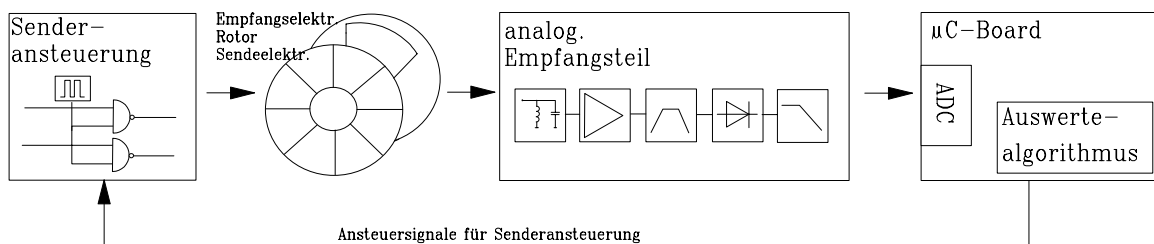


Abbildung 3.2: Blockschaltbild eines kapazitiven Sensors (Meßkette und Auswerteeinheit)

### 3.1.2 Ratiometrisches Prinzip

Die Kapazität der Elektrodenstruktur sowie etwaig vorhandene parasitäre Stromwege werden stark von den äußeren Bedingungen beeinflusst, da die wirksame relative Permittivität  $\varepsilon_R$  lt. Gleichung 3.6 von z. B. der Luftfeuchtigkeit ( $\frac{\varepsilon_{R-Wasser}}{\varepsilon_{R-Luftideal}} = 81$ ) und der Temperatur abhängig ist. Bedingt durch diese Effekte stellt sich keine definierte rein kapazitive Kopplung ein und man erhält somit keinen definierten Signalpegel am Empfänger. Durch äußere Einflüsse, vor allem durch die Temperaturabhängigkeit, ist auch die Spannungsüberhöhung am Schwingkreis nicht konstant. Diese Effekte sind aber im Vergleich zum Meßzyklus so langsam, daß ihr Einfluß innerhalb eines Meßzyklus, d. h. während der Messung der 16 bzw. 32 Segmentwerte, als konstant angenommen werden kann. Da sich die für die Messung entscheidenden Parameter (Signalpegel und Verstärkung) aber von Meßreihe zu Meßreihe ändern können, ist ein Auswertalgorithmus notwendig, der gegenüber diesen Fehlern unempfindlich ist. Das verwendete ratiometrische Prinzip ist dafür sehr gut geeignet, weil es ein vergleichendes, nicht von den absoluten Werten abhängiges Prinzip ist.

Wird davon ausgegangen, daß aus den oben genannten Gründen Signalloffset  $y_{offset}$  und Verstärkung  $k$  nicht bekannt sind, so bildet sich die zu messende Größe  $y_{ist1}$  auf die gemessene Größe  $y_{mess1}$  wie folgt ab:

$$y_{mess1} = y_{ist1} \cdot k + y_{offset}$$

Werden zwei weitere Meßpunkte  $y_{mess2}$  und  $y_{mess3}$  aufgenommen, so ergibt sich das folgende Gleichungssystem mit den zugehörigen Istwerten  $y_{ist2}$  und  $y_{ist3}$ :

$$\begin{aligned} y_{mess1} &= y_{ist1} \cdot k + y_{offset} \\ y_{mess2} &= y_{ist2} \cdot k + y_{offset} \\ y_{mess3} &= y_{ist3} \cdot k + y_{offset} \end{aligned} \quad (3.1)$$

Da nur die Meßwerte  $y_{mess1}$ ,  $y_{mess2}$  und  $y_{mess3}$  bekannt sind, besteht das Gleichungssystem 3.1 aus drei Gleichungen mit fünf Unbekannten. Um das Gleichungssystem trotzdem eindeutig lösen zu können, ist es notwendig, zwei der fünf Unbekannten zu ermitteln. Beim hier verwendeten ratiometrischen Prinzip wird dieses Problem dadurch gelöst, daß in den Messungen 2 und 3 bekannte Referenzgrößen erfaßt werden. In der hier betrachteten Aufgabe, Erfassen eines Kapazitätsverlaufes (vgl. Kapitel 3.1.3), sind die Referenzgrößen Maximalwert und Minimalwert der Kapazität besonders leicht zugänglich, da sie in gewissen Bereichen (vgl. Bereich  $S_6$  und  $S_7$  bzw. Bereich  $S_2$  und  $S_3$  in Abbildung 3.3) in erster Näherung unabhängig von der Rotorstellung sind.

Werden nun Maximalwert  $y_{istmax} = y_{ist2}$  und Minimalwert  $y_{istmin} = y_{ist3}$  gemessen, so erhält man  $y_{messmax} = y_{mess2}$  und  $y_{messmin} = y_{mess3}$ . Aus Gleichungssystem 3.1 wird nun:

$$\begin{aligned} y_{mess1} &= y_{ist1} \cdot k + y_{offset} \\ y_{messmax} &= y_{istmax} \cdot k + y_{offset} \\ y_{messmin} &= y_{istmin} \cdot k + y_{offset} \end{aligned} \quad (3.2)$$

Durch Umformung des Gleichungssystems 3.2 ergibt sich:

$$\begin{aligned} \frac{y_{ist1} - y_{istmin}}{y_{istmax} - y_{istmin}} &= \frac{y_{mess1} - y_{messmin}}{y_{messmax} - y_{messmin}} \text{ bzw.} \\ y_{ist1} &= \frac{y_{mess1} - y_{messmin}}{y_{messmax} - y_{messmin}} \cdot (y_{istmax} - y_{istmin}) + y_{istmin} \end{aligned} \quad (3.3)$$

Wie aus den Gleichungen 3.1, 3.2 und 3.3 hervorgeht, ist es mit Hilfe von drei Meßpunkten möglich, additive und multiplikative Fehler zu eliminieren. Werden vier Meßpunkte aufgenommen, kann diese Redundanz für eine weitere Fehlerverminderung verwendet werden. Dazu ist es notwendig, neben den ersten drei Meßpunkten noch einen weiteren zu finden, der in Hinblick auf die spätere Anwendung mit negativer Steigung, d. h. sinkender Meßwert bei steigendem Istwert, gewählt wird (vgl. sinkender Kapazitätswert  $C_{S1E}$  in Abbildung 3.4). Dieser Meßpunkt wird zum besseren Verständnis  $y_{minus}$  genannt, der Meßpunkt  $y_{mess1}$  in  $y_{plus}$  umbenannt. Für die Meßpunkte  $y_{plus}$  und  $y_{minus}$  gilt:

$$\begin{aligned} y_{plus} &= +y_{ist1} \cdot k + y_{offset} \\ y_{minus} &= -y_{ist1} \cdot k + y_{offset} \end{aligned}$$

Nach Anschreiben und Umformung des neu entstandenen Gleichungssystems 3.4, läßt sich eine Meßformel 3.5 finden, die als Ergebnis eine auf das Intervall  $[-1, 1]$  beschränkte Interpolationsgröße  $\xi_{interpol}$  liefert.

$$\begin{aligned} y_{messplus} &= +y_{ist1} \cdot k + y_{offset} \\ y_{messminus} &= -y_{ist1} \cdot k + y_{offset} \\ y_{messmax} &= y_{istmax} \cdot k + y_{offset} \\ y_{messmin} &= y_{istmin} \cdot k + y_{offset} \end{aligned} \quad (3.4)$$

$$\xi_{interpol} = \frac{y_{messplus} - y_{messminus}}{y_{messmax} - y_{messmin}} \quad (3.5)$$

Die Anwendung dieser formalen Überlegungen und die Vorteile, die sich aus der Anwendung der 4-Punkt-Meßmethode ergeben, werden in Kapitel 3.2.3 gezeigt.

### 3.1.3 Erklärung des Sensorprinzipes anhand des Plattenmodelles

Bei den folgenden Betrachtungen wird von einem idealen Plattenkondensatormodell unter Vernachlässigung der Streufelder ausgegangen. Dieses Plattenkondensatormodell kann auch als Abwicklungsdarstellung der in Abbildung 3.1 gezeigten Elektrodenstruktur betrachtet werden. Die Kapazität  $C$  des Plattenkondensators ist dann (vgl. [PRE94]):

$$C = \varepsilon \cdot \frac{A}{d} \quad (3.6)$$

Wobei  $A$  die Elektrodenfläche,  $d$  der Elektrodenabstand und  $\varepsilon$  die Permittivität bezeichnen. Üblicherweise wird die Permittivität  $\varepsilon$  als Produkt zwischen der relativen Permittivität  $\varepsilon_R$  und der elektrischen Feldkonstante  $\varepsilon_0$  dargestellt ( $\varepsilon = \varepsilon_0 \cdot \varepsilon_R$ ).

Für weitere Beschreibungen ist es sinnvoll, den Begriff der längenbezogenen Kapazität oder auch Kapazitätsbelag  $C'$  zu verwenden. Hierbei wird davon ausgegangen, daß sich die wirksame Elektrodenfläche  $A$  aus dem Produkt der Tiefe  $a$  und Breite  $x_S$  der Elektroden ergibt. Wird die Tiefe  $a$  als konstant angenommen, ergibt sich:

$$\begin{aligned} C' &= \varepsilon \cdot \frac{a}{d} \\ C &= C' \cdot x_S \end{aligned} \quad (3.7)$$

Durch Einbringung einer geerdeten Mittelelektrode – im folgenden Rotor ( $R$ ) genannt – wird die wirksame Elektrodenbreite  $x_S$  und somit die Kapazität  $C$  geändert. Für die weiteren Betrachtungen wird von einer in Abbildung 3.3 schwarz dargestellten Plattenstruktur ( $S_1, E_1, R$ ) ausgegangen. Der Rotor besteht aus zwei Platten der Breite  $4 \cdot x_S$  in einem Abstand von ebenfalls  $4 \cdot x_S$ . Eine räumliche Periode von  $x_{PER} = 8 \cdot x_S$  wird für noch folgende Betrachtungen eingeführt. Zur Lagebeschreibung des Plattenkondensators in Bezug auf den

Rotor wird  $x_H$  als horizontale Verschiebung der Freistellung zwischen den beiden Rotorplatten und dem Nullpunkt definiert. Der Nullpunkt ist in die Mitte der Kondensatorplatten ( $S_1$ ,  $E_1$ ) gelegt. Die Kapazität  $C_{S1E}(x_H)$  zwischen diesen beiden Kondensatorplatten  $S_1$  und  $E_1$  lässt sich in Abhängigkeit von der Rotorstellung durch folgende Gleichung ausdrücken:

$$C_{S1E}(x_H) = C' \int_{\frac{-x_S}{2}}^{\frac{+x_S}{2}} \text{rect}\left(\frac{x - x_H}{4 \cdot x_S}\right) dx \quad (3.8)$$

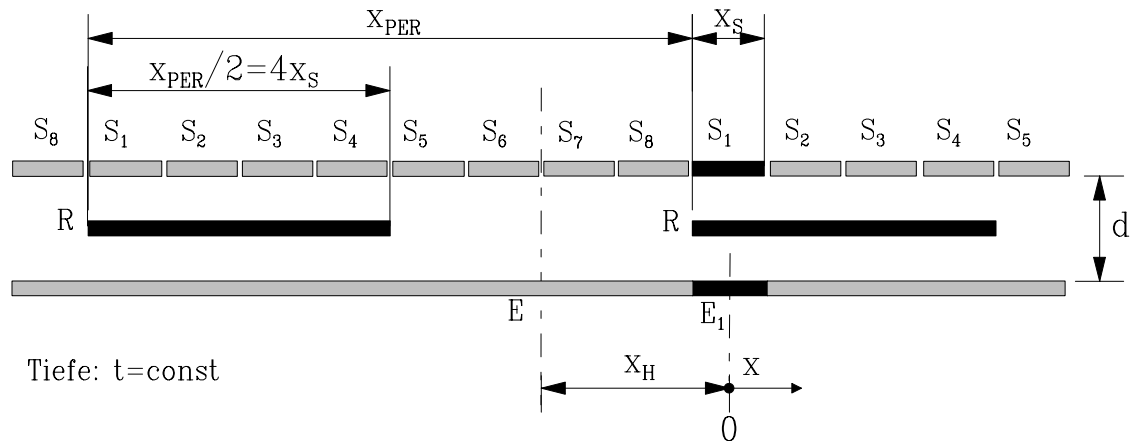


Abbildung 3.3: Prinzipielle Plattenanordnung bestehend aus Sendekondensatorplatten  $S_n$ , Empfangskondensatorplatte  $E$  und in  $x$ -Richtung verschiebbarem Rotor  $R$ .

Der durch Gleichung 3.8 ausgedrückte Kapazitätsverlauf  $C_{S1E}(x_H)$  ist in Abbildung 3.4 dargestellt und läßt vier charakteristische Bereiche erkennen:

1. Bereich: keine Kapazität  
 $|x_H| > 2.5x_S :$   
 $C_{S1E}(x_H) = 0$
2. Bereich: konstante Kapazitätzunahme  
 $-2.5x_S \leq x_H \leq -1.5x_S :$   
 $C_{S1E}(x_H) = C'' \cdot (+2.5x_S + x_H)$
3. Bereich: maximale Kapazität  
 $|x_H| < 1.5x_S :$   
 $C_{S1E}(x_H) = C_{SEmax} = C'' \cdot x_S$  (vgl. Gleichung 3.7)
4. Bereich: konstante Kapazitätsabnahme  
 $+1.5x_S \leq x_H \leq +2.5x_S :$   
 $C_{S1E}(x_H) = C'' \cdot (-2.5x_S + x_H)$

Erweitert man die hier beschriebene Anordnung derart, daß mehrere Sendekondensatorplatten  $S_n$  aneinandergereiht werden und eine gemeinsame Empfängerplatte  $E$  gebildet wird (vgl. Abbildung 3.3 hellgrau eingezeichnet), so führt diese räumliche Fortsetzung dazu, daß

sich zu jeder Rotorposition  $x_H$  ein Satz von Segmenten  $S_i$  finden läßt, welcher die zuvor beschriebenen vier Bereiche abdeckt. Der erhaltene Satz von Segmenten erfüllt die in Kapitel 3.1.2 beschriebenen Anforderung für die Anwendung des ratiometrischen Prinzips, wodurch es möglich wird, die Rotorposition durch Anwendung der Gleichung 3.5 wie folgt zu ermitteln.

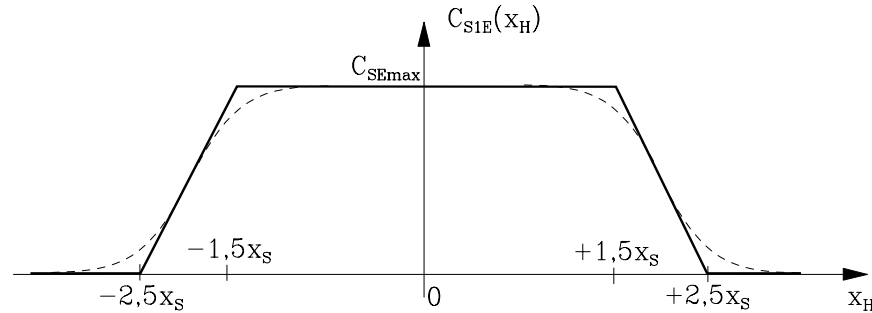


Abbildung 3.4: Kapazitätsverlauf von  $C_{S1E}(x_H)$  ideal: durchgezogene Linie, Einfluß der Streufelder: strichlierte Linie.

Die bisher vernachlässigten Streufelder verringern den linearen Bereich, wie in Abbildung 3.4 strichliert angedeutet ist. Werden die Streufelder zweier benachbarter Segmente überlagert, so ergibt sich wieder ein homogenes Feld, der lineare Bereich wird dadurch vergrößert. Um die Rotorposition möglichst genau bestimmen zu können, werden daher zu jedem der vier charakteristischen Bereiche möglichst viele passende Segmente zu Segmentgruppen zusammengefaßt. Für die in Abbildung 3.3 gezeigte Rotorposition ergibt sich damit:

$$\begin{aligned}
 y_{messplus} &= C_{S7E} + C_{S8E} + C_{S1E} + C_{S2E} \\
 y_{messminus} &= C_{S3E} + C_{S4E} + C_{S5E} + C_{S6E} \\
 y_{messmax} &= 2 \cdot (C_{S6E} + C_{S7E}) \\
 y_{messmin} &= 2 \cdot (C_{S2E} + C_{S3E})
 \end{aligned}$$

Für die Bestimmung von  $y_{messmax}$  und  $y_{messmin}$  können aufgrund der Streufeldverläufe nur zwei Segmente gewählt werden. Zur betragsmäßigen Anpassung ist daher eine Multiplikation mit dem Faktor 2 nötig. Durch Anwendung der Gleichung 3.5 wird nun das Intervall der Rotorposition  $[-2 \cdot x_S, +2 \cdot x_S]$  auf das Intervall  $[-1, 1]$  abgebildet.

Um die gewonnen Erkenntnisse und den beschriebenen Algorithmus zur Bestimmung der Rotorposition in einem autonom arbeitenden Meßprogramm absetzen zu können, ist es notwendig, die in diesem Fall durch Überlegungen gewählten Segmente, ebenfalls automatisch bestimmen zu können. Hierzu dient ein in Kapitel 3.2.3.3 beschriebener Algorithmus zur Bestimmung der ungefähren Rotorlage.

### 3.1.4 Anwendung auf den Winkelsensor

Der am IEMW bereits entwickelte Winkelsensor laut [BRA92] baut auf das in Kapitel 3.1.3 beschriebene Sensorgrundprinzip auf. Für den Winkelsensor wurden die Elektroden rotationsymmetrisch angeordnet (vgl. Abbildung 3.1), wodurch es möglich ist, anstelle der Position

$x_H$  den Winkel zu messen. In [BRA92] und [FAB98] ist die Anwendung auf Winkel- und Winkelgeschwindigkeitssensoren eingehend behandelt. Eine praktische Realisierung ist in [KOEL97] dokumentiert.

## 3.2 Konzept der Momentenmessung

Wird mit Hilfe einer Welle ein Moment übertragen, so erfährt die Welle eine Verdrehung, die entsprechend der Materialeigenschaften in direktem Zusammenhang mit dem Betrag des angreifenden Moments liegt. Gelingt es, die Verdrehung unter Berücksichtigung verschiedener Einflußgrößen, wie z. B. Temperatur und Alterung zu messen und ein entsprechendes Kennfeld der Torsionswelle aufzunehmen, so ist eine Momentenbestimmung möglich. Ausgehend von dem Konzept des Winkel- und Winkelgeschwindigkeitssensors (vgl. Kapitel 3.1.4) wurde ein Relativwinkelsensor gebaut, der die beschriebene Verdrehung messen kann.

### 3.2.1 Torsion einer mechanischen Welle

Geht man davon aus, daß bei Verdrehung einer Welle durch das angreifende Moment der Gültigkeitsbereich des Hook'schen Gesetzes nicht verlassen wird, d. h. ein linearer Zusammenhang zwischen Torsionsspannung  $\tau_t$  und Formänderung  $v$  besteht, so gilt laut [BOEG85]:

Zwei benachbarte Querschnitte einer Welle (vgl. Abbildung 3.5) werden durch Torsionsbeanspruchung gegeneinander verdreht. Wird vor der Verformung der Welle der Länge  $l$  eine Markierungslinie zwischen den Punkten  $A$  und  $B$  angenommen, dann wird daraus nach der Verformung die Schraubenlinie  $\overline{AC}$ . Zugleich dreht sich der Radius  $\overline{OB}$  um den Kreismittelpunkt  $O$  in die Stellung  $\overline{OC}$ , das heißt, die beiden Stirnflächen der Welle haben sich um den Verdrehwinkel  $\varphi_{rel}$  gegeneinander verdreht. Die stärkste Verformung zeigt die Randfaser - Formänderung  $v$  (Bogen  $BC$ ). Das Hook'sche Gesetz für Torsion lautet:

$$\tau_t = \frac{v}{l} \cdot G \quad \text{bzw.} \quad (3.9)$$

$$\varphi_{rel} = \frac{\tau_t l}{Gr} \cdot \frac{180^\circ}{\pi} \quad (3.10)$$

Wobei für die Torsionsspannung  $\tau_t$  der Quotient aus angreifenden Moment  $M$  zu polarem Widerstandsmoment  $W_P$  eingesetzt wird (vgl. Gleichung 3.11) und  $G$  das Schubmodul ist. Das polare Widerstandsmoment  $W_P$  ist eine geometrieabhängige Größe und läßt sich für eine massive Welle mit Radius  $r$  nach Gleichung 3.12 berechnen.

$$\tau_t = \frac{M}{W_P} \quad (3.11)$$

$$W_P = \frac{\pi}{16} \cdot (2r)^3 \quad (3.12)$$



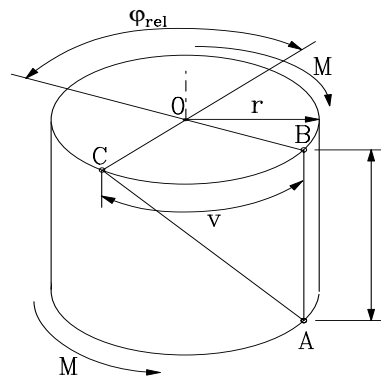


Abbildung 3.5: Formänderung bei Torsionsspannung lt. [BOEG85]

**Beispiel für eine Torsionswelle:** Eine Torsionswelle aus Stahl mit einem Schubmodul  $G = 80.000 \text{ N/mm}^2$ , mit einer Wellenlänge  $l = 20 \text{ cm}$  und einem Radius  $r = 2.5 \text{ mm}$  ergibt sich bei einem Moment  $M = 2 \text{ Nm}$  ein Verdrehwinkel  $\varphi_{rel} = 4.66^\circ$ . Das Moment entspricht einer Lenkkraft von  $10 \text{ N}$  auf einem Lenkraddurchmesser von  $40 \text{ cm}$ .

### 3.2.2 Mechanischer Aufbau

Die Idee des kapazitiven Momentensensors ist es, anstelle eines Rotors beim Winkelsensor zwei Rotoren zwischen Sende- und Empfangselektrode einzubringen. Diese Rotoren werden nun mit dem Verdrehwinkel  $\varphi_{rel}$  (vgl. Kapitel 3.2.1) relativ zueinander verdreht. Wie in Kapitel 3.2.3 dargestellt wird, läßt sich unter Verwendung des ratiometrischen Prinzips ein Algorithmus finden, der sowohl den Relativwinkel  $\varphi_{rel}$  zwischen den beiden Rotoren als auch den Absolutwinkel  $\varphi_{abs}$  erfassen kann.

Aus den bisherigen Überlegung ergeben sich für die Realisierung eines Momentensensors folgende mechanische Bedingungen:

Der durch das angreifende Moment entstehende Verdrehwinkel  $\varphi_{rel}$  der Torsionswelle, muß als Relativverdrehung zwischen den zwei Rotoren wirken. Der Abstand zwischen den Elektroden (Senderelektrode, Rotor 1, Rotor 2, Empfängerelektrode) muß aber so gering wie möglich gehalten werden, da sonst die Kopplung zu klein und der Einfluß der Streufelder zu groß wird (vgl. nichtlinearer Verlauf der Kapazität  $C_{S1E}(x_H)$  in Abbildung 3.4). Dieser Forderung steht die Tatsache entgegen, daß der Verdrehwinkel  $\varphi_{rel}$  der Länge  $l$  der Welle direkt proportional ist (vgl. Gleichung 3.10). Um einen meßbaren Verdrehwinkel  $\varphi_{rel}$  zu bekommen, wird bei der Messung mit Torsionswellen die Verdrehung zwischen Wellenanfang und -ende gemessen. Damit die geringe Distanz der beiden Rotoren trotzdem eingehalten werden kann, werden zwei konzentrische Wellen verwendet, bei der die innere Welle die Torsionswelle ist. Die äußere Welle (Hohlwelle) ist nur am Wellenende mit der Torsionswelle fix verbunden und bleibt so unbelastet. Erfährt die Torsionswelle nun eine Verdrehung, so ist die Relativverdrehung der äußeren Welle zur Torsionswelle gleich dem Verdrehwinkel  $\varphi_{rel}$ . Wird nun ein Rotor mit der Torsionswelle und der zweite Rotor mit der äußeren Welle fix verbunden, so dient diese äußere Welle als mechan. "Rückführung" des Verdrehwinkels  $\varphi_{rel}$  (siehe Abbildung 3.6).

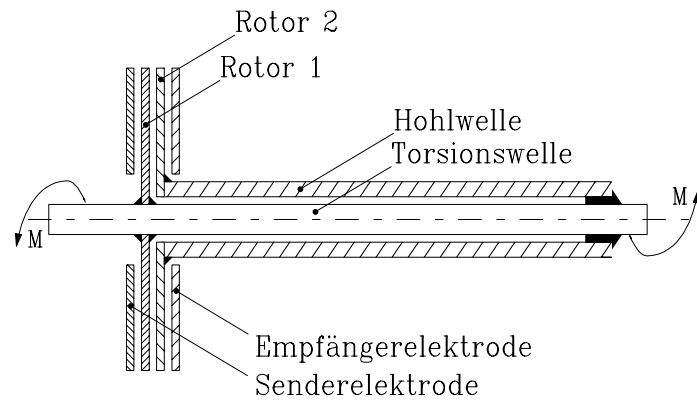


Abbildung 3.6: Mechanische "Rückführung" des Verdrehwinkels  $\varphi_{rel}$

Das Prinzip zweier konzentrischer Wellen findet sich auch bei Lenksäulen im Automobilbau wieder. Das Lenkrad ist über eine Torsionswelle mit der Lenkmechanik verbunden. Um einen Totalausfall der Lenkung bei Bruch der Torsionswelle zu verhindern, wird die Welle in einer Hohlwelle geführt. Diese ist beim Lenkrad fest mit der Lenkwelle verbunden und am Wellenende über einen mechanischen Totgang mit der Lenkwelle verkoppelt. Bei Bruch der Torsionswelle kann die Hohlwelle die Lenkkräfteübertragung gewährleisten, wobei sich ein zusätzliches Lenkspiel von ca.  $5^\circ$  bemerkbar macht.

Unter Rücksichtnahme auf eine eventuelle Anwendung des Sensors im Automobilbau, wurde ein Prüfstand entwickelt, der auf diese Struktur aufbaut, der in [WAN99] eingehend beschrieben ist.

### 3.2.3 Elektrodenstruktur und Auswertung

Da der maximale Verdrehwinkel  $\varphi_{rel}$  von Torsionswellen je nach Länge und Durchmesser nur wenige Grad beträgt, vgl. Kapitel 3.2.1, ist eine Relativwinkelmessung auf  $\pm 5^\circ$  ausreichend.

Am Beispiel der in Abbildung 3.7 gezeigten Elektrodenstruktur, bestehend aus Sender- und Empfangsfläche, sowie den beiden Rotoren, wird die Relativwinkelmessung erklärt. Die Elektrodenstruktur zeigt eine Senderelektrode mit 16 Segmenten (Segmentteilung  $22.5^\circ$ ) und zwei asymmetrische, zueinander gespiegelte Rotorflügel (Öffnungswinkel  $90^\circ$  und  $150^\circ$ , Flügel je  $60^\circ$ ). In [ZDI99] sind noch andere, weiterentwickelte Rotorstrukturen gezeigt, die mit einer 32-segmentigen Senderelektrode bessere Ergebnisse erzielen. Zum besseren Verständnis wird an dieser Stelle die einfachere 16 Segmentstruktur behandelt, welche aber auf den gleichen Prinzipien beruht.

Wird die in Abbildung 3.7 gezeigte Elektrodenstruktur, in einer "Abwicklungsdarstellung", vergleichbar mit der Darstellung von Wicklungsschemata bei Motoren, zweidimensional dargestellt, so ergibt sich die in Abbildung 3.8 gezeigte Anordnung, vergleichbar mit der Plattenanordnung in Abbildung 3.3.

Um die Beschreibung des Algorithmus zu vereinfachen, wird das folgende Gedankenmodell verwendet. In diesem idealisierten Modell wird von einer homogenen Feldverteilung ausgegangen. Unter der Voraussetzung, daß die Feldlinien nur in axialer Richtung (vgl.

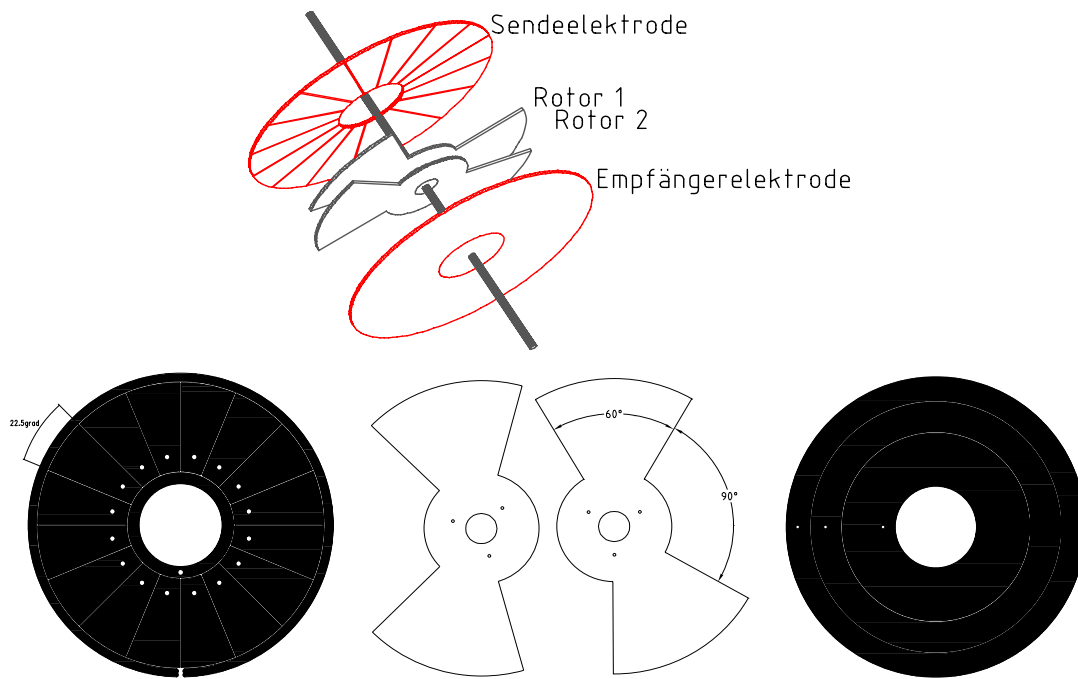


Abbildung 3.7: Elektrodenstruktur des Relativwinkelsensors (oben: 3D-Ansicht, unten von links nach rechts: Senderelektrode, Rotor 1, Rotor 2, Empfängerelektrode)

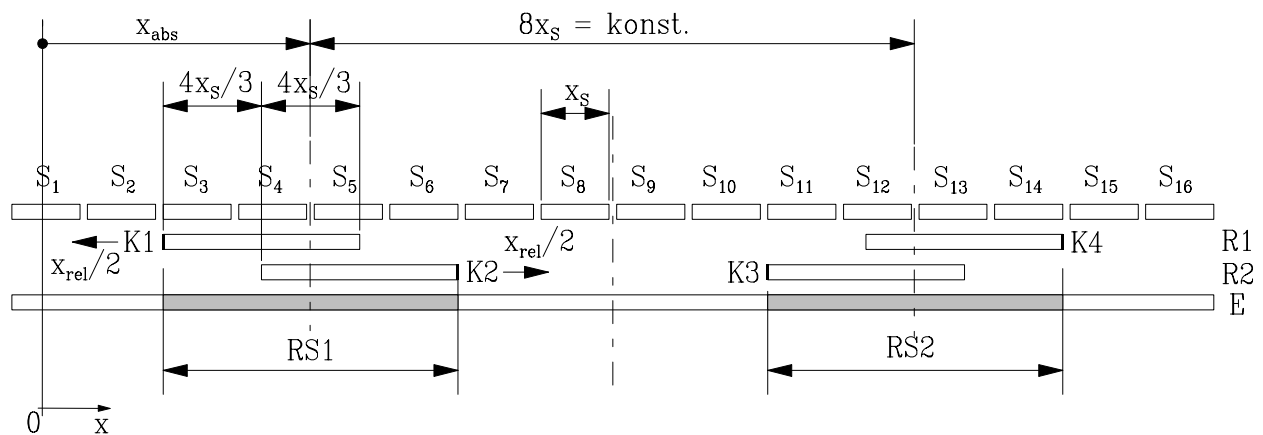


Abbildung 3.8: "Abwicklungsdarstellung" der Elektrodenstruktur aus Abbildung 3.7 in Null-lage ( $x_{rel} = 0$  bzw.  $\varphi_{rel} = 0$ )

z-Richtung in Abbildung 3.8) verlaufen, erhält man eine Projektion der Rotorflächen auf die Empfängerfläche  $E$ , vergleichbar mit dem Schattenbild bei planparallelem Lichteinfall. Durch Ansteuerung der 16 Segmente  $S_i$  wird das Schattenbild des Rotors auf die Empfängerfläche abgebildet, wobei jeder der 16 Meßwerte die mittlere Helligkeit erfaßt (entspricht der wirksamen Kondensatorfläche). In Abbildung 3.8 sind die Schattenbilder  $RS_1$ ,  $RS_2$  der Rotoren  $R_1$ ,  $R_2$  eingezeichnet. Ändert sich der Absolutwinkel  $\varphi_{abs}$ , werden die Rotorschatten in ihrer Lage verändert, eine Relativwinkeländerung bewirkt eine Änderung der Breite der Rotorschatten.

Wird das verwendete Gedankenmodell nun auf die Zusammenhänge beim Momentensensor angewendet, so bedeutet das:

Abhängig von der Rotorstellung ergeben sich, durch die Ansteuerung der Segmente, 16 Meßwerte - im folgenden Segmentwerte  $SW$  genannt. In der verwendeten Sensorelektronik sind diese Meßwerte Spannungen, die proportional der Einkoppelkapazität (vgl. Helligkeitswert) der einzelnen Sendesegmente in den Empfängerschwingkreis sind (vgl. Sensorelektronik, Empfängerschaltung lt. [WAN99]).

### 3.2.3.1 Relativwinkelbestimmung

Für die Schattenbildung sind nur die Rotorkanten  $K_1$ ,  $K_2$ ,  $K_3$  und  $K_4$  maßgebend, da die übrigen Rotorkanten vom jeweils anderen Rotor überdeckt werden (vgl. Abbildung 3.8). Werden nun die Rotoren  $R_1$  und  $R_2$  um je  $x_{rel}/2$  gegeneinander verschoben, so vergrößert sich die Distanz  $\overline{K_1K_2}$  um  $x_{rel}$  (Verbreiterung des Rotorschattens  $RS_1$ ), während sich  $\overline{K_3K_4}$  um  $x_{rel}$  (Verschmälerung des Rotorschattens  $RS_2$ ) verringert. Die Mittenlagen der Rotorschatten  $RS_1$  und  $RS_2$  bleiben von dieser Relativverschiebung unbeeinflusst.

Werden nun die Segmentwerte  $SW_1$  bis  $SW_8$  zum Meßwert  $y_{relminus}$  und die Segmentwerte  $SW_9$  bis  $SW_{16}$  zum Meßwert  $y_{relplus}$  zusammengezählt, sind schon zwei der vier Größen für die Anwendung der ratiometrischen Gleichung (siehe Gleichung 3.5) gefunden. Um den Maximalwert  $y_{relmax}$  zu erhalten, werden die Segmentwerte der unbedeckten Segmente ( $SW_1$ ,  $SW_8$ ,  $SW_9$ ,  $SW_{16}$ ) verwendet und mit dem Faktor 2 zur betragsmäßigen Anpassung multipliziert. Analoges gilt für den Minimalwert  $y_{relmin}$ , angewandt auf die voll bedeckten Segmente ( $SW_4$ ,  $SW_5$ ,  $SW_{12}$ ,  $SW_{13}$ ). Es ergibt sich der folgende Gleichungssatz:

$$\begin{aligned} y_{relminus} &= SW_1 + SW_2 + SW_3 + SW_4 + SW_5 + SW_6 + SW_7 + SW_8 \\ y_{relplus} &= SW_9 + SW_{10} + SW_{11} + SW_{12} + SW_{13} + SW_{14} + SW_{15} + SW_{16} \\ y_{relmax} &= 2 \cdot (SW_1 + SW_8 + SW_9 + SW_{16}) \\ y_{relmin} &= 2 \cdot (SW_4 + SW_5 + SW_{12} + SW_{13}) \end{aligned}$$

Dieser Formelsatz läßt sich nun in Gleichung 3.5 einsetzen, um den Relativwinkelinterpolationswert zu erfassen (vgl. Rotorpositionserfassung Kapitel 3.1.3):

$$\xi_{rel} = \frac{y_{relplus} - y_{relminus}}{y_{relmax} - y_{relmin}} \quad (3.13)$$

Mit Hilfe der Gleichung 3.13 wird der Relativverschiebungsbereich  $[-4 \cdot x_s, +4 \cdot x_s]$  auf den Bereich  $[-1, +1]$  abgebildet. Hierbei sind allerdings noch Nebenbedingungen für die maximale bzw. minimale Ausdehnung der Rotorschatten  $RS_1$  und  $RS_2$  zu beachten, welche die Gültigkeit des angegebenen Bereiches auf das Intervall  $[-4/3 \cdot x_s, +4/3 \cdot x_s]$  einschränken. Die Nebenbedingungen ergeben sich aus den folgenden Überlegungen (vgl. Abbildung 3.8): Der Rotorschatten  $RS_1$  kann eine maximale Breite  $\overline{K_1K_2}_{max} = 4 \cdot 4/3 \cdot x_s$  annehmen, da sonst der Rotorschatten  $RS_1$  in zwei Teile zerfallen würde. Der Rotorschatten  $RS_2$  kann eine minimale Breite  $\overline{K_3K_4}_{min} = 2 \cdot 4/3 \cdot x_s$  annehmen, da sonst der Rotorschatten  $RS_2$  sich wieder verbreitern würde.

Da in der Abwicklungsdarstellung die Segmentbreite  $x_s$  einem Winkel von  $22.5^\circ$  in der rotationssymmetrischen Elektrodenstruktur entspricht, liegt der tatsächliche Meßbereich im Intervall  $[-30^\circ, +30^\circ]$ .

In den bisherigen Betrachtungen wurde auf das Vorzeichen der Relativverschiebung ( $\pm x_{rel}$  bzw.  $\pm \varphi_{rel}$  bzw. positives/negatives Moment) noch nicht näher eingegangen. Ausgehend von der Abbildung 3.8 mit der angenommenen Nulllage ( $x_{rel} = 0$  bzw.  $\varphi_{rel} = 0$ ) läßt sich nun das Vorzeichen definieren: Positive Relativverschiebung  $+x_{rel}$  bedeutet Vergrößerung des Rotorschattens  $RS_1$ , und Verkleinerung des Rotorschattens  $RS_2$ , negative Relativverschiebung  $-x_{rel}$  bedeutet Verkleinerung des Rotorschattens  $RS_1$  und Vergrößerung des Rotorschattens  $RS_2$ .

Um das Vorzeichen der Verschieberichtung bestimmen zu können, ist also eine eindeutige Unterscheidung zwischen den Rotorschatten  $RS_1$  und  $RS_2$  notwendig. Dazu werden die Rotoren  $R_1$  und  $R_2$  gegeneinander vorverdrehen, sodaß in der neuen Nulllage der Rotorschatten  $RS_1$   $4 \cdot x_S + 2/3 \cdot x_S$  ( $105^\circ$ ) und Rotorschatten  $RS_2$   $4 \cdot x_S - 2/3 \cdot x_S$  ( $75^\circ$ ) breit sind.

Durch die unterschiedliche Breite der Rotorschatten in der neu angenommenen Nulllage ist somit eine eindeutige Unterscheidung von Rotorschatten  $RS_1$  und Rotorschatten  $RS_2$  möglich, wobei der verbleibende Meßbereich von  $\pm 15^\circ$  nun schmaler ist. Der Meßbereich liegt aber immer noch über dem geforderten von  $\pm 5^\circ$ .

### 3.2.3.2 Absolutwinkelbestimmung

Wie schon erwähnt, bleibt die absolute Position  $x_{abs}$  der Mittelpunkte der Rotorschatten  $RS_1$  und  $RS_2$  von einer Relativverschiebung unbeeinflusst. Der Abstand zwischen den Mittelpunkten der Rotorschatten  $RS_1$  und  $RS_2$  kann als  $8 \cdot x_S = konst.$  angegeben werden (vgl. Abbildung 3.8).

Aufgrund dieser Tatsache ist es nun möglich, den von den Segmenten  $S_9$  bis  $S_{16}$  überdeckten Bereich gedanklich über den der Segmente  $S_1$  bis  $S_8$  zu legen. Die in Abbildung 3.8 gezeigte Anordnung erscheint dann nur noch acht Segmente breit, wobei die Mittelpunkte der Rotorschatten immer übereinanderliegen. Bei einer Absolutverschiebung von  $x_{abs}$  liegen die Rotorkanten  $K_1$ ,  $K_2$ ,  $K_3$  und  $K_4$  bei:

$$\begin{aligned} x_{K1} &= x_{abs} - 2 \cdot x_S - \frac{x_{rel}}{2} \\ x_{K2} &= x_{abs} + 2 \cdot x_S + \frac{x_{rel}}{2} \\ x_{K3} &= x_{abs} - 2 \cdot x_S + \frac{x_{rel}}{2} \\ x_{K4} &= x_{abs} + 2 \cdot x_S - \frac{x_{rel}}{2} \end{aligned}$$

Betrachtet man nun die Mittenposition von  $x_{K1}$  und  $x_{K3}$  sowie  $x_{K2}$  und  $x_{K4}$  so ergibt sich:

$$\begin{aligned} \frac{x_{K1} + x_{K3}}{2} &= x_{abs} - 2 \cdot x_S \\ \frac{x_{K2} + x_{K4}}{2} &= x_{abs} + 2 \cdot x_S \end{aligned}$$

Diese beiden Mittenpositionen beschreiben die Kanten eines fiktiven Rotor der Breite  $4 \cdot x_S$ , womit sich eine Anordnung wie in Abbildung 3.3 aus Kapitel 3.1.3, über das allgemeine Plattenmodell erkennen läßt.

Um die absolute Lage der Rotoren  $x_{abs}$  in Bezug auf die Sendesegmente und damit den Absolutwinkelinterpolationswert  $\xi_{abs}$  zu bestimmen, können nun wieder die für die Anwendung des ratiometrischen Prinzips notwendigen Größen gebildet werden. Für die in Abbildung 3.8 gezeigte Rotorposition sind das:

$$\begin{aligned} y_{absplus} &= (SW_1 + SW_9) + (SW_2 + SW_{10}) + (SW_3 + SW_{11}) + (SW_4 + SW_{12}) \\ y_{absminus} &= (SW_5 + SW_{13}) + (SW_6 + SW_{14}) + (SW_7 + SW_{15}) + (SW_8 + SW_{16}) \\ y_{absmax} &= 2 \cdot ((SW_1 + SW_9) + (SW_8 + SW_{16})) \\ y_{absmin} &= 2 \cdot ((SW_4 + SW_{12})) + (SW_5 + SW_{13}) \end{aligned}$$

Dieser Formelsatz läßt sich nun in Gleichung 3.5 einsetzen, um den Absolutwinkelinterpolationswert  $\xi_{abs}$  zu erfassen (vgl. Rotorpositionserfassung Kapitel 3.1.3):

$$\xi_{abs} = \frac{y_{absplus} - y_{absminus}}{y_{absmax} - y_{absmin}} \quad (3.14)$$

Mit Hilfe der Gleichung 3.14 wird der Absolutverschiebungsbereich  $[0, 8x_S]$  auf den Bereich  $[-1, +1]$  abgebildet.

Für die Absolutwinkelmessung ergibt sich somit ein Meßbereich von  $180^\circ$ . Die für die Vorzeichenbestimmung bei der Relativwinkelmessung (vgl. Kapitel 3.2.3.1) notwendige Vorverdrehung der Rotoren ( $R_1$  gegenüber  $R_2$ ) und die dadurch bedingte Einschränkung des Relativwinkelmeßbereiches, kann hier zur Erweiterung des Absolutwinkelmeßbereiches verwendet werden. Durch die unterschiedliche Breite der beiden Rotorschatten  $RS_1$  und  $RS_2$ , sind diese eindeutig zu unterscheiden, wodurch sich für die Absolutwinkelmessung eine Verdopplung des Meßbereiches auf  $360^\circ$  ergibt.

### 3.2.3.3 Bestimmung der ungefähren Rotorlage

Die in den Kapiteln 3.2.3.1 und 3.2.3.2 beschriebenen Verfahren zur Bestimmung des Relativ- und Absolutwinkels setzten die Kenntnis der ungefähren Rotorlage auf ein Segment genau voraus, um die richtigen Segmente auswählen zu können. Für die Vorzeichenbestimmung bei der Relativwinkelmessung und die Verbreiterung des Absolutwinkelmeßbereiches auf  $360^\circ$  muß der im Folgenden beschriebene Algorithmus auch die beiden Rotorschatten  $RS_1$  und  $RS_2$  eindeutig unterscheiden können.

Für den Algorithmus ist es notwendig, eine Nulllage (Absolutwinkel  $\varphi_{abs} = 0$ , Relativwinkel  $\varphi_{rel} = 0$ ) zu definieren. Die Segmentwerte bei dieser Nulllage werden in einer Liste  $H(i)$  mit  $i = 1..16$  festgehalten.  $i$  steht als Index für die Segmente lt. Abbildung 3.8. Die tatsächliche Meßreihe  $SW(i)$  mit  $i = 1..16$  wird nun mit den Nulllagenwerten  $H(i)$  verglichen, wobei die Indizes der Nulllagenwerte  $H(i)$  solange zyklisch vertauscht werden, bis sich eine maximale Übereinstimmung der Nulllagenwerte  $H(i)$  mit den Meßwerten  $SW(i)$  ergibt. Mathematisch wird dies durch die Korrelationsfunktion beschrieben. Es gilt allgemein für die Korrelation  $R_{xy}(\tau)$ :

$$R_{xy}(\tau) = \frac{1}{T} \int_{-\frac{T}{2}}^{+\frac{T}{2}} x(t)y(t + \tau) dt, \quad (3.15)$$

wobei  $\tau$  die Verschiebung zwischen den Signalen  $x(t)$  und  $y(t)$  und  $T$  die Periodendauer ist. Durch eine periodische Fortsetzung von  $H(i)$  durch  $H(i) = H(i + 16)$  läßt sich die auf diskrete Werte übertragene Korrelation  $R_{SW,H}$  nach Gleichung 3.15 anschreiben als:

$$R_{SW,H}(j) = \frac{1}{n} \sum_{i=1}^n SW(i)H(i+j) \quad (3.16)$$

Dabei ist  $n$  die Anzahl der Segmente und  $j$  die diskrete Verschiebung der Indizes.

Um die Verdrehung des Rotors gegenüber seiner durch die Nulllagenwerte  $H(i)$  definierten Nulllage angeben zu können, muß jener Verschiebungswert  $j$  gesucht werden, bei dem die Korrelation  $R_{SW,H}$  sein Maximum aufweist. Dieser Verschiebungswert  $j$  wird im weiteren als Segmentoffset bezeichnet und gibt die Rotorlage in Bezug auf die Nulllage auf  $\pm 1/2$ -Segment genau an.

Da für die Auswertung durch den Algorithmus nicht der Betrag der Korrelation  $R_{SW,H}(j)$  wichtig ist, sondern nur der Segmentoffset  $j$  bei dem die Korrelation  $R_{SW,H}$  maximal wird, kann die Division mit  $1/n$  entfallen. Außerdem dürfen die Nulllagewerte  $H(i)$  auf ganzzahlige Werte skaliert werden. Für die in Abbildung 3.8 gezeigte Struktur, läßt sich laut [ZDI99] eine Liste der Nulllagenwerte  $H(i) = [-3 \quad -1 \quad +3 \quad +3 \quad +3 \quad +1 \quad -3 \quad -3 \quad -3 \quad -3 \quad +1 \quad +3 \quad +3 \quad +3 \quad +3 \quad -1 \quad -3]$  angeben, wobei die zu den Indizes  $i = 1..16$  korrespondierenden Werte von links nach rechts angeführt sind.

Die Absetzung der hier skizzierten Algorithmen in die verwendete Sensorsoftware ist in Kapitel 6 dokumentiert. Eine Umsetzung der Algorithmen in MatLab findet sich in [ZDI99].

# Kapitel 4

## Projektübersicht

Wie im Vorwort beschrieben, wurde das Gesamtprojekt "Kapazitiver Momentensensor" von drei Diplomanden durchgeführt. Es ergaben sich drei Teilprojekte, die in dazugehörigen Diplomarbeiten dokumentiert sind. Zur besseren Orientierung soll dieses Kapitel eine kurze Projektübersicht geben und zeigen, in welchem Zusammenhang die Teilgebiete stehen, die in dieser Diplomarbeit, in [WAN99] und [ZDI99] detailliert beschrieben sind.

Abbildung 4.1 zeigt neben den Sensorkomponenten, bestehend aus Senderansteuerung, Sendeelektrode, Rotoren, Empfangselektrode, Empfängerschaltung und Mikroprozessorboard, auch die einzelnen Komponenten, der für den Momentensensor entwickelten Meß- und Prüfumgebung. Diese besteht aus dem Prüfstand mit Schrittmotor und Referenzwinkelsensor, sowie die Schnittstellen zum PC, mit welcher eine Steuerung und Auswertung realisiert wurden. Zur Illustration des Projektes dient Abbildung 4.2, eine Photographie des Prüfstandes mit zu messendem Sensor und Benutzerumgebung.

Die Anpassung des bereits vorhandenen Prüfstandes, um eine definierte Relativverdrehung zwischen den Rotorflügeln einstellen zu können, ist in [WAN99] dargestellt. Dort finden sich auch die Schaltungsentwicklungen für die Senderansteuerung, den analogen Empfangsteil und das Mikroprozessorboard, wobei analoger Empfangsteil und Mikroprozessorboard auf einer Platine ausgeführt sind. Außerdem zeigt [WAN99] ein mögliches Kapazitätsmodell der verwendeten Elektrodenstruktur.

In 6 ist die Sensorsoftware (Segmentansteuerung, Meßalgorithmen, Adaptionalgorithmen) für den Mikroprozessor (Motorola HC12) dokumentiert. Für die Schnittstelle zwischen PC und Prüfstand wurde ein Dynamic Data Exchange-Server (DDE-Server) entwickelt, welcher zusätzlich eine Benutzerschnittstelle anbietet, durch die ein unproblematischer Zugriff auf den Meßplatz ermöglicht wird. Eine Anleitung zu dieser Software ist in Kapitel 7.3 beschrieben.

Ein Vergleich von verschiedenen Rotorstrukturen und die zugehörigen Meßergebnisse sind in [ZDI99] zu finden. Die Auswertungen der Messungen wurden in MatLab <sup>1</sup> realisiert (vergl. [ZDI99]). Dies hat den Vorteil, daß bei Messung der einzelnen Segmentwerte, die verschiedenen Algorithmen auch unter MatLab unabhängig vom Sensor getestet werden können. Die Arbeit mit DDE unter MatLab ist ebenfalls in [ZDI99] beschrieben.

---

<sup>1</sup>MATLAB 5.1 copyright 1984-1997 by The Math Works, Inc.



Zur Optimierung und Verbesserung der Sensorstrukturen ist die Kenntnis der elektrischen Feldverläufe, abhängig vom jeweiligen Sensordesign, notwendig. Kapitel 5 zeigt eine Feldsimulation und deren Anwendung auf die optimale Wahl verschiedener geometrischer Parameter. Eine zusätzliche Anwendung der Simulation kann in [ZDI99] nachgelesen werden. Sie zeigt unter Verwendung dieser Simulation die Auswirkung einer Rotorverkipfung.

Die Resultate dieser gemeinsamen Diplomarbeit wurden in zwei englischsprachigen Papers zusammengefaßt. Das erste Paper [BRA00] wurde für die

17<sup>th</sup> IEEE  
Instrumentation and Measurement Technology Conference  
in Baltimore (Maryland, USA)

eingereicht, und wurde am 3. Mai 2000 von Hrn. Univ.-Prof. Dipl.-Ing. Dr. Georg Brasseur vorgetragen. Das Paper wurde anstatt eines englischsprachigen Abstracts als Anhang A angeführt.

Diese Diplomarbeit wurde auch auf der

IMEKO 2000  
International Measurement Confederation  
in der Hofburg, Wien

von Stefan Cermak präsentiert [CER00]. Der Vortrag dazu fand am 25. September 2000 statt.

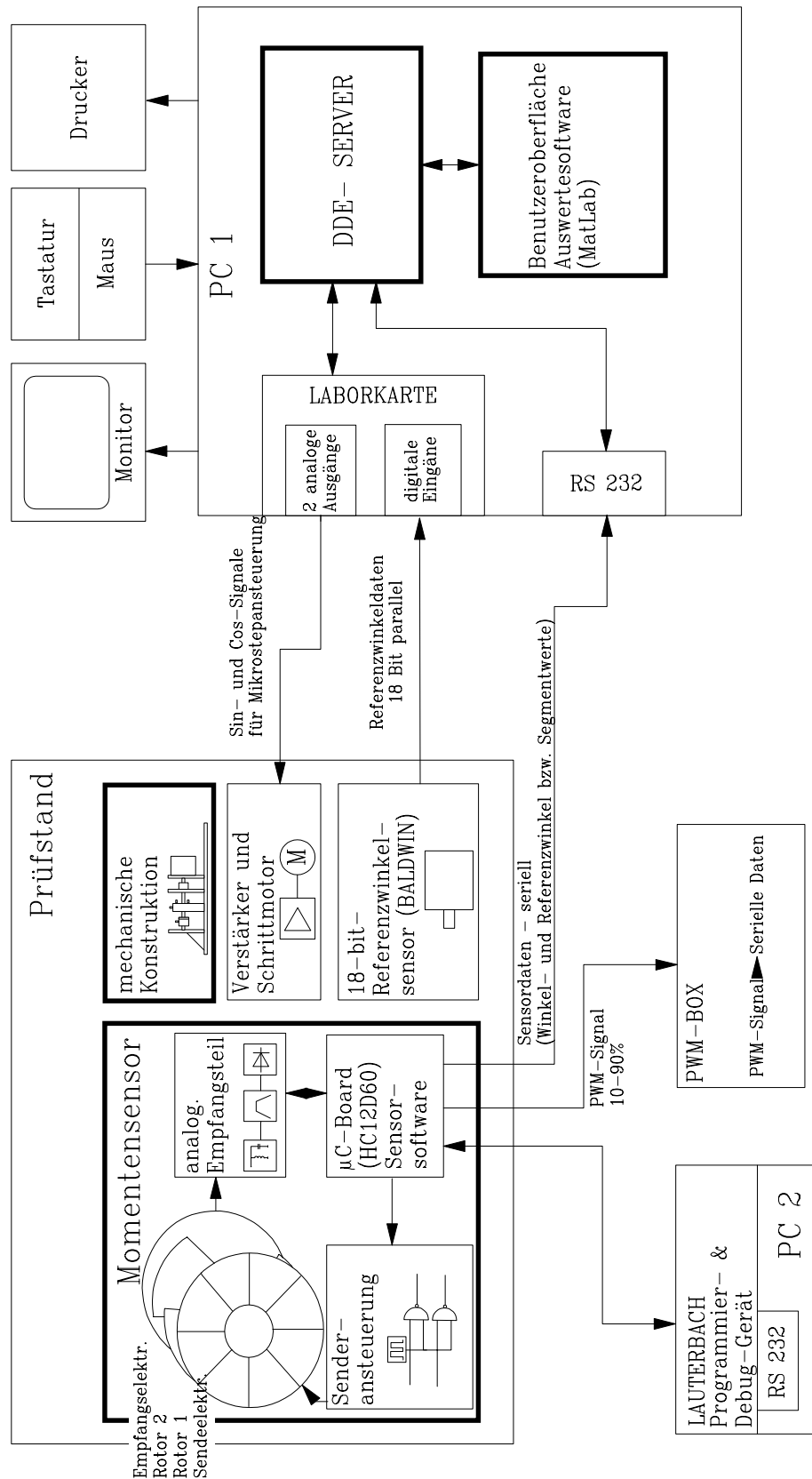


Abbildung 4.1: Blockschaltbild zur Projektübersicht - Die fettumrandeten Blöcke sind Gegenstand des Gesamtprojektes "Kapazitiver Momentensensor"

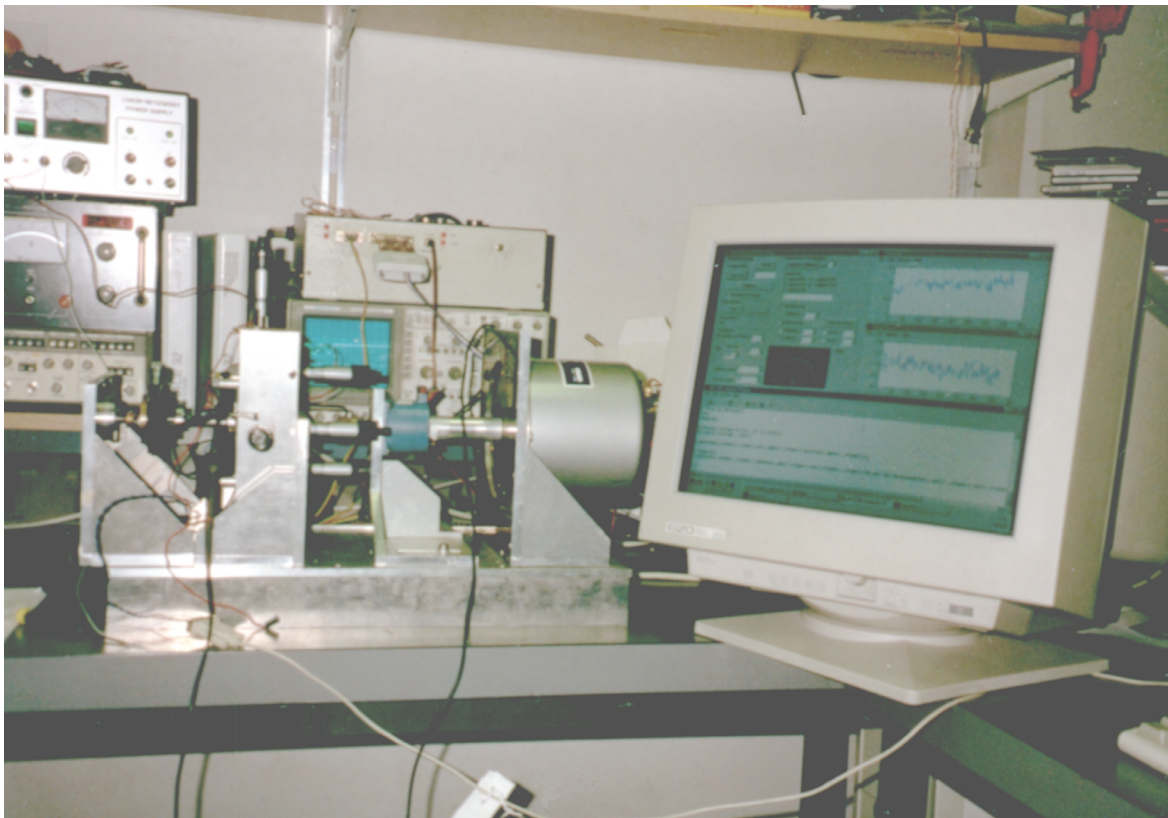


Abbildung 4.2: Photo der Meßaufbauten - links-mitte: Prüfstand mit (von links nach rechts): Momentensensor in Positioniereinheit, Schrittmotor, Referenzwinkelsensor (BALDWIN), darüber: Verstärker für Schrittmotoransteuerung, rechts: Auswertemonitor

# Kapitel 5

## Feldsimulation

In diesem Kapitel wird versucht, wesentliche Charakteristika des Sensors durch Simulation am Rechner nachzubilden. Dadurch soll es möglich werden, den Einfluß gewisser Geometrieparameter aufzuzeigen, beziehungsweise in weiterer Folge Designvorschläge für eine bessere Sensorgeometrie zu erhalten.

Der reale Sensor mißt die Kapazität zwischen Sender und Empfänger mit einer Trägerfrequenz. Die Feldsimulation hingegen, die in diesem Kapitel beschrieben werden soll, geht von elektrostatischen Näherungen aus, wodurch sich das Feldproblem auf eine Kapazitätsberechnung vereinfacht.

In Kapitel 3 wurde die grundsätzliche Struktur des Sensors vorgestellt. Bis jetzt wurden jedoch noch keine Überlegungen über die zu wählenden Geometrieparameter angestellt. So ist zum Beispiel die Sensorhöhe  $d$  (Abstand zwischen Sender und Empfänger, siehe Abb. 3.3) in ihrer minimalen Abmessung durch die Toleranz der Rotorposition in axialer Richtung gegeben:

Beim realisierten Sensorprototyp wurde von  $\pm 1$  mm Toleranz  $d_{tol}$  ausgegangen. Die beiden Rotorflügel  $R1$  und  $R2$  (Abb. 3.8) sind jeweils 1 mm dick ( $d_{r1}; d_{r2}$ ), womit sich eine Minimalhöhe  $d_{min} = d_{r1} + d_{r2} + 2d_{tol} = 4$  mm zwischen Sender und Empfänger ergibt. Um sicherzugehen, daß keiner der Rotorflügel  $R1$  und  $R2$  mit den Sendelektroden  $S_i$ , oder dem Empfänger  $E$  in Kontakt kommt, wurde ein Abstand von  $d = 6$  mm gewählt. Um den Einfluß der Sensorhöhe zu verifizieren, wurden auch Messungen mit einem vergrößerten Abstand von  $d = 12$  mm ausgeführt.

Durch den Einfluß der Streufelder auf die Segment-Empfängerkapazitäten  $C_{SiE}$  (siehe Kapazitätsverlauf in Abb. 3.4) wird einerseits der Bereich linearer Kapazitätsänderung leicht gekrümmt, andererseits ist auch der Bereich einer näherungsweise konstanten Kapazität verschmälert. Es werden somit alle vier Größen der ratiometrischen Formel 3.14 verfälscht. Der Einfluß der Streufelder bewirkt somit einen Winkelfehler (bzw. Relativwinkelfehler), dessen Einfluß durch die Feldsimulation untersucht werden soll.

Da der Streufeldverlauf (Bereich der Abflachung in Abb. 3.4) durch die Wahl der Sensorhöhe  $d$  vollständig definiert ist, kann durch eine Segmentverbreiterung der mittlere lineare Bereich vergrößert werden. Dies wird durch Vergrößerung des Innenradiuses der Segmente erreicht. Der Kapazitätsmeßbereich des Sensors muß also nach außen, zu größeren Durchmessern und Segmentbreiten, geschoben werden. Je weiter der Kapazitätsmeßbereich nach außen

geschoben wird, desto größer ist der Fehler, welcher durch die Rotorverkipfung entsteht, da der Höhenversatz der Rotorflächen mit wachsendem Radius zunimmt.

Der hier vorgestellte Gedankengang wirft nun die Frage nach dem optimalen Außendurchmesser auf.

Um diese und ähnliche Fragen beantworten zu können, ist es nützlich, den Einfluß der Segmentgeometrien auf die Streufelder, und in weiterer Folge auf die Meßgröße zu kennen. Hierzu wurde ein Simulationspaket in Matlab entwickelt, welches die Segment-Empfängerkapazität  $C_{SiE}$  durch eine Feldsimulation ermittelt.

Im Zuge dieser Diplomarbeit wurde der ursprüngliche Algorithmus (Algorithmus 1) leicht adaptiert (Algorithmus 2). Durch diese Variation sind nun größere Streufelder erlaubt. Dies war nötig, da der Sensor in seiner Anwendung als Drehmomentsensor doppelt so viele Segmente braucht, und daher mehr Streufelder entstehen. In sämtlichen folgenden Simulationsanwendungen wurden die beiden Algorithmen (1 und 2) zur Auswertung verwendet, und deren Resultate verglichen.

## 5.1 Die Struktur der Feldsimulation

Das Feldproblem läßt sich durch die drei plattenförmigen Körper – Sender, Rotor und Empfänger – komplett beschreiben. Die Struktur läßt sich somit durch eine Dreieckschaltung von drei Ersatzkapazitäten, wie in Abbildung 5.1 gezeigt, beschreiben.  $C_{SiE}$  ist die Kapazität zwischen einem Sendesegment  $Si$  und dem Empfänger  $E$ . Sie ist jene Kapazität, aus deren Kenntnis auf die Rotorposition rückgeschlossen werden kann.  $C_{SiR}$  ist die Kapazität, welche zwischen Sendersegment  $Si$  (mit  $1 \leq i \leq 16$ ) und Rotor  $R$  auftritt. Die Kapazität  $C_{RE}$  tritt zwischen Rotor  $R$  und Empfänger  $E$  auf. Die Empfangselektrode wird nur ganz schwach über die Sender-Empfängersegmentkapazität  $C_{SiE}$  angeregt. Da diese Impedanz im Vergleich zu der an der Empfangsfläche angekoppelten Schwingkreisimpedanz viel größer ist, darf die Empfangsfläche  $E$  als Massepotential betrachtet werden. Eine detaillierte Beschreibung der Empfangsschaltung ist in [WAN99] nachzulesen. Da der hier verwendete Sensor mit einem geerdeten Rotor  $R$  aufgebaut ist (siehe Kap. 3.1.3), kann die Kapazität  $C_{RE}$  zwischen Rotor  $R$  und Empfänger  $E$  im Ersatzschaltbild als kurzgeschlossen betrachtet werden.

Zur Berechnung der Kapazitäten  $C_{SiR}$  wird das Sendesegment  $Si$  gegenüber dem Rotor  $R$  und dem Empfänger  $E$  auf ein vorgegebenes Potential gelegt. Über die folgende Feldberechnung werden die Ladungen auf den drei Körpern berechnet. Die Quotienten zwischen den Ladungen und den anliegenden Spannungen entsprechen dabei den gesuchten Kapazitäten. Wobei die Ladung am Rotor zur Kapazität  $C_{SiR}$  proportional ist, und die Ladung auf dem Empfänger  $E$  zur Kapazität  $C_{SiE}$  proportional ist. Die Ladung auf dem Sender wird durch die Summe der beiden Kapazitäten erzeugt.

Um nicht das Feld im allgemeinem 3-dimensionalen Raum lösen zu müssen, wurde auf die schon im Kapitel 3.1.3 angewandte 2-dimensionale Abwicklungsdarstellung (siehe Bild 3.8) übergegangen. Der Empfänger  $E$  ist in diesem Fall eine, ins unendliche ausgedehnte, ebene Platte. Sie liegt auf Massepotential. Daher läßt sich ihr Einfluß auf die Feldverteilung durch die Spiegelungsmethode berücksichtigen (siehe [PRE94]).

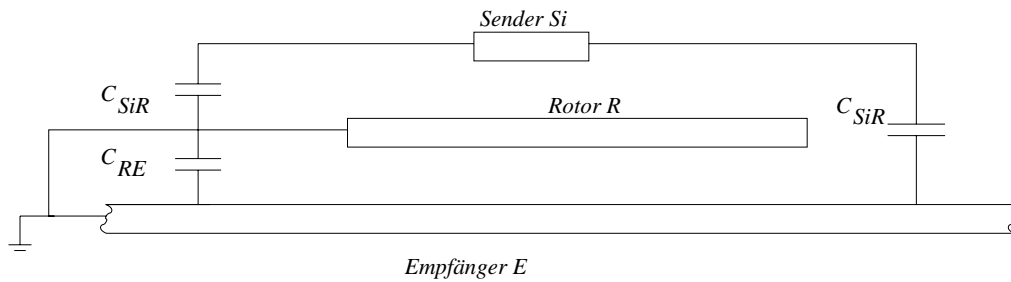


Abbildung 5.1: Symbolische Plattenanordnung, mit eingezeichneten Ersatzkapazitäten. Da Rotor und Empfänger gemeinsam auf Massepotential liegen, verschwindet der Einfluß von  $C_{RE}$ .

Da Rotor  $R$  und Empfänger  $E$  unterschiedliche Größen haben, und sie sich nicht notwendigerweise übereinander befinden müssen, darf nicht das vereinfachte Modell eines homogen geladenen Plattenkondensators verwendet werden.

Dies wirkt sich vor allem auf den Rotor  $R$  aus, bei dem sich die Ladungen zum größten Teil unter dem Sendesegment  $Si$ , beziehungsweise auf der zum Sender näher gelegenen Seite des Rotors befinden. Die Abbildung 5.2 zeigt eine typische Ladungsverteilung.

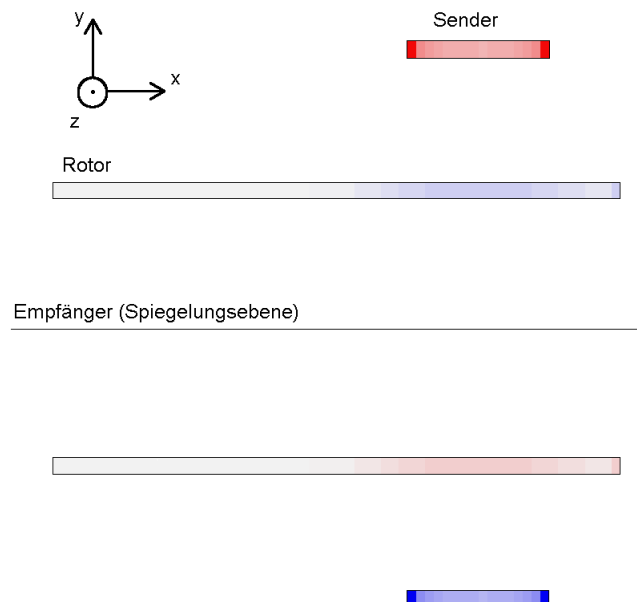


Abbildung 5.2: Graphische Darstellung einer simulierten Ladungsverteilung. Die Polarität der Ladungen ist durch die Farben Rot und Blau dargestellt. Die Intensität der Farben ist ein Maß für die Ladungsdichte. Am Sender ist die Verdrängung der Ladungen (Ladungen gleicher Polarität stoßen sich ab) an die Plattenränder zu erkennen. Beim Rotor hingegen bewirken die Ladungen am Sender, daß sich die Ladungen zum Großteil in der Mitte der Platte, wo der Abstand zum Sender minimal wird, konzentrieren.

Um mit dieser ungleichmäßigen Ladungsverteilung rechnen zu können, wurden der Rotor

und der Sender in viele "kleine Streifen" aufgeteilt. Für diese "kleinen Streifen" kann dann näherungsweise die Ladung als homogen verteilt angesetzt werden. Da die Anzahl der Streifen maßgebend für die Qualität der Lösung ist, aber auch mit der Qualität der Lösung die Rechenzeit steigt, ist die Lösung ein Kompromiß zwischen Lösungsqualität und Rechenzeit.

Das Feld für einen homogen geladenen Streifen, sowie dessen Spannung, läßt sich wie in Kapitel 5.2 gezeigt wird, berechnen. Durch Superposition der Einzelfelder der einzelnen Streifen läßt sich das gesamte Feldbild näherungsweise abbilden und die gesuchten Kapazitäten berechnen.

## 5.2 Das Feld einer Streifenladung

Ausgegangen wird hierbei von dem Feld einer Linienladung, wie es in [PRE94] vorgestellt wird. Ist  $\tau$  die Linienladungsdichte,  $\varrho$  die Entfernung zwischen der Linienladung und dem Betrachtungspunkt und  $\vec{e}_\varrho$  ein Einheitsvektor, welcher von der Linienladung zum Betrachtungspunkt zeigt, dann berechnet sich die Feldstärke  $\vec{E}_{lin}$  nach

$$\vec{E}_{lin} = \frac{\tau}{2\pi\epsilon_0} \cdot \frac{\vec{e}_\varrho}{\varrho} \quad (5.1)$$

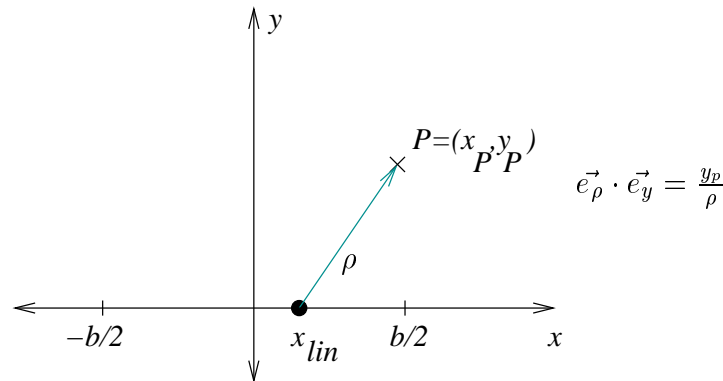


Abbildung 5.3: Die Abbildung zeigt eine Linienladung im Punkt  $(x_{lin}, 0)$  mit eingezeichneten Radiusvektor zum Punkt  $(x_P, y_P)$ .

Im nächsten Schritt wechselt man von der linienhaften Betrachtungsweise auf eine flächenhafte. Die Feldstärke im Punkt  $P$  erhält man dann aus der gedanklichen Superposition unendlich vieler aneinandergereihter differentiell kleiner Linienladungsbeiträge. Die Linienladungsdichte  $\tau$  entspricht dann  $\sigma dx$ , und das Teilfeld einer Linienladung  $\vec{E}_{lin}$  geht über in den differentiell kleinen Beitrag  $d\vec{E}$ . Durch Aufintegration dieser differentiell kleinen Feldstärkeanteile  $d\vec{E}$ , erhält man das elektrische Feld einer Streifenladung. Die Breite  $b$  der Streifenladung reicht von  $x = -b/2$  bis  $x = b/2$ . Für die Anwendung in der Simulation ist es nicht notwendig den gesamten Feldverlauf zu kennen, es reicht den Anteil der Feldstärke in  $y$ -Richtung  $E_y$  an einem beliebigen Punkt  $\vec{P}$  zu kennen. Die Spannung zwischen zwei Punkten gleicher  $x$ -Koordinate ergibt sich als Integral der Feldstärke  $\vec{E}_{lin}$  entlang der

y-Richtung. In Abb. 5.3 befindet sich eine Darstellung der angenommenen geometrischen Verhältnisse. Nun läßt sich die Integralgleichung für das Feld im Punkt  $\vec{P} = x_P \cdot \vec{e}_x + y_P \cdot \vec{e}_y$  in y-Richtung zu

$$E_y(x_P, y_P) = \int_{-b/2}^{+b/2} \frac{\sigma}{2\pi\epsilon_0} \cdot \frac{\vec{e}_\varrho \cdot \vec{e}_y}{\sqrt{y_P^2 + (x_P - x_{lin})^2}} dx_{lin} \quad (5.2)$$

angeben. Das innere Produkt  $\vec{e}_\varrho \cdot \vec{e}_y$  kann durch

$$\vec{e}_\varrho \cdot \vec{e}_y = \frac{y_P}{\sqrt{y_P^2 + (x_P - x_{lin})^2}} \quad (5.3)$$

ausgedrückt werden. Damit ergibt sich das zu lösende Integral

$$E_y(x_P, y_P) = \frac{\sigma y_P}{2\pi\epsilon_0} \int_{-b/2}^{+b/2} \frac{1}{y_P^2 + (x_P - x_{lin})^2} dx_{lin} \quad (5.4)$$

Durch Auflösen des Integrales (nach [BAR90]) ergibt sich die elektrische Feldstärke  $E_y$  in Richtung der y-Achse zu

$$E_y(x_P, y_P) = \frac{\sigma}{2\pi\epsilon_0} \left( \arctan \frac{x_P - b/2}{y_P} - \arctan \frac{x_P + b/2}{y_P} \right) \quad (5.5)$$

Mit Gleichung 5.5 und 5.6 kann nun die Spannung zwischen zwei übereinander liegenden Punkten berechnet werden. Dazu wird ein Integrationsweg parallel zur y-Achse mit dem Startpunkt  $h_1$  und dem Endpunkt  $h_2$  gewählt. Die Abbildung 5.4 zeigt die zur Formel 5.7 gehörende Geometrie.

$$U(h_1, h_2) = \int_{h_1}^{h_2} E_y dy_P \quad (5.6)$$

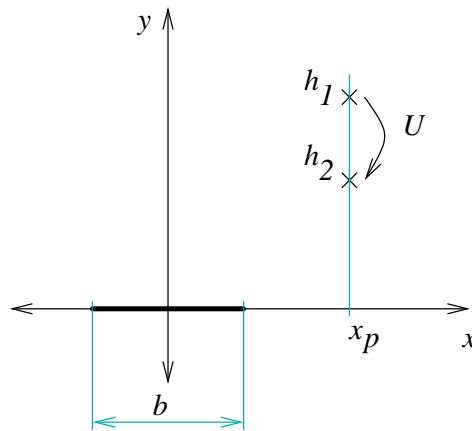


Abbildung 5.4: Die Abbildung zeigt die zur Formel 5.7 gehörende geometrische Anordnung.

Die Lösung des Integrales 5.6 (ermittelt mit Hilfe des Programmes Mathematica 3.0) lautet



$$\begin{aligned}
U(h_1, h_2) = \frac{\sigma}{8\pi\epsilon_0} & \left[ 4 \cdot h_1 \cdot \arctan\left(\frac{b-2x_P}{2h_1}\right) + 4 \cdot h_1 \cdot \arctan\left(\frac{b+2x_P}{2h_1}\right) - \right. \\
& 4 \cdot h_2 \cdot \arctan\left(\frac{b-2x_P}{2h_2}\right) - 4 \cdot h_2 \cdot \arctan\left(\frac{b+2x_P}{2h_2}\right) + \\
& b \cdot \log\left(b^2 - 4bx_P + 4(h_1^2 + x_P^2)\right) - 2 \cdot x_P \cdot \log\left(b^2 - 4bx_P + 4(h_1^2 + x_P^2)\right) + \\
& b \cdot \log\left(b^2 + 4bx_P + 4(h_1^2 + x_P^2)\right) + 2 \cdot x_P \cdot \log\left(b^2 + 4bx_P + 4(h_1^2 + x_P^2)\right) - \\
& b \cdot \log\left(b^2 - 4bx_P + 4(h_2^2 + x_P^2)\right) + 2 \cdot x_P \cdot \log\left(b^2 - 4bx_P + 4(h_2^2 + x_P^2)\right) - \\
& \left. b \cdot \log\left(b^2 + 4bx_P + 4(h_2^2 + x_P^2)\right) - 2 \cdot x_P \cdot \log\left(b^2 + 4bx_P + 4(h_2^2 + x_P^2)\right) \right] \quad (5.7)
\end{aligned}$$

Die Formel 5.7 drückt die Spannung zwischen den zwei übereinander liegenden Punkten  $(x_P, h_1)$  und  $(x_P, h_2)$  aus. Die Punkte dürfen im gesamten Feldraum liegen, wobei der Fall  $h_1 = 0$  und  $h_2 = 0$  noch separat abgehandelt werden muß.

In diesem Fall kommt es zu einer Division durch Null. Der kritische Term ist von der Form:

$$\lim_{x \rightarrow 0} \left[ x \cdot \arctan\left(\frac{a}{x}\right) \right] \quad (5.8)$$

Da jedoch

$$\lim_{x \rightarrow 0} \left[ \arctan\left(\frac{a}{x}\right) \right] = \pm \frac{\pi}{2} \quad (5.9)$$

gilt, folgt, daß die Lösung zu Gleichung 5.8 zu Null wird.

Die Gleichung 5.7 beschreibt eine Flächenladung auf der x-Achse im Intervall zwischen  $-b/2$  und  $+b/2$ . In der Simulation wird diese Formel wiederholt angewandt, wobei die Ladung immer in  $x$ - und in  $y$ - Richtung verschoben wird. Die Gleichung 5.7 und diese Koordinatenverschiebung sind in einem M-File wie folgt abgesetzt:

```

function [U]=Spannungsfeldeinerplatte(breit,hoch,tief,mitte,Q,x,y);
%
% function [U]=Spannungsfeldeinerplatte(breit,hoch,tief,mitte,Q,x,y);
%
% Die Funktion berechnet die Spannung, die sich durch eine
% homogene Flächenladung "Q", der Breite "breit" und der
% Tiefe "tief" ergibt. Die Ladung ist parallel zur x-Achse
% und befindet sich auf der Höhe (y-Wert) "hoch". Die Mitte
% der Ladung (x-Koordinate) befindet sich auf "mitte". Die
% Punkte zwischen welchen die Spannung angegeben wird
% sind (x,0) und (x,y).

sigma=Q/(tief*breit); %berechne die Flächenladungsdichte
h1=-hoch;             %Koordinatentransformation
h2=y-hoch; x=x-mitte;
```

```

b=breit; %die Breite wird in der Formel b
          %genannt

% Achtung! Im Falle von h1 oder h2 gleich 0 kommt es zu
% einer Division durch null! Dieser Fall muß separat
% behandelt werden

if (h1==0)
    U=0;
else
    U=U+4*h1*atan((b-2*x)/(2*h1))+4*h1*atan((b+2*x)/(2*h1));
end; if (h2~=0)
    U=U-4*h2*atan((b-2*x)/(2*h2))-4*h2*atan((b+2*x)/(2*h2));
end;

U=U+ b*log(b^2-4*b*x+4*(h1^2+x^2));
U=U-2*x*log(b^2-4*b*x+4*(h1^2+x^2));
U=U+ b*log(b^2+4*b*x+4*(h1^2+x^2));
U=U+2*x*log(b^2+4*b*x+4*(h1^2+x^2));
U=U- b*log(b^2-4*b*x+4*(h2^2+x^2));
U=U+2*x*log(b^2-4*b*x+4*(h2^2+x^2));
U=U- b*log(b^2+4*b*x+4*(h2^2+x^2));
U=U-2*x*log(b^2+4*b*x+4*(h2^2+x^2));

k=sigma/(8*pi*8.854187818e-12);
U=k*U;

```

### 5.3 Beschreibung des Feldproblems

Wie im Kapitel 5.1 erwähnt, hängt die Empfangselektrode auf Massepotential, wodurch die Spiegelungsmethode angewandt werden kann. Aus der Spiegelungsmethode folgt eine Plattenanordnung nach Bild 5.5. Im Fall des unabgedeckten Rotors ist die Kapazität ein Maximum. Der Wert hängt allerdings, zufolge der Streufelder, von der Breite des Fensters zwischen 2 Rotorflügel ab. Daher ist es sinnvoll zwei Rotorflügel zu simulieren. Für den exakten Feldverlauf wäre eine periodische Fortsetzung der Rotorflügel notwendig, da aber die restlichen Rotorflügel durch ihre große Entfernung zum Sender einen verschwindenden Einfluß haben, müssen sie nicht in der Simulation berücksichtigt werden.

Gedanklich werden der Sender  $S$  und Rotor  $R$  in kleine Streifen zerschnitten, für welche näherungsweise eine konstante Ladungsverteilung (siehe Bild 5.2) angenommen wird. Um akzeptable Simulationsergebnisse zu bekommen, wurde der Sender in  $T_{Sender} = 16$  Teile, und der Rotor in  $T_{Rotor} = 64$  Teile zerlegt. Um die Feldstärke bzw. Spannung in einem beliebigen Raumpunkt zu berechnen, müssen die  $T_{Sender} + 2 \cdot T_{Rotor} = 144$  Teilladungen bekannt sein, und die Gleichung 5.5  $2 \cdot (T_{Sender} + 2 \cdot T_{Rotor}) = 288$  mal gelöst werden, wobei jede Teilplatte gespiegelt mit negativer Ladung auftritt. Für die Berechnung der Kapazität

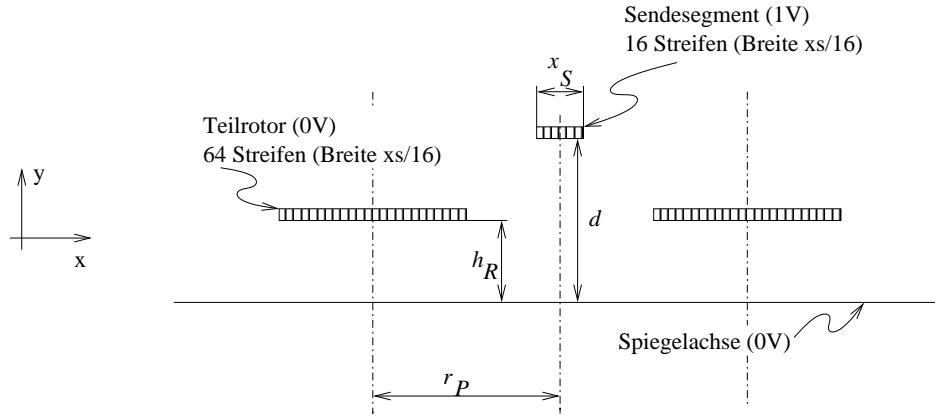


Abbildung 5.5: Segmentierte Plattenanordnung für das zu lösende Feldproblem bei unabgedecktem Rotor (maximaler Kapazität).

ist das Verhältnis zwischen Ladung und Spannung ausschlaggebend. Die Empfangselektrode liegt auf Masse, und bietet sich daher als Referenzebene für die Spannungsberechnung an. Die Spannung eines jeden Raumpunktes kann daher aus der Potentialdifferenz zwischen dem Punkt und einem direkt darunter(darüber) liegenden Punkt auf der Empfangselektrode berechnet werden. Die in Kapitel 5.2 hergeleitete Spannungsgleichung 5.7 ist daher auch mit ihrer Einschränkung auf vertikal übereinander liegende Start und Endpunkte anwendbar.

### 5.3.1 Aufstellen des beschreibenden Gleichungssystem

Die Simulation besteht aus  $T_{max} = 2 \cdot (T_{Sender} + 2 \cdot T_{Rotor})$  (z. B. 288) Einzelteilen. Durch Anwendung der Gleichung 5.7 kann die Spannung  $U_{m,l}$  auf der  $m$ -ten Platte berechnet werden, welche durch die Ladung auf dem  $l$ -ten Streifen hervorgerufen wird. Hierbei wird  $\sigma$  durch den Ausdruck  $\sigma = Q/A$  ersetzt, wobei  $A$  die Streifenfläche ist. Durch Einsetzen der Geometriedaten läßt sich die Gleichung dann auf die Form  $U_m = Q_l \cdot k_{m,l}$  bringen, wobei  $k_{m,l}$  sämtliche Geometriedaten zusammenfaßt. Kennt man nun alle  $T_{max}$  Ladungen, so läßt sich die Spannung auf der  $m$ -ten Platte  $U_m$  zufolge des Superpositionsprinzips berechnen.

$$U_m = \sum_{l=1}^{T_{max}} Q_l \cdot k_{m,l} \quad (5.10)$$

Faßt man die Spannungen der einzelnen Streifen in einem Spannungsspaltenvektor  $\mathbf{U}$ , die einzelnen Ladungen in einem Ladungszeilenvektor  $\mathbf{Q}$  und die Geometriedaten zu einer Geometriematrix  $\mathbf{K}$  zusammen, so läßt sich das Gleichungssystem durch

$$\mathbf{U} = \mathbf{K} \cdot \mathbf{Q} \quad (5.11)$$

mit

$$\mathbf{U} = \begin{pmatrix} U_1 \\ U_2 \\ \vdots \\ U_{T_{max}} \end{pmatrix}, \quad \mathbf{Q} = \begin{pmatrix} Q_1 & Q_2 & \dots & Q_{T_{max}} \end{pmatrix},$$

$$\mathbf{K} = \begin{pmatrix} K_{1,1} & K_{1,2} & \dots & K_{1,T_{max}} \\ K_{2,1} & K_{2,2} & \dots & K_{2,T_{max}} \\ \vdots & \vdots & \ddots & \vdots \\ K_{T_{max},1} & K_{T_{max},2} & \dots & K_{T_{max},T_{max}} \end{pmatrix}$$

angeben. Es bietet sich nun noch die Substitution von  $\mathbf{K}$  durch  $\mathbf{C}^{-1}$  an.  $\mathbf{C}$  hat dann den Stellenwert einer Kapazitätsmatrix, und Gleichung 5.10 läßt sich durch  $\mathbf{Q} = \mathbf{C} \cdot \mathbf{U}$  angeben. Dieser Ausdruck entspricht dem in [PRE94] Kapitel 10.3 vorgestellten Schema. Die  $\mathbf{K}$ -Matrix läßt sich durch  $T_{max}^2$ -faches lösen der Gleichung 5.7 bestimmen. Notwendigerweise gibt es zu jedem Ladungsvektor  $\mathbf{Q}$  genau einen passenden Spannungsvektor  $\mathbf{U}$ , und umgekehrt. Der Rang der Matrix  $\mathbf{K}$  ist also immer  $T_{max}$ . Sie ist immer invertierbar. Zur Kapazitätsbestimmung wird nun angenommen, daß der Sender auf einem Potential von 1 V liegt und der Rotor auf einem Potential von 0 V. Die Kapazität zwischen Rotor  $R$  und Sender  $S$  errechnet sich dann aus der Summe der negativen Ladungen am Rotor. Die Summe der Ladungen am Sender entsprechen der Summe der Kapazitäten zwischen Sender  $S$  und Empfänger  $E$  sowie Sender  $S$  und Rotor  $R$ . Da die Ladungen am Rotor, im Gegensatz zu denen am Sender, ein negatives Vorzeichen haben, ergibt die Aufsummation des Ladungsvektors die gesuchte Kapazität  $C_{SiE}$

$$\begin{aligned} U_m &= \begin{cases} 1V & \text{für } i \leq T_{Sender} \\ 0V & \text{für } i > T_{Sender} \end{cases} \\ \mathbf{Q} &= \mathbf{C} \cdot \mathbf{U} \\ C_{SiR} &= - \sum_{j=T_{Sender}+1}^{T_{Sender}+2 \cdot T_{Rotor}} Q_j \\ C_{SiE} &= \sum_{j=1}^{T_{Sender}+2 \cdot T_{Rotor}} Q_j \end{aligned} \quad (5.12)$$

Mit den oben angeführten Überlegungen, und der bereits vorgestellten M-Funktion: "Spannungsfeldeinerplatte", läßt sich nun eine M-Funktion erstellen, welche in Abhängigkeit von Rotorposition und Sensorhöhe die erwartete Kapazität berechnet:

```
function [Cse]=BerechneCse(rp,rotorhoch,hoch);
%
% function [Cse]=BerechneCse(rp,rotorhoch,hoch);
%
% Die Funktion berechnet die erwartete Kapazität zwischen
% Sender und Empfänger, bei gegebener Sensorhöhe,Rotorhöhe
% und Rotorposition. Die Angaben werden in mm erwartet.
%
% Die restlichen Sensorparameter sind als konstante im
% Programm fix vergeben.
%
% Ihre Werte sind:
%
```

```

% Senderbreit=6mm / die Breite des Senders
% tief=10mm / die Tiefe der Platten
% Rotorbreit=4*Senderbreit / der Rotor überdeckt 4
%           Sendesegmente
%
% Der Rotor wird in 64 Teile und der Sender in 16 Teile
% zerschnitten.

Senderbreit=6e-3;
tief=10e-3;
Rotorbreit=4*Senderbreit;
Rotorteile=64;
Senderteile=16;
Q=1;

% Berechne die K-Matrix

for Senderteil=1:Senderteile
    Sendermitte=-Senderbreit/2-Senderbreit/Senderteile/2
                +Senderbreit/Senderteile*Senderteil;

    for Senderteil2=1:Senderteile
        Sendermitte2=-Senderbreit/2-Senderbreit/Senderteile/2
                    +Senderbreit/Senderteile*Senderteil2;
        [K1]=Spannungseinflusseinerplatte(
            Senderbreit/Senderteile, hoch,tief,
            Sendermitte2, Q,Sendermitte,hoch);
        [K2]=Spannungseinflusseinerplatte(
            Senderbreit/Senderteile,-hoch,tief,
            Sendermitte2,-Q,Sendermitte,hoch);
        K(Senderteil,Senderteil2)=K1+K2;
    end;

    for teilrotor=1:Rotorteile
        Rotx=rp-Rotorbreit/2-Rotorbreit/Rotorteile/2
            +teilrotor*Rotorbreit/Rotorteile;
        [K1]=Spannungseinflusseinerplatte(
            Rotorbreit/Rotorteile, rotorhoch,tief,
            Rotx, Q,Sendermitte,hoch);
        [K2]=Spannungseinflusseinerplatte(
            Rotorbreit/Rotorteile,-rotorhoch,tief,
            Rotx,-Q,Sendermitte,hoch);
        K(Senderteil,teilrotor+Senderteile)=K1+K2;
    end;

    for teilrotor=1:Rotorteile

```

```

    Rotx=rp-Rotorbreit/2-Rotorbreit/Rotorteile/2
        +teilrotor*Rotorbreit/Rotorteile+2*Rotorbreit;
    [K1]=Spannungseinflusseinerplatte(
        Rotorbreit/Rotorteile, rotorhoch,tief,
        Rotx,Q,Sendermitte,hoch);

    [K2]=Spannungseinflusseinerplatte(
        Rotorbreit/Rotorteile,-rotorhoch,tief,
        Rotx,-Q,Sendermitte,hoch);

    K(Senderteil,teilrotor+Senderteile+Rotorteile)=K1+K2;
end;
end;

for teilrotor=1:Rotorteile
    Rotx=rp-Rotorbreit/2-Rotorbreit/Rotorteile/2
        +teilrotor*Rotorbreit/Rotorteile;

    for Senderteil=1:Senderteile
        Sendermitte=-Senderbreit/2-Senderbreit/Senderteile/2
            +Senderbreit/Senderteile*Senderteil;
        [K1]=Spannungseinflusseinerplatte(
            Senderbreit/Senderteile, hoch,tief,
            Sendermitte, Q,Rotx,rotorhoch);
        [K2]=Spannungseinflusseinerplatte(
            Senderbreit/Senderteile,-hoch,tief,
            Sendermitte,-Q,Rotx,rotorhoch);
        K(Senderteile+teilrotor,Senderteil)=K1+K2;
    end;

    for teilrotor2=1:Rotorteile
        Rotx2=rp-Rotorbreit/2-Rotorbreit/Rotorteile/2
            +teilrotor2*Rotorbreit/Rotorteile;
        [K1]=Spannungseinflusseinerplatte(
            Rotorbreit/Rotorteile, rotorhoch,tief,
            Rotx2, Q,Rotx,rotorhoch);
        [K2]=Spannungseinflusseinerplatte(
            Rotorbreit/Rotorteile,-rotorhoch,tief,
            Rotx2,-Q,Rotx,rotorhoch);
        K(Senderteile+teilrotor,Senderteile+teilrotor2)=K1+K2;
    end;

    for teilrotor2=1:Rotorteile
        Rotx2=rp-Rotorbreit/2-Rotorbreit/Rotorteile/2
            +teilrotor2*Rotorbreit/Rotorteile+2*Rotorbreit;
        [K1]=Spannungseinflusseinerplatte(

```

```

        Rotorbreit/Rotorteile, rotorhoch,tief,
        Rotx2,Q,Rotx,rotorhoch);

    [K2]=Spannungseinflusseinerplatte(
        Rotorbreit/Rotorteile,-rotorhoch,tief,
        Rotx2,-Q,Rotx,rotorhoch);

    K(Senderteile+teilrotor,
        Senderteile+teilrotor2+Rotorteile)=K1+K2;
end;
end;

for teilrotor=1:Rotorteile
    Rotx=rp-Rotorbreit/2-Rotorbreit/Rotorteile/2
        +teilrotor*Rotorbreit/Rotorteile+2*Rotorbreit;

    for Senderteil=1:Senderteile
        Sendermitte=-Senderbreit/2-Senderbreit/Senderteile/2
            +Senderbreit/Senderteile*Senderteil;
        [K1]=Spannungseinflusseinerplatte(
            Senderbreit/Senderteile, hoch,tief,
            Sendermitte,Q,Rotx,rotorhoch);

        [K2]=Spannungseinflusseinerplatte(
            Senderbreit/Senderteile,-hoch,tief,
            Sendermitte,-Q,Rotx,rotorhoch);

        K(Senderteile+teilrotor+Rotorteile,Senderteil)=K1+K2;
    end;

    for teilrotor2=1:Rotorteile
        Rotx2=rp-Rotorbreit/2-Rotorbreit/Rotorteile/2
            +teilrotor2*Rotorbreit/Rotorteile;
        [K1]=Spannungseinflusseinerplatte(
            Rotorbreit/Rotorteile, rotorhoch,tief,
            Rotx2,Q,Rotx,rotorhoch);
        [K2]=Spannungseinflusseinerplatte(
            Rotorbreit/Rotorteile,-rotorhoch,tief,
            Rotx2,-Q,Rotx,rotorhoch);
        K(Senderteile+teilrotor+Rotorteile,
            Senderteile+teilrotor2)=K1+K2;
    end;

    for teilrotor2=1:Rotorteile
        Rotx2=rp-Rotorbreit/2-Rotorbreit/Rotorteile/2
            +teilrotor2*Rotorbreit/Rotorteile+2*Rotorbreit;

```

```

[K1]=Spannungseinflusseinerplatte(
    Rotorbreit/Rotorteile, rotorhoch,tief,
    Rotx2, Q,Rotx,rotorhoch);
[K2]=Spannungseinflusseinerplatte(
    Rotorbreit/Rotorteile,-rotorhoch,tief,
    Rotx2,-Q,Rotx,rotorhoch);
K(Senderteile+teilrotor+Rotorteile,
    Senderteile+teilrotor2+Rotorteile)=K1+K2;
end;
end;

% setze den Spannungsvektor
U=( [ones(1,Senderteile) zeros(1,Rotorteile*2)] )';

% Berechne die Kapazit"atsmatrix
C=K^-1;

% Berechne die Ladungsverteilung
Q=C*U;

%Berechne die Sender-Empf"angerkapazit"at
Cse=sum(Q);

```

Diese Funktion kann durch die Formel eines Plattenkondensators getestet werden, man hat allerdings Sorge zu tragen, daß der Plattenabstand, bezogen auf die Plattenausmaße, klein ist und so der Einfluß des Streufeldes vernachlässigbar ist. Da die Plattendimensionen mit 8 mm mal 10 mm durch die Simulation fix vorgegeben sind, bietet sich eine Dicke von 0.1 mm an. Damit der Rotor nicht in die Rechnung mit eingeht, wird er auf 1 m Höhe gesetzt. Der Aufruf

```
Csim=berechneCse(0,1,1e-4)
```

liefert als Ergebnis  $C_{sim} = 5.5891 \text{ pF}$ . Einsetzen in die Formel

$$C_{plat} = \epsilon_0 \cdot \frac{A}{l} = \epsilon_0 \cdot \frac{6\text{mm} \cdot 10\text{mm}}{0.1\text{mm}} \quad (5.13)$$

ergibt mit 5.3125 pF ein leicht geringeres Ergebnis als bei der Simulation. Die geringere Kapazität läßt sich durch die vernachlässigten Streufelder begründen. Der Test der Simulationsfunktion war somit erfolgreich. Nun können erste Aussagen über die zu erwartenden Kapazitäten gemacht werden. So beträgt die tatsächliche Höhe des Sensors 6 mm. Die Kapazität zwischen Sender und Empfänger errechnet sich mit

```
Csim=berechneCse(0,1,6e-3)
```



zu 260 fF, einem deutlich höheren Wert als die  $C_{plat} = 88 \text{ fF}$ , welche sich aus der Plattenkondensatorformel ergeben. Dies zeigt, daß bei derart großen Abständen die Kapazität zum Großteil durch die Streukapazitäten gebildet wird. Hierbei muß angemerkt werden, daß die Feldsimulation nur die Streufelder an 2 der 4 Kanten berücksichtigt. Die tatsächliche Kapazität ist also noch um einiges größer. Eine gute Abschätzung der tatsächlichen Kapazität kann dadurch erreicht werden, in dem der Einfluß der Streufelder über die Umfanglänge hochgerechnet wird.

$$C_{tat} = C_{plat} + (C_{sim} - C_{plat}) \cdot \frac{(Senderbreite + Sendertiefe)}{Sendertiefe} \quad (5.14)$$

Die erwartete Kapazität liegt in diesem Fall also bei 363 fF. Betrachtet man den Einfluß des Rotors, und bringt den Rotor derart ins Feld, daß sich der Sender gerade zwischen den Flügeln befindet,

```
Csim=berechncse(-24e-3,3e-3,6e-3)
```

so sinkt die Kapazität  $C_{sim}$  auf 193.94 fF ab. Die Kapazität des abgedeckten Senders

```
Csim=berechncse(0,3e-3,6e-3)
```

beträgt mit 43.8 fF nur 19 % der maximalen Kapazität. Die Streufelder verhindern also eine komplette Abschirmung, und die Annahme, ein abgeschirmtes Segment habe keine Kapazität zum Empfänger, ist widerlegt. Diese Erkenntnis hat aber keinerlei Einfluß auf die Meßauswertung, da durch die Anwendung des ratiometrischen Algorithmus der additive Fehler des Streufeldes eliminiert wird.

## 5.4 Der Segmentverlauf

Nachdem nun die Funktion `berechncse` zur Kapazitätsberechnung zur Verfügung steht, kann der Verlauf der Kapazität über die Rotorposition berechnet werden.

Dazu wird der Rotor so verschoben, daß ein anfängliches Rotorloch (Bereich zwischen zwei Rotorflügel) durch einen Rotorflügel ersetzt wird. In der Simulation wird diese Verschiebung in 80 Punkte zerlegt.

Durch gespiegeltes Erweitern kann nun der Kapazitätsverlauf einer kompletten Umdrehung simuliert werden. Durch mehrfaches Verschieben kann ein kompletter Segmentverlaufsdatsatz erstellt werden und über den Algorithmus aus [ZDI99] ausgewertet werden. Auf diese Weise kann der aus dem Streufeldverlauf zu erwartende Fehler simuliert werden.

Das folgende Skript führt diese Verschiebung durch:

```

% Dieses Skript berechnet den Kapazitätsverlauf,
% welcher sich durch die Rotorverschiebung ergibt,
% und berechnet den daraus resultierenden Winkelfehler

start=cputime;

% Bilde den Kapazitätsverlauf

for i=-80:0;
    now=cputime;
    [cret]=BerechneCse(i*0.3e-3,6e-3/2,6e-3);
    disp([ 'BerechneCse Rechenzeit '
           num2str(cputime-now) 's Restzeit '
           num2str((cputime-start)/(i+81)*(-i)/60) 'Min'])
    Cse(i+81)=cret;
end;

x=(-80:80)/10;

figure;
plot(x,[Cse Cse(80:-1:1)]);
title('Segmentkapazitätsverlauf');

figure;
seg=[Cse Cse(79:-1:1)];
seg=[seg seg];
seg=[seg seg];
seg=[seg seg];
seg=[seg seg];

% Konstruiere den Segmentverlauf der 8 Segmente

for i=1:8
    data.seg(i,1:320)=seg((1:320)+(i-1)*20);
end data.RefPhi=((0:319)+0.5)/320*360';
data.RefPhi=data.RefPhi+78.75;
data.RefPhi=mod(data.RefPhi,180);

% Auswerten der Segmentdatensätze

data=EvalOffset(data);
data=EvalPhi(data,'42.0');
data.Phiold=mod(data.Phiold,180);
data=EvalPhi(data,'42.1');
data.PhiNew=mod(data.PhiNew,180);

```

```
subplot(2,1,1);
plot(1:320,mod(data.Phiold-data.RefPhi'+90,180)-90);
subplot(2,1,2);
plot(1:320,mod(data.PhiNEW-data.RefPhi'+90,180)-90);
```

Abbildung 5.6 basiert auf dem eben vorgestellten Skript. Es wurden 3 verschiedene Rotorhöhen berechnet und deren Segmentverlauf verglichen. Die Sensorhöhe betrug 6mm. Die Rotorhöhe gibt an, um wie weit der Rotor vom Empfänger entfernt war. Es zeigt sich, daß der Bereich maximaler Kapazität kaum von der Rotorhöhe abhängt, während der Bereich minimaler Kapazität stark von der Rotorhöhe abhängt. Wandert der Rotor in Richtung Sender, so wird der Segmentverlauf scharfkantiger, der Bereich minimaler Kapazität verbreitert sich. Dies deutet auf einen geringeren Winkelfehler hin. Abbildung 5.7 stellt den zu erwartenden Winkelfehler, welcher sich durch den Segmentverlauf bei 3 mm Höhe ergibt, dar. Abbildung 5.8 vergleicht einen gemessenen mit einem berechneten Kapazitätsverlauf und zeigt damit die Qualität der Simulation.

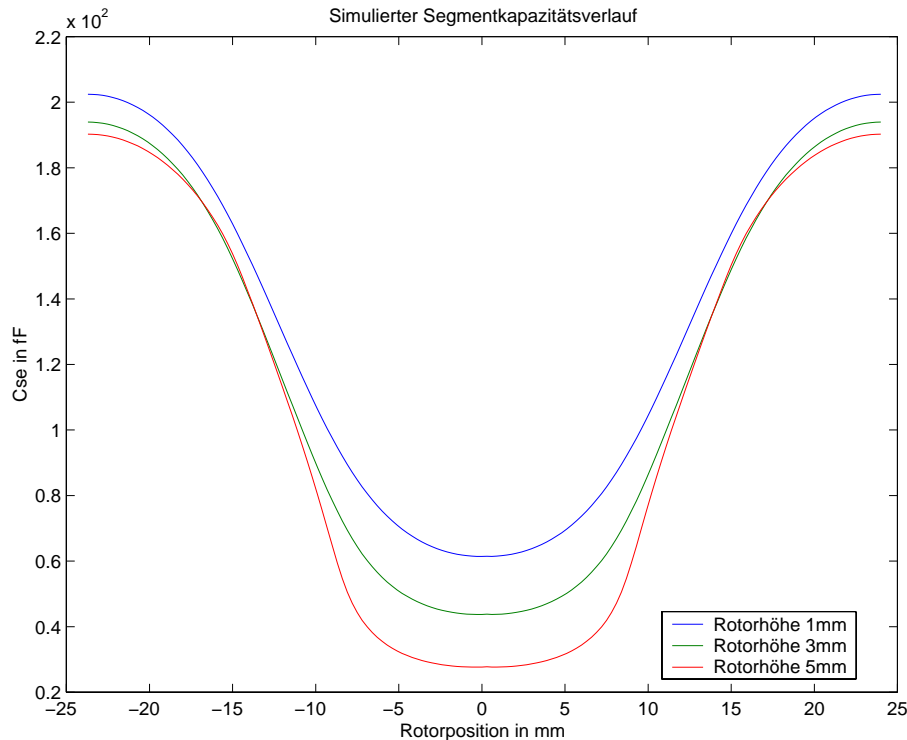


Abbildung 5.6: Vergleich des Segmentverlaufes bei verschiedenen Rotorhöhen. Man erkennt, daß hauptsächlich der minimale Kapazitätswert beeinflusst wird.

## 5.5 Simulationsanwendung

Nachdem es mit der Funktion `berechneCse` möglich ist den Kapazitätsverlauf zu berechnen und somit auf den resultierenden Winkelfehler zu schließen, kann der Einfluß verschiedener

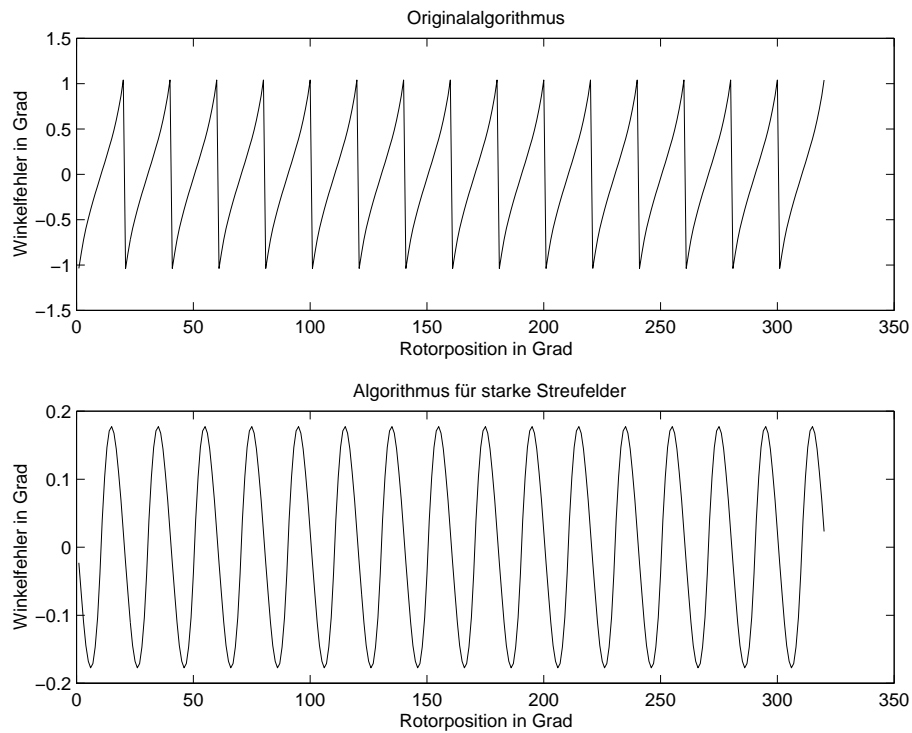


Abbildung 5.7: *Simulierter Winkelfehlerverlauf bedingt durch den Streufeldverlauf bei zwei unterschiedlichen Auswertalgorithmen.*

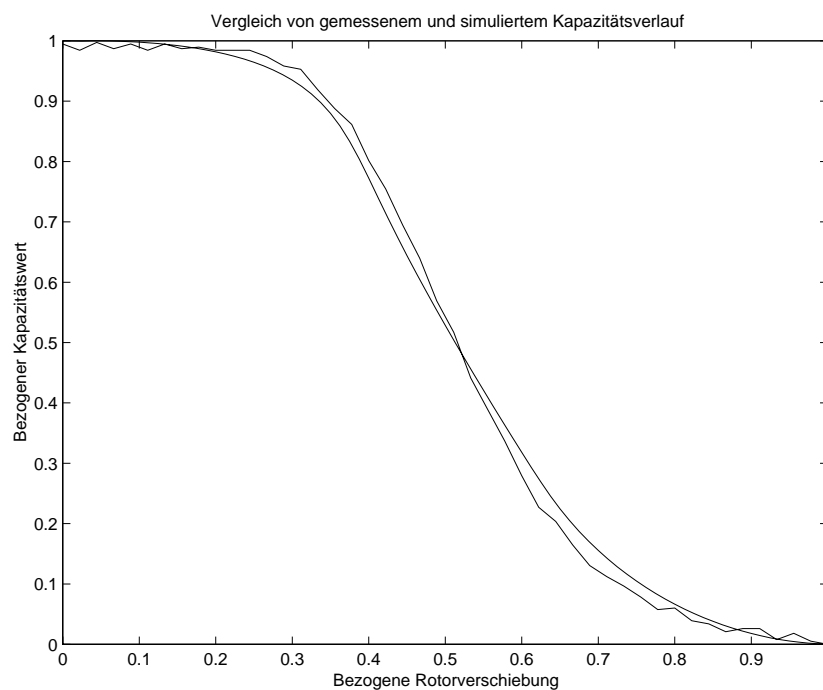


Abbildung 5.8: *Vergleich eines berechneten und eines gemessenen Kapazitätsverlaufes.*

Geometrieparameter simuliert werden. Dazu wird jeweils ein Parameter variiert und der dazugehörige Winkelfehlerverlauf während der Rotordrehung, wie in Abbildung 5.7, berechnet.

Der dabei maximal auftretende Fehler wird als stellvertretend für den Fehlerverlauf angesehen, und es wird die Abhängigkeit dieses maximalen Fehlers über Geometrieparameter gesucht. Ihr Minimum ist die optimale Wahl des Parameters.

### 5.5.1 Die optimale Sensorhöhe

In diesem Kapitel soll nun der Einfluß der Senderhöhe auf den Winkelfehlerverlauf getestet werden. Mit steigendem Sender-Empfänger Abstand vergrößert sich der Einfluß der Streufelder und ein größerer Winkelfehler wird erwartet. Ein größerer Sender- Empfängerabstand erlaubt aber eine Erhöhung der zulässigen Toleranz in Achsrichtung. Es ist also von Interesse, die zu erwartende Fehleränderung zu kennen. Es wurde die Feldsimulation dazu verwendet, um auf den zu erwartenden maximalen Winkelfehler bei verschiedenen Sensorhöhen zu schließen.

Die Sensorhöhe wurde im Bereich von 2 mm bis 17 mm mit einer Schrittweite von 0.5 mm variiert. Der Rotor wurde in dieser Simulation immer in der Sensormitte gehalten. Für jede Sensorhöhe wurde der Kapazitätsverlauf berechnet. Aus diesen Daten wurden dann Segmentverläufe generiert und diese anschließend mit den Algorithmen 1 und 2 ausgewertet (Vergl. Kapitel 5). Aus dem resultierenden Winkeldaten wurde dann der maximale Fehler gesucht, und mit der dazugehörigen Sensorhöhe gespeichert.

Das folgende Skript führt diese Simulation durch

```
index=0;
for hoch=2:0.5:17

    index=index+1;

    for i=-80:0;
        [cres]=Berechncse(i*0.3e-3, hoch*1e-3/2, hoch*1e-3);
        Cse(i+81)=cres;
    end;

    x=(-80:80)/10;

    seg=[Cse Cse(79:-1:1)];
    seg=[seg seg];
    seg=[seg seg];
    seg=[seg seg];
    seg=[seg seg];

    for i=1:8
        data_seg(i,1:320)=seg((1:320)+(i-1)*20);
    end

    data.RefPhi=((0:319)+0.5)/320*360)';
```

```

data.RefPhi=data.RefPhi+78.75;
data.RefPhi=mod(data.RefPhi,180);

data=EvalOffset(data);
data=EvalPhi(data,'42.0');
data.Phiold=mod(data.Phiold,180);
data=EvalPhi(data,'42.1');
data.Phinew=mod(data.Phinew,180);

Fehler.old(index)=max(abs(mod(data.Phiold-data.RefPhi'+90,180)-90));
Fehler.new(index)=max(abs(mod(data.Phinew-data.RefPhi'+90,180)-90));
Fehler.Rotorhoch(index)=hoch;

subplot(2,1,1);
plot(Fehler.Rotorhoch,Fehler.old);
subplot(2,1,2);
plot(Fehler.Rotorhoch,Fehler.new);

pause(0.5);

end;

subplot(2,1,1);
xlabel('Sensorhöhe d in mm')
ylabel('Fehler in Grad')
title('Einfluß der Sensorhöhe auf den maximalen Winkelfehler.')
```

```

subplot(2,1,2);
xlabel('Sensorhöhe d in mm')
ylabel('Fehler in Grad')
```

Die Abbildung 5.9 ist das Ergebnis der zuvor beschriebenen Simulation. Die Segmentdaten wurden mit den zwei Varianten der Algorithmen ausgewertet. Der erste Algorithmus basiert auf der Originalformel, wie sie in Kapitel 3.2.3.2 vorgestellt wurde. Hierbei dienen die zwei am meisten abgedeckten Segmente als Minimumreferenz, und die beiden am wenigsten abgedeckten Segmente als Maximumreferenz. Es zeigt sich aber, daß mit wachsenden Streufeldern immer schon eines der beiden Signale zu variieren beginnt, und dadurch das Referenzsignal (Max-Min) nicht konstant bleibt, sondern variiert. Dadurch entsteht der wesentliche Anteil des Winkelfehlers. Nimmt man in der ratiometrischen Formel nicht beide Maxima (und Minima) Segmente, sondern wählt gezielt das größere (kleinere) aus, kann dieser Effekt zum Großteil eliminiert werden, es läßt sich, wie Abb. 5.9 zeigt, eine Genauigkeitssteigerung erreichen. Diese beiden Varianten wurden in der Simulation verglichen.

Im oberen Teil der Abbildung 5.9 ist der Fehler beim Originalalgorithmus (Algorithmus 1) dargestellt. Er wächst zunächst, wie zu erwarten mit wachsender Sensorhöhe an, beginnt aber dann ab zirka 14 mm wieder zu fallen. Durch Betrachtung der Segmentverläufe ist

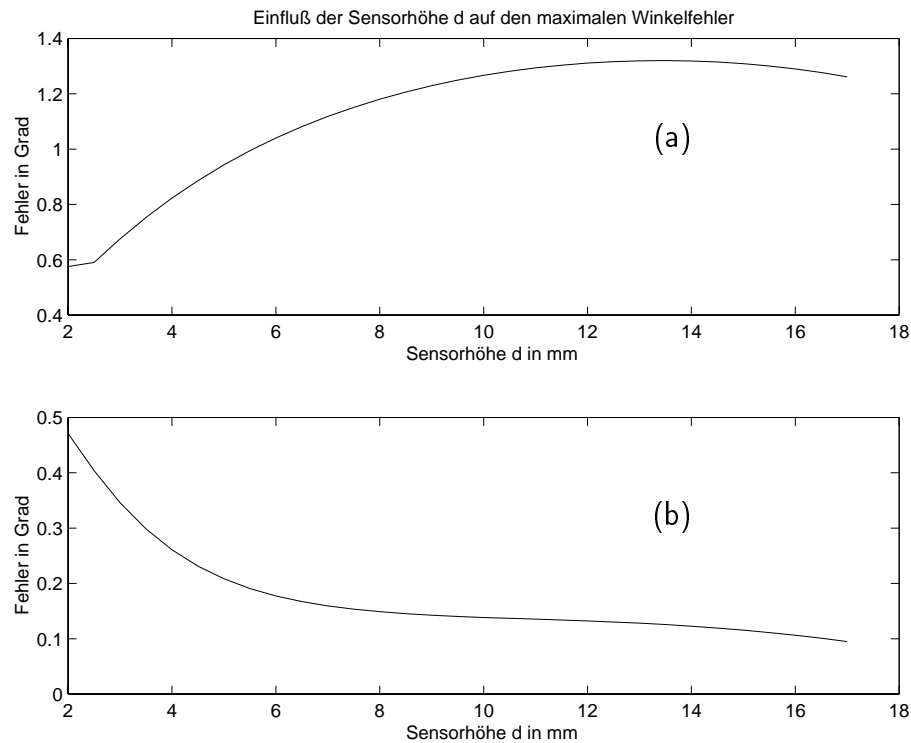


Abbildung 5.9: Simulierter Einfluß der Sensorhöhe auf den maximalen Winkelfehler. Es zeigt sich, daß beim Algorithmus 2 (b) der Fehler durchgehend kleiner ist als beim Basisalgorithmus (a), und sogar bei größeren Abständen  $d$  kleinere Werte zeigt. Algorithmus 2 ist daher bevorzugt einzusetzen.

ersichtlich, daß ab dieser Entfernung die Streufelder soweit reichen, daß die Simulation, welche anstatt der benötigten periodischen Rotorflügel Struktur nur 2 Ersatzflügel verwendet, nicht mehr gültig ist. Für diese Abstände wäre ein 3. Rotorflügel in der Simulation notwendig, um die eigentlich periodische Rotorstruktur wieder hinreichend zu nähern.

Der verbesserte Algorithmus 2 weist bei größeren Plattenabständen einen wesentlich geringeren Winkelfehlerverlauf auf.

### 5.5.2 Die optimale Rotorhöhe

Im für mechanischen Achsversatz optimalen Fall befindet sich der Rotor in der Mitte des Sensors, wodurch ein Maximum an mechanischer Versatztoleranz ermöglicht wird. Hier soll nun untersucht werden, wie das Optimum aus der Sichtweise des minimalen Winkelfehlers aussieht.

Hierbei wird von einer konstanten Sensorhöhe von 6 mm ausgegangen. Die Rotorhöhe wird von 0,3 mm bis 5,7 mm in 0,05 mm Schritten erhöht. Für jede Rotorhöhe wird der Kapazitätsverlauf, wenn der Rotor  $R$  unter dem Sender  $S$  durchtritt, berechnet. Mit diesen Daten wird ein Segmentverlauf erstellt, den Auswertealgorithmen übergeben und der Winkelfehler berechnet. Damit wird ein Datensatz maximaler Fehler über Rotorhöhe generiert.

```

index=0;

for hoch=0.1:0.05:5.8

    index=index+1;

    for i=-80:0;
        [cret]=berechnecse(i*0.3e-3,6e-3,hoch*1e-3);
        Cse(i+81)=cret;
    end;

    x=(-80:80)/10;

    seg=[Cse Cse(79:-1:1)];
    seg=[seg seg];
    seg=[seg seg];
    seg=[seg seg];
    seg=[seg seg];

    for i=1:8
        data.seg(i,1:320)=seg((1:320)+(i-1)*20);
    end

    data.RefPhi=((0:319)+0.5)/320*360)';

    data.RefPhi=data.RefPhi+78.75;
    data.RefPhi=mod(data.RefPhi,180);

    data=EvalOffset(data); data=EvalPhi(data,'42.0');
    data.Phiold=mod(data.Phiold,180); data=EvalPhi(data,'42.1');
    data.Phinew=mod(data.Phinew,180);

    Fehler.old(index)=max(abs(mod(data.Phiold-data.RefPhi'+90,180)-90));
    Fehler.new(index)=max(abs(mod(data.Phinew-data.RefPhi'+90,180)-90));
    Fehler.Rotorhoch(index)=hoch;

    subplot(2,1,1);
    plot(Fehler.Rotorhoch,Fehler.old);
    subplot(2,1,2);
    plot(Fehler.Rotorhoch,Fehler.new);
    pause(0.5);

end;

subplot(2,1,1);
xlabel('Rotorhöhe in mm')

```



```

ylabel('Fehler in Grad')

title('Einfluß der Rotorhöhe auf den Winkelfehler')

subplot(2,1,2);
xlabel('Rotorhöhe in mm')
ylabel('Fehler in Grad')

```

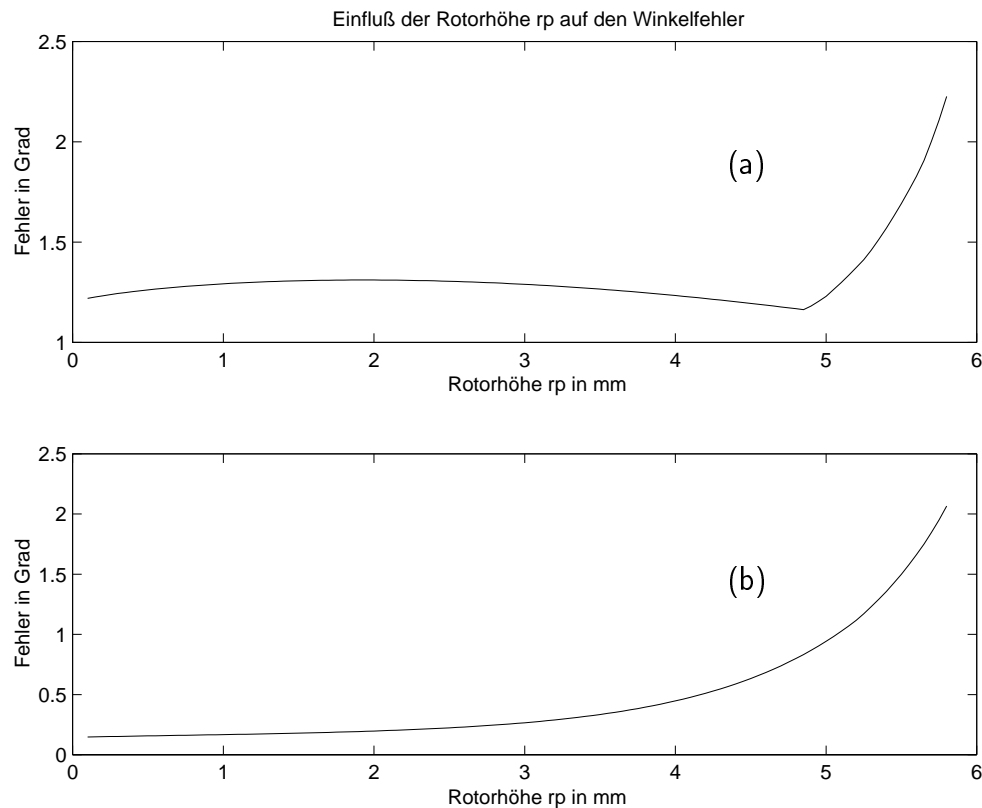


Abbildung 5.10: *Simulierter Einfluß der Rotorhöhe auf den Winkelfehler bei Verwendung des Basisalgorithmus (a) und des Algorithmus 2 (b).*

Die Abbildung 5.10 stellt den resultierenden Fehlerverlauf bei Rotorhöhenvariation dar. Dabei wurden die zwei Arten des Algorithmus verglichen. Der Fehlerverlauf (a) ergibt sich durch Anwenden des Originalalgorithmus 1 wie in 3.2.3.2 dargestellt ist. Der Algorithmus 2 (b) berücksichtigt, daß sich nicht alle aus der linearen Theorie vorausgesagten Maximum- und Minimumsignale tatsächlich noch in ihren Extremwerten befinden. Dadurch läßt sich eine wesentliche Genauigkeitssteigerung erreichen. Die Simulation zeigt, daß Rotorpositionen in der Nähe der Empfangsfläche zu bevorzugen sind.

### 5.5.3 Einfluß der Rotorverkippung

Eine Anwendung der Simulation befindet sich in [ZDI99], wo aus der Rotorverkippung der resultierende Relativwinkelfehler simuliert wird. Es wird hier gezeigt, daß schon durch eine

Rotorverkipfung von  $1/2$  mm auf einer Distanz von 95 mm (also  $0.32^\circ$  Schiefelage) ein Fehler im Bereich von  $0.25$  Grad entstehen kann. Die Simulationsergebnisse stimmen, wie aus Abbildung 5.11 ersichtlich, mit den gemessenen Ergebnissen in einem weiten Bereich überein.

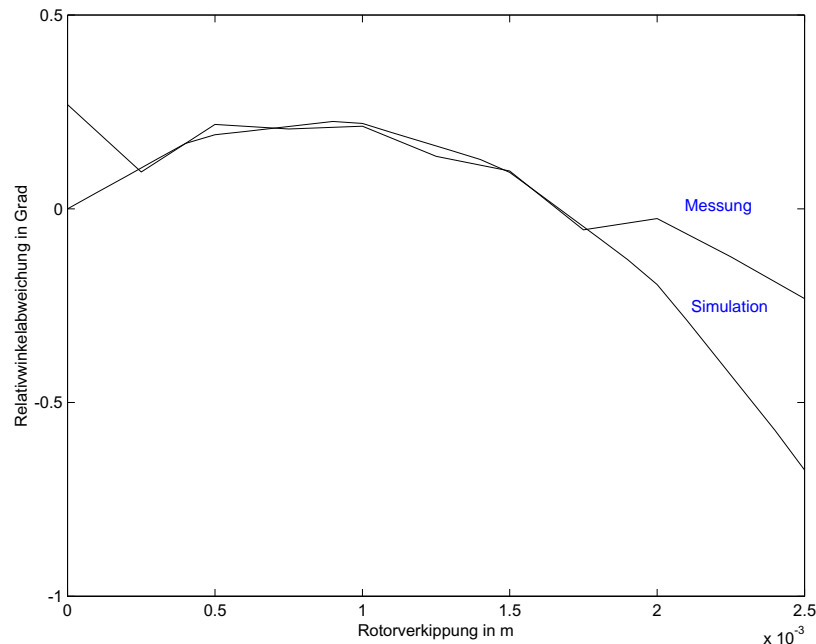


Abbildung 5.11: Einfluß der Rotorverkipfung auf die Relativwinkelabweichung (Vergleich Simulation und Messung)

## 5.6 Resumé

In allen untersuchten Fällen war der maximale Fehler bei Algorithmus 2 kleiner als bei Algorithmus 1. Wie [ZDI99] gezeigt hat, können schon Rotorverkipfungen um  $0.32^\circ$  zu zusätzlichen Fehlern führen, welche in der Größe des momentanen Fehlers sind (vergleiche dazu Kapitel 8). Die Auswertung mit den momentan auftretenden Streufeldern führt bei Algorithmus 2 zu keiner Beeinträchtigung der Meßresultate. Ein kleinerer Sensordurchmesser führt zu einem kleineren Rotorversatz bei Rotorverkipfung. Da sich der Einfluß des Verkipfungsfehlers minimiert, und die Simulation andeutet, daß die gesteigerten Streufelder sich nicht negativ auf den Winkelfehler auswirken, sollte so die Sensorgenauigkeit gesteigert werden können.

Die Simulation konnte auch deutlich aufzeigen, daß ein empfangernaher Rotor den Winkelfehler reduziert.

Vor allem das überraschende Resultat, daß der Winkelfehler nicht notwendigerweise mit steigenden Streufeldeinfluß mitsteigen muß, und die Sensorgröße daher reduziert werden sollte, lieferte einen entscheidenden Hinweis für die Weiterentwicklung des Sensors.

# Kapitel 6

## Die Sensor-Software

Die in [WAN99] entwickelte Sensor-Hardware besteht aus einem HC12 Mikrocontroller der Segment-Ansteuerung, einem Analog-Empfangsteil, einem seriellen Interface sowie einem PWM Ausgang.

Es wurde für den Sensor eine Software entwickelt, welche zyklisch, alle 20 ms, die Segmentmeßwerte erfaßt und den Winkel sowie den Relativwinkel berechnet. Das zyklische Messen garantiert zum einen, daß der PWM Ausgang des Sensors immer aktuelle Daten bereitstellt, andererseits wird es dadurch möglich, die Meßsignale auf einem Oszilloskop darzustellen. Durch die periodische Auswertung des Winkels kann leicht auf die Drehzahl und in Kombination mit dem Moment auf die Wellenleistung geschlossen werden. Der PWM Ausgang liefert den aktuellen Winkelmeßwert. Weiteres kann über das serielle Interface ein Datensatz bestehend aus Segmentmeßwerten, Grob- und Feinwinkel sowie der Relativwinkel ausgelesen werden.

Diese Sensor-Software wurde in ANSI C entwickelt. Als Compiler diente ein Compiler der Firma HIWARE. Zum Debuggen und Flashen der Software wurde ein Interface der Firma LAUTERBACH verwendet.

Die Software ist in diesem Kapitel beschrieben und teilt sich in folgende Komponenten:

- *main* – Der Hauptdatei fällt lediglich die Aufgabe der Initialisierung zu, da der Sensor ausschließlich auf Interrupt Ereignisse reagiert.
- *per\_int* – Der periodische Interrupt stellt im wesentlichen eine Echtzeitumgebung dar. Er organisiert den periodischen Aufruf der für die Berechnung wesentlichen Prozeduren.
- *measure* – Hier sind alle Funktionen für die Meßwerterfassung zusammengefaßt.
- *algorithm* – Hier ist der Algorithmus abgesetzt, welcher für die Bestimmung der Winkelgrößen aus den Meßwerten notwendig ist.
- *PWM* – Die Sensor-Hardware verfügt auch über einen PWM-Ausgang, der wahlweise mit einer Meßgröße belegt werden kann. Die Prozeduren zum Arbeiten mit dem PWM-Ausgang sind hier gekapselt.

- *ser* – Die serielle Schnittstelle dient zur Übertragung der Meßwerte zu einem PC, um die Meßwertauswertung zu ermöglichen. In der Bibliothek “*ser*” sind alle dafür notwendigen Funktionen eingebunden.

Die Abbildungen 6.1, 6.2 und 6.3 geben einen Überblick der Ablauffolge des Programms.

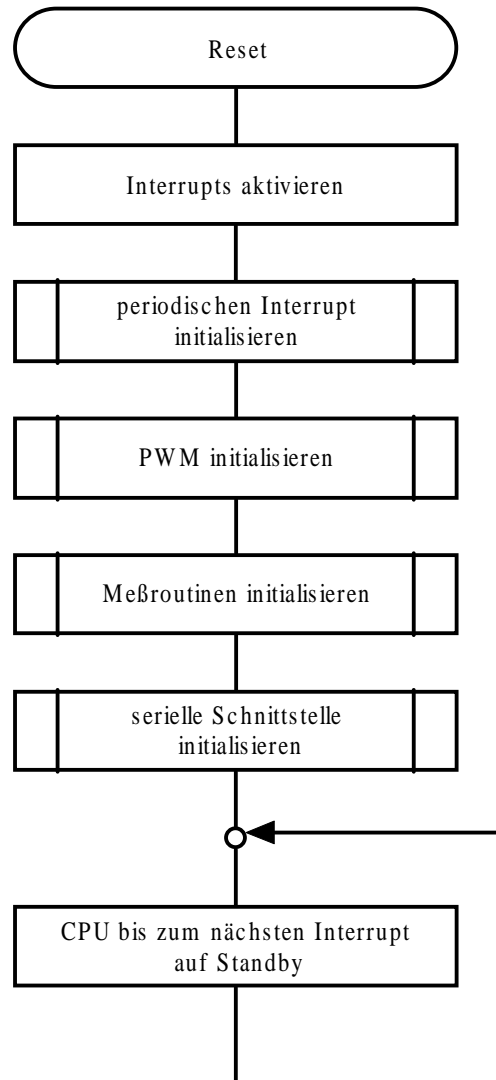


Abbildung 6.1: *Flußdiagramm der Hauptdatei - “main”*

## 6.1 Die Hauptdatei

### 6.1.1 main.h

Zuerst soll die Struktur der Datei “*main.h*” vorgestellt werden. Dann wird kontrolliert, ob die Datei “*main.h*” schon einmal eingebunden wurde. Wenn ja, werden die Deklarationen in ihr durch einen Makrocompilerbefehl übersprungen. Auf diese Weise lassen sich Mehrfachdeklarationen verhindern. Dies passiert durch die Definition der Variablen “*\_main*”.

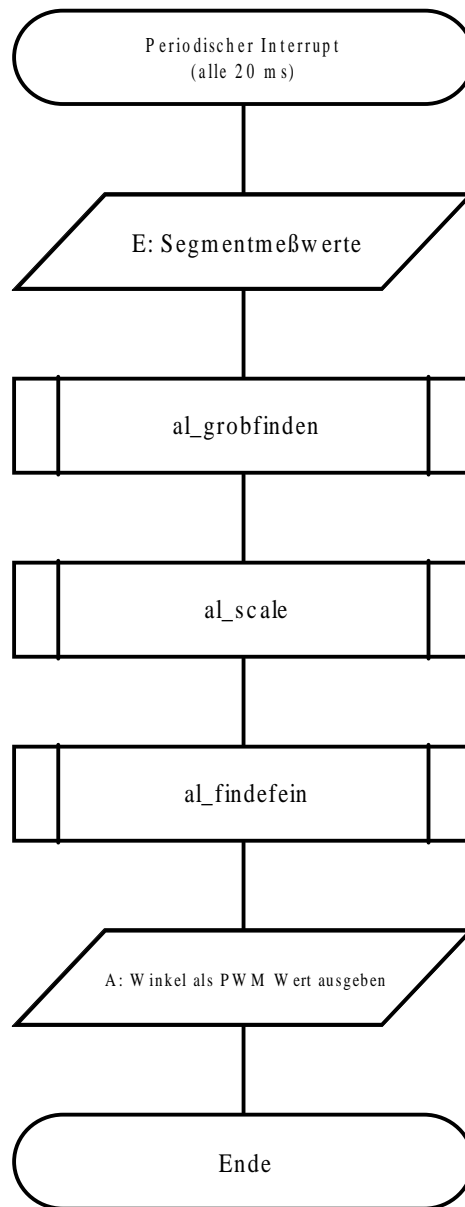


Abbildung 6.2: Flußdiagramm des periodischen Interrupts - "per\_int"

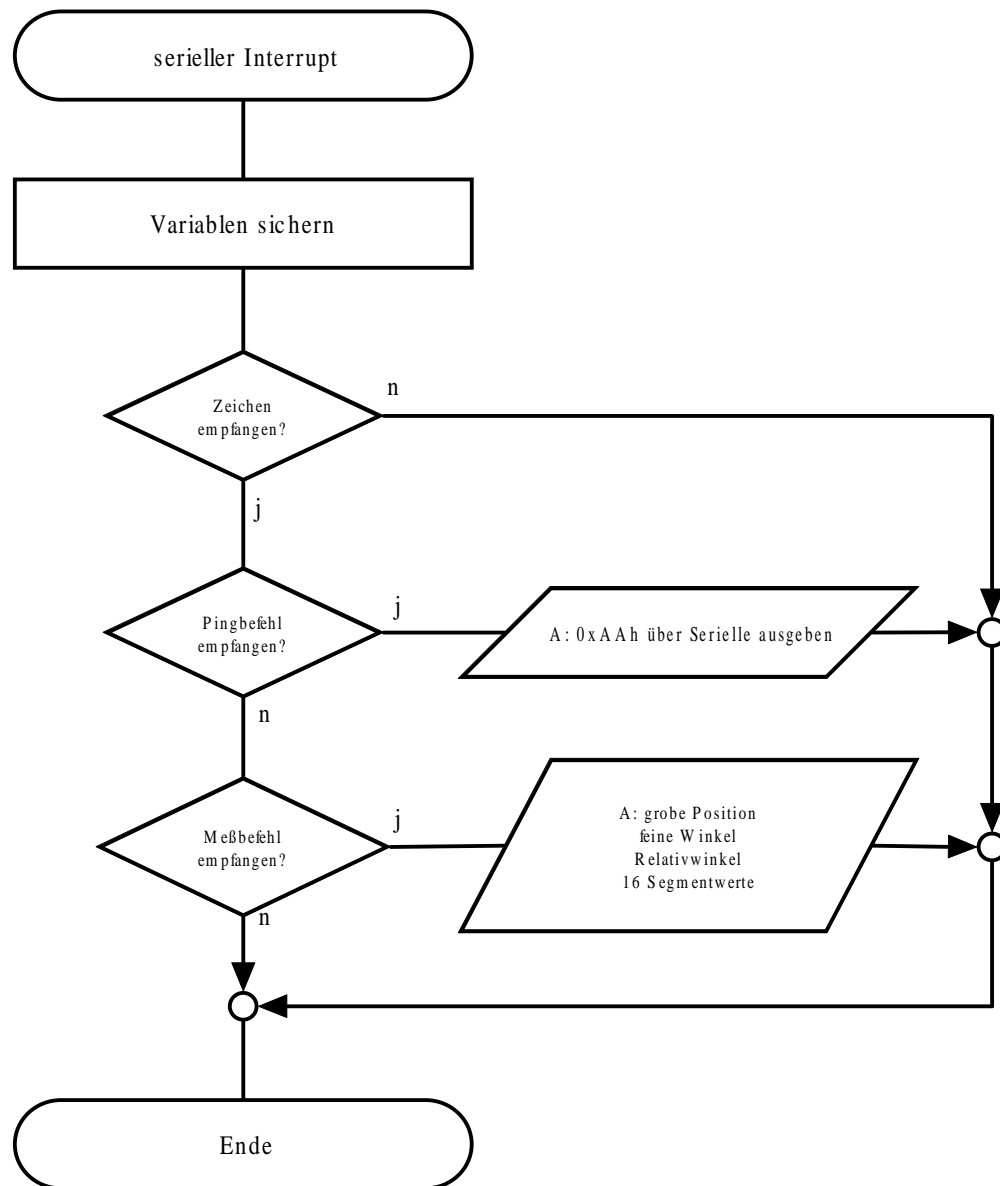


Abbildung 6.3: Flußdiagramm des seriellen Interrupts - "ser"

```
#ifndef _main
#define _main
.
.
.
#endif
```

Die wichtigsten den Algorithmus betreffenden Variablen werden global definiert:

```
extern int segment[16];
extern int scaled[16];
extern unsigned char grob;
extern long deltaalpha;
extern long alpha;
```

Der Sensor erfaßt 16 Segmentmeßwerte. Die Meßwerte werden in einem Array namens *“segment”* vom Typ Integer erfaßt. Eine wesentliche Verminderung des Meßfehlers läßt sich durch eine Skalierungstabelle für die Segmente erreichen, siehe dazu Kapitel 6.4.2.5. Diese aufbereiteten Segmentmeßwerte werden in einem globalen Array namens *“scaled”* gespeichert. Wie schon in Kapitel 3.2.3.3 beschrieben, muß die grobe Rotorlage vor dem Algorithmus für die Winkelauswertung bekannt sein. Die grobe Rotorlage in 16'tel-Winkelmeßbereichsschritten (22.5° oder 11.25° pro LSB-Schritt) wird in der 8-Bit Variablen *“grob”*, die berechneten Werte für den Relativwinkel und den Winkel werden in den Variablen *“deltaalpha”*, und *“alpha”* gespeichert. Bei beiden Variablen ist die LSB Schrittweite je 1/1000 Grad. Diese Auflösung ist wesentlich höher als die Meßgenauigkeit, so kann sichergestellt werden, daß die Rechengenauigkeit zu Folge der Rundungsfehler vernachlässigbar ist. Diese hohe Auflösung ermöglicht es, gut aufgelöste Fehlerkurven zu zeichnen, wodurch systematische Meßfehler leichter erfaßt werden können. Durch die hohe Auflösung müssen die beiden Winkelwerte in einer 32-Bit-Variablen gespeichert werden.

Im nächsten Programmschritt werden alle benötigten Programmbibliotheken eingebunden.

```
#include <hidef.h>
#include <float.h>
#include <stdlib.h>
#include <math.h>
#include <signal.h>
```

Das Programm ist aus Übersichtlichkeitsgründen in mehrere Source-Dateien aufgeteilt. Zu jeder C-Datei gehört eine Header-Datei, in welcher die exportierten Funktionen deklariert werden. Diese Header-Dateien werden in die Datei *“main.h”* importiert.

```
#include "ramreg.h"
#include "per_int.h"
#include "pwm.h"
#include "measure.h"
#include "ser.h"
#include "algorithm.h"
```

Den importierten Dateien sind in diesem Abschnitt einzelne Kapitel gewidmet, mit Ausnahme der Datei *“ramreg.h”*. Sie definiert die einzelnen Pointer auf die Prozessorregister. Die Bezeichnungen entsprechen dabei den in [MOT98] verwendeten Registernamen.

Da es bei verschiedenen Prozessoren unterschiedliche Definitionen der booleschen Werte *true* und *false* gibt, ist es sinnvoll, diese einmal im Programm zu definieren.

```
#define true      (1==1)
#define false    !(1==1)
```

## 6.1.2 main.c

Nach der Header-Datei kann nun die Hauptdatei *“main.c”* folgen. Zuerst wird natürlich die Header-Datei eingebunden und diese in *“main.h”* als global exportierte Variablen definiert.

```
#include "main.h"

int segment[16];
int scaled[16];
unsigned char grob;
long deltaalpha;
long alpha;
```

### 6.1.2.1 main.c - main\_init

Alle Initialisierungsaufgaben sind in der Funktion *main\_init* zusammengefaßt. In ihr werden zuerst zwei Prozessorregister gesetzt, wodurch der Watchdog des Controllers aktiviert wird. Der Watchdog dient dazu, ein Festlaufen (z.B. Endlosschleifen) des Programms zu verhindern. Das ausführende Programm muß in zyklischen Abständen ein Register beschreiben. Wenn das Programm jedoch aus irgendeinem Grund "stecken" bleibt, wird der Prozessor nach Ablauf der festgelegten Watchdogzeit resettet.

```
void main_init(void)
{
    RTICTL |= 0x20;
    COPCTL |= 0x07;
```

Zuerst wird das 5. Bit des RTICTL Registers gesetzt. Es hat den Namen RSBCK (Realtime Stop Background Mode). Dadurch wird sowohl die Echtzeituhr als auch der Watchdog im Backgroundmodus gestoppt. Dies ermöglicht ein Debuggen des Prozessors, ohne daß ein Watchdog ausgelöst wird. Die unteren 3 Bits des COPCTL Registers definieren die Zeitbasis des Watchdogs. Befinden sich alle 3 auf Null, so ist der Watchdog ausgeschaltet. Sind sie alle auf eins gesetzt, ist der Watchdog auf  $X_{Clock}/2^{23}$  programmiert. Der  $X_{Clock}$  ist ein interner Prozessor-Takt, welcher die Hälfte des Quarztaktes hat. Da der Prozessor



mit einem Quarz von 15 MHz betrieben wird, wird hier die Watchdogzeit auf 1.12 sec. ( $= 1 / (\frac{f_{\text{quarz}}}{2^{23}})$ ) gesetzt.

Als Taktquelle für den Prozessor wird der externe Quarz gewählt, dies geschieht durch Setzen des 5. Bits des CLKSEL Registers.

```
CLKSEL |= 0x20;
```

Am Ende der Funktion *“main\_init”* werden die Prozessorinterrupts eingeschaltet und die Init-Funktionen der restlichen Programmteile aufgerufen.

```
EnableInterrupts;

pi_Init();
pwm_Init();
me_Init();
ser_Init();
}
```

#### 6.1.2.2 main.c - main

Die Messung wird durch einen periodischen Interrupt ausgeführt. Der Kommunikationsprozeß über die serielle Schnittstelle ist ebenfalls auch interruptgesteuert, dadurch bleibt dem Hauptprogramm nach der Initialisierung nichts mehr zu tun. Der Prozessor wird über den Assemblerbefehl *“wai”* in einen Schlafmodus gesetzt. Aus diesem erwacht er mit dem nächsten Interrupt. Ist die Interruptserviceroutine abgearbeitet, wird das Programm nach dem Waitbefehl fortgesetzt. Daher ist der Waitbefehl in eine Endlosschleife eingebunden.

```
void main(void)
{
    main_init();

    while(true)
    {
        asm wai
    };
}
```

## 6.2 Der periodische Interrupt

### 6.2.1 per\_int.h

Die Datei *“per\_int.h”* verriegelt sich mit der gleichen Methode gegen Mehrfacheinbindungen, wie es in Kapitel 6.1.1 schon erklärt ist. In diesem Fall wird eine Variable namens *“\_per\_int”* verwendet. Exportiert wird in diesem Falle nur die Initfunktion *“pi\_init”*.

```
#ifndef _per_int
    void pi_Init(void);
#define _per_int
#endif
```

## 6.2.2 per\_int.c

### 6.2.2.1 per\_int.c - pi\_init

Nach Einbinden der Datei *“main.h”* wird zuerst die Initialisierungsroutine *“pi\_init”* deklariert.

```
#include "main.h"
```

```
void pi_Init(void)
{
    MCCTL    = 0x00;
    MCCTL    = 0xCC;
```

Zuerst wird das MCCTL Register zurückgesetzt. Anschließend werden die Bits 2, 3, 6 und 7 auf HIGH gesetzt. Bei gesetztem Bit 7 wird ein Interrupt ausgeführt, wenn der Zählstand durch Null läuft. Durch Setzen von Bit 6 wird der Zähler anschließend selbständig wieder auf den im MCCNT Register gespeicherten Wert gesetzt. Durch Setzen von Bit 3 wird das MCCNT Register mit dem im Force Load Register gespeicherten Wert gefüllt. Mit Bit 2 wird der “Modulus Down Counter” eingeschaltet.

```
MCCNT    = 0x0100;
MCCTL    = 0xCD;
MCCNT    = 0x927C;
}
```

In das MCCNT Register wird nun 100h geschrieben. Anschließend wird durch das MCCTL Register ein Laden des Zählers verursacht. Da dabei das nullte Bit 1 und das erste Bit 0 ist, wird ein Takteiler von 4 gegenüber dem M-Clock gewählt. Der Zähltakt ist somit 1.875 MHz. Der Zählstand von 256 entspricht daher 136  $\mu$ s. Die Zeit ist so gewählt, daß der Prozessor nach dieser Zeit mit der Initialisierung fertig sein wird. Nach Verstreichen dieser Zeit wird der erste Interrupt zur Meßwerterfassung ausgewählt. Mit dem Interrupt wird dann jeweils der aktuelle Wert des MCCNT Registers geladen. Um die gewünschte Zeitbasis von 20 ms zu erreichen, muß das MCCNT Register mit 927Ch geladen werden.

### 6.2.2.2 per\_int.c - pi\_PeriodicInterrupt

Bei dieser Funktion handelt es sich um eine Interruptserviceroutine. Sie besitzt den Interrupt Vektor 25 und wird deshalb beim Auftreten des Moduls Down Counter Interrupts ausgeführt. Gleich am Anfang wird das 7. Bit des MCFLG gesetzt, dies bewirkt ein Löschen des Interruptflags. Anschließend wird das Watchdog Register beschrieben, um den Watchdog zurückzusetzen.

```
void interrupt 25 pi_PeriodicInterrupt(void)
{
    MCFLG  |= 0x80;
    COPRST  = 0x55;
    COPRST  = 0xAA;
}
```

Nun werden der Reihe nach Funktionen aufgerufen, welche zuerst die Segmente messen, dann die grobe Position suchen, die gemessenen Segmentwerte skalieren und die Winkel und Relativwinkelmeßwerte errechnen. Zuletzt wird eine Funktion zur PWM Ausgabe aufgerufen. Sie bekommt als Parameter den auszugebenden Wert und den maximalen auszugebenden Wert.

```
me_Measure_seg();
al_grobfinden();
al_scale();
al_findefein();
pwm_SetRatioPWM (alpha, 3600000);
}
```

## 6.3 Die Meßroutine

### 6.3.1 measure.h

Die Header-Datei definiert hier zwei zu exportierende Funktionen. Es handelt sich hierbei um die Initialisierungsfunktion *“me\_Init”* und die Meßfunktion *“me\_Measure\_seg”*. Diese Funktion mißt die 16 Segmentmeßwerte und legt diese in der globalen Variablen *“segment”* ab.

```
#ifndef _measure
void me_Init(void);
void me_Measure_seg(void);
#define _measure #endif
```

### 6.3.2 measure.c

Zuerst werden einige Makros definiert, um das Programm lesbarer zu machen. Dadurch läßt sich das Programm auch leichter auf Hardwareänderungen anpassen.

```
#include "main.h"

#define EnableBinaryPorts DDRA = 0xff; DDRB = 0xff;
#define AmpOn  PORTP &= 0x7F
#define AmpOff PORTP |= 0x80
#define WakeUpAnalog 50
#define WaitSeg 100
```

Die Treiber für die 16 Sendesegmente sind mit den beiden 8 Bit breiten Ports A und B (siehe [WAN99]) verbunden. Sie müssen beide als Ausgang gesetzt sein. Dies geschieht, in dem die beiden Datenrichtungsregister(DDRA und DDRB) auf FFh gesetzt werden. Diese beiden Zuweisungen wurden als Makro *“EnableBinaryPorts”* definiert.

Um Strom zu sparen, wird die Meßschaltung zwischen den Messungen abgeschaltet. Als Steuerleitung dient hierbei die 8. Leitung des Port P. Ist diese Leitung auf HIGH, so ist der Empfangsteil deaktiviert. Für das Ein- und Ausschalten wurden ebenfalls zwei Makros Namens *“AmpOn”* und *“AmpOff”* definiert.

Nach dem Einschalten des Analogteils dauert es eine kurze Zeit bis die Schaltung arbeitet. Danach können die Segmente gemessen werden, wobei es auch nach dem Einschalten der Segmente eine gewisse Zeit benötigt, bis der Ausgang der Analogschaltung seinen Endwert erreicht hat. Diese beiden Zeiten sind in den Konstanten *“WakeUpAnalog”* und *“WaitSeg”* gespeichert. Diese beiden Konstanten sind die Parameter für die Funktion *“sleep”*.

### 6.3.2.1 measure.c - sleep

Die Funktion *“sleep”* dient dazu, den Programmablauf für eine kurze Zeit anzuhalten. Dies geschieht durch eine Schleife. Der Befehl `#pragma inline` bewirkt, daß beim Compilieren nicht wie üblich eine Funktion erstellt wird, die über einen Sprungbefehl aufgerufen wird und die Variablen durch den Stack bekommt, sondern es wird die Funktion, überall wo sie aufgerufen wird, direkt eingefügt. Dadurch wird zwar das compilierte Programm etwas länger, dafür wird die Zeit, welche während des Aufrufens der Funktion vergeht, eliminiert.

Die Verzögerungszeit bei Aufruf der Funktion ergibt sich durch die Schleifenabarbeitungszeit, welche proportional zum übergebenen Parameter ist, und der Verzögerungszeit, die sich beim Aufruf ergibt. Um die Verzögerungszeit der Funktion *“sleep”* möglichst proportional zum übergebenen Parameter zu halten, soll die Verzögerungszeit beim Aufruf ein Minimum werden. Daher wurde der inline-Befehl verwendet.

```
#pragma INLINE
void sleep(int i)
{
    int l;
    for (l=0;l<i;l++) {}
}
```

### 6.3.2.2 measure.c - me\_init

Die Initialisierungsroutine `me_init` konfiguriert zuerst den AD-Umsetzer des Mikrocontrollers. Im ATD0CTL2 Register werden die Flags APDU, AFFC und AWAI gesetzt. Nach einem Reset ist der Analog-Digitalumsetzer aus Energieverbrauchsgründen deaktiviert. Zum Aktivieren muß das APDU Flag gesetzt werden. Das Setzen des AFFC Flag bewirkt, daß beim Beschreiben des Wandlerregisters automatisch das korrespondierende Statusregister rückgesetzt wird. Durch Setzen des AWAI Registers wird der AD-Umsetzer im Waitmodus des Prozessors deaktiviert.

Das Bit 8 des ATDCTL4 Register ist das S10BM Flag. Damit kann zwischen 10 und 8 Bit Wandlermodus ausgewählt werden. In dieser Anwendung ist das Flag gesetzt, der Wandler arbeitet daher im 10 Bit Modus. Die nächsten beiden Bits bestimmen die Sampletime. Sie sind beide auf 1 gesetzt, was eine maximale Sampletime von  $8,5\mu s$  bewirkt. Dies wurde gewählt, da eine höhere Sampletime geringeres Rauschen bewirkt. Die letzten 5 Bits dienen dazu, den Wandlertakt einzustellen. Dieser muß zwischen 500 kHz und 2 MHz liegen und wird vom P-Clock(7.5 MHz) hergeleitet. Das gewählte Teilverhältnis ist 4, damit ergibt sich ein Wandlertakt von 1.875 MHz. Als letztes wird das Makro *“EnableBinaryPorts”* aufgerufen, wodurch die Ports A und B als Ausgangsports definiert werden.

```
void me_Init(void)
{
    ATDOCTL2    = 0xE0;
    ATDOCTL4    = 0xE1;
    EnableBinaryPorts;
}
```

### 6.3.2.3 measure.c - me\_Measure\_ad

Diese Funktion dient zum Erfassen der an AD00 anliegenden Spannung. Dabei wird diese Spannung intern 8 mal gemessen, wodurch sich mit Hilfe statistischer Mittelung ein Auflösungsgewinn von 3 Bit ergibt, solange zumindest das LSB durch stochastisches Rauschen überlagert ist, siehe dazu [SRUF92]. Die Funktion bekommt beim Aufruf einen Pointer auf einen Integer übergeben, in welchen sie den Meßwert einträgt. Um die Aufrufzeit zu minimieren, wurde die Funktion wieder als Inline Funktion definiert. Zum Starten einer Wandlung muß das ATDOCTL5 Register beschrieben werden. Der Wert 40h bewirkt hierbei, daß der Pin AN00 acht mal in Serie gewandelt wird. Die gemessenen Werte werden dann in die acht Ausgangsregister geschrieben. Diese Register sind 16 Bit breit und liegen im Speicher ab Adresse 0x70h. Zu jedem der 8 Ergebnisregister gehört ein Statusflag im ATDOSTAT Register. Es wird gesetzt, sobald ein neuer Wert ins Ergebnisregister gelangt und wird beim Auslesen automatisch zurückgesetzt. Gleich nach jeder Einzelwandlung wird, während die nächste Wandlung beginnt, der neue Wert ausgelesen und verarbeitet. Durch die negative Spitzenwertgleichrichtung in der Meßsignalaufbereitung bewirkt eine größere Kapazität eine Verringerung der Meßspannung. Aus Übersichtlichkeitsgründen wird jedoch der Meßwert, indem er von 8192 subtrahiert wird, gespiegelt. Die möglichen Rückgabewerte liegen damit im Intervall zwischen 0 und 8192, wobei höhere Werte einer höheren Kapazität entsprechen.

```
#pragma INLINE
void me_Measure_ad(int * analogvalue)
{
    ATDOCTL5    = 0x40;
    (*analogvalue)=8192;
    while(!(ATDOSTAT & 0x01));
    (*analogvalue) -= (*(unsigned int*)(0x70) >> 6);
```

```

while(!(ATDOSTAT & 0x02));
(*analogvalue) -= (*(unsigned int*)(0x72) >> 6);
while(!(ATDOSTAT & 0x04));
(*analogvalue) -= (*(unsigned int*)(0x74) >> 6);
while(!(ATDOSTAT & 0x08));
(*analogvalue) -= (*(unsigned int*)(0x76) >> 6);
while(!(ATDOSTAT & 0x10));
(*analogvalue) -= (*(unsigned int*)(0x78) >> 6);
while(!(ATDOSTAT & 0x20));
(*analogvalue) -= (*(unsigned int*)(0x7A) >> 6);
while(!(ATDOSTAT & 0x40));
(*analogvalue) -= (*(unsigned int*)(0x7C) >> 6);
while(!(ATDOSTAT & 0x80));
(*analogvalue) -= (*(unsigned int*)(0x7E) >> 6);
}

```

#### 6.3.2.4 measure.c - me\_Measure\_seg

Nun folgt die Funktion *“me\_Measure\_seg”*. Sie wird von *“pi\_PeriodicInterrupt”* aus aufgerufen und aktualisiert die 16 Kapazitätsmeßwerte in der Variablen *“segment”*. Zuerst wird der Analogteil aktiviert. Nun pausiert das Programm eine kurze Zeit, bis der Analogteil einsatzbereit ist. Dann wird jeweils ein Pin des Ports A eingeschaltet, anschließend eine kurze Zeit gewartet und anschließend der Meßwert in die Variable *“segment”* übernommen. Dies geschieht anschließend auch mit Port B, bis alle 16 Meßwerte erfaßt sind. Zum Schluß deaktiviert die Funktion die Empfangsschaltung wieder.

```

void me_Measure_seg(void)
{
    int i;
    int out=1;

    AmpOn;
    sleep(WakeUpAnalog);
    for (i=0;i<8;i++)
    {
        PORTA= out;
        sleep(WaitSeg);
        me_Measure_ad(&segment[i]);
        out=out*2;
    }

    PORTA=0;
    out=1;
    for (i=0;i<8;i++)
    {
        PORTB= out;

```

```

    sleep(WaitSeg);
    me_Measure_ad(&segment[i+8]);
    out=out*2;
}
PORTB=0; AmpOff;
}

```

## 6.4 Der Algorithmus

### 6.4.1 Algorith.h

In ihr werden vier Funktionen exportiert. Die Aufrufreihenfolge ist dabei strikt vorgegeben. Das Gliedern in diese vier Teile dient nur dazu, um einzelne Befehlsgruppen leicht durch andere gleichwertige ersetzen zu können. Die Funktion *“al\_grobfinden”* muß als erstes nach dem Erfassen der Segmentwerte durch die Funktion *“me\_Measure\_seg”* aufgerufen werden. Sie ermittelt aus dem Array *“segment”* die grobe Rotorposition. Danach muß die Skalierungsfunktion *“al\_scale”* ausgeführt werden. Durch sie wird die Kalibrierungstabelle (siehe Kapitel 6.4.2.5) aktualisiert, und es werden die Meßwerte skaliert. Diese skalierten Meßwerte werden dann in das Array *“scaled”* geschrieben. Für Messungen ohne Skalierung steht alternativ die Funktion *“al\_noscale”* zur Verfügung, welche die Meßwerte aus dem Array *“segment”* einfach in das Array *“scaled”* verschiebt. Zuletzt wird durch Aufrufen der Funktion *“al\_findefein”* der eigentliche Algorithmus zur Winkel und Relativwinkelbestimmung aufgerufen.

```

#ifndef _algorithm
void al_grobfinden(void);
void al_noscale(void);
void al_scale(void);
void al_findefein(void);
#define _algorithm
#endif

```

### 6.4.2 Algorith.c

Es werden zwei globale Variablen definiert. Diese beiden Variablen werden von der Funktion *“al\_scale”* als Flags verwendet. Die Variable *“first”* zeigt an, daß es sich um den ersten Programmdurchlauf handelt. Ist *“first”* true müssen einige Initialisierungsaufgaben durchgeführt werden. Die Variable *“valid”* wird am Anfang auf *false* gesetzt. Wurde der Rotor so weit gedreht, daß die Segmentskalierungstabelle komplett ist, wird *“valid”* auf *true* gesetzt. Wie im Kapitel 3.2.3.3 beschrieben, wird die grobe Rotorlage durch Korrelation mit einem Datensatz bestimmt. Die in Kapitel 3.2.3.3 angegebenen Werte sind als Konstante *“correl\_sig”* definiert. Die dazugehörige Rotorposition ist in Abbildung 5.1 dargestellt.

```
#include "main.h"

int first=true;
int valid=false;

const int correl_sig[16]= {-3,-1, 3, 3,
                           3, 1,-3,-3,
                           -3,-3, 1, 3,
                           3, 3,-1,-3};
```

#### 6.4.2.1 Algorithm.c - al\_grobfinden

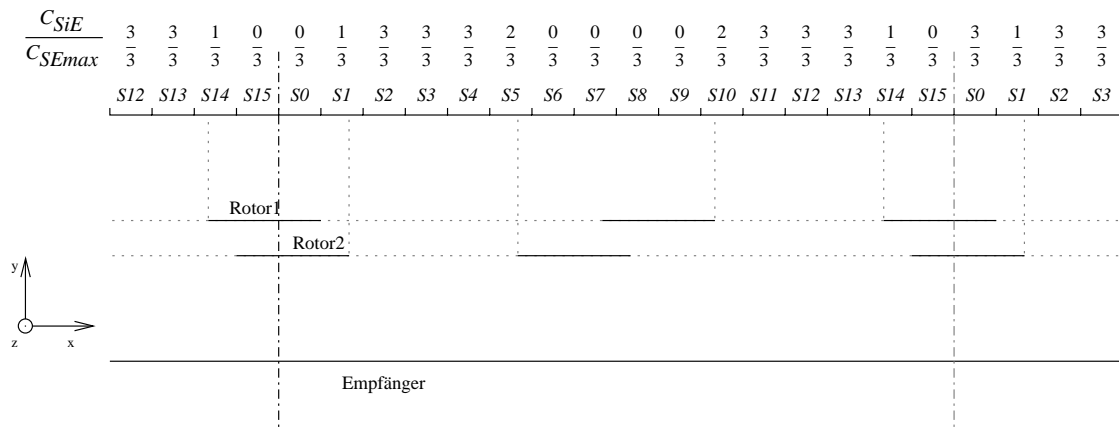


Abbildung 6.4: Planare Abwicklungsdarstellung des Sensors. Die beiden Rotoren sind in ihrer neutralen Lage (keine Relativverschiebung) eingezeichnet. Über den Segmenten ist die ohne Streufelder wirkende, auf die Segmentkapazität bezogene, Sender-Empfänger Kapazität eingetragen

Wie in Kapitel 3.2.3.3 beschrieben, wird die Korrelationsformel 3.16 16 mal angewandt. Dabei wird das Vergleichssignal jedesmal um 1 nach links verschoben. Die 16 Korrelationsergebnisse werden in dem Array "correl\_value" gespeichert.

```
void al_grobfinden(void)
{
    long correl_value[16];
    int checkpos,i;
    long maxvalue; int maxpos;

    for(checkpos=0;checkpos<16;checkpos++)
    {
        correl_value[checkpos]=0;
        for(i=0;i<16;i++)
```



```

    correl_value[checkpos]+=
        segment[i]*correl_sig[(i-checkpos+32)%16];
}

```

Die 16 Korrelationsergebnisse entsprechen nun dem Übereinstimmungsgrad mit den 16 ausgezeichneten Rotorlagen. Der Index des größten Korrelationsergebnis entspricht somit der groben Rotorlage.

```

maxvalue=matchedvalue[0];
maxpos=0;

for(checkpos=1;checkpos<16;checkpos++)
{
    if (correl_value[checkpos]>maxvalue)
        {maxvalue=correl_value[checkpos];maxpos=checkpos;}
}
grob=maxpos;
}

```

Die Segmentzugriffe erfolgen im restlichem Algorithmus nun immer mit der Transformation

$$\text{Virtuellersegmentindex} = \text{Segmentindex} + \text{grob} \quad (6.1)$$

Beim *Virtuellensegmentindex* ist die Rotorverschiebung kompensiert. Es darf damit immer von einer Konfiguration, wie sie in Abbildung 6.4 dargestellt ist, ausgegangen werden.

#### 6.4.2.2 Algorith.c - minimum

Im Kapitel 5.5.1 wurde erwähnt, daß der Meßfehler deutlich reduziert werden kann, wenn als Minimumsignalwert nicht die beiden Segmentwerte der abgedeckten Segmente, sondern das Kleinere der beiden verwendet wird. Die Begründung liegt darin, daß zwar beide Segmente abgedeckt sind, aber eines der abgedeckten Segmente schon so nahe an einer Rotorkante ist, daß dessen Streufelder schon über den Rotor hinausragen. Dieses Segment hat somit schon eine erhöhte Kapazität.

Um das Meßergebnis zu verbessern, muß daher der kleinste Segmentwert der abgedeckten Segmente genommen werden. Da in der Anwendung als Momentensensor die Segmente über beide Rotorflügel hinweg gemessen wird und unter jedem Rotorflügel 2 abgedeckte Segmente liegen, müssen vier Segmentwerte untersucht werden. Dazu wurde die Hilfsfunktion „*minimum*“ geschrieben. Sie sucht den kleinsten Wert aus den vier übergebenen Zahlen heraus.

```

unsigned int minimum (int w1,int w2,int w3,int w4)
{
    int i;

```

```

i=w1;
if (w2<i) i=w2;
if (w3<i) i=w3;
if (w4<i) i=w4;

return(i);
}

```

#### 6.4.2.3 Algorith.c - maximum

Die Funktion ist das Gegenstück zu der Funktion *“minimum”* aus Kapitel 6.4.2.2 und sucht den größten Wert aus den vier übergebenen Zahlen heraus.

```

unsigned int maximum (int w1, int w2,int w3, int w4)
{
    int i;

    i=w1;
    if (w2>i) i=w2;
    if (w3>i) i=w3;
    if (w4>i) i=w4;

    return(i);
}

```

#### 6.4.2.4 Algorith.c - al\_noscale

Die gemessenen Werte werden in das Array *“segment”* gespeichert. Der Algorithmus zur Winkelauswertung selbst erwartet die Meßwerte in dem Array *“scaled”*. Das Skalieren der Meßwerte ist prinzipiell nicht nötig, verringert aber den Winkelmeßfehler. Die Funktion *“al\_noscale”* dient als Ersatzfunktion für die Funktion *“al\_scale”*. Durch sie ist es leicht möglich, die Adaption in der Software auszuschalten.

```

void al_noscale (void)
{
    int i;
    for (i=0;i<16;i++)
        scaled[i]=segment[i];
}

```

#### 6.4.2.5 Algorith.c - al\_scale

Entfernt man den Rotor aus der Anordnung, mißt man im Idealfall bei allen Segmenten den gleichen Wert. In der Praxis sind aber alle Werte unterschiedlich. So können zum Beispiel die einzelnen Segmentkapazitäten variieren, da der Abstand der Sendesegmente zur Empfangsfläche durch die Unebenheiten der Platine nicht konstant ist. Zur Ansteuerung der Segmente werden normale Gatter mit CMOS-Pegel verwendet. Diese garantieren allerdings keinen genauen Ausgangspegel, somit kann die ansteuernde Amplitude für jedes Segment variieren. Ein weiterer Fehler kann dadurch entstehen, daß die einzelnen Segmenttreiber direkt auf dem Empfänger koppeln können. (siehe [WAN99])

Bei all diesen Fehlern handelt es sich um multiplikative und additive Fehler. Notiert man sich den maximalen und den minimalen Meßwert eines Segmentes, so kann man jeden einzelnen späteren Meßwert normieren. Normiert man so alle 16 Segmente, so werden diese segmentabhängigen Einflüsse eliminiert.

Um jedes Segment für sich skalieren zu können, müssen die maximalen und die minimalen Segmentwerte jedes Segmentes erfaßt und gespeichert werden. Sie werden in den statischen Arrays "max" und "min" erfaßt. Wird die Prozedur nach dem Einschalten das erste Mal aufgerufen, werden die Arrays mit Werten versehen, welche in der realen Anwendung nie vorkommen (kleiner als Null, oder größer als 8192 (8-fache Aufsummation des 10-Bit AD-Umsetzers)). Dadurch ist es möglich jederzeit festzustellen, ob schon für jedes Segment ein gültiger Minimum- bzw. Maximumwert vorhanden ist.

```
void al_scale(void)
{
    int i;
    static int max[16];
    static int min[16];

    if (first==0)
    {
        for(i=0;i<16;i++)
        {
            max[i]=-10;
            min[i]=9999;
            first=1;
        }
    }
}
```

Aus der Definition der Korrelationsfunktion folgt, daß die abgedeckten Segmente (siehe Abbildung 5.1) immer auf

$$\text{Minimumsegmentindex} = (grob + n) \bmod 16 \text{ mit } n \in \{15; 0; 7; 8\} \quad (6.2)$$

und die Maximumsegmente immer auf

$$\text{Maximumsegmentindex} = (grob + n) \bmod 16 \text{ mit } n \in \{4; 3; 11; 12\} \quad (6.3)$$

liegen. Es treten immer zwei Segmente mit einem Extremwert nebeneinander auf. Wie schon im Kapitel 5.5.1 erörtert, ist davon nur eines optimal. Die vorgeschlagene Selektion muß auch bei der Adaptionstabelle vorgenommen werden.

```

if (segment[(grob+15)%16]<segment[(grob+ 0)%16])
    { min[(grob+15)%16]=segment[(grob+15)%16];}
else { min[(grob+ 0)%16]=segment[(grob+ 0)%16];}

if (segment[(grob+ 7)%16]<segment[(grob+ 8)%16])
    { min[(grob+ 7)%16]=segment[(grob+ 7)%16];}
else { min[(grob+ 8)%16]=segment[(grob+ 8)%16];}

if (segment[(grob+ 3)%16]>segment[(grob+ 4)%16])
    { max[(grob+ 3)%16]=segment[(grob+ 3)%16];}
else { max[(grob+ 4)%16]=segment[(grob+ 4)%16];}

if (segment[(grob+11)%16]>segment[(grob+12)%16])
    { max[(grob+11)%16]=segment[(grob+11)%16];}
else { max[(grob+12)%16]=segment[(grob+12)%16];}

```

Wenn die Skalierungstabelle das letzte Mal noch nicht gültig war wird kontrolliert, ob sie mittlerweile gültige Werte enthält.

```

if (valid==0)
{
    valid=1;
    for(i=0;i<16;i++)
    {
        if (max[i] ==-10 ) { valid=0;}
        if (min[i]==9999) { valid=0;}
    }
}

```

Ist die Skalierungstabelle gültig, werden die gemessenen Segmentwerte auf das Intervall [1028 8196] skaliert, ansonsten werden die Meßwerte durch Aufruf von "al\_noscale" übernommen.

```

if (valid==1)
{
    for(i=0;i<16;i++)
    {
        scaled[i]=(int)((((long)(7168)*(long)(segment[i]-min[i]))
        \\\
        /
        + (long)(1028)));
        // -----
        ((long)(max[i]-min[i]))
    }
} else {al_noscale();}
}

```

#### 6.4.2.6 Algorithm.c - al\_findefin

Nachdem die grobe Rotorlage bekannt ist, kann die in Kapitel 3.2.3.1 hergeleitete Gleichung 3.13 für die Relativverdrehung und die im Kapitel 3.2.3.2 hergeleitete Gleichung 3.14 für die Absolutverdrehung angewendet werden. Das Signal  $y_{absplus}$  aus Gleichung 3.14 bildet sich aus den Segmenten um die beiden freien linken Rotorkanten 4, 5, 6, 7, 12, 13, 14, 15 und das Signal  $y_{absminus}$  aus den Segmenten um die beiden freien rechten Rotorkanten 0, 1, 2, 3, 8, 9, 10, 11 (vgl. Abb. 6.4).

```
void al_findefin (void)
{
    long yabsplus,yabsminus,min,max,zw,yrelplus,yrelminus;

    //plusalpha=4 5 6 7 12 13 14 15
    yabsplus =(long)scaled[( 4+grob)%16]+
                (long)scaled[( 5+grob)%16]+
                (long)scaled[( 6+grob)%16]+
                (long)scaled[( 7+grob)%16]+
                (long)scaled[(12+grob)%16]+
                (long)scaled[(13+grob)%16]+
                (long)scaled[(14+grob)%16]+
                (long)scaled[(15+grob)%16];

    //minusalpha=0 1 2 3 8 9 10 11
    yabsminus =(long)scaled[( 0+grob)%16]+
                (long)scaled[( 1+grob)%16]+
                (long)scaled[( 2+grob)%16]+
                (long)scaled[( 3+grob)%16]+
                (long)scaled[( 8+grob)%16]+
                (long)scaled[( 9+grob)%16]+
                (long)scaled[(10+grob)%16]+
                (long)scaled[(11+grob)%16];
```

Das Signal  $y_{relplus}$  aus Formel 3.13 wird aus den Segmentwerten 12, 13, 14, 15, 0, 1, 2, 3 und das Signal  $y_{relminus}$  aus den Segmentwerten 4, 5, 6, 7, 8, 9, 10, 11 gebildet.

```
//plusdelta=12 13 14 15 0 1 2 3
yrelplus= (long)scaled[(12+grob)%16]+
            (long)scaled[(13+grob)%16]+
            (long)scaled[(14+grob)%16]+
            (long)scaled[(15+grob)%16]+
            (long)scaled[( 0+grob)%16]+
            (long)scaled[( 1+grob)%16]+
            (long)scaled[( 2+grob)%16]+
            (long)scaled[( 3+grob)%16];
//minusdelta=4 5 6 7 8 9 10 11
```

```

yrelminus= (long)scaled[( 4+grob)%16]+
            (long)scaled[( 5+grob)%16]+
            (long)scaled[( 6+grob)%16]+
            (long)scaled[( 7+grob)%16]+
            (long)scaled[( 8+grob)%16]+
            (long)scaled[( 9+grob)%16]+
            (long)scaled[(10+grob)%16]+
            (long)scaled[(11+grob)%16];

```

Der hier angewandte Algorithmus verwendet nur jeweils das kleinste abgedeckte Segment als Minimumsegment, und das größte freie Segment als Maximumsegment. Dadurch kann der Einfluß der Streufelder stark vermindert werden.

Durch die Multiplikation mit 45000 bzw. 90000 in den ratiometrischen Formeln wird erreicht, daß ein LSB 1/1000 Grad entspricht, denn wie im Kapitel 3.2.3.2 und Kapitel 3.2.3.1 beschrieben, liefert die ratiometrische Gleichung auf 1 normierte Werte. Diese müssen somit abschließend noch mit dem Gültigkeitsbereich der Gleichung multipliziert werden. Wie im Kapitel 3.2.3.1 erwähnt, sind die beiden Rotoren mit einem Offsetwinkel von 15 Grad montiert, um auch das Vorzeichen der Verdrehung bestimmen zu können. Dieser Offsetwinkel muß intern bei der Relativwinkelbestimmung wieder abgezogen werden.

```

//min= (15 0 7 8)*2 bzw
min=(long)minimum(scaled[(15+grob)%16],
                  scaled[( 0+grob)%16],
                  scaled[( 7+grob)%16],
                  scaled[( 8+grob)%16])*8;
//max=(3 4 11 12)*2 bzw
max=(long)maximum(scaled[( 3+grob)%16],
                  scaled[( 4+grob)%16],
                  scaled[(12+grob)%16],
                  scaled[(11+grob)%16])*8;

zw= ((long)(plusalpha-minusalpha))*45000/(max-min);

if (zw+(long)grob*22500>0)
    {alpha=(zw+(long)grob*22500);}
else
    {alpha=zw+(long)grob*22500+360000;};

zw= ((long)(plusdelta-minusdelta))*90000/(max-min)-15000;

deltaalpha=zw;
}

```

## 6.5 Die PWM-Ausgabe

### 6.5.1 PWM.h

Der Sensor gibt ein 1 kHz PWM-Signal mit einem Tastverhältnis zwischen 10 und 90 Prozent aus.

Die dazu benötigten Routinen werden in der Header-Datei bereitgestellt. Die nötigen Prozessorkonfigurationen werden durch die Initialisierungsfunktion *“pwm\_Init”* realisiert. Die Funktion *“pwm\_SetRatioPWM”* setzt dann ein neues Tastverhältnis.

```
#ifndef _pwm
void pwm_Init(void);
void pwm_SetRatioPWM(int value, int maxvalue);
#define _pwm
#endif
```

### 6.5.2 pwm.c

Es wird ein Makro *“set\_pwm\_value”* definiert, welches den übergebenen Wert in einen Integerwert wandelt und in das PWMDUTY Register schreibt.

```
#include "main.h"
#define f_quarz 15000000
#define f_PWM 1000
#define set_pwm_value(value) PWDTY0_1 = (int)(value)
```

#### 6.5.2.1 pwm.c - pwm\_Init

Zuerst wird durch Beschreiben der PWCLK Registers erreicht, daß die beiden 8 Bit PWM-Kanäle 0 und 1 zu einem 16 Bit PWM-Kanal verbunden werden. Der P-Clock Teiler wird dabei auf 1 gesetzt. Der A-Takt ist damit auf 7.5 MHz gesetzt. Durch Beschreiben des PWPOL Registers mit 0 wird der A-Takt als Zählreferenz für das 1. PWM Register gewählt und definiert, daß nach Erreichen der Dutyzeit den PWM-Ausgang von LOW auf HIGH schaltet. Dabei ist noch anzumerken, daß der PWM-Treiber nach [WAN99] invertierend wirkt.

Der kombinierte PWM-Kanal wird auf Port P Pin 0 ausgegeben. Für die Ausgabe muß dann nur noch das Port P Richtungsregister DDRP gesetzt werden.

```
void pwm_Init(void)
{
    PWCLK    = 0x40;
    PWPOL    = 0x00;
    DDRP     |= 0x01;
```





Die Befehle bestehen aus einem 8-Bit breiten Kommando, welches durch eine Zahl ausgedrückt wird. Empfängt der Sensor ein Zeichen über die Schnittstelle, so wird es als Befehl interpretiert. Der Sensor akzeptiert in seiner aktuellen Version 2 Befehle. Der Befehl `C_ping` (0x01h) bewirkt, daß der Sensor eine 8-Bit lange Zahl (0xAAh) über die serielle Schnittstelle sendet. Der Ping Befehl dient als Testmittel für die prinzipielle Funktionsweise der seriellen Schnittstelle.

Durch den Befehl `C_messen` (0x02h) werden die internen Variablen des Sensors übertragen. Die übertragenen Variablen sind der Reihe nach:

- Grobposition als Byte. 1 LSB entspricht 22.5° bzw 12.25°
- Absolutwinkel als Long-Integer. 1 LSB entspricht 1/1000°
- Relativwinkel als Long-Integer. 1 LSB entspricht 1/1000°
- 16 Segmentwerte je als Integer. 1 LSB entspricht 0.61 mV

```
#ifndef _seri
void ser_Init(void);
#define C_ping 1
#define C_messen 2
#define _seri
#endif
```

## 6.6.2 ser.c

Am Anfang der Datei "ser.c" gibt es einige globale Definitionen. Die serielle Kommunikation soll mit einer Geschwindigkeit von 19200 Baud laufen. Der Takt für die serielle Kommunikation wird vom M-Clock (7.5 MHz) hergeleitet. Der Teiler *BR* für die serielle Schnittstelle errechnet sich nach

$$BR = \frac{M_{CLK}}{16 \cdot Baudrate} \quad (6.4)$$

zu 24.

Um die Lesbarkeit des Quelltextes zu erhöhen, wurden noch 2 weitere Makros definiert. Für die serielle Ausgabe steht ein 1 Byte großer Puffer zur Verfügung. Durch Beschreiben dieses Puffers beginnt die serielle Ausgabe unverzüglich. Das Makro "*seriell\_putbyte(Byte)*" dient zum Beschreiben dieses Puffers. Nach dem Beschreiben dieses Registers wird durch ein HIGH des 8. Bit des "*SC0SR1*" Registers angezeigt, daß der Puffer voll ist. Vor der nächsten Ausgabe eines Zeichens muß gewartet werden, bis dieses Bits wieder auf LOW geht und damit anzeigt, daß der Puffer wieder frei ist. Für diese Aufgabe wird ein Makro "*wait\_until\_seriell\_buffer\_empty*" bereitgestellt.

```
#include "main.h"
#define baud19200 24
#define wait_until_seriell_buffer_empty while
    ((int)(SC0SR1&0x80)==0) {}
#define seriell_putbyte(Byte) SC0DRL=((char)(Byte))
```

Bevor die Meßwerte über die serielle Schnittstelle ausgegeben werden, werden sie zwischengespeichert. Dafür sind die folgenden Variablen definiert:

```
int outgrob;  
long outalpha;  
long outdeltaalpha;  
long outseg[16];
```

#### 6.6.2.1 ser.c - ser\_putword

Zur Ausgabe eines Bytes steht bereits ein Makro zur Verfügung. Die 16 Segmentmeßwerte sind jedoch 16-Bit-Variablen. Die Funktion *“ser\_putword”* dient zur Ausgabe von 16-Bit-Variablen. Dabei werden die Variablen im Intelformat, zuerst das Lowbyte, dann das Highbyte gesendet.

```
void ser_putword( unsigned int data )  
{  
    wait_until_seriell_buffer_empty;  
    seriell_putbyte( (char)(data & 0x00ff) );  
    wait_until_seriell_buffer_empty;  
    seriell_putbyte( (char)(data >>8) );  
}
```

#### 6.6.2.2 ser.c - ser\_putlong

Diese Funktion dient zur Ausgabe einer Variablen im Longformat. Die Variable wird dazu in 2 Integer zerlegt, welche über die Funktion *“ser\_putword”* übertragen werden.

```
void ser_putlong( long data )  
{  
    ser_putword((unsigned int)(data & 0x0FFFF));  
    ser_putword((unsigned int)(data >> 16));  
}
```

#### 6.6.2.3 ser.c - command\_ping

Das Ping-Kommando stellt die primitivste Möglichkeit zum Testen der Kommunikation mit dem Sensor dar. Wird das Ping-Kommando geschickt, antwortet der Sensor durch Übermittlung der Zahl AAh.

```
void command_ping(void)  
{  
    wait_until_seriell_buffer_empty;  
    seriell_putbyte(0xAA);  
}
```

#### 6.6.2.4 ser.c - command\_messen

Durch Aufrufen des Meßkommandos wird die ermittelte Grobposition, die beiden berechneten Winkelwerte und die 16 Segmentmeßwerte übertragen. Dabei wird folgende Reihenfolge eingehalten:

- Grobposition als Byte. 1 LSB entspricht  $22.5^\circ$  bzw  $12.25^\circ$
- Absolutwinkel als Long-Integer. 1 LSB entspricht  $1/1000^\circ$
- Relativwinkel als Long-Integer. 1 LSB entspricht  $1/1000^\circ$
- 16 Segmentwerte je als Integer. 1 LSB entspricht 0.61 mV

```
void command_messen(void)
{
    wait_until_seruell_buffer_empty;
    seriell_putbyte(outgrob);
    ser_putlong(outalpha);
    ser_putlong(outdeltaalpha);
    {
        int i;
        for (i=0;i<16;i++)
        {
            ser_putword((unsigned int)(outseg[i]));
        }
    }
}
```

#### 6.6.2.5 ser.c - ser\_Init

Zur Initialisierung der seriellen Schnittstelle muß die Baudrate gesetzt, der Sender und Empfänger eingeschaltet und der Empfanginterrupt aktiviert werden.

```
void ser_Init(void)
{
    SC0BR=baud19200;
    SC0CR2=0x2C;
}
```

#### 6.6.2.6 ser.c - ser\_receivedata

Die Funktion "ser\_receivedata" ist eine Interruptserviceroutine. Sie wird beim Empfang eines seriellen Zeichens ausgeführt.

Zuerst werden die Meßvariablen kopiert. Dadurch wird sichergestellt, daß diese nicht während einer eventuellen Datenübertragung versehentlich geändert werden.

```
void interrupt 20 ser_receivedata(void)
{
    volatile unsigned char command;
    unsigned char sr;

    outgrob=grob;
    outalpha=alpha;
    outdeltaalpha=deltaalpha;
    {
        int i;
        for (i=0;i<16;i++) {outseg[i]=segment[i];};
    }
}
```

Danach wird das serielle Statusregister gelesen. Ist der Empfangspuffer voll, es wurde also ein Zeichen empfangen, wird dieses ausgelesen und anschließend interpretiert.

[illegible]

# Kapitel 7

## Windows-Software zur Meßplatz Automatisierung

Zur Sensorvermessung steht ein Prüfstand zur Verfügung([WAN99]).

Der Prüfstand wird über eine Laborkarte (PC-LabCard PCL 818) angesprochen, die zwei analoge Ausgänge zur Verfügung stellt und über ein Treibermodul einen Schrittmotor im Mikrostepbetrieb ansteuert. So kann jeder beliebige Winkel angefahren werden. Der Schrittmotor ist an einen optischen Referenzwinkelsensor (BALDWIN 694BL) angeschlossen und liefert über einen digitalen Ausgang (18 bit, 16 bit verwendet) die Referenzwinkelwerte an die Laborkarte.

Für diesen Prüfstand ist eine am Institut entwickelte Meßsoftware vorhanden. Die alte Prüfstandsoftware ist für DOS unter Verwendung von Borland Turbo Pascal 7.0 geschrieben. Die Software stellt ein interaktives Menü zur Verfügung, mit dem Startpunkt, Endpunkt und Schrittweite der Meßreihe eingegeben werden. Die Software selbst ist modular aufgebaut, steuert die Laborkarte und verwaltet die Meßdaten. Das Software-Paket stellt auch noch Prozeduren für die Verwaltung der seriellen Schnittstelle zur Verfügung.

Die alte Software weist zwei entscheidende Nachteile auf:

Da die Kommunikation zwischen Sensor und Prüfstand-PC sich laufend ändern kann, muß diese Software für jeden Sensor umgeschrieben bzw. erweitert werden.

Durch die heutige Verwendung von Windows NT als primäre Arbeitsplattform ergibt sich ein weiteres Problem. Die Meßplatzsoftware fährt während der Messung verschiedene Winkel an. Während dieser Zeit muß der Schrittmotor kontinuierlich mit neuen Daten versorgt werden, was unter Windows NT bei DOS-Applikationen nicht möglich ist. Da die Auswertung der Daten aber über Windowssoftware (Matlab) erfolgt, muß der Rechner für jeden Meßvorgang zweimal (WIN->DOS,messen,DOS->WIN,auswerten) rebootet werden.

Um ein effizientes Arbeiten zu ermöglichen, wurde eine neue Prüfstand-Software entwickelt, die unter Windows NT läuft und eine einfache Möglichkeit bietet, die Kommunikation mit dem Sensor zu konfigurieren.

Aufgabe der Software ist die Positionsansteuerung und Meßdatenerfassung, während die Datenauswertung dann über ein eigenes Programm (z.B. Matlab oder Labview) erfolgen sollte. Dazu ist die Software mit einer DDE-Schnittstelle ausgerüstet. Eine Implementierung

der Ankoppelung dieser Prüfstand-Software mittels DDE-Schnittstelle unter Matlab kann in [ZDI99] nachgelesen werden.

Die Handhabung dieser Software wird im folgenden Kapitel beschrieben. Abschließend erfolgt noch eine Spezifikation der DDE-Schnittstelle.

Die neue Software wurde unter Borland Delphi 2 entwickelt, da mit dieser Programmiersprache sehr leicht und schnell Standard-Windows-Applikationen mit entsprechender Benutzerfreundlichkeit implementiert werden können. Auf eine genaue Dokumentation des Quellcodes dieser Software wird hier verzichtet, da der Delphi Code von sich aus leicht und verständlich lesbar ist.

## 7.1 Bedienungsanleitung für die Meßplatz Software

Nach dem Starten von "Project1.exe" erscheint ein Windows Fenster nach Abbildung 7.1.

### 7.1.1 Meßplatzsteuerung

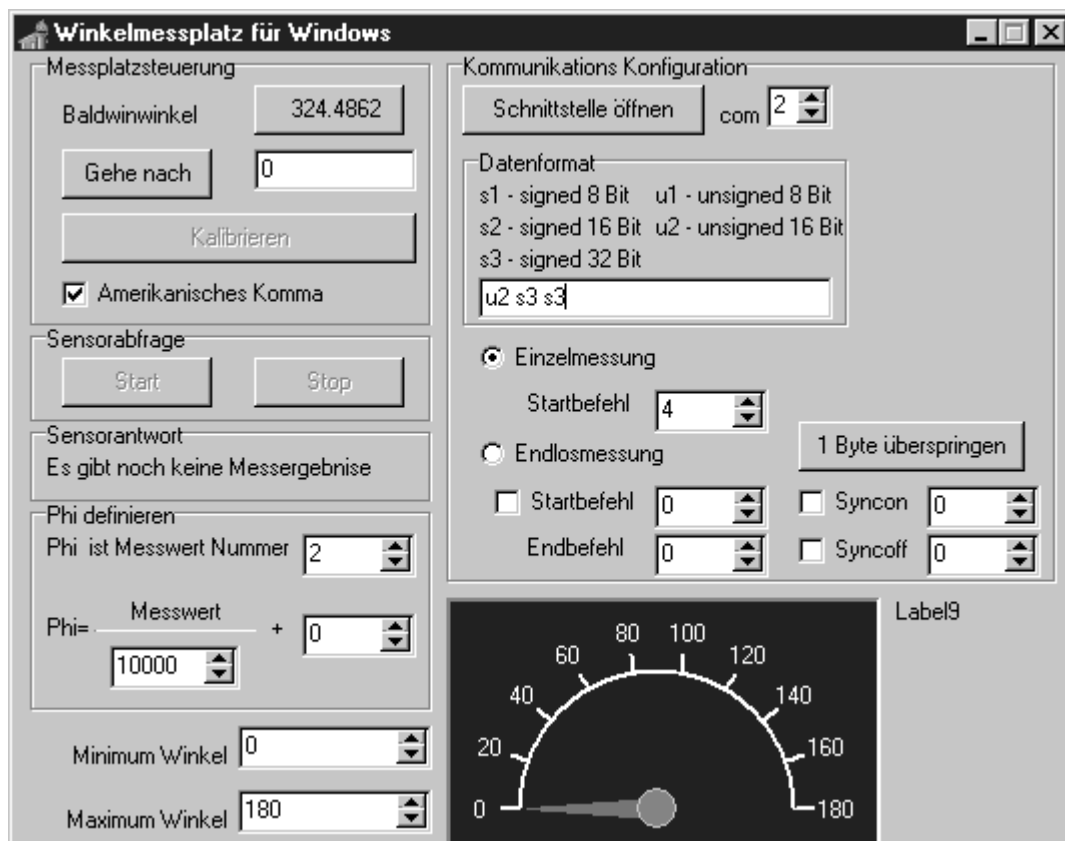


Abbildung 7.1: Das Hauptfenster der Prüfstandssoftware

Die Steuerung des Prüfstandes erfolgt über die Gruppe Meßplatzsteuerung, welche sich links oben im Fenster befindet.

Durch Betätigen des Knopfes, der den Zahlenwert des Baldwinwinkels enthält, wird die Position der Welle durch den Referenzwinkelsensor (BALDWIN 694BL) festgestellt und der eingelesene Wert wird auf den Knopf eingetragen.

Unter diesem Knopf befindet sich ein Eingabefeld. In das kann der Zielwinkel der Welle eingetragen werden. Durch Anklicken des Knopfes "Gehe nach" wird diese Zielposition angefahren. Die Prüfstand Software hat eine Multiturnfunktion, es sind daher Winkelwerte größer als  $360^\circ$  und kleiner  $0^\circ$  erlaubt. Die Anzahl der Umdrehungen wird von der Software verwaltet. Da die Software regelmäßig die Wellenposition abfragt, werden auch Umdrehungen erfaßt, die von Hand ausgeführt werden.

Durch Drücken des Knopfes "Kalibrieren" werden die Winkelwerte des Referenzsensors und des Testsensors aufgenommen. Die Differenz der beiden Werte wird dann zur Korrektur des Referenzwertes genommen. Beide Winkel sollten ab jetzt synchron sein. Um die Funktion Kalibrieren anwenden zu können, muß zuerst das Sensorprotokoll (mehr dazu im Kapitel 7.1.2) eingestellt werden.

Bei Verwendung der DDE-Schnittstelle wird die Referenzposition als String übergeben. Hierbei tritt das Problem auf, daß manche Programme diese Strings über Windowsfunktionen andere über eigene, meist ANSI-C standard library Funktionen, wandeln lassen. Solange man im amerikanischen Sprachraum ist, treten dabei keine Probleme auf. Im deutschen Sprachraum dient jedoch ein Komma, im Gegensatz zum Punkt beim amerikanischen Sprachraum, zur Abtrennung der Nachkommastellen. So erwartet Matlab einen Punkt, Labview aber (solange das System auf deutschsprachig installiert ist) ein Komma, um die Nachkommastellen abzutrennen. Um mit jedem Programm arbeiten zu können, kann definiert werden, ob ein Komma oder ein Punkt zum Trennen gewählt werden soll.

Beim Empfang von Kommazahlen ist die Art der Trennung immer frei wählbar, da die Software "Project1" beide akzeptiert.

## 7.1.2 Kommunikations Konfiguration

Das Meßplatzfenster ermöglicht es, die Kommunikation mit dem Sensor über die serielle Schnittstelle zu konfigurieren. Nach Auswahl der Schnittstelle kann durch Drücken der Taste "Schnittstelle öffnen" die Kommunikation mit dem Port geöffnet werden. Die Taste ändert dabei ihre Funktion und den Text auf "Schnittstelle schließen". Nach dem Drücken der Taste "Schnittstelle öffnen" erscheint ein Windowsstandarddialogfenster, in welchem die Parameter der seriellen Übertragung (Baudrate usw.) eingestellt werden. Die Form dieses Fensters kann von Windowsversion zu Windowsversion variieren, und wird deswegen nicht genauer beschrieben.

Damit die Meßplatzsoftware mit einer Vielzahl von Sensoren arbeiten kann, werden zwei einfache Kommunikationsprotokolle unterstützt.

Protokoll 1, Einzelmessung, geht davon aus, daß jede Messung einzeln initiiert werden muß. Protokoll 2, Endlosmessung, geht davon aus, daß nach Aktivierung des Sensors dieser periodisch die Meßdaten über die serielle Schnittstelle schickt, bis der Stopbefehl kommt.

Unabhängig von der Wahl des Protokolles kann ein Datensatz aus einer beliebigen Anzahl von 8, 16 oder 32 Bit Binärzahlen bestehen. Unter der Rubrik Datenformat kann die Reihenfolge der übertragenen Werte definiert werden. In Abbildung 7.1 wird zum Beispiel eine Abfolge von einer vorzeichenlosen 16 Bitzahl, und von zwei folgenden 32 Bitzahlen definiert.

Sollte es dennoch nicht möglich sein, eines der beiden Protokolle zu unterstützen, kann die Meßplatzsoftware dennoch zur Steuerung des Schrittmotors und zum Auslesen des Referenzwinkelsensors verwendet werden.

#### 7.1.2.1 Protokoll 1 - Einzelmessung

Vor jeder Übertragung erwartet der Sensor einen 8 Bit langen Befehl. Dieser Modus wurde in Kapitel 6.6 gewählt, wobei der Sender bei Erhalt des Befehls 2 beginnt die

- Grobeposition als Byte (8 Bit),
- Winkel als Long (32 Bit),
- Differenzwinkel als Long (32 Bit) und die
- 16 Segmentwerte je als Int (je 16 Bit)

zu senden. Die für diese Übertragung notwendigen Konfigurationen sind:

Datenformat:

u1 s3 s3 u2 u2 u2 u2 u2 u2 u2 u2 u2 u2 u2 u2 u2 u2

Startbefehl: 2

Durch Drücken der Start Taste wird der Startbefehl gesendet und die Sensorantwort aufgezeichnet. Die Sensorantwort wird in die Rubrik "Sensorantwort" eingetragen.

#### 7.1.2.2 Protokoll 2 - Endlosmessung

In diesem Fall wird davon ausgegangen, daß der Sensor von sich aus periodisch die Meßdaten in dem unter "Datenformat " definierten Format überträgt. Die Taste "1 Byte überspringen" ermöglicht ein manuelles synchronisieren.

Zusätzlich wird als Option ein 1-Byte langer Start und Stop Befehl empfohlen. Diese Befehle werden durch Drücken der Tasten "Start" und "Stop" übermittelt, und können von der Software verarbeitet werden. Eine weitere Option wird mit den "Syncon"- und "Syncoff"-Byte geboten. Diese sollten vom Sensor vor und hinter jedem Datensatz angehängt werden. Dadurch kann sich die Meßplatzsoftware selbst stabilisieren.



### 7.1.3 Phi definieren

Hier kann definiert werden, welcher der übertragenen Werte der Winkel ist, und seine Umrechnung in Grad. Nach definieren dieser Umrechnung kann durch Drücken des "Kalibrieren"-Knopfes das Offset zwischen Referenzsensor und Prüfsensor eliminiert werden. Zusätzlich wird der zuletzt eingelesene Winkelmeßwert in der Anzeige rechts unten gezeigt. Die Anzeige stellt den Winkel in Form einer analogen Zeigerposition in dem, mittels der "Minimum Winkel"- und "Maximum Winkel"-Felder, eingestellten Bereich dar.

## 7.2 Die DDE Schnittstelle

DDE ist eine für Microsoft Windows <sup>1</sup> (NT, 95, 98) definierte Schnittstelle zum Austausch von Daten bzw. zur Steuerung von Anwendungen über Makros. DDE baut auf eine Server/Client-Struktur auf, bei der der Server dem Client Daten zur Verfügung stellt. Je nach Anwendung kann diese als Server, Client oder beides dienen. Für DDE sind drei wesentliche Komponenten notwendig:

1. Anwendung - Windows Programm mit dem kommuniziert wird (DDE-Server), beispielsweise Microsoft EXCEL. In dem hier betrachteten Fall ist die Anwendung das Programm "project1".
2. Thema - Das Thema ist der Gegenstand der DDE-Kommunikation, beispielsweise ein Microsoft EXCEL-File "tabelle1.xls". Beim Programm "project1" ist das Thema ein Objekt mit Namen "Messplatz".
3. Elemente - Das Thema enthält Elemente mit den zur Verfügung stehenden Daten, beispielsweise eine Zelle in einer Microsoft EXCEL-Tabelle. Im Objekt "Messplatz" sind diese Elemente der Referenzwinkelsensor, der Schrittmotor und der zu messende Sensor.

Der DDE-Client kann die Elemente (Items) des DDE-Servers auslesen, beziehungsweise beschreiben, wobei jedes Item eine Zeichenkette (String) beinhalten kann. Der DDE-Client kann auch Makrobefehle an den DDE-Server schicken. Dies geschieht durch Übermittlung einer Zeichenkette (Befehlsstring).

## 7.3 Spezifikation der DDE-Schnittstelle

Die Software ist mit einer DDE-Schnittstelle ausgestattet. Dadurch kann der Prüfstand mit einer Vielzahl von Anwendungsprogrammen bedient werden. In diesem Kapitel werden die zur Verfügung gestellten Variablen und Befehle erklärt. In [ZDI99] ist eine Ausführung der DDE Prüfstandsbedienung ausgeführt. Informationen zur DDE Anbindung unter Borland Delphi können in [DOB97] nachgelesen werden.

---

<sup>1</sup>Windows NT, Windows 95, Windows 98, EXCEL sind Produkte der Microsoft Corporation

Die Prüfstandssoftware trägt den Namen "Project1". Sie stellt neben ihrer interaktiven Benutzeroberfläche einen DDE-Server dar. Der DDE-Server hat ein Thema namens "Messplatz". Das Thema "Messplatz" stellt 3 Variablen (Items) und 3 Befehle (Makros) zur Verfügung.

Die zur Zeit implementierten DDE-Items (bei ihnen muß unbedingt auf die Groß/Kleinschreibung geachtet werden) sind:

- **GetSensor** - Hier ist die letzte gelesene Sensorantwort enthalten. Die Werte sind in Text konvertiert und sind mit einem Leerzeichen voneinander getrennt.
- **GetBaldwin** - Hier ist die letzte gemessene Position des Referenzsensors in Grad als Text gespeichert.
- **SetAngle** - In diese Variable wird der Winkelssollwert der Welle eingetragen. Das Beschreiben bewirkt ein unverzügliches Anfahren der Sollposition. Es sind sowohl negative Werte als auch Werte größer 360° erlaubt.

Die zur Zeit implementierten DDE-Makros (hier ist die Groß/Kleinschreibung nicht von Bedeutung):

- **GetSensor** - Das Aufrufen des Befehls bewirkt das erneute Auslesen des Sensorwertes und das Aktualisieren der gleichnamigen Variablen.
- **GetBaldwin** - Das Aufrufen des Befehls bewirkt das erneute Auslesen des Referenzsensors und das Aktualisieren der gleichnamigen Variablen.
- **SetAngle** - Bewirkt ein neuerliches Anfahren der letzten, in "SetAngle" gespeicherten, Wellenposition.

# Kapitel 8

## Spezifikation des Prototypes

Von den im Rahmen des Projektes getesteten Strukturen (vgl. dazu [ZDI99] ) konnte die inverse Vierflügelstruktur als die mit den besten Ergebnissen erkannt werden.

Unter Verwendung der genannten Rotorstruktur ergibt sich mit der in [WAN99] beschriebenen Hardware unter Verwendung der in 6 dokumentierten Software die folgende Spezifikation des Prototypen:

### Elektrodenstruktur lt. Anhang:

- Sendeelektrode: 32 Segmente, Segmentteilung  $11.25^\circ$
- Rotorstruktur: Inverse Vierflügelstruktur (symmetrische Rotoren mit zwei je  $45^\circ$  breiten Flügeln)

### Platzbedarf:

- Elektrodenstruktur mit Distanzring - Zylinder: Durchmesser 110 mm, Distanz zwischen Sender- und Empfangselektrode  $d = 5mm$
- Senderansteuerung über Steckverbindung direkt an der Rückseite der Sendeelektrode, Höhe ca. 20 mm
- Empfängerschaltung und HC12 Board auf Leiterplatte Durchmesser: 110 mm, über 300 mm langes geschirmtes Kabel mit Empfängerfläche verbunden, Positionierung beliebig.

### Relativwinkelmessung:

- Meßbereich:  $\pm 5^\circ$
- Meßunsicherheit:  $\pm 0.25^\circ$
- Auflösung:  $0.001^\circ$

**Absolutwinkelmessung:**

- Meßbereich:  $180^\circ$
- Meßunsicherheit:  $\pm 0.12^\circ$
- Auflösung:  $0.001^\circ$

**Kennfeld des Sensors:** In Abbildung 8.1 ist der Relativwinkelmeßabweichung in Abhängigkeit vom Referenz- und Referenzrelativwinkel dargestellt. In Abbildung 8.2 und Abbildung 8.3 sind die mittlere Meßabweichung des Relativwinkels in Abhängigkeit vom Referenzabsolutwinkel bzw. vom Referenzrelativwinkel dargestellt. In Abbildung 8.4 ist der Winkelfehler in Abhängigkeit vom Referenz- und Referenzrelativwinkel dargestellt. In Abbildung 8.5 und Abbildung 8.6 sind die mittlere Meßabweichung des Absolutwinkels in Abhängigkeit vom Referenzabsolutwinkel bzw. vom Referenzrelativwinkel dargestellt.

**Hardwareausführung:** Stromlaufpläne, Fertigungsunterlagen und Konstruktionszeichnungen von der Senderansteuerung, der Empfängerschaltung, dem HC12-Board sowie der Sende-, Empfangs- und Rotorelektrode für diesen Momentensensor befinden sich in [WAN99].

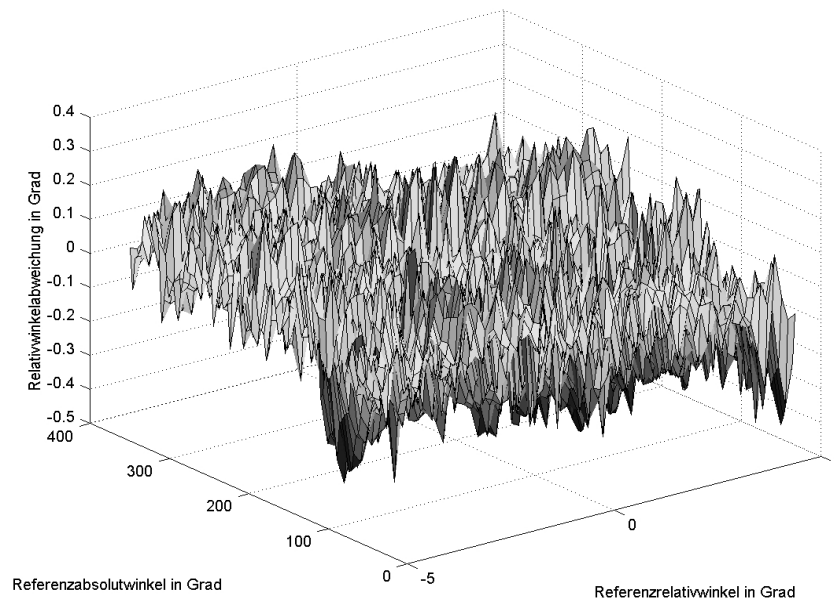


Abbildung 8.1: *Gemessene Relativwinkelabweichung über Referenzrelativwinkel und Referenzabsolutwinkel - Inverse Vierflügelstruktur*

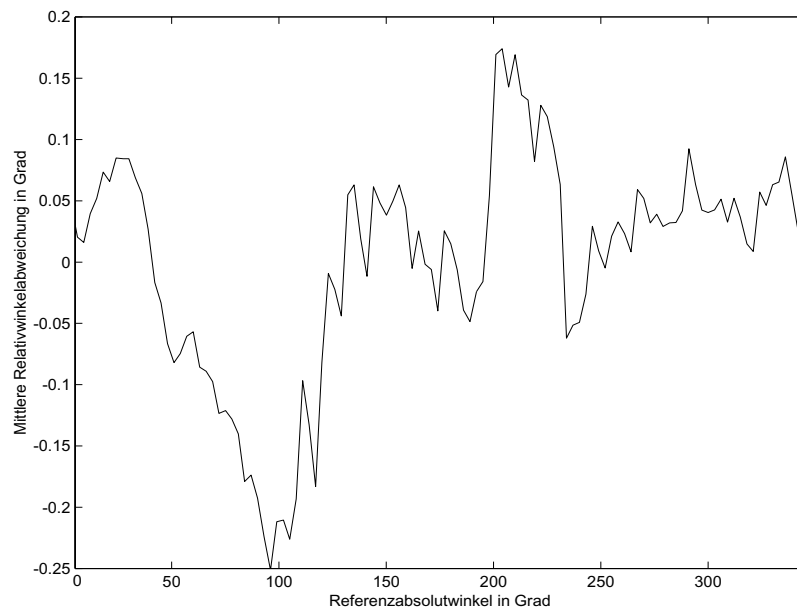


Abbildung 8.2: *Mittlere Relativwinkelabweichung über Referenzabsolutwinkel - Inverse Vierflügelstruktur. Die Mittlere Abweichung geht ab einem Absolutwinkel von  $180^\circ$  von  $\pm 0.2^\circ$  auf  $\pm 0.08^\circ$  zurück, weil dann alle Segmentwerte bereits einmal kalibriert werden konnten.*

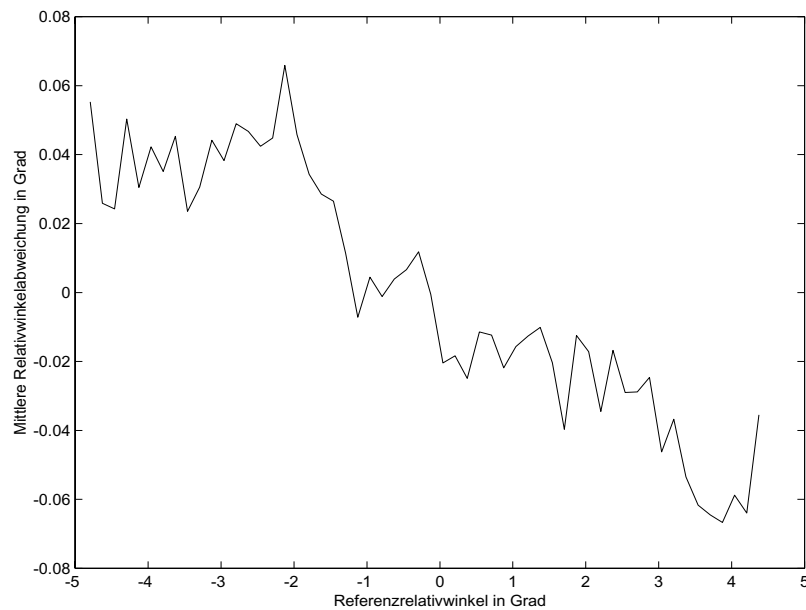


Abbildung 8.3: *Mittlere Relativwinkelabweichung über Referenzrelativwinkel - Inverse Vierflügelstruktur*

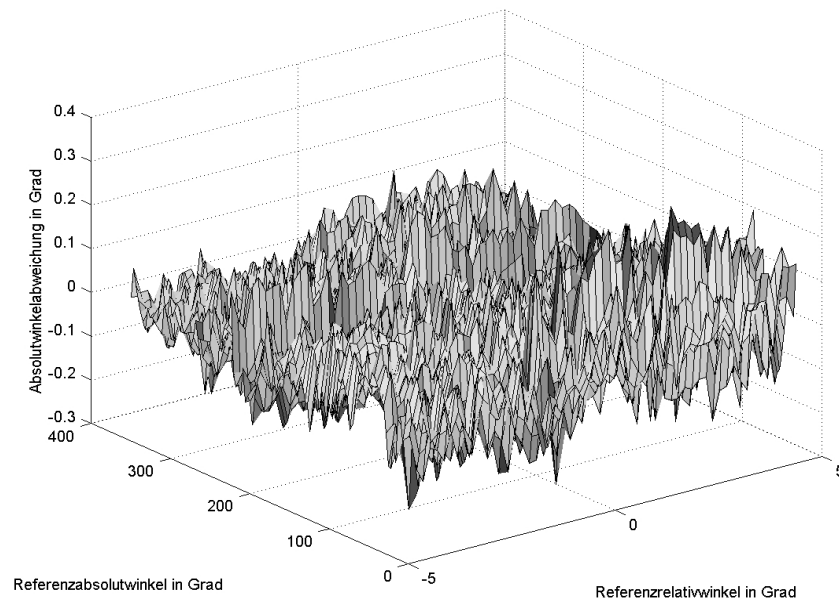


Abbildung 8.4: Gemessene Absolutwinkelabweichung über Referenzrelativwinkel und Referenzabsolutwinkel - Inverse Vierflügelstruktur

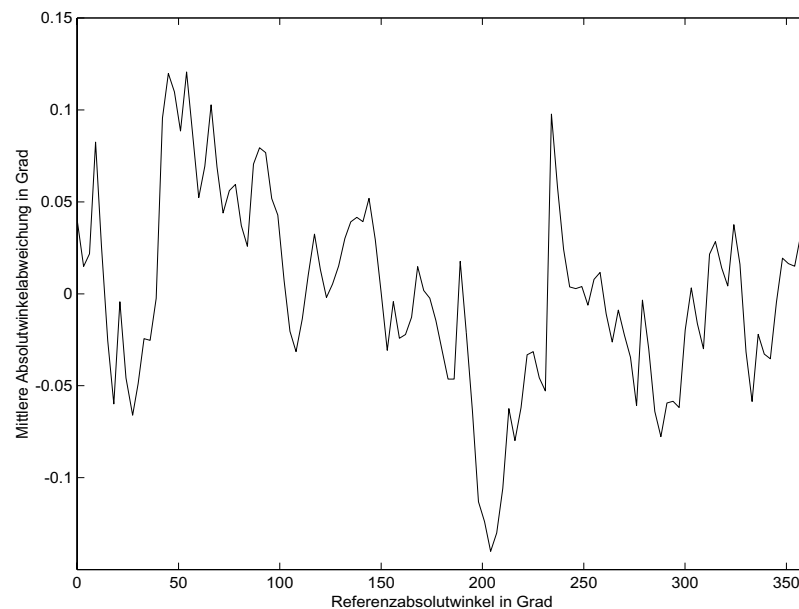


Abbildung 8.5: Mittlere Absolutwinkelabweichung über Referenzabsolutwinkel - Inverse Vierflügelstruktur. Die Mittlere Abweichung geht ab einem Absolutwinkel von  $180^\circ$  von  $\pm 0.12^\circ$  auf  $\pm 0.08^\circ$  zurück, weil dann alle Segmentwerte bereits einmal kalibriert werden konnten.

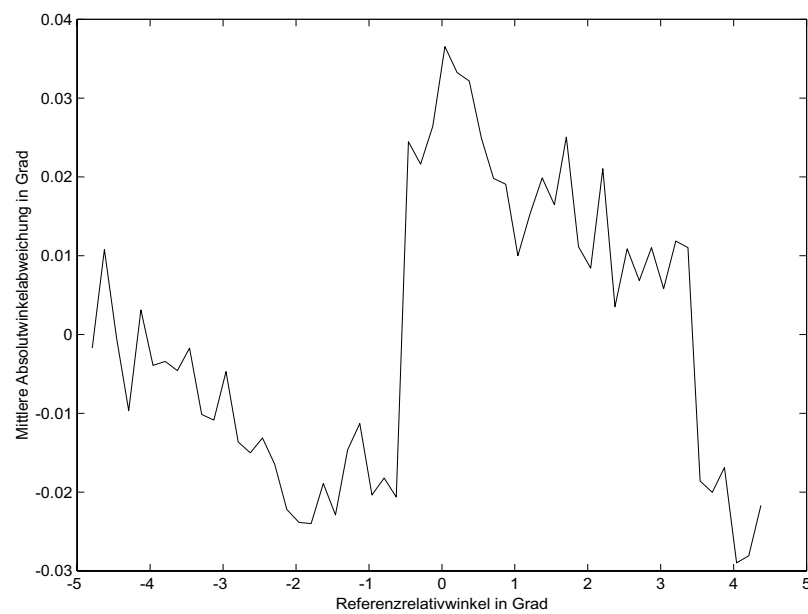


Abbildung 8.6: *Mittlere Absolutwinkelabweichung über Referenzrelativwinkel - Inverse Vierflügelstruktur*

# Kapitel 9

## Ausblick

Der Prototyp des kapazitiven Momentensensors ist in Hinblick auf eine spätere Einsetzbarkeit für die Lenkkraftmessung im Automobil zu verbessern, wofür in erster Linie eine Verbesserung der Meßgenauigkeit nötig ist.

Um den Sensor direkt in die Lenksäulenmechanik integrieren zu können, ist eine wesentliche Verkleinerung der gesamten Elektrodenanordnung notwendig. Hier könnte eine Strukturvereinfachung hilfreich sein. Eine Möglichkeit dazu wäre die Reduzierung von 32 auf 24 Segmente, was zu einem einfacheren Aufbau, d. h. größere Segmentteilung bei weniger Leitungen von der Senderansteuerung führt. Die Anzahl der zu messenden Segmentwerte würde von 16 auf 12 reduziert. Eine größere Segmentteilung bedeutet eine gröbere Auflösung der Segmentbereiche, weil die Bitanzahl des A/D-Umsetzers ja konstant bleibt, dafür würde die Streufeldproblematik geringer.

Die durch das Downscaling entstehenden Probleme mit den Streufelder sind noch zu untersuchen. Bei der Reduzierung des Außendurchmessers muß folgendes beachtet werden: Der innere Radius des Empfangsrings darf aufgrund des stärkeren Einflusses der Streufelder bei kleinen Radien nicht wesentlich reduziert werden. Der Außendurchmesser darf nur soweit verringert werden, daß genügend kapazitive Kopplung zwischen Sendersegment und Empfangsring bestehen bleibt, um einen ausreichenden Signalhub zu gewährleisten. Durch die Verwendung einer 24 Segmente-Struktur ist es möglich, auch den inneren Radius weiter zu verkleinern.

Neben Veränderung der Geometrieverhältnisse bei den bisher verwendeten Anordnungen, wäre es auch möglich die Elektrodenstruktur nicht in Scheibenform sondern als konzentrische Hohlzylinder zu realisieren. In dieser Struktur sind auf der Innenseite des Zylindermantel des äußersten Hohlzylinders die Sendesegmente aufgebracht. Die Rotoren sind die beiden inneren Hohlzylinder, aus deren Mantel die Rotorstruktur ausgeschnitten ist. Der innerste Zylinder (oder die Welle selbst) wirkt als Empfangselektrode.

Beim Bau des Prototypen wurde auf das EMV-Verhalten, d. h. Abstrahlverhalten und Empfindlichkeit gegen elektromagnetische Störungen von außen, noch nicht näher eingegangen. Auch die mechanische Robustheit des Sensors, sein Temperaturverhalten und der Einfluß von Verschmutzung und Feuchtigkeit wurden noch nicht verifiziert. Für die Weiterentwicklung in Richtung industrielle Einsetzbarkeit sind diese Einflüsse unbedingt zu beachten.



# Literaturverzeichnis

- [BRA92] Georg Brasseur, Thomas Eberharter : *Kapazitiver Drehwinkelsensor*, Österreichisches Patent AT 398245; 1994 und Europ. Pat. EP 0551 066, 1992 und US Pat. 5,598,153.
- [BRA93] Georg Brasseur, Thomas Eberharter : *Capacitive Angular Displacement Transducer*, US Patent Appl. S/N 08/087,261;1993
- [BRA96] Georg Brasseur : *Analysis of a Novel Noncontact Capacitive Angular Displacement Transducer*, eingereicht für IEEE Transactions on Instrumentation and Measurement.
- [BRA97] Georg Brasseur : *Kapazitiver Drehwinkel- und Winkelgeschwindigkeitssensor und Meßeinrichtung für einen solchen*, Österreichische Patentanmeldung A505-97
- [FAB97] Tibor Fabian, Georg Brasseur : *A Robust Capacitive Angular Speed Sensor*, in Proceedings of IEEE Conference on Instrumentation and Measurement Technology (IMTC/97), Ottawa, Canada, May 19-21, pp. 1267-1272, 1997.
- [BRA00] St. Cermak, F. Wandling, W. Zdiarsky , P. Fulmek, G. Brasseur : *Capacitive Sensor for Relative Angle Measurement*, in Proceeding of IEEE Conference on Instrumentation and Measurement Technology (IMTC2000), Baltimore, USA, May 1-4, pp 830-834, 2000.
- [CER00] St. Cermak, F. Wandling, W. Zdiarsky , P. Fulmek, G. Brasseur : *Capacitive Sensor for Torque Measurement*, in Proceeding of International Measurement Confederation (IMEKO2000), Wien, Österreich, September 25-28, TC 3.
- [FAB98] Tibor Fabian : *Vergleich und Optimierung der Ansteuerverfahren eines kapazitiven Winkel- und Winkelgeschwindigkeitssensors*, Diplomarbeit, TU-Wien, 1998
- [KOEL97] Wolfgang Köllner : *Signalprozessorgestützte Auswertung von Meßsignalen eines automobiltauglichen Drehwinkelsensors*, Diplomarbeit, TU-Wien, 1997
- [WAN99] Florian Wandling : *Hardwareentwicklung für einen kapazitiven Drehmomentensensor*, Diplomarbeit, TU-Wien, 1999
- [ZDI99] Wolfgang Zdiarsky : *Vergleich verschiedener Rotorstrukturen eines kapazitiven Drehmomentensensors*, Diplomarbeit, TU-Wien, 1999

- [JNG94] Gerben de Jong : *Smart Capacitive Sensors, physical, geometrical and electronic aspects*, Dissertationsschrift, Delft University Press, 1994
- [PRE94] Adalbert Prechtel : *Grundlagen der Elektrotechnik*, Buch zur Vorlesung, 1994
- [FAS94] G. Fasching: *Werkstoffe für die Elektrotechnik*, 3. Auflage, Springer-Verlag, 1994
- [SRUF92] E. Schröder : *Elektrische Meßtechnik*, 5. Auflage, Carl Hanser Verlag, 1995
- [BOEG85] Alfred Böge: *Mechanik und Festigkeitslehre*, Vieweg Verlag, 1985
- [DOB97] W. Doberenz, T. Kowalski *Borland Delphi 3 für Profis*, Hanser-Verlag, 1997
- [MOT98] Motorola: *Semiconductor Technical Data*,  
*Data Sheet Nr.: MC 68HC912D60TS/D*, March 1998
- [BAR90] Hans-Jochen Bartsch : *Taschenbuch mathematischer Formeln*, 1990

# Anhang A

## Capacitive Sensor for Relative Angle Measurement

Dieses Paper wurde auf der

17<sup>th</sup> IEEE  
Instrumentation and Measurement Technology Conference  
in Baltimore (Maryland, USA)

präsentiert. Der Vortrag fand am Mittwoch, dem 3. Mai 2000 statt und wurde von Hrn. Univ.-Prof. Dipl.-Ing. Dr. Georg Brasseur gehalten.

# Capacitive Sensor for Relative Angle Measurement

St. Cermak, F. Wandling, W. Zdiarsky, P. Fulmek  
Institut für industrielle Elektronik und Materialwissenschaften  
Technische Universität Wien, A-1040 Vienna, Austria, Europe

G. Brasseur  
Institut für Meßtechnik und Meßsignalverarbeitung  
Technische Universität Graz, A-8010 Graz, Austria, Europe

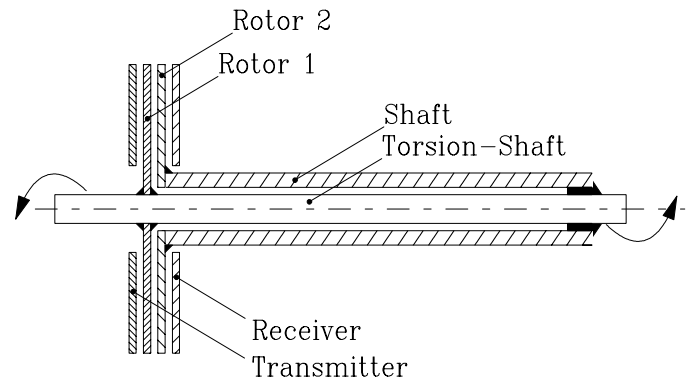
## Abstract

*Based on a capacitive angle/angular speed sensor a sensor measuring the relative angle between two rotating shafts has been developed. Two rotatable grounded electrodes are placed between two sensor plates. The relative angle between the two rotors and the absolute position of the rotor blades are calculated from measurements of the capacitive coupling between different transmitting stator segments. A prototype of this sensor has been developed with a range of the relative angle of  $\pm 7.5^\circ$  with a resolution of  $0.1^\circ$ .*

**Keywords:** capacitive sensor, relative angle sensor, torque measurement

## Introduction

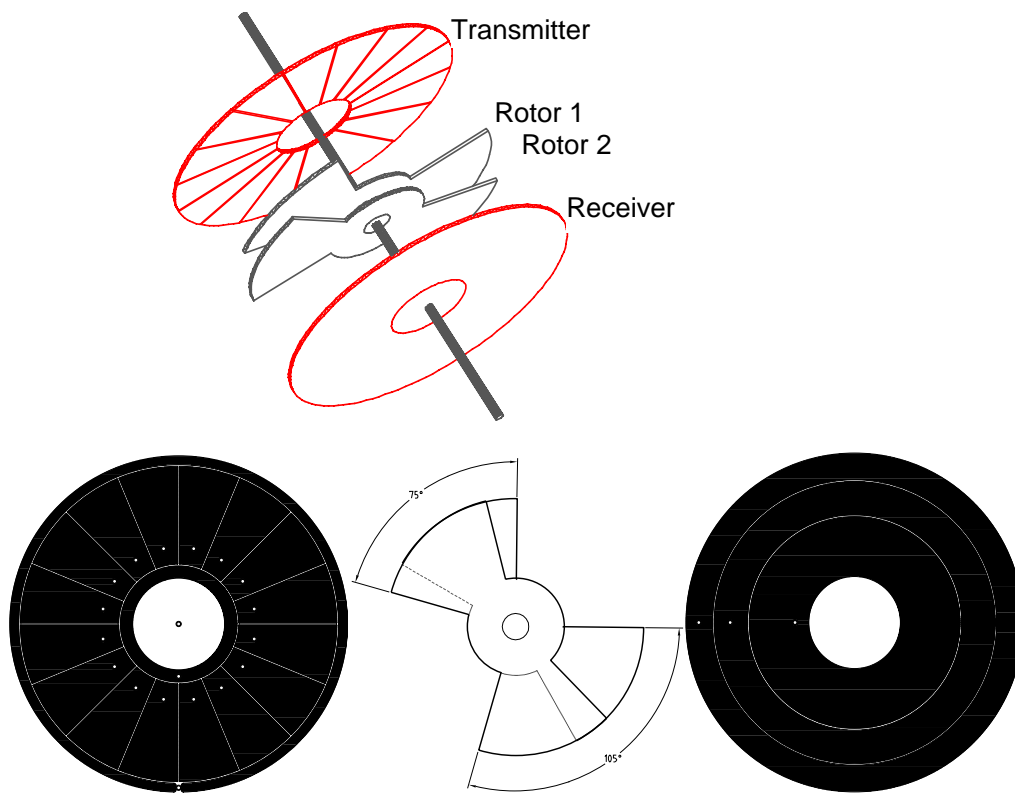
The capacitive angle/angular speed sensor described in [1], [2], [3] and [4] has been modified by mounting two asymmetric rotors on two concentric shafts between the sensor stators. Both grounded rotors realise a single effective rotor with a variable geometry, depending on the relative angle between the rotating shafts. The absolute angle and angular velocity of both rotors are measured as described in [1] and [3] with a modified geometry and algorithm. These modifications additionally allow to measure the relative angle between the two rotors within a range of  $\pm 10^\circ$ . Applying the relative rotation angle of a torsion shaft to the two rotors using two concentric shafts as shown in Figure 1, allows to measure the torque transmitted by the rotating shaft using Hooke's Law.



**Figure 1 – Mechanical construction for applying the torsion to the rotors of the relative angle sensor.**

## Working principle

Figure 2 shows the electrode structure of the capacitive sensor. One stator plate is used as transmitter with 16 transmitting segments with center angles of  $22.5^\circ$ , the other stator contains the receiving ring electrode. The shaft electrically connects the conductive rotors to ground potential. These two rotors with asymmetrically arranged blades and a center angle of  $60^\circ$  (Figure 2) are mounted mirror-symmetrically on two concentric shafts. If one rotor is placed above the other, as shown in Figure 2, a structure, very similar to the rotor structure for the angular speed sensor discussed in [3], is formed. The rotor for the absolute angle or angular speed sensor consists of two rotor blades with an open angle of  $90^\circ$ . The relative angle rotor-pair forms two blades whose open angles can vary between  $60^\circ$  and  $120^\circ$  depending on the relative angle between the two rotors, while the sum of both blade angles always stays at  $180^\circ$ . In order to measure the direction of the torsion or the sign of the torque, it is



**Figure 2 – Sensor topology of a capacitive relative angle sensor**  
**Transmitter plate with 16 segments, two mirror-symmetrical rotors, receiving electrode.**

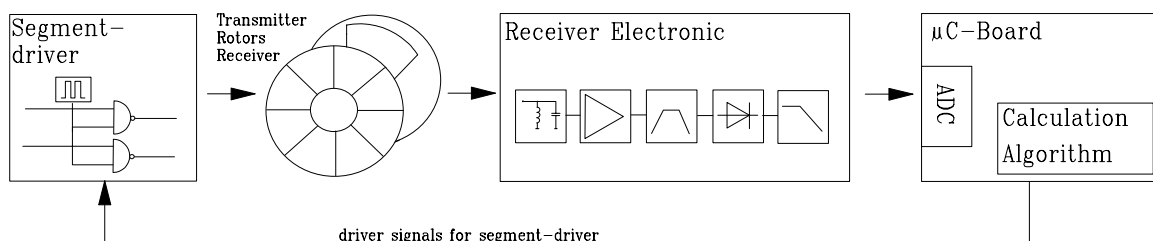
necessary to distinguish between the two rotor blades. So the open angle of one rotor blade is allowed to be between  $60^\circ$  and  $90^\circ$ , the other between  $90^\circ$  and  $120^\circ$ . In order to allow relative movements in both directions the zero position of the relative angle is defined for blades with center angles of  $75^\circ$  and  $105^\circ$ , respectively. As long as the absolute value of the relative angle is below  $15^\circ$ , the two blades can be distinguished by verifying their size.

Depending on the relative angle the electrically effective size of the rotor blades is changed. These changes influence the capacitive coupling between transmitting segments and the receiving electrode. In one

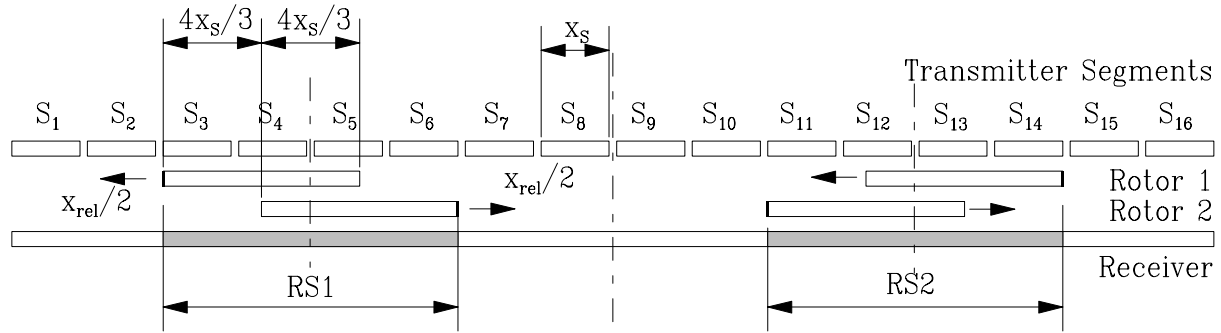
measurement cycle a pulse sequence is applied to each segment. Depending on the rotor position and the effective size of the rotor the received signals change for each segment. By applying a ratiometric algorithm to the received signals, the signed relative angle between the rotors and the absolute angle of both rotors within  $360^\circ$  are calculated.

### Electrical circuit and signal conditioning

As figure 3 shows, the micro controller uses a segment driver to switch the carrier signal between the transmitter



**Figure 3 – Blockdiagram of the relative angle sensor electronic.**



**Figure 4 – Schematic geometry of the sensor.**

segments. The carrier frequency now is coupled capacitively from the transmitter to the receiver. The receiver is connected to the receiver electronic, whose first stage is a resonant circuit. The resonant circuit is tuned to the carrier frequency and acts as a low noise, narrow band amplifier. The following stages consist of an amplifier, a very sharp bandfilter, an electric rectifier unit and a low pass filter. This signal is fed into an ADC input of the controller. The micro controller samples 16 segment values (SV), each corresponding to an excited segment. Further details to the signal conditioning circuits are given in [5].

### Algorithm for the relative angle measurement

Figure 4 shows the schematic geometry of the sensor topology for a symmetric configuration of the rotor. The grounded rotor blades absorb a part of the electric field, they produce an electrical shadow image on the receiver electrode. If the rotor blades move relatively, as shown in Figure 4, the rotor shadow RS1 will be enlarged, and the rotor shadow RS2 reduced. This also means that the sum of SV1 to SV8 decreases and the sum of SV9 to SV16 increases proportional to the relative angle. The segment value of the completely shadowed and completely free segments are not influenced by the rotor displacement, which is a necessary condition to apply the ratiometric algorithm.

The relative angle between the rotors is calculated by evaluating the expression

$$\begin{aligned}
 SV_{\max} &= 2 \cdot (SV_1 + SV_8 + SV_9 + SV_{16}) \\
 SV_{\min} &= 2 \cdot (SV_4 + SV_5 + SV_{12} + SV_{13}) \\
 \text{relative angle} &= \frac{-\sum_{i=1}^8 (SV_i) + \sum_{l=9}^{16} (SV_l)}{SV_{\max} - SV_{\min}} \cdot 90^\circ
 \end{aligned}$$

In practice the reference value, i.e. the difference-term in the denominator expression is not constant and a calibration is necessary. It takes one turn to initialise the calibration table, and reach the final accuracy.

To determine the direction of the relative movement, an asymmetry in the rotor geometry is introduced, as shown in Fig. 2. This asymmetry doesn't change the evaluation principle, because it's just an offset to the relative movement. To calculate the relative angle from the expression shown above it's necessary to know the absolute position of the resulting rotor. For this reason a second algorithm is used, which evaluates the rough rotor position with a resolution of one segment width (22.5° for a 16 segment transmitter). This rough positioning algorithm calculates the convolution of the measured segment values with a set of values defined for the zero-position. The maximum of the result gives the offset of the actual rotor position to the defined zero-position.

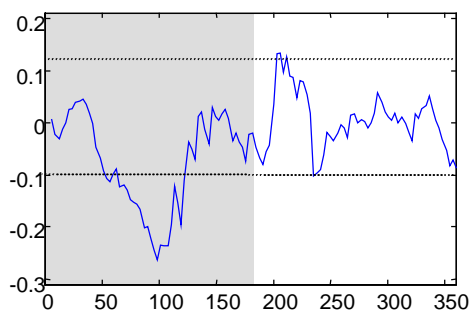
### Experimental results

Our test bench consists of an optical reference sensor with a resolution of 20''. This sensor and a stepper motor are connected on a shaft. The stepper-motor was actuated in microscope-mode. The second shaft (see Figure 1) is connected to the first shaft using an angular positioning unit, whose resolution is 1'. A personal computer controls the stepper-motor, and reads out the reference sensor and out prototype sensor.

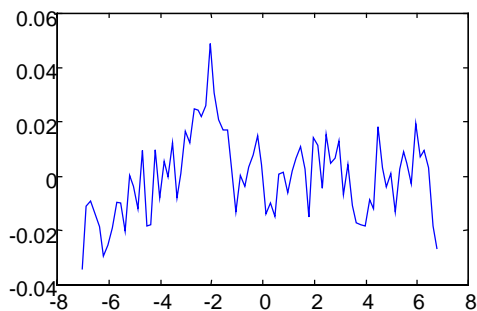
The first measurement was based on the sensor topology described above. The measurement result showed, that this rotor structure is very sensible to radial displacement. Therefore two further structures have been developed with a 180°-symmetry, and a thus reduced measurement range of 180° for the absolute angle and a range of the relative angle of ±7.5°. The new structures use 32 transmitting segments with centre angles of 11.25°

and rotors with four blades and centre angles of  $30^\circ$ . Each segment is electrically connected to its mirror segment producing again 16 segment values for each measurement cycle. As described in [6] this structure is insensitive against radial displacement.

The third structure measured was the so called “inverse topology”. This structure is the mechanical negation of the topology explained before: the size of the blades is constant while the free angles between the blades vary with the relative angle. Both, inverse and normal structure, uses an effective rotor-shape with 4 rotor blades. While both rotors of the normal structure consist of 4 rotor blades, the rotors of the inverse structure just need 2 rotor blades. So the rotor of the inverse topology can be easier manufactured.



**Figure 5 – relative angle error versus absolute position**



**Figure 6 – relative angle error versus relative angle range**

Figure 5 and Figure 6 show the measurement results we reached using the inverse topology. As mentioned before the sensor needs calibration. In the case of the inverse structure  $180^\circ$  are needed to calibrate the sensor. The gray shaded area in Figure 5 gives the results of the uncalibrated sensor. After calibration the error is within  $\pm 0.1^\circ$ . Figure 6 shows the relative angle error versus the relative angle. This error is below  $0.08^\circ$ .

## Conclusions

Based upon the described sensor prototype it is possible to realise contact less torque sensors for rotating shafts, measuring the torque in mechanical systems in harsh environments. Future developments will concentrate in increasing the resolution of the relative angle measurement. The dynamical performance of the sensor will be evaluated.

## References

- [1] Georg Brasseur, Thomas Eberharder : *Kapazitiver Drehwinkelsensor*, "Österreichisches Patent AT 398245; 1994 und Europ. Pat. EP 0551 066, 1992 und US Pat. 5,598,153.
- [2] Georg Brasseur, Thomas Eberharder : *Capacitive Angular Displacement Transducer*, US Patent Appl. S/N 08/087,261;1993
- [3] Tibor Fabian, Georg Brasseur : *A Robust Capacitive Angular Speed Sensor in Proceedings of IEEE Conference on Instrumentation and Measurement Technology (IMTC/97)*, Ottawa, Canada, May 19-21, pp. 1267-1272, 1997.
- [4] Tibor Fabian : *Vergleich und Optimierung der Ansteuerverfahren eines kapazitiven Winkel- und Winkelgeschwindigkeitssensors*, Diplomarbeit, TU-Wien, 1998
- [5] Florian Wandling: *Hardwareentwicklung für einen kapazitiven Drehmomentsensor*, Diplomarbeit, IAEQ, TU-Wien, 1999
- [6] Wolfgang Zdiarsky: *Vergleich verschiedener Rotorstrukturen eines kapazitiven Drehmomentsensors*, Diplomarbeit, IAEQ, TU-Wien, 1999

## Corresponding author:

Prof. Georg Brasseur,  
 Institut für Meßtechnik und Meßsignalverarbeitung,  
 Technische Universität Graz,  
 Kopernikusgasse 24/IV, A-8010 Graz,  
 Austria, Europe  
 Phone: +43 316 873 7270,  
 Fax: +43 316 873 7266,  
 E-Mail: brasseur@emt.tu-graz.ac.at

# Index

- Abkürzungen, vii
- Addaption, 63
- Algorithmus
  - 1, 25, 41, 42, 45
  - 2, 25, 41, 43, 45, 61, 65
- Anforderungen, 2
- Baldwin, 75
- Baudrate, 69, 75
- beschreibende Gleichungssystem, 31
- DDE
  - Item, 78
  - Makro, 78
  - Schnittstelle, 77
- Dehnungsmeßstreifen, 3
- Elektrodenstruktur, 14
  - Ersatzschaltbild, 25
- Endlosmessung, 75
- Feld
  - Linienladung, 27
  - Streifenladung, 28
- Kapazitätsverlauf, 37
- Korrelation, 18, 59
- Ladungsverteilung, 26
- Magnetoelastischer Momentensensor, 4
- Meßkette, 6
- Meßplatzsoftware, 73
- Meßplatzsteuerung, 74, 75
- Momentenmessung
  - Dehnungsmeßstreifen, 3
  - Hebel, 3
  - kapazitiv, 6
  - magnetoelastisch, 4
  - piezoelektrisch, 4
  - Schwingsaiten, 5
- optimale
  - Rotorhöhe, 43
  - Sensorhöhe, 41
- Paper, 21, 87
- Piezoelektrischer Momentensensor, 4
- Plattenkondensator, 36
- Plattenkondensatormodell, 9
- PWM, 47, 67
- ratiometrisch
  - Gleichung, 16, 18
  - Prinzip, 7
- Schwingsaiten, 5
- Segmentwerte, 58
- seriell, 68
  - Befehl, 69, 72, 76
  - Protokoll, 69, 76
- Simulation
  - Rotorhöhe, 43
  - Rotorverkipfung, 45
  - Sensorhöhe, 41
- Spannung
  - Analog-Digitalumsetzer, 57
  - Streifenladung, 29
- Spezifikation, 79
- Torsion einer Welle, 3, 4, 12
- Watchdog, 52
- Winkel
  - absolut, 17, 65
  - grob, 18, 60
  - relativ, 16, 65
- Zeichen und Symbole, viii