

Erweiterung einer Auswerteelektronik für taktile Sensoren zum multimodalen Sensorsystem

Bachelorarbeit

Denis Schneider



BACHELORARBEIT

**ERWEITERUNG EINER AUSWERTE-
ELEKTRONIK FÜR TAKTILE SENSOREN
ZUM MULTIMODALEN SENSORSYSTEM**

Freigabe:

Der Bearbeiter:

Unterschriften

Denis Schneider

Betreuer:

Michael Strohmayer

Der Institutsdirektor

Prof. Dr. G. Hirzinger

Dieser Bericht enthält 81 Seiten, 38 Abbildungen und 5 Tabellen



**Hochschule
Augsburg** University of
Applied Sciences

Fakultät für
Elektrotechnik

Bachelorarbeit

Mechatronik

Denis Schneider

Erweiterung einer Auswerteelektronik für taktile Sensoren zum multimodalen Sensorsystem

Verfasser der Bachelorarbeit

Denis Schneider
Industriestraße 14
86438 Kissing

Telefon +49 176 992 859 78
denis.schneider@hs-augsburg.de

Prüfer: Prof. Dr.-Ing. Franz Raps

Thema erhalten am: 13.01.2012

Eingereicht am: 13.06.2012

Hochschule für
angewandte Wissenschaften –
Fachhochschule Augsburg
University of Applied Sciences

An der Fachhochschule 1
D-86161 Augsburg

Telefon +49 821 55 86-0
Fax +49 821 55 86-3222
www.hs-augsburg.de
info@hs-augsburg.de

Eingereicht am 13.06.2011
von Denis Schneider

Studiengang Mechatronik
Hochschule Augsburg - Fakultät für Elektrotechnik

Prüfer: Prof. Dr.-Ing. Franz Raps

Betreuer: Prof. Dr.-Ing. Franz Raps
Dipl.-Ing.(FH) Michael Strohmayer M.Sc.(TUM)

Institut für Robotik und Mechatronik
Deutsches Zentrum für Luft- und Raumfahrt e.V.

Eidesstattliche Erklärung

Hiermit erkläre ich, dass ich die vorliegende Bachelorarbeit

„Erweiterung einer Auswerteelektronik für taktile Sensoren zum multimodalen Sensorsystem“

selbständig erstellt und keine anderen als die angegebenen Hilfsmittel und Quellen verwendet habe. Die aus fremden Quellen (einschließlich elektronischer Quellen) direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht worden. Die Arbeit ist in gleicher oder ähnlicher Form oder auszugsweise im Rahmen einer anderen Prüfung noch nicht vorgelegt worden.

Oberpfaffenhofen, den 13.06.2012

Denis Schneider

Sperrvermerk

Die vorliegende Bachelorarbeit beinhaltet vertrauliche Informationen. Die Weitergabe des Inhaltes der Arbeit und eventuell beiliegender Zeichnungen und Daten im Gesamten oder in Teilen ist grundsätzlich untersagt. Es dürfen keinerlei Kopien oder Abschriften, auch nicht in digitaler Form, ohne Genehmigung gefertigt werden.

Zusammenfassung

In dieser Bachelorarbeit wird die Weiterentwicklung einer bestehenden Auswerteelektronik für polymerbasierte taktile Sensoren zu einem Sensorsystem beschrieben. Den Kern dieser Arbeit stellt die Minimierung des Raumbedarfs der elektronischen Schaltung durch Verwendung von modernen Fertigungstechniken, sowie die Erweiterung deren sensorischer Fähigkeiten dar. Desweiteren wird mit den Konzepten dieser Arbeit die Grundlage, seitens des Sensorsystems, für die Kommunikation zwischen einem taktilen Sensorsystem und den Schnittstellen einer Robotersteuerung geschaffen.

Danksagung

An dieser Stelle möchte ich mich bei Herrn Prof. Dr.-Ing. Franz Raps für die Betreuung von Seiten der Hochschule Augsburg bedanken. Von Seiten des Instituts für Robotik und Mechanik des Deutschen Zentrums für Luft- und Raumfahrt bedanke ich mich besonders bei Herrn Dipl.-Ing. (FH) Michael Strohmayer M.Sc. (TUM) für die engagierte Unterstützung, Betreuung und fachliche Inspiration.

Weiterer Dank gilt den Herren Robin Gruber, Rene Rittgarn, Nikolaus Seitz und Patrick Leyendecker für Ihr Engagement bei technischen Fragen. Für die produktive Zusammenarbeit möchte ich mich bei Christian Leupolz, Florian Schnös, Sven Grob, Daniel Pourroy und dem Team der Elektronikwerkstatt bedanken. Insbesondere danke ich meinem Vater Erich Schneider, meinem Bruder Marco Schneider und meiner Freundin Vera Hartmann die mich stets während meines Studiums unterstützt haben.

Inhaltsverzeichnis

Eidesstattliche Erklärung	I
Sperrvermerk	I
Zusammenfassung	II
Danksagung	III
Tabellenverzeichnis	VI
Abbildungsverzeichnis	VII
1 Einleitung	1
1.1 Problembeschreibung	2
1.2 Rahmenbedingungen	2
1.3 Aufgabenstellung	2
2 Stand der Technik	3
2.1 Forschungsstand taktiler Sensorsysteme	3
2.2 Entwicklungsstand der Auswerteeinheit	4
2.2.1 Auswerteelektronik	4
2.2.2 Mikrocontrollerfirmware	4
2.2.3 Visualisierungssoftware	5
2.3 Moderne Fertigungstechniken für elektronische Komponenten	6
2.3.1 Leiterplattenherstellung	6
2.3.2 Bauteilgehäuse	7
2.3.3 Mikrosystemtechnik	8
2.4 Bussysteme	9
3 Theoretische Grundlagen	11
3.1 Messtechnische Grundlagen	11
3.1.1 Transimpedanzverstärker	11
3.1.2 Aktive Filter	11
3.1.3 Druckmessung	12
3.1.4 MEMS Mikrofone	14
3.2 Spannungsversorgung	14
3.3 Schirmung gegen Störeinkopplungen	15
3.4 Immunisierung von Schaltungen gegen elektrostatische Entladung	16
3.5 Echtzeitsysteme	16
4 Weiterentwicklung und Realisierung der Elektronik	18
4.1 Redesign der Schaltung	18
4.1.1 Auswahl der Bauteile	19
4.1.2 Energieversorgung	20

4.1.3	Anpassung der Messketten	23
4.1.4	Implementierung des Schutzes vor elektrostatischen Entladungen . . .	26
4.2	Design der Leiterplatte	26
4.2.1	Aufbaukonzept	27
4.2.2	Layeraufbau	28
4.2.3	Implementierung der Schirmung	29
5	Entwicklung der Software	30
5.1	Softwarekonzept	30
5.1.1	Systemkonzept und Datenkanalisierung	30
5.1.2	Softwaremodule der Mikrocontrollerfirmware	31
5.1.3	Ablaufdiagramme	32
5.1.4	Kommunikationsprotokoll	34
5.2	Bootloader und In System Update	35
5.3	Echtzeitfähige Kommunikation	36
6	Ergebnisse	37
6.1	Verifizierung der Anforderungen	37
6.2	Sensorsystem im Betrieb	39
7	Ausblick	43
	Literaturverzeichnis	44
	Anhang	A-1
A	Schaltpläne der Evaluationsplatine	A-1
B	Layout der Evaluationsplatine	A-5
C	Kommunikationsprotokoll	A-6
D	Serienbild vom Einschlagtest	A-7
E	Quellcode der Mikrocontrollerfirmware	A-8

Tabellenverzeichnis

2.1	Häufig verwendete Sensortypen und deren prinzipbedingte Vor- und Nachteile .	3
3.1	Spannungsreglertypen mit ihren Vor- und Nachteilen	15
4.1	Stromaufnahme der verwendeten Bauelemente	21
4.2	Stromaufnahme der Schaltung gemessen im Betrieb	22
6.1	Technische Daten der Auswerteeinheit	38

Abbildungsverzeichnis

2.1	Schematische Darstellung der Auswerteelektronik	4
2.2	Grafik der Visualisierungssoftware	5
2.3	Fertigungstechniken der HDI-SBU Technik im Querschnitt	6
2.4	Prinzipskizze der Leiterplatte in Starrflexausführung	7
2.5	Grundlegende Bustopologien	9
3.1	Transimpedanzwandler	11
3.2	Operationsverstärker als aktiver Filter mit Mehrfachgegenkopplung	12
3.3	Querschnitt einer Druckmessbrücke zur Absolutdruckmessung	12
3.4	Mikroskopaufnahme eines Drucksensors mit Kompensationsschaltung	13
3.5	Prinzipskizze eines MEMS Mikrofons	14
3.6	Bearbeitung eines Ereignisses im Echtzeitchnersystem	17
4.1	Schaltbild des Sensors mit Multiplexer und Analogschaltern	19
4.2	Versuchsaufbau zur Messung von Oberflächenrauigkeiten	20
4.3	Regelung der im System benötigten Spannungen	20
4.4	Messkette	23
4.5	Simulationsmodell der Messkette	25
4.6	Sprungantwort für Widerstände von $50\text{ k}\Omega$ bis $1\text{ M}\Omega$	25
4.7	Hardwarekonzept nach Weiterentwicklung	27
4.8	Zielhardware und Leiterplattenlagenaufbau	28
4.9	Schirmungskonzept der Zielhardware	29
5.1	Konzept des Sensorverbundsystem	30
5.2	Modularer Aufbau der Software	31
5.3	Ablauf des Systemstarts	32
5.4	Hauptprogrammschleife	33
5.5	CAN Bootloader Netzwerk	35
6.1	Versuchsaufbau des Sensorsystems	39
6.2	Mobile Visualisierungseinheit im Versuchsaufbau	40
6.3	Messwerte bei Sprunganregung	40
6.4	Messwerte Kugeleinschlagversuch Übersicht	41
6.5	Messwerte Kugeleinschlagversuch Vergrößerung	41
6.6	Innengeräusche der Kavität bei Anregung mit verschiedenen Materialien	42
A.1	Schaltplan digitale Komponenten	A-1
A.2	Schaltplan analoge Komponenten	A-2
A.3	Schaltplan Stromversorgung	A-3
A.4	Schaltplan Dokumentation	A-4
B.1	Layout der vierlagigen Evaluationsplatine	A-5
C.1	Kommunikationsprotokoll	A-6
D.1	Einschlagversuch mit einer Stahlkugel als Serienbild	A-7

1 Einleitung

Die aktuellen Fortschritte in der Robotik ermöglichen es, immer komplexere Arbeiten maschinell zu erledigen. Stand die Robotik anfangs noch vor dem Problem nur gleichbleibende und damit fest programmierbare Arbeitsabläufe ausführen zu können, sind moderne Robotersysteme in der Lage autonom in den Arbeitsprozess einzugreifen. Dies trägt zur Erhöhung der Sicherheit beim Arbeiten mit Robotern bei und erlaubt es anspruchsvolle Arbeiten mit Hilfe von Robotersystemen auszuführen.

Um sicher mit der Umwelt interagieren zu können, benötigen diese Robotersysteme Sensoren, die ihre Umgebung überwachen. Mit optischen Systemen kann beispielsweise die Orientierung von Objekten im Raum erfasst werden. Weitere Objekteigenschaften, wie zum Beispiel das Gewicht oder die Oberflächenbeschaffenheit, können während des physikalischen Kontakts durch taktile Sensoren bestimmt werden.

Wie beim menschlichen Tastsinn können mit der Information über die Kraft und Fläche, mit der ein Objekt gegriffen wird, deutliche Verbesserungen bei der Handhabung von Gegenständen erzielt werden.

So kann beispielsweise mit Hilfe der Information von den taktilen Sensoren auf dem Greifer des Roboters entschieden werden, wie fest ein Gegenstand gegriffen werden muss, um die gewünschte Interaktion durchführen zu können, ohne den Gegenstand zu beschädigen. Hierdurch können mechanische Greifer eines Robotersystems deutlich effizienter und in einem breiteren Verwendungsspektrum eingesetzt werden.

Bei der Klassifizierung von Objekten spielt die Oberflächenbeschaffenheit eine wichtige Rolle. Ist die Oberflächenbeschaffenheit bekannt, kann das Robotersystem den Greifvorgang an das Objekt anpassen.

Eine weitere wichtige Funktion der menschlichen Haut ist die Detektion von Kollisionen mit Objekten. Hierbei ist in der Robotik eine möglichst kurze Reaktionszeit gewünscht, um beim Erkennen einer Kollision schnellstmöglich auf das Ereignis reagieren zu können. So kann vermieden werden, dass weder der Roboter noch das an der Kollision beteiligte Objekt oder gar ein Mensch Schaden nimmt. Ermöglicht werden kann dies durch taktile Sensorik auf Robotersystemen. Die Sensorik erkennt eine Kollision und erlaubt es der Robotersteuerung Gegenmaßnahmen wie z.B. eine Umkehr der Bewegungsrichtung einzuleiten.

Aus diesen Anwendungsszenarien ergeben sich die folgenden Anforderungen an das taktile Sensorsystem. Es muss mechanisch robust sein, sowie die Fähigkeit besitzen, die Kraftverteilung und die Oberflächenbeschaffenheit eines Objekts zu bestimmen. Zudem müssen Kollisionen erkannt und ein Durchrutschen von Gegenständen während der Manipulation detektiert werden.

Die vorliegende Bachelorarbeit befasst sich mit der Weiterentwicklung einer bestehenden Auswerteelektronik für polymerbasierte taktile Sensoren. Ein Verbund aus mehreren Sensoreinheiten, bestehend aus einem taktilen Sensor und einer Auswerteelektronik, soll zukünftig zu einer künstlichen Haut für die Robotik kombiniert werden. Für diese Arbeit stehen die Reduzierung der Bauteilgrößen, sowie die Entwicklung eines Konzepts zur Informationsverarbeitung für räumlich verteilte Sensoren im Vordergrund.

1.1 Problembeschreibung

Eine bestehende einfache Auswerteelektronik soll zu einem Sensorverbundsystem weiterentwickelt werden. Dabei wird primär gefordert, dass mehrere Sensorelektroniken im Verbund betrieben werden können und eine Schnittstelle geschaffen wird, um mit verarbeitenden Robotersteuerungen kommunizieren zu können. Zudem soll die Baugröße der bestehenden Auswerteeinheit deutlich verringert werden, um Vorteile bei der Integration der Sensoren in Robotersystemen mit geringen Platzreserven (z.B. Greifer- oder Handsysteme) zu schaffen. Darüber hinaus sollen bei der Entwicklung folgende Punkte berücksichtigt werden.

- Beibehaltung des CAN (Controller Area Network, nach [22]) Bussystems
- Verbesserung der Messgeschwindigkeit
- Reduzierung der Messwertschwankungen
- Erweiterung der Sensormodalitäten
- Schirmung gegen äußere Störeinkopplungen
- Immunisierung der Schaltung gegen elektrostatische Entladungen
- Einhaltung der Echtzeitbedingungen (vorsehen)
- Entwicklung eines Kommunikationsprotokolls für das Sensorverbundsystem
- Einfache Updatemöglichkeit der Firmware im Mikrocontroller
- Anpassung der Visualisierungssoftware (Verarbeitung von Sensorverbundsystemdaten)

1.2 Rahmenbedingungen

Die Aufgabenstellung wird als Bachelorarbeit an der Fakultät für Elektrotechnik der Hochschule Augsburg, unter Betreuung von Prof. Dr.-Ing. Franz Raps, in Kooperation mit dem Institut für Robotik und Mechatronik des Deutschen Zentrums für Luft- und Raumfahrt, unter Betreuung von Dipl.-Ing. (FH) Michael Strohmayer M.Sc. (TUM) bearbeitet. Den Ausgangspunkt für diese Arbeit stellt die Bachelorarbeit von Herrn Florian Schnös, nach [24] dar. Die Weiterentwicklung, Redesign der Schaltung, Realisierung und Erprobung sowie die Implementierung der Firmware für die Elektronik soll in einem Zeitrahmen von 5 Monaten durchgeführt werden (2 Monate Einarbeitung, 3 Monate Bearbeitung).

1.3 Aufgabenstellung

In dieser Arbeit wird eine einfache bestehende Auswerteelektronik für polymerbasierte taktile Sensoren zu einem multimodalen Sensorsystem erweitert. Zentrale Aufgabe bei der Erweiterung der Elektronik ist die Reduzierung des Platzbedarfs. Dies stellt die Grundvoraussetzung für eine zukünftige Funktion in Robotersystemen dar. Desweiteren soll die Auswerteelektronik die Fähigkeit besitzen, flexibel mit anderen Sensoreinheiten als Sensorverbundsystem zu arbeiten.

2 Stand der Technik

2.1 Forschungsstand taktiler Sensorsysteme

Taktile Sensoren können nach [8, S.292 ff], abhängig vom gewählten physikalischen Übertragungsprinzip in die folgenden aufgelisteten Kategorien eingeteilt werden:

- Resistiv
- Kapazitiv
- Tunneleffekt
- Optisch
- Ultraschallbasiert
- Magnetisch
- Piezoelektrisch

Besonders häufig werden resistive und kapazitive Übertragungsprinzipien für taktile Sensorsysteme verwendet. In Tabelle 2.1 werden die Vor- und Nachteile der Übertragungsprinzipien für den Einsatz in der Robotik betrachtet.

Sensorprinzip	Vorteile	Nachteile
Resistiv	<ul style="list-style-type: none">• Hohe Sensibilität• Geringe Kosten	<ul style="list-style-type: none">• Hoher Energiebedarf• Häufig nur ein Kontakt auswertbar
Kapazitiv	<ul style="list-style-type: none">• Hohe Sensibilität• Geringe Herstellungskosten	<ul style="list-style-type: none">• Komplexe Auswertung notwendig• Crosstalk• Hystereseverhalten

Tabelle 2.1: Häufig verwendete Sensortypen und deren prinzipbedingte Vor- und Nachteile

Taktile Sensoren (Sensoren die mechanische Eingangsgrößen in elektrisch messbare Größen umwandeln) auf Basis von resistiven und kapazitiven Sensorzellen werden kommerziell häufig zur Berührungssteuerung in Mobilfunkgeräten verwendet. Folglich sind für diese Umsetzungsprinzipien auch integrierte Auswertungsschaltungen (integrated circuits, kurz ICs) verfügbar.

Diese bieten jedoch zumeist keine Möglichkeit die Größe der einwirkenden Kraft zu erfassen, welche auf die Taxel (einzelnes Element zur Wandlung von Kraft in eine elektrisch messbare Größe) aufgebracht wird. Dies bedeutet, dass lediglich die Lage des Kontaktereignisses auf dem Sensor bestimmbar ist.

Auf Grund der in Kap. 1.2 genannten Rahmenbedingungen wird in dieser Arbeit ein resistives Übertragungsprinzip auf Basis von leitfähigen Polymeren nach [29] betrachtet. Für die Weiterentwicklung des Sensorsystems ist zu beachten, dass nicht wie üblich nach dem Spannungsteilerprinzip lediglich der Ort des Kontaktereignisses bestimmt wird, sondern sequenziell jedes Taxel ausgewertet wird. Hierdurch ist es möglich die Lage eines Kontaktereignisses über die Taxelposition und die Kraft über den Übergangswiderstand, der sich berührenden Sensorbahnen, zu bestimmen, nach [24, Kap. 4.3].

2.2 Entwicklungsstand der Auswerteeinheit

Die folgenden Unterkapitel fassen den aktuellen Entwicklungsstand der Auswerteeinheit für taktile Sensoren zusammen.

2.2.1 Auswerteelektronik

Die zu erweiternde Auswerteelektronik besteht derzeit aus einer Auswerteschaltung, die mit Hilfe eines Mikrocontrollers (Mikrocontroller Unit, kurz MCU) über einen Multiplexer eine Spalten-treiberspannung an die taktile Sensorleitungen anlegt und über die Zeilen den fließenden Strom misst. Nach Pegelanpassung und Filterung wird das Signal von einem Analog-/Digitalwandler (A/D-Wandler) digitalisiert und dem Mikrocontroller zugeführt. Von hier aus werden die Messwerte in Blöcke gegliedert und über einen CAN-Bus an den Visualisierungsrechner gesendet. Für die Spannungsversorgung kann eine Gleichspannung von 7,5 V bis 12 V verwendet werden. Die äußeren Abmaße der Auswerteelektronik betragen 49 mm x 29 mm. Die folgende Abbildung fasst schematisch den Schaltungsaufbau zusammen.

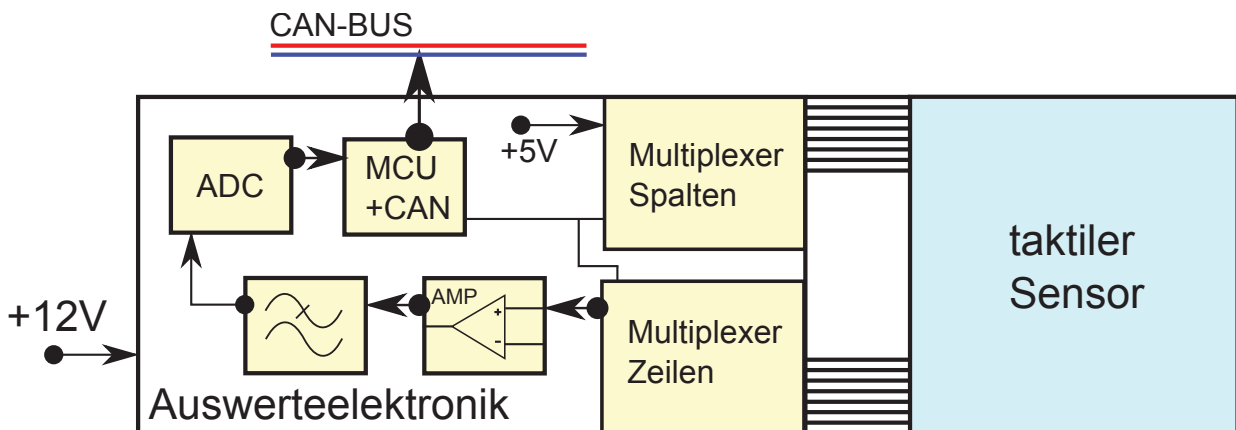


Abbildung 2.1: Schematische Darstellung der Auswerteelektronik

Hierbei muss beachtet werden, dass in Abhängigkeit des taktile Widerstands des Sensors der Messwiderstand des verwendeten Transimpedanzwandlers gegebenenfalls angepasst werden muss (Widerstand auf der Platine muss getauscht werden), um den Auflösungsverlust zu verringern. Ebenfalls muss in die Schaltung eingegriffen werden, wenn die verwendeten Filter, welche fest auf eine Grenzfrequenz von ca. $f_0 = 87 \text{ kHz}$ eingestellt sind, verändert werden sollen.

Eine funktionsfähige Auswerteelektronik, einen Visualisierungsrechner mit Visualisierungssoftware und die Quelltexte der Mikrocontrollerfirmware wurden zu Beginn der Arbeit vom Institut bereitgestellt.

2.2.2 Mikrocontrollerfirmware

Die Mikrocontroller Steuersoftware (Firmware) ist in der Programmiersprache C verfasst. Sie steuert die Funktionen der in der Auswerteeinheit verwendeten Bausteine, wie Multiplexer, A/D-Wandler und CAN-Bus Interface. Die Software ermittelt die in Blöcken übertragenen Messwerte

von 8 x 8 Taxeln. Vor Beginn der Messwerterfassung wird das System initialisiert und die Elektronik in einen funktionsfähigen Zustand überführt.

Anschließend werden alle Taxel sequenziell ausgelesen und eine Matrix zum Nullpunktgleich erzeugt. Nun werden zyklisch, durch Schalten der Multiplexer alle Taxel ausgelesen, der Nullpunktwert subtrahiert und die erzeugten Daten als Block per CAN-Bus an den Visualisierungsrechner versendet.

Die Geschwindigkeit, in der die Datenblöcke ermittelt werden, wird beim Übersetzen der Software vorgegeben. Zur Laufzeit kann nicht steuernd in das System eingegriffen werden. Ist der Empfänger nicht in der Lage, die von der Auswerteeinheit gesendeten Daten schnell genug zu verarbeiten, wartet die Elektronik bis der Empfänger weitere Daten verarbeiten kann.

2.2.3 Visualisierungssoftware

Die momentan verwendete Software zur Visualisierung der Sensordaten ist im Rahmen der Sensorentwicklung eigens verfasst worden. Die Software erzeugt eine Grafik auf dem Bildschirm des Visualisierungsrechners für die vom CAN-Bus über ein CAN-USB Interface aufgenommenen Daten.

Die Visualisierung der Kontakt Ereignisse auf einem taktilen Sensor mit 8 x 8 Taxeln erfolgt derzeit über eine Matrix aus 72 x 72 Quadern, die sich in Abhängigkeit vom Messwert in der Höhe und Farbe ändern (Abstufungen werden interpoliert). Im momentanen Stand der Software kann maximal ein angeschlossener Sensor ausgewertet und visualisiert werden.

Über ein Menü in der Visualisierungssoftware kann die Anzahl der auszuwertenden Taxel in X- und Y-Richtung, eine nachträgliche digitale Verstärkung und einen Schwellwert zum Ausblenden der Sensorwerte eingestellt werden. Zudem kann die Anzeige rotiert und skaliert werden. Um die aktuelle Auswertegeschwindigkeit zu beobachten, kann die Wiederholrate eingeblendet werden. Um die Darstellung der Werte besser zu verdeutlichen, kann wahlweise zwischen positiver und negativer Auslenkung der Quader umgeschaltet werden. Die Abbildung 2.2 zeigt die Auswertung eines an drei Taxeln belasteten Sensors, hierbei stellt der Farbverlauf von rot nach weiß steigende Werte dar.

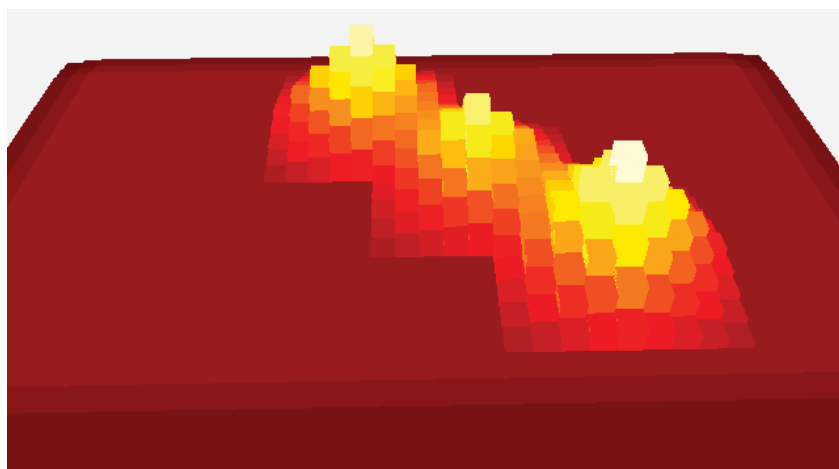


Abbildung 2.2: Grafik der Visualisierungssoftware

2.3 Moderne Fertigungstechniken für elektronische Komponenten

Um eine Auswerteelektronik mit geringen Abmaßen herzustellen, werden Multilayerfertigungstechniken (vier Lagen und mehr) eingesetzt, welche in den folgenden Abschnitten näher behandelt werden. Die Entwicklung der Leiterplatte soll nach der bestmöglichen Flächenausnutzung gestaltet werden.

2.3.1 Leiterplattenherstellung

Bei der Herstellung von Leiterplatten sind zum momentanen Stand der Technik die folgenden Fertigungsverfahren im Hinblick auf eine geringe Gesamtabmaße in Betracht zu ziehen:

- High Density Interconnection / Sequential Build Up (HDI-SBU) Multilayer
- Flexible Leiterplatten
- Starrflexible Leiterplatten

Die Fertigung im HDI-SBU Verfahren ermöglicht extrem hohe Integrationsdichten. Diese werden nach [6] durch die Verwendung von speziellen Herstellungstechniken erreicht:

- Allgemein sehr niedrige Strukturbreiten (ab $75\ \mu\text{m}$ Leiterbahnstärke (A) und Abstand (B))
- Microvias, Laserankontaktierung der Außenlagen mit Durchmessern ab $100\ \mu\text{m}$ (E)
- Blind Vias, in einer Innenlage endende Ankontaktierung (3)
- Buried Vias, nach Außen nicht sichtbare Durchkontaktierung der Kernlagen (4)

Die Abbildung 2.3 zeigt die oben genannten Maße in einer Grafik, übernommen aus [6]:

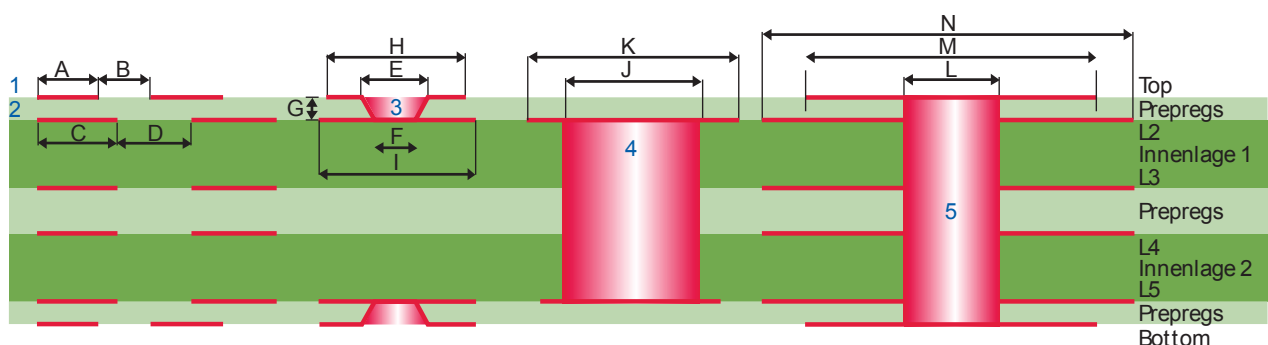


Abbildung 2.3: Fertigungstechniken der HDI-SBU Technik im Querschnitt

Die Verwendung einer Leiterplatte, die vollflexibel ist, scheidet für den vorliegenden Anwendungsfall aus, da hierbei nur maximal zwei Lagen angeboten werden, welche für die gewünschte Integrationsdichte nicht ausreichend sind.

Für die erweiterte Auswerteelektronik ist die Fertigung als starrflexible Leiterplatte vorteilhaft, da mit dieser Technik die Anschlussleitungen zum Sensor flexibel ohne Anbringen von Steckverbindern an der Elektronik realisiert werden können. Zudem sind durch den Multilayeraufbau vier oder mehr Lagen herstellbar.

Die Verwendung der Starrflextechnik erlaubt es, die benötigten Bauteile auf die starren Platinelemente aufzubringen und dadurch die mechanische Belastung der Bauteillötstellen gering zu halten. Somit bietet diese Technik für die Weiterentwicklung der Auswerteelektronik eine flexible Layoutgestaltung.

Nicht zu vernachlässigen ist, dass Fertigungstechniken, wie HDI-SBU und Starrflex auch nach Abschluss der Entwicklung stark an den schon in der Entwicklung mitwirkenden Leiterplattenhersteller binden und die Herstellungskosten je nach Lagenanzahl und Herstellungstechnik um bis zu 50% in Bezug auf die Standardmultilayer-technik bei gleicher Lagenanzahl steigen, nach [21].

Die nachfolgende Abbildung 2.4 zeigt das für die erweiterte Leiterplatte gewählte Prinzip unter Verwendung der Starrflextechnik. Hier grün abgebildet die Multilayerleiterplatte und braun die Flexverbinder aus dem Kern der Platine kommend.

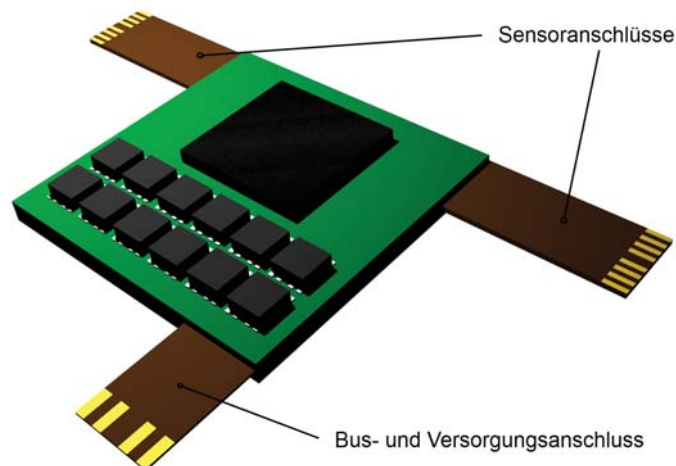


Abbildung 2.4: Prinzipskizze der Leiterplatte in Starrflexausführung

2.3.2 Bauteilgehäuse

Bei der Wahl der Bauteilgehäuse können durch die Baugröße bedingt nur oberflächenmontierbare Bauelemente (surface-mounting technology, kurz SMT) verwendet werden, da durchsteckmontierbare Bauelemente (through-hole technology, kurz THT) mit einem üblichen Rastermaß von 2,54 mm nicht für hohe Integrationsdichten geeignet sind.

Die SMT bietet große Vorteile bei der Minimierung der Gesamtbaugröße, vor allem in Kombination mit den oben genannten HDI-Leiterplatten kann so der Flächenbedarf signifikant verringert werden. Um die verwendeten Bauteile auf den institutsinternen Automaten bestücken zu können, sollte ein minimaler Abstand von 0,5 mm zwischen den Löt pads nicht unterschritten werden, da mit geringeren Padabständen die Ausschussrate steigen kann.

Aktuell erhältliche Bauteilgehäuse, nach Pindichte steigend sortiert, nach [23, Kap. 4.3]:

- PLLC (Plastic Leaded Chip Carrier)
- PQFP (Plastic Quad Flat Pack)
- QFNL (Quad Flatpack No Leads)
- BGA (Ball Grid Array)

In Hinsicht auf kleine Gesamtabmaße ist als ideales Gehäuse die BGA Form zu wählen. Jedoch wird diese Form in vielen Fällen nur für komplexe ICs mit hohen Pinzahlen angeboten. Ein Nachteil ist desweiteren der Aufwand für die Leiterplattenentflechtung, der durch die nicht von Außen zugänglichen, inneren Kontaktreihen entsteht. Um einen BGA Baustein mit hohen Pinzahlen zu entflechten, sind oft mehr als zwei Verdrahtungsebenen notwendig, die mit Microvias kontaktiert werden.

Im Gegensatz dazu sind die meisten ICs mit Pinanzahlen von 6 - 64 Pins zum Stand der Arbeit im QFNL Gehäuse erhältlich. Die QFNL Bauform besitzt, wie die BGA Bauform, keine über das Gehäuse herausragenden elektrischen Anschlüsse. Sie eignet sich für kleine bis mittlere Pinanzahlen bei moderater Verlustleistung, nach [11, Kap. 3.1]. Die thermische Leitfähigkeit, sowie der verbesserte Halt auf der Leiterplatte werden durch die große Montagefläche auf der Unterseite begünstigt.

2.3.3 Mikrosystemtechnik

Der Begriff der Mikrosystemtechnik ist in der Literatur noch nicht allgemein definiert worden, nach [18, Kap. 1]. Jedoch wird der Begriff oft im Umgang mit miniaturisierten mechanischen oder elektromechanischen Systemen verwendet. Die Herstellung von mikroelektromechanischen Systemen (MEMS) erfolgt üblicherweise mit Dünnschichttechniken.

Als Werkstoffe können nahezu beliebige Grundwerkstoffe (wie Metalle, Kunststoffe, Keramiken usw.) verwendet werden. Die grundlegenden Fertigungstechniken der Mikroelektronik auf Basis von Silizium kommen auch hier zum Einsatz, nach [18, Kap. 2.1].

Elektromechanische Sensoren, Kleinstaktoren, Pumpen, Mikroschalter und Verbindungsmechanismen für die optische Nachrichtentechnik lassen sich mit den Verfahren der Mikrosystemtechnik fertigen, nach [18, Kap. 3,4,5].

In dieser Arbeit werden Sensoren auf Basis der Mikrosystemtechnik verwendet, zu den typischen Vertretern dieser Sensortypen zählen, nach [18, Kap. 3]:

- Drucksensoren
- Beschleunigungssensoren
- Drehratensensoren

Desweiteren können mit diesen Verfahren auch andere elektromechanische Systeme in miniaturisierter Form hergestellt werden, wie zum Beispiel kleinste Mikrofone, die in Experimenten dieser Arbeit in Kapitel 4.1.1 Verwendung finden.

Die Information die von mikrosystemtechnischen Sensoren erzeugt wird, beispielsweise im Hinblick auf Beschleunigungen, spielt bei der Verarbeitung der eigentlichen Daten des taktilen Sensors nur eine sekundäre Rolle, jedoch können sie dazu beitragen, günstige Zeitpunkte, wie die Lage im Raum oder den Belastungszustand des Sensors, für die Kalibrierung oder einen Nullpunktabgleich zu ermitteln.

Mit einem integrierten Drucksensor im taktilen Teil des Sensors kann der Innendruck bestimmt werden, wodurch ermittelt werden kann, ob der Sensor gerade belastet ist. In diesem Fall darf beispielsweise kein Nullpunktabgleich durchgeführt werden.

2.4 Bussysteme

Nach [14] ist ein Bus ein System zur Übertragung von Daten zwischen zwei oder mehr Teilnehmern, über einen gemeinsamen Übertragungsweg. Hierbei sind an der Übertragung zweier Teilnehmer andere nicht beteiligt.

Somit ist definiert, dass alle an einem Bus angeschlossenen Teilnehmer miteinander kommunizieren können.

Bussysteme können in den folgenden grundlegenden Topologien oder Kombinationen daraus realisiert werden, nach [25, Kap. 1]:

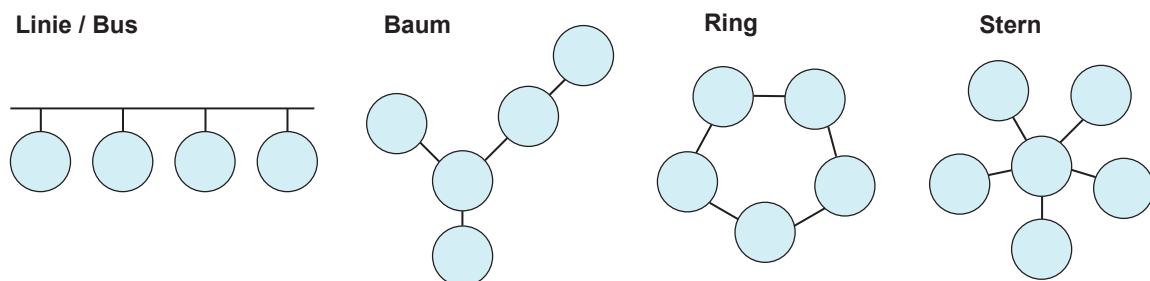


Abbildung 2.5: Grundlegende Bustopologien

Für die Integration der Auswerteeinheiten in eine Roboterhand erweisen sich die Baum- oder Linienstruktur als günstig, da mit ihnen die einzelnen Glieder z.B. des Fingers mit je einer eigenen Auswerteeinheit versehen werden können und hierbei nur eine Datenleitung längs des Fingers benötigt wird.

Zum Stand der Arbeit ist eine Vielzahl an verschiedensten Bussystemen verfügbar, jedoch stammen viele davon aus der Automatisierungstechnik und sind nur bedingt für den Einsatz in der Zielelektronik geeignet. Die kommerziell verfügbaren Bussysteme sind meist für große Produktionsanlagen mit langen Verbindungsdistancen konzipiert. Hierbei ist eine geringe Baugröße für die Busanschaltung nur selten notwendig. Zur Konfiguration und Diagnose komplexer Bussysteme ist zudem meist ein aufwendiges Softwaresystem oder eine speicherprogrammierbare Steuerung (SPS) notwendig.

Eine Auflistung verschiedener Bussysteme und ihren Spezifikationen wird in [25, Kap. 7] aufgeführt.

Demnach bietet das so genannte Controller Area Network oder kurz CAN-Bus den besten Kompromiss zwischen benötigter Bauteilgröße, einfacher Verdrahtung und Übertragungsrate.

Daher wird der CAN-Bus für die Datenübertragung zwischen den Auswerteelektroniken und dem damit verbundenen Knotenpunkt zu Datenbündelung gewählt. Der CAN-Bus bietet folgende günstige Eigenschaften für die Verwendung in der Auswerteelektronik, nach [25, Kap. 7.11]:

- Linientopologie vorteilhaft für die Integration in mechanischen Systemen
- Bis zu 1 Mbit/s Übertragungsrate bei Leitungslängen kleiner 40 m
- Bis zu 32 Busteilnehmer ohne zusätzliche Busrepeater möglich
- Einfache Verdrahtung durch geschirmte Zweidrahtleitung
- Verfügt über eine Datensicherungsschicht (Erkennen von Übertragungsfehlern und Wiederholung der Übertragung. Bei häufigen Fehlern wird der Teilnehmer vom Bus getrennt)
- Priorisierung der Nachrichten durch Identifier leicht realisierbar
- Gute Marktverfügbarkeit von Mikrocontrollern mit CAN-Interface und CAN-Transceiverbausteinen in kleinen Baugrößen

3 Theoretische Grundlagen

Dieses Kapitel fasst die theoretischen Grundlagen zum besseren Verständnis der folgenden Kapitel zusammen.

3.1 Messtechnische Grundlagen

3.1.1 Transimpedanzverstärker

Als Transimpedanzverstärker wird ein spannungsgesteuerter Operationsverstärker (VV-OPV) in einer Beschaltung zur Strom-Spannungs-Wandlung bezeichnet.

Die folgende Abbildung zeigt einen als Operationsverstärker beschalteten Transimpedanzverstärker:

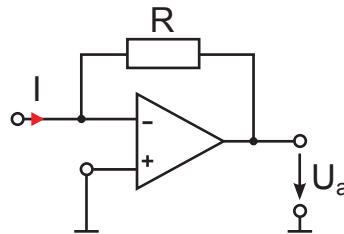


Abbildung 3.1: Transimpedanzwandler

Die Ausgangsspannung U_a für den Verstärker, bestimmt durch die Knotenregel und dem virtuellen Kurzschluss der beiden Eingänge des Operationsverstärkers:

$$U_a = -R \cdot I \quad (3.1)$$

Dieses ideale Verhalten gilt nur für den Fall, dass der Eingangswiderstand des Operationsverstärkers ∞ ist und somit kein Strom in den invertierenden Eingang fließt.

3.1.2 Aktive Filter

Aktive Filter basieren auf passiven Filtern, welche um aktive Komponenten erweitert sind (meist Operationsverstärker). Sie bieten in niedrigen bis mittleren Frequenzbereichen, bei Filtern höherer Ordnung den Vorteil, dass hierbei auf Induktivitäten verzichtet werden kann. Zudem wird das zu filternde Signal deutlich geringer belastet, als im Vergleich zu passiven Filterschaltungen.

Der in 3.1.1 beschriebene Strom-Spannungs-Wandler liefert ein invertiertes Messsignal. Ein nichtinvertiertes Signal am Eingang eines Analog-Digital-Wandlers bietet den Vorteil, dass es nicht per Berechnung invertiert werden muss, um das vom Sensor erzeugte Signal zu erhalten.

Ein aktiver, invertierender Filter zweiter Ordnung kann mit einer Mehrfachgegenkopplung realisiert werden, nach [33, Kap. 13.4.2].

Die folgende Abbildung zeigt einen Operationsverstärker als Filter mit Mehrfachgegenkopplung (Multiple Feedback Filter):

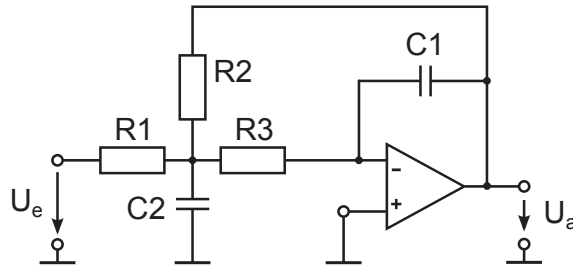


Abbildung 3.2: Operationsverstärker als aktiver Filter mit Mehrfachgegenkopplung

Die Übertragungsfunktion des Mehrfachgegekoppelten Filters lautet, nach [33, S. 865]:

$$\frac{U_a}{U_e} = \underline{A}(j\omega) = - \frac{\frac{R_2}{R_1}}{1 + j\omega C_1 \left(R_2 + R_3 + \frac{R_2 R_3}{R_1} \right) - \omega^2 C_1 C_2 R_2 R_3} \quad (3.2)$$

Die Filtercharakteristik wird durch die Wahl der Widerstände und Kapazitäten festgelegt. Hierbei wird zwischen den folgenden Filtercharakteristiken Bessel, Butterworth und Tschebyscheff unterschieden, welche sich durch ihr unterschiedliches Übertragungsverhalten je nach Anwendungsfall anbieten.

Für diese Arbeit wird im weiteren Verlauf ein Bessel-Tiefpaßfilter von Relevanz sein, da im Messsignal ein gutes Rechtecksübertragungsverhalten von Bedeutung sein wird, siehe Abschnitt 4.1.3. Die Quelle [33, Kap. 13.1] bietet eine weiterführende Beschreibung zur Auslegung von aktiven Filtern.

3.1.3 Druckmessung

Die derzeit kommerziell erhältlichen MEMS Drucksensoren zur Messung von Fluid- und Luftdruck basieren meist auf einer piezoresistiven Messbrücke, welche als Membran in einem geschlossenen Gehäuse integriert ist. Zu dieser Messmembran führen je nach Ausführung des Sensors, ob Absolut- oder Differenzdrucksensor, ein oder zwei Druckanschlüsse.

Die folgende Zeichnung zeigt den prinzipiellen Aufbau eines Absolutdrucksensors, nach [10]:

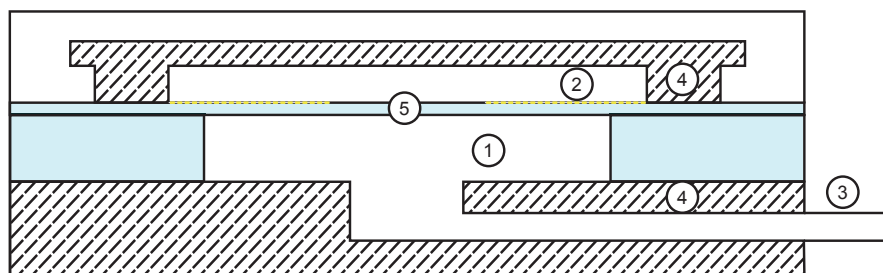


Abbildung 3.3: Querschnitt einer Druckmessbrücke zur Absolutdruckmessung

Nummerierungen in Abb. 3.3:

1. Druckkammer (für den zu messenden Druck)
2. Druckkammer (Referenzdruck)
3. Druckanschluss
4. Glasabdeckungen
5. Silizium Druckmessbrücke mit piezoresistiver Messvollbrücke (gelbe Strichlinie)

Nach [18, 3.1.5] sind Messbrücken, die eine Widerstandsänderung in Abhängigkeit vom eingebrachten Druck in die Druckkammer (1) aufweisen, nicht direkt auswertbar. Zur zuverlässigen Messwerterfassung müssen diese Sensoren temperaturkompensiert, sowie der Nullpunkt und die Empfindlichkeit kalibriert werden.

In der Praxis wird dies zumeist von einer digitalen Schaltung übernommen, welche die Messwerte der Messbrücke digitalisiert und mit Hilfe von hinterlegten Kompensations- und Kalibrierungswerten die Ausgabe errechnet. Diese Werte werden anschließend per Bussystem digital an einen Mikrocontroller übertragen oder durch einen Digital-Analog-Wandler in eine zum Druck proportionale Spannung umgesetzt.

Die Messrate dieser Sensoren liegt meist bei weniger als 1 kHz. Die kleinsten Bauteilabmessungen mit Gehäuse liegen zum derzeitigen Stand bei ca. 1,7 mm x 1,7 mm x 0,9 mm (z.B. EPCOS T5000).

Zum Anschluss an weiterverarbeitende Systeme sind Sensoren mit I²C (Inter-Integrated Circuit, nach [19]), SPI (Serial Peripheral Interface, nach [12, Kapitel 8]) und analoger Messwertausgabe erhältlich.

Die Aufnahme zeigt einen geöffneten Sensor (Honeywell HSC) mit Kompensationsschaltung:

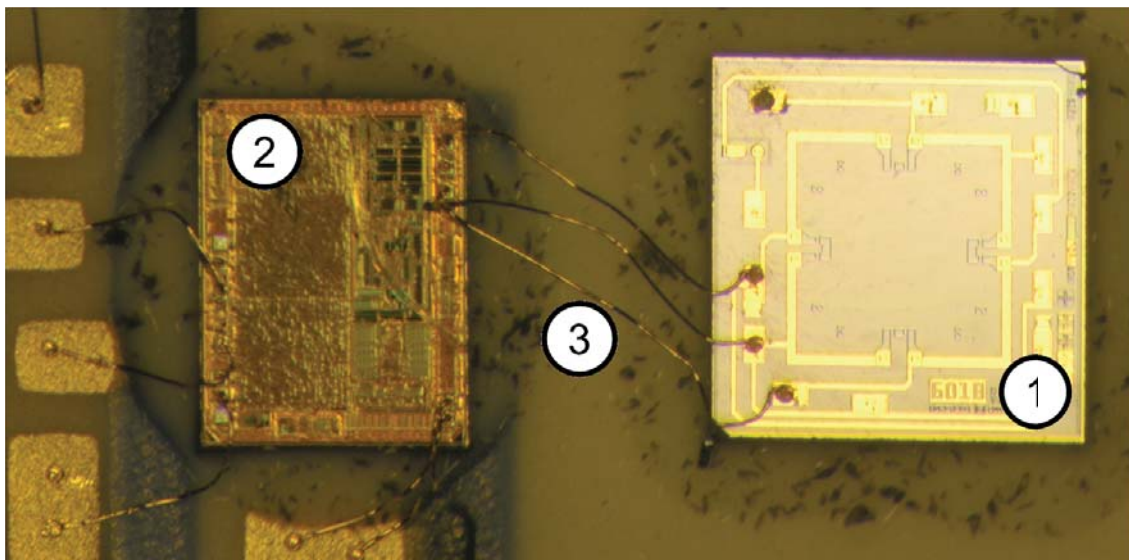


Abbildung 3.4: Mikroskopische Aufnahme eines Druckensors mit Kompensationsschaltung (1) Messbrücke, (2) Kompensationsschaltung, (3) Bonddrähte (mechanische Beschädigung der Bonddrähte durch Öffnen der Abdeckung.)

3.1.4 MEMS Mikrofone

Miniaturisierte Mikrofone zur Messung des Schalldrucks werden zumeist als Kondensatormikrofone realisiert. Hierbei wird eine dem Schall mitschwingende Membran in Kondensatoranordnung verwendet. Abbildung 3.5 nach [18, S. 116]

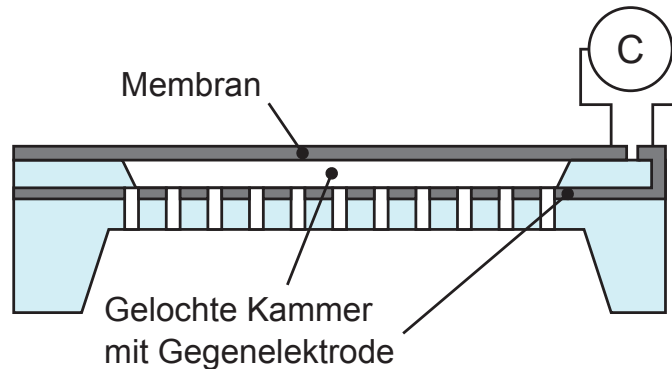


Abbildung 3.5: Prinzipskizze eines MEMS Mikrofons

Im Vergleich zu reinen Luftdrucksensoren wird eine hohe Dynamik bis 20 kHz erreicht, nach [18, Kap. 3.3.4]. Dies wird durch eine nur etwa $0,3 \mu\text{m}$ dünne, schwingfähige Membran, die an eine gelochte akustische Kammer angeschlossen ist, im Gegensatz zur Referenzdruckkammer bei Drucksensoren, ermöglicht.

Zum Stand der Recherche sind Mikrofone mit verschiedenen integrierten Signalaufbereitungsschaltungen verfügbar:

- Direkter analoger Ausgang
- Vorverstärkter analoger Ausgang
- Digitalausgang (Clock, Data) oder I²S (Inter-IC Sound Interface, nach [20])

Die äußeren Abmaße der Mikrofone liegen bei ca. 4 mm x 3 mm mit und bei ca. 3 mm x 2 mm ohne digital aufbereitetem Signalausgang.

3.2 Spannungsversorgung

Für die Versorgung mit der für den Betrieb benötigten Energie, steht der Schaltung eine Betriebsspannung von 12 V über ein Labornetzteil zur Verfügung.

Die Spannungen, die in der Auswerteelektronik für die Funktion benötigt werden (12 V, 5 V und 3,3 V), können nach den in Tabelle 3.1 beschriebenen Prinzipien geregelt werden, nach [13, Kap. 17 Abbildung 17-1], Vor- und Nachteile nach [5, Kap. 1.3]:

Reglerprinzip	Vorteile	Nachteile
Längsregler	<ul style="list-style-type: none"> • Gute Restwelligkeitsunterdrückung • Geringe Kosten • Keine Störabstrahlung • Kurze Entwicklungszeit • Geringer Bauteilbedarf 	<ul style="list-style-type: none"> • Hohe Reglerverluste • Leicht erhöhter Kühlaufwand • Nur für geringe Ströme geeignet • Meist nur ein Spannungsausgang
Schaltregler	<ul style="list-style-type: none"> • Geringe Reglerverluste • Hohe Ströme möglich 	<ul style="list-style-type: none"> • Erhöhte Restwelligkeit • Starke Störabstrahlung • Erhöhter Entwicklungsaufwand • Erhöhter Bauteilaufwand und Kosten

Tabelle 3.1: Spannungsreglertypen mit ihren Vor- und Nachteilen

Da die zu entwickelnde Schaltung nur geringe Ströme aufnehmen wird (kleiner 200mA) und zudem auf derselben Leiterplatte die analoge Signalkonditionierung stattfindet, welche durch die Störabstrahlung von Schaltreglern beeinträchtigt werden könnte, werden Längsregler zu Spannungsregelung verwendet. Die Dimensionierung der Energieversorgung wird in Abschnitt 4.1.2 genauer ausgearbeitet.

3.3 Schirmung gegen Störeinkopplungen

Die zu betrachtenden Störeinkopplungen in eine elektronische Schaltung lassen sich entsprechend ihrer physikalischen Einkopplungsprinzipien nach unterteilen, nach [27, Kap. 3]:

- Galvanische Kopplung (Leitungen)
- Kapazitive Kopplung (elektrisches Feld)
- Induktive Kopplung (magnetisches Feld)
- Elektromagnetische Kopplung (elektromagnetisches Feld)

Im zu entwickelnden System sind die Auswerteelektroniken an eine zentrale Stromversorgung angeschlossen, somit muss die galvanische Kopplung berücksichtigt werden.

Kapazitive Kopplung findet sowohl zwischen einzelnen Leitern der Auswerteelektronik, als auch als Störeinkopplung von außen, zum Beispiel durch Berühren des Sensors oder der Elektronik, statt.

Die induktive Kopplung wirkt zwischen zwei oder mehr stromdurchflossenen Leitern. Hierbei werden Störströme in den Leiterschleifen der Elektronik induziert.

Bei der elektromagnetischen Kopplung werden in den Leiterstrukturen der Elektronik von elektromagnetischen Wellenfeldern Störströme hervorgerufen.

Für die oben genannten Kopplungsmechanismen sind Gegenmaßnahmen beim Entwurf der Schaltung zu beachten.

3.4 Immunisierung von Schaltungen gegen elektrostatische Entladung

Die Immunisierung einer Schaltung gegen Beschädigung durch elektrostatische Entladung ist wichtig, um empfindliche Bauteile (meist ICs, gekennzeichnet durch ESD sensitive device Logo) zu schützen. Die große Gefahr für diese Bauelemente besteht darin, dass ihre Eingangsschaltungen oft hochohmig sind und nur geringe Eingangskapazitäten aufweisen. Zudem ist die maximale Spannung am Eingang oft geringer als 30 V, welche in diesem Fall schon bei kleinen Entladungen schnell überschritten werden.

Ist die Schaltung nicht in der Lage, die durch die Entladung eingebrachte Energie vom Anschlusspin abzuleiten, ist eine Beschädigung der empfindlichen Bauteile wahrscheinlich.

Um empfindliche Bauteile vor Schäden durch elektrostatische Entladung zu schützen, können folgende Maßnahmen ergriffen werden, nach [16]:

- Aufnahme der Energie durch Kapazitäten
- Ableitung durch Schutzdioden (TVS, Zener, Schottky)
- Ableitung durch Varistoren

Multilayervaristoren bieten den besten Schutz vor Beschädigung durch elektrostatische Entladung. Sie ermöglichen hohe Überlebenswahrscheinlichkeiten auch bei Entladungsspannungen bis zu 25 kV, nach [16, Abb. 1].

Varistoren sind nichtlineare spannungsabhängige Widerstände aus halbleitender Siliziumkarbid- (SiC) oder Zinkoxid- (ZnO) Keramik. Bei geringen Spannungen verhindern Potenzialbarrieren an den Korngrenzen einen Stromfluss, das Material ist elektrisch isolierend. Wird die Spannung erhöht, kann bei einem bestimmten Spannungsabfall pro Korngrenze die Potenzialbarriere von den Elektronen überwunden werden, der Strom steigt mit $I = \pm K \cdot |U|^{a_{var}}$, wobei die Nichtlinearitätskoeffizienten a_{var} bei SiC zwischen 5 und 7 und bei ZnO zwischen 30 und 70 liegen, nach [15, Kap. 5.3]. Hierbei ist der Faktor K eine von den geometrischen Bedingungen abhängige Konstante.

Tritt nun eine Spannungsüberhöhung am Sensor, beispielsweise durch elektrostatische Entladung auf, begrenzt der Varistor die Spannung auf die sogenannte Klemmenspannung.

3.5 Echtzeitsysteme

Um als echtzeitfähiges System zu gelten, muss es nach DIN 44300 folgender Formulierung entsprechen, nach [26, Kap. 3]:

„Unter Echtzeit versteht man den Betrieb eines Rechensystems, bei dem Programme zur Verarbeitung anfallender Daten ständig betriebsbereit sind, derart, dass die Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind. Die Daten können je nach Anwendungsfall nach einer zeitlich zufälligen Verteilung oder zu vorherbestimmten Zeitpunkten anfallen.“

Somit ist bei der Verarbeitung der Ereignisse nicht nur ein korrektes Ergebnis, sondern auch die Einhaltung der Verarbeitungszeit bis das Ergebnis vorliegt relevant. Hierzu müssen Zeitschranken definiert werden, welche dem Anwendungsfall entsprechend eine ausreichende Funktion gewährleisten (nicht zwingend schneller Ablauf).

Die folgende Abbildung zeigt eine typische Ereignisverarbeitung, nach [9, Kap. 1]:

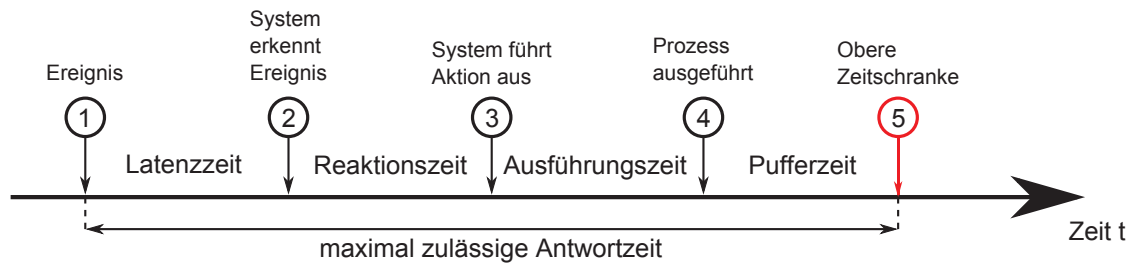


Abbildung 3.6: Bearbeitung eines Ereignisses im Echtzeitsystem

Zudem wird unterschieden zwischen:

- Harter Echtzeitbedingung, bei der beim Überschreiten der oberen Zeitschranke die Information mit der Zerstörung des Systems oder massiver Fehlfunktion bewertet wird und somit in jedem Fall verhindert werden muss.
- Weicher Echtzeitbedingung, bei der beim Überschreiten der oberen Zeitschranke, abhängig von der Dauer der Überschreitung die Information zunehmend an Wert verliert.

4 Weiterentwicklung und Realisierung der Elektronik

Dieses Kapitel beschreibt die Überarbeitung der Auswerteelektronik für polymerbasierte taktile Sensoren, um sie mit den in Abschnitt 1.1 beschriebenen Funktionen zu erweitern.

Die Entwicklung der Elektronik erfolgt in zwei Stufen, hierzu wird im ersten Schritt die Schaltung und eine Leiterplatte entworfen, mit allen Funktionen für die Entwicklung der Software. Ist die Funktion der Schaltung und Software sichergestellt, wird im zweiten Schritt diese Schaltung von den Evaluierungsanschlüssen befreit und mit minimaler Baugröße eine neue Leiterplatte entworfen.

4.1 Redesign der Schaltung

Die Elektronik soll weiterhin wie, nach [24, Kap. 5.2, 5.4.1], dezentral verteilt bleiben und sequenziell die einzelnen Taxel auswerten, jedoch erweitert um die Funktion, einzelne Taxel miteinander zu verschalten und somit verschiedene Auflösungen zu ermöglichen, nach [28].

Jedoch ist die Schaltung wie folgt geändert, um die Funktion zu erweitern oder zu verbessern:

- Austausch des Mikrocontrollers Atmega64M1 durch AT90CAN128
- Austausch der Zeilen-, Spaltenmultiplexer durch seriell gesteuerte Analogschalter
- Austausch der Operationsverstärker OPA2380 durch OPA354 (Rail-to-Rail Input/Output)
- Erweiterung der Messkette um Digitalpotentiometer
- Erweiterung der Schaltung um ESD (electrostatic discharge) - Schutz
- Erweiterung mit Dreiachsbeschleunigungssensor BMA250
- Erweiterung mit barometrischem Drucksensor BMP180
- Erweiterung mit Mikrofon zur Messung von Oberflächenrauigkeiten (Versuch)
- Erweiterung mit I²C-EEPROM (Electrically Erasable Programmable Read-Only Memory), um Kalibrierungswerte sensorabhängig zu speichern

Die oben genannten Änderungen der Auswerteelektronik ermöglichen mehr Flexibilität bei der Auswertung der taktilen Elemente, eine zuverlässige Bestimmung der Kalibrierungszeitpunkte durch erweiterte Sensormodalitäten, eine Verbesserung der Messwerte in den Bereichen der Vollausschläge, sowie eine robustere Schaltung durch ESD-Schutz.

4.1.1 Auswahl der Bauteile

Der Austausch des ursprünglichen Mikrocontrollers, den Atmega64M1, erfolgt durch den aus der selben Mikrocontrollerfamilie stammenden AT90CAN128. Hierbei kann die bereits eingerichtete Entwicklungsumgebung und der Hardwaredebugger weiter verwendet werden.

Zudem bietet der Wechsel auf den AT90CAN128 eine serielle Schnittstelle nach I²C- / TWI-Standard (Two Wire Interface), um ein auf dem taktilen Sensor untergebrachtes EEPROM auszulesen, in dem Kalibrierungs- und Sensorinformationen gespeichert werden können. Zudem können über diese Schnittstelle gleichzeitig andere Komponenten angeschlossen werden.

Erweiterten Funktionsumfang bietet der AT90CAN128 auch in der Zahl speicherbarer CAN-MOBS (Message Objects) (15 statt 6 beim Atmega64M1), der Anzahl der Ein-/Ausgangspins (53 statt 27) und der für die Softwareentwicklung wichtige Debugschnittstelle JTAG (Joint Test Action Group) gemäß IEEE-Standard 1149.1, nach [3], [4].

Ein Wechsel von Multiplexern auf seriell gesteuerte Analogschalter ermöglicht es die Zeilen- und Spaltenauswahl der taktilen Sensorbahnen beliebig zu verschalten, um so alle taktilen Elemente des Sensors zu einem Element zu verbinden. Es ist damit möglich ein Kontaktereignis zu erfassen, in dem nur noch ein Taxel ausgewertet werden muss und somit kann auf das reine Kontaktereignis schneller reagiert werden. Auch lässt sich der Sensor in einzelne Quadranten teilen und auswerten. Dies ermöglicht die sequenzielle Erhöhung des Informationsgehaltes, nach [28].

Als Analogschalter wurden Bausteine vom Hersteller Analog Devices, des Typs ADG1414 gewählt. Sie lassen sich über das SPI des Mikrocontrollers steuern, weisen einen Durchgangswiderstand von nur ca. 16 Ω ($V_{DD} = +12\text{ V}$, $V_{SS} = 0\text{ V}$) im geschalteten Zustand auf und beanspruchen nur 4 mm \times 4 mm Platinenfläche, nach [1].

Die Abbildung 4.1 zeigt die erweiterten Schaltmöglichkeiten.

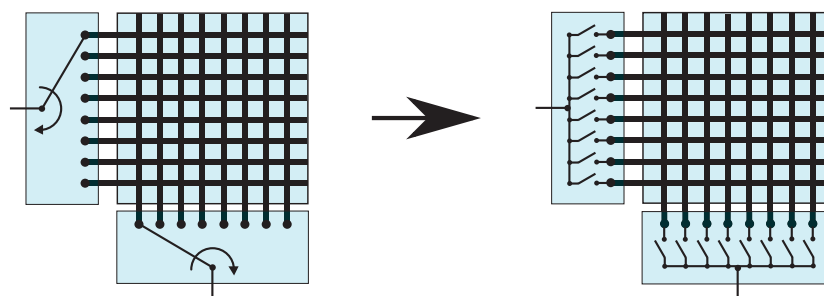


Abbildung 4.1: Schaltbild des Sensors mit Multiplexer und Analogschaltern

Ein zuverlässiges Bestimmen eines geeigneten Zeitpunktes, um einen Nullpunktgleich durchzuführen, wird durch die Erweiterung mit einem Dreiachsbeschleunigungssensor (BMA250) und einem Luftdrucksensor (BMP180) ermöglicht.

Der Nullpunktgleich kann dann durchgeführt werden, wenn der Relativdruck in einer abgeschlossenen Kavität im Sensor, in Bezug auf die Umgebung, gleich Null ist und folglich der Sensor damit unbelastet ist. Desweiteren kann die Information vom Beschleunigungssensor genutzt werden, um zu bestimmen, ob eine Belastung des Sensors durch die eigene Gewichtskraft vorliegt und dies mit in den Abgleich einzubeziehen ist.

Im Rahmen des Redesigns der Schaltung ist ein MEMS Mikrofon integriert worden. Mit diesem können Versuche zur Bestimmung der Oberflächenrauigkeit von Objekten durchgeführt werden. Die nachfolgende Prinzipskizze in Abb. 4.2 beschreibt den Versuchsaufbau zur Messung von Oberflächenschwingungen am taktilen Sensor.



Abbildung 4.2: Versuchsaufbau zur Messung von Oberflächenrauigkeiten

Ein auf dem Anschluss des taktilen Sensors untergebrachtes I²C-EEPROM speichert in der neuen Schaltung die Sensoridentifikationsnummer, Herstellungsdatum sowie die Basiskonfiguration und grundlegende Kalibrierungswerte für den Sensor.

Die eingespeicherten Kalibrierungswerte können von einem Prüfstand für taktile Sensoren nach [17] bezogen werden.

Mit den Informationen der Sensor-ID und dem Herstellungsdatum lassen sich Versuche mit den Sensoren leichter verwalten. So können beispielsweise die initialen Kennwerte der taktilen Sensoren im Speicher abgelegt werden, um eventuelle Alterungsprozesse zu untersuchen.

4.1.2 Energieversorgung

Die in der Auswertelektronik benötigten Spannungen von 5 V und 3,3 V werden aus der bereitgestellten Versorgungsspannung von 12 V erzeugt. Dies wird mittels Low-Drop-Out (kurz LDO) Linearreglern realisiert. Diese werden verwendet, da sie im Gegensatz zu Schaltreglern keine Störabstrahlung aufweisen und somit nicht die in der Nähe liegenden analogen Schaltungsteile negativ beeinflussen.

Als nachteilig erweist sich jedoch hierbei, dass Linearregler, auch Verlustregler genannt, die überschüssige Energie in Wärme umsetzen. Somit müssen die Spannungsregler für die Schaltung ausgelegt werden, um möglichst wenig thermische Energie ins System einzubringen.

Abbildung 4.3 zeigt schematisch die Spannungsregelung:

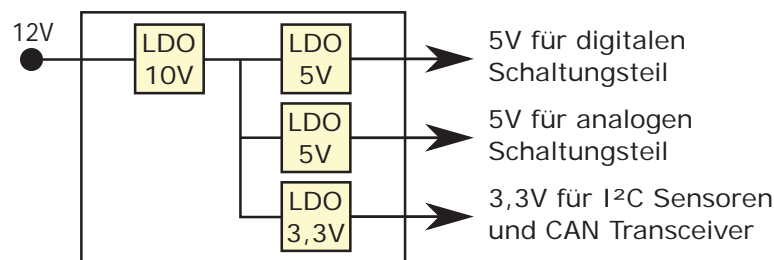


Abbildung 4.3: Regelung der im System benötigten Spannungen

Die Versorgungsspannung für den analogen und den digitalen Teil der Schaltung sind getrennt geregelt, um eine Beeinflussung der Versorgungsspannung für die analogen Komponenten zu vermeiden.

Um sicherzustellen, dass die verwendeten Spannungsregler nicht überlastet werden, darf die maximale Verlustleistung nicht überschritten werden. Dies hätte zur Folge, dass der Regler abschaltet (thermische Schutzschaltung integriert), um eine thermische Überlastung zu vermeiden. Die Schaltung würde nicht mehr mit elektrischer Energie versorgt werden.

Für die Auslegung der Spannungsversorgung wird die Stromaufnahme des Systems laut den Datenblättern der Bauelemente betrachtet:

Anzahl	Bauelement	Betriebsspannung	typ. Stromaufnahme	max. Stromaufnahme
1	MAX3051	3,3 V	35 mA	75 mA
1	BMA250	3,3 V	0,1 mA	0,1 mA
1	BMP180	3,3 V	0,6 mA	1,0 mA
1	SPU0409LE5H	3,3 V	0,25 mA	0,25 mA
1	AT90CAN128	5 V (D)*	29 mA	29 mA
1	MAX232	5 V (D)	15 μ A	15 μ A
2	LED	5 V (D)	1 mA	1 mA
1	ADS8319	5 V (A)*	5 mA	5 mA
5	OPA354	5 V (A)	7,5 mA	7,5 mA
1	ISL23425	5 V (A)	4 mA	4 mA
2	ADG1414	12 V	1 mA	1 mA

* (D) Spannungsregler für digitalen Schaltungsteil, (A) für analogen Schaltungsteil

Tabelle 4.1: Stromaufnahme der verwendeten Bauelemente

Somit ergibt sich für die Summe der Ströme für die Regler:

3,3 V: typ. 36,0 mA, max. 76,0 mA

5 V (D): typ. 35,0 mA, max. 35,0 mA

5 V (A): typ. 41,5 mA, max. 41,5 mA

12 V: typ. 2,0 mA, max 2,0 mA

Die im Regler umgesetzte Verlustleistung berechnet sich nach:

$$P_D = (U_{in} - U_{out}) \cdot I \quad (4.1)$$

Der Gesamtstrom der Schaltung (bis auf die Versorgung der ADG1414) fließt erst durch den 10 V Regler. Die Spannungsdifferenz zum Eingang beträgt für diesen Regler 2 V.

Es ergibt sich eine Verlustleistung am Regler für die typischen Ströme der Bauelemente von:

$$P_{typ. 10 V} = 2 V \cdot 112,5 mA = 225,0 mW$$

und im ungünstigsten Fall von:

$$P_{max. 10 V} = 2 V \cdot 152,5 mA = 305,0 mW$$

Analog hierzu die Berechnungen für die restlichen Regler:

$$P_{typ./max. 5 V(D)} = 5 V \cdot 35,0 mA = 175,0 mW$$

$$P_{typ./max. 5 V(A)} = 5 V \cdot 35,0 mA = 207,5 mW$$

$$P_{typ. 3,3 V} = 6,7 V \cdot 36,0 mA = 241,2 mW$$

$$P_{max. 3,3 V} = 6,7 V \cdot 76,0 mA = 509,2 mW$$

Die maximale Verlustleistung der Low-Drop-Out Regler vom Typ LP2985 lässt nach folgender Formel berechnen:

$$P_{Dmax.} = \frac{\vartheta_{Jmax} - \vartheta_{Amax}}{R\theta_{JA}} \quad (4.2)$$

Mit den Daten aus [31] gilt für den Spannungsregler,

ϑ_{Jmax} der maximalen Temperatur der Halbleitergrenzschicht von 125 °C kontinuierlich

ϑ_{Amax} der maximalen Umgebungstemperatur von 50 °C und

$R\theta_{JA}$ dem thermischen Übergangswiderstand von Grenzschicht zur Umgebung von 206 °C/W,

ergibt sich eine maximal zulässige Verlustleistung von $P_{Dmax} = 364,1 mW$ je Spannungsregler.

Diese wird von der maximalen Verlustleistung des 3,3 V Reglers im ungünstigsten Fall überschritten. Jedoch mit der typischen Stromaufnahme ist die Verwendung des Reglers möglich. Somit müssen die CAN-Bus Transceiver entsprechend selektiert werden.

Die im Schaltplan verwendeten Regler vom Typ MCP1804 werden aufgrund ihrer geringeren maximalen Verlustleistung von 234 mW bei 25 °C Umgebungstemperatur nur für Testzwecke verwendet, da die Regler vom Typ LP2985 in den passenden Spannungen zum Fertigungsdatum nicht verfügbar waren. Sie werden für die Zielhardware ersetzt und sind durch ihre Pinkompatibilität auch auf den Evaluierungsplatinen ersetzbar.

Als tatsächlich für den Betrieb notwendigen Ströme wurden ermittelt:

Platinenr.	12 V	10 V	5 V(D)	5 V(A)	3,3 V
2	65,0 mA	60,0 mA	35,0 mA	17,1 mA	7,7 mA
3	64,6 mA	60,6 mA	36,1 mA	15,5 mA	6,5 mA
4	64,4 mA	61,0 mA	36,5 mA	16,3 mA	6,5 mA

Messgerät Fluke 45, Messbereich 100mA, Auflösung 1 μA , Slow, 0,05% + 5 Digits, Raumtemperatur: 24° C
Drei Auswerteelektroniken am Bus, 1 Mbit/s, Abschlusswiderstände 120 Ω

Tabelle 4.2: Stromaufnahme der Schaltung gemessen im Betrieb

Wie die Messung zeigt, ergibt sich gerade für den verlustleistungskritischsten Spannungsregler, den 3,3 V Regler, dass der im Datenblatt angegebene benötigte Strom des CAN-Transceivers, von 35 mA - 75 mA im realen Betrieb nicht erreicht wird (unabhängig von der Zahl der Bus Teilnehmer, der Größe der Abschlusswiderstände und der Datenrate).

4.1.3 Anpassung der Messketten

Um auf der erweiterten Auswerteeinheit eine breitere Variation von verschiedenen Sensortypen und Materialien verwenden zu können, wurde die bestehende Messkette um ein Digitalpotentiometer (mit Mikrocontroller veränderbarer Widerstandswert) erweitert. Somit ist es nun möglich taktile Sensoren mit großen Unterschieden ihrer taktilen Übergangswiderstände im vollen Messbereich auszuwerten.

Dies ermöglicht es, bei entsprechender Kalibrierung, für jeden Sensor den passenden Messwiderstand zu wählen und somit den Messbereich bestmöglich zu nutzen.

Die für die Zielhardware verwendete Messkette ist wie folgt aufgebaut:

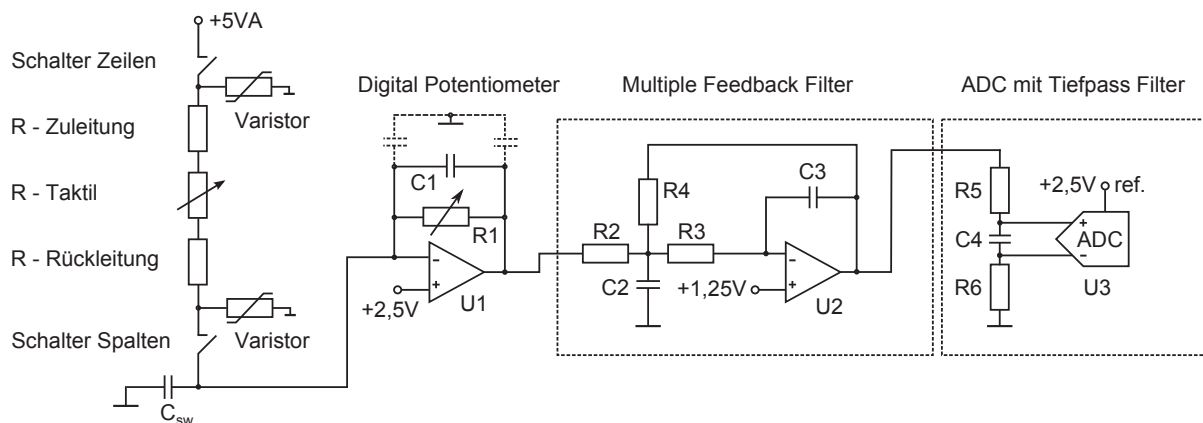


Abbildung 4.4: Messkette

Die grundlegenden Änderungen der Messkette im Vergleich zu der nach [24, Kap. 6.2.2] sind wie folgt:

- Optionale Verschaltung aller taktiler Elemente zu einem Element
- Variabler Widerstand (R1) am Transimpedanzwandler, somit variabler Messbereich
- Multiple Feedback Filter mit höherer Grenzfrequenz, wegen höherer Auswertegeschwindigkeit
- Übertragung in den Grenzbereichen der Spannungsversorgung wird verbessert, da Operationsverstärker mit Rail-to-Rail In-/Output verwendet werden
- Erzeugung der 2,5 V, 1,25 V und 2,5 V ADC-Spannungsreferenz ratiometrisch abhängig von +5 VA
- Schutz vor elektrostatischen Entladungen durch Varistoren

Um sicher zu stellen, dass die Funktion der Schaltung gewährleistet ist, wird die neu ausgelegte Messkette anschließend in einem Simulationsmodell überprüft.

Die parasitäre Kapazität C_{sw} in Abbildung 4.4 setzt sich aus den Kapazitäten der folgenden Bauteile zusammen.

- ADG1414 (Single Supply 12V): C_D, C_S (On) = 33 pF
- OPA354 Eingangsimpedanz: $10^{13} \Omega \parallel 2 \text{ pF}$
- 41206ESDA Varistor: 0.15 - 1 pF

Die Gesamtkapazität der Bauteile nach [1], [32], [7] ergibt 36 pF.

Um die 64 Taxel mit 8 Bit Messwertaufösung auszuwerten, wird aufgrund der maximalen Übertragungsrates des Bussystems eine Auswerterate von 668 Hz gewählt. Hierzu wird die maximale Buslast von etwa 70% der 1 Mbit/s Übertragungsleistung des Bussystems durch die Anzahl der je acht Messwerte umfassenden Nachrichten dividiert, die aus 131 Bit bestehen.

$$f_{max.} = \frac{0.7 \cdot 1 \cdot 10^6 \text{ Bit/s}}{\frac{64}{8} \cdot 131 \text{ Bit}} = 668 \frac{1}{s} \quad (4.3)$$

Die 668 Auslesevorgänge pro Sekunde benötigen $668 \cdot 64 = 42752$ Schaltvorgänge pro Sekunde. Der Zeitpunkt, nachdem das Signal spätestens abgetastet werden muss, liegt bei $\leq 23,4 \mu s$ (Kehrwert der Schaltvorgänge pro Sekunde), um eine konstante Auswerterate von 668 Hz zu ermöglichen. Um zeitnah auf Abfragen zu beliebigen Zeitpunkten zu reagieren, ist jedoch eine kürzere Einschwingzeit von Vorteil (angestrebter Wert $\leq 5 \mu s$), um eine schnelle Antwort zu ermöglichen.

Die Neuauslegung des aus dem Tiefpassfilters 1. Ordnung und dem nachgeschalteten mehrfach-gegekoppelten Filters 2. Ordnung wurde mit der Filterauslegungssoftware Filter Pro durchgeführt, die vom Hersteller der Operationsverstärker Texas Instruments stammt. Dies bietet den Vorteil, dass die für die Filter verwendeten Widerstands- und Kapazitätsgrößen aufeinander abgestimmt werden und zudem kann überprüft werden, ob das Verstärkungsbandbreitenprodukt der Operationsverstärker ausreichend ist.

Die gewählten Filterparameter des Tiefpassfilters, gesamt 3. Ordnung, für die Auslegung sind ein Durchlassband bis 900 kHz (das 20-fache der Schaltfrequenz) bei einer Welligkeit von maximal 1 dB. Der Filter ist als zweistufiger Filter, mit Besselfiltercharakteristik realisiert.

Hieraus resultieren die Werte für die verwendeten Bauelemente,

$$C_1 = 2,7 \text{ pF}, R_2 = 33,2 \text{ k}\Omega, R_3 = 16,5 \text{ k}\Omega, R_4 = 33,2 \text{ k}\Omega, C_2 = 10 \text{ pF}, C_3 = 2,7 \text{ pF}$$

die für den Tiefpass 1. Ordnung eine minimale Grenzfrequenz (f_0) von 1,19 MHz und den Tiefpass 2. Ordnung von konstant 1,30 MHz liefern. Da im Tiefpass 1. Ordnung der Rückkopplungswiderstand variabel ist, ergibt sich in einem sinnvoll nutzbaren Bereich von $10 \text{ k}\Omega$ bis $50 \text{ k}\Omega$ eine Grenzfrequenz von 1,19 MHz bis 5,89 MHz. Bei niedrigem Rückkopplungswiderstand gewinnt der Filter 2. Ordnung mehr an Gewichtung und begrenzt das Frequenzband weiterhin in der Nähe der angestrebten 900 kHz.

Die Widerstände R_5 und R_6 bilden zusammen mit dem Kondensator C_4 einen zusätzlichen Tiefpass 1. Ordnung mit einer Grenzfrequenz von 15,9 MHz der durch das Datenblatt des A/D-Wandlers [30] vorgegeben wird (Eingangsbeschaltung) und vernachlässigbar ist.

Das für die Filter benötigte Verstärkungsbandbreitenprodukt der Operationsverstärker beträgt für den Tiefpass 1. Ordnung 59,7 MHz und den Tiefpass 2. Ordnung 90,2 MHz, was von den Operationsverstärkern vom Typ OPA354 mit 250 MHz deutlich übertroffen wird.

Abschließend werden die neuausgelegten Filter in die Simulationssoftware TINA (von Texas Instruments) übertragen, um in der Simulation zu überprüfen, ob das gewünschte Einschwingverhalten der Filter bei Schaltvorgängen gewährleistet ist. Hierzu wird der Schaltvorgang durchgeführt und geprüft, ob das Einschwingen des Signals bei der gewünschten Schaltfrequenz noch gegeben ist.

Das erstellte Simulationsmodell übertragen in die Simulationssoftware TINA:

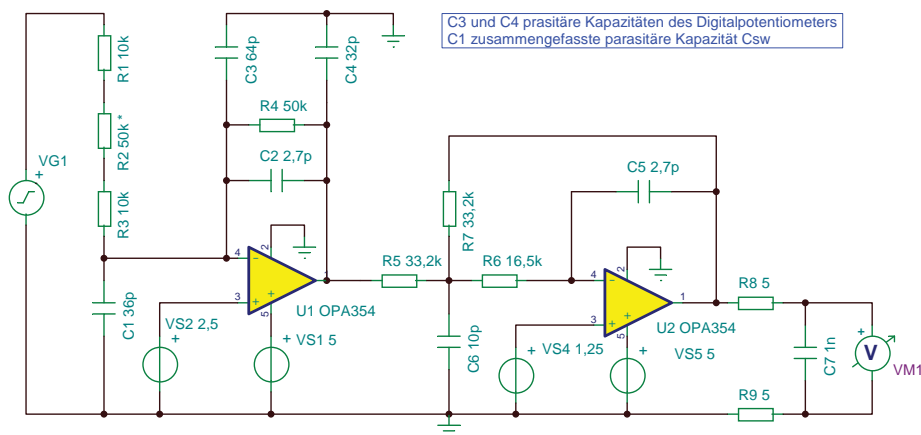


Abbildung 4.5: Simulationsmodell der Messkette

Abbildung 4.6 zeigt das Einschwingverhalten der Messkette für Widerstandswerte des Sensors im Bereich von $50k\Omega$ bis $1M\Omega$ in Abstufungen von $50k\Omega$. Der Pfeil (a) markiert den Zeitpunkt ($1,6\mu s$), ab dem die Signalschwankung kleiner als 10 mV (1 Diskretisierungsschritt bei 8 Bit) ist und der Messwert abgetastet werden darf.

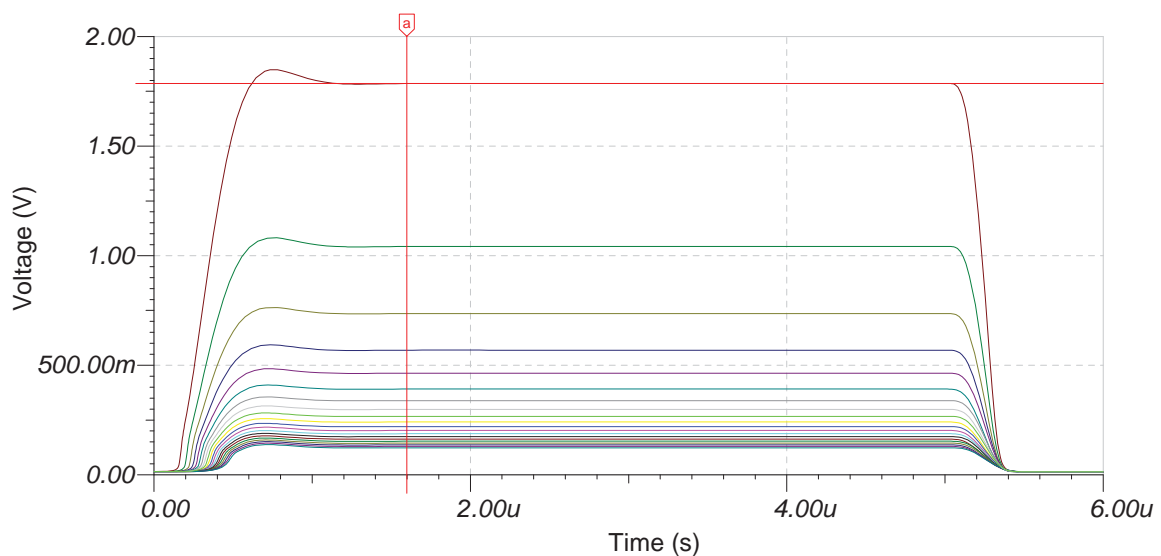


Abbildung 4.6: Sprungantwort für Widerstände von $50k\Omega$ bis $1M\Omega$

4.1.4 Implementierung des Schutzes vor elektrostatischen Entladungen

Die Schaltung ist in der Weiterentwicklung vor elektrostatischen Entladungen durch Varistoren geschützt, welche als Varistorarray realisiert sind, um Platz zu sparen. Die Varistoren sind parallel zu jeder Zeile und Spalte des taktilen Sensors in Richtung Masse der Schaltung eingesetzt (siehe Schaltplan Abb. A.2 Quadrant links oben im Anhang).

Wird eine elektrostatische Ladung über den taktilen Sensor abgeleitet, können die Varistoren die nachfolgenden Bauteile gegen die Spannungsüberhöhung schützen.

Für den ESD-Schutz sind Varistorarrays vom Hersteller Cooper Bussmann, Typ 41206ESDA verbaut. Diese begrenzen die Spannung auf den Zeilen- und Spaltenleitungen auf eine Klemmenspannung von 35 V (typisch) in Bezug auf die Systemmasse. Desweiteren wird die Messleitung durch die 0.15 pF Kapazität des Varistors nur minimal kapazitiv belastet, Daten nach [7].

4.2 Design der Leiterplatte

Der Designprozess der Leiterplatte erfolgt zweistufig. Hierbei wird zuerst eine Entwicklungsplatine aufgebaut. Sie dient zur Evaluierung der verwendeten Bauteile und zum Entwickeln der benötigten Mikrocontrollerfirmware.

Die Evaluierungshardware verfügt zusätzlich über die folgenden Schnittstellen:

- JTAG Schnittstelle zum Hardwaredebuggen
- Programmierbare Ein-/Ausgangsleitungen
- UART Schnittstelle mit Pegelwandler (einfache Kommunikation zum PC)
- Anzeigeleuchtdioden (programmierbar)
- Messpunkte für:
 - Analoge Signale der Messkette
 - Messbrücken zur Strommessung
 - SPI-Bus
 - I²C-Bus

In der zweiten Stufe werden die aus der Evaluierungsplatine gewonnenen Kenntnisse verwendet, um eine Auswerteelektronik mit minimalen Abmaßen zu entwickeln. Durch die vorherige Prüfung der Schaltung in der Evaluierungsphase müssen auf der Zielhardware nur noch die für die Funktion unbedingt notwendigen Anschlüsse vorhanden sein. Durch das Entfallen der Messpunkte und Debugschnittstellen kann auf einen Großteil der Platinenfläche in der Zielhardware verzichtet werden.

4.2.1 Aufbaukonzept

Der Aufbau der Evaluierungsplatine unterscheidet sich grundlegend von der Zielhardware, welche für den späteren Einsatz in mechatronischen Systemen vorgesehen ist.

So ist die Platine für die Evaluierung rein für Entwicklungszwecke konzipiert und verfügt über die in 4.2 erwähnten Anschlüsse. Hierzu wird eine vergleichsweise große Platinenfläche benötigt. Daraus resultiert eine Gesamtabmessungen von 75 mm x 75 mm.

Beim Entwurf der Evaluierungsplatine werden jedoch schon die Bauelemente in ihrer für die Zielhardware passenden Gehäusegröße verwendet, um ihre Montierbarkeit zu prüfen.

Sollten zudem Fehler in der Verdrahtung der Bauteile auftreten, ist es bei den in der Evaluierungsplatine verwendeten, außenliegenden Verdrahtungsebenen leichter möglich diese zu beheben, in Vergleich zu den teils innenliegenden Verdrahtungsebenen bei Multilayerplatinen mit hohen Lagenzahlen.

Die folgende Abbildung zeigt schematisch das Konzept der Hardware:

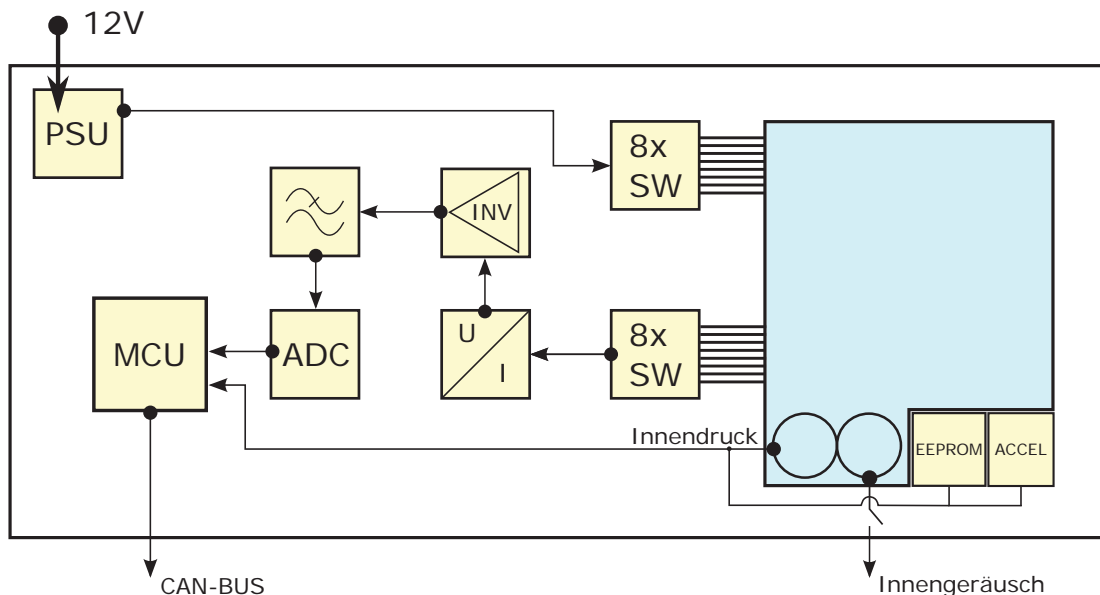


Abbildung 4.7: Hardwarekonzept nach Weiterentwicklung

Nach diesem Konzept wurde die Evaluierungsplatine entworfen und gefertigt. Der Schaltplan sowie das Platinenlayout wurde mit der Software PADS[®] vom Hersteller MentorGraphics[®] erstellt.

Der Schaltplan und die Fertigungsdaten der Evaluierungsplatine sind im Anhang in den Abschnitten A und B einsehbar.

Für die Zielhardware, die auf einem Roboter zum Einsatz kommen wird, ist es noch notwendig, dass Abmaß der Elektronik auf ein Minimum zu reduzieren. Dieser Prozess ist jedoch mit relativ geringem Aufwand verbunden, da bereits beim Erstellen des Schaltplans für die Evaluierungshardware alle Bauelemente in ihrer endgültigen Bauform und Logik gewählt wurden.

Daraus resultiert, dass lediglich die Platine neu erstellt werden muss, was bei dem verwendeten Layoutprogramm mit geringem Aufwand realisierbar ist. Somit kann in diesem Schritt mehr Aufmerksamkeit auf die mechanische Integration und eine leicht montierbare Verkabelung gelegt werden.

Die Maße der finalen Elektronik wurden abgeschätzt, in dem die in der Evaluierungshardware von allen für die Funktion irrelevanten Komponenten befreit wurde. Anschließend wurden die benötigten Bauteile flächen- und funktionsoptimiert neu angeordnet. Aus diesem Versuch folgt als Ausgangsgröße für die finale Auswerteelektronik eine Größe von 22 mm x 22 mm.

Die Abbildung 4.8 (links) zeigt dreidimensional, die auf beiden Seiten der Platine verteilten Bauelemente (schwarze Quader). Als Anschlüsse an den taktilen Sensor sowie an den CAN-Bus sind flexible Leiterplatten vorgesehen, vgl. Abb. 2.4.

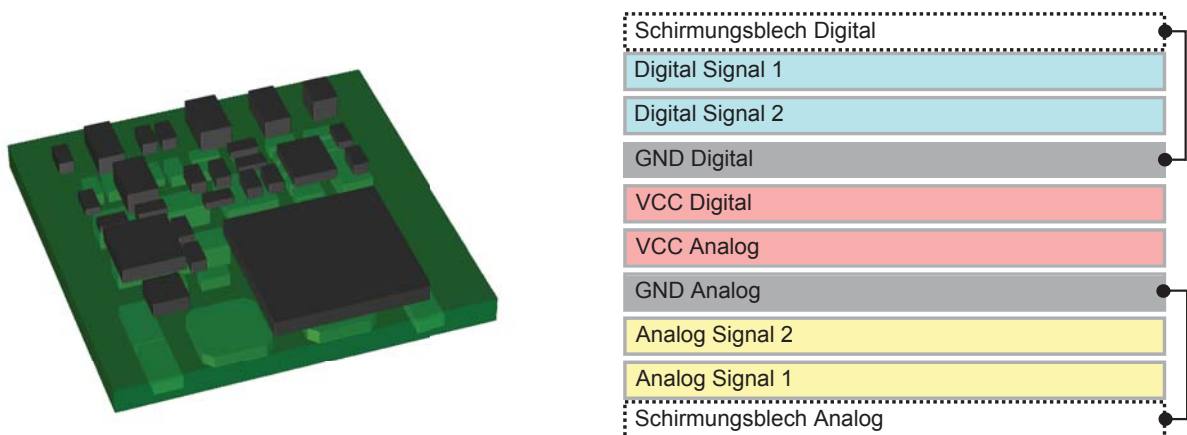


Abbildung 4.8: Zielhardware und Leiterplattenlagenaufbau

4.2.2 Layeraufbau

Das Layout der Evaluierungsplatine ist auf einem vierlagigen Aufbau realisiert. Beidseitig sind die zwei äußeren Lagen der Platine als Verdrahtungsebenen angelegt. Die inneren als sogenannte Versorgungs- und Massefläche, welche die Bauelemente mit der benötigten Spannung und Masse versorgen. Zudem ist die Versorgungsfläche räumlich in einen analogen und digitalen Bereich geteilt.

Für das Layout der Zielhardware sind mindestens sechs Ebenen notwendig, da diese durch den geringen Bauraum eine höhere Verdrahtungsdichte aufweist. Die genaue Zahl der benötigten Layer wird jedoch erst beim Layout festgelegt. Die Abbildung 4.8 (rechts) zeigt den für die Zielhardware geplanten Lagenaufbau im Querschnitt.

Nach [27, Tab. 11.1b] bietet dieser Aufbau gute Leistungswerte in den Bereichen Entkopplung (Flächenkondensator zwischen Versorgungsspannung und Masse), Intra-EMV (Schirmung der Signalleitungen gegenüber der Versorgungsspannung, kurz intra) (elektromagnetische Verträglichkeit, kurz EMV). Durch übereinander angeordnete, doppelte Signallagen werden jedoch die Signalintegrität (Beeinflussung der Signalleitungen untereinander) und die Inter-EMV (Emissivverhalten der Leiterplatte) gemindert. Um die Inter-EMV zu erhöhen, ist daher eine gekapselte Schirmung über die beiden Außenlagen vorgesehen.

4.2.3 Implementierung der Schirmung

Zur Schirmung der in 3.3 beschriebenen Störungen durch elektrische und elektromagnetische Felder wird eine Schirmung des analogen Schaltungsteils vorgesehen.

Die Auswerteelektronik muss im späteren Verlauf im Zusammenspiel mit anderen elektronischen Komponenten funktionieren. Hierzu ist es wichtig, die von außen eingekoppelten Störungen zu minimieren. Beim Einsatz auf einem Roboter ist z.B. mit Störabstrahlungen von Motorreglern, Steuerleitungen bzw. Netzteilen zu rechnen.

Um die Störungen in der Auswerteeinheit möglichst gering zu halten, sind oberflächenmontierbare Halter auf die Platine aufgebracht, an denen Schirmungsbleche (Weißblech nach DIN EN 10202 / 10203) befestigt werden. Die Bleche sind abnehmbar, um eventuelle Eingriffe in die Elektronik zu ermöglichen.

Die minimale Schirmwirkung der Hochfrequenzschirme vom Hersteller Würth Elektronik liegt bei 40 dB (Schwächung um Faktor 100) über einen Frequenzbereich von 500 MHz - 3 GHz, nach [34, Abschnitt D].

In Verbindung mit der Masselage bildet das Schirmungsblech einen „Faradayschen Käfig“ um den gekapselten Schaltungsteil (Abb. 4.8 (rechts) über den Außenlagen). Abb. 4.9 zeigt schematisch die gekapselte Zielhardware.

Sollte für spätere Anwendungen die Bauhöhe durch die Schirmungen überschritten werden, kann optional die Schirmung des unkritischeren digitalen Schaltungsteils entfernt werden. Hierbei sind jedoch Versuche im entsprechenden System durchzuführen, um eine fehlerfreie Funktion abzusichern.

Um die Störeinkopplungen zu reduzieren, müssen auch die Zuleitungen zum Sensor geschirmt werden. Dies ist mit einer dreilagigen Flexplatine realisierbar, bei der die beiden Außenlagen als Schirmung dienen.

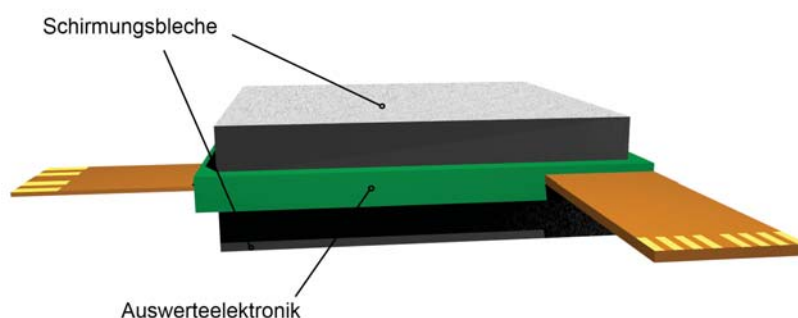


Abbildung 4.9: Schirmungskonzept der Zielhardware

Aufgrund von Lieferverzögerungen der Bauteile während der Fertigung der Evaluierungspaltine, konnte die verkleinerte Zielhardware im Rahmen dieser Arbeit nicht mehr als Prototyp realisiert werden.

5 Entwicklung der Software

Dieses Kapitel befasst sich mit der Weiterentwicklung der Software für die Auswertelektronik sowie der Software zur Weiterverarbeitung der gewonnenen Daten für die Verwendung auf einem Robotersystem.

Die in der Auswertelektronik vorgenommenen Änderungen an der elektronischen Schaltung aus Kapitel 4 haben zur Folge, dass umfassende Änderungen an der Software (Firmware) des Mikrocontrollers vorgenommen werden mussten.

Der prinzipielle Ablauf hinsichtlich der Datenerfassung ist vergleichbar zu der des ersten Prototyps, jedoch müssen an der Software folgende Punkte geändert bzw. hinzugefügt werden, um mit der erweiterten Hardware als Sensorsystem zu funktionieren:

- Implementierung der neuen Zeilen- und Spaltensteuerung, sowie Teilung in 64 Taxel, Quadranten und ein gesamtes Taxel
- Überarbeitung der CAN Kommunikation zur Filterung von nicht relevanten Nachrichten
- Einführung eines Kommunikationsprotokolls zur Steuerung der Auswerteeinheiten
- Implementierung der zusätzlichen Sensoren (Druck und Beschleunigung)
- Kommunikation mit hinzugefügtem EEPROM und Digitalpotentiometer

5.1 Softwarekonzept

Die Abläufe der weniger rechenintensiven Datenerfassung wird auf den Mikrocontrollern und die komplexere Datenaufbereitung für z.B. Robotersysteme, auf einem Computer mit deutlich höherer Rechenleistung durchgeführt.

5.1.1 Systemkonzept und Datenkanalisierung

Die Abbildung 5.1 zeigt schematisch das Konzept des Sensorverbunds:

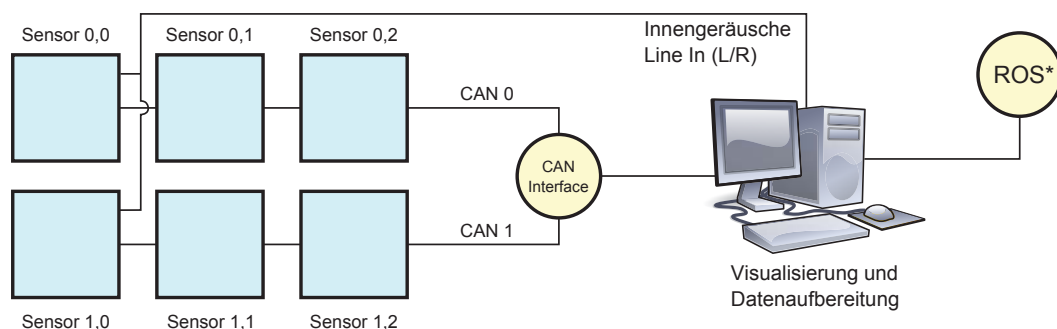


Abbildung 5.1: Konzept des Sensorverbundsystem

*ROS = Robot Operation System von Willow Garage

Um Versuche auf Robotersystemen durchführen zu können, ist die Zusammenfassung der von mehreren Sensoren erzeugten Daten (beispielsweise einer mit Sensoren ausgestatteten Roboterhand) notwendig. Diese Daten müssen aufbereitet (z.B. Erkennen von Druckpunkten oder Kanten) und für das Steuerungssystem zugänglich gemacht werden.

Der Computer dient neben der Datenverarbeitung auch als Visualisierungsrechner, um die im Betrieb erzeugten Daten grafisch darzustellen. Er wickelt die Kommunikation mit den einzelnen Auswerteelektroniken über ein dafür definiertes Steuerungsprotokoll ab, welches in Abschnitt 5.1.4 genauer beschrieben ist.

5.1.2 Softwaremodule der Mikrocontrollerfirmware

Die Abbildung 5.2 zeigt eine Übersicht der Softwaremodule der Mikrocontrollerfirmware:

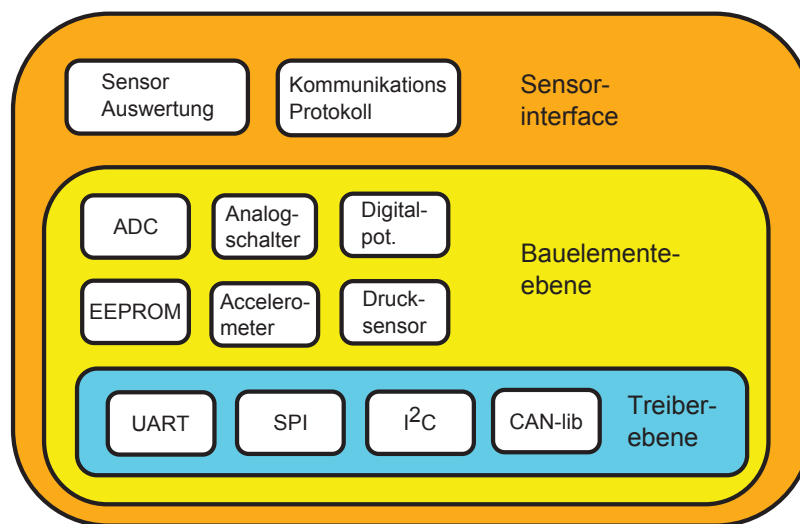


Abbildung 5.2: Modularer Aufbau der Software

Um den Quellcode für den Mikrocontroller übersichtlich und wartungsfreundlich zu gestalten, wurden zuerst die Funktionen für die Hardwarekommunikation auf der Treiber-ebene implementiert. Sie beinhaltet die Steuerungs-routinen für die Kommunikation mit den internen Schnittstellen für SPI, I²C, UART und CAN.

Zur Kommunikation mit dem CAN Bus wird die CAN-lib von Fabian Greif (Roboterclub Aachen e.V.) verwendet, welche unter der FreeBSD-Lizenz veröffentlicht ist. Sie abstrahiert die Steuerung des Mikrocontrollers von Register-ebene auf eine Befehls-ebene, in der Nachrichten definiert, gesendet und empfangen werden können. Die Bibliothek liegt als Quellcode vor und ist mit denen für das System passenden Parametern übersetzt worden.

Die darüber liegende Ebene steuert die einzelnen Bauelemente in der Auswerteelektronik durch Kommunikation mit den internen Bussystemen SPI und I²C. Hierbei wird die Steuerung der Bauelemente soweit ausgeführt, dass im weiteren Programmcode mit einfachen Befehlen Daten abgefragt und Konfigurationen (Analogschalter, Digitalpotentiometer) vorgenommen werden können.

Auf der Ebene des Sensorinterface wird die Auswertung des Sensors gesteuert. Diese ist auf Befehle abstrahiert, welche die Steuerung und Datenabfrage ermöglichen. Desweiteren ist hier das Kommunikationsprotokoll realisiert, welches die Kommunikation mit dem Computer zur Datenaufbereitung abwickelt. Diese beiden Schnittstellen ermöglichen eine Datenabfrage des Sensors, sowie dessen Steuerung ohne detaillierte Kenntnisse über die Hardware.

5.1.3 Ablaufdiagramme

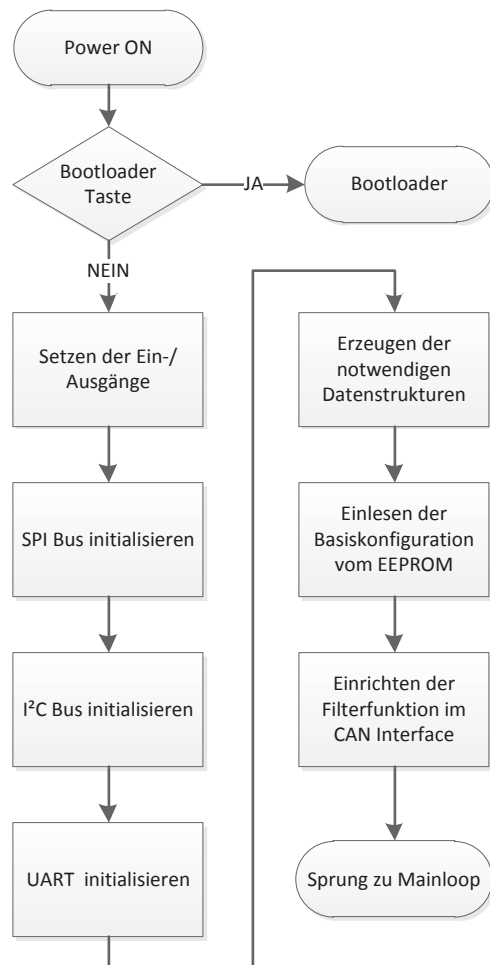


Abbildung 5.3: Ablauf des Systemstarts

Eine Initialisierung der UART Schnittstelle ist nur für die Evaluationshardware notwendig. Hierbei wird die häufig verwendete Einstellung 9600 Baud, 8 Datenbits, keine Parität, 1 Stopbit oder kurz 9600Bd 8N1 vorgenommen.

Im Anschluss an die Initialisierungen werden die für den Betrieb notwendigen Datenstrukturen, für CAN Nachrichten, CAN Filter und Sensordaten erzeugt.

Der Start des Ablaufdiagramms beginnt, indem die Schaltung mit ausreichender Betriebsspannung versorgt wird und der Oszillator der MCU betriebsbereit ist (hier als Power ON bezeichnet).

Zuerst erfolgt ein Sprung in den Anfangsbereich des Bootloaders, um mit dem Auslesen der Hardwaretaste für den Bootloader festzustellen, ob der Bootloaderbetrieb oder die Ausführung der Firmware gewünscht ist.

Die Funktion des Bootloaders wird in 5.2 genauer ausgeführt, für die weiteren Beschreibungen an dieser Stelle wird angenommen, dass die Ausführung der Firmware gewählt wurde.

Um die Auswerteelektronik in einen funktionsfähigen Grundzustand zu überführen, müssen zuerst grundlegende Register für die Steuerung der Ein-/Ausgabepins der MCU gesetzt werden. Hier wird definiert, ob der jeweilige Anschluss ein Eingang oder Ausgang ist.

Im weiteren Verlauf werden die Bussysteme SPI und I²C für die peripheren Bausteine initialisiert. Die Konfiguration für das SPI lässt wahlweise 4 MHz oder 8 MHz Taktfrequenz zu, während die MCU als Master auf dem Bus fungiert. Der Betriebsmodus ist vom Typ 0, wobei die Daten in der Mitte ihrer Übertragung, zu den steigenden Taktflanken, abgefragt werden.

Für den I²C Bus ist die Konfiguration auf eine Taktfrequenz von 400 kHz festgelegt und kann von allen Bausteinen am Bus verarbeitet werden.

Nun werden die sensorspezifischen Konfigurationsdaten aus dem EEPROM des Sensors geladen und in die Auswerteelektronik übernommen.

Den Abschluss des Systemstarts bildet die Konfiguration des CAN Interfaces, in dem die Einstellungen für die Filter gesetzt werden, um ausschließlich die für die Auswerteeinheit notwendigen Nachrichten zu empfangen.

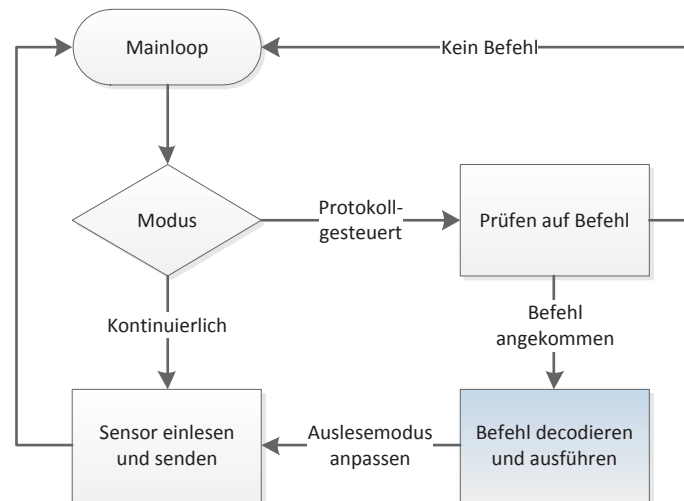


Abbildung 5.4: Hauptprogrammenschleife

Die Hauptprogrammenschleife wird nach der Initialisierung zyklisch ausgeführt und unterscheidet zwei prinzipielle Betriebsarten.

Zum Einen kann die Auswerteelektronik in einem Modus betrieben werden, in dem sie fortlaufend Sensordaten erzeugt und auf das Bussystem in bestimmten Zeitabständen sendet. Der Modus bedarf keiner weiteren Eingriffe des Datenaufbereitungsrechners.

Zum Anderen kann die Auswerteelektronik in einem Modus betrieben werden, in dem vom Datenaufbereiter Befehle geschickt werden (Polling- Betrieb), welche die angesprochene Elektronik dann umsetzt. Das Protokoll ist in Abschnitt 5.1.4 beschrieben.

Der kontinuierliche Modus bietet als Vorteil, dass der Kommunikationsoverhead für die Befehle vom Datenaufbereiter entfällt und somit effektiv mehr Daten von den Auswerteelektroniken gesendet werden können. Problematisch ist hierbei jedoch, dass bei hoher Buslast, Daten von Einheiten mit niedriger Priorität nicht mehr oder nur noch unregelmäßig übertragen werden können.

Der Pollingmodus bietet hingegen bessere Kontrolle über den Datenfluss und die Fähigkeit dynamisch einzelne Sensoren durch gezieltes Abfragen zu priorisieren. Dies erfolgt jedoch unter Verlust von Auslesegeschwindigkeit.

Durch implementieren beider Betriebsmodi können dem Anwendungsfall entsprechend die Vorteile beider Modi genutzt werden.

5.1.4 Kommunikationsprotokoll

Das Kommunikationsprotokoll definiert die Verständigung zwischen den Auswerteeinheiten und dem Datenaufbereiter. In diesem Protokoll ist die Bedeutung der Befehle vereinbart, welche Daten die Auswerteelektronik liefern muss, wenn sie mit einem bestimmten Befehl angesprochen wird. Über die Befehle dieses Protokoll kann auch steuernd in die Elektronik eingegriffen werden, wie zum Beispiel beim Anpassen der Messwiderstände oder der Auswertegeschwindigkeit.

Die globale Definition von Steuerungsbefehlen ermöglicht es, die Software des Mikrocontrollers getrennt von der des Datenaufbereitungsrechners zu entwickeln, sodass mehrere Entwickler gleichzeitig an der Software arbeiten können.

Eine Auflistung aller derzeit für die Kommunikation vorgesehenen Befehle ist im Anhang unter Abschnitt C einsehbar.

Nachrichtenaufbau allgemein:

Ein Telegramm im Protokoll besteht aus dem Identifier und dem Datenbereich. Der Identifier wird mehrfach genutzt, um die Priorität der Nachricht festzulegen (niedrigere Zahl ist höhere Priorität), zu übermitteln von welcher Einheit die Daten kommen mit der Sensor-ID und mit dem Modus in welchem Betriebsmodus sich die Einheit befindet.

Zudem wird mit „Start Taxel“ übertragen, an welcher Stelle sich der Startpunkt der Daten in der zu übermittelnden Datenmatrix befindet, wenn mehr als acht Taxel übertragen werden sollen. Zum Beispiel 8×8 Taxel werden in acht Telegrammen übertragen, somit beinhaltet bei der ersten Nachricht $\text{Start Taxel} = 0$, bei der zweiten Nachricht ist $\text{Start Taxel} = 8$ und die folgenden $n \cdot 8$.

Antwort einer Auswerteeinheit:

Sendet eine Auswerteeinheit Daten, dann übermittelt sie im Identifier ihre Priorität (größer null, höchste Priorität belegt vom Datenverarbeitungsrechner), ihre Sensor-ID, den aktuellen Betriebsmodus, den Startpunkt in der Datenmatrix, sowie im Datenbereich die Messwerte der Taxel.

Aufruf von Funktionen der Auswerteeinheit:

Hier sind die möglichen Befehle aufgelistet, die der Datenverarbeitungsrechner an eine Auswerteeinheit senden kann. Mit den verschiedenen Befehlen können Daten angefordert werden, sowie der Betriebsmodus und Systemvariablen beeinflusst oder abgefragt werden.

Die verschiedenen Modi bieten die Möglichkeit zu wählen, ob die betreffende Auswerteeinheit kontinuierlich Daten erzeugt und übermittelt (Freerunmodus) oder, ob jeder Datensatz einzeln im Pollingmodus angefordert werden muss. Dies lässt sich für die verschiedenen Modalitäten einzeln parametrieren.

Modus und Systemvariablen:

Der Block „Modus“ listet die verschiedenen Betriebsmodi auf, in dem eine Auswerteeinheit betrieben werden kann. In „Sys Variable“ sind die Befehle für die Verwendung der Systemvariablen aufgeführt, wie sie in den Funktionen „Set / Get System Variable“ verwendet werden.

Zum Stand der Arbeit sind die Grundfunktionen der Auswerteeinheit definiert. Mit fortschreitender Entwicklung des Systems wird auch der Befehlsumfang des Kommunikationsprotokolls erweitert, um den steigenden Anforderungen gerecht zu werden.

5.2 Bootloader und In System Update

Das Sensorsystem ist für den Betrieb in einem Robotersystem konzipiert. Dies hat zur Folge, dass ein Anschluss an Programmierschnittstellen von außen nicht mehr möglich sein kann. Somit muss es möglich sein, die Firmware der Auswerteelektroniken auch im eingebauten und abgeschlossenen Zustand zu aktualisieren. In diesem Sensorsystem gibt es nur die Möglichkeit, diese Aktualisierung über den CAN-Bus durchzuführen, da dies die einzige bidirektionale Datenverbindung ist.

Um In System Updates zu ermöglichen, muss die Auswerteelektronik mit einem sogenannten Bootloader ausgestattet werden, der es erlaubt über den CAN-Bus neue Firmwarestände zu programmieren. Der Hersteller der MCU (Atmel) bietet hierzu eine Programmiersoftware namens FLIP (FLexible In-System Programmer), sowie einen konfigurierbaren Bootloader an, nach [2].

Nach dem Konfigurieren und Übersetzen für die passende MCU müssen noch die Basiskonfiguration (Fuses) der MCU entsprechend [2, Kap. 3.1] gesetzt werden und der Bootloader mit dem Programmieradapter (hier JTAG ICE 3) in den Speicher MCU geladen werden. Nun kann über die Programmiersoftware FLIP die MCU über den CAN-Bus programmiert werden. Ab diesem Schritt ist nur noch ein von FLIP unterstütztes CAN Interface zur Programmierung notwendig. Somit ist es möglich, bei der Zielhardware ganz auf eine ISP oder JTAG Schnittstelle zur Programmierung zu verzichten, wenn die MCU vor der Montage auf der Platine bereits mit dem passenden Bootloader ausgestattet wird.

Bei der Konfiguration des Bootloaders sind für die sich gemeinsam auf einem Bus befindenden Auswerteelektroniken Nodes zugewiesen, welche eine eindeutige Zuordnung der Elektronik im FLIP ermöglichen. Die aus [2] übernommene Abbildung 5.5 zeigt den Anschluss von mehreren Nodes an ein CAN Interface.

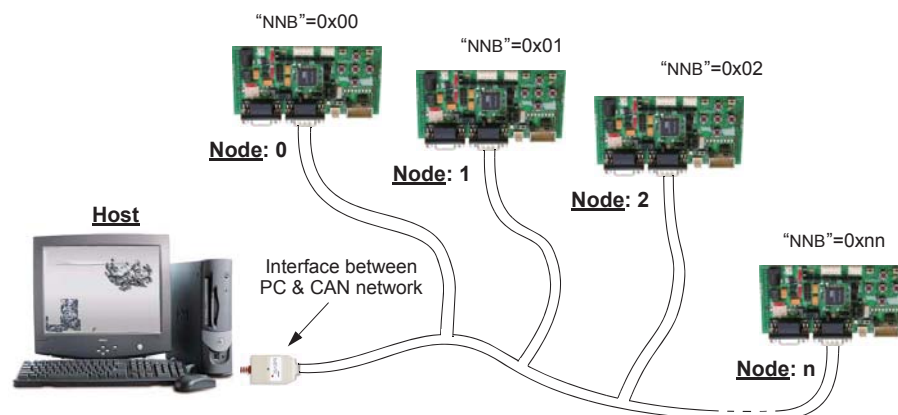


Abbildung 5.5: CAN Bootloader Netzwerk

Um den Bootloader zu aktivieren, muss ein Reset der Schaltung durchgeführt werden und gleichzeitig die Bootloadertaste gedrückt werden (siehe Abb. 5.3). Diese Funktion lässt sich softwareseitig durch das programmierbare BSB (Boot Status Byte) invertieren, sodass der Bootloader mit offenem Taster ausgeführt wird.

Ist der Bootloader auf den für die Firmwareänderung vorgesehenen Auswerteeinheiten aktiv, kann mittels FLIP zu ihr verbunden werden. Hierzu MCU Typ wählen (hier AT90CAN128) und als Interface CAN / PEAK. Beim CAN Node Setup den für das Update bestimmten Node auswählen und verbinden. Anschließend die Firmware mit Select EEPROM wählen und mit der Run-Taste den Updatevorgang starten.

Mit diesem Verfahren können alle am Bus angeschlossenen Auswerteeinheiten mit aktualisierter Firmware ausgestattet werden.

5.3 Echtzeitfähige Kommunikation

Für die ersten Versuche auf den Robotersystemen findet die Kommunikation zwischen dem Sensorsystem und dem Roboter über das ROS statt. Dies geschieht auf der high-level Ebene des ROS und kann somit nicht in Echtzeit in die Steuerung des Roboters eingreifen.

Die Auswerteelektronik liefert nach berechenbarer Zeit für die entsprechenden Abfragebefehle die angeforderten Daten. Bei einer zeitlich kontrollierten Übertragung der Daten über den CAN-Bus werden diese auch mit berechenbarer Latenzzeit an das Interface übertragen.

Um das System am Punkt der Datenaufbereitung echtzeitfähig zu halten, wird ein CAN Interface mit passenden Treibern für ein Echtzeitbetriebssystem (z.B. QNX oder Xenomai) notwendig.

Steuernde Eingriffe in den Echtzeitteil der Robotersteuerung setzen jedoch Modifikationen an der Steuerung voraus, welche weitreichende Planung und Vorlaufzeit benötigen. Somit stellt das Konzept für die ersten Versuche über das ROS zu kommunizieren eine realisierbare Alternative dar.

6 Ergebnisse

Dieses Kapitel befasst sich mit der Überprüfung der in Abschnitt 1.1 gestellten Aufgaben für diese Arbeit und der Funktion der Auswerteeinheiten im Verbund.

6.1 Verifizierung der Anforderungen

Eine Übersicht über die Einhaltung der in Abschnitt 1.1 gestellten Anforderungen wird in diesem Abschnitt ausgeführt.

Beibehaltung des CAN-Bussystems:

Der CAN-Bus wurde für die Datenübertragung zwischen den Auswerteeinheiten beibehalten.

Verbesserung der Messgeschwindigkeit:

Die Messgeschwindigkeit konnte durch die Änderungen am System von ca. 100 Hz (ausgelesene Sensorpatches mit 64 Taxeln pro Sekunde) auf 400 Hz vervierfacht werden. Durch die hinzugefügte Funktion, dass alle taktilen Elemente miteinander verschalten werden können, kann eine Abtastrate von ca. 10 kHz für den gesamten Sensor realisiert werden (bei einem Sensor pro Bus, mit Anpassungen im Freerunning Modus). Die genauen Auswertegeschwindigkeiten hängen davon ab, wie viele Sensoren am Bus angeschlossen sind, bzw. in welchem Modus diese betrieben werden.

Reduzierung der Messwertschwankungen:

Die Reduzierung der Messwertschwankungen der Auswerteeinheit wurde zugunsten der höheren Auswertegeschwindigkeit niedriger priorisiert. Die Messwertschwankungen bei einer Messung mit 1000 Messpunkten belaufen sich auf eine Standardabweichung von maximal 31 der 65535 Diskretisierungsschritte des A/D-Wandlers. Diese Messwertschwankungen sind im Vergleich zu den Schwankungen der Widerstände des Sensors gering und sind deshalb vorerst vernachlässigbar.

Erweiterung der Sensormodalitäten:

Die Auswerteeinheit konnte erfolgreich um ein Mikrofon zur Oberflächenrauigkeitsmessung, ein Accelerometer zur Bestimmung der Lage im Raum und einen Drucksensor der den Innendruck im Sensor bestimmt, erweitert werden.

Schirmung gegen äußere Störeinkopplungen:

Die Schirmung der Elektronik gegen äußere Störeinflüsse wurde im Design der Elektronik berücksichtigt und wird beim Aufbau der Zielhardware realisiert.

Immunisierung der Schaltung gegen elektrostatische Entladungen:

Zum Schutz gegen elektrostatische Entladungen wurden der Schaltung Varistoren hinzugefügt, um Spannungsüberhöhungen durch elektrostatische Entladungen gezielt zu minimieren.

Einhaltung der Echtzeitbedingungen (vorsehen):

Die Einhaltung der Echtzeitbedingungen wurde nicht umgesetzt, da hierfür umfassende Änderungen der Software des Datenerfassungsrechners notwendig sind und die Echtzeitfähigkeit des Systems für den Betrieb des Systems zum momentanen Stand noch nicht zwingend benötigt werden (Highlevelbefehle über ROS werden nicht in Echtzeit verarbeitet).

Entwicklung eines Kommunikationsprotokolls für das Sensorverbundsystem:

Die Grundfunktionen der Auswerteeinheiten werden vom aktuellen Stand des Kommunikationsprotokolls umfasst. Mit fortschreitender Entwicklung wird auch das Kommunikationsprotokoll mit weiteren Funktionen erweitert.

Einfache Updatemöglichkeit der Firmware im Mikrocontroller:

Durch einen Bootloader können die Auswerteeinheiten über den CAN-Bus mit neuer Software versorgt werden. Ein Update benötigt somit keine eigene Programmierhardware mehr und kann im eingebauten Zustand programmiert werden.

Anpassung der Visualisierungssoftware:

Entwicklungen an der bestehenden Visualisierungssoftware haben gezeigt, dass für die weiteren Arbeiten mit dem Sensorsystem und dessen Kommunikation mit Robotersystemen eine Neuentwicklung der Software erforderlich ist. Diese soll es ermöglichen, über das ROS Daten zu veröffentlichen, was nicht mit dem aktuellen Stand der Visualisierungssoftware vereinbar ist. Desweiteren sind durch die komplexeren Systemstrukturen aufwendigere Konfigurationsmöglichkeiten notwendig, was ebenfalls eine Neuentwicklung der Software befürwortet. Somit wurde projektseitig beschlossen, die Software des Datenverarbeitungsrechners grundlegend neu zu entwickeln.

Die Tabelle 6.1 fasst die technischen Daten der erweiterten Auswerteeinheit für taktile Sensoren zusammen.

Auswertbare taktile Messpunkte	64
Messrate bei 64 Taxel	bis zu 400 Hz
Messrate bei 1 Taxel	bis zu 10 kHz
Maximale Messwertaufösung	16 Bit
Standardabweichung der Messwerte	≤ 5 Bit
Abmaße	75 mm x 75 mm
Leistungsaufnahme	< 800 mW
Kalibrierung	Nullpunktgleich, variabler Messwiderstand
Systembus	CAN-Bus

Tabelle 6.1: Technische Daten der Auswerteeinheit

6.2 Sensorsystem im Betrieb

Entwicklungsumgebung für das Sensorsystem

Um die Funktionsfähigkeit des Sensorsystems nachzuweisen, wurde eine Versuchsumgebung, bestehend aus drei Auswerteeinheiten mit unterschiedlichen Sensoren, einem CAN Interface und einem Computer aufgebaut. Die Elektroniken sind gemeinsam am CAN-Bus mit dem Interface angeschlossen, welches mit dem PC verbunden ist, siehe Abbildung 5.1 (CAN 0 Zweig, Interface, Datenaufbereiterrechner).

Die Abbildung 6.1 zeigt den Versuchsaufbau:

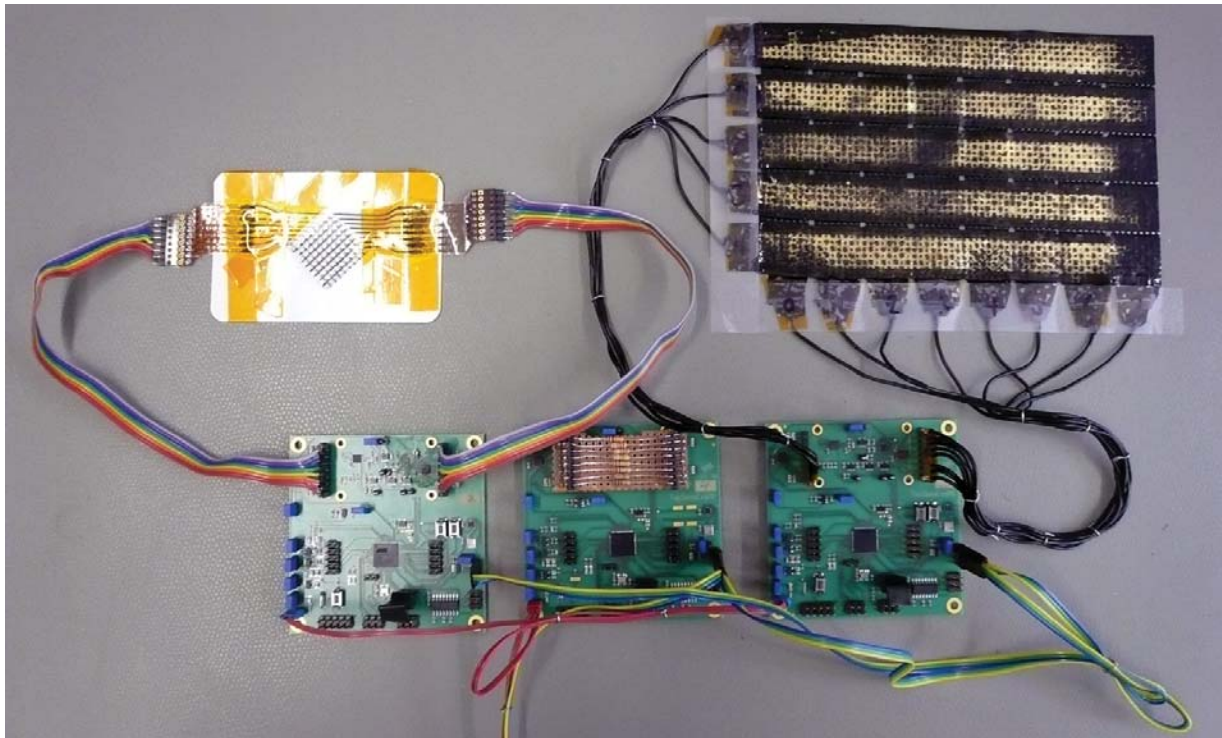


Abbildung 6.1: Versuchsaufbau des Sensorsystems mit zwei Sensoren und einem Widerstandsdummy

Dieser Versuchsaufbau legt den Grundstein zur Entwicklung der Softwarekomponenten für den späteren Einsatz auf einem Robotersystem.

Mobile Visualisierungseinheit

Für Diagnosezwecke, mobile Visualisierung und Hochgeschwindigkeitsvideoaufnahmen wurde eine angepasste Software der Auswerteelektronik entwickelt, um eine 8 x 8 Leuchtdiodenmatrix anzusteuern. Mit dieser Software kann der Datenstrom auf dem Bus in binären Zuständen (taxel belastet / unbelastet) auf der LED Matrix visualisiert werden.

Die Abbildung 6.2 zeigt einen Versuchsaufbau zur Hochgeschwindigkeitsvideoaufnahme der mobilen Visualisierungseinheit, hier links im Bild die mobile Visualisierungseinheit mit LED Matrix in der Mitte. Der Sensor wird von der Auswerteeinheit rechts im Bild ausgewertet und die Daten per CAN-Bus an die Visualisierungseinheit übertragen. Im Anhang ist in Abschnitt D der Einschlag einer Stahlkugel als Serienbild einer Hochgeschwindigkeitsaufnahme einsehbar.

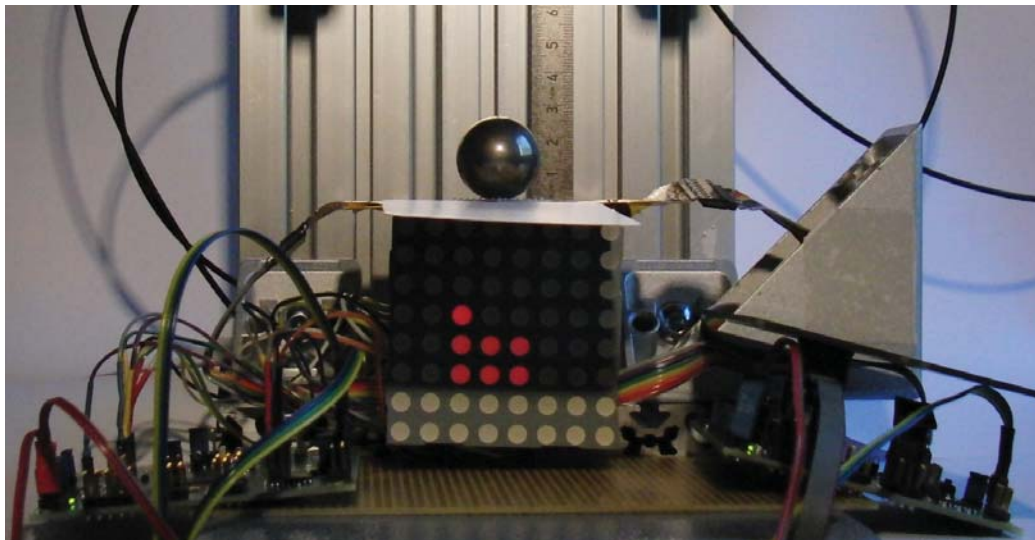


Abbildung 6.2: Mobile Visualisierungseinheit im Versuchsaufbau

Messergebnisse

Um die Reaktionsgeschwindigkeit des Systems zu überprüfen, wurde ein Versuch auf einem Fallbeil durchgeführt. Hierbei verfügt das Fallbeil über die Möglichkeit, die Kraft des Probekörpers (am Fallbeil montierter abgerundeter ($R=20\text{mm}$) Aluminiumzylinder), während er auf das Zielobjekt (Sensor) trifft zu erfassen. Diese Sprunganregung des Sensors wurde im Versuch zeitgleich mit einer Auswerteelektronik aufgezeichnet und in der Abbildung 6.3 zusammengeführt. Hierbei zeigt die blaue Strichpunktlinie die Messwerte des Sensors (als Messwerte des A/D-Wandlers) und die rote durchgehende Linie die Messwerte der Kraftmessdose des Fallbeils. Der Vergleich der Graphen zeigt, dass die Sensormesswerte der auf ihn aufgebrachtene Kraft folgen.

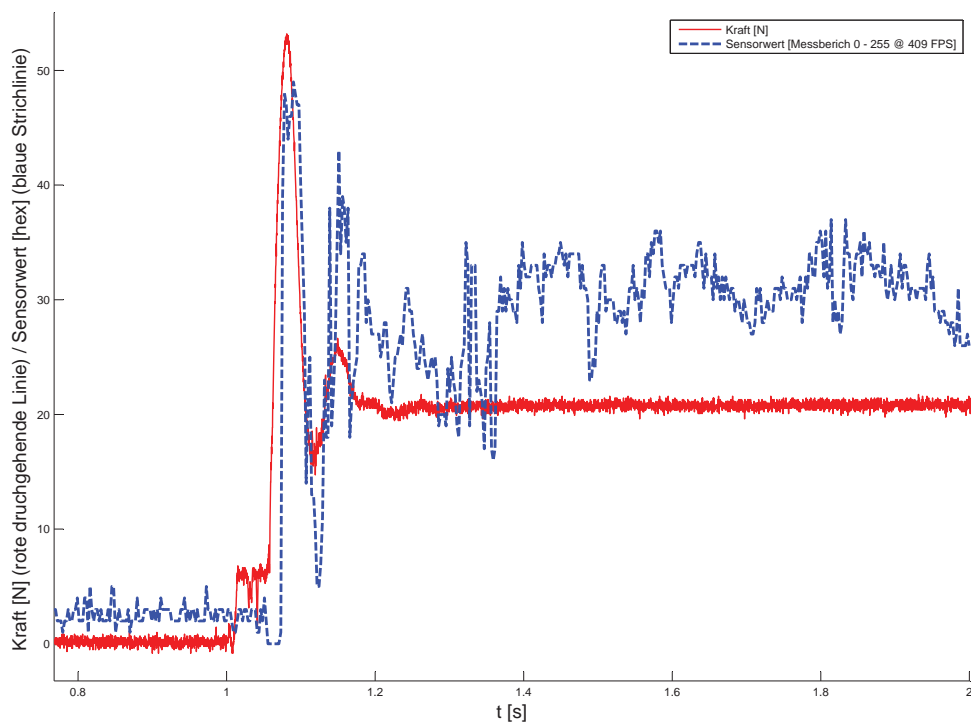


Abbildung 6.3: Messwerte bei Sprunganregung

Messwerte bei zusammengeschalteten Taxeln

Durch das Ersetzen der Multiplexer mit Analogschaltern ist es möglich, die 64 Taxel des Sensors zu einem gesamten Taxel zu verbinden. Daraus resultiert, dass der gesamte Sensor auf eine einwirkende Kraft mit nur einem Auslesevorgang überprüft werden kann, was eine deutliche Erhöhung der Abtastrate auf ein Kontaktereignis ermöglicht.

Im Versuchsaufbau von Abbildung 6.2 wurde ein Einschlag einer Stahlkugel (mit anschließendem Springen der Kugel bis zum Stillstand auf dem Sensor) in zusammengeschalteter Konfiguration, mit einer Abtastrate von 10 kHz aufgezeichnet und in die Abbildungen 6.4 und 6.5 überführt.

Die Messergebnisse belegen, dass eine signifikant höhere Auswertegeschwindigkeit erreichbar ist, wenn nur auf das Kontaktereignis geprüft wird.

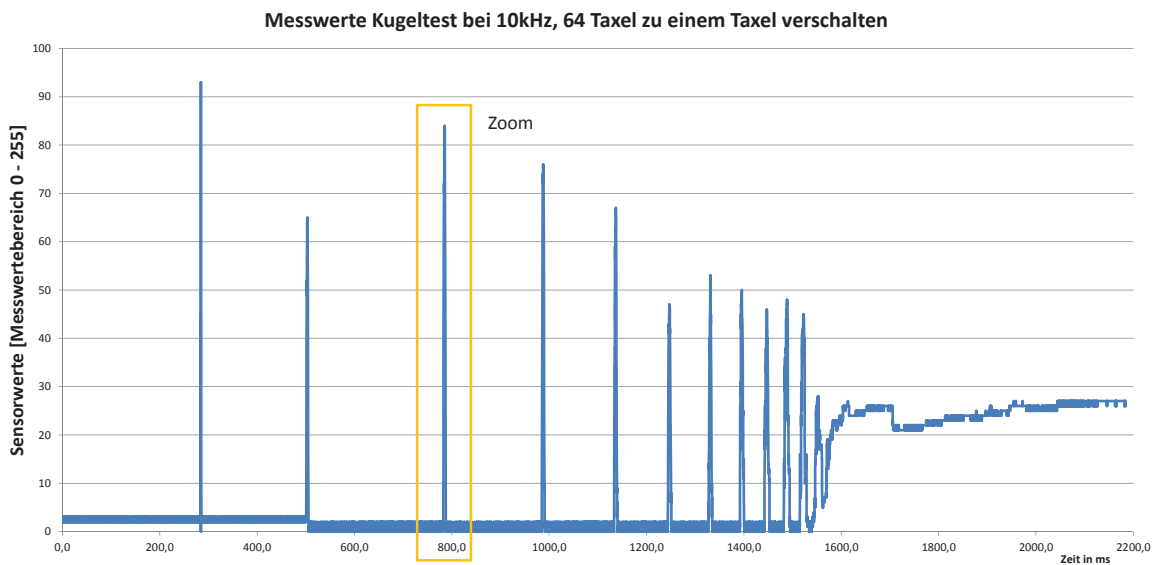


Abbildung 6.4: Messwerte Kugeleinschlagversuch Übersicht

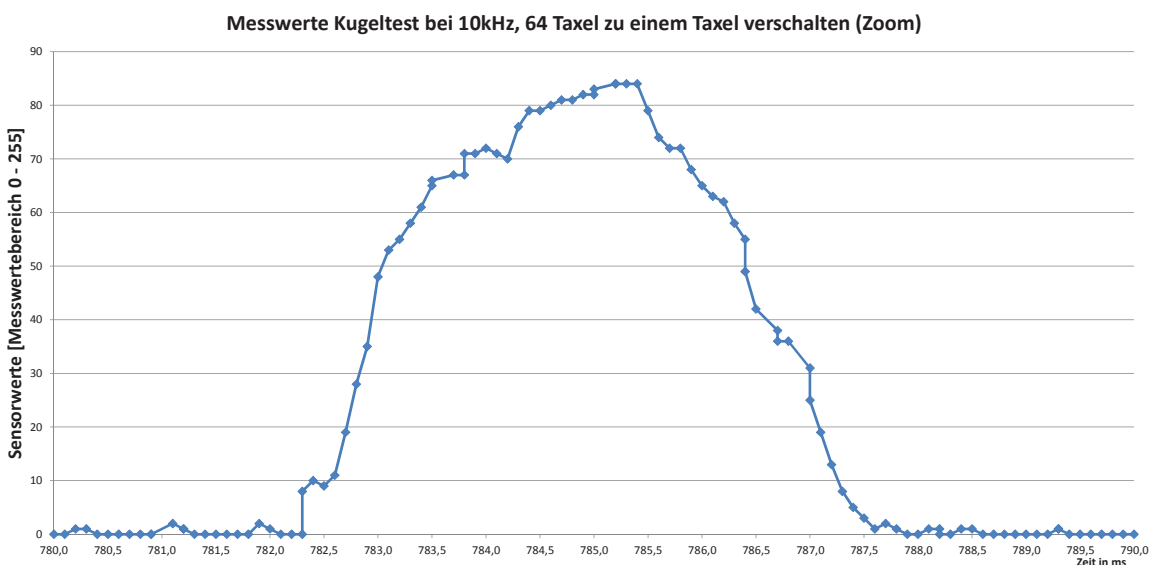


Abbildung 6.5: Messwerte Kugeleinschlagversuch Vergrößerung

Oberflächenrauigkeit

Um die Funktion der in Abschnitt 4.1.1 beschriebene Messung der Oberflächenrauigkeit zu prüfen, wurde ein Versuch mit verschiedenen Oberflächen durchgeführt.

Hierzu wurde die Oberfläche eines mit einem Mikrofon ausgestatteten Sensors mit den verschiedenen Oberflächen überstrichen und die Innengeräusche der Kavität aufgezeichnet. In dieser ersten manuell durchgeführten Versuchsreihe ist erkennbar, dass sich die erzeugten Innengeräusche in ihren beteiligten Frequenzanteilen unterscheiden. Dies wird durch die in Abbildung 6.6 dargestellte Frequenzanalyse veranschaulicht.

Der Farbverlauf stellt die Gewichtung der beteiligten Frequenzen dar, wobei die geringer beteiligten Frequenzanteile blau und mit zunehmender Beteiligung als Farbverlauf nach rot dargestellt werden.

Um vergleichbare reproduzierbare Ergebnisse zu generieren, die eine zuverlässige Oberflächenbestimmung ermöglichen, müssen weitere Versuche durchgeführt werden. Hierzu wird ein Versuchsaufbau benötigt, der mit konstanter Geschwindigkeit die Oberfläche des Sensors mit verschiedenen Materialien überstreicht.

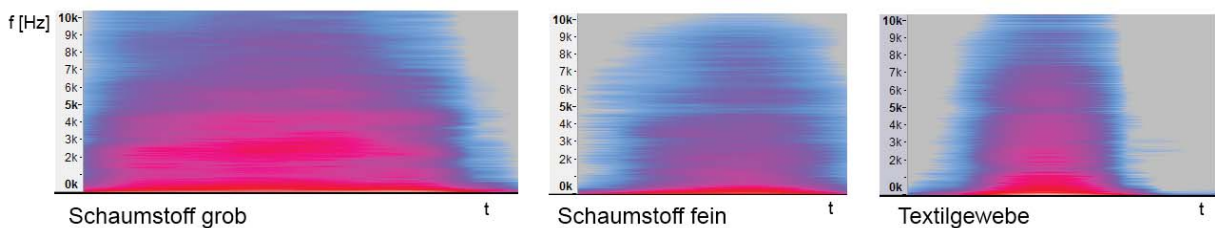


Abbildung 6.6: Innengeräusche der Kavität bei Anregung mit verschiedenen Materialien

7 Ausblick

Im Anschluss an diese Arbeit wird die Auswerteelektronik auf die Abmaße der Zielhardware von 22 mm x 22 mm reduziert. Mit der entwickelten Auswerteelektronik kann anschließend unter realitätsnahen Bedingungen ein Funktionstest an einem Robotersystem durchgeführt werden.

Dieses Experiment soll absichern, dass das Sensorsystem nicht nur im Labor, sondern auch unter realen Betriebsbedingungen auf dem Robotersystem bestehen kann. Zudem kann so für die entsprechenden Montagestellen eine passende Kontaktierung entwickelt werden.

Weitere Untersuchungen sind notwendig, um zu prüfen, wie mit den Daten der neu hinzugekommenen Sensormodalitäten verfahren werden muss, um die zusätzlichen Informationen für die Klassifizierung von Objekten nutzbar zu machen.

Mit der nächsten Generation der Auswerteeinheiten wird mit hoher Wahrscheinlichkeit ein Wechsel des Mikrocontrollers notwendig werden, um eine noch leistungsfähigere und kleinere Auswerteeinheit zu entwickeln. Die weitere Reduzierung der Abmaße ist für den Einsatz auf Roboterhänden der kommenden Generationen erforderlich, da die Anforderungen bezüglich des verfügbaren Bauraums weiter steigen werden.

Der Datenaustausch mit dem Robotersystem erfolgt über das ROS (Robot Operating System). An dieser Schnittstelle werden umfassende Softwarearbeiten notwendig werden, um eine erfolgreiche Kommunikation mit dem Roboter zu ermöglichen, da an dieser Stelle die taktilen Daten der einzelnen Sensoren logisch ausgewertet werden müssen (z.B. Kantendetektion).

Weitere Entwicklungsarbeit kann zukünftig in ein Programm fließen, um für neu hergestellte Sensoren eine Basiskonfiguration mit den ermittelten Kalibrierungsdaten zu erzeugen. Somit wird die Herstellung von Sensoren möglich, die ihre Konfiguration schon im eingebauten Speicher bereithalten.

Um, während der Interaktion mit Robotern die vom taktilen Sensorsystem erzeugten Daten besser zu visualisieren, muss auch die Visualisierungssoftware weiter verbessert werden, um einen ergonomischen Umgang mit dem System zu erreichen.

Literaturverzeichnis

- [1] ANALOG DEVICES, INC.: *Datenblatt: Serially-Controlled Octal SPST Switches ADG1414, Rev. 0, Norwood, USA, 10/2009.*
- [2] ATMEL CORPORATION: *Applikationsschrift: AVR076: AVR CAN - 4K Boot Loader, Dokument: 8247A-CAN-08/09, San Jose, USA, 2008.*
- [3] ATMEL CORPORATION: *Datenblatt: AT90CAN32/64/128, Dokument: 7679G-CAN-08/08, San Jose, USA, 2008.*
- [4] ATMEL CORPORATION: *Datenblatt: ATmega16M1/32M1/64M1/32C1/64C1, Dokument: 7647G-AVR-09/11, San Jose, USA, 2011.*
- [5] BROWN, M.: *Power supply cookbook*. EDN series for design engineers. Newnes, 2. Aufl., 2001.
- [6] CONTAG GMBH: *Produkt-Info HDI/SBU-Technik*. http://www.contag.de/uploads/pi_ti/hdi_d.pdf, 2008. Zugriffsdatum 15.02.2012, Ausgabestand D.
- [7] COOPER BUSSMANN: *Datenblatt: 41206ESDA, Four Element Array Chip, Polymer ESD Suppressor With SurgX Technology, Dokument: 41206, Florida, USA, 07/2006.*
- [8] DAHIYA, R. S. und M. VALLE: *Tactile Sensing for Robotic Applications*. Sensors (Peterborough, NH), Dezember, 2008.
- [9] ECK, R.: *Skript: Echtzeitsysteme, Georg-Simon-Ohm-Hochschule Nürnberg, 2006.*
- [10] EPCOS AG: *Datenblatt: Pressure Sensors C29 series, Dokument: AS SEN PD, 08/2009.*
- [11] FREESCALE SEMICONDUCTOR INC.: *Applikationsschrift: Quad Flat Pack No-Lead (QFN) / Micro Dual Flat Pack No-Lead (uDFN), Dokument: AN1902 Rev. 4.0, Tempe, USA, 09/2008.*
- [12] FREESCALE SEMICONDUCTOR, INC.: *Referenzhandbuch: M68HC11 Reference Manual, Dokument: M68HC11RM/D, 04/2002.*
- [13] HERING, E., K. BRESSLER und J. GUTEKUNST: *Elektronik für Ingenieure und Naturwissenschaftler*. Springer, 5. Aufl., 2005.
- [14] INTERNATIONAL ELECTROTECHNICAL COMMISSION: *IEV Online - IEV ref 351-32-10*. <http://www.electropedia.org/iev/iev.nsf/display?openform&ievref=351-32-10>, 10/2006. Zugriffsdatum 18.02.2012.
- [15] IVERS-TIFFÉE, E. und W. MÜNCH: *Werkstoffe der Elektrotechnik*. Teubner Studienskripten. Teubner, 10. Aufl., 2007.
- [16] LEE, B.: *An Overview of ESD Protection Devices*. Compliance Engineering - Annual Reference Guide, 2001. Zugriffsdatum: 27.02.2012.
- [17] LEUPOLZ, C.: *Konzeption und Aufbau eines Prüfstandes für taktile Sensoren*. Bachelorarbeit, Hochschule für angewandte Wissenschaften Augsburg, Deutsches Zentrum für Luft- und Raumfahrt, 2012.

- [18] MESCHEDER, U.: *Mikrosystemtechnik: Konzepte und Anwendungen*. Teubner, 2. Aufl., 2004.
- [19] NXP SEMICONDUCTORS: *Spezifikation: UM10204 - I²C-bus specification and user manual Rev. 4, Dokument: UM10204, 02/2012*.
- [20] PHILIPS SEMICONDUCTORS: *Spezifikation: I²S-bus specification, 06/1996*.
- [21] RANZINGER, C.: *Entscheidungskriterien für HDI-Schaltungen*. Elektronik Praxis, 10:98–99, 2007.
- [22] ROBERT BOSCH GMBH: *Spezifikation: CAN Specification 2.0, Stuttgart, Deutschland, 1991*.
- [23] SCHMITT, W. D.: *Grundlagen der Leiterplatten-Baugruppen-Entwicklung und Fertigung*. GRIN Verlag, 1. Aufl., 2009.
- [24] SCHNÖSS, F.: *Unveröffentlichtes Dokument, Entwurf und Realisierung einer integrierten Auswerteeinheit für polymerbasierte taktile Sensoren*. Bachelorarbeit, Technische Universität München, Deutsches Zentrum für Luft- und Raumfahrt, 2011.
- [25] SCHNELL, G. und B. WIEDEMANN: *Bussysteme in der Automatisierungs- und Prozesstechnik*. Vieweg+Teubner, 6. Aufl., 2006.
- [26] SCHOLZ, P.: *Softwareentwicklung eingebetteter Systeme*. Springer-Verlag Berlin Heidelberg, 1. Aufl., 2005.
- [27] SCHWAB, A. und W. KÜRNER: *Elektromagnetische Verträglichkeit*. VDI-Buch. Springer Berlin Heidelberg, 6. Aufl., 2011.
- [28] SPEETER, T. H.: *A Tactile Sensing System for Robotic Manipulation*. The International Journal of Robotics Research, Dezember, 1990.
- [29] STROHMAYR, M.: *Artificial Skin in Robotics*. Doktorarbeit, Karlsruhe Institute of Technology (KIT), 2012.
- [30] TEXAS INSTRUMENTS: *Datenblatt: 16-BIT, 500-KSPS, SERIAL INTERFACE MICRO-POWER, MINIATURE, SAR ANALOG-TO-DIGITAL CONVERTER, Dokument: SLAS600, Dallas, USA, 05/2008*.
- [31] TEXAS INSTRUMENTS: *Datenblatt: LP2985, 150-mA LOW-NOISE LOW-DROPOUT REGULATOR WITH SHUTDOWN, Dokument: SLVS522N, Dallas, USA, 06/2011*.
- [32] TEXAS INSTRUMENTS: *Datenblatt: OPA354 - 250MHz, Rail-to-Rail I/O, CMOS OPERATIONAL AMPLIFIERS, Dokument: SBOS233E, Dallas, USA, 05/2009*.
- [33] TIETZE, U. und C. SCHENK: *Halbleiter-Schaltungstechnik*. Springer Verlag Berlin Heidelberg New York, 11. Aufl., 1999.
- [34] WÜRTH ELEKTRONIK: *Datenblatt: ABSCHIRMGEHÄUSE SMD-RAHMEN WE-SHC, Artikelnummer: 36103205, Waldenburg, Deutschland, 08/2009*.

Anhang

A Schaltpläne der Evaluationsplatine

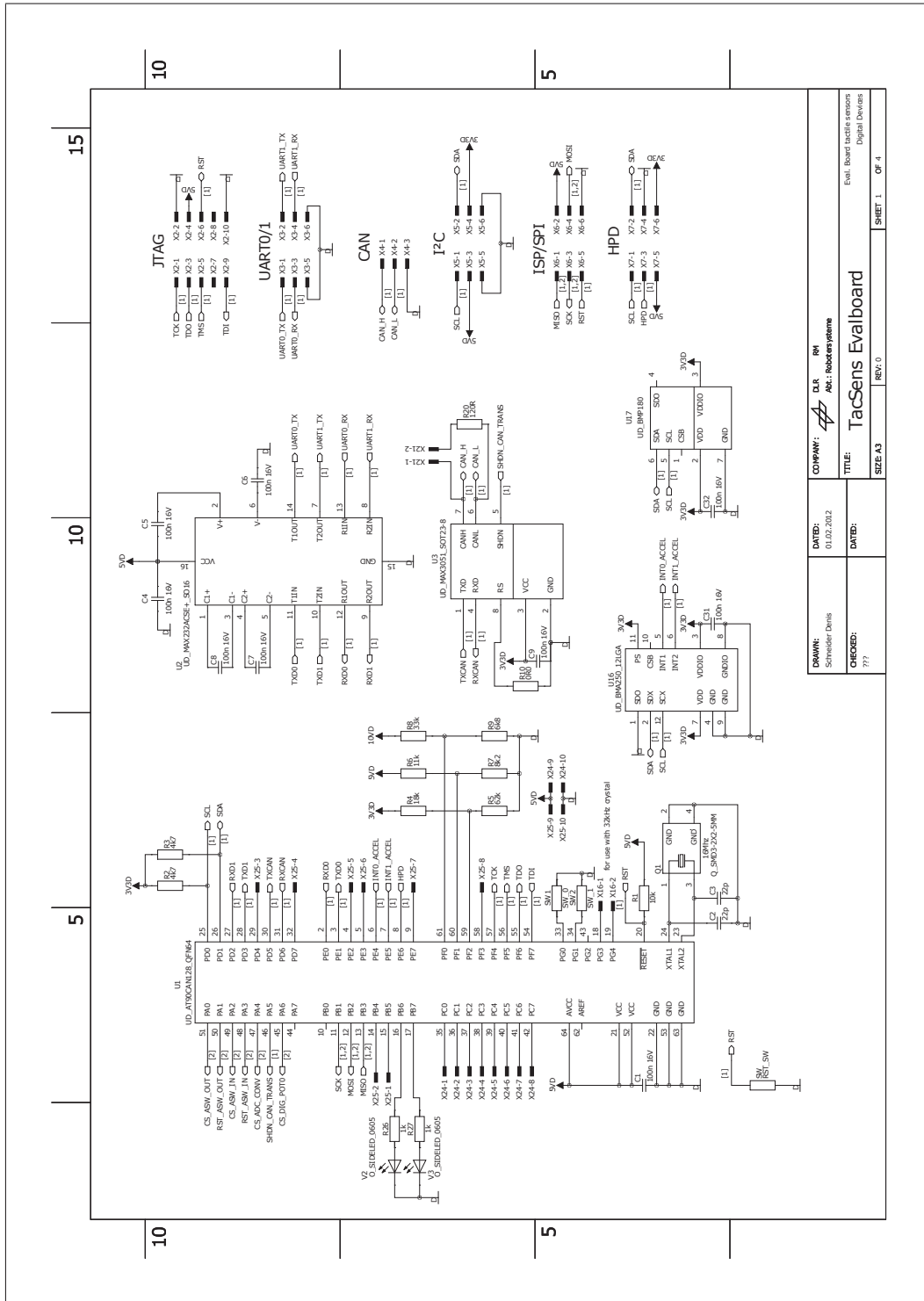


Abbildung A.1: Schaltplan digitale Komponenten

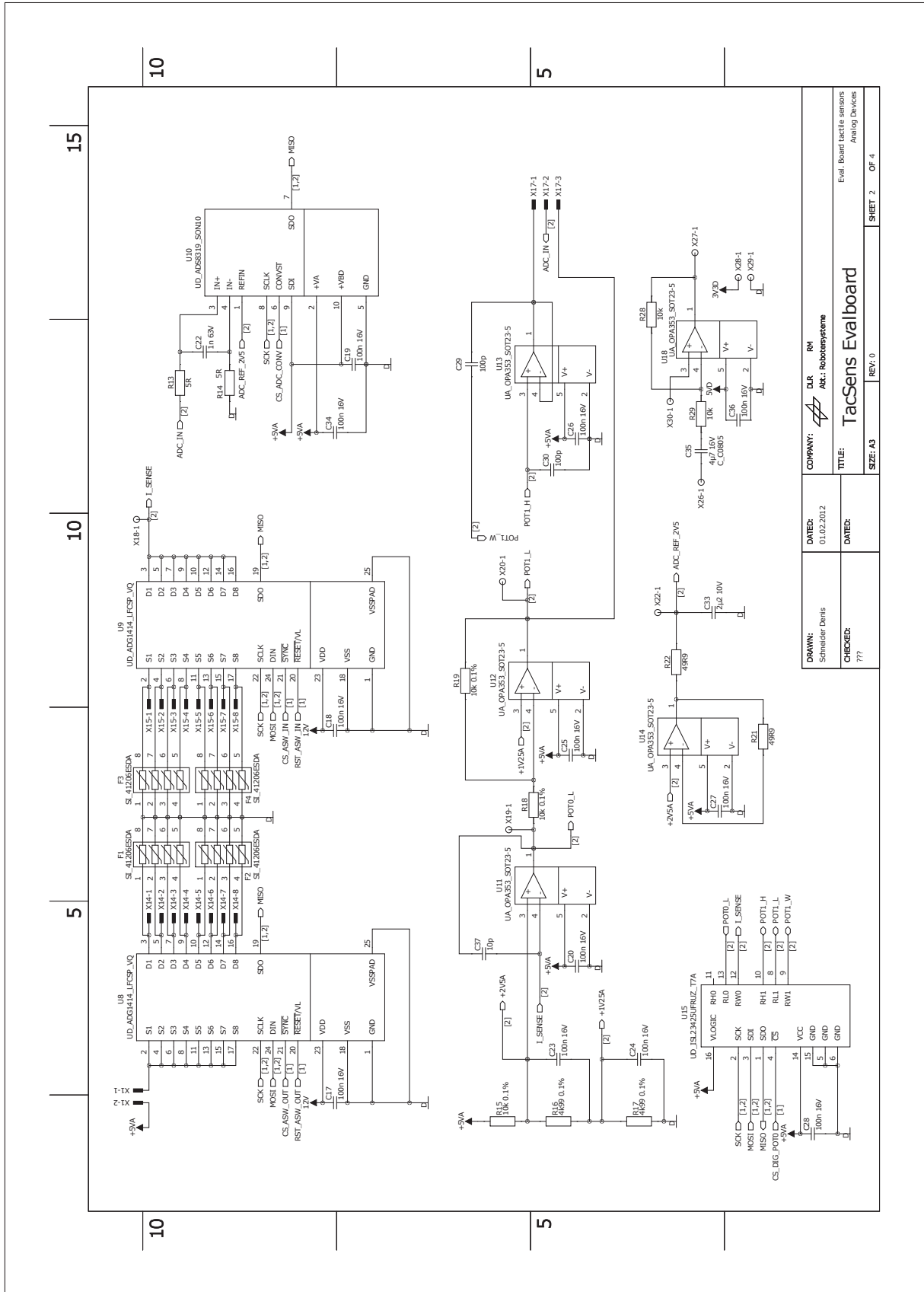


Abbildung A.2: Schaltplan analoge Komponenten

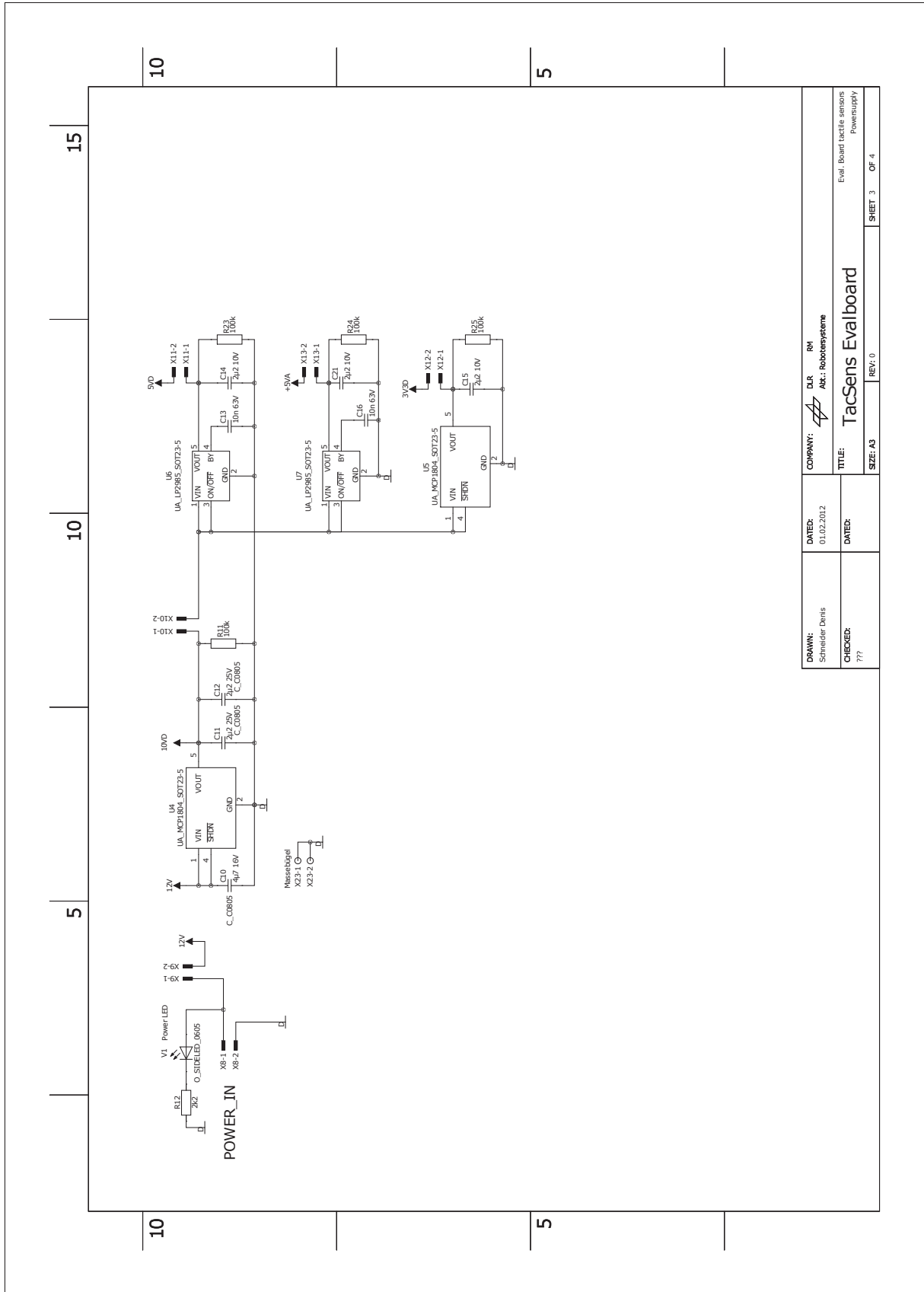


Abbildung A.3: Schaltplan Stromversorgung

DRAWN: Schneider Denis	DATEX: 01.02.2012	COMPANY: Abt. Robotersysteme	DIR RM
CHECKER: 777	DATEX: 777	TITLE: TacSens Evalboard	REV: 0
SHEET 3 OF 4			Eval. Board tactile sensors Powersupply

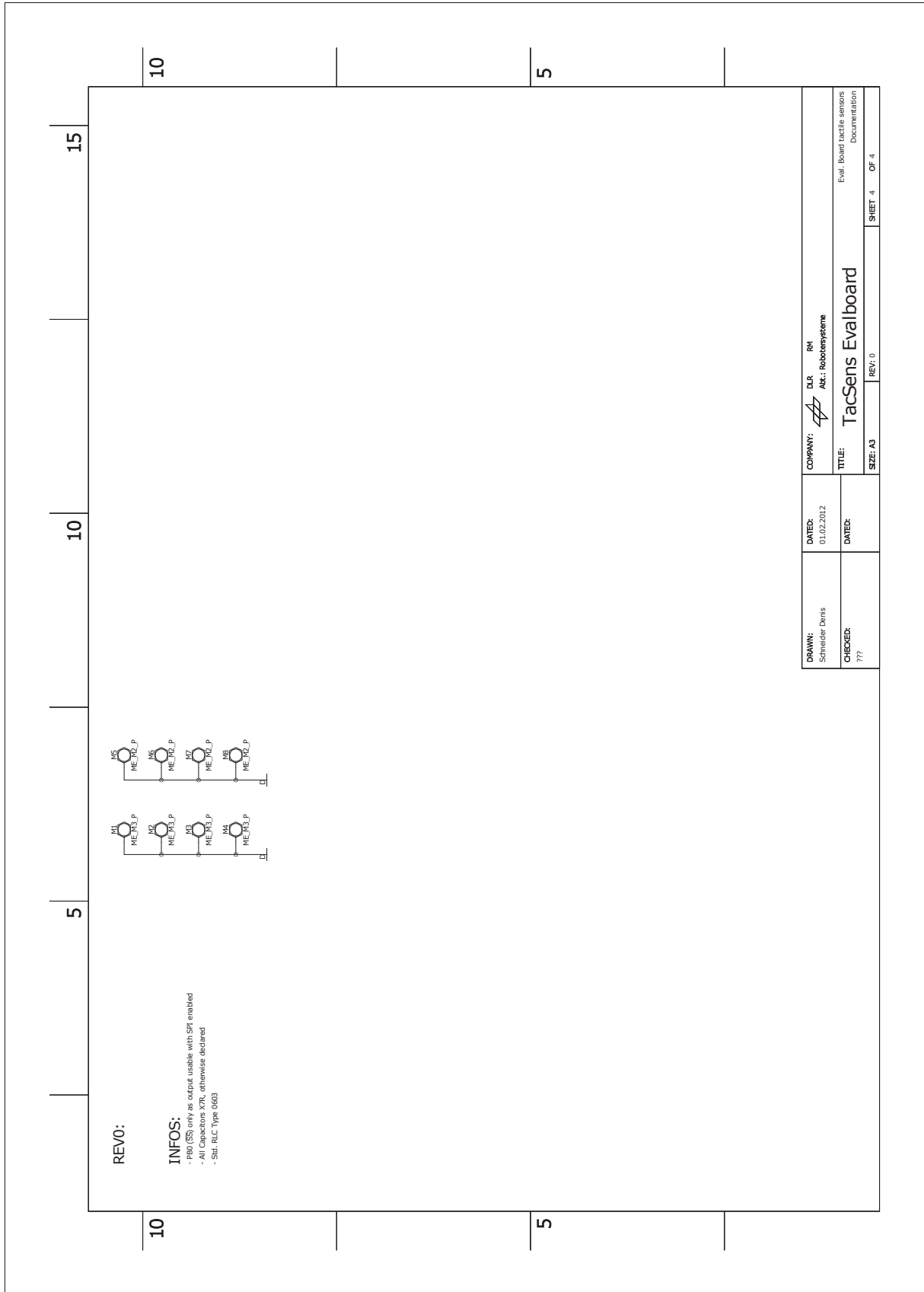
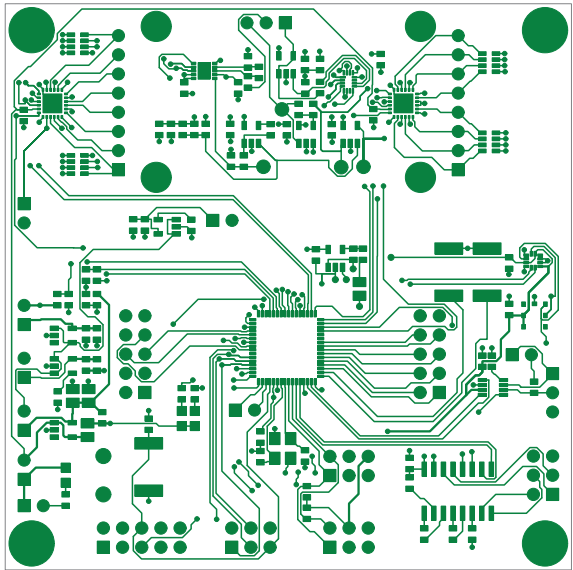
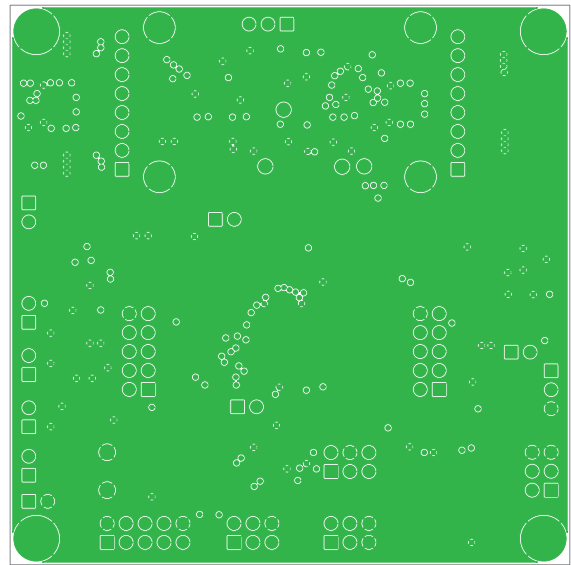


Abbildung A.4: Schaltplan Dokumentation

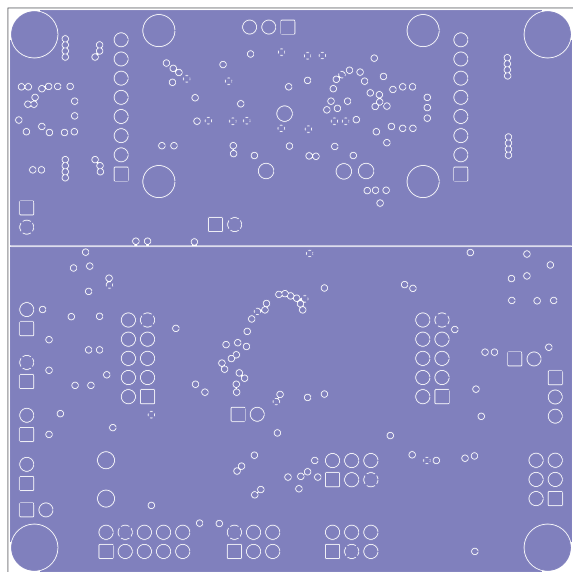
B Layout der Evaluationsplatine



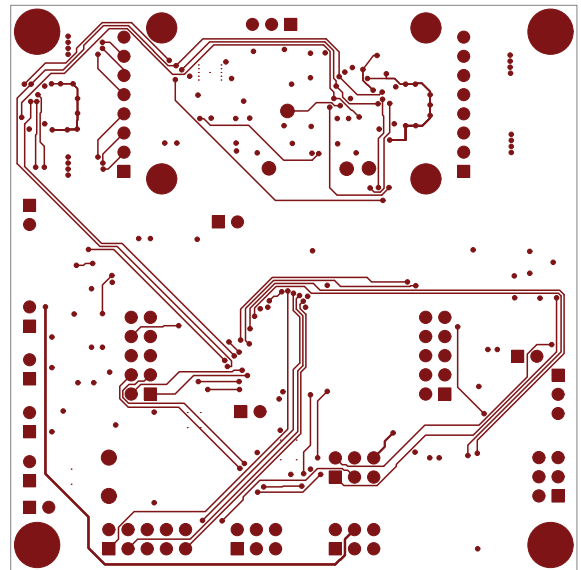
(a) TOP



(b) GND



(c) VCC



(d) BOTTOM

Abbildung B.1: Layout der vierlagigen Evaluationsplatine

C Kommunikationsprotokoll

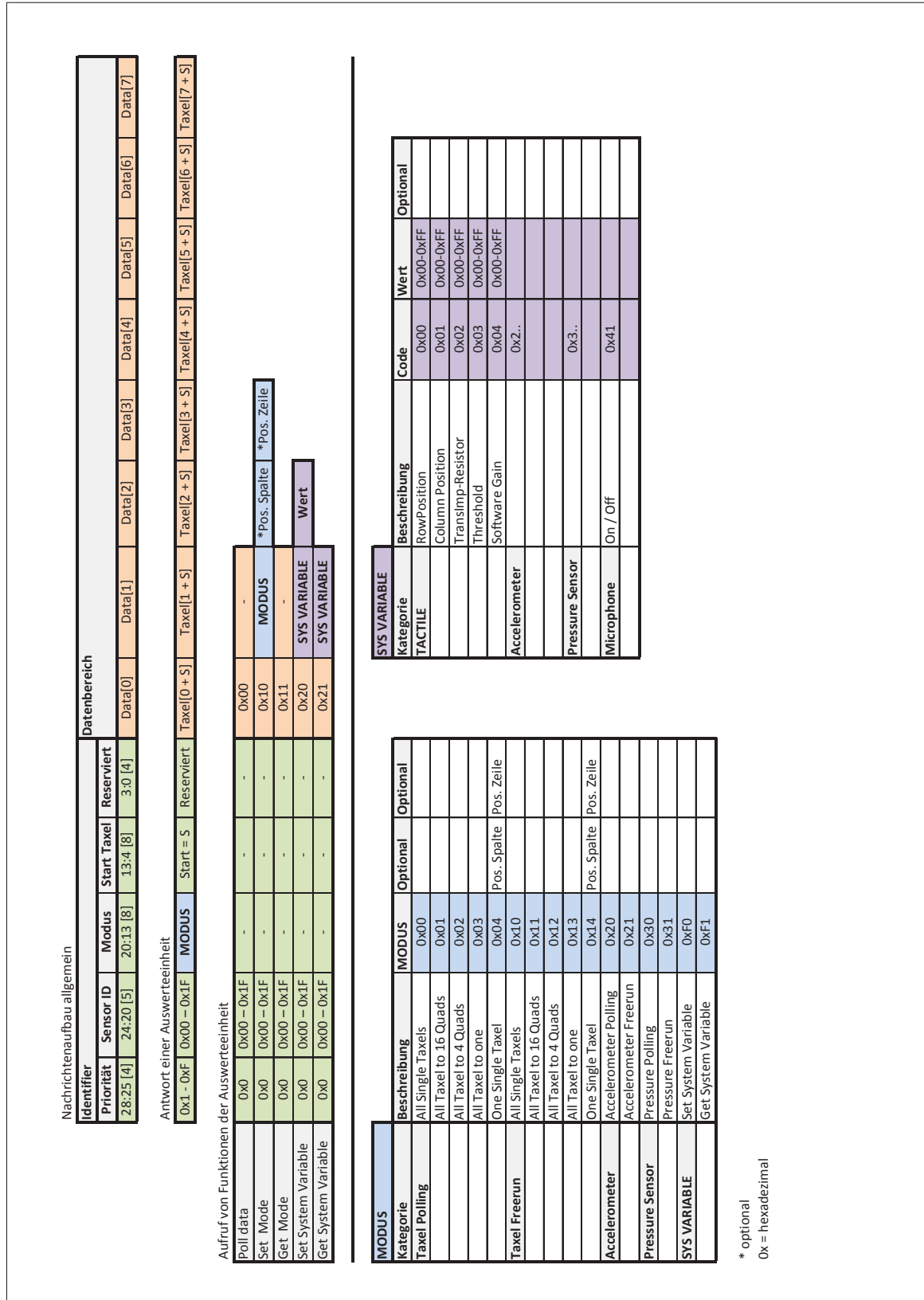


Abbildung C.1: Kommunikationsprotokoll

D Serienbild vom Einschlagtest

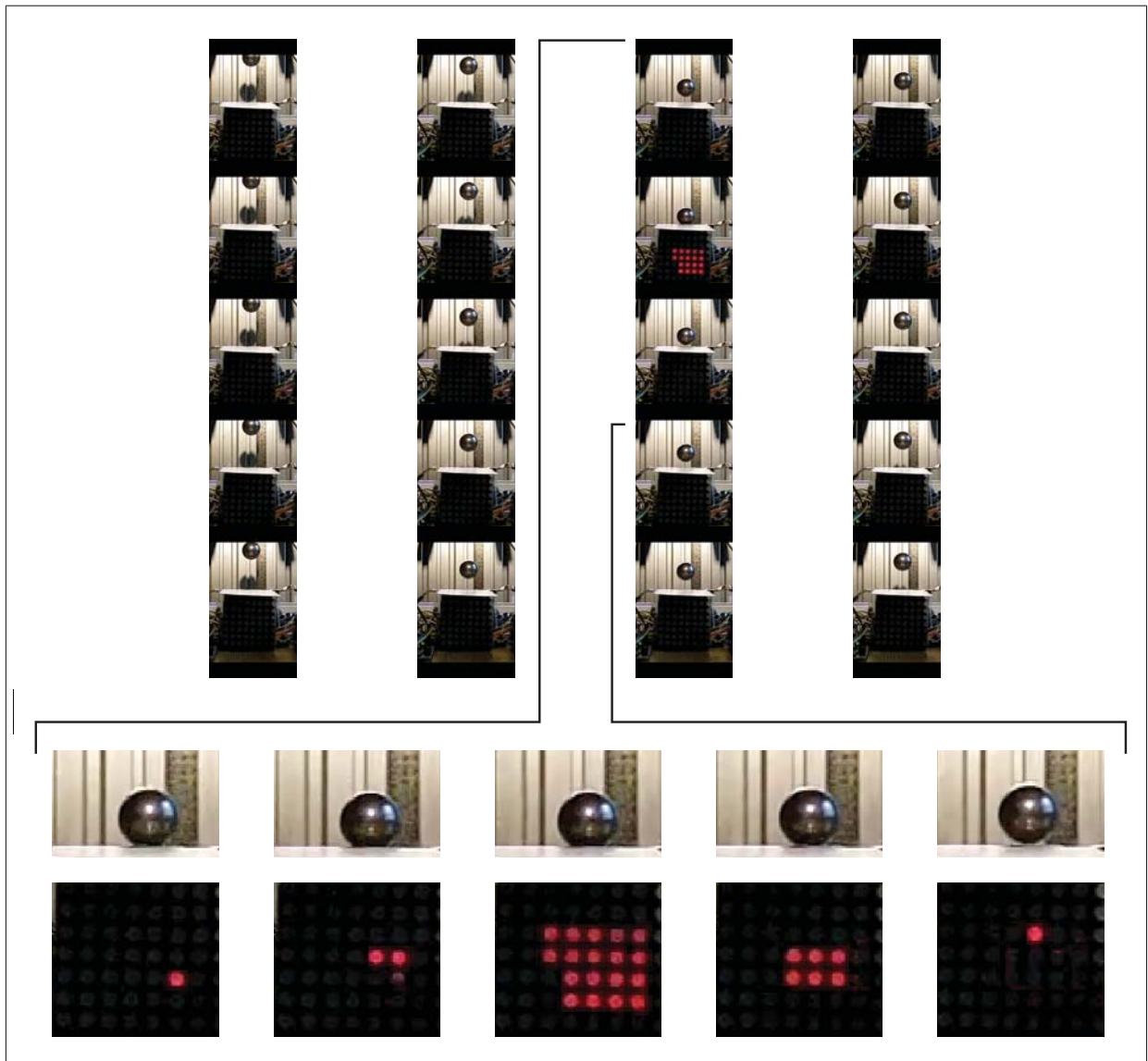


Abbildung D.1: Einschlagversuch mit einer Stahlkugel als Serienbild, Abbildung
übernommen aus [29]

E Quellcode der Mikrocontrollerfirmware

```

1 //-----
2 // FIRMA           Deutsches Zentrum für Luft- und Raumfahrt
3 // ABTEILUNG       Institut für Robotik und Mechatronik
4 // PROJEKT         Erweiterung einer Auswerteelektronik für taktile Sensoren
5 //               zum multimodalen Sensorsystem
6 // Autoren         Schneider Denis
7 // Beschreibung    TacSenseEval10.h
8 // Erstellt        07.06.2012
9 //-----
10
11 //INCLUDED STD HEADERS
12 #include <avr/io.h>
13 #include <util/twi.h>
14 #define F_CPU 16000000
15 #include <util/delay.h>
16 #include <avr/interrupt.h>
17
18 //UART for Debug
19 #define BAUD 9600UL
20 #define UBRR_VAL ((F_CPU+BAUD*8)/(BAUD*16)-1) // runden
21 #define BAUD_REAL (F_CPU/(16*(UBRR_VAL+1))) // Reale Baudrate
22 #define BAUD_ERROR ((BAUD_REAL*1000)/BAUD) // Fehler in Promille, 1000 = kein Fehler.
23
24 #if ((BAUD_ERROR<990) || (BAUD_ERROR>1010))
25     #error Systematischer Fehler der Baudrate grösser 1% und damit zu hoch!
26 #endif
27
28 #define UART_MAXSTRLEN 30
29
30 //INCLUDE OWN HEADERS
31 #include "can.h"
32
33 //DEFINES
34 #define SCL_CLOCK 400000L
35 #define EEPROM_ADR 0xA0
36 #define WAIT_STABLE_US 5
37
38 //DEFINES PINS
39 #define CS_ASW_OUT PA0
40 #define RST_ASW_OUT PA1
41 #define CS_ASW_IN PA2
42 #define RST_ASW_IN PA3
43 #define CS_ADC_CONV PA4
44 #define SHDN_CAN_TRANS PA5
45 #define CS_DIGI_POTO PA6
46
47 typedef struct
48 {
49     uint8_t id; // TODO CONTROL + EXT
50     uint8_t zoom;
51     uint8_t taxel_data[8][8];
52 } sensor_data;
53
54 //EXTERNE FUNKTIONEN
55 //spi.c
56 extern void spi_speed_4mhz(void);
57 extern void spi_speed_8mhz(void);
58 extern uint8_t spi_write_8bit(uint8_t data);
59 //i2c.c
60 extern void i2c_init(void); // I2C Bus mit in SCL_CLOCK definierter Geschwindigkeit
    initialisieren
61 extern unsigned char i2c_start(unsigned char address); // Bus starten und Gerät ansprechen

```

```

62 extern void i2c_stop(void); // Bus Stoppen
63 extern unsigned char i2c_write(unsigned char data); // Ein Byte Senden
64 extern unsigned char i2c_readAck(void); // Ein Byte einlesen und mit Ack bestätigen
65 extern unsigned char i2c_readNak(void); // Ein Byte einlesen kein Ack
66 //io.c
67 extern void io_init(void);
68 //adc.c
69 extern uint8_t adc_read_8bit(void);
70 extern uint16_t adc_read_16bit(void);
71 //analog_sw.c
72 extern void set_sw_out(uint8_t hex_chan);
73 extern void set_sw_in(uint8_t hex_chan);
74 extern void reset_sw_out(void);
75 extern void reset_sw_in(void);
76 //digi_pot.c
77 extern void sdo_tri_state(void);
78 extern void set_resistor_val(uint8_t val);
79 //read_sensor.c
80 void read_sensor(sensor_data* sensor);
81 void send_sensor(sensor_data* sensor, can_t* msg);
82 //uart.c
83 extern void uart_init();
84 extern void uart_putc(unsigned char uart_char);
85 extern void uart_puts(char *uart_string);
86 //eeprom.c
87 extern uint8_t eeprom_read_byte(uint8_t block, uint8_t adr);
88 extern void eeprom_write_byte(uint8_t block, uint8_t adr, uint8_t byte);
89 //uart.c
90 extern volatile char uart_string[UART_MAXSTRLEN + 1];

```

```

1 //-----
2 // FIRMA           Deutsches Zentrum für Luft- und Raumfahrt
3 // ABTEILUNG       Institut für Robotik und Mechatronik
4 // PROJEKT         Erweiterung einer Auswerteelektronik für taktile Sensoren
5 //                zum multimodalen Sensorsystem
6 // Autoren        Schneider Denis
7 // Beschreibung   TacSenseEval10.c
8 // Erstellt      07.06.2012
9 //-----
10
11 #include "TacSenseEval10.h"
12
13 int main(void)
14 {
15     //System Initialisierung
16     io_init();           // I/Os Initialisieren
17     spi_speed_8mhz();   // SPI Bus initialisieren
18     sdo_tri_state();    // Set SDO Line Tristate
19     i2c_init();         // I2C Bus initialisieren
20     uart_init();        // Debug Only!!!
21     reset_sw_in();      // Power On Reset
22     reset_sw_out();     // Power On Reset
23     set_resistor_val(0xFF); // Initialisierung DigiPot (std 0x80 = 25k/25k)
24
25     // eeprom_write_byte(0,0,0x03); //EEPROM SET ID
26
27     // Can Initialisierung
28     can_init(BITRATE_1_MBPS); // CAN Bus initialisieren
29     can_set_mode(NORMAL_MODE); // Normaler Betriebsmodus
30
31     can_disable_filter(CAN_ALL_FILTER);
32
33     sei(); // Interrupts global an
34

```

```

35     sensor_data sensor;
36     sensor.id = eeprom_read_byte(0,0);
37     sensor.zoom = 0;
38
39     can_t msg;
40     msg.id = sensor.id;
41     msg.length = 8;
42
43     can_filter_t filter = {
44         .id = sensor.id, //erwartet Frame mit ID:0x???
45         .mask = 0xFFFF, //Maskierung, ignoriert alle anderen IDs
46         .flags = {
47             .rtr = 0,
48         }
49     };
50
51     can_set_filter(0, &filter);
52
53     PORTB |= (1<<PB6);
54
55     while (1)
56     {
57         //do
58         //{
59         //} while (!can_check_message());
60         //can_get_message(&msg);
61
62         PORTB ^= (1<<PB6);
63         PORTB ^= (1<<PB7);
64
65         msg.data[0] = 0xff;
66         msg.data[1] = 0xff;
67         msg.data[2] = 0x08;
68         msg.data[3] = 0x08;
69         msg.data[4] = 0x00;
70         msg.data[5] = 0x00;
71         msg.data[6] = 0x00;
72         msg.data[7] = 0x00;
73
74         can_send_message(&msg);
75
76         read_sensor(&sensor);
77         send_sensor(&sensor,&msg);
78     }
79
80     return 0;
81 }

```

```

1  //-----
2  // FIRMA           Deutsches Zentrum für Luft- und Raumfahrt
3  // ABTEILUNG       Institut für Robotik und Mechatronik
4  // PROJEKT         Erweiterung einer Auswerteelektronik für taktile Sensoren
5  //                zum multimodalen Sensorsystem
6  // Autoren         Schneider Denis
7  // Beschreibung    adc.c - Ansteuerung des ADS8319 ADC
8  // Erstellt       07.06.2012
9  //-----
10
11 #include "TacSenseEval10.h"
12
13 uint8_t adc_read_8bit(void)
14 {
15     PORTA &=~ (1<<PA4); // Chipselect ADC low
16

```

```

17     asm volatile ("nop"); // nop = einen Prozessortakt warten
18
19     PORTA |= (1<<PA4); // Chipselect ADC high
20
21     asm volatile ("nop"); asm volatile ("nop"); asm volatile ("nop");
22     asm volatile ("nop"); asm volatile ("nop"); asm volatile ("nop");
23     asm volatile ("nop"); asm volatile ("nop"); asm volatile ("nop");
24     asm volatile ("nop"); asm volatile ("nop"); asm volatile ("nop");
25     asm volatile ("nop"); asm volatile ("nop"); asm volatile ("nop");
26     asm volatile ("nop"); asm volatile ("nop"); asm volatile ("nop");
27     asm volatile ("nop"); asm volatile ("nop"); asm volatile ("nop");
28     asm volatile ("nop");
29
30     PORTA &=~ (1<<PA4);
31
32     asm volatile ("nop");
33
34     uint8_t data_high = 0;
35
36     data_high = spi_write_8bit(0x00); // Daten ueber SPI einlesen
37
38     PORTA |= (1<<PA4);
39
40     return data_high;
41 }
42
43 uint16_t adc_read_16bit(void)
44 {
45     PORTA &=~ (1<<PA4);
46
47     asm volatile ("nop");
48
49     PORTA |= (1<<PA4);
50
51     asm volatile ("nop"); asm volatile ("nop"); asm volatile ("nop");
52     asm volatile ("nop"); asm volatile ("nop"); asm volatile ("nop");
53     asm volatile ("nop"); asm volatile ("nop"); asm volatile ("nop");
54     asm volatile ("nop"); asm volatile ("nop"); asm volatile ("nop");
55     asm volatile ("nop"); asm volatile ("nop"); asm volatile ("nop");
56     asm volatile ("nop"); asm volatile ("nop"); asm volatile ("nop");
57     asm volatile ("nop"); asm volatile ("nop"); asm volatile ("nop");
58     asm volatile ("nop");
59
60     PORTA &=~ (1<<PA4);
61
62     asm volatile ("nop");
63
64     uint8_t data_high, data_low = 0;
65
66     data_high = spi_write_8bit(0x00);
67     data_low = spi_write_8bit(0x00);
68
69     PORTA |= (1<<PA4);
70
71     return (data_high<<8)+data_low;
72 }

```

```

1 //-----
2 // FIRMA           Deutsches Zentrum für Luft- und Raumfahrt
3 // ABTEILUNG       Institut für Robotik und Mechatronik
4 // PROJEKT         Erweiterung einer Auswerteelektronik für taktile Sensoren
5 //                zum multimodalen Sensorsystem
6 // Autoren         Schneider Denis
7 // Beschreibung    analog_sw - Ansteuerung des ADG1414 8fach Analogschalter

```

```

8 // Erstellt          07.06.2012
9 //-----
10
11 #include "TacSenseEval10.h"
12
13 // Channel Array
14 // 0-7 Einzellige Schalter, Channel 8 = ALLE
15 uint8_t channel[] = {0b00000001,0b00000010,0b00000100,0b00001000,0b00010000,
16                      0b00100000,0b01000000,0b10000000,0xFF};
17
18 void set_sw_out(uint8_t hex_chan)
19 {
20     PORTA &=~ (1<<CS_ASW_OUT); // Chipselect Analogschalter Treiber low
21     spi_write_8bit(channel[hex_chan]);
22     PORTA |= (1<<CS_ASW_OUT); // Chipselect Analogschalter Treiber high
23 }
24
25 void set_sw_in(uint8_t hex_chan)
26 {
27     PORTA &=~ (1<<CS_ASW_IN); // Chipselect Analogschalter Eingang low
28     spi_write_8bit(channel[hex_chan]);
29     PORTA |= (1<<CS_ASW_IN); // Chipselect Analogschalter Eingang high
30 }
31
32 void reset_sw_out(void) // Reset der Schalter
33 {
34     PORTA &=~ (1<<RST_ASW_OUT);
35     PORTA |= (1<<RST_ASW_OUT);
36 }
37
38 void reset_sw_in(void) // Reset der Schalter
39 {
40     PORTA &=~ (1<<RST_ASW_IN);
41     PORTA |= (1<<RST_ASW_IN);
42 }

```

```

1 //-----
2 // FIRMA           Deutsches Zentrum für Luft- und Raumfahrt
3 // ABTEILUNG       Institut für Robotik und Mechatronik
4 // PROJEKT         Erweiterung einer Auswerteelektronik für taktile Sensoren
5 //                zum multimodalen Sensorsystem
6 // Autoren        Schneider Denis
7 // Beschreibung    digi_pot.c - Ansteuerung des ISL23425 50k Digitalpotentiometers
8 // Erstellt       07.06.2012
9 //-----
10
11 #include "TacSenseEval10.h"
12
13 void sdo_tri_state(void)
14 {
15     spi_speed_4mhz(); //SPI Geschwindigkeit reduzieren
16     PORTA &=~ (1<<CS_DIGI_POT0); // CS für Poti low
17     spi_write_8bit(0b01110000);
18     spi_write_8bit(0b01000010);
19     PORTA |= (1<<CS_DIGI_POT0); // CS für Poti high
20     spi_speed_8mhz(); //SPI Geschwindigkeit erhöhen
21 }
22
23 void set_resistor_val(uint8_t val)
24 {
25     spi_speed_4mhz();
26     PORTA &=~ (1<<CS_DIGI_POT0); // CS für Poti low
27     spi_write_8bit(0b11000000);
28     spi_write_8bit(val);

```

Anhang

```
29 PORTA |= (1<<CS_DIGI_POT0); //CS für Poti high
30 spi_speed_8mhz();
31 }
```

```
1 //-----
2 // FIRMA           Deutsches Zentrum für Luft- und Raumfahrt
3 // ABTEILUNG       Institut für Robotik und Mechatronik
4 // PROJEKT         Erweiterung einer Auswertelektronik für taktile Sensoren
5 //               zum multimodalen Sensorsystem
6 // Autoren         Schneider Denis
7 // Beschreibung    eeprom.c - Ansteuerung des externen EEPROMs
8 // Erstellt        07.06.2012
9 //-----
10
11 #include "TacSenseEval10.h"
12
13 uint8_t eeprom_read_byte(uint8_t block, uint8_t adr)
14 {
15     uint8_t byte = 0;
16     block = block << 1;
17     i2c_start(EEPROM_ADR+block);
18     i2c_write(adr);
19     i2c_start(EEPROM_ADR+1+block);
20     byte = i2c_readNak();
21     i2c_stop();
22     return byte;
23     _delay_us(2);
24 }
25
26 void eeprom_write_byte(uint8_t block, uint8_t adr, uint8_t byte)
27 {
28     block = block << 1;
29     i2c_start(EEPROM_ADR+block);
30     i2c_write(adr);
31     i2c_write(byte);
32     i2c_stop();
33     _delay_ms(5);
34 }
```

```
1 //-----
2 // FIRMA           Deutsches Zentrum für Luft- und Raumfahrt
3 // ABTEILUNG       Institut für Robotik und Mechatronik
4 // PROJEKT         Erweiterung einer Auswertelektronik für taktile Sensoren
5 //               zum multimodalen Sensorsystem
6 // Autoren         Schneider Denis
7 // Beschreibung    sensor.c - Routinen zur Auswertung des taktile Sensoren
8 // Erstellt        07.06.2012
9 //-----
10
11 #include "TacSenseEval10.h"
12
13 void read_sensor(sensor_data* sensor)
14 {
15     switch (sensor->zoom)
16     {
17         case 0: // Alle Taxel einlesen
18             for(uint8_t j=0;j<8;j++)
19             {
20                 set_sw_out(j);
21                 for(uint8_t i=0;i<8;i++)
22                 {
23                     set_sw_in(i);
24                     _delay_us(WAIT_STABLE_US);
25                     sensor->taxel_data[i][j] = adc_read_8bit();
26                 }
27             }
28     }
29 }
```

```

26         }
27     }
28     break;
29
30     case 1: // Quadranten (Taxelzahl / 4)
31     break;
32
33     case 2: // Alle zu einem verschalten
34     set_sw_out(8);
35     set_sw_in(8);
36     sensor->taxel_data[0][0] = adc_read_8bit();
37     break;
38 }
39 }
40
41 void send_sensor(sensor_data* sensor, can_t* msg)
42 {
43     switch (sensor->zoom)
44     {
45         case 0: // Alle Taxel Einlesen
46         for(uint8_t j=0;j<8;j++)
47         {
48             _delay_us(130);
49             for(uint8_t i=0;i<8;i++)
50             {
51                 msg->data[i] = sensor->taxel_data[i][j];
52             }
53             can_send_message(msg);
54         }
55         break;
56
57         case 1: // Quadranten (Taxelzahl / 4)
58         break;
59
60         case 2: // Alle zu einem verschalten
61         msg->data[0] = sensor->taxel_data[0][0];
62         can_send_message(msg);
63         break;
64     }
65 }

```

```

1 // -----
2 /*
3  * Copyright (c) 2007 Fabian Greif, Roboterclub Aachen e.V.
4  * All rights reserved.
5  *
6  * Redistribution and use in source and binary forms, with or without
7  * modification, are permitted provided that the following conditions
8  * are met:
9  * 1. Redistributions of source code must retain the above copyright
10 * notice, this list of conditions and the following disclaimer.
11 * 2. Redistributions in binary form must reproduce the above copyright
12 * notice, this list of conditions and the following disclaimer in the
13 * documentation and/or other materials provided with the distribution.
14 *
15 * THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ‘‘AS IS’’ AND
16 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
17 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
18 * ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
19 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
20 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
21 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
22 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
23 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY

```

```

24 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
25 * SUCH DAMAGE.
26 */
27 // -----
28 /**
29 * \file    can.h
30 * \brief   Header-Datei für das allgemeine CAN Interface
31 */
32 // -----
33
34 #ifndef CAN_H
35 #define CAN_H
36
37 #if defined (__cplusplus)
38     extern "C" {
39 #endif
40
41 // -----
42 /**
43 * \ingroup    communication
44 * \defgroup   can_interface Universelles CAN Interface
45 * \brief      allgemeines CAN Interface für AT90CAN32/64/128, MCP2515 und
46 *             SJA1000
47 *
48 * \author     Fabian Greif <fabian.greif@rwth-aachen.de>
49 * \author     Roboterclub Aachen e.V. (http://www.roboterclub.rwth-aachen.de)
50 *
51 * \can_sleep() and can_wakeup() functions by Frédéric Lamorce.
52 *
53 * \version    $Id: can.h 8086 2009-07-14 14:08:25Z fabian $
54 */
55 // -----
56 #include <avr/pgmspace.h>
57 #include <stdint.h>
58 #include <stdbool.h>
59
60 #include "config.h"
61
62 // -----
63 /** \ingroup    can_interface
64 * \name        Bitdefinitionen
65 */
66 //@{
67 #define ONLY_NON_RTR        2
68 #define ONLY_RTR            3
69 //@}
70
71 /** \ingroup    can_interface
72 * \brief      Bitraten fuer den CAN-Bus
73 */
74 typedef enum {
75     BITRATE_10_KBPS = 0,    // ungetestet
76     BITRATE_20_KBPS = 1,    // ungetestet
77     BITRATE_50_KBPS = 2,    // ungetestet
78     BITRATE_100_KBPS = 3,   // ungetestet
79     BITRATE_125_KBPS = 4,
80     BITRATE_250_KBPS = 5,   // ungetestet
81     BITRATE_500_KBPS = 6,   // ungetestet
82     BITRATE_1_MBPS = 7,
83 } can_bitrate_t;
84
85 /**
86 * \ingroup    can_interface
87 * \brief      Symbol um auf alle Filter zuzugreifen

```

```

88  */
89 #define CAN_ALL_FILTER          0xff
90
91 /**
92  * \ingroup      can_interface
93  * \brief       Unterstuetzung fuer Extended-IDs aktivieren
94  */
95 #ifndef SUPPORT_EXTENDED_CANID
96     #define SUPPORT_EXTENDED_CANID  1
97 #endif
98
99 /**
100  * \ingroup      can_interface
101  * \brief       Unterstützung für Zeitstempel aktivieren
102  * \warning     Wird nur vom AT90CANxxx unterstützt
103  */
104 #ifndef SUPPORT_TIMESTAMPS
105     #define SUPPORT_TIMESTAMPS      1
106 #endif
107
108 #if defined(__DOXYGEN__)
109
110     #define MCP2515_FILTER_EXTENDED(id)
111     #define MCP2515_FILTER(id)
112
113 #else
114
115     #if SUPPORT_EXTENDED_CANID
116         #define MCP2515_FILTER_EXTENDED(id) \
117             (uint8_t) ((uint32_t) (id) >> 21), \
118             (uint8_t) (((uint32_t) (id) >> 13) & 0xe0) | (1<<3) | \
119                 (((uint32_t) (id) >> 16) & 0x3)), \
120             (uint8_t) ((uint32_t) (id) >> 8), \
121             (uint8_t) ((uint32_t) (id))
122     #endif
123
124     #define MCP2515_FILTER(id) \
125         (uint8_t)((uint32_t) id >> 3), \
126         (uint8_t)((uint32_t) id << 5), \
127         0, \
128         0
129 #endif
130 //@}
131 // -----
132 /**
133  * \ingroup      can_interface
134  * \brief       Datenstruktur zum Aufnehmen von CAN Nachrichten
135  */
136 typedef struct
137 {
138     #if SUPPORT_EXTENDED_CANID
139         uint32_t id;                //!< ID der Nachricht (11 oder 29
140             Bit)
141         struct {
142             int rtr : 1;           //!< Remote-Transmit-Request-Frame
143             ?
144             int extended : 1;     //!< extended ID?
145         } flags;
146     #else
147         uint16_t id;               //!< ID der Nachricht (11 Bit)
148         struct {
149             int rtr : 1;           //!< Remote-Transmit-Request-Frame
150             ?
151         } flags;
152     #endif
153 }

```

```
150
151     uint8_t length;                //!< Anzahl der Datenbytes
152     uint8_t data[8];              //!< Die Daten der CAN Nachricht
153
154     #if SUPPORT_TIMESTAMPS
155         uint16_t timestamp;
156     #endif
157 } can_t;
158
159
160
161 // -----
162 /**
163  * \ingroup    can_interface
164  * \brief     Datenstruktur zur Aufnahme von CAN-Filtern
165  *
166  * \code
167  * rtr | Funtion
168  * ----|-----
169  * 00 | alle Nachrichten unabhaengig vom RTR-Bit
170  * 01 | ungültig
171  * 10 | empfangen nur nicht RTR-Nachrichten
172  * 11 | empfangen nur Nachrichten mit gesetztem RTR-Bit
173  * \endcode
174  *
175  * \b ACHTUNG:
176  * Funktioniert nur mit dem AT90CAN, beim MCP2515 wird der Parameter ignoriert.
177  *
178  * \code
179  * ext | Funtion
180  * ----|-----
181  * 00 | alle Nachrichten
182  * 01 | ungueltig
183  * 10 | empfangen nur Standard-Nachrichten
184  * 11 | empfangen nur Extended-Nachrichten
185  * \endcode
186  *
187  * \warning   Filter sind beim SJA1000 nur begrenzt nutzbar, man sollte ihn nur
188  *            in Systemen mit entweder Standard- oder Extended-Frames einsetzen
189  *            ,
190  *            aber nicht beidem zusammen.
191  */
192 typedef struct
193 {
194     #if SUPPORT_EXTENDED_CANID
195         uint32_t id;                //!< ID der Nachricht (11 oder 29
196         Bit)
197         uint32_t mask;              //!< Maske
198         struct {
199             uint8_t rtr : 2;        //!< Remote Request Frame
200             uint8_t extended : 2;   //!< extended ID
201         } flags;
202     #else
203         uint16_t id;                //!< ID der Nachricht 11 Bits
204         uint16_t mask;              //!< Maske
205         struct {
206             uint8_t rtr : 2;        //!< Remote Request Frame
207         } flags;
208     #endif
209 } can_filter_t;
210
211 // -----
212 /**
```

```

213 * \ingroup can_interface
214 * \brief Inhalt der Fehler-Register
215 */
216 typedef struct {
217     uint8_t rx;                //!< Empfangs-Register
218     uint8_t tx;                //!< Sende-Register
219 } can_error_register_t;
220
221 // -----
222 /**
223 * \ingroup can_interface
224 * \brief Modus des CAN Interfaces
225 */
226 typedef enum {
227     LISTEN_ONLY_MODE,          //!< der CAN Contoller empfängt nur und verhält
                                sich völlig passiv
228     LOOPBACK_MODE,            //!< alle Nachrichten direkt auf die
                                Empfangsregister umleiten ohne sie zu senden
229     NORMAL_MODE,              //!< normaler Modus, CAN Controller ist
                                aktiv
230     SLEEP_MODE                 // sleep mode
231 } can_mode_t;
232
233 // -----
234 /**
235 * \ingroup can_interface
236 * \brief Initialisierung des CAN Interfaces
237 *
238 * \param bitrate Gewuenschte Geschwindigkeit des CAN Interfaces
239 *
240 * \return false falls das CAN Interface nicht initialisiert werden konnte,
241 *         true ansonsten.
242 */
243 //*****
244 // Method: can_init
245 // FullName: can_init
246 // Access: private
247 // Returns: extern bool
248 // Qualifier:
249 // Parameter: can_bitrate_t bitrate
250 //*****
251 extern bool
252 can_init(can_bitrate_t bitrate);
253
254 // -----
255 /**
256 * \ingroup can_interface
257 *
258 * ~english
259 * \brief Put CAN interface to sleep and wake up
260 *
261 * \code
262 * // before we are going to sleep, enable the interrupt that will wake us up
263 // attach interrupt 1 to the routine
264 EICRA = 0; // int on low level
265 // Enable the interrupt1
266 EIMSK = _BV(INT1);
267
268 // put the MCP2515 to sleep
269 can_sleep();
270
271 // enable atmega sleep mode
272 cli();
273 sleep_enable();
274 // turn off BOD

```

```
275 sleep_bod_disable();
276 // and we go to sleep
277 sei();
278 sleep_cpu();
279
280 // here int1 has been executed and we are woken up
281 sleep_disable();
282
283 // disable int1
284 EIMSK = 0;
285
286 // re-enable 2515 and 2551
287 can_wake();
288 * \endcode
289 *
290 * \warning      Only implemented for the MCP2515
291 */
292 extern void
293 can_sleep(void);
294
295 extern void
296 can_wakeup(void);
297
298 // -----
299 /**
300 * \ingroup      can_interface
301 * \brief       Setzen eines Filters
302 *
303 * Für einen MCP2515 sollte die Funktion can_static_filter() bevorzugt werden.
304 *
305 * \param       number  Position des Filters
306 * \param       filter  zu setzender Filter
307 *
308 * \return      false falls ein Fehler auftrat, true ansonsten
309 */
310 extern bool
311 can_set_filter(uint8_t number, const can_filter_t *filter);
312
313 // -----
314 /**
315 * \ingroup      can_interface
316 * \brief       Filter deaktivieren
317 *
318 * \param       number  Nummer des Filters der deaktiviert werden soll,
319 *                   0xff deaktiviert alle Filter.
320 * \return      false falls ein Fehler auftrat, true ansonsten
321 *
322 * \warning     Wird nur vom AT90CAN32/64/128 unterstuetzt.
323 */
324 extern bool
325 can_disable_filter(uint8_t number);
326
327 // -----
328 /**
329 * \ingroup      can_interface
330 * \brief       Setzt die Werte für alle Filter
331 *
332 * \code
333 * // Filter und Masken-Tabelle anlegen
334 * prog_char can_filter[] = {
335 *     MCP2515_FILTER_EXTENDED(0),    // Filter 0
336 *     MCP2515_FILTER_EXTENDED(0),    // Filter 1
337 *
338 *     MCP2515_FILTER_EXTENDED(0),    // Filter 2
339 *     MCP2515_FILTER_EXTENDED(0),    // Filter 3
```

```
340 *     MCP2515_FILTER_EXTENDED(0),    // Filter 4
341 *     MCP2515_FILTER_EXTENDED(0),    // Filter 5
342 *
343 *     MCP2515_FILTER_EXTENDED(0),    // Maske 0
344 *     MCP2515_FILTER_EXTENDED(0),    // Maske 1
345 * };
346 *
347 * ...
348 *
349 * // Filter und Masken-Tabelle laden
350 * can_static_filter(can_filter);
351 * \endcode
352 *
353 * \param      *filter_array  Array im Flash des AVR's mit den Initialisierungs-
354 *                                     werten für die Filter des MCP2515
355 *
356 * \see        MCP2515_FILTER_EXTENDED()
357 * \see        MCP2515_FILTER()
358 * \warning    Wird nur vom MCP2515 unterstuetzt.
359 */
360 extern void
361 can_static_filter(const prog_uint8_t *filter_array);
362
363 // -----
364 /**
365 * \ingroup    can_interface
366 *
367 * \~german
368 * \brief      Filterdaten auslesen
369 *
370 * \param      number  Nummer des Filters dessen Daten man haben moechte
371 * \param      *filter Pointer in den die Filterstruktur geschrieben wird
372 *
373 * \return     \b 0 falls ein Fehler auftrat, \
374 *             \b 1 falls der Filter korrekt gelesen werden konnte, \
375 *             \b 2 falls der Filter im Moment nicht verwendet wird (nur AT90CAN)
376 *             , \
377 *             \b 0xff falls gerade keine Aussage moeglich ist (nur AT90CAN).
378 *
379 * \warning    Da der SJA1000 nicht feststellen kann ob der ausgelesene Filter
380 *             nun zwei 11-Bit Filter oder ein 29-Bit Filter ist werden nicht
381 *             die Filter sondern die Registerinhalte direkt zurück gegeben.
382 *             Der Programmierer muss dann selbst entscheiden was er mit den
383 *             Werten macht.
384 *
385 * \~english
386 * \warning    SJA1000 doesn't return the filter and id directly but the content
387 *             of the corresponding registers because it is not possible to
388 *             check the type of the filter.
389 */
390 extern uint8_t
391 can_get_filter(uint8_t number, can_filter_t *filter);
392
393 // -----
394 /**
395 * \ingroup    can_interface
396 * \brief      Ueberpruefen ob neue CAN Nachrichten vorhanden sind
397 *
398 * \return     true falls neue Nachrichten verfuegbar sind, false ansonsten.
399 */
400 extern bool
401 can_check_message(void);
402
403 // -----
404 /**
```

```
404 * \ingroup      can_interface
405 * \brief       Ueberprueft ob ein Puffer zum Versenden einer Nachricht frei ist.
406 *
407 * \return      true falls ein Sende-Puffer frei ist, false ansonsten.
408 */
409 extern bool
410 can_check_free_buffer(void);
411
412 // -----
413 /**
414 * \ingroup      can_interface
415 * \brief       Verschickt eine Nachricht über den CAN Bus
416 *
417 * \param       msg      Nachricht die verschickt werden soll
418 * \return      FALSE falls die Nachricht nicht verschickt werden konnte, \n
419 *              ansonsten der Code des Puffes in den die Nachricht gespeichert
420 *              wurde
421 */
422 extern uint8_t
423 can_send_message(const can_t *msg);
424
425 // -----
426 /**
427 * \ingroup      can_interface
428 * \brief       Liest eine Nachricht aus den Empfangspuffern des CAN Controllers
429 *
430 * \param       msg      Pointer auf die Nachricht die gelesen werden soll.
431 * \return      FALSE falls die Nachricht nicht ausgelesen konnte,
432 *              ansonsten Filtercode welcher die Nachricht akzeptiert hat.
433 */
434 extern uint8_t
435 can_get_message(can_t *msg);
436
437 // -----
438 /**
439 * \ingroup      can_interface
440 *
441 * \~german
442 * \brief       Liest den Inhalt der Fehler-Register
443 *
444 * \~english
445 * \brief       Reads the Contents of the CAN Error Counter
446 */
447 extern can_error_register_t
448 can_read_error_register(void);
449
450 // -----
451 /**
452 * \ingroup      can_interface
453 *
454 * \~german
455 * \brief       Überprüft ob der CAN Controller im Bus-Off-Status
456 *
457 * \return      true wenn der Bus-Off-Status aktiv ist, false ansonsten
458 *
459 * \warning     aktuell nur auf dem SJA1000
460 */
461 extern bool
462 can_check_bus_off(void);
463
464 // -----
465 /**
466 * \ingroup      can_interface
467 *
468 * \~german
```

```

468 * \brief      Setzt einen Bus-Off Status zurück und schaltet den CAN Controller
469 *              wieder aktiv
470 *
471 * \warning    aktuell nur auf dem SJA1000
472 */
473 extern void
474 can_reset_bus_off(void);
475
476 // -----
477 /**
478 * \ingroup    can_interface
479 * \brief      Setzt den Operations-Modus
480 *
481 * \param     mode    Gewünschter Modus des CAN Controllers
482 */
483 extern void
484 can_set_mode(can_mode_t mode);
485
486 #if defined (__cplusplus)
487 }
488 #endif
489
490 #endif // CAN_H

```

```

1 //-----
2 // FIRMA      Deutsches Zentrum für Luft- und Raumfahrt
3 // ABTEILUNG  Institut für Robotik und Mechatronik
4 // PROJEKT    Erweiterung einer Auswerteelektronik für taktile Sensoren
5 //           zum multimodalen Sensorsystem
6 // Autoren   Schneider Denis
7 // Beschreibung i2c.c - Kommunikationsroutinen des TWI Interfaces
8 // Erstellt  07.06.2012
9 //-----
10
11 #include "TacSenseEval10.h"
12
13 void i2c_init(void) //I2C Bus mit in SCL_CLOCK definierter Geschwindigkeit initialisieren
14 {
15     TWSR = 0;
16     TWBR = ((F_CPU/SCL_CLOCK)-16)/2;
17 }
18
19 unsigned char i2c_start(unsigned char address) //Bus starten und Gerät mit address
20         ansprechen
21 {
22     uint8_t  twst;
23
24     // START Kondition setzten
25     TWCR = (1<<TWINT) | (1<<TWSTA) | (1<<TWEN);
26
27     // Warten bis Uebertragung abgeschlossen ist
28     while(!(TWCR & (1<<TWINT)));
29
30     // Wert in TWI Status Register ueberpruefen. Prescaler bits maskieren
31     twst = TW_STATUS & 0xF8;
32     if ( (twst != TW_START) && (twst != TW_REP_START)) return 1;
33
34     // Device Adresse senden
35     TWDR = address;
36     TWCR = (1<<TWINT) | (1<<TWEN);
37
38     // Warten auf Abschluss der Uebertragung und ACK/NACK empfangen wird
39     while(!(TWCR & (1<<TWINT)));

```

```

40 // Wert in TWI Status Register ueberpruefen. Prescaler Bits maskieren
41 twst = TW_STATUS & 0xF8;
42 if ( (twst != TW_MT_SLA_ACK) && (twst != TW_MR_SLA_ACK) ) return 1;
43
44 return 0;
45 }
46
47 void i2c_stop(void) //Bus stoppen
48 {
49 // Stopbedingung senden
50 TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWSTO);
51
52 // Warten bis STOP Bedingung umgesetzt ist und der Bus freigegeben wird
53 while(TWCR & (1<<TWSTO));
54
55 }
56
57 unsigned char i2c_write( unsigned char data ) //Ein Byte senden
58 {
59     uint8_t twst;
60
61 // Daten an vorher adressiertes Device senden
62 TWDR = data;
63 TWCR = (1<<TWINT) | (1<<TWEN);
64
65 // Warten auf Abschluss der Uebertragung
66 while(!(TWCR & (1<<TWINT)));
67
68 // Wert in TWI Status Register ueberpruefen. Prescaler bits maskieren
69 twst = TW_STATUS & 0xF8;
70 if( twst != TW_MT_DATA_ACK) return 1;
71 return 0;
72 }
73
74 unsigned char i2c_readAck(void) //Ein Byte einlesen und mit Ack bestaetigen
75 {
76     TWCR = (1<<TWINT) | (1<<TWEN) | (1<<TWEA);
77     while(!(TWCR & (1<<TWINT)));
78
79     return TWDR;
80 }
81
82 unsigned char i2c_readNak(void) //Ein Byte einlesen keine Ack Bestaetigung
83 {
84     TWCR = (1<<TWINT) | (1<<TWEN);
85     while(!(TWCR & (1<<TWINT)));
86
87     return TWDR;
88 }

```

```

1 //-----
2 // FIRMA           Deutsches Zentrum für Luft- und Raumfahrt
3 // ABTEILUNG       Institut für Robotik und Mechatronik
4 // PROJEKT         Erweiterung einer Auswerteelektronik für taktile Sensoren
5 //               zum multimodalen Sensorsystem
6 // Autoren        Schneider Denis
7 // Beschreibung   io.c - Setzen der I/O Funktion
8 // Erstellt      07.06.2012
9 //-----
10
11 #include "TacSenseEval10.h"
12
13 void io_init(void)
14 {

```

```

15     DDRB &=~ (1<<PB3);
16     PORTB |= (1<<PB3); //Pullup an PB3
17     DDRB |= (1<<PB0) | (1<<PB1) | (1<<PB2); //SS, MOSI, SCK -> Ausgang
18
19     DDRA = 0xFF; //PORTA auf Ausgang für Chipselects
20
21     PORTA |= (1<<CS_ASW_OUT) | (1<<RST_ASW_OUT) | (1<<CS_ASW_IN)
22     | (1<<RST_ASW_IN) | (1<<CS_ADC_CONV) | (1<<CS_DIGI_POT0); //Alle Chip Selects 1,
        COVST für ADC nicht, SHDN_CAN_TRANS nicht
23
24     DDRB |= (1<<PB6)|(1<<PB7); //LEDs als Ausgänge setzen
25     PORTG |= (1<<PING0)|(1<<PING1); //Pullups für Schalter an
26 }

```

```

1 //-----
2 // FIRMA           Deutsches Zentrum für Luft- und Raumfahrt
3 // ABTEILUNG       Institut für Robotik und Mechatronik
4 // PROJEKT         Erweiterung einer Auswerteelektronik für taktile Sensoren
5 //                zum multimodalen Sensorsystem
6 // Autoren        Schneider Denis
7 // Beschreibung    spi.c - Kommunikationsroutinen des SPI
8 // Erstellt       07.06.2012
9 //-----
10
11 #include "TacSenseEval10.h"
12
13 void spi_speed_4mhz() // 4Mhz SPI für Digital Poti (max. 5Mhz)
14 {
15     SPCR |= (1<<SPE)|(1<<MSTR); //set
16     SPSR &=~ (1<<SPI2X);
17 }
18
19 void spi_speed_8mhz() // 8Mhz SPI für ADC / AN_SW
20 {
21     SPCR |= (1<<SPE)|(1<<MSTR); //set
22     SPSR |= (1<<SPI2X);
23 }
24
25 uint8_t spi_write_8bit(uint8_t data)
26 {
27     // Uebertragung beginnen
28     SPDR = data;
29
30     // Warten auf Abschluss der Uebertragung
31     uint8_t kill = 0; // Zähler um SPI bei fehlender Antwort freizugeben
32
33     while(!(SPSR & (1<<SPIF)) && kill < 255) // Ende wenn Daten übertragen oder
        Schleife hängt
34     {
35         kill++;
36     }
37     return (SPDR);
38 }

```

```

1 //-----
2 // FIRMA           Deutsches Zentrum für Luft- und Raumfahrt
3 // ABTEILUNG       Institut für Robotik und Mechatronik
4 // PROJEKT         Erweiterung einer Auswerteelektronik für taktile Sensoren
5 //                zum multimodalen Sensorsystem
6 // Autoren        Schneider Denis
7 // Beschreibung    uart.c - UART Kommunikation (Quelle: www.mikrocontroller.net)
8 // Erstellt       07.06.2012
9 //-----
10

```

```
11 #include "TacSenseEval10.h"
12
13 volatile uint8_t uart_str_complete = 0;    // 1 = String komplett empfangen
14 volatile uint8_t uart_str_count = 0;
15 volatile char uart_string[UART_MAXSTRLEN + 1] = "test\n";
16
17 ISR(USART0_RXC_vect)
18 {
19     unsigned char nextChar;
20
21     // Daten aus dem Puffer lesen
22     nextChar = UDR0;
23     if(uart_str_complete == 0) // wenn uart_string gerade in Verwendung, neues Zeichen
        verwerfen
24     {
25         // Daten werden erst in uart_string geschrieben, wenn nicht String-Ende/max
        Zeichenlänge
26         // erreicht ist/string gerade verarbeitet wird
27         if(nextChar != '\n' && nextChar != '\r' && uart_str_count < UART_MAXSTRLEN -
        1)
28             {
29                 uart_string[uart_str_count] = nextChar;
30                 uart_str_count++;
31             }
32         else
33             {
34                 uart_string[uart_str_count] = '\0';
35                 uart_str_count = 0;
36                 uart_str_complete = 1;
37             }
38     }
39 }
40
41 void uart_init()
42 {
43     UBRROH = UBRR_VAL >> 8;
44     UBRROL = UBRR_VAL & 0xFF;
45
46     UCSROB = (1<<TXEN0)|(1<<RXEN0)|(1<<RXCIE0); // UART TX & RX einschalten
47     UCSROC = (1<<UCSZ01)|(1<<UCSZ00); // Asynchron 8N1
48 }
49
50 void uart_putc(unsigned char uart_char)
51 {
52     while (!(UCSROA & (1<<UDRE0))) /* warten bis Senden moeglich */
53     {
54     }
55     UDR0 = uart_char;
56 }
57
58 void uart_puts(char *uart_string)
59 {
60     while (*uart_string)
61     { /* so lange *uart_string != '\0' */
62         uart_putc(*uart_string);
63         uart_string++;
64     }
65 }
```