Entwicklung und Realisierung einer Steuerung für ein Staubkonzentrationsmessgerät

F-05-1601

Entwicklung und Realisierung einer Steuerung für ein Staubkonzentrationsmessgerät

Bachelorarbeit

von

Pavlo Chupin

Wilhelm Büchner Hochschule
Fachbereich Ingenieurwissenschaften

und

Forschungsgesellschaft für angewandte Systemsicherheit und Arbeitsmedizin

vorgelegt bei:	Dr.-Ing. Thomas Beier
von:	Pavlo Chupin
Matr.-Nr.:	885557
Anschrift:	S 3, 4, 68161, Mannheim
Abgabetermin:	31.10.2016

Eidesstattliche Erklärung

Pavlo Chupin, Matrikel-Nummer: 885557

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch noch nicht veröffentlicht.

Mannheim, 28. Oktober 2016

(Pavlo Chupin)

Inhaltsverzeichnis

Liste der Symbole

Symbol	Einheit	Bedeutung
BU1, BU2, ...	-	Anschlussbuchsen
c	g/m^3	Staubkonzentration
c_{5-7}	g/m^3	Staubkonzentration, gemessen mit dem SKG 5 Typ 7
c_{7-18}	g/m^3	Staubkonzentration, gemessen mit dem SKG 7 Typ 18
C1, C2, ...	μF	Kondensatoren bzw. Kapazitäten
D_n	-	digitales Signal
F	%	relative Luftfeuchte
I	mA	elektrischer Strom allgemein
I_0	mA	Strom durch die Sendediode, welcher die Lichtintensität Φ_0 an dieser hervorruft
$I_{Empfaenger}$	mA	Strom durch den Empfänger
I_{in}	mA	Eingangsstrom
I_{mess}	mA	mit dem Multimeter gemessener Strom durch den Empfänger
I_{offset}	mA	Offsetstrom
I_{Sender}	mA	vom SKG 7 angezeigter Strom durch den Sender
$I_{Sender,ist}$	mA	tatsächlicher Strom durch den Sender
$I_{Sender,soll}$	mA	vorgegebener Strom durch den Sender
I_{SKG}	mA	vom SKG 7 angezeigter Strom durch den Empfänger
I_{max}	mA	maximaler Strom durch den Sender
IC1, IC2, ...	-	integrierte Schaltkreise
l	m	Länge des Messvolumens beim Staubkonzentrationsmessgerät
OP1, OP2, ...	-	Operationsverstärker
r^2	-	Korrelationskoeffizient bei der Regressionsanalyse
R1, R2, ...	Ω	Widerstände
R_{sense}	Ω	Messwiderstand
t	s	Zeit
T	°C	Temperatur
T1, T2, ...	-	Transistoren

U	V	elektrische Spannung allgemein
U^{+}	V	positive elektrische Spannung
$\mathsf{U}_{\mathsf{in}}^{+}$	V	positive Eingangsspannung
$\mathsf{U}_{\mathsf{in}}^{-}$	V	negative Eingangsspannung
U_{in}	V	Eingangsspannung
$\mathsf{U}_{\mathsf{out}}$	V	Ausgangsspannung
$\mathsf{U}_{\mathsf{ref}}$	V	Referenzspannung
Δc_{abs}	g/m^3	absolute Differenz zwischen c_{5-7} und c_{7-18}
Δc_{rel}	%	relative Differenz zwischen c_{5-7} und c_{7-18}
$\Delta\mathsf{I}$	%	relative Differenz zwischen $\mathsf{I}_{\mathsf{Sender,soll}}$ und $\mathsf{I}_{\mathsf{Sender,ist}}$
$\Delta\mathsf{I}_{\mathsf{abs}}$	mA	absolute Differenz zwischen $\mathsf{I}_{\mathsf{SKG}}$ und $\mathsf{I}_{\mathsf{mess}}$
$\Delta\mathsf{I}_{\mathsf{rel}}$	%	relative Differenz zwischen $\mathsf{I}_{\mathsf{SKG}}$ und $\mathsf{I}_{\mathsf{mess}}$
ϵ	$\text{m}^2\text{/g}$	Extinktionskoeffizient
Φ	W/m^2	Intensität des transmittierten Lichtes
Φ_0	W/m^2	Intensität des emittierten Lichtes

Vorwort

Um die Nachfolge des in den Ruhestand getretenen Dipl.-Ing. (FH) Helmut Ott im Fachbereich Physik/Technik des Zentrallabors der Berufsgenossenschaft Nahrungsmittel und Gastgewerbe (BGN) antreten zu können, ermöglichte mir die BGN, den Fernstudiengang Elektro- und Informationstechnik an der Wilhelm Büchner Hochschule in Darmstadt durchzuführen. Durch den Charakter des Fernstudiums konnte ich meine Arbeit bei der BGN mit dem Studium verbinden. Für die Eröffnung dieser Möglichkeit und die großzügige Beteiligung der BGN an den Kosten des Studiums danke ich dem Hauptgeschäftsführer der BGN, Herrn Klaus Marsch, herzlichst.

Für das gestellte Thema im Rahmen eines Entwicklungsprojektes der Forschungsgesellschaft für angewandte Systemsicherheit und Arbeitsmedizin e. V. (FSA), deren federführendes Mitglied die BGN ist, danke ich Herrn Prof. Dr. Siegfried Radandt.

Großer Dank gilt Herrn Dr. Thomas Beier für die Betreuung der Arbeit von Seiten der Wilhelm Bücher Hochschule.

Mein besonderer Dank gilt Herrn Dirk Lorenz für die Unterstützung in Form von Rat und Diskussionen sowie für die Übernahme der Betreuung dieser Arbeit von Seiten der FSA.

Weiterhin möchte ich mich bei Frau Brigitte Wehling und den Herren Hermann Schiebler und Michael Seithel bedanken sowie bei all jenen Mitarbeitern und Mitarbeiterinnen der BGN und FSA, die mich während meiner Bachelorarbeit in irgendeiner Form tatkräftig unterstützten.

Zusammenfassung

Die Berufsgenossenschaft Nahrungsmittel und Gastgewerbe (BGN) und die Forschungsgesellschaft für angewandte Systemsicherheit und Arbeitsmedizin e. V. (FSA) haben ein Staubkonzentrationsmessgerät (SKG) entwickelt, um Staub/Luft-Gemische aus brennbarem Staub auf Explosionsfähigkeit zu untersuchen. Das Gerät misst im Konzentrationsbereich von wenigen g/m^3 bis wenige kg/m^3. Es besteht aus einer Messeinheit (Sensor), die sich während der Messung in der Staubwolke befindet, und aus einer Steuereinheit zum Betreiben der Messeinheit und zum Anzeigen der Messwerte in Form einer elektrischen Spannung.
Im Bestreben, das SKG weiter zu optimieren, ist die Messeinheit Typ 18 entstanden. Gegenstand dieser Arbeit ist die Entwicklung einer dazu passenden und zeitgemäßen Steuereinheit, wobei dabei teilweise auf bereits vorhandene Technik zurückgegriffen wird. Die neue Steuereinheit mit der Bezeichnung SKG 7 zeichnet sich gegenüber den älteren Geräten durch folgende Neuerungen aus:

1. Das Gerät verfügt über ein Touchdisplay. Alle Ein- und Ausgaben werden darüber menügeführt getätigt.
2. Das Gerät zeigt die Staubkonzentration direkt an und nicht in Form von elektrischen Spannungswerten.
3. Der vor jeder Messung erforderliche Nullabgleich erfolgt automatisch und nicht manuell über Potentiometer.
4. Das Gerät misst die Temperatur und die relative Feuchte der Umgebungsluft.

Das SKG 7 Typ 18 wird gemäß den Vorgaben erfolgreich aufgebaut und getestet.

Kapitel 1

Einleitung

1.1 Staubexplosionen

In vielen Industriezweigen werden pulver- oder staubförmige Produkte verarbeitet oder entstehen während des Produktionsprozesses. Vom überwiegenden Teil dieser Stäube geht Brand- und Explosionsgefahr aus. 80 % aller in der Industrie vorkommenden Stäube sind brennbar und bereits eine Staubschicht von 1 mm Dicke kann in einem geschlossenen Raum – nach Aufwirbelung und Zündung – eine Staubexplosion auslösen. In der Vergangenheit kam es aufgrund dessen immer wieder zu Staubexplosionen, die im Vergleich zu Gasexplosionen oft verheerender sind. Bei Gasexplosionen sorgt der entstehende Explosionsdruck für eine rasche Ausbreitung der Gaswolke und somit für eine Verdünnung des Gas/Luft-Gemisches. Wird kein weiteres Gas zugeführt, ist die Explosion nach einigen Millisekunden auch wieder vorbei. Bei einer Staubexplosion können durch die Druckwelle eventuell vorhandene abgelagerte Staubschichten aufgewirbelt und entzündet werden. So kann es zu einer Explosionsausbreitung durch ganze Gebäude- und Anlagenteile und zu deren Zerstörung kommen [10].

Um Arbeitnehmer vor den Folgen von Staubexplosionen zu schützen, aber auch um Kosten durch Produktionsausfall oder zerstörte Anlagen einzusparen, werden die Ursachen von Staubexplosionen und -bränden untersucht. Zur Erhöhung der Kenntnisse sind nach wie vor intensive Forschungsarbeiten zwingend erforderlich [3].

Staubexplosionen sind möglich, wenn drei Voraussetzungen gleichzeitig auftreten:

1. Sauerstoffhaltige Gasatmosphäre (in der Regel Luft)
2. Brennbarer Staub, aufgewirbelt in ausreichender Menge in der sauerstoffhaltigen Gasatmosphäre (siehe Abschnitt 1.2)
3. Wirksame Zündquelle

Liegt eine explosionsfähige Atmosphäre vor, d. h. ein brennbarer Stoff feinverteilt in (Luft-)Sauerstoff innerhalb seiner Explosionsgrenzen, kann es durch eine Zündquelle (zum Beispiel heiße Oberflächen, Funken, die durch elektrostatische Aufladung oder mechanische Art entstehen, oder offenes Feuer) zu einer Explosion kommen. Abbildung 1.1 zeigt eine vereinfachte grafische Darstellung dieses Sachverhaltes [1, 3, 7].

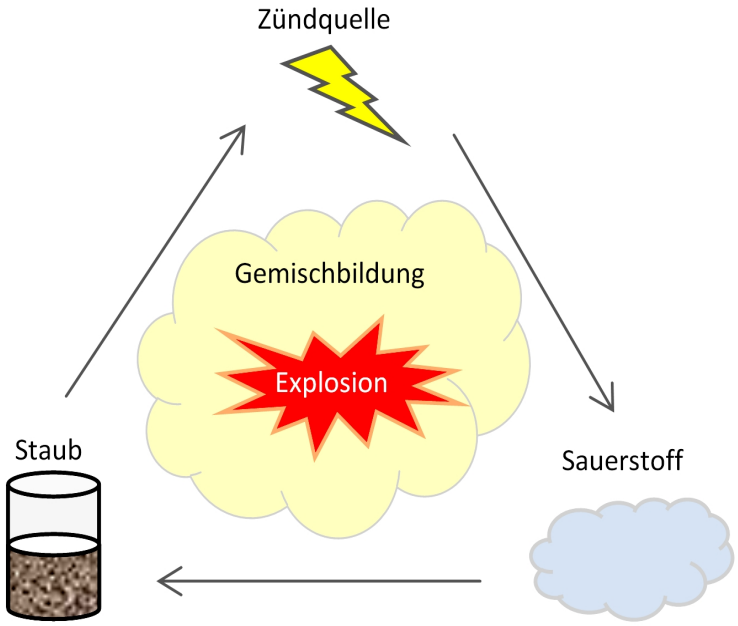

Abbildung 1.1: Bildliche Darstellung des Gefahrendreiecks

1.2 Staubkonzentration als wichtiger Parameter des Staubexplosionsschutzes

Staubexplosionen entstehen, wenn Sauerstoff, eine Zündquelle und brennbarer Staub in Form einer Staubwolke gleichzeitig auftreten. Dabei muss die Konzentration des brennbaren Staubes in der Luft innerhalb der sogenannten unteren Explosionsgrenze (UEG) und oberen Explosionsgrenze (OEG) liegen [10].

Die untere bzw. obere Explosionsgrenze gibt den unteren bzw. oberen Grenzwert der Konzentration eines brennbaren Staubes in einem Gemisch aus diesem Staub mit Luft an, bei dem sich nach dem Zünden eine von der Zündquelle unabhängige Flamme gerade nicht mehr selbständig fortpflanzen kann. Bei Staubkonzentrationen unterhalb der unteren Explosionsgrenze liegt zu wenig Brennstoff, bei Staubkonzentrationen oberhalb der oberen Explosionsgrenze zu viel Brennstoff vor. Der Staubkonzentrationsbereich, der zwischen der UEG und OEG liegt, wird *Explosionsbereich* genannt. Bei Staubkonzentrationen, die außerhalb des Explosionsbereiches liegen, ist keine Zündwilligkeit mehr gegeben und damit eine Explosionsgefährdung ausgeschlossen (Abbildung 1.2) [3]. Die Kenntnis der Staubkonzentrationen in der Luft in Silos, Gebäuden oder Anlagen ist deshalb von fundamentaler Bedeutung wenn auf dem Gebiet der Staubexplosionen geforscht wird oder etwa Risikobetrachtungen bezüglich Staubexplosionen angestellt werden sollen [10].

Für brennbare Stäube umfasst der Explosionsbereich Staubkonzentrationen von etwa $15\,g/m^3$ bis wenige kg/m^3. Für diesen Konzentrationsbereich gab es lange Zeit keine brauchbaren Messverfahren. Weil ein Arbeitsschwerpunkt der BGN bzw. FSA die Untersuchung von Staubexplosionen sowie der angewandte Staubexplosionsschutz

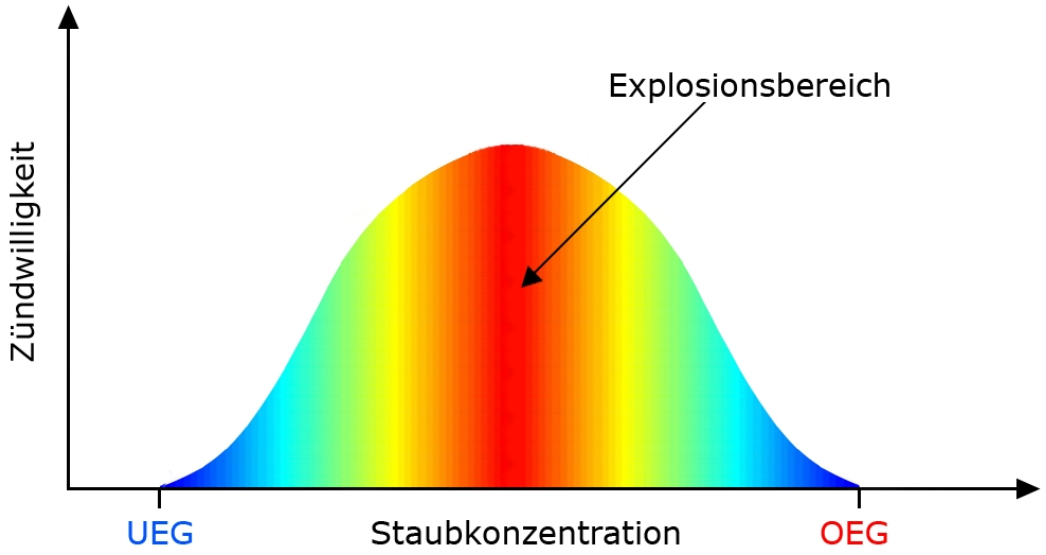

Abbildung 1.2: Bildliche Darstellung des Explosionsbereichs und der Explosionsgrenzen.

ist, wurde unter Federführung der FSA vor mehr als 25 Jahren ein spezielles Staubkonzentrationsmessgerät (SKG) entwickelt. Dieses Gerät wurde über die Jahre stetig weiterentwickelt [3, 5, 7, 10].

1.3 Zielsetzung

Zu Beginn dieser Arbeit lag die Messeinheit Typ 18 vor (siehe Kapitel 2). Mit dem Begriff Messeinheit wird der Sensor oder Messkopf des Staubkonzentrationsmessgerätes bezeichnet. Typ 18 bedeutet, dass es sich um die 18. Variante (Weiterentwicklung) der Messeinheit handelt. Der Autor hatte diese im Rahmen seiner Tätigkeit bei der FSA zuvor aus dem Typ 17 entwickelt. Zu dieser Messeinheit Typ 18 gab es keine passende Steuereinheit zur Auswertung und Anzeige der von der Messeinheit kommenden Messsignale. Die Steuereinheiten tragen die Bezeichnung SKG gefolgt von der Nummer ihrer Entwicklungsstufe, z. B. zuletzt das SKG 6. Ziel dieser Arbeit war es deshalb, eine Lösung zur Steuerung der Messeinheit Typ 18 zu entwickeln, welche gegenüber den älteren Steuereinheitstypen einige elektronische Neuerungen aufweisen sollte:

1. Das Gerät verfügt über ein Display zur direkten Anzeige der Staubkonzentration sowie über ein Eingabefeld zum Austausch notwendiger Daten zwischen Gerät und Bediener[1].

2. Mit Hilfe einer elektronischen Steuerung wird das Sendersignal geregelt und das Empfängersignal aufgearbeitet.

[1] Alle älteren Steuereinheitstypen werden durchweg manuell über Potentiometer eingestellt. Die Messwerte werden über ein Zahlendisplay in der Einheit Millivolt angezeigt.

3. Vor jeder Messung ist ein Nullabgleich erforderlich, der mit der neuen Messeinheit automatisch erfolgt.

4. Die Temperatur und Feuchte der Umgebungsluft wird parallel zur Staubkonzentrationsmessung messtechnisch erfasst und angezeigt.

Kapitel 2

Das Staubkonzentrationsmessgerät

2.1 Allgemeines

Lange Zeit war das SKG 5 Typ 7 im Einsatz (Abbildung 2.1, links). Bei den Messungen mit diesem Gerät traten immer wieder die beiden Schwierigkeiten auf, dass zum einen die Messeinheit aufgrund ihrer Größe nicht oder nur an ungünstiger Stelle positioniert werden konnte und zum anderen die optischen Linsen der Messeinheit schnell verschmutzten. Die Lösung dieser Probleme führte zu Weiterentwicklungen des Systems bis zum SKG 5 Typ 15 [3, 7]. Die letzte größere Weiterentwicklung des Staubkonzentrationsmessgerätes wurde von [10] durchgeführt. Ziel war es hierbei,

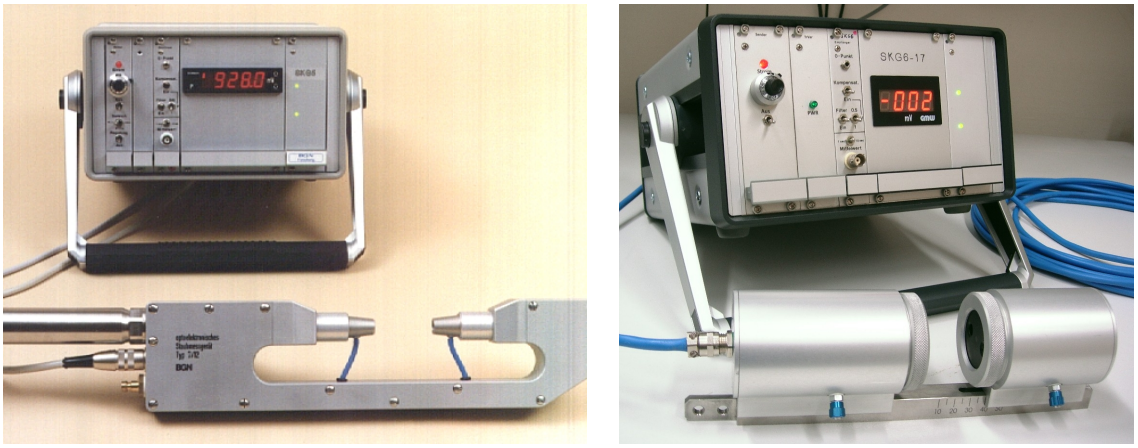

Abbildung 2.1: SKG 5 Typ 7 (links) und SKG 6 Typ 17 (rechts).

das Gerät für den Einsatz in explosionsgefährdeten Zonen vorzubereiten. Dies führte zum SKG 6 Typ 17. Weil bei diesem Gerät häufig erhebliche Störsignale auftraten, wurde es vom Autor dieser Arbeit im Rahmen seiner Tätigkeit bei der FSA einer genauen Analyse unterzogen. Es zeigte sich, dass die Störsignale aufgrund von kapazitiven Einflüssen der langen Verbindungsleitung zwischen Messeinheit und Steuereinheit herrührten. Zur Behebung dieses Problems wurde die Verstärkung des Messsignals von der Steuereinheit in die Messeinheit verlegt. Es entstand die Messeinheit Typ 18.

2.2 Messprinzip

Zur Staubkonzentrationsmessung befindet sich die Messeinheit im Staub/Luft-Gemisch. Die Messeinheit erfasst die Messdaten und leitet sie über ein Kabel zur Steuereinheit weiter. Das Messprinzip basiert auf der Schwächung eines Lichtstrahls beim Durchlaufen einer Staubwolke. Dieses Prinzip wird auch Transmissionsverfahren genannt. In der Messeinheit Typ 18 befinden sich nebeneinander angeordnet in einem Gehäuse ein Sender und ein Empfänger. Der Sender emittiert einen Lichtstrahl einer bestimmten Intensität Φ_0. Der emittierte Lichtstrahl läuft durch das Staub/Luft-Gemisch zu einem Prisma. Am Prisma wird der Lichtstrahl um 180° umgelenkt und läuft durch das Staub/Luft-Gemisch zurück zum Empfänger. Der Empfänger erfasst die durch das Staub/Luft-Gemisch reduzierte Lichtintensität Φ. Dieses Prinzip zur Messung der Staubkonzentration ist in Abbildung 2.2 dargestellt.

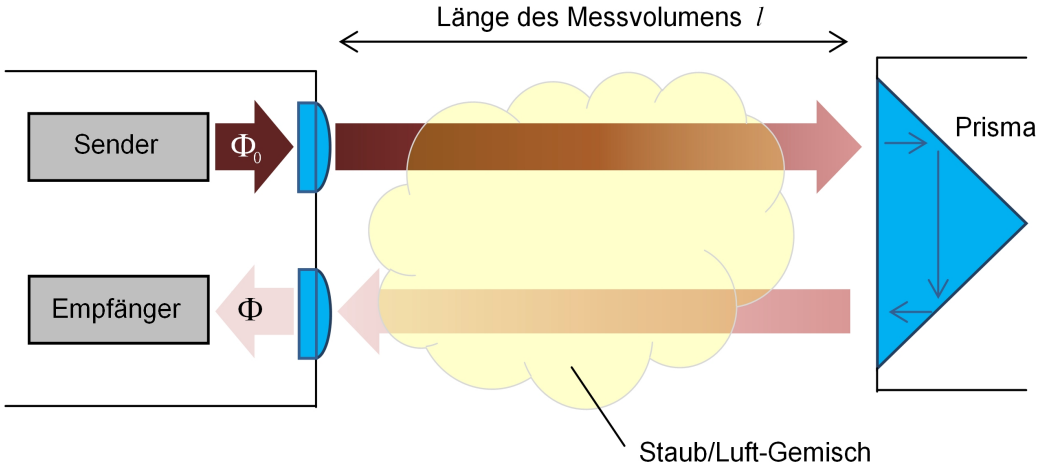

Abbildung 2.2: Schematisch dargestelltes Messprinzip der Staubkonzentrationsmessung. Der Empfänger erfasst die durch das Staub/Luft-Gemisch reduzierte Lichtintensität Φ.

Der Anteil der durchgelassenen Strahlungsintensität Φ hängt über das Lambert-Beer-Gesetz

$$\Phi = \Phi_0 \cdot e^{-\epsilon \cdot 2l \cdot c} \tag{2.1}$$

mit der vom Sender abgestrahlten Lichtintensität Φ_0 zusammen. Durch Umstellen und Logarithmieren erhält man daraus eine Funktion $c(\Phi_0, \Phi, l, \epsilon)$, die die Staubkonzentration c in Abhängigkeit von den Lichtintensitäten Φ_0 und Φ, dem Extinktionskoeffizienten ϵ und der halben Länge des Lichtweges durch das Staub/Luft-Gemisch l beschreibt:

$$c = -\frac{\ln\left(\frac{\Phi}{\Phi_0}\right)}{\epsilon \cdot 2l} \tag{2.2}$$

Außer dem Extinktionskoeffizienten ϵ, der über eine Kalibriermessung am zu prüfenden Staub ermittelt werden muss, sind alle anderen Größen aus (2.2) durch die Einstellungen am Staubkonzentrationsmessgerät und durch den Messvorgang bekannt.

2.3 Aufbau der Messeinheit Typ 18

Die Messeinheit Typ 18 ist in Abbildung 2.3 als CAD-Grafik dargestellt. Abbildung 2.4 zeigt den schematischen Aufbau der Messeinheit. Die Messeinheit Typ 18 besteht aus zwei zylindrischen Gehäusen, die auf einer Stahlschiene befestigt sind. In einem Gehäuse befinden sich die optoelektronischen Elemente, die zum Messen der Lichtschwächung durch den Staub bzw. die Bestimmung der Staubkonzentration nötig sind. Dies sind die Sende- und Empfänger- bzw. Fotodiode, die beide in einem schmalen Wellenlängenbereich um 950 nm arbeiten. Vor beiden Dioden befindet sich jeweils eine Linse zur Bündelung des Lichtes. Im zweiten Gehäuse sitzt ein Dreiecksprisma für die 180°-Umlenkung des Lichtstrahls.

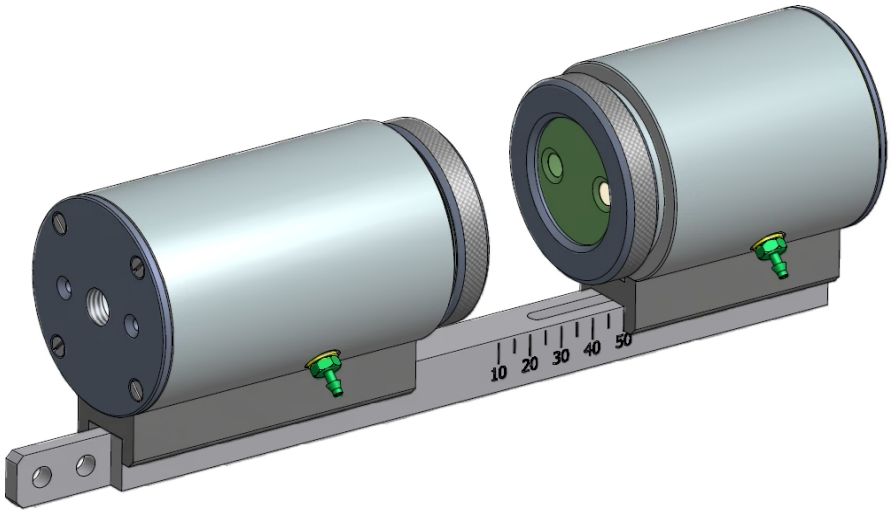

Abbildung 2.3: Ansicht der Messeinheit Typ 18 (CAD-Grafik) [12].

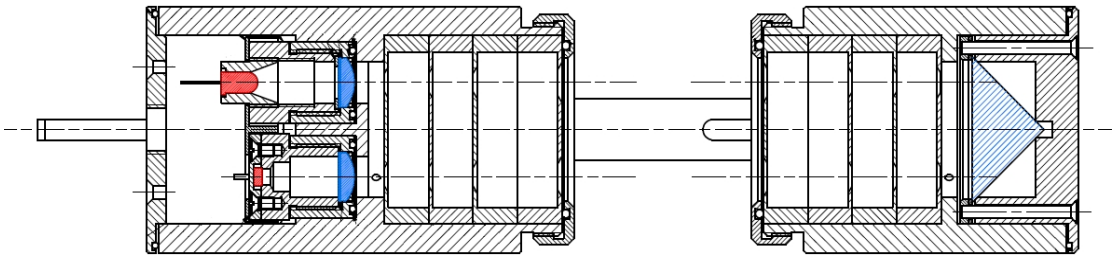

Abbildung 2.4: Längsschnitt durch die Messeinheit Typ 18 [12].

Zum Schutz vor Staubablagerungen auf den optischen Bauteilen (Linsen und Prisma) sind vor diesen sogenannte Blendenblöcke angeordnet. Jeder dieser beiden Blendenblöcke besteht aus vier Blenden, die jeweils zwei Bohrungen für den Durchtritt des Lichtstrahls besitzen (Abbildung 2.5). Die Blenden werden durch je eine Rändel-Überwurfmutter in den beiden zylindrischen Gehäusen gehalten. Durch das Abschrauben dieser Überwurfmuttern können die Blenden entnommen und gesäubert werden. Auch die optischen Bauteile werden so zugänglich für Reinigungszwecke.

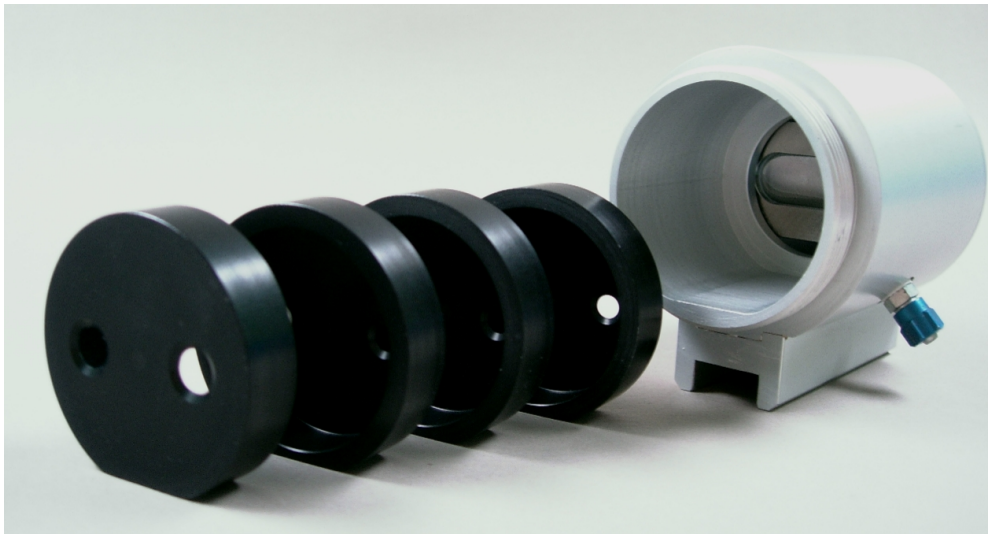

Abbildung 2.5: Die in die Zylindergehäuse eingebauten Blenden schützen die optischen Elemente der Messeinheit vor Verschmutzung.

Im Gehäuse, welches den Infrarot-Sender und -Empfänger sowie die Linsen enthält, befindet sich weiterhin eine Verstärkerschaltung. Die Spannungsversorgung für dieses elektronische Bauteil und für die optoelektronischen Bauteile sowie das Auslesen der Messdaten erfolgt über eine 10 m lange, vierpolige Leitung von der Steuereinheit aus. Über zwei Adern der Leitung wird der Senderdiodenstrom im Bereich von 0 mA bis 20 mA geregelt. Über die verbleibenden beiden Adern wird das Fotodiodensignal ausgelesen. Dieses Signal beträgt je nach empfangener Lichtintensität Werte von 0 mA bis 20 mA. Die Messeinheit ist standardmäßig folgendermaßen eingestellt:

- Der Senderdiodenstrom beträgt 8 mA.
- Der Abstand l zwischen den sich gegenüberstehenden Stirnflächen der beiden Zylindergehäuse beträgt 50 mm.
- Es befindet sich kein Staub im Messvolumen $\Rightarrow$ Der gemessene Fotodiodenstrom beträgt 20 mA.

2.4 Kalibrierung

Wie in Abschnitt 2.2 im letzten Absatz dargelegt wurde, ist für die Staubkonzentrationsmessung die Kenntnis des materialspezifischen Extinktionskoeffizienten ϵ zwingend erforderlich. Dieser muss für jede Staubart durch ein Kalibrierverfahren bestimmt werden.
Dazu wird in einem ersten Schritt ein infrarotdurchlässiges Kalibriergefäß, welches auf einer Rührplattform steht, in den Strahlengang der Messeinheit eingebracht (Abbildung 2.6). In das Kalibriergefäß wird ein Volumen von 120 ml Ethanol (96 %) gefüllt. Die Messeinheit wird so in der Halterung justiert, dass das Kalibriergefäß mitsamt Ethanol optimal durchstrahlt wird, was sich an einem maximalen Empfängersignal erkennen lässt. Anschließend wird ein Nullableich durchgeführt, d. h. die Lichtintensität

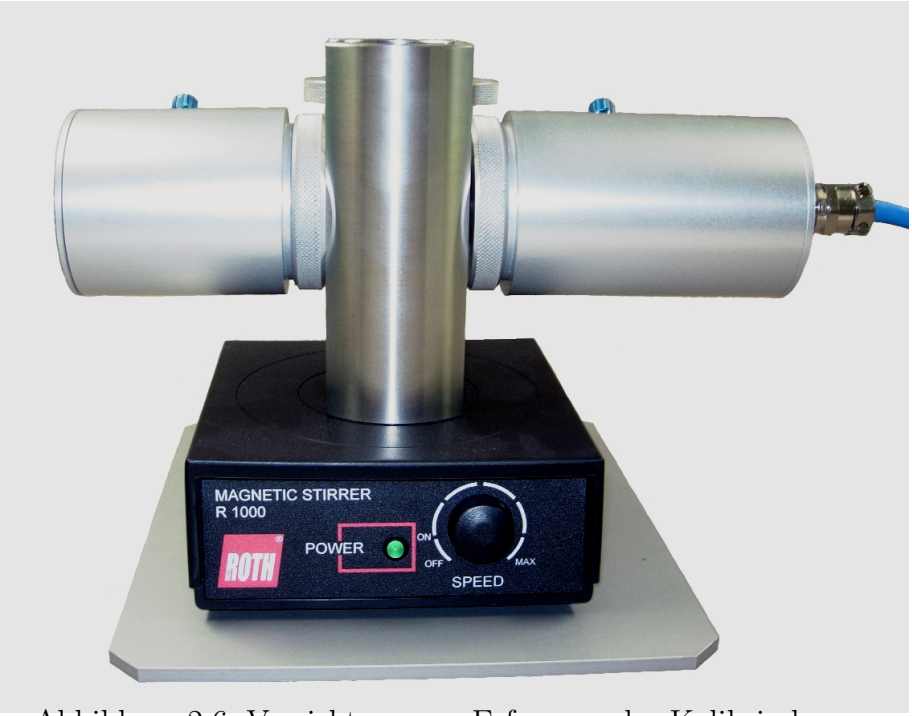

Abbildung 2.6: Vorrichtung zur Erfassung der Kalibrierkurve.

Φ_0 eingestellt. Weil bei den älteren Staubkonzentrationsmessgeräten die Lichtintensität in Form eines Spannungswertes angezeigt wird, geschieht hier der Nullabgleich, indem die Leistung der Sendediode so lange an einem Potentiometer reguliert wird, bis auf dem Steuergerät 1.000 mV angezeigt werden. Die Messeinheit Typ 18 liefert hingegen ein Stromsignal, weshalb auf 20 mA geregelt wird.
Der zweite Schritt besteht darin, mit der Präzisionslaborwaage Staubportionen von 2×3 mg, 6 mg, 2×12 mg und 36 mg abzuwiegen, die dann sukzessiv unter ständigem Rühren in das Ethanol gegeben werden. Auf diese Weise wird die Staubkonzentration im Kalibriergefäß von 0 g/m^3 über 25 g/m^3, 50 g/m^3, 100 g/m^3, 200 g/m^3, 300 g/m^3 bis 600 g/m^3 systematisch erhöht. Nach jeder Zugabe einer Staubportion wird das Ethanol/Staub-Gemisch kurz und kräftig aufgerührt, um alle Partikel gleichmäßig im gesamten Volumen zu verteilen. Zum Ablesen des Messwertes wird der Rührer auf einen kleineren Wert geschaltet, damit nicht versehentlich der durch das Rühren entstehende Flüssigkeitskegel in den Strahlengang der Messeinheit gerät. Dieser Vorgang muss möglichst zügig geschehen, da die Teilchen zu sedimentieren beginnen bzw. unter Umständen vom Ethanol in ihren Eigenschaften verändert werden. Letzteres betrifft vor allem kristalline Stäube, bestehend aus transparenten Körnern, weil das Ethanol deren Brechungsindex verändern kann [7].
Für jede eingestellte Konzentration c wird die Lichtintensität Φ durch Ablesung des proportionalen Spannungs- oder Strommesswertes ermittelt. Die Messwerte werden über der Staubkonzentration aufgetragen. Es entstehen Kalibrierpunkte. In diese Kalibrierpunkte wird die Gleichung (2.1) mit einer rechnergestützten Regressionsanalyse eingepasst. Weil sowohl die Länge l als auch die vom Sender emmitierte Lichtintensität Φ_0 bekannt ist, entspricht das Einpassen der Gleichung (2.1) dem Aufsuchen des richtigen Extinktionskoeffizienten ϵ. Ein Beispiel für solch eine Kalibrierkurve ist in Abbildung 2.7 zu sehen.

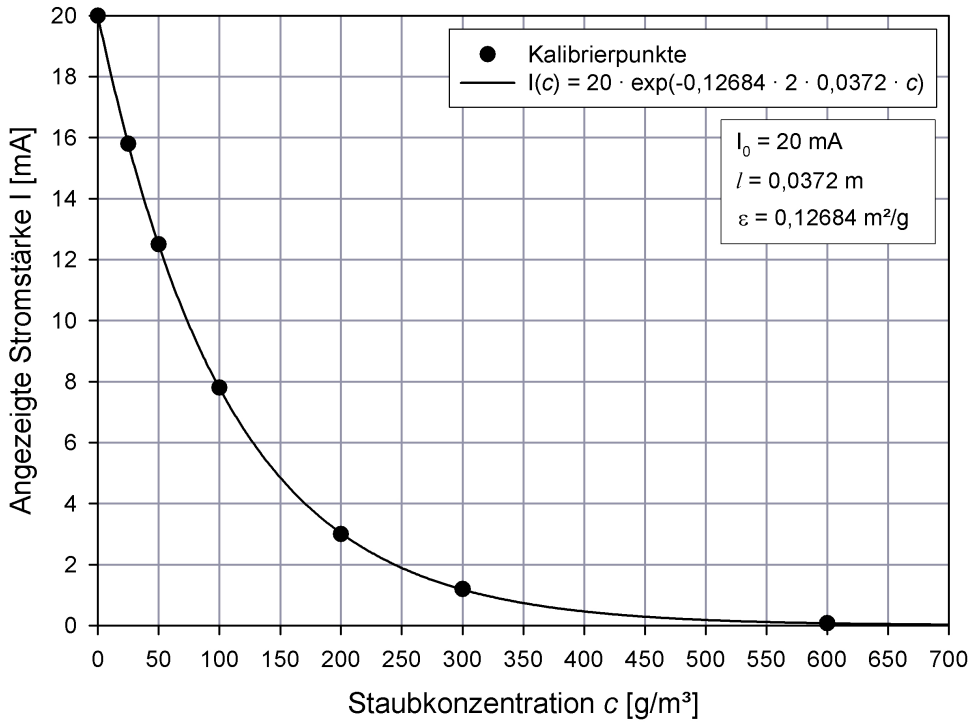

Abbildung 2.7: Ergebnis der Kalibrierkurvenerstellung für Maisstärke. Es wurde ein Extinktionskoeffizient von $\epsilon = 0,12684\,\mathrm{m}^2/\mathrm{g}$ errechnet.

Für Stäube, die sich suspendiert in Ethanol stark verändern oder lösen, wurde ein alternatives Kalibrierverfahren entwickelt, welches in [13] detailliert beschrieben ist.

Kapitel 3

Entwicklung der Steuereinheit

3.1 Analyse der Ausgangssituation

Zu Beginn der Arbeit lag die im Kapitel 2.3 beschriebene Messeinheit Typ 18 vor. Die Messeinheit besitzt eine Infrarot-Leuchtdiode (Sender), eine Fotodiode, die im Wellenlängenbereich des Senders empfindlich ist (Empfänger) und eine nach- bzw. vorgeschaltete Elektronik zur Signalverstärkung und zur Vermeidung von Signalstörungen. Der Sender kann mit einem Strom von 0 mA bis 20 mA gespeist werden. Der Empfänger bzw. seine Verstärkerschaltung gibt proportional zur empfangenen Lichtintensität einen Strom von 0 mA bis 20 mA ab. Es war eine Lösung zur Steuerung der Messeinheit und zur Auswertung der von ihr gelieferten Messdaten zu entwickeln. Dabei mussten einige Vorgaben an das neu entstehende Staubkonzentrationsmessgerät berücksichtigt werden, die von Seiten der FSA gefordert wurden.

3.2 Anforderungen an das Messsystem

3.2.1 Vorgaben durch die FSA

Das zu entwickelnde Staubkonzentrationsmessgerät sollte laut Vorgabe der FSA folgende Funktionen besitzen:

1. Die Staubkonzentration wird direkt angezeigt.

2. Es können beliebige Stäube untersucht werden (Kalibriermöglichkeit).

3. Der vor jeder Messung erforderliche Nullabgleich wird automatisch durchgeführt.

4. Temperatur und relative Luftfeuchte der Umgebungsluft werden gemessen und angezeigt.

5. Die an das Gerät angeschlossene Messeinheit Typ 18 kann in Zone 20[1] betrieben werden.
6. Das Gerät verfügt über eine Datenbank für Stäube und deren Extinktionskoeffizienten.
7. Das System muss kompakt und mobil einsetzbar sein.
8. Die Herstellungskosten (exklusive Arbeitsstunden) liegen unter 5.000 Euro.
9. Das System kann von einem Bediener ohne fundierte Kenntnisse der Messtechnik bedient werden.
10. Die Steuereinheit verfügt über einen Datenspeicher.
11. Messdaten können komfortabel auf einen Computer übertragen werden.
12. Die Darstellung der zeitlichen Entwicklung der Staubkonzentration ist möglich.
13. Die im Gerät integrierte Staubdatenbank ist erweiterbar.

Die Erfüllung der gesamten 13 Anforderungen konnte in der für die Bachelorarbeit vorgesehenen Zeit von maximal 6 Monaten (inklusive 2 Monate Verlängerung) mit hoher Wahrscheinlichkeit nicht erreicht werden. Die FSA versah die Anforderungen 1. bis 9. mit der höchsten Priorität, weshalb sie als verbindliche Ziele für diese Arbeit ausgewählt wurden. Die Punkte 10. bis 13. blieben als optionale Ziele.

3.2.2 Anforderungen an die Steuerung

Aus den im vorigen Abschnitt 3.2.1 aufgelisteten (extrahierten) Vorgaben der FSA und den Eigenschaften der vorliegenden Messeinheit Typ 18 ergaben sich folgende Aufgaben bzw. Anforderungen für die Entwicklung einer Steuerung:

1. Der Strom durch den Sender musste im Bereich 0 mA - 20 mA regelbar sein.
2. Die Messung des Empfängerstroms von 0 mA - 20 mA musste realisiert werden.
3. Die Messdaten mussten für die Auswertung aufbereitet werden (Filterung).
4. Ein automatischer Nullabgleich musste entwickelt werden.
5. Das Messsystem musste für beliebige Stäube kalibrierbar sein.
6. Die Berechnung der Staubkonzentration musste geräteintern durchgeführt werden.
7. Ein System für Temperatur- und Feuchtemessung musste entwickelt und integriert werden.

[1] Zone 20 ist ein Bereich, in dem explosionsfähige Staub/Luft-Gemische häufig und über lange Zeiträume vorkommen.

8. Eine Speichermöglichkeit für eine Staubdatenbank musste geschaffen werden.
9. Eine Datenanzeige musste vorhanden sein.
10. Die Gerätegröße war zu beachten.
11. Die Kosten für Beschaffung und Herstellung waren zu beachten.
12. Die Bedienerfreundlichkeit musste sichergestellt werden.
13. Der Explosionsschutz musste berücksichtigt werden.

3.3 Konzeptentwicklung

3.3.1 Analoge Messung und Steuerung

Prinzipiell könnte die Messeinheit Typ 18 schon mit einem Labornetzteil und einem Multimeter betrieben werden. Das Netzteil speist Sender und Empfänger. Mit Hilfe eines Potentiometers könnte der erforderliche Strom durch den Sender (0 mA - 20 mA) eingestellt werden. Die Messung der Stromstärke im Empfängerstromkreis könnte mit einem Multimeter erfolgen. Eine solche Lösung würde allerdings nur die FSA-Vorgaben 2., 7. und 8. erfüllen und wurde deshalb nie in Erwägung gezogen.

Plausibler war dagegen, die neue Steuereinheit weitgehend auf dem vorhandenen SKG 6 aufzubauen. Die Funktionsweise des Gerätes ist in [10] beschrieben. Das SKG 6 erfüllt bereits die FSA-Anforderungen 2., 5., 7., 8. und 9.. Alle anderen Vorgaben (bis auf 4.) sind jedoch mit analoger Technik nicht oder nur mit sehr großem zeitlichen Aufwand realisierbar. Deshalb wurde auch dieser Lösungsansatz verworfen.

3.3.2 Rechnergestützte Mess- und Regelverfahren

Dank der enormen Rechenleistung moderner Digitalrechner finden rechnergestützte Messsysteme heutzutage eine sehr breite Verwendung. Rechnergestützte Messsysteme besitzen folgende Grundfunktionen [4]:

- Messwerterfassung
- Messwertverarbeitung
- Messwert- bzw. Messergebnissausgabe

Einige Vorteile solcher Systeme sind [6]:

- Automatisierung kompletter Messabläufe
- Sichere und effiziente Speicherung von Messdaten

- Ersatz vom komplexen und an eine spezielle Aufgabe gebundenen Hardware-Komponenten durch flexible Software-Module, z. B. bei der Filterung
- Gute Visualisierungsmöglichkeiten

Die Struktur eines rechnergestützten Systems zur Datenerfassung und Steuerung ist in Abbildung 3.1 gezeigt.

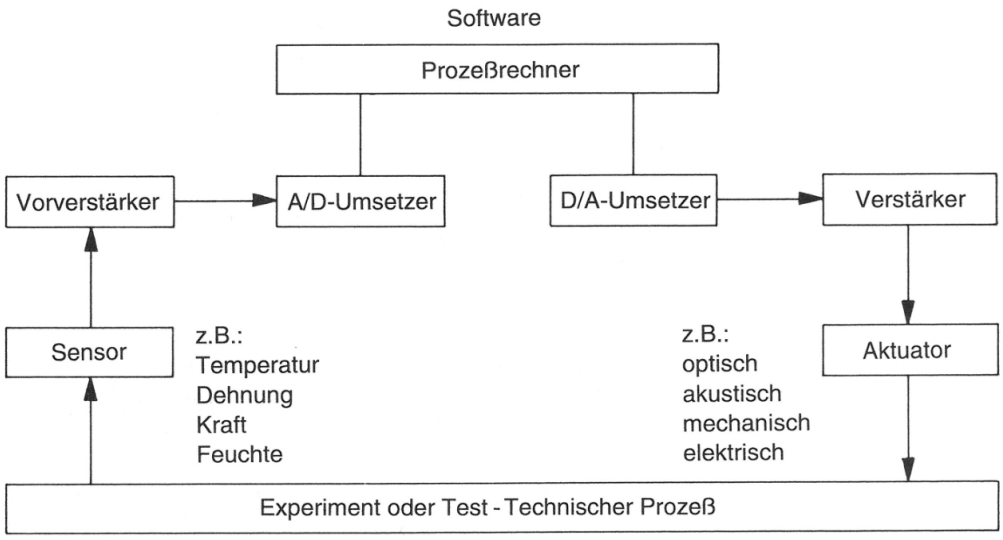

Abbildung 3.1: Struktur eines rechnergestützten Datenerfassungssystems (Bild entnommen aus [14]).

PC-Messkarten und Systeme, die über serielle Schnittstelle mit dem Rechner kommunizieren, gelten heute als standardisierte Lösungen in der Messdatenerfassung (Abbildung 3.2). Nach der Erfassung werden die Daten durch ein spezielles Anwenderprogramm (z. B. LabVIEW oder MATLAB), welches auf dem PC installiert ist, weiterverarbeitet. Die Messeinheit Typ 18 könnte an ein solches Messsystem angeschlossen werden, wie in Abbildung 3.3 dargestellt ist. Das Messsystem könnte das Empfängersignal erfassen und den Senderstrom einstellen. Die aufgenommenen Daten könnten gefiltert und nach der Auswertung dargestellt werden. Die Filterung, Auswertung und die Darstellung der Messdaten wären durch die Verwendung einer spezifischen Software sehr flexibel. Mit dem System könnten sowohl ein automatischer Nullabgleich, die Kalibrierung auf verschiedene Stäube als auch Konzentrationsmessungen durchgeführt werden. Die Anwendersoftware kann auf jedem Rechner installiert werden, an den die Datenerfassungshardware angeschlossen ist. Ein solches System erfüllt alle FSA-Anforderungen bis auf die Punkt 5. und 8.. Wegen Punkt 7. müsste ein Laptop beschafft werden, der obendrein für eine raue Messumgebung geeignet sein muss (Kosten: ca. 1.500 Euro). Eine geeignete PC-Messkarte bewegt sich im Anschaffungspreis in der gleichen Größenordnung. Der Kauf einer Steuer- und Auswertesoftware schlägt noch einmal mit 4.000 Euro zu Buche, wie eine Preisanfrage bei National Instruments nach der Software LabVIEW ergab. Insgesamt würden also Kosten von rund 7.000 Euro entstehen, was das vorgegebene Budget von 5.000 Euro deutlich überschreitet. Dazu kämen weitere Kosten für die Umsetzung der Forderung, dass die Messeinheit in Zone 20 eingesetzt werden kann. Aus diesem Grund wurde diese Lösung verworfen.

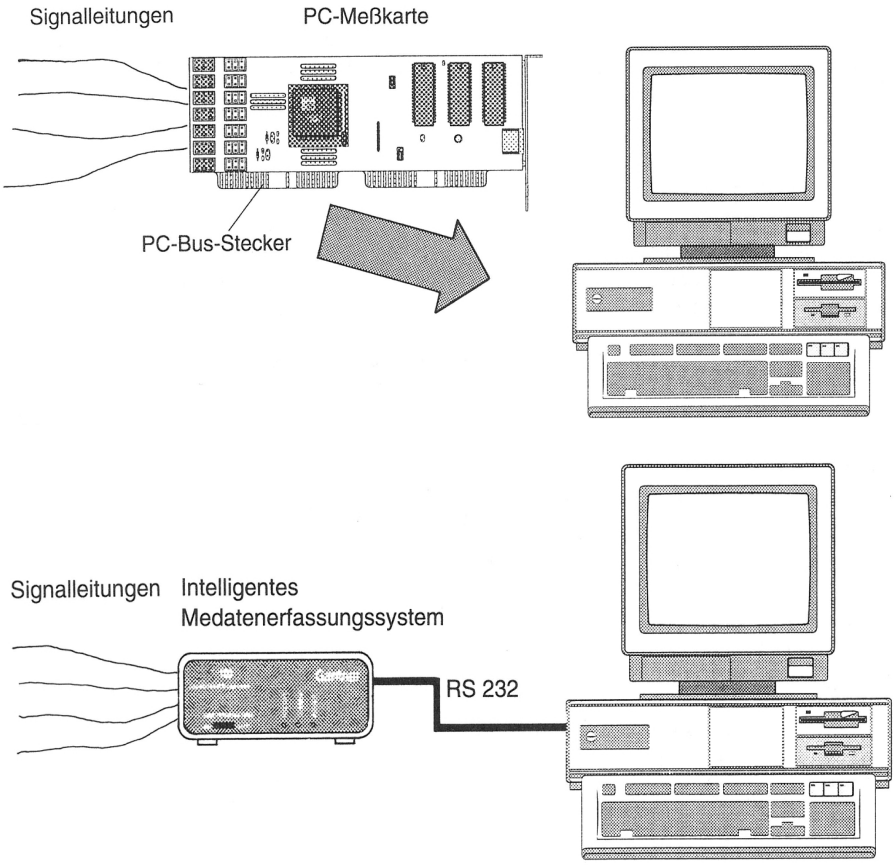

Abbildung 3.2: Messung und Steuerung über eine PC-Karte (oben) oder über eine serielle Schnittstelle eines Rechners (unten). Bilder entnommen aus [14]).

Abbildung 3.3: Betrieb der Messeinheit Typ 18 über einen Rechner mit eingebauter PC-Karte und Steuer- und Auswertesoftware (schematisiert).

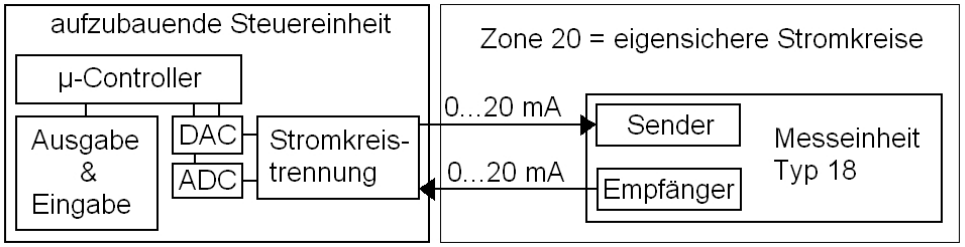

Abbildung 3.4: Ausgewählte Lösung für die aufzubauende Steuereinheit (schematisiert).

3.3.3 Embedded System

Eine dritte Möglichkeit war, die Vorteile der bisher genannten Lösungsansätze zu einer tragfähigen Gesamtlösung zu kombinieren.
Vom vorhandenen SKG 6 sollte demnach so viel übernommen werden wie möglich. Das galt insbesondere für die eigensichere Bauweise des Gerätes, d. h. die Forderung nach Einsatzmöglichkeit des Staubkonzentrationsmessgerätes in Zone 20 wäre damit erfüllt.
Der Einsatz eines Mikrocontroller-Boards im Gerät (Embedded System) würde die Vorteile einer Computersteuerung in den Lösungsansatz einbringen. Da die Mess- und Steuersoftware für einen Mikrocontroller selbst geschrieben werden müsste, entfielen damit die hohen Kosten für die Anschaffung einer professionellen kommerziellen Anwendersoftware. Wegen der relativen Einfachheit der Messaufgabe wurde der Programmieraufwand für den Mikrocontroller als beherrschbar eingeschätzt.
Die Erfassung und Auswertung der Messdaten würden mit einer solchen Lösung geräteintern ausgeführt. Die Auswertung der Daten würde die Berechnung der Konzentration gemäß Formel (2.2) aus dem gemessenen Empfängersignal bedeuten. Alle nötigen mathematischen Operationen und der automatische Nullabgleich benötigen keine aufwendige Software. Sie könnten mit einem 8 Bit-Mikrocontroller durchgeführt werden. Durch den Einsatz eines Mikrocontrollers können weitere digitale Bauelemente wie z. B. Speicher oder moderne Touchdisplays in das Gerät integriert werden. Damit ließen sich alle Anforderungen aus Abschnitt 3.2.1 an das Staubkonzentrationsmessgerät erfüllen. Ein solches Messsystem ist in Abbildung 3.4 schematisiert dargestellt.

Auch die Kosten würden mit einer solchen Lösung gering gehalten, da die Preise für in Frage kommende Mikrocontroller im Bereich um 100 Euro liegen. Durch den zusätzlichen Einsatz von Technik aus dem SKG 6 würden die Material- und Fertigungskosten für den Aufbau der neuen Steuereinheit weit unter den geforderten 5.000 Euro Höchstgrenze bleiben. Selbst wenn die rechnergestützte Lösung aus dem vorigen Abschnitt 3.3.2 mit einer selbst programmierten Steuer- und Auswertesoftware anstatt einer teuren kommerziellen Anwendersoftware betrieben würde, wäre das hier beschriebene Embedded System erheblich kostengünstiger. Hinzu kommt, dass durch den Aufbau eines solchen Systems die Tradition eines vollkommen eigenständigen Gerätes gewahrt bleibt, wodurch die FSA einen gewissen Werbeeffekt erzielen und so an Prestige gewinnen kann.

Aufgrund der dargestellten erheblichen Vorteile dieses Lösungsansatzes wurde er zur Realisierung ausgewählt.

Kapitel 4

Die realisierte Lösung

4.1 Aufbau der Steuereinheit

Die Auswahl der Lösung mit einem integrierten Mikrocontroller hatte zur Folge, dass ein erheblicher Teil der Entwicklungszeit für Programmierarbeiten aufgewendet werden musste (siehe Anhang C). Dazu war das Einarbeiten in die Programmiersprache C erforderlich, wozu [15] als Hilfsmittel benutzt wurde. Der positive Aspekt dieses Lösungsweges war, dass das Gerät weitgehend modular aufgebaut werden konnte, d. h. die Entwicklung umfangreicher elektronischer Schaltungen entfiel.
Die gebaute Steuereinheit ist in Abbildung 4.1 zu sehen. Die Abbildung 4.2 zeigt das Blockschaltbild der Steuereinheit. Das System besteht aus sieben Blöcken:

1. Spannungsversorgung

   Dieser Block gewährleistet die Spannungsversorgung aller elektronischen Komponenten der Steuereinheit. Er besteht aus vier Spannungsversorgungsbausteinen. Baustein eins ist das Stromversorgungsmodul STEP-PS/1AC/24DC/0.5 von Phoenix Contact. Es liefert eine Spannung von 24 V und einen maximalen Strom von 0,5 A an seinem Ausgang. Das Modul versorgt den Trennblock (siehe 2.). Baustein zwei des Blocks Spannungsversorgung ist ein Netzteil des Typs MINI-PS-100-240AC/2x15DC/1, ebenfalls von Phoenix Contact. Das Netzteil liefert ±15 V und einen maximalen Strom von 1,5 A. Es versorgt die beiden Treiber-Blöcke für den Sender und den Empfänger (siehe 5. und 6.). Der dritte Baustein ist eine 5 V-Spannungsversorgung. Sie besteht aus einem Low Dropout-Baustein des Typs L7805A und versorgt den Regeleinheit-Block (siehe 3.). Der vierte Spannungsregler LD1117V33 von STMicroelectronics liefert eine Spannung von 3,3 V. Sie wird für die Versorgung der Ein- und Ausgabeeinheit (siehe 4.) verwendet. Die beiden Module von Phoenix Contact (Abbildung 4.3) werden mit einer 220 V Wechselspannung betrieben und sind auf einer Hutschiene befestigt. Der 5 V-Versorgungsbaustein befindet sich auf einer Platine. Der Baustein LD1117V33 ist zusammen mit einem Kühlkörper auf einer Platine seitlich innen am Gehäuse installiert. Bausteine drei und vier werden von Baustein zwei gespeist.

Abbildung 4.1: Fotografie der aufgebauten Steuereinheit SKG 7 mitsamt angeschlossener Messeinheit Typ 18.

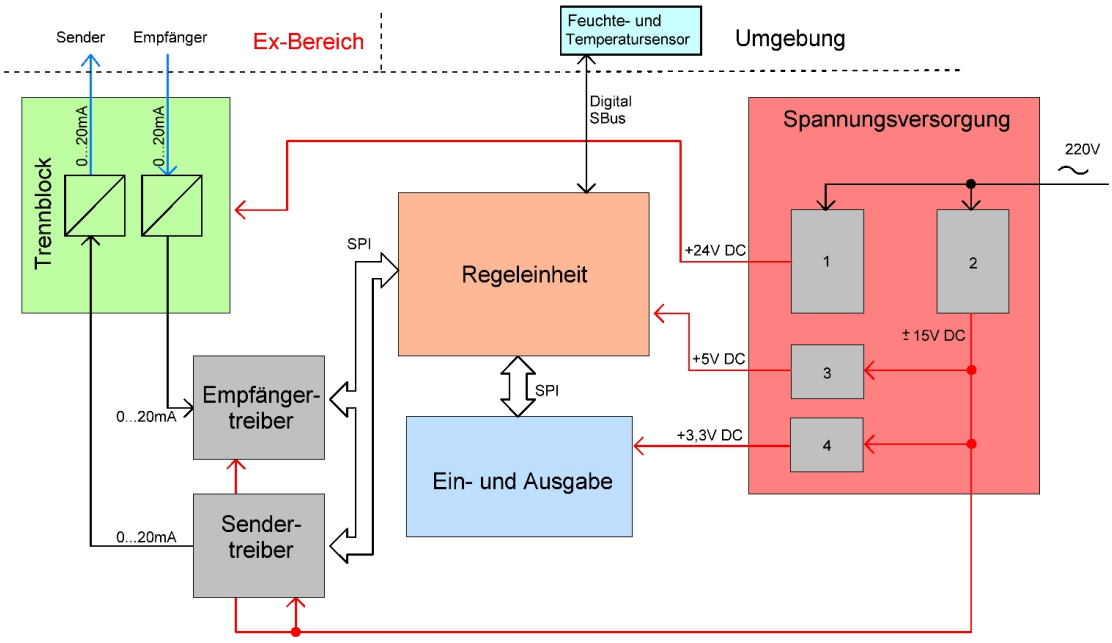

Abbildung 4.2: Blockschaltbild der Steuereinheit SKG 7. Das Gerät besteht aus sieben Blöcken: Spannungsversorgung, Trennblock, Regeleinheit, Ein- und Ausgabeeinheit, Sendertreiber, Empfängertreiber und Temperatur- und Feuchtemessung.

Abbildung 4.3: Versorgungsbausteine von Phoenix Contact STEP-PS/1AC/24DC /0.5 (links) und MINI-PS-100-240AC/2x15DC/1 (rechts).

2. Trennblock

   Dieser Block (Abbildung 4.4) sorgt dafür, dass die Messeinheit Typ 18 in explosionsgefährdeten Zonen eingesetzt werden darf. Mit dem Trennblock wird die Zündschutzart *Eigensicherheit* umgesetzt, d. h. in der Messeinheit treten keine Ströme und Spannungen auf, die eine Staubexplosion zünden können [10]. Der Block besteht aus zwei Trennverstärkern von Phoenix Contact. Der Trennverstärker eins des Typs MACX-MCR-EX-SL-IDSI-I ist ein Ausgangstrennverstärker. Er überträgt das analoge Stromsignal (0 mA - 20 mA) eigensicher vom Sendertreiber zum Sender in der Messeinheit. Der Trennverstärker zwei des Typs MACX-MCR-EX-SL-RPSSI-I ist ein Eingangstrennverstärker. Er überträgt das analoge Signal (0 mA - 20 mA) des Empfängers eigensicher von der Messeinheit zum Empfängertreiber.

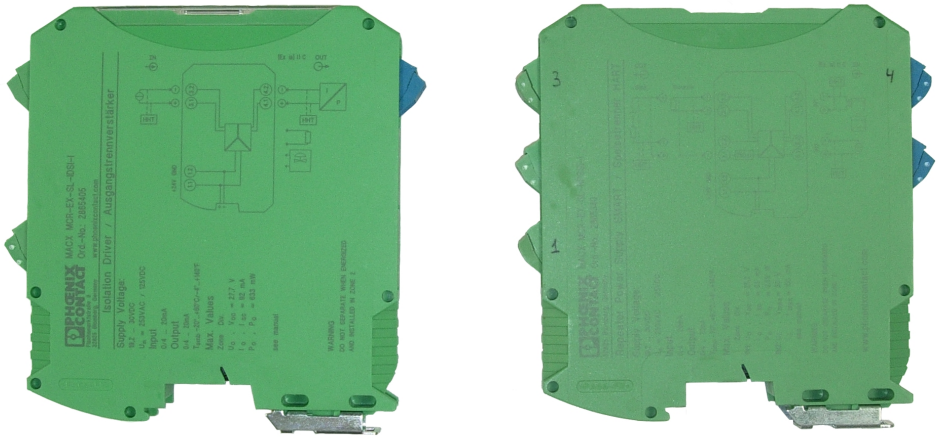

Abbildung 4.4: Ausgangstrennverstärker MACX-MCR-EX-SL-IDSI-I (links) und Eingangstrennverstärker MACX-MCR-EX-SL-RPSSI-I (rechts).

3. Regeleinheit

   Die Regeleinheit ist ein AVR XMEGA-A1 XPLAINED-Board (Abbildung 4.5). Sie steuert das gesamte Staubkonzentrationsmessgerät SKG 7 Typ 18. Das Board verfügt über alle zur Regelung anderer Geräteblöcke nötigen Schnittstellen. Über eine SPI-Schnittstelle werden die Sender- und Empfängertreiber und die Ein- und Ausgabeeinheit gesteuert. Das Herzstück des Mikrocontroller-Boards ist ein XMega-Mikrocontroller des Typs ATxmega128A1. In den Mikrocontroller wird über eine JTAG-Schnittstelle ein Programm geladen, das die Ablaufsteuerung aller Ereignisse im System übernimmt. Der Programmcode (geschrieben in C) und die Flussdiagramme sind in Anhang C abgedruckt.

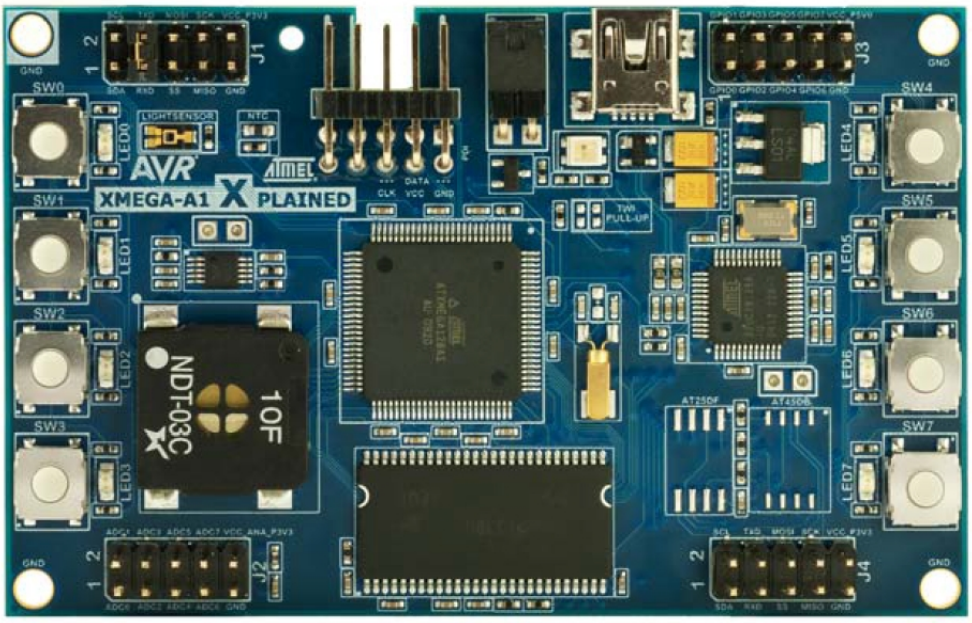

Abbildung 4.5: Die Regeleinheit AVR XMEGA-A1 XPLAINED.

4. Ein- und Ausgabeeinheit

   Die Ein- und Ausgabeeinheit ist ein Touchdisplay des Typs 4LCD-FT843 von 4D-Systems (Abbildung 4.1). Es ist ein HMI[1]-Modul. Über das Touchdisplay wird der Informationsaustauch zwischen Steuereinheit und Bediener des Gerätes abgewickelt. Alle nötigen Informationen werden auf dem Display ausgegeben. Gleichzeitig können die für die Messung notwendigen Parameter über das Touchdisplay eingegeben werden.

5. Sendertreiber

   Dieser Block steuert den Strom durch den Sender in der Messeinheit. Der Treiber erhält den Wert für den einzustellenden Strom vom Block Regeleinheit über eine SPI-Schnittstelle und stellt den benötigten Senderstrom ein.

[1] human-machine-interface

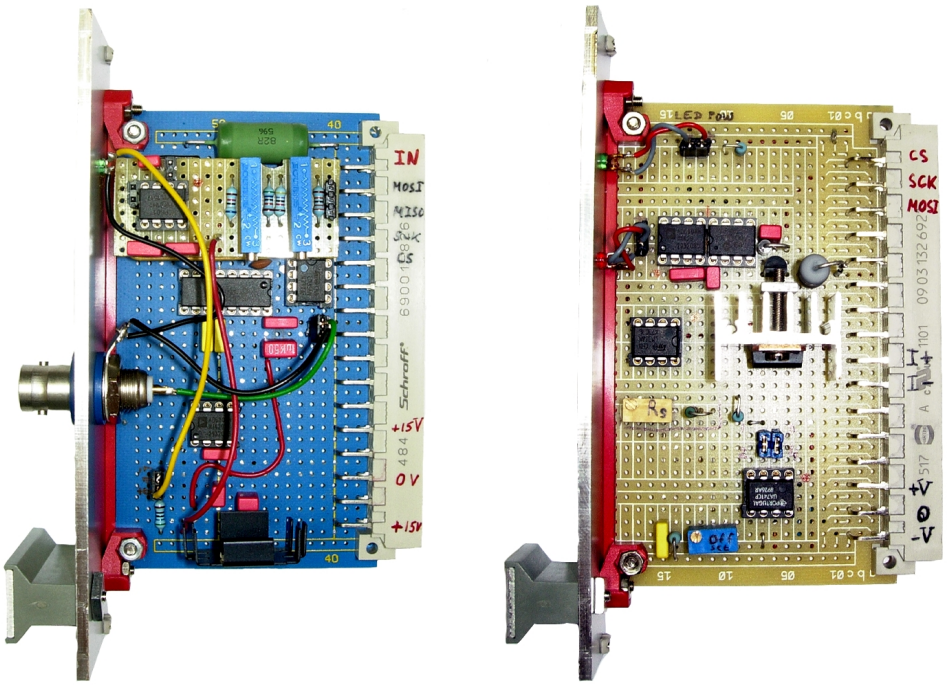

Abbildung 4.6: Sendertreibereinschubkarte (rechts) und Empfängertreibereinschubkarte (links).

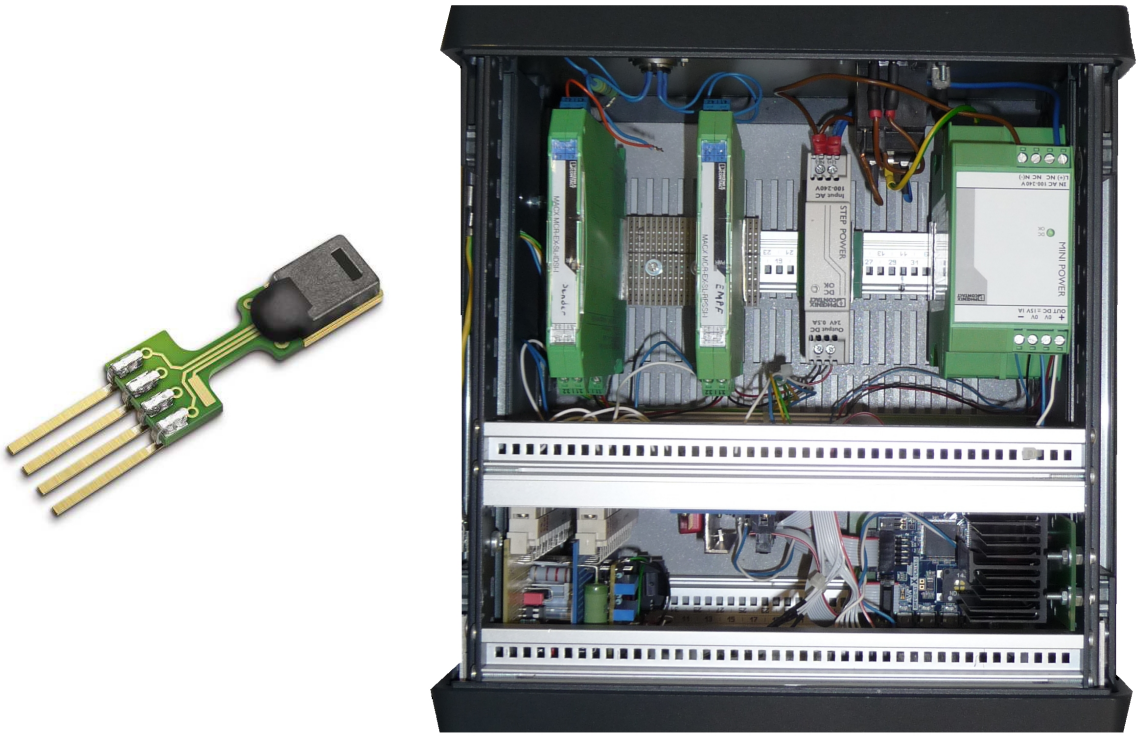

Abbildung 4.7: Sensor SHT75 von Sensirion für die Temperatur- und Luftfeuchtemessung (links) und Blick von oben in die geöffnete Steuereinheit (rechts).

6. Empfängertreiber

   Der Treiber für den Empfänger misst den Strom, der vom Verstärker der Fotodiode (Empfänger) aus der Messeinheit kommt. Dieser Stromwert kann jederzeit vom Regeleinheit-Block über eine SPI-Schnittstelle abgefragt werden.

7. Feuchte- und Temperatursensor

   Dieser Block (Abbildung 4.7, links) misst Temperatur und relative Luftfeuchte in der Umgebung und sendet die gemessenen Daten an die Regeleinheit.

In Abbildung 4.7, rechts sind die einzelne Blöcke, eingebaut in das Gerät, zu sehen.

4.2 Einstellung des Sendersignals

Der Sender in der Messeinheit Typ 18 ist eine Infrarotlicht-Diode des Typs TSAL6200. Fließt ein elektrischer Strom durch die Diode, emittiert die Diode infrarotes Licht einer bestimmten Intensität Φ. Weil vor jeder Messung die abgestrahlte Lichtintensität Φ_0 bekannt und normiert sein muss (siehe Abschnitte 2.2 und 2.4), ist der Senderstrom auf den Wert I_0 zu regeln. Der eingestellte Senderstrom bleibt im Lauf der Messung konstant, d. h. die Infrarot-Diode emittiert während dieser Zeit konstant die Lichtintensität Φ_0. Die schematische Darstellung der Sendersignal-Steuerung ist in Abbildung 4.8 gezeigt.

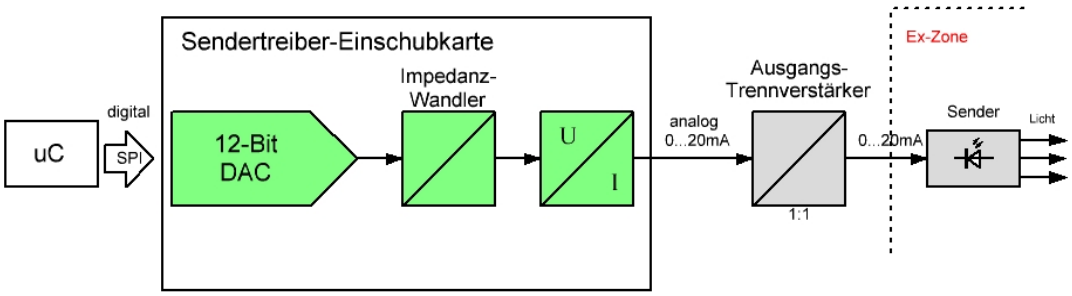

Abbildung 4.8: Schematische Darstellung der Sendersignal-Regelung.

Der einzustellende Strom I_0 wird vom Mikrocontroller (Regeleinheit) vorgegeben. Der Mikrocontroller kommuniziert über eine digitale Schnittstelle (SPI) mit dem Sendertreiber. Der Sendertreiber empfängt die Daten vom Mikrocontroller und stellt den gefordeten Strom durch den Sender ein. Der Treiber besteht aus einem DAC-Baustein, einem Impedanzwandler und einem Spannung/Strom-Wandler (U/I-Umsetzer). Der DAC-Baustein empfängt den digitalen Wert für den einzustellenden Strom und gibt ihn als Spannungssignal auf seinen Ausgang. Das Spannungssignal regelt mithilfe eines U/I-Umsetzers den Strom durch den Sender. Um den Ausgang des DAC-Bausteines nicht zu stark zu belasten, ist ein Impedanzwandler zwischen DAC und U/I-Umsetzer eingebaut. Zwischen Sendertreiber und Sender ist ein Trennverstärker geschaltet, der das Stromsignal in den explosionsgefährdeten Bereich überträgt. Der Ausgang des Trennverstärkers speist einen eigensicheren Stromkreis, den der Sender in der Messeinheit

und das Verbindungskabel bilden. Dabei wird die in den explosionsgefährdeten Bereich gelieferte elektrische Energie derart begrenzt, dass sie als potentielle Zündquelle ausgeschlossen wird.

Das Schaltbild des Sendertreibers ist in Abbildung 4.9 dargestellt. Das digitale Signal $\mathsf{D_n} = 0 \ldots 4095$, das vom Mikrocontroller-Board gesendet wird, gelangt auf einen 12 Bit-D/A-Umsetzer (MCP4921 von Microchip). Durch die Pull-Down-Widerstände R1 und R2 werden undefinierte Pegelniveaus am Eingang des DACs vermieden. Der DAC-Baustein setzt das digitale 12 Bit-Eingangssignal in ein analoges Spannungssignal $\mathsf{U_{out}} = 0 \ldots \mathsf{U_{ref}}$ um. Die Referenzspannung $\mathsf{U_{ref}}$ wird vom Baustein AD680 geliefert. Er erzeugt eine stabile Referenzspannung $\mathsf{U_{ref}} = 2{,}50\,\mathrm{V} \pm 5\,\mathrm{mV}$. Die Ausgangsspannung $\mathsf{U_{out}}$ des Umsetzers wird nach der Formel

$$\mathsf{U_{out}} = \frac{\mathsf{U_{ref}} \cdot \mathsf{D_n}}{4096} \tag{4.1}$$

berechnet und kann den Wert 0 V bis 2,50 V annehmen.
Das Ausgangssignal gelangt über den Impedanzwandler auf die U/I-Umsetzer-Stufe. Diese Stufe stellt einen Regelkreis dar, welcher aus einem Operationsverstärker, einem Transistor und dem Messwiderstand $\mathsf{R_{sense}}$ besteht. Der Operationsverstärker OP2 übernimmt die Rolle eines Reglers. Die Spannung am Ausgang des Operationsverstärkers (und somit der Strom durch den MOSFET-Transistor T1) wird so geregelt, dass die Summe seiner Eingangsspannungen $\mathsf{U_{in}^+} - \mathsf{U_{in}^-} = 0$ bleibt. Mit $\mathsf{U_{in}^+} = \mathsf{U_{in}^-}$ folgt die $\mathsf{U_{in}^+}$-Eingangsspannung immer der $\mathsf{U_{in}^-}$-Spannung. Die Spannung $\mathsf{U_{in}^-}$ wird vom DAC definiert. Die Eingangsspannung $\mathsf{U_{in}^+}$ ist gleich dem Spannungsabfall am Messwiderstand $\mathsf{R_{sense}}$. Sie definiert den Strom I durch den $\mathsf{R_{sense}}$-Widerstand[2]. Dieser Strom wird somit gemäß der Formel

$$\mathsf{I} = \frac{\mathsf{U_{in}^+}}{\mathsf{R_{sense}}} = \frac{\mathsf{U_{in}^-}}{\mathsf{R_{sense}}} = \frac{\mathsf{U_{ref}} \cdot \mathsf{D_n}}{4096 \cdot \mathsf{R_{sense}}} \tag{4.2}$$

bestimmt. Im Maximum erreicht er den Wert von

$$\mathsf{I} = \frac{2{,}50\,\mathrm{V} \cdot 4095}{4096 \cdot 125\,\Omega} = 19{,}995\,\mathrm{mA} \tag{4.3}$$

In den Strompfad ist der Sender eingeschlossen. Somit fließt der eingestellte Strom I durch den Sender. Der Strom I fließt auch durch eine rote LED, die in der Frontplatte eingebaut ist. Fließt Strom durch den Sender, leuchtet die Diode auf.
Der Widerstand R4 dient der Strombegrenzung durch den Sender. Wird der Transistor T1 im Fall eines Defektes kurzgeschlossen, würde im Senderstromkreis ein maximaler Strom von

$$\mathsf{I_{max}} = \frac{\mathsf{U^+}}{\mathsf{R4} + \mathsf{R_{sense}}} = \frac{+15\,\mathrm{V}}{(120 + 125)\,\Omega} = 61{,}22\,\mathrm{mA} \tag{4.4}$$

fließen, wenn er nicht vom Ausgangstrennverstärker begrenzt würde.

[2]Der Eingangswiderstand des Operationsverstärkers beträgt einige MΩ. Der Strom fließt somit nahezu komplett über den $\mathsf{R_{sense}}$-Widerstand ab.

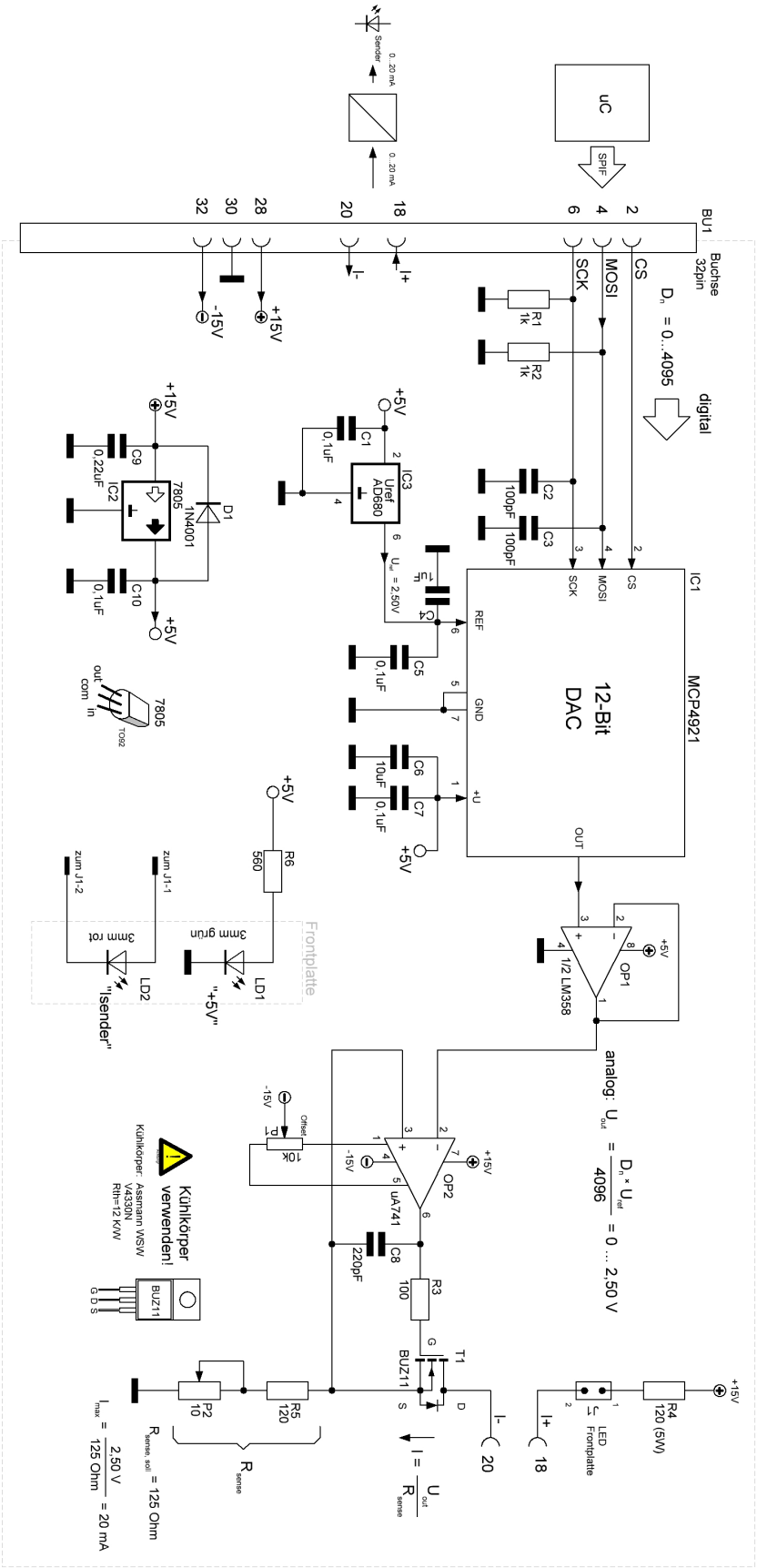

Abbildung 4.9: Schaltbild des Sendertreibers.

Das RC-Glied R3 - C8 dient dem besseren Stabilitätsverhalten des Regelkreises. Der Operationsverstärker OP2 wird mit der Spannung $\pm 15\,\mathrm{V}$ versorgt. Diese Spannung liefert der Block Spanungsversorgung, genauer das MINI-PS-100-240AC/2x15DC/1. Das Netzteil versorgt auch den 7805-Spannungswandler, welcher für die Bereitstellung der benötigten $+5\,\mathrm{V}$-Spannung zuständig ist. Mit dieser Spannung werden die aktiven Bauelemente des Treibers gespeist: der DAC MCP4921, der OP1 LM358 und der IC3 AD680. Das Vorliegen der $+5\,\mathrm{V}$-Spannung wird an der Frontplatte durch eine grüne LED signalisiert.
Die Kondensatoren C1 bis C7 sind Bypass-Kondensatoren. Sie dienen der Spannungssiebung an den Eingängen der aktiven Bauteile.

4.3 Messung des Empfängersignals

Wie das Empfängersignal gemessen wird, ist in Abbildung 4.10 schematisch dargestellt. Die in die Messeinheit eingebaute Empfängerdiode (Fotodiode) des Typs BPW34F absorbiert das auf die Sensorfläche einfallende Licht. Der Empfänger wandelt die Intensität des empfangenen Lichtes Φ proportional in eine Spannung (mV-Bereich) um. Diese Spannung wird in der Messeinheit durch eine Schaltung verstärkt und in ein Stromsignal von $0\,\mathrm{mA}$ bis $20\,\mathrm{mA}$ umgewandelt. Das Stromsignal wird vom Eingangstrennverstärker aufgenommen und weiter zum Empfängertreiber-Block geleitet. Der Eingangstrennverstärker liefert die Versorgungsspannung für die Empfänger-Elektronik in der Messeinheit und trennt das Stromsignal zwischen dem explosionsgefährdeten und sicheren Bereich. Im Empfängertreiber wird das ankommende Stromsignal in ein Spannungssignal umgewandelt, gefiltert und digitalisiert. Das nun digitale Signal wird über eine digitale SPI-Schnittstelle zum Mikrocontroller-Board zur weiteren Verarbeitung übergeben. Der Mikrocontroller nimmt die Daten auf, berechnet die Staubkonzentration c in $\mathrm{g/m^3}$ und gibt ihren Wert auf dem Display aus.

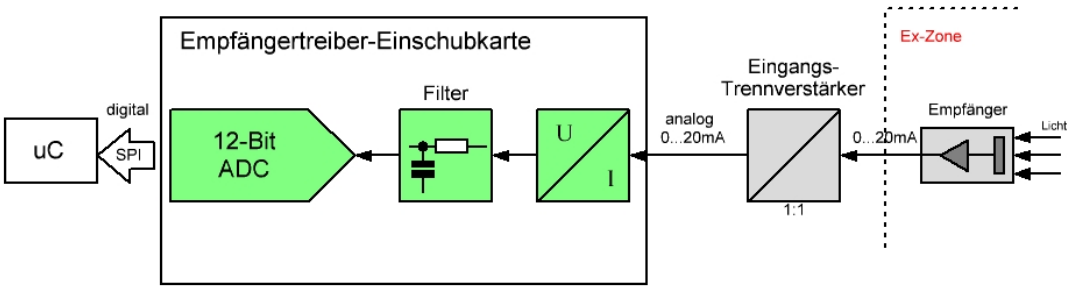

Abbildung 4.10: Schematische Darstellung der Messung des Empfängersignals.

Das Schaltbild des Empfängertreibers ist in Abbildung 4.11 dargestellt. Das von der Messeinheit kommende analoge Stromsignal $\mathsf{I_{in}}$ wird an Pin 2 der BU1-Buchse eingespeist. Der Strom fließt anschließend durch den Widerstand R1. Dadurch wird eine analoge Spannung $\mathsf{U_{in}} = \mathsf{R1} \cdot \mathsf{I_{in}}$ erzeugt.
Die Spannung $\mathsf{U_{in}}$ wird durch einen aktiven Butterworth-Tiefpassfilter zweiter Ordnung aufgenommen. Der Filter wird durch den Operationsverstärker OP1 gebildet.

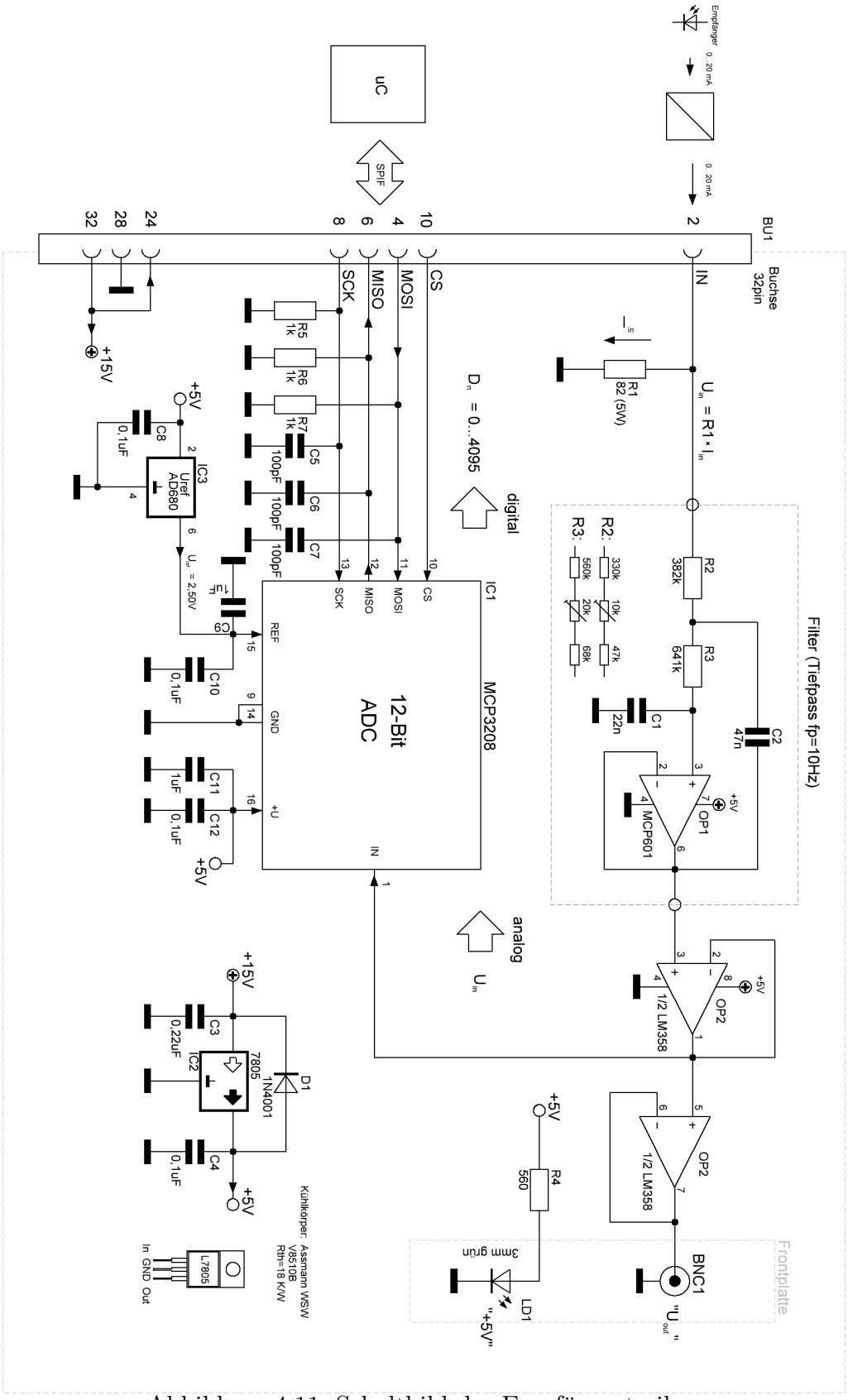

Abbildung 4.11: Schaltbild des Empfängertreibers.

Seine Grenzfrequenz liegt bei 10 Hz. Die Frequenzen oberhalb von 10 Hz werden stark unterdrückt. Der Verstärkungsfaktor des Filters beträgt 1.
Das so aufbereitete NF-Signal U_{in} gelangt über einen Impedanzwandler auf den Eingang eines A/D-Wandlers. Über einen BNC-Stecker in der Frontplatte kann das Signal auch direkt abgegriffen werden.
Als externe Referenzspannungsquelle für den A/D-Wandler dient der Referenzbaustein IC3. Er liefert eine stabile und präzise Referenzspannung von $U_{ref} = 2{,}50\,\text{mV}$.
Der 12 Bit-A/D-Wandler (MCP3208 von Microchip) kommuniziert über den SPI-Bus mit dem Mikrocontroller-Board. Nach einer Anfrage vom Mikrocontroller wandelt der Umsetzer das analoge Eingangssignal U_{in} in einen digitalen Wert D_n gemäß der Formel

$$D_n = U_{in} \cdot \frac{4096}{R_{ref}} = I_{in} \cdot \frac{R_1 \cdot 4096}{R_{ref}} \tag{4.5}$$

um. Nach der Umwandlung gelangt der Wert D_n zum Mikrocontroller.

Mit $R1 = 82\,\Omega$ und $U_{ref} = 2{,}50\,\text{V}$ ergibt sich als maximaler Strom $I_{max} = 2{,}5\,\text{V}/82\,\Omega = 30{,}48\,\text{mA}$, der mit einer Auflösung von $I_{max}/4096 = 7{,}44{\cdot}10^{-3}\,\text{mA}$ gemessen werden kann.
Die Widerstände R5 bis R7 sind Pull-Down-Widerstände. Durch sie werden undefinierte Spannungspegel an den Datenleitungen vermieden. Die Widerstände begrenzen gleichzeitig den maximalen Stromfluss durch die Datenleitungen.
Die Versorgung der Schaltung mit +15 V erfolgt über die Pins 28 und 32 der BU1-Buchse. Aus dieser Spannung erzeugt der Linearregler IC2 eine Spannung von +5 V, mit der alle aktiven Bauteile der Schaltung gespeist werden.

4.4 Messung von Temperatur und Luftfeuchte

Die Messung der Umgebungstemperatur und der relativen Luftfeuchte erfolgt mit einem Sensor des Typs SHT75 von Sensirion. Der Sensor ist auf einem einzelnen Siliziumchip integriert und liefert digital ein vollständig kalibriertes Ausgangssignal. Der Sensor ist in einem kleinen Gehäuse untergebracht, welches seitlich an der Messeinheit von außen befestigt ist (siehe Abbildung 4.12).
Der Sensor ist mit seinen vier Anschlüssen über einen vom Autor aufgebauten Adapter an die Regeleinheit angeschlossen. Sie versorgt den Sensor mit einer 3,3 V-Spannung und fragt gleichzeitig die Messdaten vom Sensor über eine serielle Schnittstelle ab. Von der Regeleinheit werden die empfangenen Daten ausgewertet und zur Anzeige auf dem Display an die Ein- und Ausgabeeinheit weitergeleitet.
Die Luftfeuchtigkeit wird in % mit einer Genauigkeit von $\pm 1{,}8\,\%$ gemessen. Der Messbereich beträgt 0 % bis 100 %. Die Temperatur wird in °C im Bereich von $-40\,°\text{C}$ bis $+125°\text{C}$ gemessen. Die Genauigkeit beträgt dabei $\pm 0{,}3\,°\text{C}$.

Abbildung 4.12: Gehäuse mit dem Sensor für Temperatur- und Feuchtemessung.

4.5 Bedienung des Gerätes

4.5.1 Übersicht

Die Steuereinheit wird über ein Touchdisplay (Ein- und Ausgabeeinheit) bedient. Abbildung 4.13 zeigt eine Übersicht über die Menüs, über die das Gerät gesteuert wird.

4.5.2 Displaykalibrierung

Nach dem Einschalten des Gerätes wird zunächst der Touchdisplay kalibriert. Dabei werden auf dem Display drei Punkte angezeigt, die vom Bediener angetastet werden müssen (Abbildung 4.14). Nach der erfolgreichen Kalibrierung der Touchfunktion erscheint automatisch das Start-Menü des Gerätes.

4.5.3 Start-Menü

Das Start-Menü (Abbildung 4.15) erscheint nach dem Einschalten des Gerätes und nach der Kalibrierung der Touchfunktion. Neben der Bezeichnung des zu messenden Staubes und seiner Konzentration werden auch die Messwerte für die relative Feuchtigkeit und die Temperatur der Umgebungsluft angezeigt. Weiterhin erscheinen der Strom durch den Sender I_{Sender}, das Stromsignal des Empfängers $I_{Empfaenger}$ und der Offset des Empfängerstroms I_{offset}. Die Einheiten der gemessenen Größen sind in der Tabelle 4.1 aufgelistet.

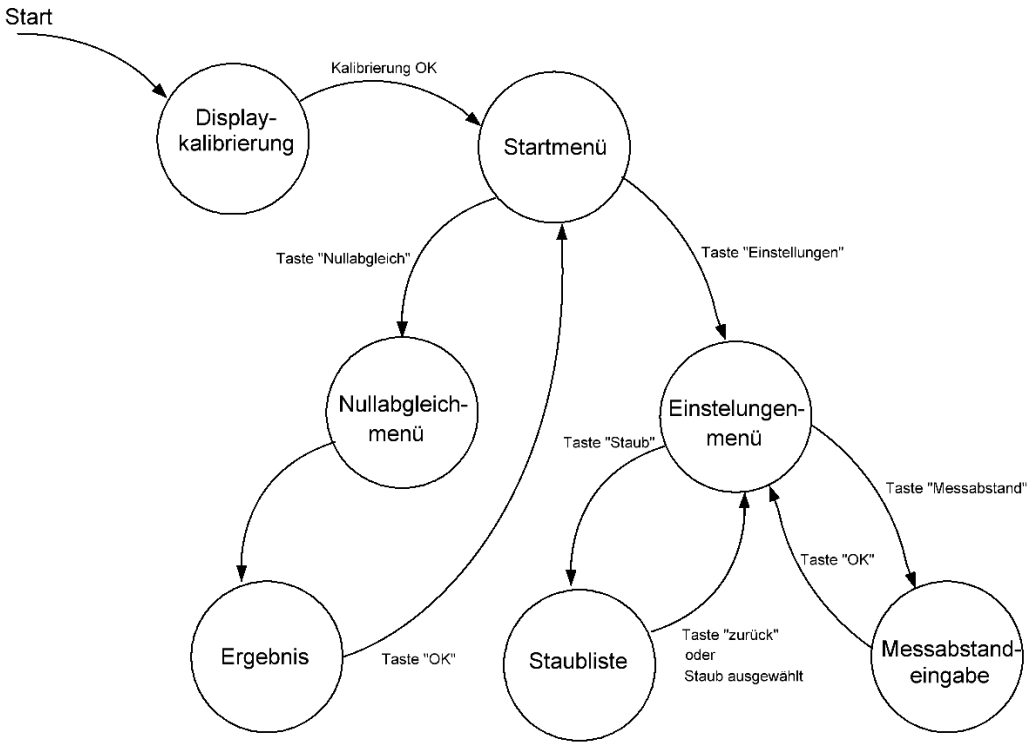

Abbildung 4.13: Übersicht über die Menüs der Steuereinheit.

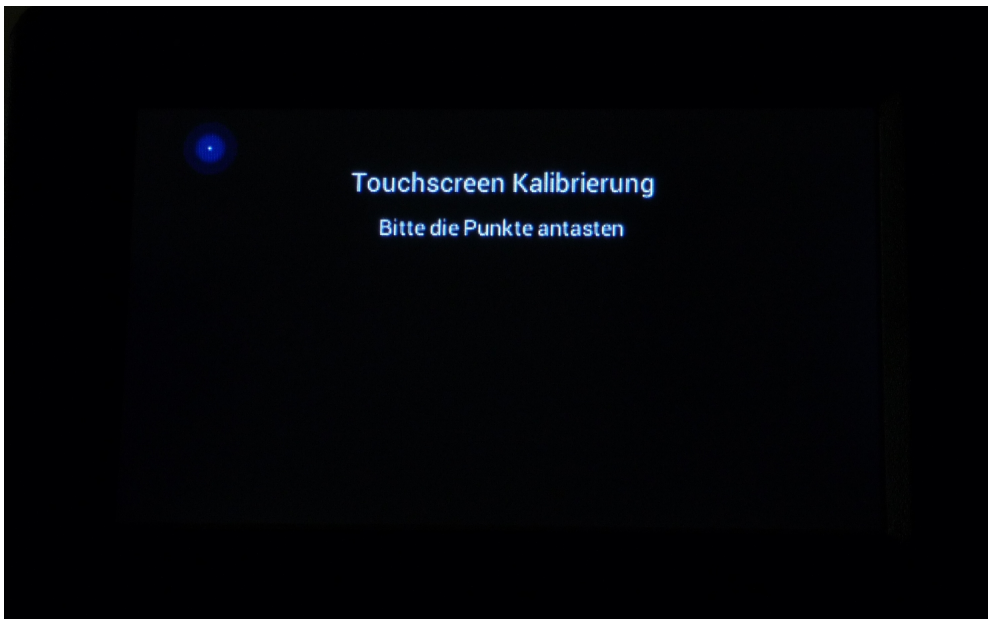

Abbildung 4.14: Kalibrierung der Touchfunktion des Displays.

Abbildung 4.15: Start-Menü der Steuereinheit.

Messgröße	Einheit
Konzentration	$\frac{\mathrm{g}}{\mathrm{m}^3}$
Relative Luftfeuchtigkeit	%
Umgebungstemperatur	°C
Senderstrom	mA
Empfängerstrom	mA
Empfängeroffsetstrom	mA

Tabelle 4.1: Einheiten der im Start-Menü ausgegebenen Größen.

Für einen zuverlässigen Betrieb benötigt der Empfängerverstärker in der Messeinheit Typ 18 einen Mindeststrom, der kleiner als 1 mA ist. Dieser Strom wird als Empfängeroffsetstrom I_{offset} bezeichnet. Er wird beim Einschalten des Gerätes bei angeschlossener Messeinheit ermittelt und im Start-Menü angezeigt. Der angezeigte Strom $I_{Empfänger}$ ist der gesamte Empfängerstrom, also inklusive des Offsetstroms. Der Empfängeroffsetstrom wird bei der Berechnung der angezeigten Staubkonzentration berücksichtigt. Er ist bei der Kalibrierung der Messeinheit zu berücksichtigen.

Über die Tastfelder „Nullabgleich“ und „Einstellungen“ gelangt man zu den entsprechenden Unter-Menüs.

4.5.4 Menü Nullabgleich

Nachdem im Start-Menü die Taste „Nullabgleich“ gedrückt wurde, erscheint das Nullabgleich-Menü. Der Nullabgleich wird automatisch gestartet, d. h. das

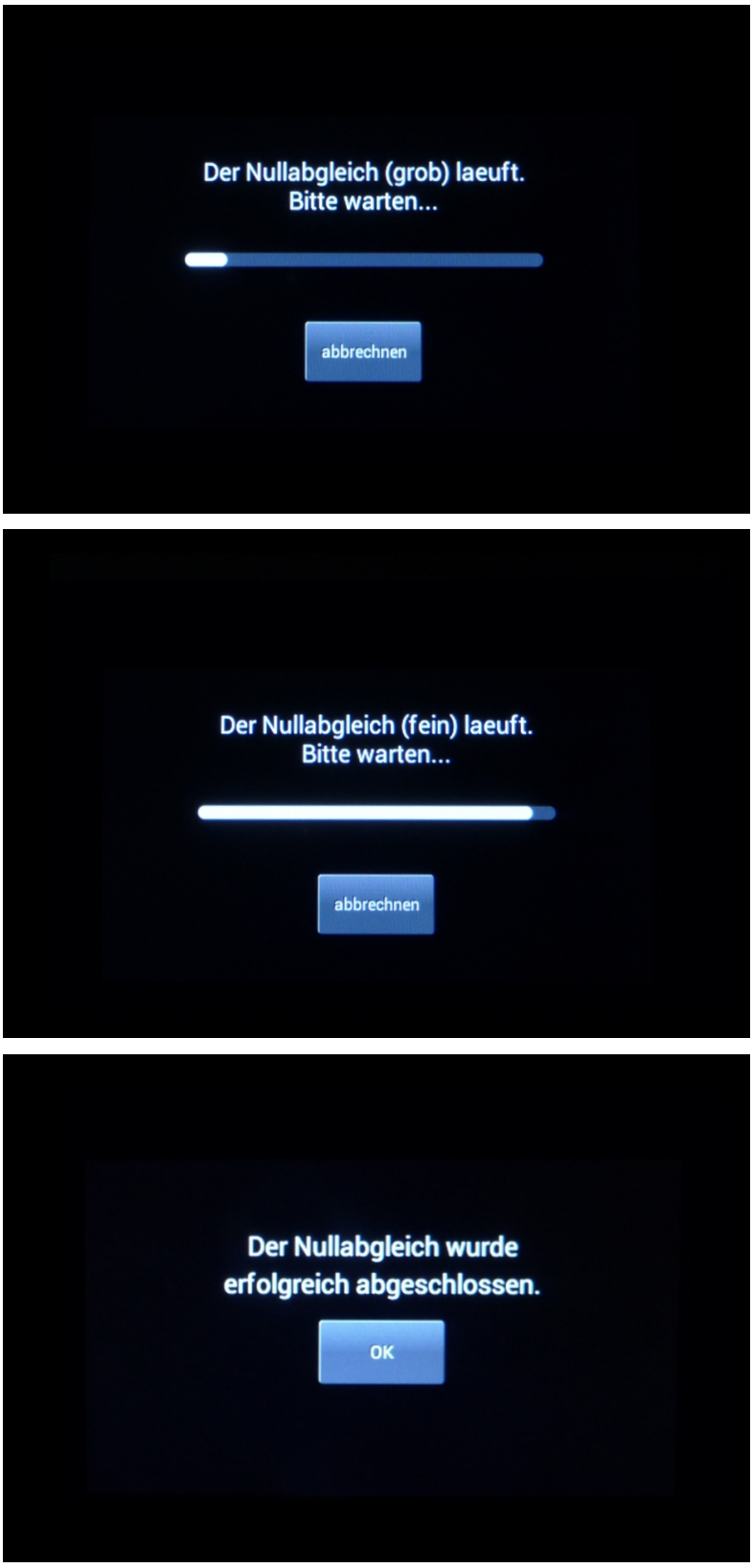

Abbildung 4.16: Menü Nullabgleich: Automatischer Grobabgleich (oben), automatischer Feinabgleich (Mitte) und Meldung bei erfolgreichem Nullabgleich.

Abbildung 4.17: Menü Nullabgleich: Meldung bei fehlgeschlagenem Nullabgleich.

Messsystem bereitet sich auf die bevorstehende Messung vor. Dabei wird der Strom durch den Sender von der Steuereinheit nacheinander in zwei Stufen (grob und fein) automatisch so geregelt, dass der Empfänger ein Stromsignal vom 20 mA + I_{offset} liefert (Abbildung 4.16, oben und Mitte). Der Einstellungsprozess kann über eine Minute Zeit in Anspruch nehmen (siehe auch Abschnitt 6.2.3). Der Status des Nullabgleichvorgangs wird durch das Progressbar-Element angezeigt. Über die Taste „abbrechen“ kann der Nullabgleich jeder Zeit abgebrochen werden.

Wurde die Einstellung der Messeinheit erfolgreich durchgeführt, wird der Benutzer darüber informiert und nach Drücken der Taste „OK“ in das Start-Menü zurückgeführt (Abbildung 4.16, unten).

Der Nullabgleich kann auch scheitern, was zur Anzeige des Bildschirms in Abbildung 4.17 führt. Nach dem Drücken der Taste „OK“ erscheint das Start-Menü.

4.5.5 Menü Einstellungen

Das Menü Einstellungen (Abbildung 4.18, oben) wird durch das Antippen der Taste „Einstellungen“ im Start-Menü aufgerufen. Über die Taste „Staub“ kann aus einer Liste ein Staub ausgewählt werden, dessen Konzentration zu messen ist (Abbildung 4.18, Mitte). Über die „Messabstand“-Taste kann der Wert für l in der Einheit mm eingegeben werden (Abbildung 4.18, unten). Durch Betätigen der Taste „Zurück“ im Einstellungen-Menü erscheint erneut das Start-Menü.

Abbildung 4.18: Menü Einstellungen (oben). Menü zur Auswahl eines Staubes für eine Messung (Mitte). Menü zur Eingabe des Messabstandes l (unten). Die Standardeinstellung für den Messabstand beträgt $l = 50\,\mathrm{mm}$.

Kapitel 5

Test des SKG 7 Typ 18

5.1 Aufnahme einer Kalibrierkurve

Nach Fertigstellung der Steuereinheit musste die korrekte Funktion des Gerätes experimentell geprüft werden. Dazu wurde zunächst untersucht, ob das Gerät seine Hauptfunktion, nämlich die Staubkonzentrationsmessung, zufriedenstellend erfüllt. Das neue Gerät SKG 7 Typ 18 wurde zu diesem Zweck mit der älteren und sehr bewährten Ausführung SKG 5 Typ 7 verglichen. Eine einwandfreie Funktionsweise konnte beim verwendeten SKG 5 Typ 7 als sicher vorausgesetzt werden. Mit beiden Geräten wurde je eine Kalibrierkurve für Maisstärke gemäß dem in Abschnitt 2.4 beschriebenen Verfahren aufgenommen. In Tabelle 5.1 und Abbildung 5.1 sind die Ergebnisse dieser Messungen aufgelistet bzw. grafisch dargestellt.
Die Kalibrierung des SKG 5 Typ 7 für Maisstärke ergab einen Extinktionskoeffizienten von $\epsilon = 0,13529\,\mathrm{m^2/g}$ bei einem Korrelationskoeffizienten von $r^2 = 0,99693$. Die Messung mit dem SKG 7 Typ 18 lieferte $\epsilon = 0,13587\,\mathrm{m^2/g}$ bei $r^2 = 0,99948$.
Die beiden r^2 nahe 1 und der visuelle Eindruck der Kurven in Abbildung 5.1 zeigen die hohe Güte der Regressionsrechnung. Die ermittelten ϵ stimmen bis auf die dritte Stelle hinter dem Komma überein. Die Abweichung von $\Delta\epsilon = 0,00058\,\mathrm{m^2/g} = 0{,}4\,\%$

SKG 5 Typ 7		SKG 7 Typ 18	
c [g/m^3]	U [mV]	c [g/m^3]	I [mA]
0,0	1.035	0,0	19,94
32,5	844	29,2	14,96
60,8	741	59,2	11,02
107,5	597	107,5	6,79
225,0	345	216,7	2,15
330,0	220	330,8	0,31
643,3	39	635,8	0,09

Tabelle 5.1: Ergebnisse der Kalibrierpunkterstellung für Maisstärke mit dem SKG 5 Typ 7 und dem SKG 7 Typ 18.

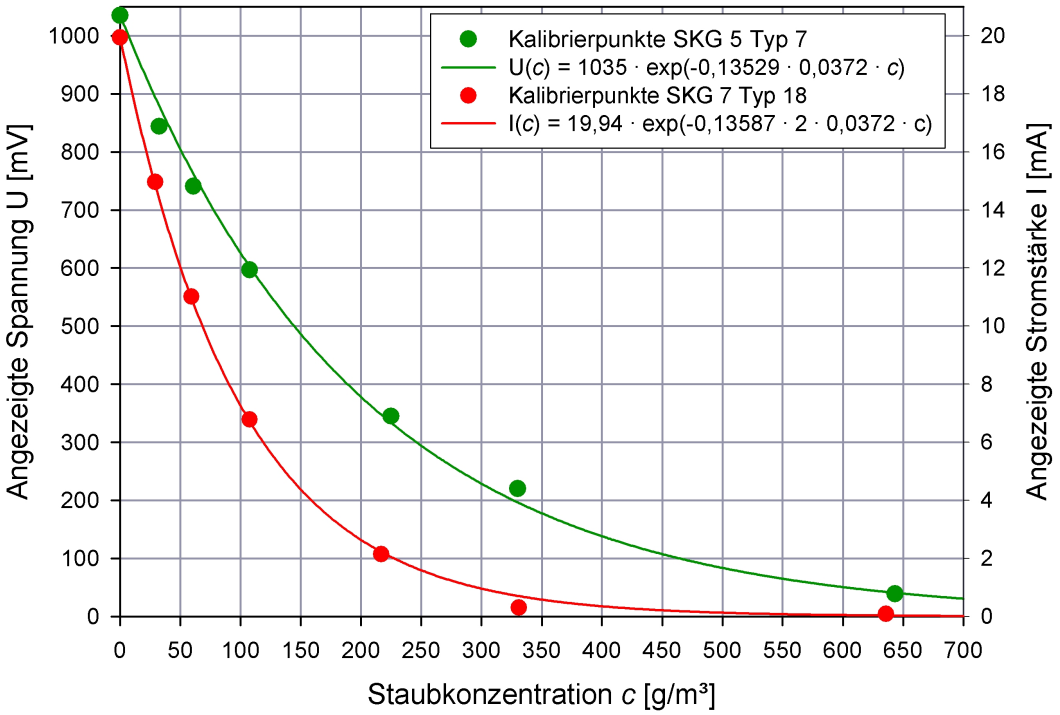

Abbildung 5.1: Kalibrierkurven für Maisstärke aufgenommen mit dem SKG 5 Typ 7 und SKG 7 Typ 18.

ist minimal und liegt innerhalb der Streuungsbreite, die bei der Bestimmung von ϵ überlicherweise auftritt. Diesen Funktionstest hat das SKG 7 Typ 18 bestanden.

5.2 Vergleichende Staubkonzentrationsmessung

Nach der erfolgreichen Aufnahme einer Kalibrierkurve sollte in einem weiteren Experiment untersucht werden, ob das SKG 7 Typ 18 eine unbekannte Staubkonzentration erkennen kann. Als Referenz wurde erneut das SKG 5 Typ 7 herangezogen. Als Material fand die Maisstärke aus Abschnitt 5.1 Verwendung.
Um den Aufwand einer Staubkonzentrationsmessung in Luft (Staubwolkenerzeugung und anschließende Reinigung) zu vermeiden, wurde erneut auf eine Ethanol/Maisstärke-Suspension für die Messungen zurückgegriffen. Die Testreihe, bestehend aus insgesamt drei Versuchen, wurde an der bekannten Kalibrierstation durchgeführt. Bei jedem Versuch wurde zuerst das Kalibriergefäß mit 120 ml Ethanol gefüllt. Danach wurden sukzessiv drei Maisstärkeproben unbekannter Masse in das Gefäß gegeben. Die Staubkonzentration der Ethanol/Maisstärke-Suspensionen wurde mit beiden Geräten gemessen. Aus den gemessenen Spannungswerten, die das SKG 5 Typ 7 lieferte, wurden die Staubkonzentrationen mit Hilfe von (2.2) berechnet. Die vom SKG 7 Typ 18 gemessenen Staubkonzentrationen konnte direkt auf dem Display abgelesen werden. Die Messergebnisse sind in der Tabelle 5.2 zusammengestellt.
Die von beiden Geräten gelieferten Staubkonzentrationsmesswerte stimmen im Rahmen der Fehlergrenzen überein. Etwas größere Abweichung zeigten sich bei höher-

Versuch 1				
Probe Nr.	c_{5-7} [g/m^3]	c_{7-18} [g/m^3]	Δc_{abs} [g/m^3]	Δc_{rel} [%]
1	77	74	3	3,9
2	154	152	2	1,3
3	244	268	24	8,9
Versuch 2				
Probe Nr.	c_{5-7} [g/m^3]	c_{7-18} [g/m^3]	Δc_{abs} [g/m^3]	Δc_{rel} [%]
1	71	67	4	5,8
2	150	151	1	0,6
3	253	272	19	7,1
Versuch 3				
Probe Nr.	c_{5-7} [g/m^3]	c_{7-18} [g/m^3]	Δc_{abs} [g/m^3]	Δc_{rel} [%]
1	51	46	5	9,8
2	138	139	1	0,7
3	226	245	19	7,7

Tabelle 5.2: Ergebnisse der Staubkonzentrationsvergleichsmessungen.

en Staubkonzentrationen (maximal 8,9 %), wobei das SKG 7 Typ 18 gegenüber dem SKG 5 Typ 7 zu höheren Messwerten hin tendiert. Der Hauptgrund dafür ist, dass der Empfängerstrom beim SKG 7 stets etwas zu niedrig angezeigt wird. Dies gilt vor allem für höhere Staubkonzentrationen bzw. für niedrige Empfängerströme (Abschnitt 5.5). Darüberhinaus wird bei der Messeinheit Typ 18 das Messvolumen zweimal vom Lichtstrahl durchlaufen. Dadurch ist die Kalibrierkurve im betreffenden Staubkonzentrationsbereich flacher als beim Typ 7 (Abbildung 5.1). Je flacher die Kalibrierkurve ist, desto größer sind die Messunsicherheiten in c. Bei einer Messung in Luft wird diesem Umstand damit begegnet, dass der Messabstand l verkleinert wird. Da bei den hier durchgeführten Messungen l jedoch durch die Abmessungen des Kalibriergefäßes auf 37,2 mm festgelegt ist, ist der Messfehler beim SKG 7 Typ 18 bei höheren Staubkonzentrationen größer als beim SKG 5 Typ 7. Folgende weitere Faktoren sorgen für Messfehler:

1. Eine zeitlich stabile und vollständige homogene Verteilung des Staubes im Kalibriergefäß ist nicht erreichbar.

2. Die Ausrichtung des Kalibriergefäßes im Strahlengang kann nicht exakt reproduziert werden.

3. Es wurden die in Abschnitt 5.1 ermittelten unterschiedlichen Extinktionskoeffizienten verwendet.

5.3 Stabilität des Senderstroms

Bei dieser Untersuchung wurde geprüft, ob sich der eingestellte Senderstrom im Laufe der Zeit verändert (Drift). Die Regeleinheit wurde so programmiert, dass sie einen definierten Strom durch den Sender beim Einschalten des Gerätes vorgibt. Drei Versuche wurden durchgeführt, wobei der Strom von 5 mA über 10 mA auf 15 mA im dritten Versuch eingestellt wurde. Die Erfassung des Senderstroms geschah mithilfe eines Datenloggers mit einer Messrate von 1 Hz. Die Messdauer pro Versuch betrug eine Stunde. Die Messergebnisse sind in der Abbildung 5.2 grafisch dargestellt. Für jeder Kurve wurde die Neigung des horizontalen Astes berechnet. Sie beträgt für jede Kurve Null. Der eingestellte Strom bleibt dauerhaft konstant.

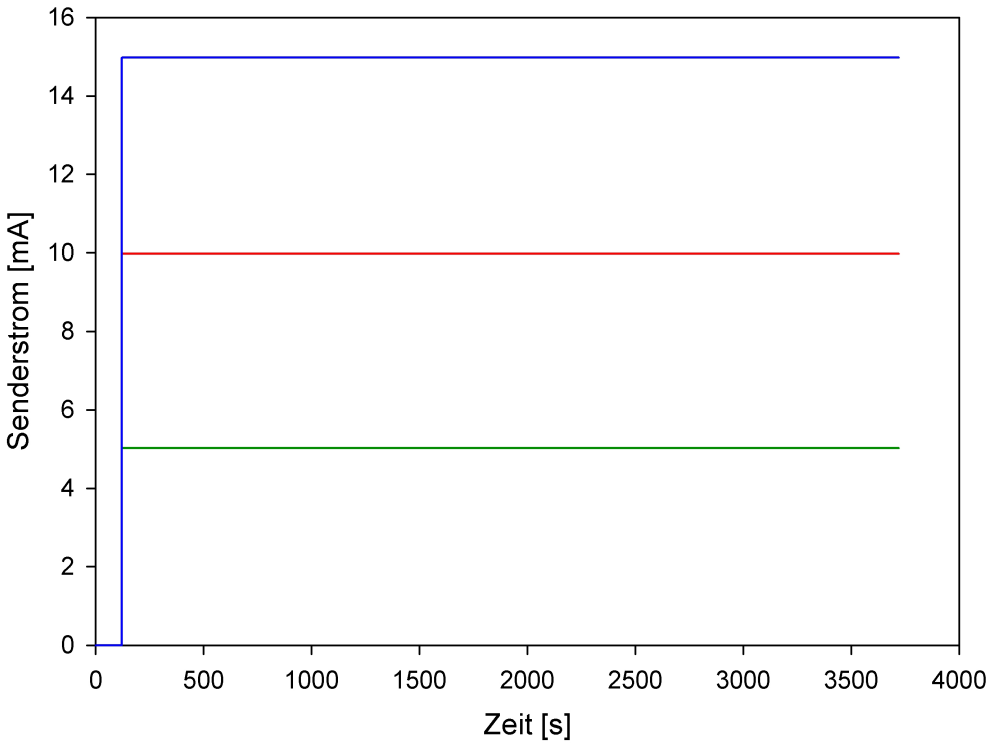

Abbildung 5.2: Überprüfung der Senderstromstabilität.

5.4 Linearität des Senderstroms

Das Ziel dieses Tests war es zu untersuchen, ob der tatsächliche Strom durch den Sender den von der Regeleinheit angeforderten Wert annimmt und wie hoch die maximale Abweichung vom Soll-Wert ist. Dies wurde für den gesamten Bereich von 0 mA bis 20 mA untersucht. Dazu wurde die Regeleinheit so programmiert, dass sie den Sendertreiber dazu auffordert, den Senderstrom nach jeweils 30 s solange in 0,25 mA-Schritten zu erhöhen bis 20 mA erreicht sind. Beginnend bei 0 mA ergeben sich daraus $20/0,25 = 80$ Schritte. Der Strom durch den Sender $I_{Sender,ist}$ wurde mithilfe eines Datenloggers gemessen. Bei einer Messrate von 15 Hz wurden für jede Stufe $30 \cdot 15 = 450$ Messwerte

aufgenommen. Um die zeitlichen Schwankungen des Senderstroms bildlich zeigen zu können, wurde ein zeitlicher Ausschnitt ($t = 90 \ldots 360\,\mathrm{s}$) der Messergebnisse in Abbildung 5.3 grafisch dargestellt. Im Diagramm sind die Stromschwankungen um den Soll-Wert deutlich zu erkennen.

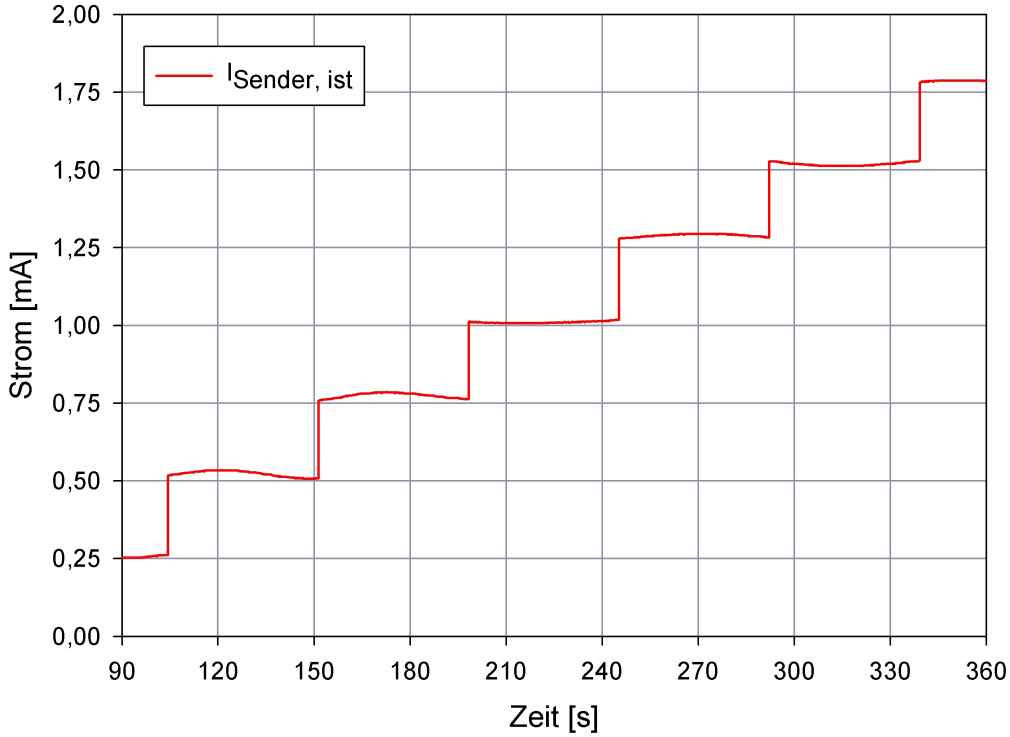

Abbildung 5.3: Schwankungen des gemessenen Senderstroms $I_{Sender,ist}$ um den eingestellten Soll-Wert $I_{Sender,soll}$.

Für jede der 80 Stufen wurde der Mittelwert über die 450 Messwerte gebildet und über den Soll-Stromwerten aufgetragen. Zusätzlich wurden in dieses Diagramm, welches in Abbildung 5.4 zu sehen ist, auch die Soll-Stromwerte eingezeichnet. Beide Kurven sind nahezu deckungsgleich, d. h. $I_{Sender,ist}$ weicht im zeitlichen Mittel nur sehr geringfügig von $I_{Sender,soll}$ ab.

Wichtig war nun die Beantwortung der Frage, wie groß die maximalen Differenzen zwischen Ist- und Soll-Werten des Senderstroms pro Stufe sind. In Abbildung 5.5 ist diese maximale Abweichung des gemessenen Stromes vom angeforderten Soll-Strom in % zu sehen. Sie beträgt maximal 11,9 % bei $I_{Sender,soll} = 0{,}25\,\mathrm{mA}$ und fällt mit steigendem Strom durch den Sender. Dieses Ergebnis ist plausibel, da sich bei sehr kleinen Stromwerten Störungen und Rauschen besonders bemerkbar machen. Wie verschiedene Versuche mit dem SKG 7 Typ 18 beim Nullabgleich jedoch zeigten, können prinzipiell keine kleineren Senderströme als 5,29 mA entstehen. Bei einem eingestellten Senderstrom von $I_{Sender,soll} = 5{,}29\,\mathrm{mA}$ beträgt die maximale Abweichung des Ist-Stromwertes nur noch 0,73 % oder 0,038mA absolut. Die Abweichungen sinken weiter bei steigenden Soll-Stromwerten. Diese geringen Abweichungen stellen für eine ausreichend genaue Staubkonzentrationsmessung kein Problem dar.

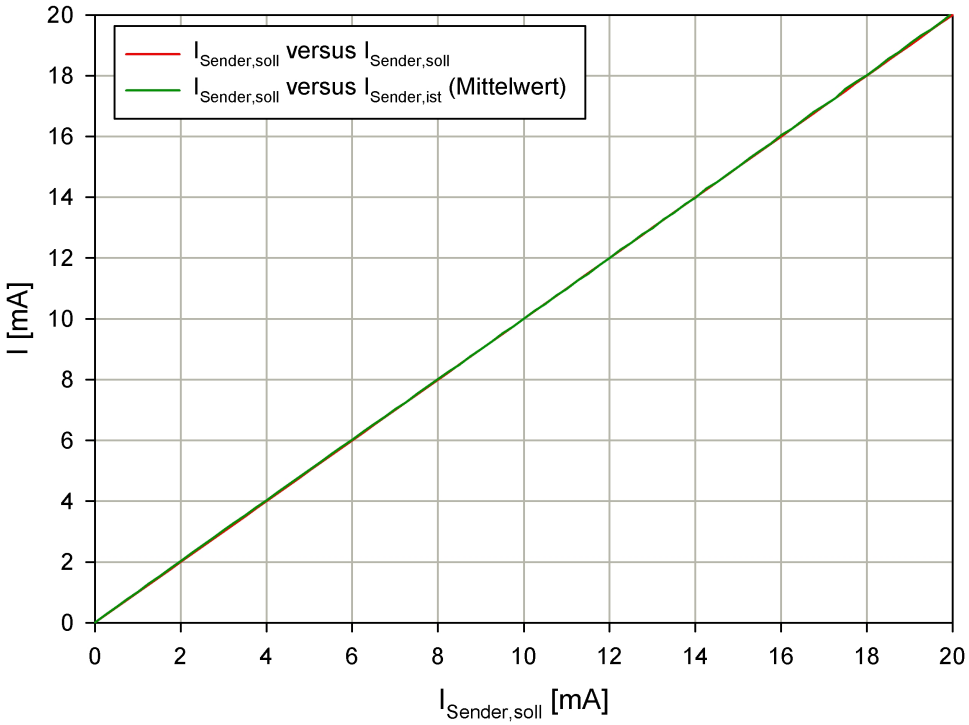

Abbildung 5.4: Zeitlich gemittelte Ist-Werte und Soll-Werte des Senderstroms aufgetragen über den Letztgenannten (oben).

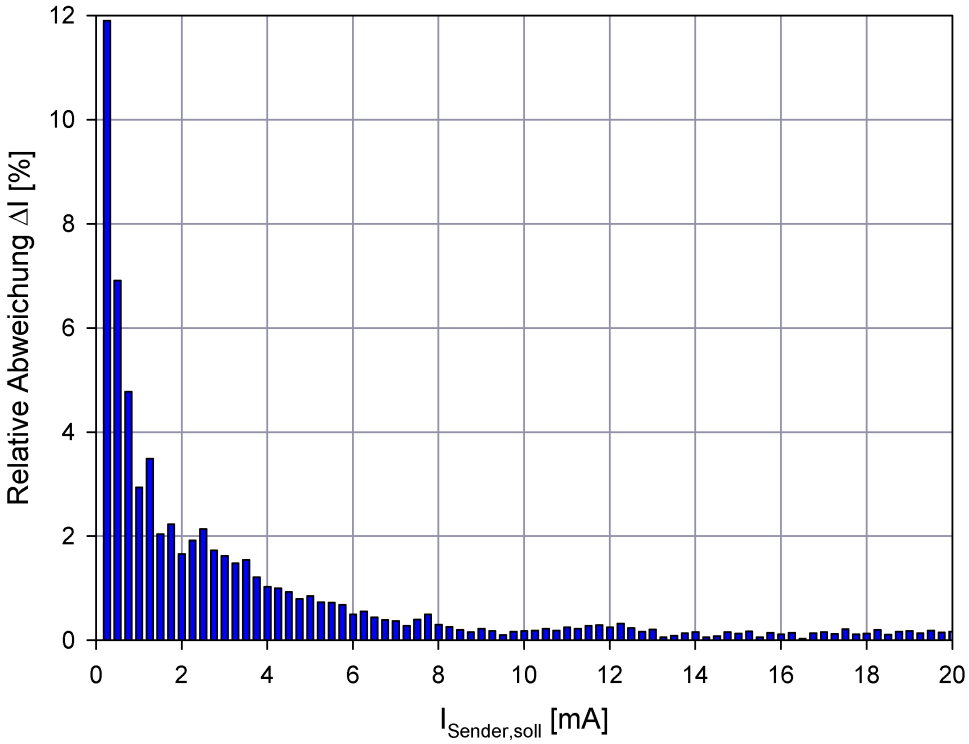

Abbildung 5.5: Maximale relative Abweichung des Senderstrom-Ist-Werts vom Soll-Wert als Funktion vom Soll-Wert des Senderstroms.

5.5 Untersuchung der Empfängerstrommessung

Bei dieser Untersuchung wurde geprüft, wie genau der Empfängerstrom von der Steuereinheit gemessen und angezeigt wird. Dazu wurde der Empfängerstrom mit einem Präzisionsmultimeter[1] mit hoher Auflösung gemessen und mit dem auf dem Display des SKG 7 angezeigten Empfängerstrom verglichen. Die Abweichung der vom SKG 7 angezeigten Stromwerte von den gemessenen Strömen wurde in % bezogen auf die gemessenen Stromwerte berechnet. Das Ergebnis des Tests ist in der Abbildung 5.6 zu sehen. Die maximale Abweichung betrug 0,059 mA bzw. knapp 7 %, wobei sie mit steigendem Empfängerstrom stark fällt. Auffällig ist, dass die Abweichung stets positiv ist, d. h. das SKG 7 zeigt tendenziell weniger Strom an als tatsächlich durch den Empfänger fließt (Tabelle D.1). Dies gilt insbesondere für kleine Empfängerströme. Die Konsequenz daraus ist, dass systematisch zu hohe Staubkonzentrationen angezeigt werden, was vor allem bei hohen Staubkonzentrationen zum Tragen kommt. Dies erklärt die durchweg höheren Staubkonzentrationsmesswerte gegenüber dem SKG 5 Typ 7 in Abschnitt 5.2.

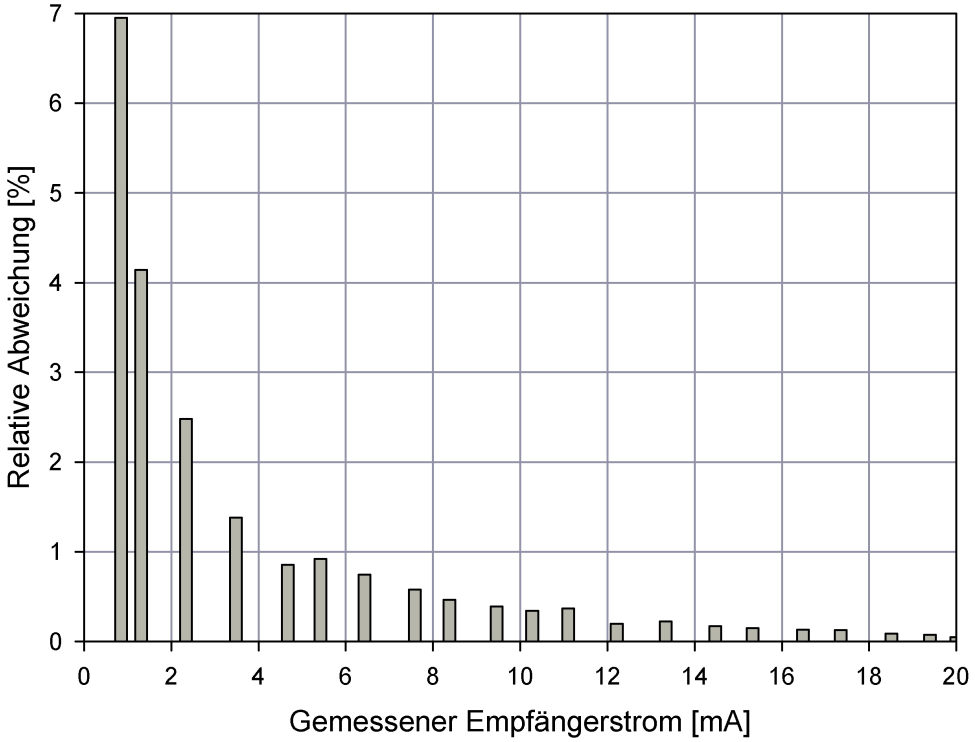

Abbildung 5.6: Relative Abweichung zwischen tatsächlichem und am SKG 7 angezeigtem Empfängerstroms.

Für die Suche nach der Ursache dieses Verhaltens stand keine Zeit mehr zur Verfügung, weshalb an dieser Stelle auf den Ausblick (Abschnitt 6.2.8) verwiesen werden muss. Mögliche Fehlerquelle könnte die Elektronik im Empfängertreiber sein oder die Berechnung der Staubkonzentration im Mikrocontroller oder eine Kombination aus beiden.

[1] Model Kethley 2001.

Im Empfängertreiber könnten geringe Offsetströme in den Operationsverstärkern OP1 und OP2 auftreten. Möglich wäre auch, dass der Messwiderstand R1 nicht exakt genug bestimmt wurde. Auch die Berechnungsoperationen im Mikrocontroller könnten der Grund sein, da sie wegen der 8 Bit-Auflösung mit einem kleinen Fehler behaftet sind.

5.6 Gerätelangzeittest

Das Ziel des Testes war, die Funktionsfähigkeit des Gerätes für lange Betriebsdauern zu überprüfen. Das Gerät wurde für eine Zeit von 12 Stunden eingeschaltet. Im Gerät wurde ein Ni-CrNi Thermoelement zur Temperaturmessung eingebaut. Als Messpunkt wurde die Mitte des Geräteinneren ausgewählt (Abbildung 5.7).

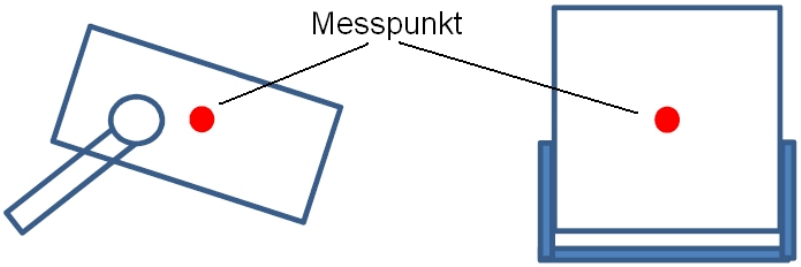

Abbildung 5.7: Temperaturmesspunkt im SKG 7 während des Langzeittests (schematisiert).

Nach zwölf Stunden Laufzeit wurde das Gerät auf seine Funktionsfähigkeit überprüft. Das Gerät arbeitete vollkommen fehlerfrei. Das Diagramm in Abbildung 5.8 zeigt den gemessenen Temperaturverlauf. Nach dem Einschalten des Gerätes steigt die Temperatur so lange, bis sich ein thermisches Gleichgewicht im Geräteinneren eingestellt hat. Die Wärmeabgabe durch die elektronischen Bauteile ist dann genauso groß wie der Wärmeverlust durch die Lüftungsschlitze des Gehäuses. Über Nacht sank die Temperatur im Labor, was das Sinken der Temperatur im Geräteinneren zur Folge hatte.

5.7 Feuchte- und Temperaturmessung

Das SKG 7 soll dazu in der Lage sein, die Temperatur und relative Feuchte der Umgebungsluft zu messen und anzuzeigen. Ob diese Messung korrekt von der Steuereinheit durchgeführt wird, wurde erneut durch eine Vergleichsmessung überprüft. Als Referenzmessgerät kam das kalibrierte Novasina MS 1 zum Einsatz. Gemessen wurde am Donnerstag, den 27.10.2016 um 09:10 Uhr im Freien in 1 m Höhe über dem Erdboden. Um 10:00 Uhr wurde im Elektronik-Labor der BGN nochmals gemessen. Die Messergebnisse sind in Tabelle 5.3 aufgelistet.
Die Messtoleranzen des Novasina MS 1 sind mit 1 % bei der Luftfeuchte angegeben und mit 0,3 °C bei der Temperatur. Zusammen mit den in Abschnitt 4.4 angegebenen Toleranzen des im SKG 7 verwendeten Sensors SHT75 dürfen bei korrekter Funktion beider

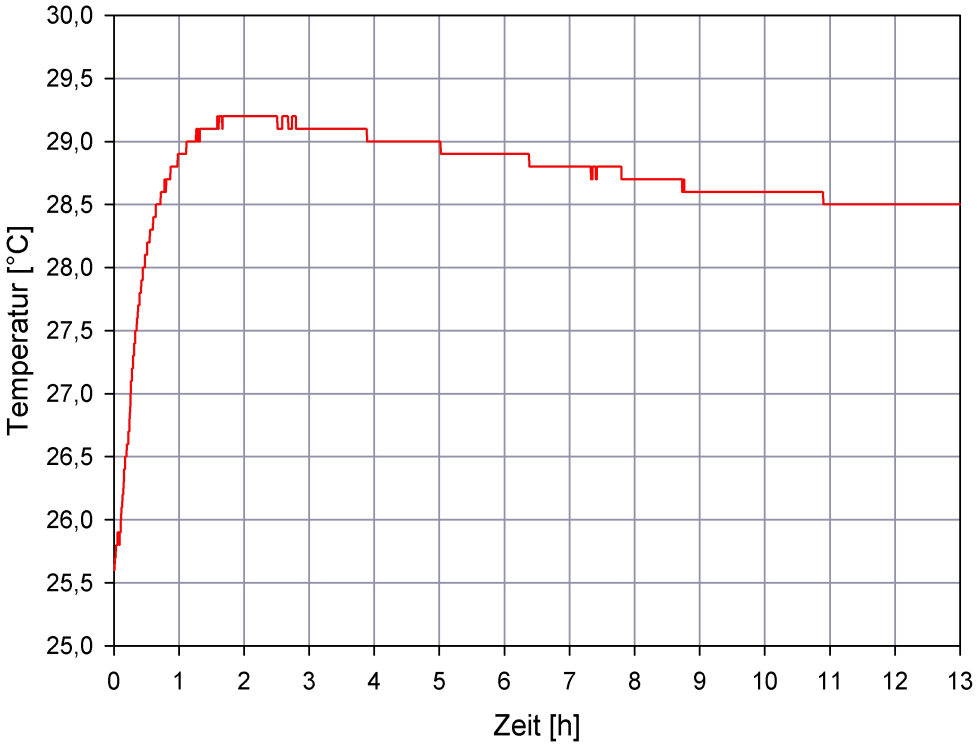

Abbildung 5.8: Zeitlicher Temperaturverlauf im Inneren des SKG 7 während des Langzeittests.

Messsysteme die Messergebnisse in Tabelle 5.3 maximal um 2,8 % in der Luftfeuchte und um 0,6 °C in der Temperatur voneinander abweichen. Diese Fehlertoleranzen wurden eingehalten. Die Feuchte- und Temperaturmessung beim SKG 7 funktioniert ordnungsgemäß.

	Novasina MS 1		SKG 7	
	T [°C]	F [%]	T [°C]	F [%]
Draußen	8,4	85,3	8,2	86,4
Elektroniklabor	21,7	39,2	21,6	40,7

Tabelle 5.3: Ergebnisse der vergleichenden Temperatur- und Feuchtemessung.

Kapitel 6

Diskussion und Ausblick

6.1 Diskussion der Ergebnisse

Alle im Abschnitt 1.3 (bzw. Abschnitt 3.2.1) aufgelisteten Ziele der Bachelorarbeit wurden erreicht. Nach einer Untersuchung der Lösungsmöglichkeiten zur Steuerung der Messeinheit wurde eine Embedded System-Lösung favorisiert, entwickelt und gebaut. Das Gerät trägt den Namen SKG 7 (Staubkonzentrationsmessgerät 7. Entwicklungsstufe).

1. Das SKG 7 verfügt über ein Touchdsiplay, über welches der Apparat bedient wird. Auf dem Display wird die Staubkonzentration in g/m^3 direkt angezeigt.

2. Der Strom durch den Sender wird vom Gerät geregelt. Auch wird das Empfängersignal automatisch gemessen und ausgewertet.

3. Die Zielvorgabe, das System mit einem automatischen Nullabgleich zu versehen, wurde realisiert.

4. Die Integration einer Feuchte- und Temperaturmessung in der Umgebung der Steuereinheit wurde erreicht. Die durch einen Sensor gewonnenen Feuchte- und Temperaturdaten werden auf dem Display ausgegeben.

Die korrekte Funktionsweise der Steuereinheit bzw. des kompletten Messsystems SKG 7 Typ 18 wurde durch verschiedene Untersuchungen bestätigt (Kapitel 5).

6.2 Ausblick

Neben den in Abschnitt 1.3 beschriebenen Zielen dieser Arbeit, die mit guten Ergebnissen erreicht wurden, gab es weitere, optionale Ziele (Abschnitt 3.2.1). An dieser Stelle sollen die nicht verwirklichten optionalen Ziele sowie weitere mögliche Verbesserungen und Optimierungen aufgezeigt werden.

6.2.1 Verwaltung der Stäube

In der aktuellen Ausführung des Gerätes wird der zu messenden Staub aus einer festen Liste von Stäuben ausgewählt. Zu dieser Liste können keine Stäube über das Touchdisplay hinzugefügt werden. Im Hinblick auf zukünftige Messungen mit unterschiedlichsten Stäuben sollte eine Verwaltungsfunktion mit folgenden Eingabemöglichkeiten eingebaut werden:

- Hinzufügen eines Staubes mitsamt seines Extinktionskoeffizienten zur Staubliste und dessen permanente Speicherung.
- Ändern von Einträgen in der Staubliste.
- Löschen von Einträgen in der Staubliste.

6.2.2 Messdatenaufzeichnung und -speicherung

Die realisierte Ausführung des SKG 7 kann lediglich die gegenwärtig vorherrschende Staubkonzentration in einem Medium (in der Regel Luft) anzeigen. Oft ist jedoch auch der zeitliche Verlauf der Staubkonzentration von Interesse. Deshalb ist ein weiteres Ziel, den Staubkonzentrationsverlauf über der Zeit auf dem Display des Gerätes während der Messung darstellen zu können. Durch einen eingebauten Speicher könnten die Messdaten direkt im Gerät gespeichert und in ihrer Gesamtheit zur Anzeige gebracht werden. Die Integration eines Speichers ist aus diesem Grund anzustreben.

6.2.3 Optimierung der Nullabgleichsroutine

Während des Nullabgleichs wird der Strom durch den Sender von 0 mA sukzessiv um 0,25 mA erhöht bis der Empfänger einen Strom von 18 mA liefert. Danach wird der Senderstrom schrittweise um 0,005 mA erhöht bis der Empfänger einen Strom von 19,95 mA bis 20,05 mA liefert. Der Empfängerstrom wird bei jedem Schritt gemessen. Die Einstellung und Messung des Stromes nimmt einige Millisekunden in Anspruch. Die gesamte Nullabgleichsroutine kann daher länger als eine Minute dauern. Eine Optimierung bzw. Beschleunigung der Methode ist daher begrüßenswert.

6.2.4 Implementierung einer Kalibrierroutine

Die Ermittlung des Extinktionskoeffizientes eines Staubes erfolgt mithilfe eines Regressionsanalyse-Programms auf einem Computer. Um von einem externen Computer unabhängig zu werden, könnte eine Kalibrierroutine in das Gerät implementiert werden. Dies könnte so ausgeführt werden, dass der Benutzer die Masse der Staubproben und das Ethanolvolumen im Kalibriergefäß über das Touchdisplay eingibt. Die Ermittlung und Speicherung des Extinktionskoeffizienten könnte im Gerät durch ein Programm durchgeführt werden.

6.2.5 Schnittstelle zum PC

Über eine Schnittstelle zu einem Computer, die dem SKG 7 derzeit noch fehlt, könnten folgende Funktionen realisiert werden:

- Steuerung des Gerätes über einen PC.
- Übertragung und Speicherung von Messdaten auf einen PC.
- Aktualisierung der Gerätesoftware.
- Ein Remote Sensing, d. h. Messung über große Entfernungen.
- Gleichzeitige Aufnahme von Messdaten von mehreren Messgeräten.

6.2.6 Displayschutz

An den Einsatzorten des SKG 7 herrschen gelegentlich raue Umgebungsbedingungen. Zum Beispiel muss damit gerechnet werden, dass das Gerät in staubreicher Umgebung eingesetzt wird, was zu Staubablagerungen auf dem Touchdisplay führen kann. Möglich sind auch Messungen im Winter, wobei das Gerät im Freien eingesetzt wird. Das Touchdisplay ist in der Frontplatte eingebaut und hat direkten Kontakt zur Umgebung. Wird es verschmutzt oder nass, funktioniert das Touchdisplay möglicherweise nicht mehr einwandfrei. Der Einbau eines Schutzes für das Touchdisplay ist deshalb zu erwägen. Dies könnte eine spezielle Schutzfolie sein, welche die Touchfunktion des Displays nicht behindert.

6.2.7 Platinen

Die Sender- und Empfängertreiber sind als Prototypen auf einer Lochrasterplatine realisiert. Eine Lochrasterplatine ist sehr gut für einen Prototypaufbau geeignet. Die Bauteile können beliebig auf der Platine platziert werden, relativ schnell verdrahtet und ohne Aufwand ein und ausgebaut werden.
Für eine mögliche Serienherstellung des Gerätes bringen Lochrasterplatinen den großen Nachteil mit sich, dass die Verdrahtung der elektronischen Bauteile manuell durchgeführt wird. Dies nimmt sehr viel Zeit in Anspruch und verursacht als Folge hohe Herstellungskosten. Die hohen Herstellungszeiten lassen sich durch die Entwicklung und Herstellung einer Leiterplatine optimieren. Wurde das Layout der Platine einmal entwickelt, können die Platinen bei einem Platinenhersteller bestellt werden. Die Kosten dafür sind stückzahlabhängig, liegen aber schon bereits ab 10 Platinen im Preisbereich einer Lochrasterplatine. Die Bestückungszeit einer solchen Platine ist erheblich kleiner als diejenige einer Lochrasterplatine. Der Prozess kann noch weiter optimiert werden, indem SMD-Bauteile eingesetzt werden. Durch Verwendung von SMD-Bauteilen kann der Hersteller die Platine automatisiert bestücken.

In Sender- und Empfangertreiber befinden sich 12 Bit-D/A- und -A/D-Bauteile. Um eine 12 Bit-Auflösung der Elektronik zu gewährleisten, müssen bestimmte Designregeln im Layout eingehalten werden. Einige Punkte davon sind:

- Die Bypass-Kondensatoren sollen so nah wie möglich an den Pins der Bauteile platziert werden.
- Die Fläche des GND-Signals soll so groß wie möglich sein.
- Die Signalleitungen sollen so kurz wie möglich sein.
- Der digitale und der analoge Teil der Platine sollen getrennt voneinander vorliegen.

Alle diese Voraussetzungen sind auf einer Leiterplatine leichter als auf einer Lochrasterplatine umzusetzen. Das ist ein weiterer Vorteil der Leiterplatten.

6.2.8 Empfängerstrom

In den Abschnitten 5.2 und 5.5 wurde festgestellt, dass das SKG 7 die Tendenz zeigt, vor allem bei vorliegenden hohen Staubkonzentrationen etwas zu große Konzentrationswerte anzuzeigen. Der Ursache dieses systematischen Fehlers muss auf den Grund gegangen und der Fehler behoben werden. Dieser Punkt hat oberste Priorität vor allen anderen, die in diesem Abschnitt genannt wurden.

Literaturverzeichnis

[1] W. Bartknecht: *Staubexplosionen*, Springer-Verlag, 1987.

[2] T. Beier, T. Mederer: *Messdatenverarbeitung mit LabVIEW*, 1. Auflage, Carl Hanser Verlag, München, 2015.

[3] P. Dyrba: *Verkleinerung des Staubkonzentrationsmessgeräts SKG 5*, Diplomarbeit, Fachhochschule Heidelberg und Forschungsgesellschaft für angewandte Systemsicherheit und Arbeitsmedizin, 2009.

[4] R. Felderhoff, U. Freyer: *Elektrische und elektronische Messtechnik*, 7. Auflage, Carl Hanser Verlag, München, 2003.

[5] I. Kukujzev: *Entwicklung einer neuen Kalibriermethode für das Staubkonzentrationsmessgerät SKG 5*, Diplomarbeit, Hochschule Mannheim und Forschungsgesellschaft für angewandte Systemsicherheit und Arbeitsmedizin, 2008.

[6] R. Lerch; *Elektrische Messtechnik*, 4. Auflage, Springer-Verlag, Heidelberg, 2007.

[7] M. Nicklisch: *Verbesserung des Staubkonzentrationsmessgeräts SKG 5*, Diplomarbeit, Fachhochschule Mannheim und Forschungsgesellschaft für angewandte Systemsicherheit und Arbeitsmedizin, 2005.

[8] W. Richter: *Elektrische Meßtechnik*, 3. Auflage, Verlag Technik, Berlin, 1994.

[9] H. Schiebler: *Private Mitteilung*, Berufsgenossenschaft Nahrungsmittel und Gastgewerbe, 2016.

[10] A. Schiessl: *Umgestaltung des Staubkonzentrationsmessgeräts SKG 5 zum Einsatz in explosionsgefährdeten Zonen*, Bachelorarbeit, Hochschule Mannheim und Forschungsgesellschaft für angewandte Systemsicherheit und Arbeitsmedizin, 2012.

[11] E. Schrüfer, L. Reindl, B. Zagar: *Elektrische Messtechnik*, 11. Auflage, Carl Hanser Verlag, München, 2014.

[12] M. Seithel: *Private Mitteilung*, Berufsgenossenschaft Nahrungsmittel und Gastgewerbe, 2016.

[13] Verein Deutscher Ingenieure: VDI Richtlinie 2263 Blatt 9: Staubbrände und Staubexplosionen, Gefahren - Beurteilung - Schutzmaßnahmen, Bestimmung des Staubungsverhaltens von Schüttgütern, Beuth Verlag, 2008.

[14] H. Weber: Rechnergestützte Meßverfahren, 1. Auflage, Vogel Buchverlag, 1996.

[15] J. Wolf: *Grundkurs C*, 2. Auflage, Rheinwerk Verlag, Bonn, 2016.

Anhang A

Schaltpläne

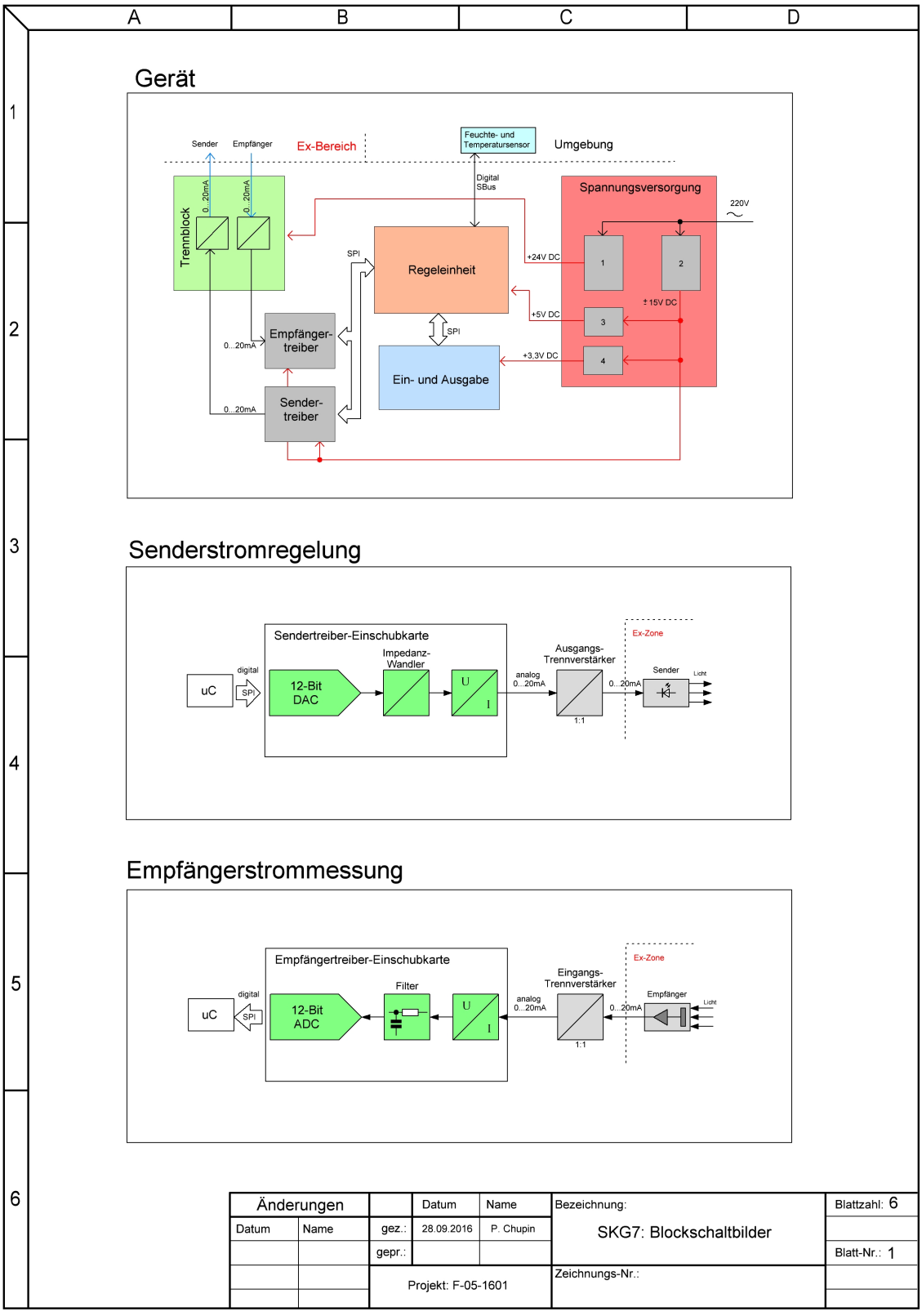

Abbildung A.1: Blockschaltbild für das SKG 7 sowie für den Sender- und Empfängertreiber.

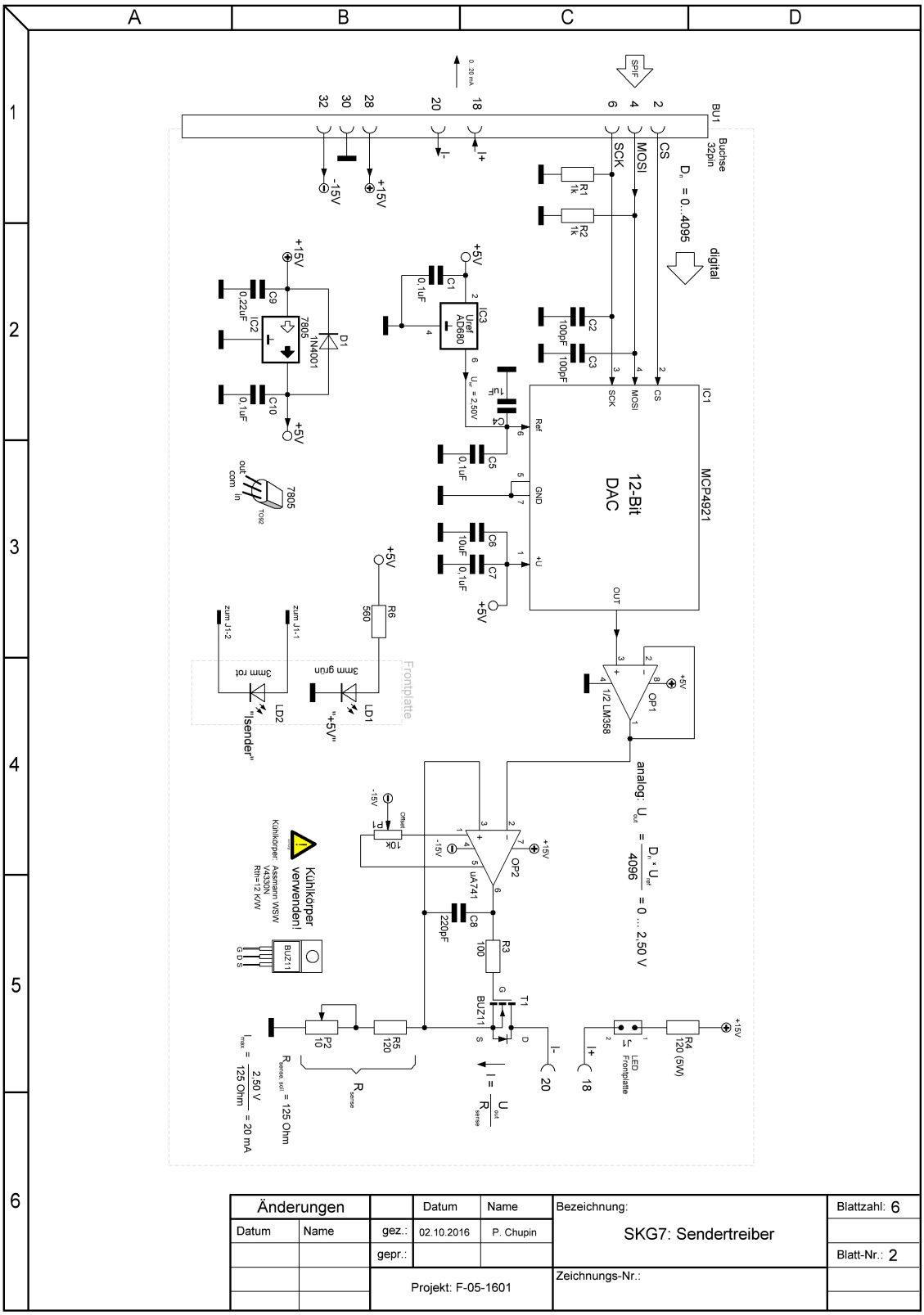

Abbildung A.2: Schaltplan für den Sendertreiber.

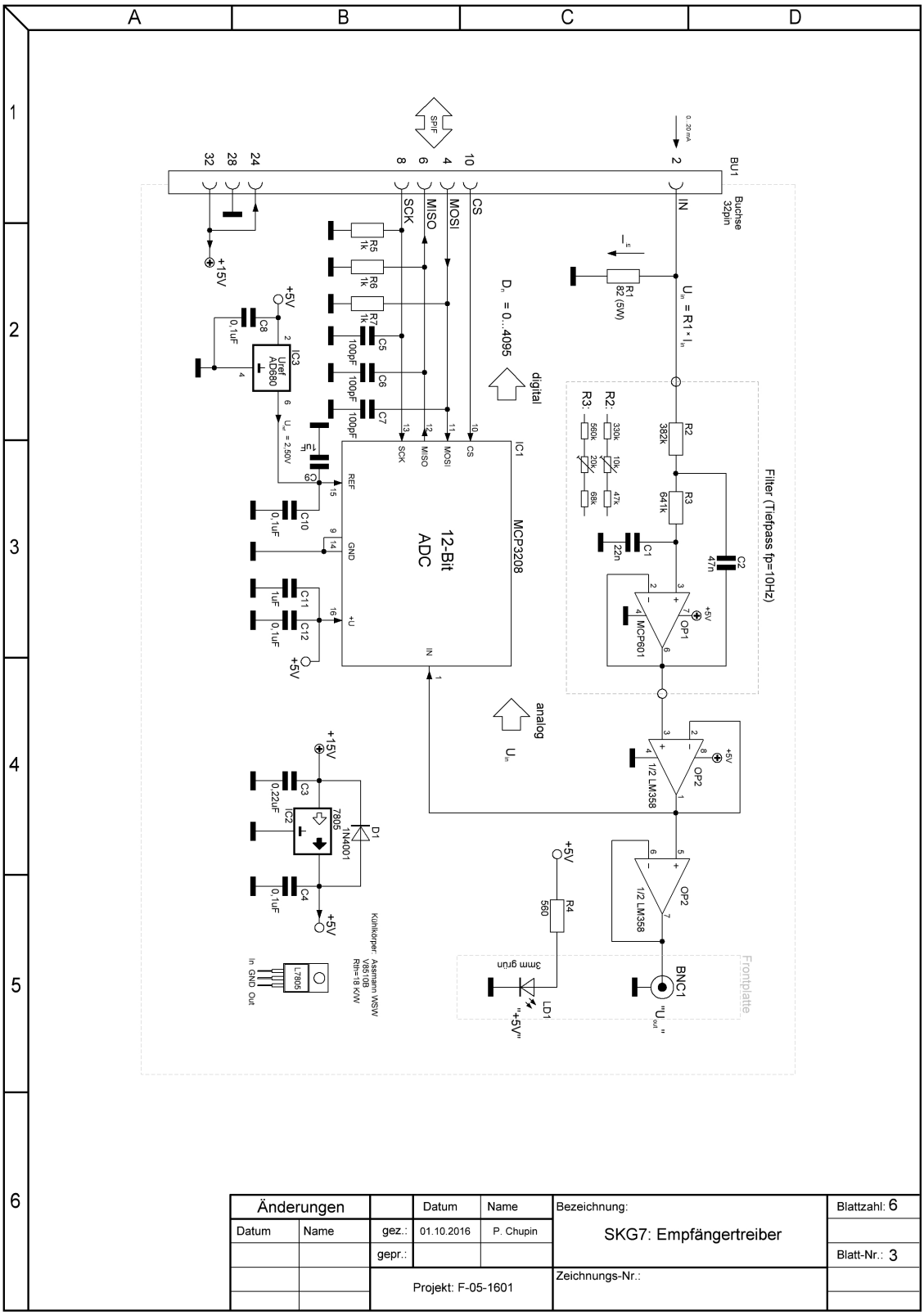

Abbildung A.3: Schaltplan für den Empfängertreiber.

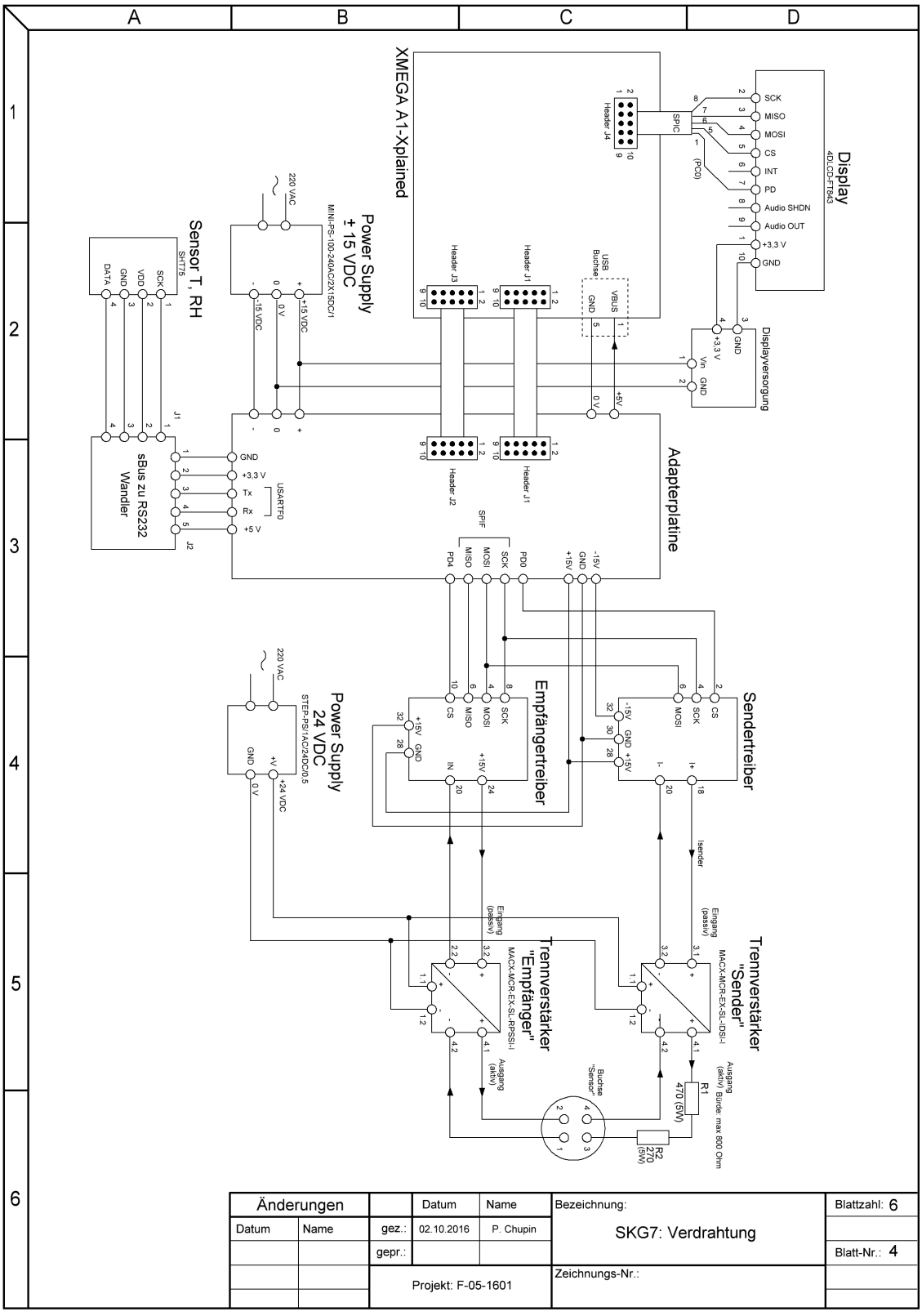

Abbildung A.4: Verdrahtungsschaltplan für das SKG 7.

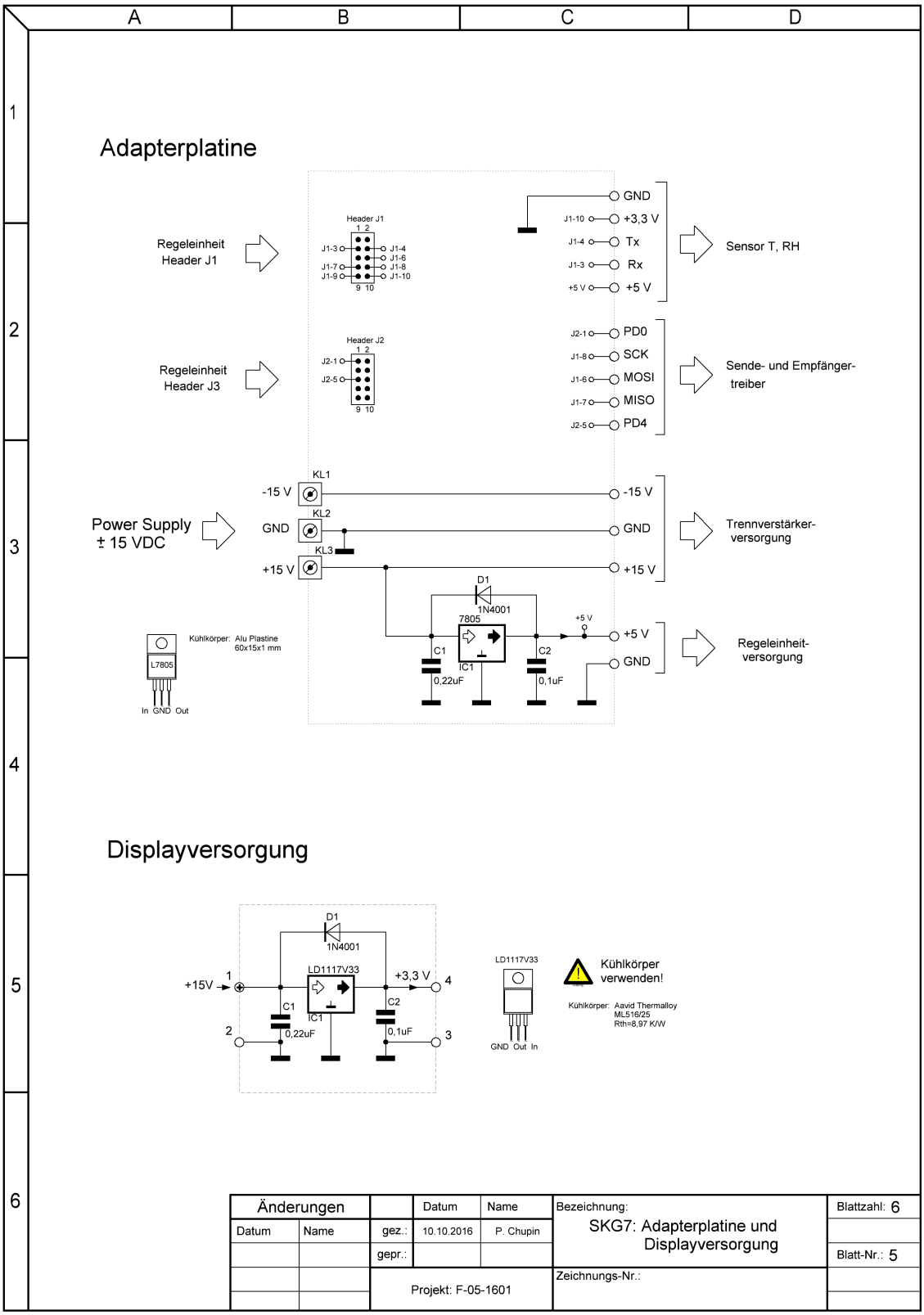

Abbildung A.5: Schaltplan für die Adapterplatine und die Displayversorgung.

Sendertreiber	
Widerstände	
R1, R2	1 kOhm
R3	100 Ohm
R4, R5	120 Ohm, 5 W
R6	560 Ohm
P1	10 kOhm
P2	10 Ohm
Kondensatoren	
C1, C5, C7, C10	0,1uF
C2, C3	100 pF
C4	1 uF
C6	10 uF
C8	220 pF
C9	0,22 uF
Halbleiter	
IC1	MCP4921
IC2	LM7805AC2, TO-92,100 mA
IC3	AD680
OP1	LM358
OP2	uA741
T1	BUZ11, TO-220
D1	1N4001
LD1	LED 3 mm, grün
LD2	LED 3 mm, rot
Sonstiges	
BU1	Messerleiste abgewinkelt, RM = 2,54 mm, 32 pin, 3 Reihen, DIN41612, Typ Harting 09031326921
J1	Stiftleiste gerade, 2-polig, RM =2,54 mm
-	IC-Fassung, 8-polig, MP08.3 STG BU
-	Kühlkörper Assmann WSW V4330N

Empfängertreiber	
Widerstände	
R1	82 Ohm, 5 W
R2	330 kOhm + P1 + 47 kOhm
R3	560 kOhm + P2 + 68 kOhm
R4	560 Ohm
R5, R6, R7	1 kOhm
P1	10 kOhm
P2	20 kOhm
Kondensatoren	
C1	22 nF
C2	47 nF
C3	0,22 uF
C4, C8, C10, C12	0,1 uF
C5, C6, C7	100 pF
C9, C11	1 uF
Halbleiter	
IC1	MCP3208
IC2	L7805CV, TO-220, 1 A
IC3	AD680
OP1	MCP601
OP2	LM358
D1	1N4001
LD1	LED 3 mm, grün
Sonstiges	
BU1	Messerleiste abgewinkelt, RM = 2,54 mm, 32 pin, 3 Reihen, DIN41612, Typ Harting 09031326921
BNC1	BNC-Steckverbinder Buchse, J01001F1222
-	IC-Fassung, 8-polig, MP08.3 STG BU
-	Kühlkörper Assmann WSW V8510B

Verdrahtung	
Widerstände	
R1	470 Ohm, 5 W
R2	270 Ohm, 5 W
Sonstiges	
-	Einbaukupplung, 4-polig, Flansch, Typ Lumberg 0308-04
-	Flachbandkabel, RM = 1,27 mm, 10-adrig
-	Buchsenleiste 10-polig, RM = 2,54 mm, Typ 10 FCI
-	Mini USB B Stecker, BKL10120252

Adapterplatine	
Kondensatoren	
C1	0,22 uF
C2	0,1 uF
Halbleiter	
IC1	L7805CV, TO-220, 1 A
D1	1N4001
Sonstiges	
J1, J2	Stiftleiste, RM = 2,54 mm, 10-polig, Typ BKL10120554
KL1, KL2, KL3	Schraubklemme, 1-polig, RM = 2,54 mm
-	Kühlkörper: Alu-Plastine 60x15x1 mm

Displayversorgung	
Kondensatoren	
C1	0,22 uF
C2	0,1 uF
Halbleiter	
D1	1N4001
IC1	LD1117V33
Sonstiges	
-	Kühlkörper: Aavid Thermalloy ML516/25

Änderungen			Datum	Name	Bezeichnung:	Blattzahl: 6
Datum	Name	gez.:	15.10.2016	P. Chupin	SKG7: Stückliste	
		gepr.:				Blatt-Nr.: 6
		Projekt: F-05-1601			Zeichnungs-Nr.:	

Abbildung A.6: Stückliste.

Anhang B

Konstruktionszeichnungen

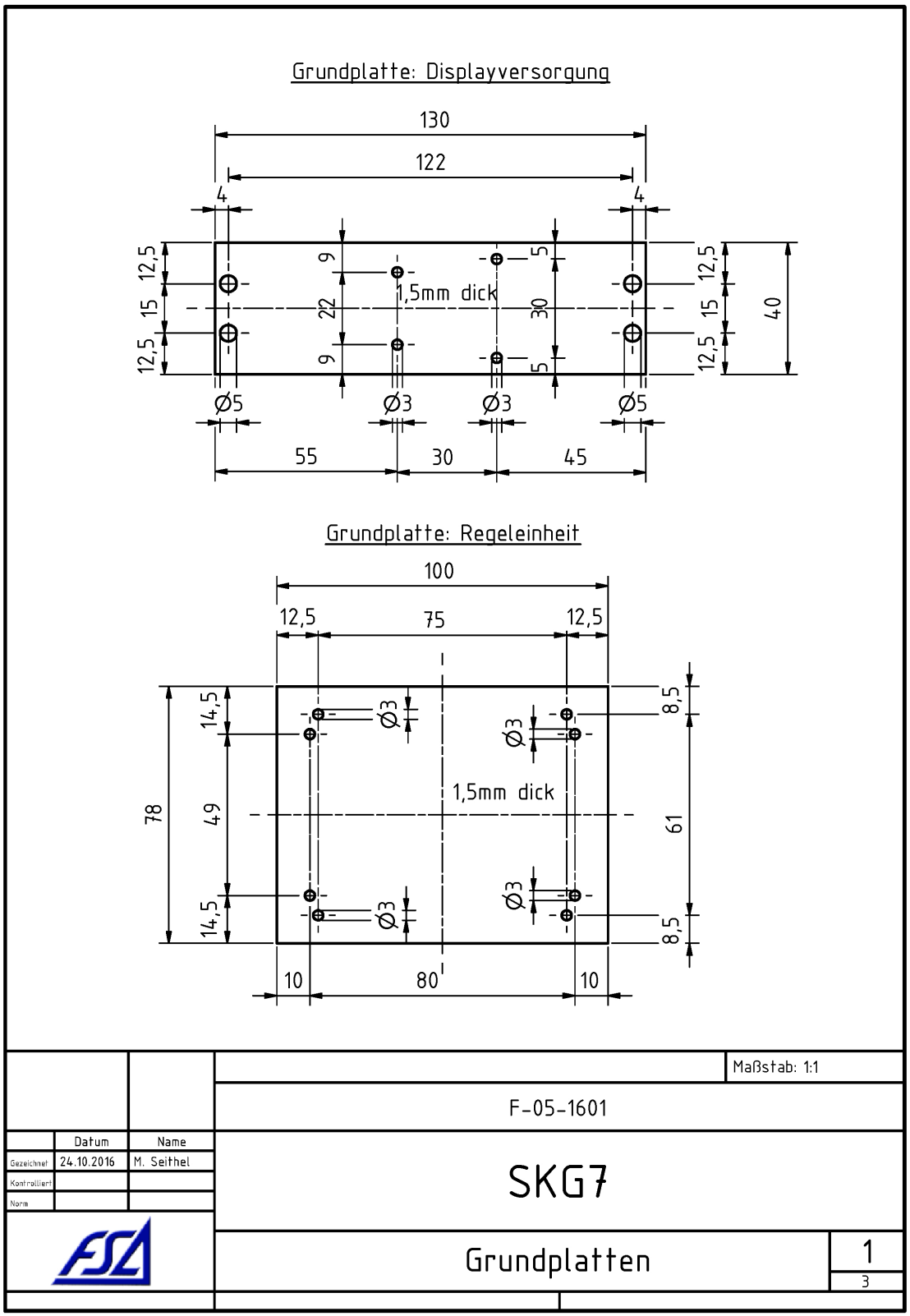

Abbildung B.1: CAD-Zeichnung für die Grundplatten [12].

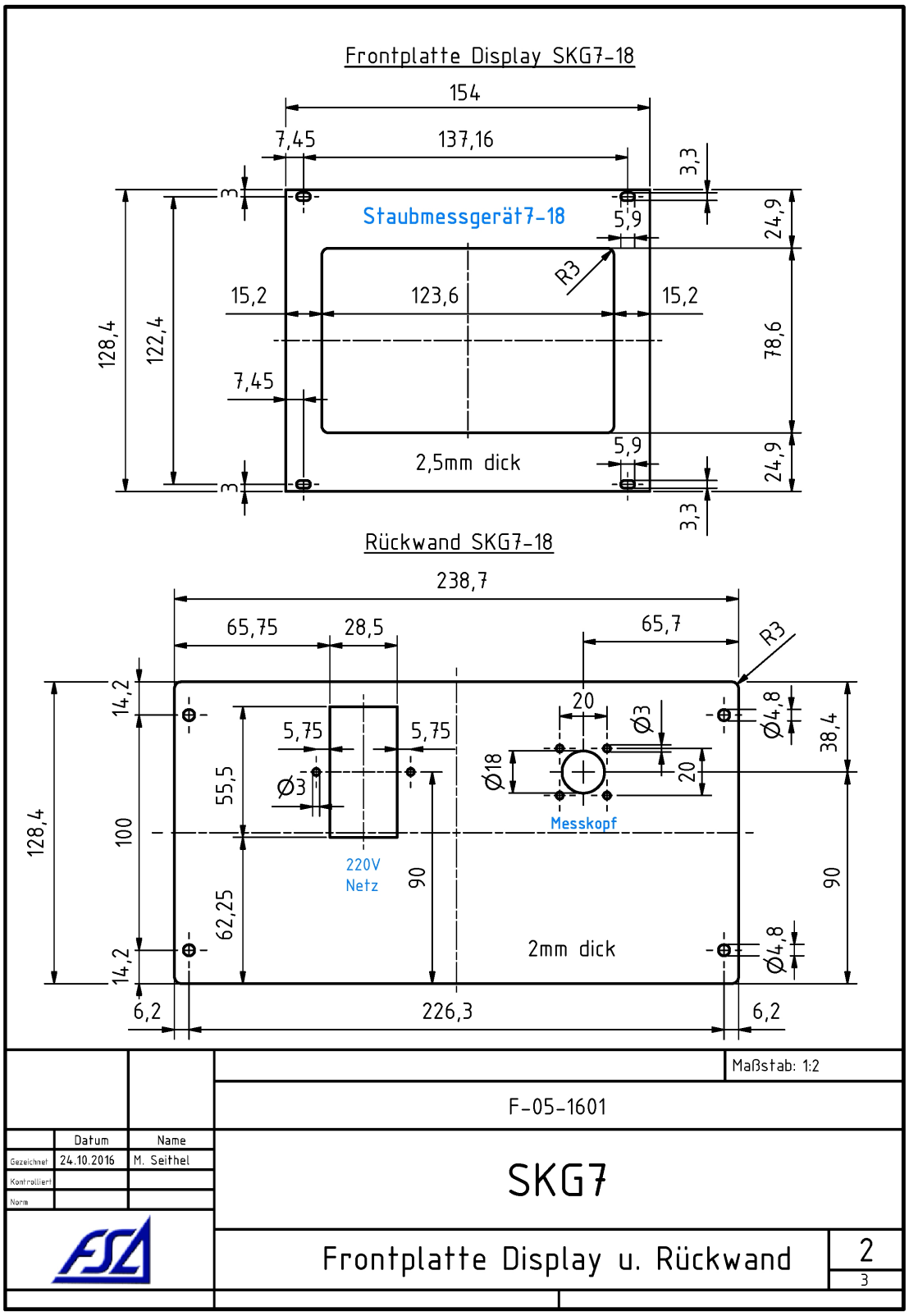

Abbildung B.2: CAD-Zeichnungen für die Display-Frontplatte und die Rückwand [12].

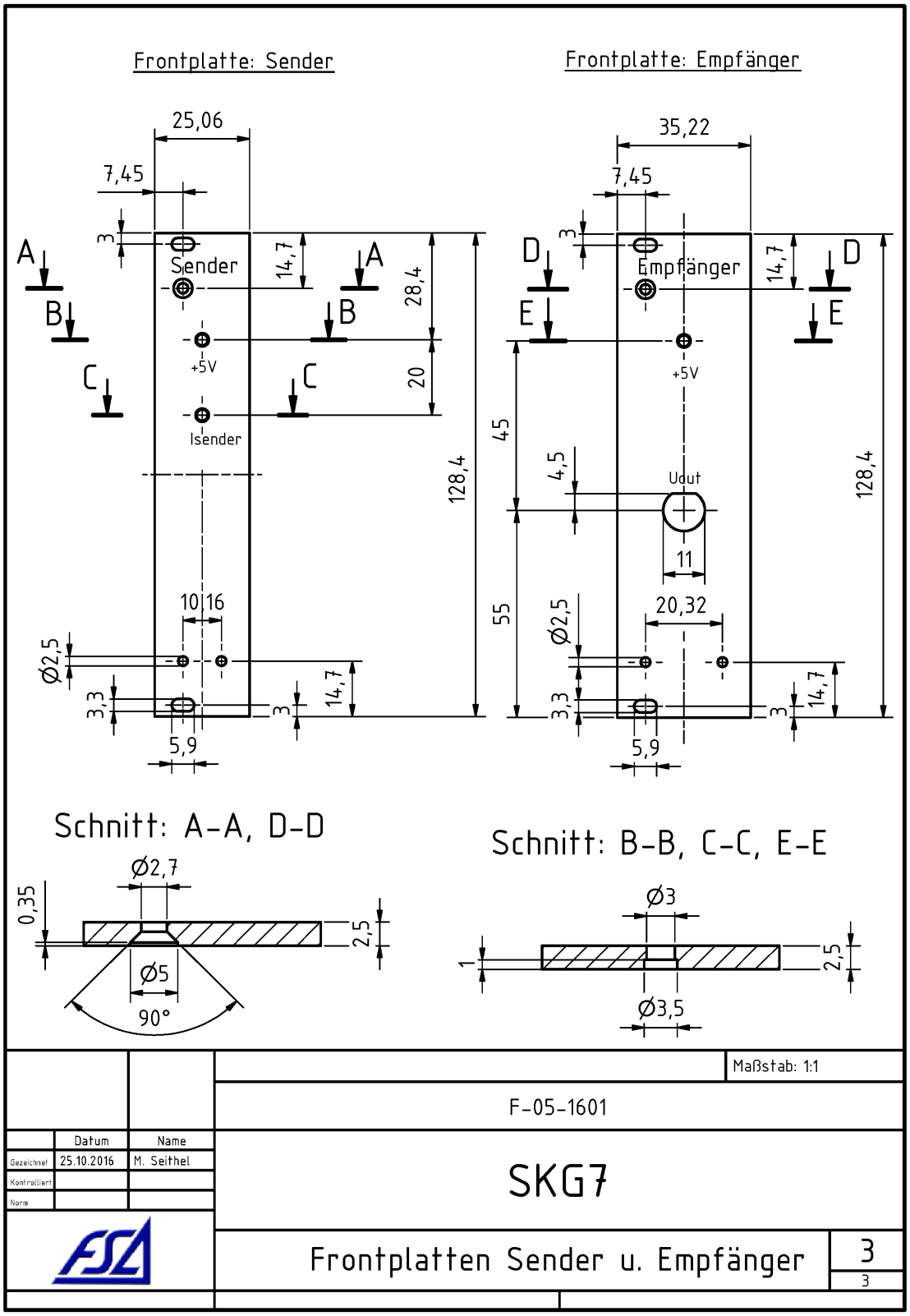

Abbildung B.3: CAD-Zeichnungen für die Frontplatten des Sender- und Empfängertreibers [12].

Anhang C

Programme

C.1 Code für den Mikrocontroller

```
/*
Programm fuer die Regeleinheit vom
Staubkonzentrationsmessgeraet SKG7
Datum: 15.10.2016
*/

/*
 * Einbinden header files fuer alle Treiber, die von
 * Atmel Software Framework (ASF) eingefuegt wurden.
 */

#include <asf.h>
#include <ioport.h>
#include <string.h>
#include <stdlib.h>
#include <conf_usart.h>
#include <math.h> // Mathematische Funktionen

#include "ft800.h" // Display Bibliothek einbinden

// USART fuer Sensor konfigurieren:
#define CONF_BOARD_ENABLE_USARTF0

#define USART_SERIAL_1                    &USARTF0
#define USART_SERIAL_1_BAUDRATE           9600
#define USART_SERIAL_1_CHAR_LENGTH        USART_CHSIZE_8BIT_gc
#define USART_SERIAL_1_PARITY             USART_PMODE_DISABLED_gc
#define USART_SERIAL_1_STOP_BIT           false

static uint8_t dli=0; // globale Variable, die nur innerhalb dieser Datei sichtbar und
gueltig ist
                                  // wird fuer die DL-Write-Commands verwendet

// Empfaenger-Strom-Offset-Variable:
static double IOffsetEmpfmA=0; // globale Variable, die nur innerhalb dieser Datei
sichtbar und gueltig ist


// SPI-Schnittstelle konfigurieren:
extern struct spi_device spi_device_conf; // Konfig. Stuktur fuer SPI

// SPIF - CS Pins einstellen (zwei Geraete kreieren):
struct spi_device spi_Sendertreiber = {
	.id = IOPORT_CREATE_PIN(PORTD, 0) // Sender-Karte: CS -> gpio0 Header J3
};
struct spi_device spi_Empfaengertreiber = {
	.id = IOPORT_CREATE_PIN(PORTD, 4) // Empfaenger-Karte: CS -> gpio4 Header J3
};


// Staub Struktur
struct STRC_Dust {
	char Name[20];
	double ExtKoeff;
};

// Einstellungen
```

```
struct STRC_Settings {
	struct STRC_Dust	Staub;
	double			Messabstand; // Messabstand  (Messeinheit)
};

// Liste der Staeube
static struct STRC_Dust StaubListe[10];
static int StaubListeCount;

// ------------------------------------ initFT800() ---------------------------------
----
// Initialisierung des Displays
uint8_t initFT800(void)
{
	uint8_t dev_id = 0;			// Variable fuer Display-ID
	// Reset the FT800:
	ioport_set_pin_low(PD_PIN);
	delay_ms(20);
	ioport_set_pin_high(PD_PIN);
	delay_ms(20);

	//WAKE
	HOST_CMD_ACTIVE();
	delay_ms(500);

	//Ext Clock
	HOST_CMD_WRITE(CMD_CLKEXT);		// CLK_EXT Command (0x44) senden

	//PLL (48M) Clock
	HOST_CMD_WRITE(CMD_CLK48M);		// CLK_48M Command (0x62) senden

	dev_id = HOST_MEM_RD8(REG_ID);		// Display-ID auslesen
	if(dev_id != 0x7C)			// ID muss 0x7C sein
	{
		return 0;
	}

	HOST_MEM_WR8(REG_GPIO, 0x00);			// REG_GPIO auf 0 setzen -> LCD
DISP Signal aus
	HOST_MEM_WR8(REG_PCLK, 0x00);			// Pixel Clock Output disable

	HOST_MEM_WR16(REG_HCYCLE, 548);		//  H_Cycle -> 548
	HOST_MEM_WR16(REG_HOFFSET, 43);		//  H_Offset -> 43
	HOST_MEM_WR16(REG_HSYNC0, 0);		//  H_SYNC_0 -> 0
	HOST_MEM_WR16(REG_HSYNC1, 41);		//  H_SYNC_1 -> 41
	HOST_MEM_WR16(REG_VCYCLE, 292);		//  V_Cycle -> 292
	HOST_MEM_WR16(REG_VOFFSET, 12);		//  V_OFFSET -> 12
	HOST_MEM_WR16(REG_VSYNC0, 0);		//  V_SYNC_0 -> 0
	HOST_MEM_WR16(REG_VSYNC1, 10);		//  V_SYNC_1 -> 10
	HOST_MEM_WR8(REG_SWIZZLE, 0);		//  SWIZZLE -> 0
	HOST_MEM_WR8(REG_PCLK_POL, 1);		//  PCLK_POL -> 1
	HOST_MEM_WR8(REG_CSPREAD, 1);		//  CSPREAD -> 1
	HOST_MEM_WR16(REG_HSIZE, 480);		//  H_SIZE -> 480
	HOST_MEM_WR16(REG_VSIZE, 272);		//  V_SIZE -> 272

	/* Konfiguration touch & audio */
	HOST_MEM_WR8(REG_TOUCH_MODE, 0x03);	// touch on: continous
	HOST_MEM_WR8(REG_TOUCH_ADC_MODE, 0x01);	// touch mode: differential (default)
	HOST_MEM_WR8(REG_TOUCH_OVERSAMPLE, 0x0F);	// touch oversampling: max
```

```
	HOST_MEM_WR16(REG_TOUCH_RZTHRESH, 1200); // touch resistance threshold (1200 -
typisch)
	HOST_MEM_WR8(REG_VOL_SOUND, 0xFF);	// the volume: maximum

	/*  display list  */
	HOST_MEM_WR32(RAM_DL+0, CLEAR_COLOR_RGB(0,0,0));  // Farbe setzen: Schwarz
	HOST_MEM_WR32(RAM_DL+4, CLEAR(1,1,1));            // Loeschen
	HOST_MEM_WR32(RAM_DL+8, DISPLAY());               // End Display List

	HOST_MEM_WR8(REG_DLSWAP, DLSWAP_FRAME);           // display list aktivieren fuer
naechsten Frame

	HOST_MEM_WR8(REG_GPIO_DIR, 0x80);                 // Disp GPIO Richtung setzen
	HOST_MEM_WR8(REG_GPIO, 0x80);                     // Enable Disp (wenn benutzt
wird)
	HOST_MEM_WR16(REG_PWM_HZ, 0x00FA);                // Hintergrundbeleuchtung PWM
frequenz
	HOST_MEM_WR8(REG_PWM_DUTY, 0x80);

	HOST_MEM_WR8(REG_PCLK, 0x05);                     // Jetzt ist das Display
eingeschaltet

	// ClearScreen (Displayinhalt löschen):
		cmd(CMD_DLSTART);
		cmd(CLEAR_COLOR_RGB(0,0,0));
		cmd(CLEAR(1,1,1));
		cmd(DISPLAY());
		cmd(CMD_SWAP);

	return 1; // wenn initialisierung erfolgt --> eins zurueckgeben
}

// ------------------------------- calibrateFT800() -----------------------------------
-----------------
// Touchdisplay kalibrieren
void calibrateFT800 (void)
{
	while(!cmd_ready);                              // warten bis Co-Processor bereit
ist
	// Touchfeld kalibrieren:
	//
	cmd(CMD_DLSTART);
	cmd(CLEAR_COLOR_RGB(0,0,0));
	cmd(CLEAR(1,1,1));
	cmd_text(240,50,27,OPT_CENTER,"Touchscreen Kalibrierung");
	cmd_text(240,80,26,OPT_CENTER,"Bitte die Punkte antasten");

	cmd_calibrate();           // Kalibrierungsfunktion aufrufen

	while(!cmd_ready);         // warten bis Co-Processor bereit ist

	// ClearScreen (Displayinhalt löschen):
	cmd(CMD_DLSTART);
	cmd(CLEAR_COLOR_RGB(0,0,0));
	cmd(CLEAR(1,1,1));
	cmd(DISPLAY());
	cmd(CMD_SWAP);

}
```

```
//----------------------- ClearScreen (Displayinhalt löschen): -------------------------
-----------------
void clrscr(void)
{
	cmd(CMD_DLSTART);
	cmd(CLEAR_COLOR_RGB(0,0,0));
	cmd(CLEAR(1,1,1));
	cmd(DISPLAY());
	cmd(CMD_SWAP);
}
// ------------------------------------------------------------------------------------
------------------

// ---------------------------------- set_strom (uint16) -------------------------------
-----------------
// stellt den Senderstrom ein. Nimmt den Wert 0...4095 ein, was dem Strom 0...20mA
enspricht
//
void set_strom(uint16_t Isoll_digital)
{

	uint8_t byte1=112; // 0111 0000
	uint8_t byte2=0;
	byte1=byte1|(Isoll_digital>>8);
	byte2=byte2|(Isoll_digital);
	uint8_t data_sender[2]={byte1,byte2}; //

	spi_select_device(&SPIF,&spi_Sendertreiber);
	spi_write_packet(&SPIF, data_sender,2);
	spi_deselect_device(&SPIF,&spi_Sendertreiber);
}

// ---------------------------------- int get_strom (void) ----------------------------
-----------------
// misst den Empfaengerstrom und gibt den digitalen Wert 0...4095 zurueck
//
int get_strom(void)
{
	uint8_t data_ADC_rd[2]={0,0,0}; //
	uint8_t data_ADC_byte0 =0;
	uint8_t data_ADC_byte1 =0;
	uint16_t dataADC=0;
	uint16_t dataADC_mittelwert=0;

	for (int i=0; i<1; i++) // max 16 mal!
	{
		spi_select_device(&SPIF,&spi_Empfaengertreiber);	// CS -> low
		spi_put(&SPIF, 6);
	// Den ersten Byte senden: 00000110 (single-ended Eingang, CH0): 0--0--0--0--0--
start-Bit(1)--SGL/Diff-Bit(1)--D2-Bit(0)
		while(!spi_is_tx_ok(&SPIF));                                        //
warten
		spi_put(&SPIF, 0);
	// Send Byte: 00000000 Den zweiten Byte senden: 00000110 (single-ended): D1-Bit(0)-
-D0-Bit(0)--0--0--0--0--0--0
		while(!spi_is_tx_ok(&SPIF));                                        //
warten
```

```
                spi_read_single(&SPIF, data_ADC_rd);                            // Daten lesen
(Byte 0)
                spi_put(&SPIF, 0);
        // Send Byte: 00000000
                while(!spi_is_tx_ok(&SPIF));                                            //
warten bis alles verschickt wird
                spi_read_single(&SPIF, (data_ADC_rd+1));              // Daten lesen (Byte
1)
                spi_deselect_device(&SPIF,&spi_Empfaengertreiber);    // CS -> high

                dataADC=(data_ADC_rd[0])&15; // Byte 1: 4 MSBits loeschen
                dataADC=dataADC<<8;
                dataADC=dataADC|data_ADC_rd[1];

                dataADC_mittelwert=dataADC_mittelwert+dataADC;
        }
        delay_s(1);
        dataADC_mittelwert=dataADC_mittelwert/1;
        return(dataADC_mittelwert);
}

// --------- Start-Menue: lcd_start_menue (Staubname, Konz, Temp, Feuchte, Isender, Iempf,
Ioffset) ----------
// gibt das Start-Menue aus
void lcd_start_menue(const char* StaubName, const char* Konztr, const char* TempC, const
char* LuftRH, const char* Isender, const char* Iempf, const char* Ioffset)
{
        while(!cmd_ready());
        char charArray1[30]; // string fuer den Isender
        char charArray2[30]; // string fuer Konzentration
        char charArray3[30]; // string fuer I_empfaenger
        char charArray4[30]; // string fuer I_offset
        char charArray5[30]; // string fuer Temperatur-Wert
        char charArray6[30]; // string fuer Luftfeuchtigkeit-Wert

        // Alle Elemente des charArrays auf null setzen:
        for (int i=0; i<30; i++)
        {
                charArray1[i]=0;
                charArray2[i]=0;
                charArray3[i]=0;
                charArray4[i]=0;
                charArray5[i]=0;
                charArray6[i]=0;
        }
        strcpy(charArray1,"I_Sender: ");         // "I_Sender: " - String in charArray
schreiben
        strcat(charArray1, Isender);
        strcat(charArray1," mA");

        strcpy(charArray2,"Konzentration: ");    // "Konzentration: " - String in charArray
schreiben
        strcat(charArray2, Konztr);                     // den Wert anhaengen
        strcat(charArray2," g/m3");                     // string " g/m3" anhaengen

        strcpy(charArray3,"I_Empfaenger: ");     // "I_Empfaenger: " - String in charArray
schreiben
        strcat(charArray3, Iempf);
        strcat(charArray3," mA");
```

```
      strcpy(charArray4,"I_offset: ");        // "I_offset: " - String in charArray
schreiben
      strcat(charArray4, Ioffset);                  // den Wert anhaengen
      strcat(charArray4," mA");                     // string " mA" anhaengen

      strcpy(charArray5,"T_Umgeb: ");               // "T_Umgeb: " - String in
charArray schreiben
      strcat(charArray5, TempC);                    // den Wert anhaengen
      strcat(charArray5," C");                      // string " C" anhaengen

      strcpy(charArray6,"Luftfeuchte: ");           // "Luftfeuchte: " - String in
charArray schreiben
      strcat(charArray6, LuftRH);                   // den Wert anhaengen
      strcat(charArray6," %");                      // string " %" anhaengen

    // Ausgabe:
      cmd(CMD_DLSTART);
      cmd(CLEAR_COLOR_RGB(0,0,0));
      cmd(CLEAR(1,1,1));
      cmd_text(80, 20, 28, 0, "Staub: ");
      cmd_text(150, 20, 28, 0, StaubName);    // Staubname
      cmd_text(80, 50, 28, 0, charArray2);    // Konzentartion
      cmd_text(80, 90, 26, 0, charArray5);    // T_Umgebung: xx.xx C
      cmd_text(80, 110, 26, 0, charArray6);   // Luftfeuchtigkeit: xx%
      cmd_text(80, 140, 26, 0, charArray1);   // I_Sender - Zeile ausgeben
      cmd_text(80, 160, 26, 0, charArray3);   // I_Empfaenger - Zeile
      cmd_text(250, 160, 26, 0, charArray4);  // I_Empfaenger_offset - Zeile
      cmd(COLOR_RGB(255,255,255));
      cmd_fgcolor(COL_LIGHT_BLUE);                  // die Farbe von Knopf definieren
      cmd(TAG(1));
      cmd_button(10, 210, 120, 50, 26, 0, "Nullabgleich"); // Nullabgleich-Taste, (tag=1)
      cmd(TAG(2));
      cmd_button(140, 210, 120, 50, 26, 0, "Einstellungen"); // Einstellungen-Taste
(tag=2)
      cmd(DISPLAY());
      cmd(CMD_SWAP);
      while(!cmd_ready());

}

// ------------------------------------ null_abgleich_ok(uint_8) ----------------------
---------------
// gibt die Meldung ueber erfolgreichen oder nicht erfolgreichen Nullabgleich
// wird von nullabgleich-Funktion aufgerufen
void null_abgleich_ok(uint8_t AbgleichOK)
{
      if (AbgleichOK==1) // wenn Abgleich ok ist - Meldung "alles OK"
      {
            cmd(CMD_DLSTART);
            cmd(CLEAR_COLOR_RGB(0,0,0));
            cmd(CLEAR(1,1,1));

            cmd_text(240, 70, 28, OPT_CENTER, "Der Nullabgleich wurde");
            cmd_text(240, 100, 28, OPT_CENTER, "erfolgreich abgeschlossen.");

            cmd(TAG(1)); // Taste: tag = 1
            cmd_button(190, 130, 100, 50, 26, 0, "OK");
            cmd(DISPLAY());
```

```
		cmd(CMD_SWAP);
		while (1)
		{
			uint32_t tag = HOST_MEM_RD32(REG_TOUCH_TAG);	// den Tag-Wert
auslesen
			if(tag==1) break;
	// wenn der Taste "Abbrechen" (tag 1) gedrueckt wurde - schleife beenden
		}
	}
	else if (AbgleichOK==0) // wenn Abgleich nicht ok ist - Meldung
	{
		cmd(CMD_DLSTART);
		cmd(CLEAR_COLOR_RGB(0,0,0));
		cmd(CLEAR(1,1,1));

		cmd_text(240, 70, 28, OPT_CENTER, "Der Nullabgleich");
		cmd_text(240, 100, 28, OPT_CENTER, "konnte nicht durchgefuehrt werden!");

		cmd(TAG(1)); // Taste tag = 1
		cmd_button(190, 130, 100, 50, 26, 0, "OK");
		cmd(DISPLAY());
		cmd(CMD_SWAP);
		while (1)
		{
			uint32_t tag = HOST_MEM_RD32(REG_TOUCH_TAG); // den Tag-Wert auslesen
			if(tag==1) break;  // wenn der Button "Abbrechen" (tag 1) gedrueckt
wurde - schleife beenden
		}
	}


}


// ------------------------------------ nullabgleich() ---------------------------------
------------------
// Mit der Funktion wird die Messeinheit auf Null abgeglichen
uint16_t nullabgleich(void)
{
	// Variablen definieren:
	uint16_t Senderstrom=0;
	int Empfstrom=0;
	double Empfstrom_mA=0;
	uint8_t Erfolg=2;
	int k=0;

	// Menue ausgeben:
	while(!cmd_ready());
	cmd(CMD_DLSTART);
	cmd(CLEAR_COLOR_RGB(0,0,0));
	cmd(CLEAR(1,1,1));
	cmd_text(240, 50, 28, OPT_CENTER, "Der Nullabgleich (grob) laeuft.");
	cmd_text(240, 75, 28, OPT_CENTER, "Bitte warten...");
	cmd_bgcolor(0x404040); // Backgroundcolour des Progressbars (dunkelgrau)
	cmd_progress(90, 120, 300, 12, OPT_FLAT, (0), 100); // progress bar
	cmd(TAG(1)); // Taste tag = 1
	cmd_fgcolor(0x404040); // Farbe der Taste: dunkelgrau
	cmd_button(190, 180, 100, 50, 26, 0, "abbrechnen"); // "abbrechen" - Taste
```

```
	cmd(DISPLAY());
	cmd(CMD_SWAP);
	while(!cmd_ready());

	// grobe Einstellung:
	for (int i=0; i<81; i++) // solange Empf-Strom kleiner als 19mA ist
	{
		// Senderstrom um 0,244mA erhoehen:
		Senderstrom=Senderstrom+50; // um 0,244 mA max bis 81*50=4050 -> 19,78 mA
		if (Senderstrom<=4095)
		{
			set_strom(Senderstrom);
		}
		delay_ms(100);

		// Empf-Strom messen:
		Empfstrom=get_strom();
		// den Empf-Strom in mA berechnen:
		Empfstrom_mA=Empfstrom*0.00748529-IOffsetEmpfmA;
		if ((Empfstrom_mA>18.00)) // Wenn Empfaengerstrom groesser 18mA
		{
			break;
		}
		if((Senderstrom>=2047)&&(Empfstrom<=267))		// wenn Senderstrom
groesser 10mA und Empf-Strom kleiner 2mA ist
		{
			break;
		}
		if((Senderstrom==4050)&&(Empfstrom_mA<19.00))	// wenn Senderstrom 19,78 mA
ist und Empf-Strom kleiner 19mA ist
		{
			break;
		}

		while(!cmd_ready());
		cmd(CMD_DLSTART);
		cmd(CLEAR_COLOR_RGB(0,0,0));
		cmd(CLEAR(1,1,1));
		//cmd(CMD_COLDSTART); // set co-processor engine to default values
		cmd_text(240, 50, 28, OPT_CENTER, "Der Nullabgleich (grob) laeuft.");
		cmd_text(240, 75, 28, OPT_CENTER, "Bitte warten...");
		cmd_bgcolor(0x404040); // Backgroundcolour des Progressbars (dunkelgrau)
		cmd_progress(90, 120, 300, 12, OPT_FLAT, Empfstrom, 2730); // progress bar
100%=2730 entspricht dem Empfaengerstrom von 20mA

		cmd(TAG(1)); // button tag = 1
		cmd_fgcolor(0x404040); // Farbe des Buttons: dunkelgrau
		cmd_button(190, 180, 100, 50, 26, 0, "abbrechnen");

		cmd(DISPLAY());
		cmd(CMD_SWAP);
		while(!cmd_ready());

		uint32_t tag = HOST_MEM_RD32(REG_TOUCH_TAG); // den Tag-Wert auslesen
		if(tag==1) break;  // wenn der Button "Abbrechen" (tag 1) gedrueckt wurde -
schleife beenden
	}

	// feine Einstellung:
```

```
	for (int i=0; i<4000; i++)
	{
		// Senderstrom erhoehen:
		Senderstrom=Senderstrom+1; // um 0,004884mA
		// Senderstrom einstellen:
		if (Senderstrom<=4095)
		{
			set_strom(Senderstrom);
		}

		delay_ms(100);
		Empfstrom=get_strom();
		// den Empf-Strom in mA berechnen:
		Empfstrom_mA=Empfstrom*0.00748529-IOffsetEmpfmA;
		if ((Empfstrom_mA>=19.95)&&(Empfstrom_mA<=20.00)) // Wenn Empfaengerstrom
groesser oder gleich 19,95mA und kleiner als 20,00mA ist
		{
			Erfolg=1;
			break;
		}
		else if((Senderstrom>=2047)&&(Empfstrom<=267)) // wenn Senderstrom groesser
10mA und Empf-Strom kleiner 2mA ist
		{
			Erfolg=0;
			break;
		}
		if((Senderstrom==4050)&&(Empfstrom_mA<19.00)) // wenn Senderstrom 19,78 mA
ist und Empf-Strom kleiner 19mA ist (Linsen verschmutzt, Kabelbruch usw)
		{
			Erfolg=0;
			break;
		}

		while(!cmd_ready());
		cmd(CMD_DLSTART);
		cmd(CLEAR_COLOR_RGB(0,0,0));
		cmd(CLEAR(1,1,1));
		cmd_text(240, 50, 28, OPT_CENTER, "Der Nullabgleich (fein) laeuft.");
		cmd_text(240, 75, 28, OPT_CENTER, "Bitte warten...");
		cmd_bgcolor(0x404040); // Backgroundcolour des Progressbars (dunkelgrau)
		cmd_progress(90, 120, 300, 12, OPT_FLAT, Empfstrom, 2730); // progress bar

		cmd(TAG(1)); // button tag = 1
		cmd_fgcolor(0x404040); // Farbe des Buttons: dunkelgrau
		cmd_button(190, 180, 100, 50, 26, 0, "abbrechnen");

		cmd(DISPLAY());
		cmd(CMD_SWAP);
		while(!cmd_ready());
		uint32_t tag = HOST_MEM_RD32(REG_TOUCH_TAG); // den Tag-Wert auslesen
		if(tag==1) break;  // wenn der Button "Abbrechen" (tag 1) gedrueckt wurde -
schleife beenden
	}

	if (Erfolg==0)
	{
		null_abgleich_ok(0); // "Nullabgleich nicht erfolgreich" - Meldung
		Senderstrom=0;
	}
```

```
	else if (Erfolg==1)
	{
		null_abgleich_ok(1); // "Nullabgleich erfolgreich" - Meldung
	}

	return Senderstrom;
} // ende nullabgleich()

// ---------------------------------- lcd Edit Functions (char* und double) ------------
---------------
//
double lcd_Edit_Double(double dblVal,char* sInfo,int len, int frac)
{
	bool bShowMenu = true ;
	char sEditDBL[10];
	dtostrf(dblVal,len,frac,sEditDBL);
	char charCount = strlen(sEditDBL);

	while(bShowMenu)
	{
		while(!cmd_ready());
		cmd(CMD_DLSTART);
		cmd(CLEAR_COLOR_RGB(0,0,0));
		cmd(CLEAR(1,1,1));

		cmd_text(240, 20, 28, OPT_CENTER, sInfo);

		//Wert anzeigen

		cmd_text(240, 50, 28, OPT_CENTER, sEditDBL);

		// Tastatur
		cmd_keys(150, 99, 150, 40, 27, 0, "789");
		cmd_keys(150, 142, 150, 40, 27, 0, "456");
		cmd_keys(150, 185, 150, 40, 27, 0, "123");
		cmd_keys(150, 228, 150, 40, 27, 0, "0,");

		cmd(TAG(1));
		cmd_button(15, 99, 60, 40, 27, 0, "<"); //  < - taste

		cmd(TAG(2));
		cmd_button(405, 99, 60, 40, 26, 0, "OK"); // Button OK

		cmd(DISPLAY());
		cmd(CMD_SWAP);
		while(!cmd_ready());
		delay_ms(300);
		while(true)
		{
			uint32_t tag = HOST_MEM_RD32(REG_TOUCH_TAG);

			if(tag == 1) {//                                                    Löschen
				charCount--;
				if (charCount == 255) charCount=0;
				sEditDBL[charCount]='\0';
				break;
			}
			if(tag == 2) {//                                                    OK
				bShowMenu = false;
```

```
                        break;
                }
                if ((tag == ',')){//                                    Komma
                        if (charCount<8){
                                sEditDBL[charCount] = '.';
                                charCount++;
                                sEditDBL[charCount] = '\0';
                        }
                        break;
                }
                if ((tag >= '0') && (tag <='9')){ //           0 1 2 3 4 5 6 7 8 9
                        if (charCount<8){
                                char c = tag;
                                sEditDBL[charCount] = c;
                                charCount++;
                                sEditDBL[charCount] = '\0';
                        }
                        break;
                }
        }
    }
    dblVal = atof(sEditDBL);
    return dblVal;
}
char* lcd_Edit_String(char* sVal, char* sInfo){
    bool bShowMenu = true ;
    bool bCAPS = true;
    bool bNumbers =false;

    const int MAXSTRLEN = 20;

    char sEditSTR[MAXSTRLEN];

    strcpy(sEditSTR,sVal);

    char charCount = strlen(sEditSTR);

    while(bShowMenu)
    {
        while(!cmd_ready());
        cmd(CMD_DLSTART);
        cmd(CLEAR_COLOR_RGB(0,0,0));
        cmd(CLEAR(1,1,1));

        cmd_text(240, 20, 28, OPT_CENTER, sInfo);

        //Wert anzeigen

        cmd_text(240, 50, 28, OPT_CENTER, sEditSTR);

        if (bNumbers){
            cmd_keys(150, 99, 150, 40, 27, 0, "789");
            cmd_keys(150, 142, 150, 40, 27, 0, "456");
            cmd_keys(150, 185, 150, 40, 27, 0, "123");
            cmd_keys(150, 228, 150, 40, 27, 0, "0,#");
        }
        else{
            if (bCAPS){
                cmd_keys(15, 142, 450, 40, 27, 0, "QWERTZUIOP");
```

```
                cmd_keys(15, 185, 450, 40, 27, 0, "ASDFGHJKL");
                cmd_keys(15, 228, 450, 40, 27, 0, "^YXCVBNM#");
        }
        if(!bCAPS){
                cmd_keys(15, 142, 450, 40, 27, 0, "qwertzuiop");
                cmd_keys(15, 185, 450, 40, 27, 0, "asdfghjkl");
                cmd_keys(15, 228, 450, 40, 27, 0, "^yxcvbnm#");
        }
}

cmd(TAG(1));
cmd_button(15, 99, 60, 40, 27, 0, "<"); //  < - taste

cmd(TAG(2));
cmd_button(405, 99, 60, 40, 26, 0, "OK"); // Button OK

cmd(DISPLAY());
cmd(CMD_SWAP);
while(!cmd_ready());
delay_ms(300);
while(true)
{
        uint32_t tag = HOST_MEM_RD32(REG_TOUCH_TAG);

        if(tag == 1) {//                                                Löschen
                charCount--;
                if (charCount == 255) charCount=0;
                sEditSTR[charCount]='\0';
                break;
        }
        if(tag == 2) {//                                                OK
                bShowMenu = false;
                break;
        }
        if (tag == '#'){// # Zahlen Buchstaben
                bNumbers = !bNumbers;
                break;
        }
        if (tag == '^'){// ^ GroßKlein Schreibung
                bCAPS = !bCAPS;
                break;
        }
        if ((tag >= '0') && (tag <='9')){ //           0 1 2 3 4 5 6 7 8 9
                if (charCount < MAXSTRLEN){
                        sEditSTR[charCount] = (tag - '0');
                        charCount++;
                        sEditSTR[charCount] = '\0';
                }
                break;
        }
        if (tag > 47 && tag < 123) //von A bis z'
        {
                if (charCount < MAXSTRLEN){
                        char c = tag;
                        sEditSTR[charCount] = c;
                        charCount++;
                        sEditSTR[charCount]='\0';
                        //nur der Erste Buchstabe Groß
                        if (charCount==0) bCAPS=false;
```

```
                                    break;
                                }
                            }
                        }
                    }
                    return sEditSTR;
}

//------------------------------ lcd_Select_Staub(struct) ------------------------------
------------------
//  wählt den Staub aus einer Liste
struct STRC_Dust lcd_Select_Staub(struct STRC_Dust orgDust)
{
                    delay_ms(300);
                    bool bShowMenu = true ;
                    int i,iOffset=0;
                    struct STRC_Dust ret;

                    while(bShowMenu)
                    {
                            while(!cmd_ready());
                            cmd(CMD_DLSTART);
                            cmd(CLEAR_COLOR_RGB(0,0,0));
                            cmd(CLEAR(1,1,1));
                            cmd_text(240, 20, 28, OPT_CENTER, "Staubliste");

                            //fülle die 5 Buttons mit den Stäuben ab iOffset
                            for (i=0; i<5; i++ ){       // TAG 1 2 3 4 5 6 7 8 9 10 11
                                    cmd(TAG(iOffset+i+1));

                                    if (iOffset+i < StaubListeCount){
                                       cmd_fgcolor(COL_LIGHT_BLUE);
                                       cmd_button(15, 50 + (i)*40, 340, 30, 27, 0,
StaubListe[iOffset+i].Name); // Staub
                                        }
                                    else if (iOffset+i == StaubListeCount){
                                        cmd_fgcolor(0x404040);
                                        cmd_button(15, 50 + (i)*40, 340, 30, 27, 0, "neuer Staub");
// Staub
                                        }
                            }
                            //---------------------------------------------      Hoch Button Tag
21 --------------------
                            cmd(TAG(21));
                    // TAG 21
                            cmd_fgcolor(COL_LIGHT_BLUE);
                            //
                            cmd_button(360, 50, 50, 50, 26, 0, "<<");
                            //---------------------------------------------      Runter Button
Tag 22 --------------------
                            cmd(TAG(22));
                    // TAG 22
                            cmd_fgcolor(COL_LIGHT_BLUE);
                            //
                            cmd_button(360, 105, 50, 50, 26, 0, ">>");
                            //---------------------------------------------      Zurück Button
Tag 23 --------------------
                            cmd(TAG(23));
                    // TAG 23
```

```
                cmd_fgcolor(0x404040);
            //dunkelgrau
                cmd_button(360, 190, 120, 50, 26, 0, "zurueck");

                cmd(DISPLAY());
                cmd(CMD_SWAP);

                // ---------------------------- waret auf CoProzessor --------------
---------------------

                while(!cmd_ready());

                // ------------------------------------------------   Tastenabfrage
Loop
                while(1) //
                {
                        uint32_t tag = HOST_MEM_RD32(REG_TOUCH_TAG);
                        if (tag == 23) {                                //zurück
                                bShowMenu = false;
                                return orgDust;
                        }
                        if (tag <= 11 && tag >= 1){             //es wurde ein Knopf
gewählt

                                if (tag == StaubListeCount+1){

                                        struct STRC_Dust NewDust;

                                        strcpy(NewDust.Name ,
lcd_Edit_String("","Staubname eingeben"));
                                        delay_ms(300);
                                        NewDust.ExtKoeff = lcd_Edit_Double(0,"Epsilon in
m2/g",7,5);
                                        StaubListe[StaubListeCount] = NewDust;
                                        StaubListeCount++;
                                        delay_ms(300);
                                        ret = NewDust;
                                        return ret;
                                }
                                ret = StaubListe[tag-1];
                                return ret;
                        }
                        if (tag == 21) {                                // nach oben
                                if (iOffset > 0 ) iOffset--;
                                break;
                        }
                        if (tag == 22) {                                // nach unten
                                if(iOffset < (11-5)) iOffset++;
                                break;
                        }
                }
        }
}// end lcd_Select_Staub


// -------------------------------- lcd_Edit_Messabstand(double) ----------------------
-----------------------
// Messabstand aendern
double lcd_Edit_Messabstand(double mab)
```

```
{
	return lcd_Edit_Double(mab*1000,"Messabstand in mm",5,1)/1000;
	delay_ms(300);
}// end lcd_Edit_Messabstand
//-------------------------------------------------------------------------------------
----------------


// ------------------------------ lcd_EditSettings(struct) ---------------------------
------------------------
// Menue "Einstellungen"
struct STRC_Settings lcd_EditSettings(struct STRC_Settings Setting)
{
	delay_ms(300);
	bool bShowMenu = true ;

	while(bShowMenu)
	{
		while(!cmd_ready());
		cmd(CMD_DLSTART);
		cmd(CLEAR_COLOR_RGB(0,0,0));
		cmd(CLEAR(1,1,1));

		cmd_text(240, 20, 28, OPT_CENTER, "Einstellungen");
		cmd_fgcolor(COL_LIGHT_BLUE);

		// Button Staub Tag 1 ------------------------------------------------
		cmd(TAG(1));
	// TAG 1
		cmd_button(30, 60, 160, 40, 26, 0, "Staub");		// Staub
		//
		Name anzeigen
		cmd_text(200,60,22,0,Setting.Staub.Name);
		//
		ExtKoeff anzeigen
		char sExtKoeffInfo[15];
		char sExtKoeff[10];
		dtostrf(Setting.Staub.ExtKoeff,8,5,sExtKoeff);
		strcpy(sExtKoeffInfo,"eps = ");
		strcat(sExtKoeffInfo,sExtKoeff);
		strcat(sExtKoeffInfo,"m2/g");
		cmd_text(200,80,22,0,sExtKoeffInfo);

		// ---------------------------------------- Button Messabstand Tag 2 ----
------------------------------------------
		cmd_fgcolor(COL_LIGHT_BLUE);
		cmd(TAG(2));
	// TAG 2
		cmd_button(30, 120, 160, 40, 26, 0, "Messabstand"); // Messabstand

		//
		Wert anzeigen
		char sMABInfo[15];
		char sMessAb[10];//Wert als char

		strcpy(sMABInfo,"l = ");
		dtostrf(Setting.Messabstand * 1000,6,1,sMessAb);
		strcat(sMABInfo,sMessAb);
		strcat(sMABInfo,"mm");
```

```
            cmd_text(200,130,22,0, sMABInfo);

            //-----------------------------------------      Zurück Button Tag 3 --
-----------------
            cmd(TAG(3));
      // TAG 3
            cmd_fgcolor(0x404040);
      //dunkelgrau
            cmd_button(330, 190, 120, 50, 26, 0, "zurueck");
            cmd(DISPLAY());
            cmd(CMD_SWAP);
            // ---------------------------- waret auf CoProzessor ---------------------
--------------

            while(!cmd_ready());

            // ------------------------------------------------   Tastenabfrage Loop
            while(1) // teste auf
            {
                  uint32_t tag = HOST_MEM_RD32(REG_TOUCH_TAG);

                  if (tag == 3) {//zurück
                        bShowMenu = false;
                        break;
                  }
                  if (tag == 2) {//Messabstand editieren
                        Setting.Messabstand =
lcd_Edit_Messabstand(Setting.Messabstand);
                        break;
                  }
                  if (tag == 1) {//Staub waehlen
                        Setting.Staub = lcd_Select_Staub(Setting.Staub);
                        break;
                  }
            }
      }
      return Setting;
}
// ------------------------------------------------------------------------------------
----------------

// ----------------------------- spi_init_pins(void) ----------------------------------
----------------
// Pins fuer SPI initialisieren
void spi_init_pins(void)
{
      // Initialize the pins used by the SPI interface SPIC (Display):
      ioport_configure_port_pin(&PORTC, PIN4_bm, IOPORT_INIT_HIGH | IOPORT_DIR_OUTPUT);
// CS
      ioport_configure_port_pin(&PORTC, PIN5_bm, IOPORT_INIT_HIGH | IOPORT_DIR_OUTPUT);
// MOSI
      ioport_configure_port_pin(&PORTC, PIN6_bm, IOPORT_DIR_INPUT);
            // MISO
      ioport_configure_port_pin(&PORTC, PIN7_bm, IOPORT_INIT_HIGH | IOPORT_DIR_OUTPUT);
// SCK

      // Initialize the pins used by the SPI interface SPIF (Sender-Karte, Empfaenger-
Karte):
```

```
	ioport_configure_port_pin(&PORTD, PIN0_bm, IOPORT_INIT_HIGH | IOPORT_DIR_OUTPUT);
	// CS Pin DAC (Sender-Karte) -> Ausgang
	ioport_configure_port_pin(&PORTD, PIN4_bm, IOPORT_INIT_HIGH | IOPORT_DIR_OUTPUT);
	// CS Pin ADC (Empfaenger-Karte) -> Ausgang

	ioport_configure_port_pin(&PORTF, PIN4_bm, IOPORT_PULL_UP | IOPORT_DIR_INPUT);
	// Enable pull-up on own chip select (SS)
	ioport_configure_port_pin(&PORTF, PIN5_bm, IOPORT_INIT_HIGH | IOPORT_DIR_OUTPUT);
	// MOSI -> Ausgang
	ioport_configure_port_pin(&PORTF, PIN6_bm, IOPORT_DIR_INPUT);
			// MISO -> Eingang
	ioport_configure_port_pin(&PORTF, PIN7_bm, IOPORT_INIT_HIGH | IOPORT_DIR_OUTPUT);
	// SCK -> Ausgang
}


//
================================================================================================
=============
// ======================================== M A I N
=====================================================
//
================================================================================================
=============

int main (void)
{

	/* Insert system clock initialization code here (sysclk_init()). */
	sysclk_init();
	board_init();
	delay_ms(10);
	ioport_init();
	/* Insert application code here, after the board has been initialized. */

	int Iempf=0;                    // Variable, die den Wert des Empfaenger-Stroms
speichert
	double Iempf_mA=0;              // Variable, die den Wert des Empfaenger-Stroms
in mA speichert (I)
	double I0empf_mA=0.01;          // Variable, die den Wert des Empfaenger-Stroms
in mA speichert (I0)

	uint16_t Isender=0;             // Variable, die den Wert des Senderstroms
speichert (16 Bit)
	double Isender_mA=0;       // Variable, die den Wert des Sender-Stroms in mA
speichert

	char StaubName[20] = "Maisstaerke";          // StaubName
	double Konztr=0;                                  // Konzentrationsvariable
	double ExtKoeff=0.13587;                     // Extinktionskoeffizient
	double Messabstand=0.0372;                   // Messabstand in m

	int i_mess=0; // laufvariable

	// Wetterdaten:
	uint8_t temperatur_string[]="xx.xx"; // Anfangswert
	uint8_t feuchte_string[]="xx.xx"; // Anfangswert
	uint8_t received_byte;
```

```
	Bool kk=0;
	uint8_t tx_buf[] = "\n\rHello AVR world ! : ";
	uint8_t tx_length = 22;

	//fuelle die StaubListe mit 5 Staeuben
	strcpy(StaubListe[0].Name, "Maisstaerke1");
	StaubListe[0].ExtKoeff = 0.13587;
	strcpy(StaubListe[1].Name , "Weizenmehl");
	StaubListe[1].ExtKoeff = 0.03125;
	strcpy(StaubListe[2].Name , "Puderzucker");
	StaubListe[2].ExtKoeff = 0.25489;
	strcpy(StaubListe[3].Name , "Zuckerstaub");
	StaubListe[3].ExtKoeff = 0.04839;
	strcpy(StaubListe[4].Name , "Kakaopulver");
	StaubListe[4].ExtKoeff = 0.03753;
	StaubListeCount = 5;

	// DeviceSetting ist die aktuelle Einstellung des Geraets
	struct STRC_Settings DeviceSetting;
	//stelle das DeviceSetting ein
	strcpy(DeviceSetting.Staub.Name , "Maisstaerke");
	DeviceSetting.Staub = StaubListe[0];
	DeviceSetting.Messabstand = Messabstand;

	char charArrayKonz[10];			// String Konzentration
	char charArrayIe[6];		// String fuer den Empfaengerstrom-Wert
	char charArrayIeoffs[6];	// String fuer den Empfaengerstrom-Offset-Wert
	char charArrayIs[6];		// String fuer den Senderstrom-Wert
	char charArrayTemp[6];			// String fuer die Umgeb-Temperatur
	char charArrayFeuchte[6];	// String fuer die Feuchte

	for (int i=0; i<6; i++)			// die Elemente der Strings auf null setzten
	{
		charArrayIe[i]=0;
		charArrayIs[i]=0;
		charArrayIeoffs[6]=0;
		charArrayTemp[6]=0;
		charArrayFeuchte[6]=0;
	}
	for (int i=0; i<10; i++)	// die Elemente der Strings auf null setzten
	{
		charArrayKonz[i]=0;
	}

	spi_init_pins(); // Initialisierung der Pins fuer SPIC (Displaysteuerung) und SPIF
(Steuerung der Sender- und Empfaenger-Karten)

	// SPI-Schnittstelle fuer Displaysteuerung initialisieren:
	spi_master_init(&SPIC);
				// init SPI API (Display)
	spi_master_setup_device(&SPIC, &spi_device_conf, SPI_MODE_0, 1000000, 0);  // SPIC
- Parameter einstellen (Display-Steuerung)
	spi_enable(&SPIC);
				// Enable SPI-C

	spi_master_init(&SPIF);
	spi_master_setup_device(&SPIF, &spi_Sendertreiber, SPI_MODE_0, 1000000, 0);
	// DAC - Senderkarte
```

```
	spi_master_setup_device(&SPIF, &spi_Empfaengertreiber, SPI_MODE_0, 100000, 0);
	// ADC - Empfaengerkarte
	spi_enable(&SPIF);
                                    // Enable SPI-F

	// USARTF0 initialisieren: (ueber USARTF0 -> Kommunikatoin mit dem Sensor)
	// USARTF0 Optionen:
		static usart_rs232_options_t USART_SERIAL_1_OPTIONS = {
			.baudrate = USART_SERIAL_1_BAUDRATE,
			.charlength = USART_SERIAL_1_CHAR_LENGTH,
			.paritytype = USART_SERIAL_1_PARITY,
			.stopbits = USART_SERIAL_1_STOP_BIT
		};

	// Initialize usart driver in RS232 mode
		usart_init_rs232(USART_SERIAL_1, &USART_SERIAL_1_OPTIONS);

	// Senderstrom 0mA einstellen:
	set_strom(Isender);
	delay_s(1);

	// Display starten:
	while(!initFT800()); // Display initialisieren
	//clrscr(); // Clear Screen

	// Touch-Funktion kalibrieren:
	calibrateFT800();

/* Test StromSet
while (1)
{
	set_strom(3069);	// Senderstrom einstellen 15mA
	delay_s(1);
	while(1);
}
*/
	// Empfaenger-Offset messen:
	Iempf=get_strom();                                                    // digitalen
wert aus ADC auslesen (integer-Wert)
	IOffsetEmpfmA=Iempf*0.00748529;                                 // den Stromwert in mA
berechnen
	dtostrf(IOffsetEmpfmA, 5, 2, charArrayIeoffs); // double to string konvertieren (2
Stellen nach Komma)


// ----------------------------------- Endlosschleife ------------------------
while(1)
{

	//lese die aktuellen Einstellungen ein
	strcpy(StaubName , DeviceSetting.Staub.Name);
	ExtKoeff = DeviceSetting.Staub.ExtKoeff;
	Messabstand = DeviceSetting.Messabstand;

	// die Elemente der Strings auf null setzten
	for (int i=0; i<6; i++)
	{
		charArrayIe[i]=0;
		charArrayIs[i]=0;
```

```
	}
	for (int i=0; i<10; i++)
	{
		charArrayKonz[i]=0;
	}


	i_mess++;		// Laufvar. erhoehen
	if (i_mess==10) // Empfaengerstrom jedes 10. Durchlauf der Schleife messen,
Senderstrom einstellen
	{
		i_mess=0;	// Laufvariable reset

		Iempf=get_strom();			// digitalen wert aus ADC auslesen
(integer-Wert)
		Iempf_mA=Iempf*0.00748529;	// den Stromwert in mA berechnen

		Isender_mA=Isender*0.004884;	// den Senderstromwert in mA berechnen
		if (Isender<=4095)
		{
		set_strom(Isender);			// Senderstrom einstellen
		}

		// Feuchte und Temperatur messen (Sensor abfragen):
		while (!kk)
			{
				received_byte = usart_getchar(USART_SERIAL_1);
				if (received_byte=='A')		// Char 'A' -> Anfang der
Messdaten
				{
					for (int i=0; i<12; i++) // 12 Chars auslesen
Txx.xxFxx.xx
					{
						received_byte = usart_getchar(USART_SERIAL_1);
						tx_buf[i]=received_byte;
					}
					kk=1;
				}

			} // end while(!kk)
		kk=0; // kk reset

		// Die ausgelesenen Werte kopieren:
		for (int i=0; i<5; i++)
		{
			temperatur_string[i]=tx_buf[i+1];
			feuchte_string[i]=tx_buf[i+7];
		}
	} // End if (i_mess==10)


	//Konzentration berechnen:
	Konztr=0;				// Wert auf Null setzten
	if (Iempf_mA<=I0empf_mA)   // IOempf_mA -> der I-Wert nach dem Nullabgleich ->
Intensitaet Io (Der gemessene I darf nicht groesser als Io sein)
	{
		Konztr=-1*log((Iempf_mA-IOffsetEmpfmA)/(I0empf_mA-IOffsetEmpfmA))/(ExtKoeff
* 2 * Messabstand); // Konzentration berechnen. (natural logarithm ln)
```

```
			if (Konztr<=0)                    // wenn der Wert negativ ist
			{
				Konztr=0;
			}
			else if (Konztr>50000)     // wenn der Wert zu hoch ist
			{
				Konztr=8888.888;
			}
		}

		dtostrf(Konztr, 8, 3, charArrayKonz); // double to string konvertieren (3 Stellen
nach Komma)


		//  ---------- Start-Menue aktualisieren: ---------------
		lcd_start_menue(StaubName, charArrayKonz, temperatur_string, feuchte_string,
charArrayIs, charArrayIe, charArrayIeoffs);

		uint32_t tag = HOST_MEM_RD32(REG_TOUCH_TAG); // tag-register auslesen - welche
Taste wurde gedrueckt?
		if(tag == 1) // Taste "Nullabgleich" wurde gedrueckt
		{
			Isender=nullabgleich();           // funktion "nullabgleich" aufrufen
			Iempf=get_strom();                // Empfaenger-Strom messen (digital)
			I0empf_mA=Iempf*0.00748529; // den Empfaengerstromwert in mA berechnen
(Intensitaet I0 nach dem Nullabgleich)
		}
		else if (tag==2) // Taste "Einstellungen" wurde gedrueckt
		{
			DeviceSetting = lcd_EditSettings(DeviceSetting); // Menue "Einstellungen"
aufrufen
		}


} // end while(1)

//ioport_toggle_pin_level(OSCI_PIN);     // trigger Oscilloscope (extern trigger input)
wird fuer Tests verwendet

}
```

C.2 Bibliotheken und Code für das Touchdisplay

```
// ft800 bibliothek

#ifndef _FT800_H_
#define _FT800_H_

/* FT800 Power Modes */
#define CMD_ACTIVE  0x00
#define CMD_STANDBY 0x41
#define CMD_SLEEP   0x42
#define CMD_PWRDOWN 0x50

/* FT800 Clock Switching */
#define CMD_CLKINT 0x48
#define CMD_CLKEXT 0x44
#define CMD_CLK48M 0x62
#define CMD_CLK36M 0x61

/* MISC */
#define CMD_CORERST 0x68

// Farben:
#define COL_LIGHT_BLUE 0x003870 // light blue

/* Definitions used for FT800 co processor command buffer */
#define FT_DL_SIZE           (8*1024)  //8KB Display List buffer size
#define FT_CMD_FIFO_SIZE     (4*1024)  //4KB coprocessor Fifo size
#define FT_CMD_SIZE          (4)       //4 byte per coprocessor command of EVE

#define FT800_VERSION "1.9.0"
#define ADC_DIFFERENTIAL     1UL
#define ADC_SINGLE_ENDED     0UL
#define ADPCM_SAMPLES        2UL
#define ALWAYS               7UL
#define ARGB1555             0UL
#define ARGB2                5UL
#define ARGB4                6UL
#define BARGRAPH             11UL
#define BILINEAR             1UL
#define BITMAPS              1UL
#define BORDER               0UL

#define CMDBUF_SIZE          4096UL
#define CMD_APPEND           4294967070UL
#define CMD_BGCOLOR          4294967049UL
#define CMD_BITMAP_TRANSFORM 4294967073UL
#define CMD_BUTTON           4294967053UL
#define CMD_CALIBRATE        4294967061UL
#define CMD_CLOCK            4294967060UL
#define CMD_COLDSTART        4294967090UL
#define CMD_CRC              4294967043UL
#define CMD_DIAL             4294967085UL
#define CMD_DLSTART          4294967040UL
#define CMD_EXECUTE          4294967047UL
#define CMD_FGCOLOR          4294967050UL
#define CMD_GAUGE            4294967059UL
#define CMD_GETMATRIX        4294967091UL
#define CMD_GETPOINT         4294967048UL
```

```
#define CMD_GETPROPS          4294967077UL
#define CMD_GETPTR            4294967075UL
#define CMD_GRADCOLOR         4294967092UL
#define CMD_GRADIENT          4294967051UL
#define CMD_HAMMERAUX         4294967044UL
#define CMD_IDCT              4294967046UL
#define CMD_INFLATE           4294967074UL
#define CMD_INTERRUPT         4294967042UL
#define CMD_KEYS              4294967054UL
#define CMD_LOADIDENTITY      4294967078UL
#define CMD_LOADIMAGE         4294967076UL
#define CMD_LOGO              4294967089UL
#define CMD_MARCH             4294967045UL
#define CMD_MEMCPY            4294967069UL
#define CMD_MEMCRC            4294967064UL
#define CMD_MEMSET            4294967067UL
#define CMD_MEMWRITE          4294967066UL
#define CMD_MEMZERO           4294967068UL
#define CMD_NUMBER            4294967086UL
#define CMD_PROGRESS          4294967055UL
#define CMD_REGREAD           4294967065UL
#define CMD_ROTATE            4294967081UL
#define CMD_SCALE             4294967080UL
#define CMD_SCREENSAVER       4294967087UL
#define CMD_SCROLLBAR         4294967057UL
#define CMD_SETFONT           4294967083UL
#define CMD_SETMATRIX         4294967082UL
#define CMD_SKETCH            4294967088UL
#define CMD_SLIDER            4294967056UL
#define CMD_SNAPSHOT          4294967071UL
#define CMD_SPINNER           4294967062UL
#define CMD_STOP              4294967063UL
#define CMD_SWAP              4294967041UL
#define CMD_TEXT              4294967052UL
#define CMD_TOGGLE            4294967058UL
#define CMD_TOUCH_TRANSFORM   4294967072UL
#define CMD_TRACK             4294967084UL
#define CMD_TRANSLATE         4294967079UL

#define DECR                  4UL
#define DECR_WRAP             7UL
#define DLSWAP_DONE           0UL
#define DLSWAP_FRAME          2UL
#define DLSWAP_LINE           1UL
#define DST_ALPHA             3UL
#define EDGE_STRIP_A          7UL
#define EDGE_STRIP_B          8UL
#define EDGE_STRIP_L          6UL
#define EDGE_STRIP_R          5UL
#define EQUAL                 5UL
#define GEQUAL                4UL
#define GREATER               3UL
#define INCR                  3UL
#define INCR_WRAP             6UL
#define INT_CMDEMPTY          32UL
#define INT_CMDFLAG           64UL
#define INT_CONVCOMPLETE      128UL
#define INT_PLAYBACK          16UL
#define INT_SOUND             8UL
```

```
#define INT_SWAP            1UL
#define INT_TAG             4UL
#define INT_TOUCH           2UL
#define INVERT              5UL

#define KEEP                1UL
#define L1                  1UL
#define L4                  2UL
#define L8                  3UL
#define LEQUAL              2UL
#define LESS                1UL
#define LINEAR_SAMPLES      0UL
#define LINES               3UL
#define LINE_STRIP          4UL
#define NEAREST             0UL
#define NEVER               0UL
#define NOTEQUAL            6UL
#define ONE                 1UL
#define ONE_MINUS_DST_ALPHA 5UL
#define ONE_MINUS_SRC_ALPHA 4UL
#define OPT_CENTER          1536UL
#define OPT_CENTERX         512UL
#define OPT_CENTERY         1024UL
#define OPT_FLAT            256UL
#define OPT_MONO            1UL
#define OPT_NOBACK          4096UL
#define OPT_NODL            2UL
#define OPT_NOHANDS         49152UL
#define OPT_NOHM            16384UL
#define OPT_NOPOINTER       16384UL
#define OPT_NOSECS          32768UL
#define OPT_NOTICKS         8192UL
#define OPT_RIGHTX          2048UL
#define OPT_SIGNED          256UL
#define PALETTED            8UL
#define FTPOINTS            2UL
#define RECTS               9UL

#define RAM_CMD             1081344UL
#define RAM_DL              1048576UL
#define RAM_G               0UL
#define RAM_PAL             1056768UL
#define RAM_REG             1057792UL


#define REG_ANALOG          1058104UL
#define REG_ANA_COMP        1058160UL
#define REG_CLOCK           1057800UL
#define REG_CMD_DL          1058028UL
#define REG_CMD_READ        1058020UL
#define REG_CMD_WRITE       1058024UL
#define REG_CPURESET        1057820UL
#define REG_CRC             1058152UL
#define REG_CSPREAD         1057892UL
#define REG_CYA0            1058000UL
#define REG_CYA1            1058004UL
#define REG_CYA_TOUCH       1058100UL
#define REG_DATESTAMP       1058108UL
```

```
#define REG_DITHER           1057884UL
#define REG_DLSWAP           1057872UL
#define REG_FRAMES           1057796UL
#define REG_FREQUENCY        1057804UL
#define REG_GPIO             1057936UL
#define REG_GPIO_DIR         1057932UL
#define REG_HCYCLE           1057832UL
#define REG_HOFFSET          1057836UL
#define REG_HSIZE            1057840UL
#define REG_HSYNC0           1057844UL
#define REG_HSYNC1           1057848UL
#define REG_ID               1057792UL
#define REG_INT_EN           1057948UL
#define REG_INT_FLAGS        1057944UL
#define REG_INT_MASK         1057952UL
#define REG_MACRO_0          1057992UL
#define REG_MACRO_1          1057996UL
#define REG_OUTBITS          1057880UL
#define REG_PCLK             1057900UL
#define REG_PCLK_POL         1057896UL
#define REG_PLAY             1057928UL
#define REG_PLAYBACK_FORMAT  1057972UL
#define REG_PLAYBACK_FREQ    1057968UL
#define REG_PLAYBACK_LENGTH  1057960UL
#define REG_PLAYBACK_LOOP    1057976UL
#define REG_PLAYBACK_PLAY    1057980UL
#define REG_PLAYBACK_READPTR 1057964UL
#define REG_PLAYBACK_START   1057956UL
#define REG_PWM_DUTY         1057988UL
#define REG_PWM_HZ           1057984UL
#define REG_RENDERMODE       1057808UL
#define REG_ROMSUB_SEL       1058016UL
#define REG_ROTATE           1057876UL
#define REG_SNAPSHOT         1057816UL
#define REG_SNAPY            1057812UL
#define REG_SOUND            1057924UL
#define REG_SWIZZLE          1057888UL
#define REG_TAG              1057912UL
#define REG_TAG_X            1057904UL
#define REG_TAG_Y            1057908UL
#define REG_TAP_CRC          1057824UL
#define REG_TAP_MASK         1057828UL
#define REG_TOUCH_ADC_MODE   1058036UL
#define REG_TOUCH_CHARGE     1058040UL
#define REG_TOUCH_DIRECT_XY  1058164UL
#define REG_TOUCH_DIRECT_Z1Z2 1058168UL
#define REG_TOUCH_MODE       1058032UL
#define REG_TOUCH_OVERSAMPLE 1058048UL
#define REG_TOUCH_RAW_XY     1058056UL
#define REG_TOUCH_RZ         1058060UL
#define REG_TOUCH_RZTHRESH   1058052UL
#define REG_TOUCH_SCREEN_XY  1058064UL
#define REG_TOUCH_SETTLE     1058044UL
#define REG_TOUCH_TAG        1058072UL
#define REG_TOUCH_TAG_XY     1058068UL
#define REG_TOUCH_TRANSFORM_A 1058076UL
#define REG_TOUCH_TRANSFORM_B 1058080UL
#define REG_TOUCH_TRANSFORM_C 1058084UL
#define REG_TOUCH_TRANSFORM_D 1058088UL
```

```
#define REG_TOUCH_TRANSFORM_E 1058092UL
#define REG_TOUCH_TRANSFORM_F 1058096UL
#define REG_TRACKER           1085440UL
#define REG_TRIM              1058156UL
#define REG_VCYCLE            1057852UL
#define REG_VOFFSET           1057856UL
#define REG_VOL_PB            1057916UL
#define REG_VOL_SOUND         1057920UL
#define REG_VSIZE             1057860UL
#define REG_VSYNC0            1057864UL
#define REG_VSYNC1            1057868UL


#define REPEAT                1UL
#define REPLACE               2UL
#define RGB332                4UL
#define RGB565                7UL
#define SRC_ALPHA             2UL
#define TEXT8X8               9UL
#define TEXTVGA               10UL
#define TOUCHMODE_CONTINUOUS  3UL
#define TOUCHMODE_FRAME       2UL
#define TOUCHMODE_OFF         0UL
#define TOUCHMODE_ONESHOT     1UL
#define ULAW_SAMPLES          1UL
#define ZERO                  0UL

#define DEV_ID_LOC 0x102400

#define VERTEX2F(x,y) ((1UL<<30)|(((x)&32767UL)<<15)|(((y)&32767UL)<<0))
#define VERTEX2II(x,y,handle,cell)
((2UL<<30)|(((x)&511UL)<<21)|(((y)&511UL)<<12)|(((handle)&31UL)<<7)|(((cell)&127UL)<<0))
#define BITMAP_SOURCE(addr) ((1UL<<24)|(((addr)&1048575UL)<<0))
#define CLEAR_COLOR_RGB(red,green,blue)
((2UL<<24)|(((red)&255UL)<<16)|(((green)&255UL)<<8)|(((blue)&255UL)<<0))
#define TAG(s) ((3UL<<24)|(((s)&255UL)<<0))
#define COLOR_RGB(red,green,blue)
((4UL<<24)|(((red)&255UL)<<16)|(((green)&255UL)<<8)|(((blue)&255UL)<<0))
#define BITMAP_HANDLE(handle) ((5UL<<24)|(((handle)&31UL)<<0))
#define CELL(cell) ((6UL<<24)|(((cell)&127UL)<<0))
#define BITMAP_LAYOUT(format,linestride,height)
((7UL<<24)|(((format)&31UL)<<19)|(((linestride)&1023UL)<<9)|(((height)&511UL)<<0))
#define BITMAP_SIZE(filter,wrapx,wrapy,width,height)
((8UL<<24)|(((filter)&1UL)<<20)|(((wrapx)&1UL)<<19)|(((wrapy)&1UL)<<18)|(((width)&511UL)<<
9)|(((height)&511UL)<<0))
#define ALPHA_FUNC(func,ref) ((9UL<<24)|(((func)&7UL)<<8)|(((ref)&255UL)<<0))
#define STENCIL_FUNC(func,ref,mask)
((10UL<<24)|(((func)&7UL)<<16)|(((ref)&255UL)<<8)|(((mask)&255UL)<<0))
#define BLEND_FUNC(src,dst) ((11UL<<24)|(((src)&7UL)<<3)|(((dst)&7UL)<<0))
#define STENCIL_OP(sfail,spass) ((12UL<<24)|(((sfail)&7UL)<<3)|(((spass)&7UL)<<0))
#define POINT_SIZE(size) ((13UL<<24)|(((size)&8191UL)<<0))
#define LINE_WIDTH(width) ((14UL<<24)|(((width)&4095UL)<<0))
#define CLEAR_COLOR_A(alpha) ((15UL<<24)|(((alpha)&255UL)<<0))
#define COLOR_A(alpha) ((16UL<<24)|(((alpha)&255UL)<<0))
#define CLEAR_STENCIL(s) ((17UL<<24)|(((s)&255UL)<<0))
#define CLEAR_TAG(s) ((18UL<<24)|(((s)&255UL)<<0))
#define STENCIL_MASK(mask) ((19UL<<24)|(((mask)&255UL)<<0))
#define TAG_MASK(mask) ((20UL<<24)|(((mask)&1UL)<<0))
#define BITMAP_TRANSFORM_A(a) ((21UL<<24)|(((a)&131071UL)<<0))
```

```
#define BITMAP_TRANSFORM_B(b) ((22UL<<24)|(((b)&131071UL)<<0))
#define BITMAP_TRANSFORM_C(c) ((23UL<<24)|(((c)&16777215UL)<<0))
#define BITMAP_TRANSFORM_D(d) ((24UL<<24)|(((d)&131071UL)<<0))
#define BITMAP_TRANSFORM_E(e) ((25UL<<24)|(((e)&131071UL)<<0))
#define BITMAP_TRANSFORM_F(f) ((26UL<<24)|(((f)&16777215UL)<<0))
#define SCISSOR_XY(x,y) ((27UL<<24)|(((x)&511UL)<<9)|(((y)&511UL)<<0))
#define SCISSOR_SIZE(width,height)
((28UL<<24)|(((width)&1023UL)<<10)|(((height)&1023UL)<<0))
#define CALL(dest) ((29UL<<24)|(((dest)&65535UL)<<0))
#define JUMP(dest) ((30UL<<24)|(((dest)&65535UL)<<0))
#define BEGIN(prim) ((31UL<<24)|(((prim)&15UL)<<0))
#define COLOR_MASK(r,g,b,a)
((32UL<<24)|(((r)&1UL)<<3)|(((g)&1UL)<<2)|(((b)&1UL)<<1)|(((a)&1UL)<<0))
#define CLEAR(c,s,t) ((38UL<<24)|(((c)&1UL)<<2)|(((s)&1UL)<<1)|(((t)&1UL)<<0))
#define END() ((33UL<<24))
#define SAVE_CONTEXT() ((34UL<<24))
#define RESTORE_CONTEXT() ((35UL<<24))
#define RETURN() ((36UL<<24))
#define MACRO(m) ((37UL<<24)|(((m)&1UL)<<0))
#define DISPLAY() ((0UL<<24))

#define FT_GPU_NUMCHAR_PERFONT (128)
#define FT_GPU_FONT_TABLE_SIZE (148)



/* FT800 font table Struktur */
typedef struct FT_Gpu_Fonts
{
      uint8_t      FontWidth[FT_GPU_NUMCHAR_PERFONT];
      unsigned int FontBitmapFormat;
      unsigned int FontLineStride;
      unsigned int FontWidthInPixels;
      unsigned int FontHeightInPixels;
      unsigned int PointerToFontGraphicsData;
} FT_Gpu_Fonts_t;


// ---------------- FT800 Funktionen:---------------------
void HOST_CMD_ACTIVE(void);              // Wake-Up
void HOST_CMD_WRITE(uint8_t CMD); // 5-Bit Befehl senden


void HOST_MEM_WR8(uint32_t addr, uint8_t data);       // 8-Bit in Register mit Adresse
addr schreiben
void HOST_MEM_WR16(uint32_t addr, uint32_t data);     // 16-Bit in Register mit Adresse
addr schreiben
void HOST_MEM_WR32(uint32_t addr, uint32_t data);     // 32-Bit in Register mit Adresse
addr schreiben
uint8_t HOST_MEM_RD8(uint32_t addr);                        // 8-Bit aus Register mit
Adresse addr lesen
uint32_t HOST_MEM_RD16(uint32_t addr);                      // 16-Bit aus Register mit
Adresse addr lesen
uint32_t HOST_MEM_RD32(uint32_t addr);                      // 32-Bit aus Register mit
Adresse addr lesen

// CMD-Funktionen:
uint8_t cmd_ready(void);                      // pruefen ob Co-Proz bereit ist
uint8_t cmd(uint32_t data);                   // CMD-Funktion
```

```
uint8_t cmd_execute(uint32_t data);            // CMD-Ausfuhrung
void cmd_calibrate(void);                      // Touch-Display Kalibrierungsfunktion
abrufen

void cmd_progress(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t options, uint16_t
val, uint16_t range);      // zeichnet progress bar
void cmd_track(int16_t x, int16_t y, int16_t w, int16_t h, int16_t tag);
                                               // Tracking
void cmd_spinner(int16_t x, int16_t y, uint16_t style, uint16_t scale);
                                                     // Spinner zeichnen
void cmd_slider(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t options, uint16_t
val, uint16_t range);      // Slider zeichnen
void cmd_text(int16_t x, int16_t y, int16_t font, uint16_t options, const char* str);
                                          // Text ausgeben
void cmd_button(int16_t x, int16_t y, int16_t w, int16_t h, int16_t font, uint16_t
options, const char* str); // Taste ausgeben
void cmd_keys(int16_t x, int16_t y, int16_t w, int16_t h, int16_t font, uint16_t options,
const char* str);          // Tastatur ausgeben
void cmd_memzero(uint32_t ptr, uint32_t num);
                                                                    // Null in Speicher
schreiben

void cmd_fgcolor(uint32_t c);                  // FG Colour setzen
void cmd_bgcolor(uint32_t c);                  // BG Colour setzen
void cmd_gradcolor(uint32_t c);                // Gradient Colour setzen

void cmd_gradient(int16_t x0, int16_t y0, uint32_t rgb0, int16_t x1, int16_t y1, uint32_t
rgb1);                     // Gradient zeichnen

// Matrix-Funktionen:
void cmd_loadidentity(void);
void cmd_setmatrix(void);
void cmd_rotate(int32_t angle);
void cmd_translate(int32_t tx, int32_t ty);

#endif
```

```
// FT800 Bibliothek

#include <asf.h>
#include <stdlib.h>
#include <string.h>
#include <ioport.h>

#include "ft800.h"

// SPI-Schnittstelle fuer Display:
// Enable - pin -> PC4
struct spi_device spi_device_conf = {
       .id = IOPORT_CREATE_PIN(PORTC, 4)
};

// ------------- Funktionen ------------------
//

// 5-Bit Befehl an ft800 schicken:
void HOST_CMD_WRITE(uint8_t CMD)
{
  uint8_t data_buffer[3]={CMD,0x00,0x00};
  spi_select_device(&SPIC,&spi_device_conf);
  spi_write_packet(&SPIC, data_buffer,3);
  spi_deselect_device(&SPIC,&spi_device_conf);
}

// Wake-Up - Befehl
void HOST_CMD_ACTIVE(void)
{
  uint8_t data_buffer[3]={0x00,0x00,0x00};
  spi_select_device(&SPIC,&spi_device_conf);
  spi_write_packet(&SPIC, data_buffer,3);
  spi_deselect_device(&SPIC,&spi_device_conf);
}

// 8-Bit in Register mit Adresse addr schreiben:
void HOST_MEM_WR8(uint32_t addr, uint8_t data)
{
       uint8_t data_buffer[4]={((addr>>16)|0x80), (addr>>8), addr, data};
       spi_select_device(&SPIC, &spi_device_conf);
       spi_write_packet(&SPIC, data_buffer, 4);
       spi_deselect_device(&SPIC,&spi_device_conf);
}

// 16-Bit in Register mit Adresse addr schreiben:
void HOST_MEM_WR16(uint32_t addr, uint32_t data)
{
uint8_t data_buffer[5]={((addr>>16)|0x80), (addr>>8), addr, data, (data>>8)};
spi_select_device(&SPIC,&spi_device_conf);
spi_write_packet(&SPIC, data_buffer,5);
spi_deselect_device(&SPIC,&spi_device_conf);
}

// 32-Bit in Register mit Adresse addr schreiben:
void HOST_MEM_WR32(uint32_t addr, uint32_t data)
{
```

```
 uint8_t data_buffer[7]={((addr>>16)|0x80), (addr>>8), addr, data, (data>>8), (data>>16),
(data>>24)};
 spi_select_device(&SPIC,&spi_device_conf);
 spi_write_packet(&SPIC, data_buffer,7);
 spi_deselect_device(&SPIC,&spi_device_conf);
}

// 8-Bit aus Register mit Adresse addr lesen:
uint8_t HOST_MEM_RD8(uint32_t addr)
{
uint8_t recieved_data8[] = {0};                              // buffer for recieved Byte
// Address and dummy Byte:
uint8_t data_buffer[4]={((addr>>16)|0x00), (addr>>8), addr, 0x00};

spi_select_device(&SPIC,&spi_device_conf);            // CS pin low
spi_write_packet(&SPIC, data_buffer, 4);       // send 3 Address Bytes and 1 dummy Byte
spi_read_packet(&SPIC, recieved_data8, 1);            // read data Byte
spi_deselect_device(&SPIC,&spi_device_conf);    // CS pin high
return recieved_data8[0];                                    // Return readed Byte

}

// 16-Bit aus Register mit Adresse addr lesen: // wird nicht benutzt
uint32_t HOST_MEM_RD16(uint32_t addr)
{

}

// 32-Bit aus Register mit Adresse addr lesen:
uint32_t HOST_MEM_RD32(uint32_t addr)
{

uint8_t recieved_data8[4] = {0, 0, 0, 0};             // create buffer for recieved Data
uint32_t recieved_data32=0;
// Address and dummy Byte:
// Byte1: 0 0 AddrBits21-16
// Byte2: AddrBits15-8
// Byte3: AddrBits7-0
// Byte4: dummy Byte
// Read Bytes: Byte0 ... Byte n
uint8_t data_buffer[4]={((addr>>16)&0x3F), (addr>>8)&0xFF, addr&0xFF, 0x00};

spi_select_device(&SPIC,&spi_device_conf);            // CS pin low
spi_write_packet(&SPIC, data_buffer, 4);       // send 3 Address Bytes and 1 dummy Byte
spi_read_packet(&SPIC, recieved_data8, 4);            // read 4 Byte data
spi_deselect_device(&SPIC,&spi_device_conf);    // CS pin high
recieved_data32=recieved_data8[3];
recieved_data32=recieved_data32<<8;
recieved_data32=recieved_data32|recieved_data8[2];
recieved_data32=recieved_data32<<8;
recieved_data32=recieved_data32|recieved_data8[1];
recieved_data32=recieved_data32<<8;
recieved_data32=recieved_data32|recieved_data8[0];
return recieved_data32;                                          // Return readed 4
Byte
}

// ------------------------------ CMD Funktionen: ---------------------------------
uint8_t cmd_execute(uint32_t data)
```

```
{
       uint32_t cmdBufferRd = 0;
    uint32_t cmdBufferWr = 0;

    cmdBufferRd = HOST_MEM_RD32(REG_CMD_READ);
    cmdBufferWr = HOST_MEM_RD32(REG_CMD_WRITE);

    uint32_t cmdBufferDiff = cmdBufferWr-cmdBufferRd;

    if( (4096-cmdBufferDiff) > 4)
    {
        HOST_MEM_WR32(RAM_CMD + cmdBufferWr, data);
        HOST_MEM_WR32(REG_CMD_WRITE, cmdBufferWr + 4);
        return 1;
    }
    return 0;
}

uint8_t cmd(uint32_t data)
{
       uint8_t tryCount = 255;
       for(tryCount = 255; tryCount > 0; --tryCount)
       {
              if(cmd_execute(data)) { return 1; }
       }
       return 0;
}

uint8_t cmd_ready(void)
{
    uint32_t del_mask12bit=4095; // 0000 0000 0000 0000 0000 1111 1111 1111
       uint32_t cmdBufferRd = (HOST_MEM_RD32(REG_CMD_READ)&del_mask12bit); // read 12 bit
       uint32_t cmdBufferWr = (HOST_MEM_RD32(REG_CMD_WRITE)&del_mask12bit); // read 12 bit
       return ((cmdBufferRd) == (cmdBufferWr)) ? 1 : 0; // return 1 if ready, return 0 -
if not ready
}

// Kalibrierung:
void cmd_calibrate(void)
{
       cmd(CMD_CALIBRATE); // send calibrate command
       cmd(0);
}

// progress bar:
void cmd_progress(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t options, uint16_t
val, uint16_t range)
{
       cmd(CMD_PROGRESS);
       cmd( ((uint32_t)y<<16)|(x & 0xffff) );
       cmd( ((uint32_t)h<<16)|(w & 0xffff) );
       cmd( ((uint32_t)val<<16)|(options & 0xffff) );
       cmd( (uint32_t)range );
}

// Track:
void cmd_track(int16_t x, int16_t y, int16_t w, int16_t h, int16_t tag)
{
    cmd(CMD_TRACK);
```

```
    cmd( ((uint32_t)y<<16)|(x & 0xffff) );
    cmd( ((uint32_t)h<<16)|(w & 0xffff) );
    cmd( (uint32_t)tag );
}

// Spinner:
void cmd_spinner(int16_t x, int16_t y, uint16_t style, uint16_t scale)
{
    cmd(CMD_SPINNER);
    cmd( ((uint32_t)y<<16)|(x & 0xffff) );
    cmd( ((uint32_t)scale<<16)|style );

}

// Slider
void cmd_slider(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t options, uint16_t
val, uint16_t range)
{
       cmd(CMD_SLIDER);
       cmd( ((uint32_t)y<<16)|(x & 0xffff) );
       cmd( ((uint32_t)h<<16)|(w & 0xffff) );
       cmd( ((uint32_t)val<<16)|(options & 0xffff) );
       cmd( (uint32_t)range );
}

// Text:
void cmd_text(int16_t x, int16_t y, int16_t font, uint16_t options, const char* str)
{
       /*
              i: data pointer
              q: str  pointer
              j: loop counter
       */

       uint16_t i,j,q;
       const uint16_t length = strlen(str);
       if(!length) return ;

       uint32_t* data = (uint32_t*) calloc((length/4)+1, sizeof(uint32_t));

       q = 0;
       for(i=0; i<(length/4); ++i, q=q+4)
       {
              data[i] = (uint32_t)str[q+3]<<24 | (uint32_t)str[q+2]<<16 |
(uint32_t)str[q+1]<<8 | (uint32_t)str[q];
       }
       for(j=0; j<(length%4); ++j, ++q)
       {
              data[i] |= (uint32_t)str[q] << (j*8);
       }

       cmd(CMD_TEXT);
       cmd( ((uint32_t)y<<16)|(x & 0xffff) );
    cmd( ((uint32_t)options<<16)|(font & 0xffff) );
       for(j=0; j<(length/4)+1; ++j)
       {
              cmd(data[j]);
       }
       free(data);
```

```
}

// Button:
void cmd_button(int16_t x, int16_t y, int16_t w, int16_t h, int16_t font, uint16_t
options, const char* str)
{
	/*
		i: data pointer
		q: str  pointer
		j: loop counter
	*/

	uint16_t i,j,q;
	const uint16_t length = strlen(str);
	if(!length) return ;

	uint32_t* data = (uint32_t*) calloc((length/4)+1, sizeof(uint32_t));

	q = 0;
	for(i=0; i<(length/4); ++i, q=q+4)
	{
		data[i] = (uint32_t)str[q+3]<<24 | (uint32_t)str[q+2]<<16 |
(uint32_t)str[q+1]<<8 | (uint32_t)str[q];
	}
	for(j=0; j<(length%4); ++j, ++q)
	{
		data[i] |= (uint32_t)str[q] << (j*8);
	}

	cmd(CMD_BUTTON);
	cmd( ((uint32_t)y<<16)|(x & 0xffff) );
	cmd( ((uint32_t)h<<16)|(w & 0xffff) );
    cmd( ((uint32_t)options<<16)|(font & 0xffff) );
	for(j=0; j<(length/4)+1; ++j)
	{
		cmd(data[j]);
	}
	free(data);
}

// Tastatur:
void cmd_keys(int16_t x, int16_t y, int16_t w, int16_t h, int16_t font, uint16_t options,
const char* str)
{
	/*
		i: data pointer
		q: str  pointer
		j: loop counter
	*/

	uint16_t i,j,q;
	const uint16_t length = strlen(str);
	if(!length) return ;

	uint32_t* data = (uint32_t*) calloc((length/4)+1, sizeof(uint32_t));

	q = 0;
	for(i=0; i<(length/4); ++i, q=q+4)
	{
```

```
            data[i] = (uint32_t)str[q+3]<<24 | (uint32_t)str[q+2]<<16 |
(uint32_t)str[q+1]<<8 | (uint32_t)str[q];
      }
      for(j=0; j<(length%4); ++j, ++q)
      {
            data[i] |= (uint32_t)str[q] << (j*8);
      }

      cmd(CMD_KEYS);
      cmd( ((uint32_t)y<<16)|(x & 0xffff) );
      cmd( ((uint32_t)h<<16)|(w & 0xffff) );
   cmd( ((uint32_t)options<<16)|(font & 0xffff) );
      for(j=0; j<(length/4)+1; ++j)
      {
            cmd(data[j]);
      }
      free(data);
}

// Null schreiben in Speicher:
void cmd_memzero(uint32_t ptr, uint32_t num)
{
      cmd(CMD_MEMZERO);
      cmd(ptr);
      cmd(num);
}

// FG Colour setzen:
void cmd_fgcolor(uint32_t c)
{
      cmd(CMD_FGCOLOR);
      cmd(c);
}

// BG Colour setzen:
void cmd_bgcolor(uint32_t c)
{
      cmd(CMD_BGCOLOR);
      cmd(c);
}

// Gradient Colour setzen:
void cmd_gradcolor(uint32_t c)
{
      cmd(CMD_GRADCOLOR);
      cmd(c);
}

// Gradient:
void cmd_gradient(int16_t x0, int16_t y0, uint32_t rgb0, int16_t x1, int16_t y1, uint32_t
rgb1)
{
      cmd(CMD_GRADIENT);
      cmd( ((uint32_t)y0<<16)|(x0 & 0xffff) );
      cmd(rgb0);
      cmd( ((uint32_t)y1<<16)|(x1 & 0xffff) );
      cmd(rgb1);
}
```

```
// Matrix Funktionen:
void cmd_loadidentity(void)
{
	cmd(CMD_LOADIDENTITY);
}

void cmd_setmatrix(void)
{
	cmd(CMD_SETMATRIX);
}

void cmd_rotate(int32_t angle)
{
	cmd(CMD_ROTATE);
	cmd(angle);
}

void cmd_translate(int32_t tx, int32_t ty)
{
	cmd(CMD_TRANSLATE);
	cmd(tx);
	cmd(ty);
}
```

C.3 Flussdiagramme

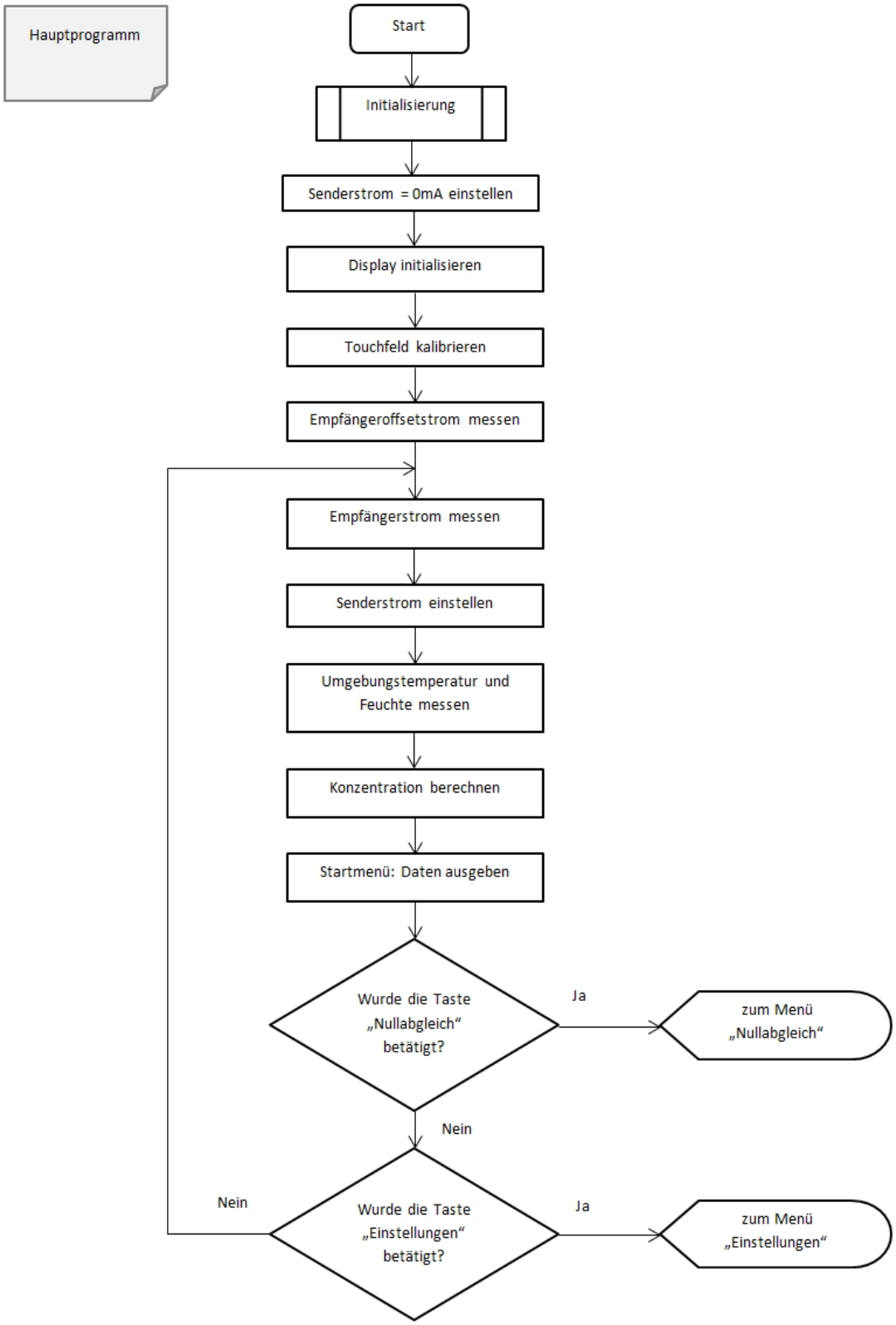

Abbildung C.1: Flussdigramm zum Hauptprogramm.

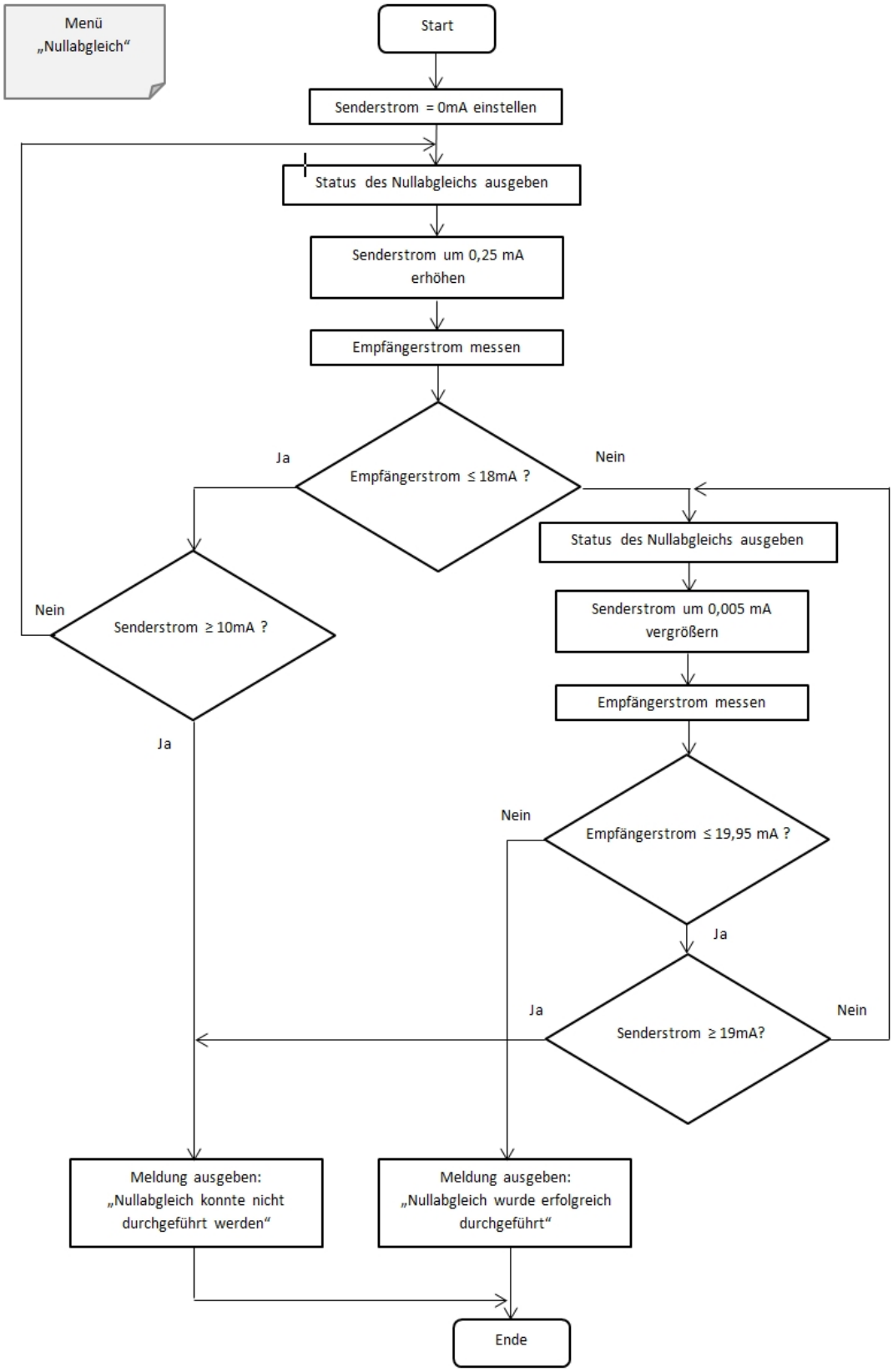

Abbildung C.2: Flussdigramm zum automatischen Nullabgleich.

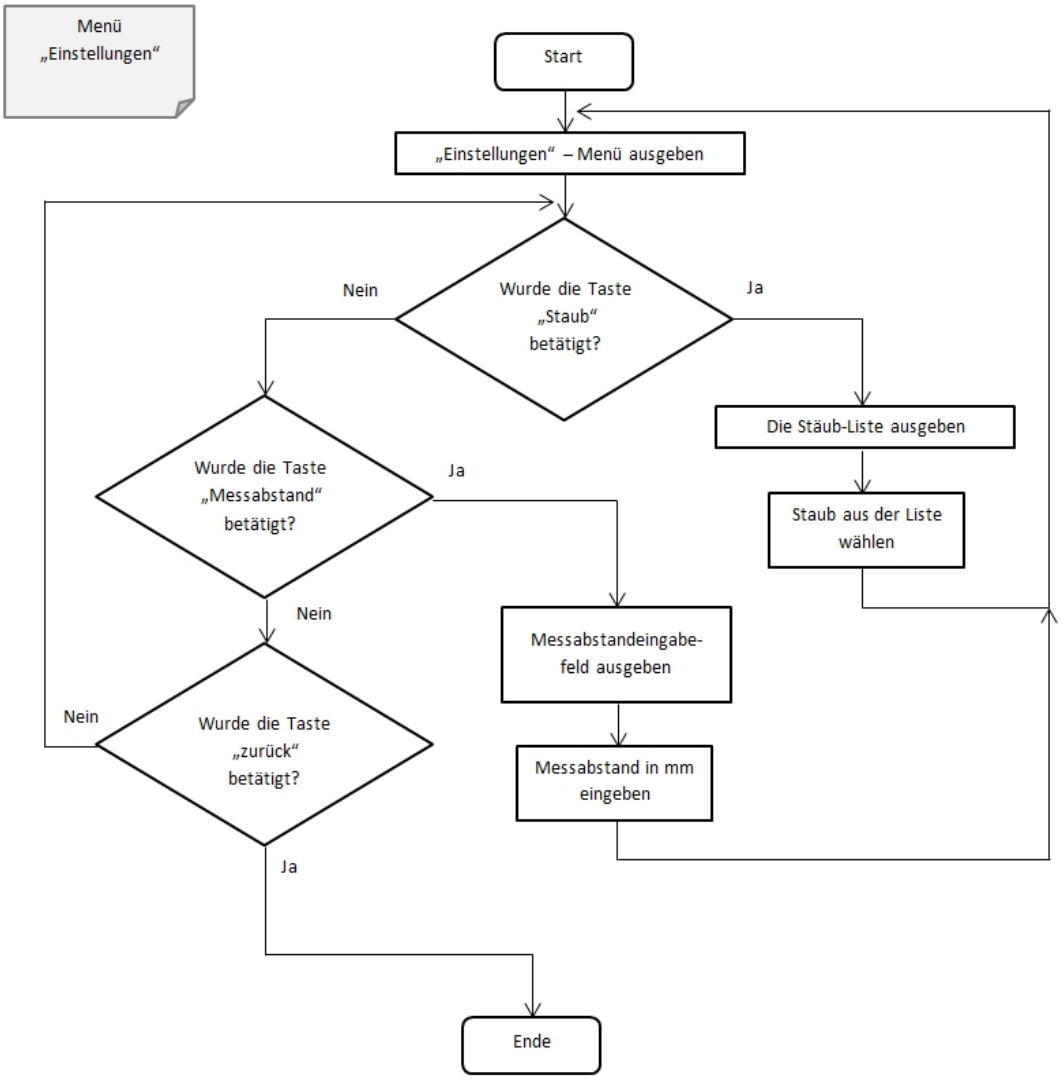

Abbildung C.3: Flussdigramm zum Menü Einstellungen.

Anhang D

Messdaten

I_{SKG} [mA]	I_{mess} [mA]	ΔI_{abs} [mA]	ΔI_{rel} [%]
0,7900	0,8490	0,0590	6,9494
1,2500	1,3040	0,0540	4,1411
2,2800	2,3380	0,0580	2,4808
3,4300	3,4780	0,0480	1,3801
4,6300	4,6700	0,0400	0,8565
5,3700	5,4200	0,0500	0,9225
6,3800	6,4280	0,0480	0,7467
7,5400	7,5840	0,0440	0,5802
8,3400	8,3790	0,0390	0,4654
9,4200	9,4570	0,0370	0,3912
10,2400	10,2750	0,0350	0,3406
11,0600	11,1010	0,0410	0,3693
12,1900	12,2140	0,0240	0,1965
13,3100	13,3400	0,0300	0,2249
14,4500	14,4750	0,0250	0,1727
15,3100	15,3330	0,0230	0,1500
16,4600	16,4820	0,0220	0,1335
17,3300	17,3520	0,0220	0,1268
18,5000	18,5160	0,0160	0,0864
19,3800	19,3950	0,0150	0,0773
19,9900	20,0000	0,0100	0,0500

Tabelle D.1: Am SKG 7 abgelesener Empfängerstrom und mit dem Präzisionsmultimeter gemessener Empfängerstrom sowie die absolute und relative Abweichung zwischen beiden.