Hochschule für Angewandte Wissenschaften Hamburg
*Hamburg University of Applied Sciences*

# Bachelorthesis

Torben Hensel

## Entwicklung eines neuartigen optischen Partikelsensors zur Luftgütemessung

*Fakultät Technik und Informatik*
*Department Informations- und Elektrotechnik*

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

Torben Hensel

# Entwicklung eines neuartigen optischen Partikelsensors zur Luftgütemessung

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung
im Studiengang Informations- und Elektrotechnik
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. rer. nat. Rasmus Rettig
Zweitgutachter : Prof. Dr.Ing. Ralf Wendel

Abgegeben am 14. September 2016

**Torben Hensel**

## Thema der Bachelorthesis

Entwicklung eines neuartigen optischen Partikelsensors zur Luftgütemessung

## Stichworte

Partikelsensor, Luftgüte, Mie-Streuung, Messtechnik

## Kurzzusammenfassung

Diese Arbeit umfasst die Entwicklung und den Aufbau eines Prototypen eines Partikelsensors, welcher auf Basis der Mie-Streuung Partikelgrößen unterscheidet.

**Torben Hensel**

## Title of the paper

Development of an advanced optical partical detector for air quality monitoring

## Keywords

particle sensor, air quality, Mie scattering, sensor technology

## Abstract

This report describes the development and the construction of a prototyp of a particle sensor, that categorices the partical size on base of its Mie scattering

# Danksagung

An dieser Stelle bedanke ich mich bei allen, die mich bei der Erstellung der Abschlussarbeit unterstützt und motiviert haben. Besonders möchte ich mich bei meiner Familie bedanken für die viel Geduld mir gegenüber. Außerdem bedanke ich mich bei Prof. Dr. rer. nat. Rasmus Rettig für die Betreuung und bei Prof. Dr. Ralf Wendel für die Übernahme des Zweitgutachtens dieser Arbeit.
Ein spezieller Dank geht an alle, die diese Arbeit Korrektor gelesen haben.

# Inhaltsverzeichnis

# Tabellenverzeichnis

# Abbildungsverzeichnis

# Abkürzungsverzeichnis

| | |
|---|---|
| ASCII | American Standard Code for Information Interchange |
| GPIO | General-purpose input/output |
| HAW | Hochschule für angewandte Wissenschaften Hamburg |
| I2C | Inter-Integrated Circuit |
| IDE | Integrated Design Environment |
| IR | Infrarot |
| ISR | Interrupt Service Routine |
| LAN | Local Area Network |
| LED | Light-Emitting Diode |
| rel. LF | Relative Luftfeuchtigkeit |
| SPI | Serial Peripheral Interface |
| SSH | Secure Shell |
| UART | Universal Asynchronous Receiver Transmitter |
| USB | Universal Serial Bus |
| WLAN | Wireless Local Area Network |

# 1. Einführung

Im Rahmen des Urban Mobility Lab der Fakultät Technik und Informatik der HAW kam es zu der Überlegung, welche Auswirkung die Elektrifizierung des öffentlichen Linienbusverkehrs auf die Feinstaubkonzentration in Straßennähe hat.
Daraus ergab sich die Idee eines portablen, optischen Partikelsensors, der die Abhängigkeit der Streuverteilung von dem Verhältnis von Partikeldurchmesser zu der Wellenlänge des gestreuten Lichtes nutzt um die Partikel zu kategorisieren.

## 1.1. Inhalt der Thesis

Diese Thesis wird im Urban Mobility Lab, finanziert durch die Fakultät Technik und Informatik der HAW, erarbeitet.
Sie umfasst die gesamte Entwicklung des Partikelsensors, startend mit einem Überblick über die physikalischen Eigenschaften der Streuung von Licht an Partikeln, im Besonderem der Mie-Steuung. An das V-Modell angelehnt wird eine Anforderungsanalyse, für den Einsatz als mobile Messeinheit, durchgeführt. Ein Hauptaugenmerk wird dabei auf den Einsatz im Freien gelenkt. Das Ergebnis wird als Grundlage für den Systementwurf genutzt, um die notwendigen Systemkomponenten zu definieren. Die Komponentenauswahl, Schaltungserstellung und Programmierung der Systemkomponenten erfolgt in der Realisierung. Abschließend wird der Gesamtaufbau getestet und die Messwerte mit einem abschließendem Ausblick und Fazit bewertet.

# 2. Grundlagen

## 2.1. optische Streuung

Als Steuung bezeichnet man die Ablenkung eines Objektes, hier eine elektromagnetische Welle, durch die Wechselwirkung mit einem anderen Objekt. Bei der optischen Streuung sind dieses meist Atome, Moleküle und Feinstaub.
Die optische Streuung kann in drei Bereichen eingeteilt werden:

- Objektdurchmesser « Wellenlänge: dort kann die Rayleigh-Streuung benutzt werden
- Qbjektdurchmesser » Wellenlänge: dort kann klassische, geometrische Streuung genutzt werden
- Objektdurchmesser in der Größenordnung der Wellenlänge: hier kann nur die Mie-Streuung genutzt werden

Da Feinstaubpartikel großteils im dritten Größenordnungsbereich befinden, bietet die Mie-Streuung einen guten Ansatz zu dessen Vermessung.

### 2.1.1. Mie-Streuung

Die Mie-Streuung ist eine exakte Lösung der Maxwell-Gleichungen für die Streuung elektromagnetischer Wellen an kugelförmigen Objekten, jeder Größe.
Ihr Streumuster ist davon abhängig, in welchem Größenverhältnis die Wellenlänge und der Partikeldurchmesser zu einander stehen. In Abbildung 2.1 sind drei Steuverteilungen dargestellt, alle Plots wurden mit der Software MiePlot in der Version 4503 erstellt. Es fällt auf, dass die Vorwärtsstreuung mit steigender Partikelgröße zu nimmt.
In Abbildung 2.2 werden die Streumuster bei 400 nm und 900 nm Wellenlänge am gleichen Objekt dargestellt. Hier sieht man, bei 400 nm ist die Vorwärtstreuung ausgeprägter als bei 900 nm Wellenlänge. Es läst sich damit erklären, dass der Partikel für die Strahlung mit 400 nm Wellenlänge im Verhältnis zu sich selbst größer ist als für die Strahlung mit 900 nm .
In Abbildung 2.3 werden die Steuermuster bei 400 nm und 900 nm Wellenlänge an Objekten,

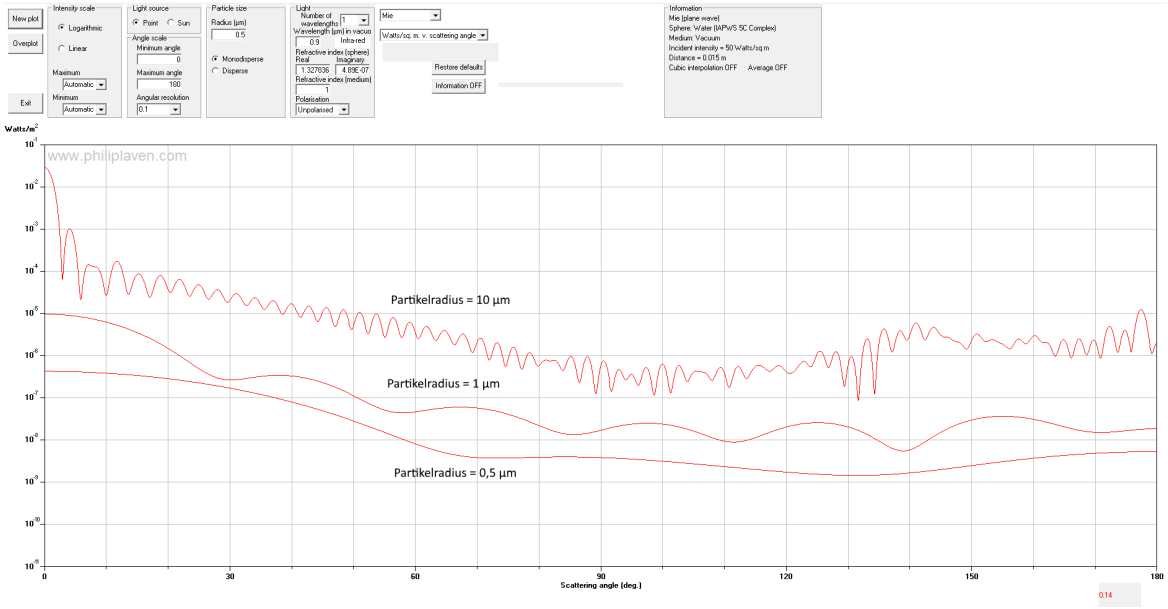


Abbildung 2.1.: Plot dreier Streuverteilungen bei 900 nm Wellenlänge und Partikelradien von 10 µm, 1 µm und 0,5 µm

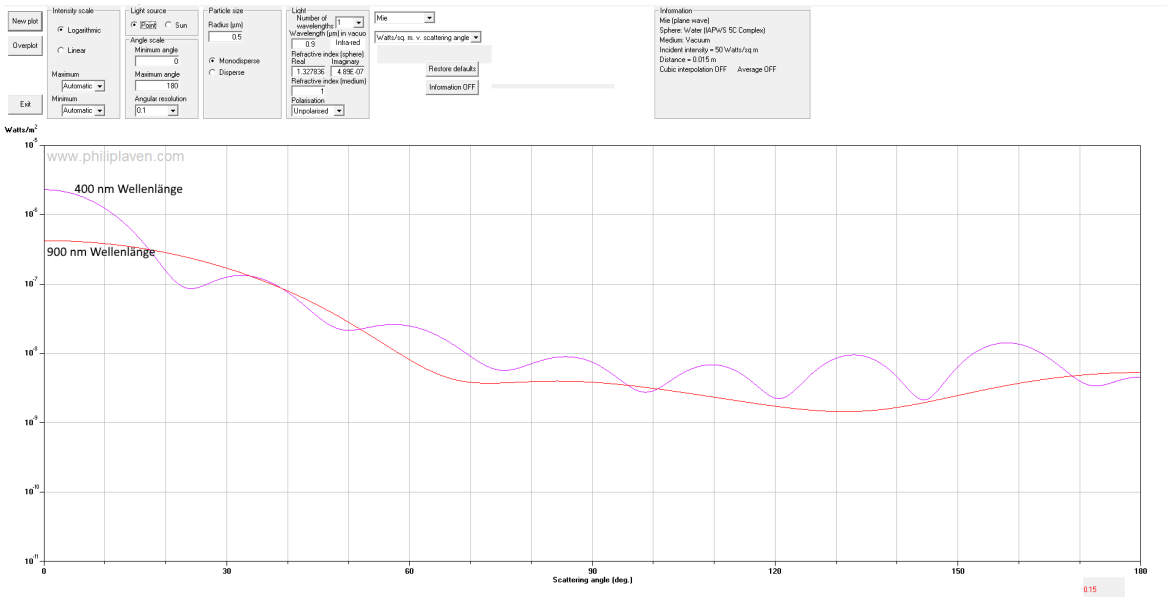


Abbildung 2.2.: Plot zweier Streuverteilungen bei 400 nm und 900 nm Wellenlänge und 0,5 µm Partikelradien

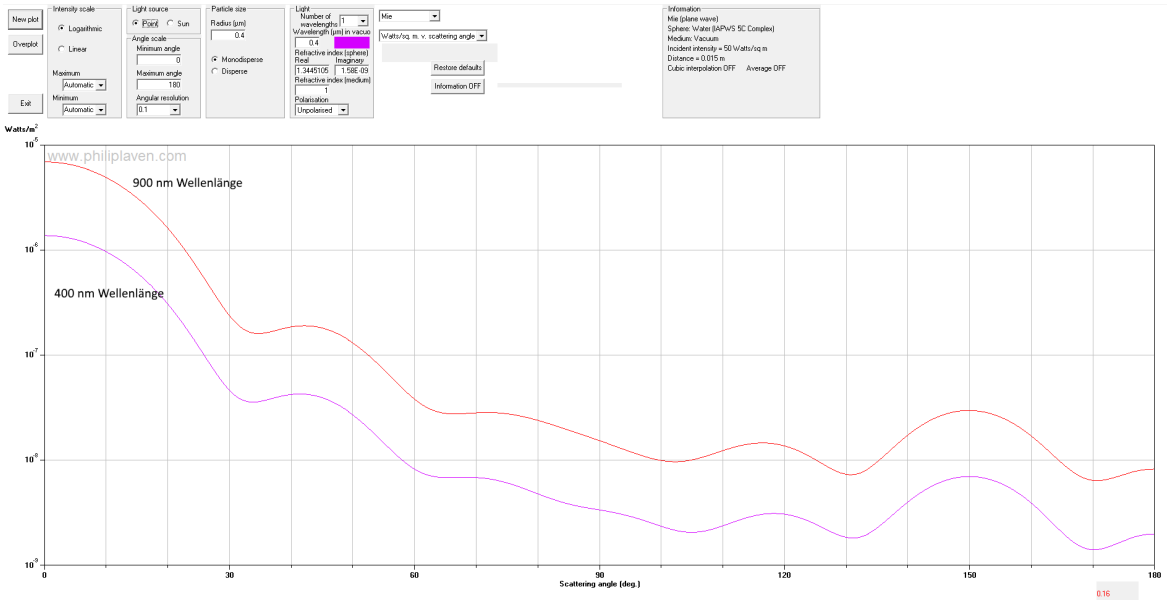


Abbildung 2.3.: Plot zweier Streuverteilungen bei 400 nm und 900 nm Wellenlänge und 0,5 µm Partikelradien

deren Radius der jeweiligen Wellenlänge entsprechen. Es zeigt, dass die winkelabhängige Verteilung bei beiden gleich ist, nur die Intensität ist vom Partikeldurchmesser abhängig. Diese Effekte sollen in dem zu entwickelnden Sensor gemessen werden können.

# 3. Anforderungsanalyse

Der Partikelsensor soll sowohl als autonome Datenerfassungseinheit fungieren können, als auch einfach in ein bestehendes System eingebunden werden können.
Als Einsatzgebiet ist die Verwendung als mobile Messstation in Verkehrsbereichen angedacht. Daraus ergeben sich die folgende Anforderungen.

## 3.1. Umweltanforderungen

Der Sensor soll in Hamburg im Freien einsetzbar sein. Laut Deutschem Wetterdienst beträgt dort im langjährigen Mittel die tiefste mittlere Tagestiefsttemperatur etwa -2 °C im Januar und die höchste mittlere Tageshöchsttemperatur etwa 23 °C im Juli [ ICDC (2010) ]. Als Standardabweichung von der mittleren Tagestemperatur werden etwa 5 °C angegeben, für die Tageshöchst- und -tiefstwerte muss also mit einer noch stärkeren Schwankung gerechnet werden. Deshalb wird die untere Betriebstemperaturgrenze der Komponenten auf -20 °C festgesetzt. Für die obere Grenze wird noch ein zusätzlicher Puffer eingeräumt, wegen eventueller Aufheizung des Gehäuses durch Sonneneinstrahlung und somit auf 60 °C festgelegt.
Außerdem erfordert der Einsatz im Freien, dass keine Niederschläge durch die Luftzuführung in das Gehäuse eindringen können, auch nicht bei starkem Wind. An den Luftzuführungen muss das Eindringen von Insekten verhindert werden.
Gegenüber Erschütterungen beim Transport muss das System als mobile Einheit robust sein.

## 3.2. Mechanische Anforderungen

Der Transport einer Einheit soll ohne komplizierte Verpackung und Sicherung möglich sein. Die Masse des Gesamtsystems soll 2 kg und Außenmaße von 200 mm x 100 mm x 100 mm nicht übersteigen.

## 3.3. Elektrische Anforderungen

Die Laufzeit im Akkubetrieb soll mindestens 24h betragen. Das Gerät soll robust gegen elektromagnetische Störfelder in seiner Umgebung sein, der normale Messbetrieb soll nicht von alltäglichen Störquellen, wie Mobilfunkgeräte oder WLAN-Geräte, in einem Umkreis von 1 m beeinflusst werden. Außerdem soll das Gerät die Richtlinien für elektromagnetische Verträglichkeit der CE-Kennzeichnung erfüllen.

## 3.4. Sensorische Anforderungen

Der Sensor soll zwei Größenklassen, < 1 µm und > 10 µm Durchmesser, an Partikeln unterscheiden können, und für diese die Konzentration in der Luft bestimmen können, optional ist eine zusätzliche zwischen den anderen liegende Größenklasse. Die Messgenauigkeit soll vergleichbar mit handelsüblichen Handgräten sein. Die Ansprechzeit soll unter 30 Sekunden liegen, um auch kurze Ereignisse, wie zum Beispiel einen an einer Haltestelle haltenden Bus, zu detektieren.
Außerdem sollen parallel folgende Umweltdaten gemessen werden, die Lufttemperatur mit einer Messgenauigkeit von +/-1 °C , der Luftdruck mit +/-1 kPa und Luftfeuchtigkeit mit +/-5 % rel. LF .
Die Messfrequenz soll einstellbar sein, mit mindestens einem Maximalwert von 1 Hz.
Optional soll eine Zeiterfassung der Messungen möglich sein.

## 3.5. Schnittstellenanforderungen

Die Messdaten müssen auf einem Wechseldatenspeicher abgelegt werden. Dieser soll eine Speicherkapazität für Messdaten von 10 Tagen besitzen. Optional können sie über ein anderes Interface ausgelesen werden, zum Beispiel WLAN oder USB.
Die Daten sollen in einem sowohl gut von Software als auch vom Menschen lesbaren Format vorliegen.
Der Akkumulator muss ohne Aus- und Einbau geladen werden können. Eine externe Ladebuchse ist optional, sowie ein Laden im Messbetrieb.

Tabelle 3.1.: Übersicht aller Anforderungen

| **Umweltanforderungen** | **Mindestanforderung** | **Optimum** | **Auflösung** | **Genauigkeit** |
|---|---|---|---|---|
| Temperaturbereich | -20 °C bis +60 °C | - | - | - |
| Niederschlagsfest | kein Eindringen von Niederschlag von Oben | bei starkem, seitlichem Wind | - | - |
| Insektensicher | kein Eindringen von Insekten möglich | - | - | - |
| Robustheit gegen Erschütterung | kann ohne Polsterung transportiert werden | - | - | - |
| **Mechanische Anforderungen** | | | | |
| Transportabel | ohne Verpackung transportabel | Mit einer Hand | - | - |
| Masse | maximal 2 kg | maximal 1 kg | - | - |
| Außenmaße | 200 mm x 200 mm x 100 mm | 200 mm x 100 mm x 100 mm | - | - |
| **Elektrische Anforderungen** | | | | |
| Laufzeit im Akkubetrieb | mindestens 24 h | - | - | - |
| elektromagnetische Störungen | keine Störung durch alltägliche Elektronik, z.B. Mobilfunkgeräten, in 1 m Abstand gestört | in direktem Kontakt gestört | - | - |
| Elektromagnetische Verträglichkeit (EMV) | Erfüllt Richtlinien für CE-Kennzeichen | - | - | - |
| **Sensorische Anforderungen** | | | | |
| Partikeldurchmesser | 2 Größenklassen | zusätzliche Klasse da zwischen | <1 µm und >10 µm | - |
| Partikelkonzentration | separat für alle Größenklassen | - | - | 10% Vollausschlag |
| Ansprechzeit | maximal 30 s | 1 s | - | - |
| Messfrequenz | einstellbar von 1/60 Hz bis zu 1 Hz | - | - | - |
| Lufttemperatur | benötigt | - | 1/10 °C | +/- 1 °C |
| relative Luftfeuchtigkeit ( rel. LF ) | benötigt | - | 1 % rel. LF | +/- 5% rel. LF |
| Luftdruck | benötigt | - | 10 Pa | +/-1 kPa |
| **Schnittstellenanforderungen** | | | | |
| Speichern der Messdaten | intern auf Wechseldatenträger mit Speicherplatz für 10 Messtage | > 10 Messtage | - | - |
| Externes Auslesen der Daten über | drahtgebundene Schnittstelle oder durch Wechsel des Datenträgers | über drahtlose Schnittstelle | - | - |
| Datenformat | Maschinen und Menschen lesbar | - | - | - |
| Akkumulator Ladeeinrichtung | ohne Aus- und Einbau möglich | externe Ladebuchse und laden während des Betriebes | - | - |

# 4. Systementwurf

Um die Komplexität und Dauer der Entwicklung zu begrenzen, sollen außer für die sensitive Einheit möglichst Fertigkomponenten verwendet werden. Im folgenden Abschnitt ist die Identifizierung und Spezifikation dieser dargestellt.

## 4.1. Grundkonzept

Das Gerät soll sowohl als eigenständiger Datensammler fungieren als auch als zusätzliche Sensoreinheit für bestehende Datenlogger. Daher wird es in zwei große Bereiche unterteilt, zum einen der Energieversorgungs- und Datenspeicherbereich, zum anderen die sensitive Einheit mit Messwertaufbereitung.
Die Datenspeicherung soll mit einen Einplatinencomputer als Kernstück erfolgen. Dieser ist für das Konfigurieren und Auslesen der sensitiven Einheit sowie Speicherung der Messwerte auf einem Wechseldatenträger zuständig. Die Energieversorgung soll über ein Akkumulator mit integrierter Ladeschaltung und einer selbst entwickelten, schaltbaren Spannungsversorgungseinheit erfolgen.
Die sensitive Einheit besteht aus einer Auswerte- und Steuereinheit, sowie der optischen Partikelmesseinheit. Die Auswerte- und Steuereinheit besteht aus einem Mikrocontroller-Entwicklungsbord und einer selbst entwickelten Interfaceplatine zur Anpassung der Ausgangspegel der Partikelmesseinheit. Die optische Partikelmesseinheit besteht aus einer optischen Kammer, einer Elektronik zur Partikelmessung und einem Lüfter zum Luftaustausch. In die Elektronik werden fertige Sensoren für die zusätzlichen Umweltmesswerte, Lufttemperatur, Luftdruck und relative Luftfestigkeit, integriert.
Das Gehäuse wird Zweiteilig entsprechend dem Grundkonzept, wobei beide Teile mit einem Steckverbinder für die Durchführung der Signal- und Energieleitungen verbunden sind.

Die benötigten Komponenten und ihre Zuordnung zu den umgesetzten Anforderungen sind in Tabelle 4.1 dargestellt.

Tabelle 4.1.: Übersicht benötigter Komponenten und der damit umgesetzten Anforderungen

| **Umweltanforderungen** | **umgesetzte Anforderung** | **relevante Komponente** |
|---|---|---|
| Niederschlagsfest | kein Eindringen von Niederschlag von Oben und bei seitlichem Wind | Gehäuse |
| Insekten sicher | x | Gehäuse |
| Robustheit gegen Erschütterung | kann ohne Polsterung transportiert werden | Gehäuse |
| **Mechanische Anforderungen** | | |
| Transportabel | Ohne Verpackung mit einer Hand | Gehäuse |
| Außenmaße | 200 mm x 200 mm x 100 mm | Gehäuse |
| **Elektrische Anforderungen** | | |
| Laufzeit im Akkubetrieb | mindestens 24 h | Akkumulator |
| **Sensorische Anforderungen** | | |
| Partikeldurchmesser | 2 Größenklassen | optische Partikelmesseinheit |
| Partikelkonzentration | separat für alle Größenklassen | optische Partikelmesseinheit |
| Ansprechzeit | maximal 30 s | Lüfter |
| Messfrequenz | einstellbar von 1/60 Hz bis zu 1 Hz | Einplatinencomputer / Mikrocontroller-Entwicklungsbord |
| Lufttemperatur | Auflösung 1/10 °C, Genauigkeit +/- 1 °C | zusätzliche Umweltsensoren |
| relative Luftfeuchtigkeit ( rel. LF ) | Auflösung 1 % rel. LF, Genauigkeit +/- 5 % rel. LF | zusätzliche Umweltsensoren |
| Luftdruck | Auflösung 10 Pa, Genauigkeit +/- 1 kPa | zusätzliche Umweltsensoren |
| **Schnittstellenanforderungen** | | |
| Speichern der Messdaten | intern auf Wechseldatenträger mit Speicherplatz für 10 Messtage | Wechseldatenträger |
| Externes Auslesen der Daten über | drahtgebundene Schnittstelle oder durch Wechsel des Datenträgers | Wechseldatenträger / Einplatinencomputer |
| Datenformat | Maschinen und Menschen lesbar | Einplatinencomputer |
| Akkumulator Ladeeinrichtung | ohne Aus- und Einbau möglich | Akkumulator |

## 4.2. Komponenten

Die benötigten Komponenten werden in diesem Abschnitt genauer spezifiziert und ihre Mindestanforderungen definiert. Alle elektronischen Komponenten haben als gemeinsame Anforderung eine Stromversorgungsschnittstelle mit der Spannungsversorgungseinheit.

### 4.2.1. Energieversorgung und Datenspeicherung

#### Einplatinencomputer

Der Einplatinencomputer dient zur Konfiguration und zum Auslesen des Mikrocontroller-Entwicklungsbords und dem Speichern der Messdaten auf dem Wechseldatenträger. Außerdem stellt er die Schnittstelle zum dahtgebundenem Auslesen der gespeicherten Messdaten für andere Computersysteme bereit. Auch wird zum Programmieren und Debuggen eine Schnittstelle benötigt.
Mindestanforderungen:

- Schnittstelle zum Mikrocontroller-Entwicklungsbord
- Schnittstelle zum dahtgebundenem Auslesen der gespeicherten Messdaten mit hoher Kompatibilität zu anderen Computersystemen
- Schnittstelle zum Wechseldatenträger
- Schnittstelle zum Programmieren und Debuggen
- Betriebssystem mit den benötigten Grundfunktionalitäten
- Rechenleistung zur Bewältigung der Aufgaben

#### Wechseldatenträger

Der Wechseldatenträger muss Messdaten für 10 Messtage aufnehmen können.
Mindestanforderungen:

- Schnittstelle zum Einplatienenrechner
- Speicherplatz für 10 Messtage

#### Spannungsversorgungseinheit

Die Spannungsversorgungseinheit übernimmt die Verteilung von Akkumulator und Anpassung aller Spannungen an die anderen Komponenten, sowie das Schalten der Lasten.
Mindestanforderungen:

- Schnittstelle für Schaltsignale zum Mikrocontroller-Entwicklungsbord
- Schnittstelle zum Akkumulator
- Schittstellen zur Spannungsversorgung aller elektrischen Komponenten
- Verteilung und Anpassung der Versorgungsspannung für alle Komponenten

#### Akkumulator

Der Akkumulator speist Strom in Spannungsversorgungseinheit zur Weiterverteilung ein. Er muss über genug Kapazität verfügen um 24 Stunden Messbetrieb zu gewährleisten. Das Laden des Akkumulators muss ohne Ausbau möglich sein.
Mindestanforderungen:

- Schnittstelle zur Spannungsversorgungseinheit
- Schnittstelle zum Laden des Akkumulator ohne Ausbau
- Genügend Kapazität für 24 Stunden Messbetrieb

### 4.2.2. Sensitive Einheit

#### Mikrocontroller-Entwicklungsbord

Das Mikrocontroller-Entwicklungsbord steuert die Elektronik zur Partikelmessung und führt eine Analog-Digital-Wandlung der Messwerte aus und wertet diese aus. Außerdem stellt es die Schnittstellen zum Einplatinencomputer und den zusätzlichen Umweltsensoren für Lufttemperatur, Luftfeuchtigkeit und Luftdruck bereit. Des weiteren steuert es die Spannungsversorgung der Elektronik zur Partikelmessung, der Interfaceplatine, des Lüfters und der zusätzlichen Umweltsensoren. Es wird außerdem eine Schnittstelle zum Debuggen und Programmieren des Mikrocontrollers benötigt.

Mindestanforderungen:

- Schnittstelle zur Spannungsversorgungseinheit

- Schnittstelle zur Interfaceplatine
- Schnittstelle zu den zusätzlichen Umweltsensoren
- Schnittstelle zum Einplatinencomputer
- Schnittstelle zum Programmieren und Debuggen des Mikrocontrollers
- Genügend Rechenleistung zur Auswertung und Weiterleitung der Messwerte
- Analog-Digital-Wandler mit ausreichender Auflösung und Abtastrate.

#### Interfaceplatine

Die Interfaceplatine passte die Pegel der Ausgangssignale der Elektronik zur Partikelmessung an den Spannungsbereich des Analog-Digital-Wandlers an und schützt diesen vor Überspannungen und Verpolung.

Mindestanforderungen:

- Schnittstelle zum Mikrocontroller-Entwicklungsbord
- Schnittstelle zur Elektronik zur Partikelmessung

#### Optische Kammer

In der optischen Kammer findet die Messung der Partikel statt. Hier für muss sie mit Luft von Außen durchströmt sein und gleichzeitig darf kein Licht von Außen in sie einfallen. Die Partikel sollen mit Hilfe der Mie-Streuung vermessen werden und dafür die Wellenlängenabhänigkeit und die Abhängigkeit des Streumusters von Verhältnis von Partikelgröße zu Wellenlänge genutzt werden. Hier für muss die optische Kammer so aufgebaut sein, dass die Streustrahlung unter fünf verschiedenen Winkeln und bei zwei Wellenlängen erfasst werden kann und möglichst wenig Licht in ihr ungewollt auf die photosensitiven Elemente reflektiert wird.

Mindestanforderungen:

- Luft durchströmt
- lichtdicht
- Möglichkeit der Aufnahme der Elektronik zur Partikelmessung
- reflexionsarmes Inneres

### Elektronik zur Partikelmessung

Die Elektronik zur Partikelmessung muss in der optischen Kammer Licht zweier Wellenlängen erzeugen und die Streustrahlung unter 5 Winkeln messen können. Die gemessen Streusignale müssen so verstärkt werden, dass sie von der Interfaceplatine weiterverarbeitet werden können.

Mindestanforderungen:

- Schnittstelle zur Interfaceplatine
- passt zu der optischen Kammer
- erzeugt Licht zweier Wellenlängen
- misst Streustrahlung unter 5 Winkel

### zusätzlichen Umweltsensoren

Für die Messung der zusätzlichen Umweltdaten, werden Sensor für Lufttemperatur, Luftdruck und relative Luftfeuchtigkeit benötigt. Diese sollen als fertige Bauteile in die Elektronik zur Partikelmessung integriert werden und die Daten innerhalb der optischen Kammer erfassen. Dabei ist es egal, ob es sich um einzelne oder kombinierte Sensoren verwendet werden. Sie sollen digital mit dem Mikrocontroller-Entwicklungsbord ausgelesen werden.

Mindestanforderungen:

- Schnittstelle zum Mikrocontroller-Entwicklungsbord
- Lufttemperatursensor mit 1/10 °C Auflösung und +/-1 °C Genauigkeit
- Luftdrucksensor mit 10 Pa Auflösung und +/- 1 kPa Genauigkeit
- Sensor für relative Luftfeuchtigkeit mit 1% rel. LF Auflösung und +/- 5 % rel. LF Genauigkeit
- in optische Kammer passend

#### Lüfter

Um eine Ansprechzeit von unter 30 Sekunden zu gewährleisten, wird ein Lüfter zum Austausch der Luft in der optischen Kammer und aus der Umgebung benutzt. Dafür muss dieser mit der optischen Kammer und einer Gehäuseöffnung verbunden sein.

Mindestanforderungen:

- genügend Leistung für ausreichenden Luftaustausch

### 4.2.3. Gehäuse

Dem Grundkonzept entsprechend soll das Gehäuse aus zwei Teilen bestehen, damit die sensitive Einheit auch einzeln als Sensor an anderen Datenloggern betrieben kann. Beide Teile müssen über eine feste Verbindung, aber auch wieder lösbare, mechanisch verbunden werden können. Die Signal- und Stromleitungen soll über einen gemeinsamen Steckverbinder von einem Teil in den anderen geführt werden. Das Gesamtgerät soll ohne Verpackung Transportiert werden können, dabei muss es und die inneren Befestigungen der anderen Komponenten so stabil sein, dass es dabei auftretende Erschütterungen übersteht. Die Außenhülle muss weiterhin Druck von Außen durch zum Beispiel andere transportierte Gegenstände widerstehen können. Das Gehäuse muss auf Grund der optischen Messmethode lichtundurchlässig sein. Für den Einsatz im Freien ergeben sich noch die Anforderung der Dichtigkeit gegen über Niederschlägen, hier soll auch der Schutz bei starkem Seitenwind umgesetzt werden, und der Schutz vor Insekten, damit diese nicht durch die Luftöffnungen in die Messeinheit eindringen können. Zur guten Handhabung soll es in einer Hand tragbar sein.

Mindestanforderungen:

- besteht aus 2 Teilen
- Teile können fest mechanisch verbunden werden und einfach wieder getrennt werden
- Steckverbinder zur Durchführung der Signal- und Stromleitungen zwischen den Teilen
- feste Montage der anderen Komponenten im Inneren möglich
- Außenhülle robust für unverpackten Transport
- lichtundurchlässig
- kein Eindringen von Niederschläge auch bei starkem Wind möglich
- Luftöffnungen geschützt vor Eindringen von Insekten

- mit einer Hand tragbar

# 5. Realisierung

## 5.1. Komponentenauswahl

In diesem Abschnitt werden die zuvor definierten Komponenten ausgewählt.

### 5.1.1. Einplatinencomputer

Als Einplatinencomputer wird ein *Raspberry Pi B+* ausgewählt. Dieser ist ein weit verbreiteter Einplatinencomputer mit guter Dokumentation. Der *BCM2835-Chip*, ein *System on a Chip*, bildet die Basis des *Raspberry Pi B+*. Er enthält einen ARM1176JZFS Prozessor mit einer Taktrate von 700 MHz und einen Arbeitsspeicher von 512 MB. Die Rechenleistung ist mehr als ausreichend um die Messdaten abzurufen und zu speichern und dabei noch eine externe Schnittstelle zum Zugriff auf die gespeicherten Daten bereitzustellen. Ein weiterer Grund für die Auswahl ist, dass schon in dem bestehenden Datenlogger des *Urban Mobility Labs* dieser Computer genutzt wird und somit eine gute Kompatibilität gewährleistet wird und Erfahrungswerte im Umgang mit ihm bestehen.[Wegner (2015)]
Von den Schnittstellen, diese sind in Tabelle 5.1 aufgelistet, werden der I2C-Bus 1 für die Kommunikation mit dem Mikrocontroller-Entwicklungsbord und die LAN-Verbindung zum Zugriff auf die gespeicherten Messdaten genutzt. Außerdem wird ein GPIO-Pin als Interrupt-Quelle verwendet, an den das Signal "neue Messdaten bereit"des Mikrocontroller-Entwicklungsbords angeschlossen ist.

Als Betriebssystem wird *Raspbian*, eine Open-Source-Linux-Distribution speziell für den *Raspberry Pi* angepasst, in der Version *Jessie Lite* vom 27.05.2016 verwendet. Es basiert auf der Debian GNU/Linux-Distribution, welche sehr weit verbreitet ist.

Tabelle 5.1.: Übersicht Schnittstellen des Raspberry Pi B+

| Schnittstelle | Ausführung | Anzahl | Bemerkung |
|---|---|---|---|
| USB 2.0 | USB A-Buchse | 4 | |
| 10/100-Mbit-Ethernet | RJ45 Buchse | 1 | Zugriff auf gespeicherte Daten |
| GPIO 3,3 V | Stiftleiste 2,54, 2x20 | 26 | einer als Interrupt-Quelle für neue Messdaten |
| SPI | Stiftleiste 2,54 | 1 | belegt 3 GPIO-Pins |
| UART (3,3 V) | Stiftleiste 2,54 | 1 | belegt 2 GPIO-Pins |
| I2C | Stiftleiste 2,54 | 1 | belegt 2 GPIO-Pins, Kommunikation mit dem Mikrocontroller-Entwicklungsbord |
| HDMI | HDMI-Buchse | 1 | |
| DSI | ZIF 15 | 1 | |
| CSI-2 | ZIF 15 | 1 | |

### 5.1.2. Wechseldatenträger

Es sollen 10 Messtage an Daten auf dem Datenträger gespeicht werden können, bevor er geleert werden muss. Die maximale Datenmenge ergibt sich bei der höchsten Messrate, 1 Hz. Die Zusammensetzung eines Datensatzes ist in Tabelle 5.2 aufgeführt.

Tabelle 5.2.: Zusammensetzung eines Datensatzes

| Datentype | Größe in Byte | Anzahl | Summe in Byte |
|---|---|---|---|
| Partikelmessung | 4 | 10 | 80 |
| Luftdruck | 3 | 1 | 3 |
| Temperatur Mikrocontroller | 4 | 1 | 4 |
| Lufttemperatur | 3 | 1 | 3 |
| optionale ADC Messwerte | 4 | 4 | 16 |
| Summe gesamter Datensatz | | | 106 |

Daraus ergeben sich für 24 Stunden Messzeit bei 1 Hz Messrate eine Datengröße von:

$$24h \cdot 106Byte \cdot 1Hz = 24 \cdot 60 \cdot 60s \cdot 106Byte \cdot 1\frac{1}{s}$$
$$= 9158400Byte = 9,1584MByte$$

Für 10 Messtage ergeben sich das etwa 92 MByte Rohdaten, selbst bei einer ASCII-codierte Speicherung, die zum Beispiel das 10-fache an Speicher belegen könnte, ist es noch unter einem Gigabyte in 10 Tagen.

Es wird für das Betriebssystem des Raspberry Pis eine Micro-SD-Card benötigt, daher bietet es sich an, diese auch zur Speicherung der Messdaten zu nutzen. Das ausgewählte Betriebssystem belegt etwa 2 GByte auf der SD-Card des Raspberry Pis, somit reicht für die Anforderung eine 4 GByte *Micro-SD-Card*.
Die Datenrate ist selbst bei ASCII-Codierung mit zirka 1060 Bytes/Sekunde so gering, dass sie für die Auswahl der SD-Card keine Rolle spielt.

### 5.1.3. Spannungsversorgungseinheit

Spannungsversorgungseinheit muss die in Tabelle 5.3 auf geführten Komponenten versorgen und schalten.

Tabelle 5.3.: Übersicht der zu erzeugenden und schaltenden Spannungen und Ströme

| Kompnente | Schalten | Spannungen [V] | Stromaufnahme* [mA] |
|---|---|---|---|
| Einplatinencomputer mit Wechseldatenträger | nein | 5 | 250 |
| Mikrocontroller-Entwicklungsbord | nein | 5 | 20 |
| Interfaceplatine | ja | +/-12 | 25 |
| Elektronik zur Partikelmessung | ja, 3 separate Verbraucher | 2x 5; +/-12 | 2x 20; 50 |
| zusätzlichen Umweltsensoren | ja | 1,8; 2,5 | < 1; < 1 |
| Lüfter | ja | 5 | 180 |

* alle Ströme sind Abschätzungen aus Datenblättern und früheren Messungen

Da es keine Fertigbaugruppe für diese spezielle Anwendung gibt, wird die Spannungsversorgungseinheit selbst entworfen. Dieses wird im Kapitel 5.2.1 *Schaltungsentwicklung / Spannungsversorgungseinheit* ( S. 34 ) beschrieben.

### 5.1.4. Akkumulator

Als Akkumulator wird eine *EasyAcc Brilliant 15000 mAh Power Bank* gewählt. Dieser erfüllt die Anforderungen in das gewählte Gehäuse zu passen und ohne Ausbau ladbar zu sein.

Außerdem besitzt er schon eine Ladeschaltung und 5 V Ausgangsspannung, die hauptsächlich benötigt wird.
Aus den Schätzungen in Tabelle 5.2.1 ergibt sich eine Leistungsaufnahme von etwa 4,5 W, daraus werden in 24 h 112,5 Wh Energie, der Akkumulator speichert allerdings nur 15 Ah x 3,7 V = 55,5 Wh. Er wird trotzdem ausgewählt, da es keine Alternativen mit mehr Kapazität in diesem Bauformat gibt. [EasyAcc (2016)]

### 5.1.5. Mikrocontroller-Entwicklungsbord

Es wird das *Tiva C Series TM4C123G LaunchPad Evaluation Board* vom Texas Instruments ausgewählt. Das Kernstück bildet ein Texas Instruments *TM4C123GH6PMI* Mikrocontroller, ein ARM Cortex-M4 Prozessor mit bis zu 80 MHz Prozessortakt. Dieser erfüllt mit seinen zwei ADCs die Anforderung die Messwerte analog-digital-wandeln zu können. Diese lösen mit 12 Bit auf, tasten mit bis zu einem Megaherz Rate ab und besitzen eine interne Spannungsreferenz von 3 V. Die sonstigen Schnittstellenvorgaben werden auch erfüllt und er bietet genügend Rechenleistung zum Auswerten der Messwerte. Außerdem besitzt es eine USB-Schnittstelle zum Programmieren und Debuggen und die integriere Entwicklungsumgebung *Code Composer Studio 6* für Windows Pc ist vertraut, so dass keine lange Einarbeitung nötig ist. Das Bord ist kompakt, sodass es gut in das Gehäuse passt.

### 5.1.6. Interfaceplatine

Die Interfaceplatine muss die Ausgangspegel der Elektronik zur Partikelmessung an die 3 V der ADCs des Mikrocontroller-Entwicklungsbord anpassen und diese vor Überspannungen schützen. Da es keine Fertigbaugruppe für diese spezielle Anwendung gibt, wird die Spannungsversorgungseinheit selbst entworfen. Dieses wird im Kapitel 5.2.2 *Schaltungsentwicklung / Interfaceplatine* ( S. 38 ) beschrieben.

### 5.1.7. optische Kammer

Die optische Kammer ist eine Eigenentwicklung. Diese wird in Kapitel 5.2.2 ( S. 38 ) beschrieben.

### 5.1.8. Elektronik zur Partikelmessung

Die Elektronik zur Partikelmessung ist eine Eigenentwicklung. Diese wird in Kapitel 5.2.3 *Schaltungsentwicklung / Elektronik zur Partikelmessung* ( S. 41 ) beschrieben.

### 5.1.9. zusätzlichen Umweltsensoren

Als Lufttemperatur- und relative Luftfeuchtigkeitssensor wird der *SHTC1* von Sensirion gewählt, als Luftdrucksensor das *Xtrinsic MPL3115A2 I2C Precision Altimeter* von Freescale Semiconductor. Beide Sensoren kommunizieren mit dem Mikrocontroller-Entwicklungsbord über einen I2C-Bus. In Tabelle 5.4 sind die Mindestanforderungen und Datenblattangaben gegenübergestellt.

Tabelle 5.4.: Übersicht der Mindestanforderungen und Datenblattangaben der zusätzlichen Umweltsensoren

| | Mindestanforderung | Datenblatt |
|---|---|---|
| Messfrequenz Lufttemperatur* | 1 Hz | 69 Hz |
| Messfrequenz relative Luftfeuchtigkeit* | 1 Hz | 69 Hz |
| Messfrequenz Luftdruck** | 1 Hz | 100 Hz |
| Auflösung Lufttemperatur* | 1/10 °C | 1/100 °C |
| Auflösung relative Luftfeuchtigkeit* | 1 % rel. LF | 1/100 % rel. LF |
| Auflösung Luftdruck** | 10 Pa | 1,5 Pa |
| Genauigkeit Lufttemperatur* | +/- 1 °C | +/- 0,5 °C |
| Genauigkeit relative Luftfeuchtigkeit* | +/- 5 % rel. LF | +/- 4,5 % rel. LF |
| Genauigkeit Luftdruck** | +/-1 kPa | +/-0,4 kPa*** |

* Sensirion SHTC1 [SHTC1 (2013)]

** Xtrinsic MPL3115A2 I2C Precision Altimeter [MPL3115A2 (2013)]

*** nur für -10 °C bis 70 °C spezifiziert

### 5.1.10. Lüfter

Es wird der *Laptop-CPU-Kühler UDQFYZH07C* als Lüfter ausgewählt. Es ist ein CPU-Kühler mit integriertem Lüfter für den Toshiba Tecra 8200 Laptop. Er ist sehr flach und wenn der Kühlkörperteil abgeschnitten ist, passt er sehr gut in das Gehäuse. Für einseitiges Ansaugen

von Luft werden die Einlässe auf der Rückseite abgeklebt, wie in Abbildung 5.1 zusehen. Der Lüfter benötigt nach Händlerangaben 180 mA Strom bei 5 V Betriebsspannung. [Pollin-1 (2016)]

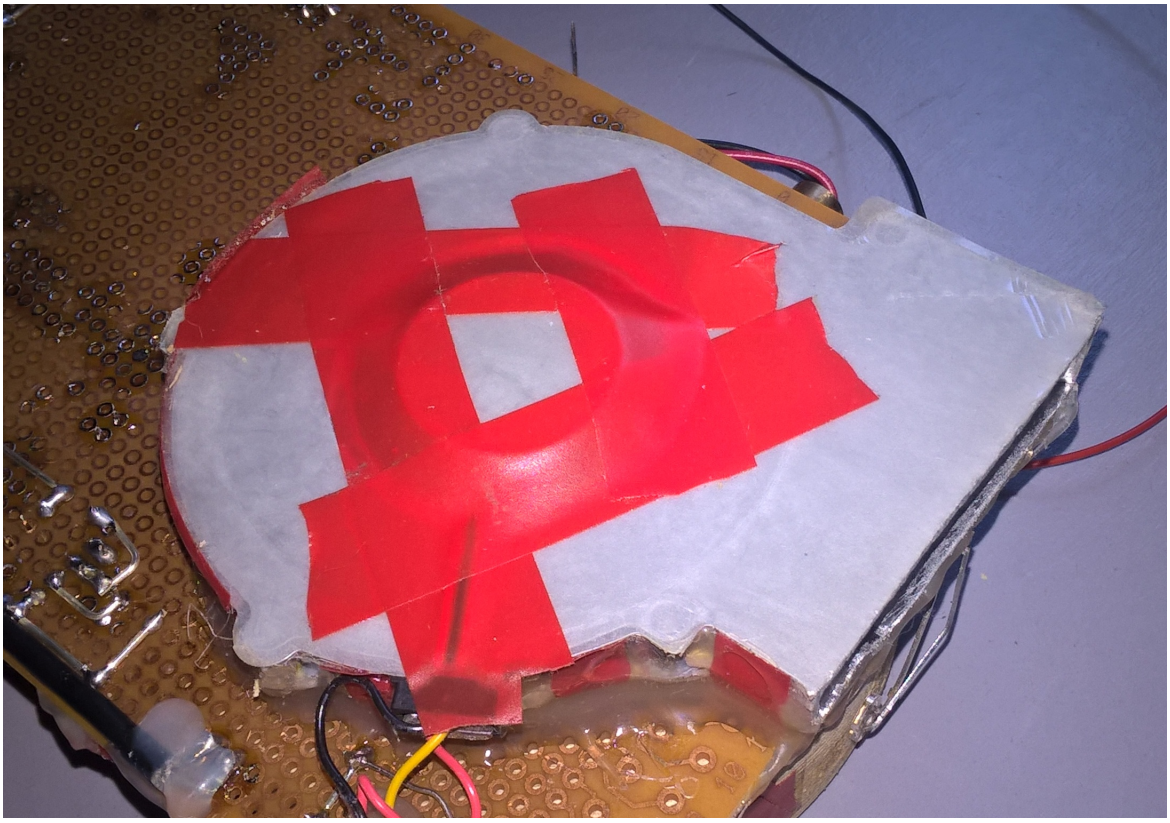

Abbildung 5.1.: Gekürzter Lüfter mit abgeklebter Rückseite auf Testaufbau

### 5.1.11. Gehäuse

Als Basis für das Gehäuse werden zwei ABS-Kunststoffgehäuse vom Typ NBF-32006 von Bud Industries verwendet. Dieses Gehäuse ist IP66 klassifizierte, staubdicht und geschützt gegen starkes Strahlwasser. Es besitzt einen Klappdeckel mit 2 Riegeln, was das Öffnen für Arbeiten im Inneren, zum Beispiel Laden des Akkumulators oder Auslesen der Messdaten, erleichtert. Das Gehäuse kann auch über zwei Laschen fest verschlossen werden, wenn es für den Außeneinsatz nötig ist.

Die *Energieversorgung und Datenspeicherung* sind in einem und die *sensitive Einheit* ist in dem anderen Gehäuse montiert. Abbildung 5.3 zeigt die Positionierungen der Baugruppen

im Gehäuse.

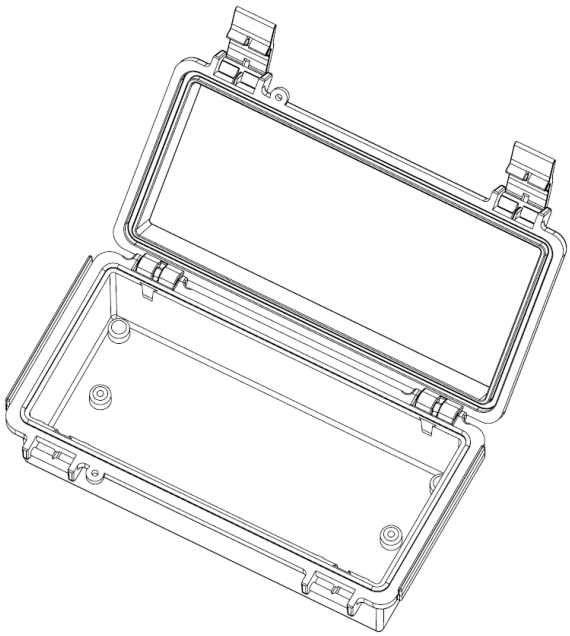

Abbildung 5.2.: Zeichnung ABS-Kunststoffgehäuse vom Typ NBF-32006 von Bud Industries, aus Datenblatt [Bud-Industries] entnommen

Um das Eindringen von Insekten in das Gehäuse zu verhindern, werden die Luftöffnungen von innen mit engmaschigem Textilgitter versehen. Als Spritzwasserschutz sind Wannen innen an den Luftöffnungen angesetzt.
Die Teilgehäuse sind mit Schrauben an den 4 Ecken Boden an Boden verbunden. Signal- und Versorgungsleitung werden über eine 40 Pol Steckverbindung zwischen den Gehäusen geführt. Als Steckverbinder wird ein Stiftleisten Stecker-Buchsen-Paar verwendet, in der Ausführung 2 x 20 Kontakte und 2,54 mm Rastermaß. Die Belegung ist in Tabelle 5.5 aufgeführt.

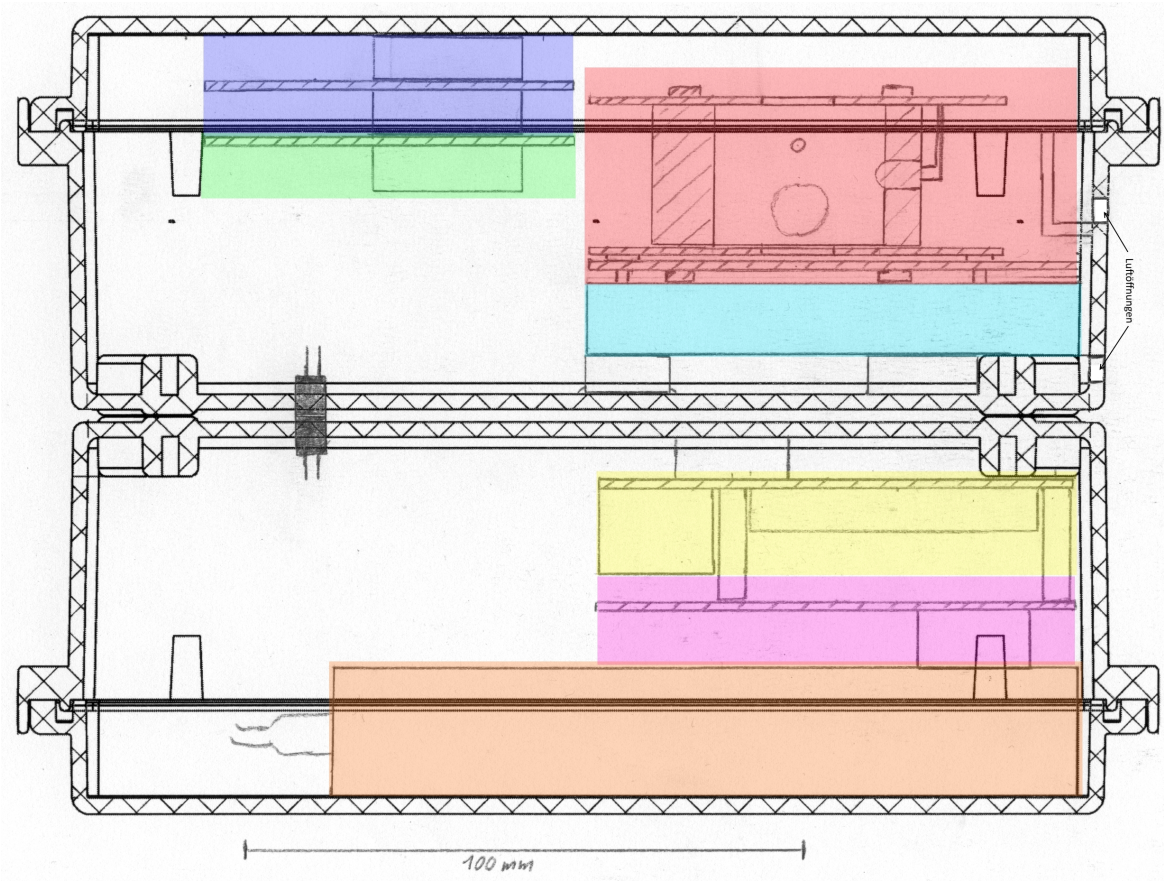


Abbildung 5.3.: Schnittzeichnung: Gehäuse mit eingesetzten Komponenten; blau = Mikrocontroller-Entwicklungsbord, grün = Interfaceplatine, rot = optische Kammer und Elektronik zur Partikelmessung, türkis = Lüfter, gelb = Einplatinencomputer, lila = Spannungsversorgungseinheit, orange = Akkumulator

Tabelle 5.5.: Pinbelegung Gehäusesteckverbindung

| Pin | Signal | Spannungsverteilungs Ausgang |
|---|---|---|
| 1 | GND | PowerOut |
| 2 | +5 V | |
| 3 | GND | Out1 |
| 4 | +3,3 V | |
| 5 | GND | |
| 6 | +5 V | |
| 7 | GND | Out2 |
| 8 | +1,8 V | |
| 9 | GND | |
| 10 | +2,5 V | |
| 11 | GND | |
| 12 | +5 V | |
| 13 | GND | Fan |
| 14 | +5 V | |
| 15 | GND | Laser1 |
| 16 | +5 V | |
| 17 | GND | Laser2 |
| 18 | +5 V | |
| 19 | GND | Sensor |
| 20 | -12 V | |
| 21 | GND | |
| 22 | +12 V | |
| 23 - 31 | nicht belegt | |
| | | **ab hier Signalleitungen** |
| 32 | IRQ - New Data | Mikrocontroller -> Raspberry Pi |
| 33 | Sensor | Mikrocontroller -> Spannungsverteilung |
| 34 | Out1 | |
| 35 | Out2 | |
| 36 | Fan | |
| 37 | Laser1 | |
| 38 | Laser2 | |
| 39 | I2C SCL | Mikrocontroller <-> Raspberry Pi |
| 40 | I2C SDA | |

## 5.2. Schaltungsentwicklung

Alle Schaltpläne und Layout wurden mit der Software Eagle, Version 7.6, erstellt.

### 5.2.1. Spannungsversorgungseinheit

Die Aufgaben der Spannungsversorgungseinheit sind:

- Verteilung der Versorgungsspannung von Akkumulator zu den Verbrauchern
- Schutz der Schaltung vor verpolter Versorgungsspannung
- Filterung von Störungen, die durch die Verbraucher erzeugt werden
- Filterung der Versorgungsspannung
- Schalten der Verbraucher durch externen Input

Der Schaltplan ist in Abbildung 5.4 dargestellt und die zugehörige Stückliste findet sich in Anhang A.1.
Als Verpolungsschutz dient der P-Kanal MOSFET *Q1* vom Typ FDN306P. Dieser ist so eingebaut, dass seine Source-Drain-Strecke nur aufschaltet, wenn die Spannungsquelle richtig gepolt ist, da nur dann eine negative Spannung an seinem Gate-Eingang gegenüber dem Source-Eingang anliegt. Dieser Transistor ist für Spannungen Drain-Source von -12 V und Gate-Source von +/-8 V spezifiziert [Fairchild]. In diesem Gerät wird nur eine 5 V Spannungsversorgung genutzt, somit erfüllt der Transistor die Aufgabe.
Die Filterung der Versorgungsspannung erfolgt mit den Kondensatoren *C1* und *C2*. Es wurde ein Paar aus Elektrolytkondensator, mit hoher Kapazität zur Spannungsstützung, und einem Keramik-SMD-Kondensator, zum Filtern von höheren Frequenzen, gewählt.
Es ist ein Anschluss JP2 für einen Schalter für die Versorgungsspannung vorgesehen, dieser ist im Prototypen mit 2 Jumper-Brücken bestückt. Außerdem wird die gefilterte Versorgungsspannung an den Anschlüssen JP9 und JP10 bereitgestellt.
Die schaltbare Verteilung erfolgt über die Anschlüsse *JP3* - *JP5*, *JP7*, *JP8* und *JP11*. Die Schaltsignale werden über den Anschluss *JP6* auf die Platine geführt. Im ersten Entwurf wurde bei allen Ausgängen die Masseleitung geschaltet, dieses erfolgt mit 2 parallel geschalteten NPN-Transistoren, dabei schaltet ein hoher Pegel am Eingang den Ausgang ein. Die Parallelschaltung soll den Spannungsabfall über den Transistoren verringern. Da aber an den Ausgängen *JP3*, *JP4* und *JP11* ein fester Massbezug benötigt wird, wurde hier die Schaltung geändert. Jetzt wird die positive Zuleitung geschaltet, dafür wird der gleiche P-Kanal-MOSFET genutzt wie für den Verpolungsschutz. Die Schaltung arbeitet so, dass der Ausgang nur beim aktiven auf Masse Ziehens des Eingangs aktiviert wird. Das wird mit

einem Pull-Up-Widerstand gegen die Versorgungsspannung bewirkt. Dieses hat auch den Vorteil, dass die Ausgänge bei nicht Beschalten des Einganges sicher abgeschaltet sind.
Die Ausgänge *OUT1* und *OUT2* sind so ausgelegt, dass sie mit einem beziehungsweise zwei linearen Spannungsreglern, der Bauform SOT23-5, bestückt werden können, um zusätzliche Spannungen zu liefern. Im Prototypen ist in *OUT1* ein 3.3 V und in *OUT2* ein 1,8 V und ein 2,5 V Regler, *IC1* - *IC3*, verbaut.
In *OUT Sensor* wird mit einem DC-DC-Wandler eine symmetrische +/-12 V Spannung erzeugt. Es wurde ein Recom 2 Watt Version gewählt, da diese einen maximalen Ausgangsstrom von 84 mA liefert und in Tabelle 5.3 für diese Spannung ein Strom von 75 mA abgeschätzt worden ist. Die Ausgänge *OUT3 FAN*, *OUT4 Laser1* und *OUT5 Laser2* werden besonders mit einem L-C-Glied gefiltert, da dort im Fall von *OUT4 Laser1* und *OUT5 Laser2* mit einer Frequenz von 5 kHz Leuchtdioden geschaltet werden, wobei bis zu 50 mA Strom geschaltet wird. Im Fall von *OUT3 FAN* zeugt der Gleichstrommotor des angeschlossenem Lüfters bei jedem Umpolen seiner Wicklung einen starken Strompuls. Dieser ist besonders schädlich für die analoge Signalverarbeitung, da er nicht zu festen Zeitpunkten innerhalb eines Messzykluses auftritt und somit beim Digitalisieren als Störung mit gemessen wird, wenn er in falschen Moment auftritt. Bei *OUT4 Laser1* und *OUT5 Laser2* ist dieses unkritischer, da durch die Ansteuerung sichergestellt ist, dass nie kurz nach einem Schaltvorgang gemessen wird.

Als Grundform für das Layout, wurde der Formfaktor des Raspberry Pis gewählt, da beides in dieselbe Gehäusehälfte gebaut wird und es so stapelbar wird. Beim Layout wurde darauf geachtet, dass keine Masseflächen mehr zwischen Filter und Ausgangsanschluss liegen um Rückwirkungen auf die Masse zu vermeiden. Des Weiteren wurden die Ausgänge so weit wie es die Abmasse der Platine ermöglichen auseinander gelegt. In Abbildung 5.5 ist das fertige Layout zu sehen.

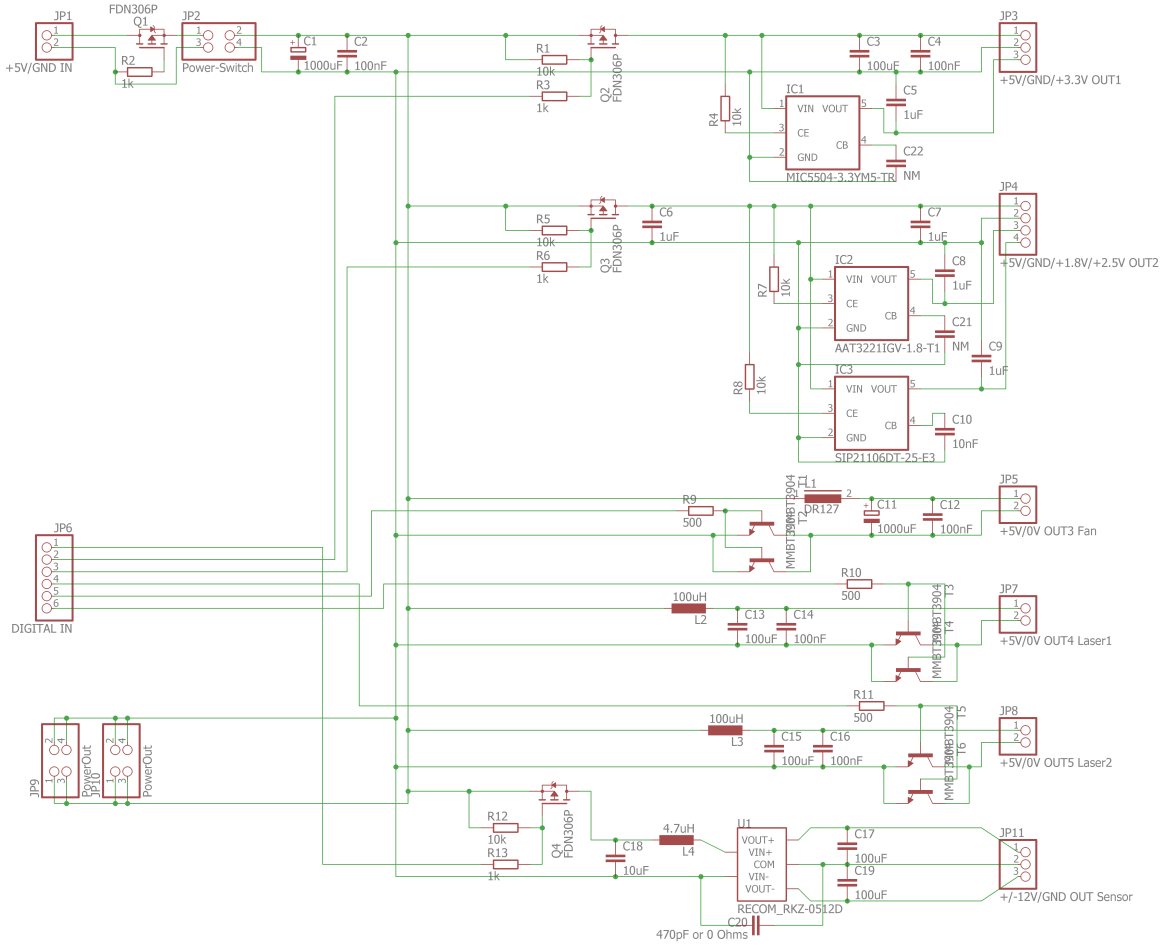


Abbildung 5.4.: Schaltplan der Spannungsversorgungseinheit

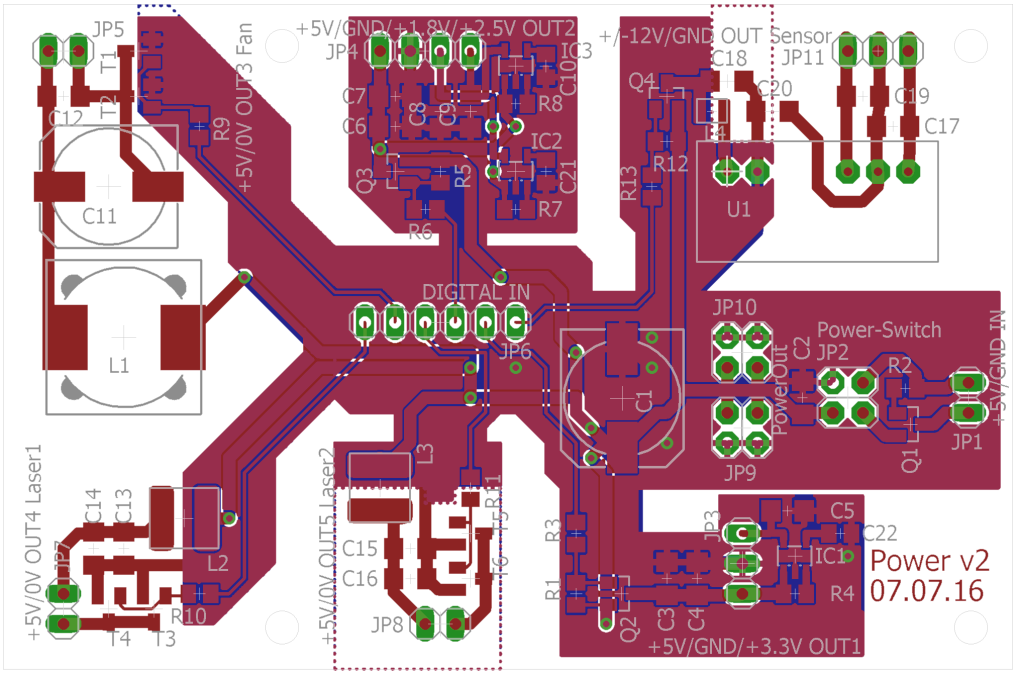


Abbildung 5.5.: Layout der Spannungsversorgungseinheit

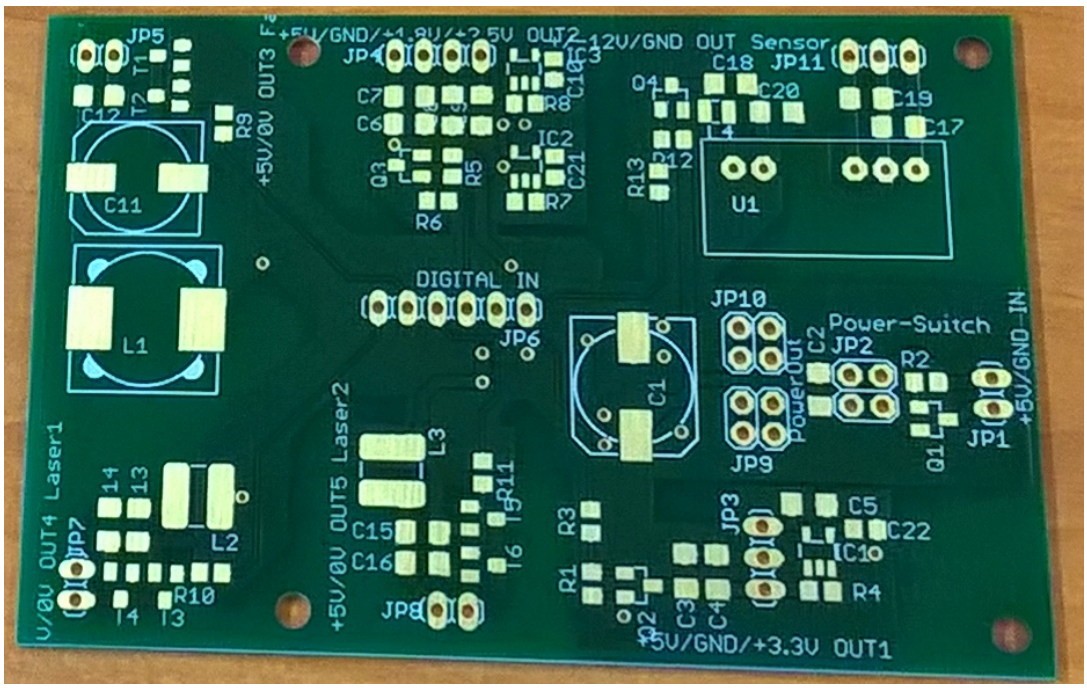


Abbildung 5.6.: unbestückte Platine der Spannungsversorgungseinheit

### 5.2.2. Interfaceplatine

Die Aufgaben der Interfaceplatine sind:

- Anpassung der Ausgangsspannungen der Elektronik zur Partikelmessung an die 3 V Referenzspannung des ADCs des Mikrocontrollers
- Pufferung der Messwerte vor dem ADC des Mikrocontrollers
- Schutz des ADCs des Mikrocontroller vor Über- und Unterspannungen

Der Schaltplan ist in Abbildung 5.7 dargestellt und die zugehörige Stückliste findet sich in Anhang A.2.
Jede Aufgabe wird in einem eigenen Schaltungsblock bearbeitet. Die Anpassung der Pegel erfolgt durch eine invertierende Operationsverstärkerschaltung, deren Verstärkungsfaktor von 0 bis 10 über ein Potentiometer in der Rückkopplung eingestellt werden kann. Die Pufferung und Rückinvertierung erfolgt in einer zweiten invertierenden Operationsverstärkerschaltung mit fest eingestelltem Verstärkungsfaktor von 1. Der Schutz des ADCs erfolgt über 2 Schottky Dioden, eine in Sperrrichtung gegen Masse und die andere in Durchlassrichtung gegen die Versorgungsspannung des Mikrocontrollers. Die Diode gegen Masse wird bei negativen Ausgangsspannungen unterhalb der Masse leitend und die gegen die Versorgungsspannung bei Ausgangsspannungen oberhalb der Versorgungsspannung des Mikrocontrollers.

Diese dreistufige Schaltung wird für 7 Eingangssignale erstellt, wobei 5 für die Signale der Elektronik zur Partikelmessung und 2 als Reserve für spätere Zusatzfunktionen vorgesehen sind.

Das Layout ist als Aufsteckplatine für das Mikrocontroller-Bord ausgelegt, so dass keine extra Verkabelung nötig wird. Abbildung 5.9 zeigt das fertige Layout und Abblidung 5.8 die produzierte, unbestückte Platine.

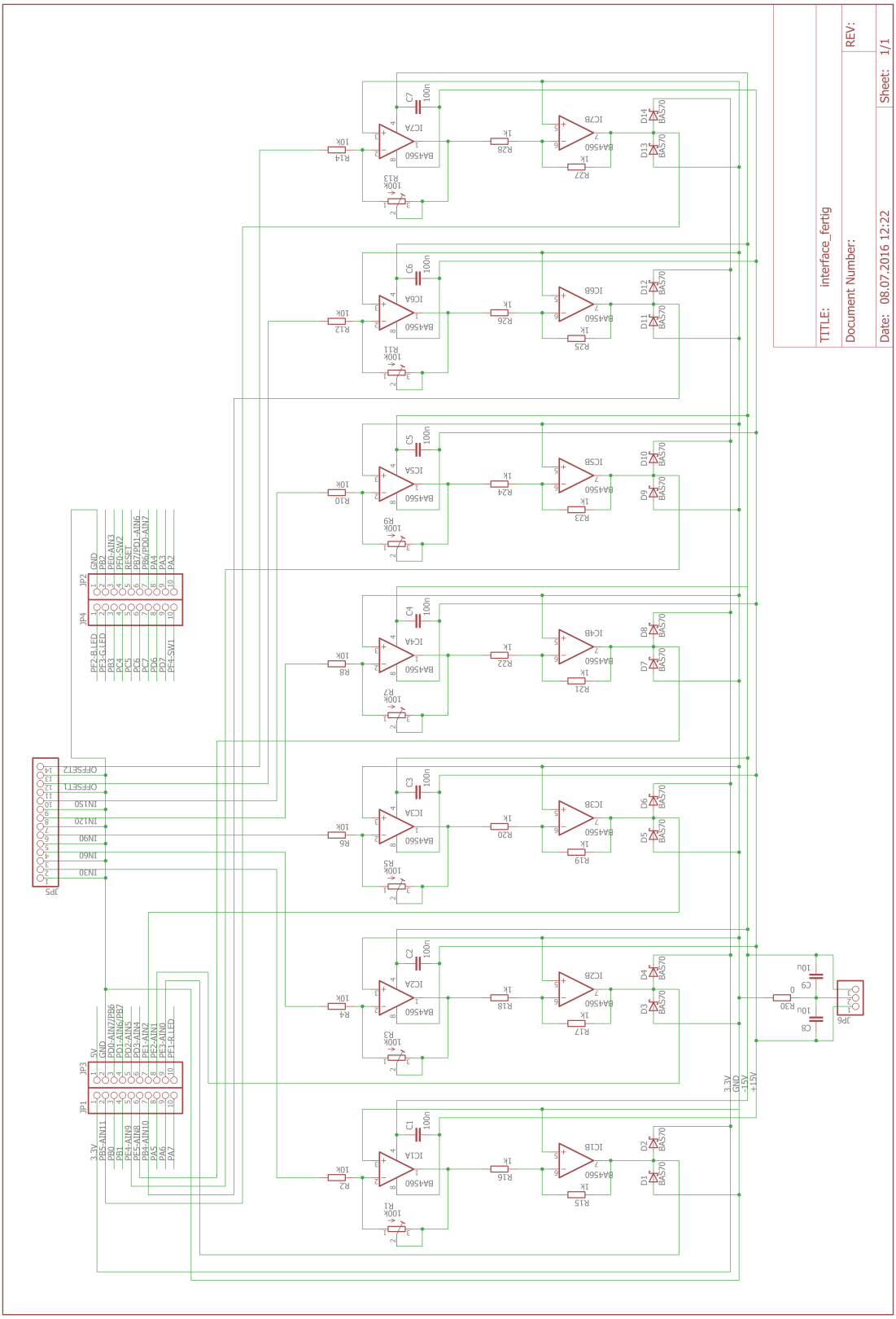


Abbildung 5.7.: Schaltplan der Interfaceplatine

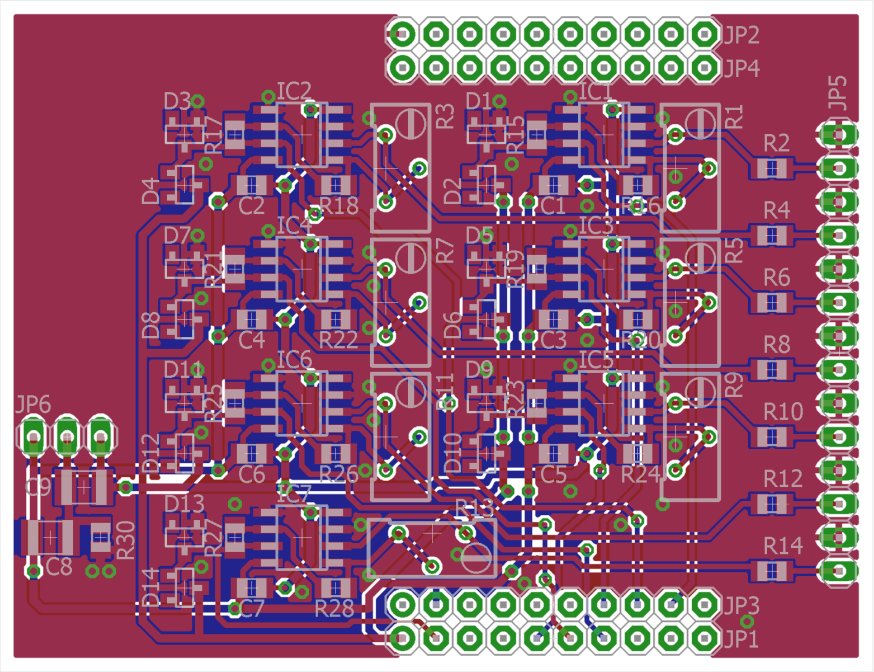


Abbildung 5.8.: Layout der Interfaceplatine

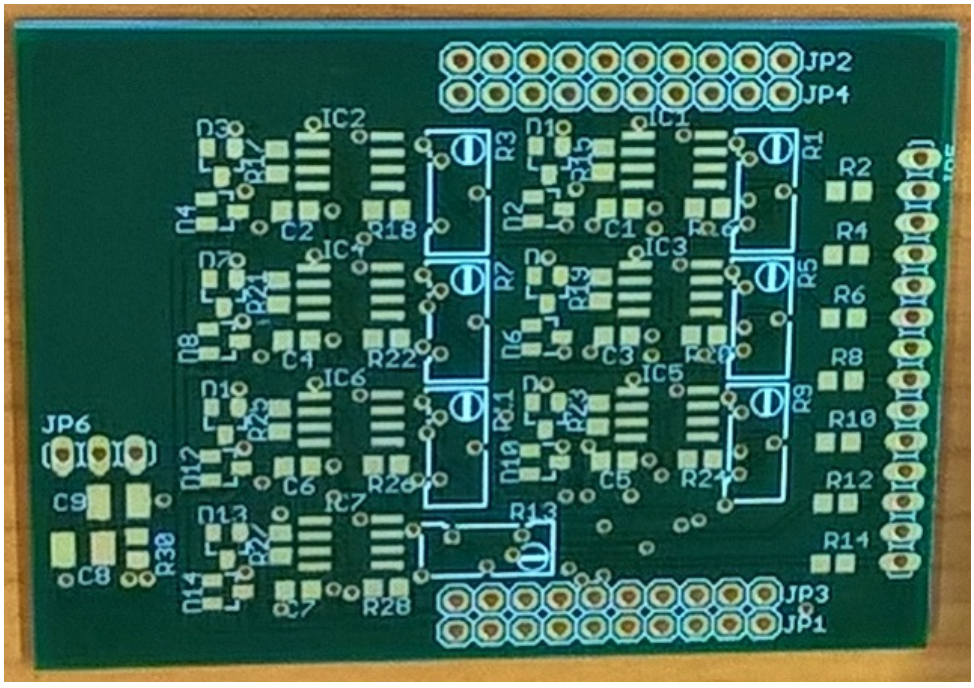


Abbildung 5.9.: unbestückte Interfaceplatine

### 5.2.3. Elektronik zur Partikelmessung

Die Aufgaben der Elektronik zur Partikelmessung sind:

- Erzeugen zweiter zeitlich auf einander folgender Lichtpulse mit je einer Frequenz von 5000kHz und je 1/3 der Periode Länge.
- Auffangen von gestreutem Licht und Umsetzung in eine Spannung
- Träger der zustätzlichen Umweltsensoren

Zur Erzeugung der Lichtimpulse werden 2 Leuchtdioden genutzt. Diese werden direkt von Mikrocontroller über die Spannungsversorgungseinheit geschaltet, so dass hierfür keine zusätzlich Schaltung nötig ist. Damit ein möglichst großer Unterschied in der Wellenlänge der beiden Lichtpulse besteht, wird eine blau und eine infrarot strahlende Leuchtdiode verwendet.
Die beiden anderen Aufgaben werden auf 2 Platinen verteilt, damit die digitalen Signale der zusätzlichen Umweltsensoren nicht die teilweise in Nanoampere liegenden Ströme der optischen Sensoren stören.

Der Schaltplan der Platine für die optische Detektion ist in Abbildung 5.10 dargestellt und die zugehörige Stückliste findet sich in Anhang A.3. Diese Platine wird im folgenden *Sensor Top* genannt.
Die Schaltung der optischen Detektion kann in 3 Stufen unterteilt werden. Der erster Abschnitt besteht aus einem Phototransistor und einem normalen NPN-Transistor, welche zu einer Darlington-Schaltung kombiniert werden. Dieses ist erforderlich, da sehr kleine Lichtintensitäten erwartet werden. Diese Schaltung liefert das Lichtintensität abhänge Stromsignal, dass in der nächsten Stufe in ein Spannungssignal umgewandelt wird.
Die zweite Stufe ist eine Strom zu Spannung Operationsverstärkerschaltung. Sie ähnelt dem invertierendem Verstärker, in Funktion und Aufbau. Die Ausgangsspannung ergibt sich aus Eingangsstrom mal Rückkopplungswiderstand. Es werden im Schaltplan zwei Widerstände in Reihe vorgesehen, damit eine einfachere Anpassung der Rückkopplung möglich ist.
Die dritte Stufe ist eine Filterstufe, sie soll Gleichanteile und hochfrequentes Rauschen ausfiltern. Hierfür ist ein Bandpass mit Mittenfrequenz von 5000 Hz und einem Verstärkungsfaktor von 20 vorgesehen. Dieser wird ein einer Multiple-Feedback-Anordnung aufgebaut, die Bauteilwerte wurden mit Hilfe des *Texas Instrument WEBBENCH Design Center* TI (2016) bestimmt. Die Multiple-Feedback-Anordnung gewählt worden, da sie das Signal wieder in den positiven Spannungsbereich zurück invertiert und die Möglichkeit bietet durch Bauteiltausch auch Hoch- und Tiefpässe zu erzeugen.
Diese dreistufige Schaltung wird für alle 5 Winkel unter denen gemessen werden soll aufgebaut. Als Basis für das Layout wird eine runde Scheibe mit einem Durchmesser von 75

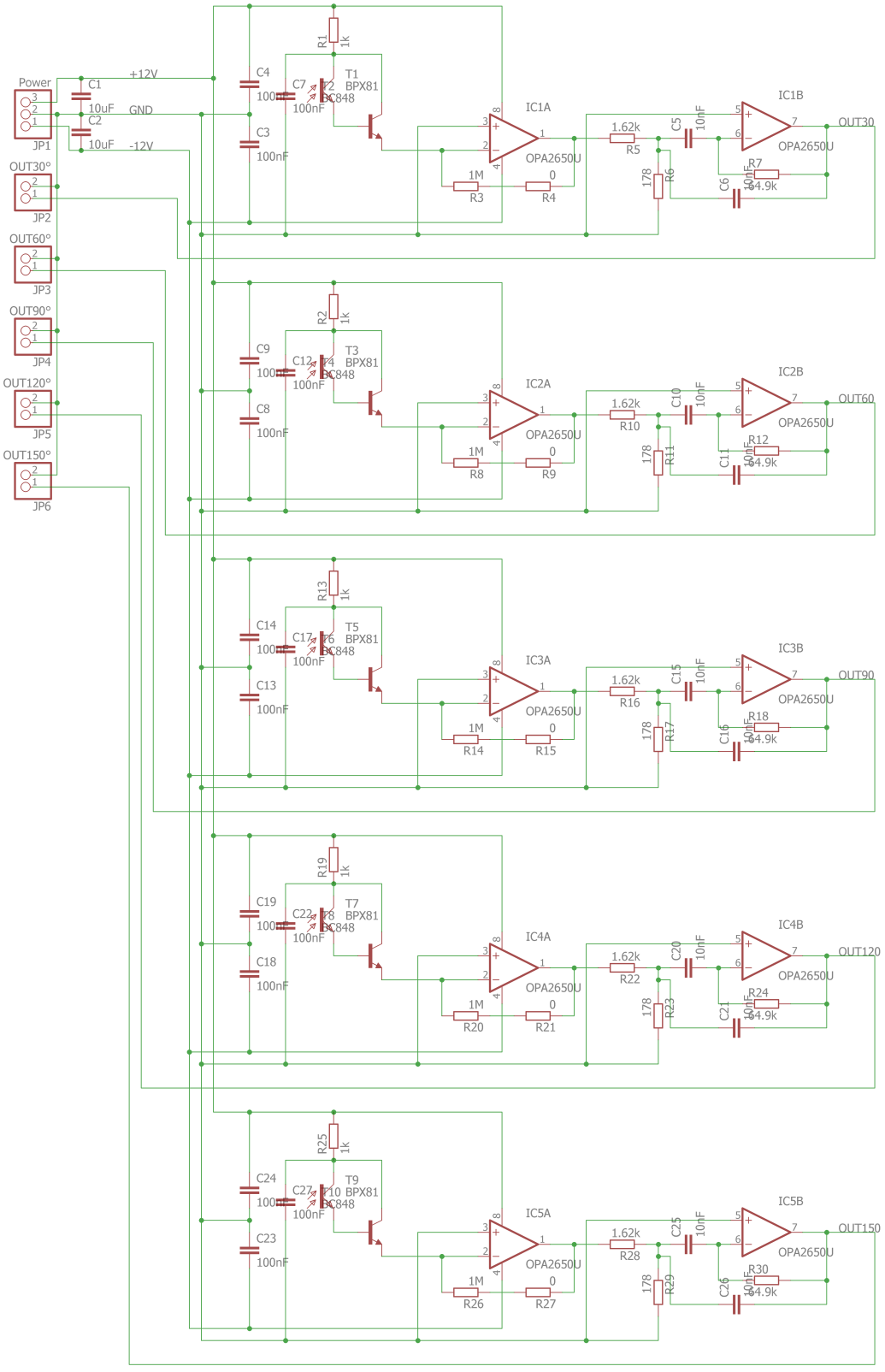


Abbildung 5.10.: Schaltplan der Platine *Sensor Top*

mm und einem 13 mm Loch, als Luftdurchlass, in der Mitte gewählt. Die einzelnen Blöcke werden so auf der Platine platziert, dass die Phototransistoren direkt in die optische Kammer gesteckt werden können. Außerdem wird darauf geachtet, dass die Verbindung vom Phototransistor zum Verstärker möglichst kurz ist. Das Layout ist in Abbildung 5.11 dargestellt und in Abbildung 5.12 ist die produzierte, unbestückte Platine zu sehen

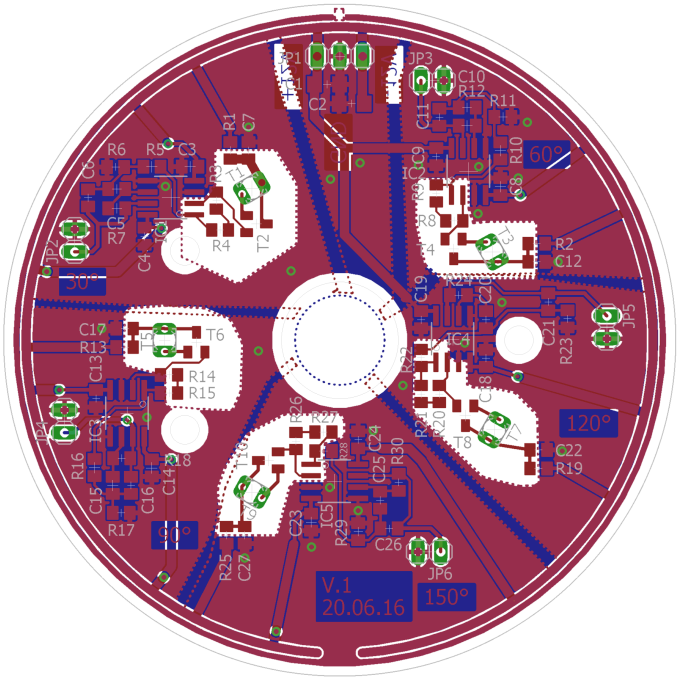


Abbildung 5.11.: Layout der Platine *Sensor Top*

Der Schaltplan der Platine für die zusätzlichen Umweltsensoren ist in Abbildung 5.13 dargestellt und die zugehörige Stückliste findet sich in Anhang A.4. Diese Platine wird im folgenden *Sensor Bottom* genannt.

Es wird die selbe Form der Platine wie für Sensor Top benutzt, so können beide zusammen als Deckel und Boden der optischen Kammer dienen. Die Umweltsensoren sind nach Datenblatt beschaltet. Die Pull-Up-Widerstände der Datenleitung und das Taktleitung werden gegen 1,8 V geführt, da beide mit diesem Interfacepegel arbeiten können. Des Weiteren werden noch zwei Schaltungsblöcke zur optischen Detektion auf die Platine gebracht. Sie stimmen mit denen von der Platine Sensor Top bis auf die dritte Stufe über ein, hier ist nur ein invertierender Verstärker vorgesehen. Als vierte Stufe folgt eine Addierschaltung, die beide Detektorsignale zusammenführen kann und dann auf bis zu 5 als Spannungsteiler verwendete Potentiometer gibt. Dieser Teil der Schaltung ist nur für mögliche nachfolge

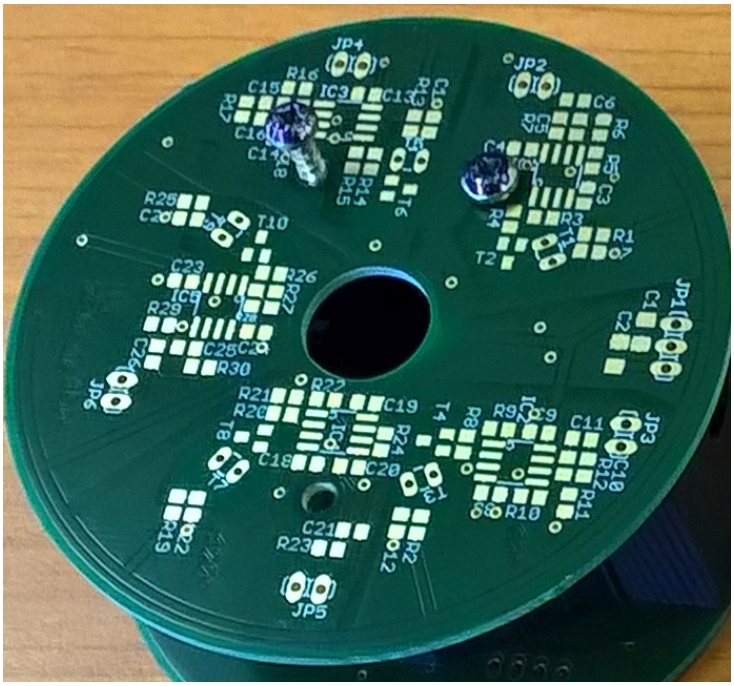


Abbildung 5.12.: unbestückte Platine *Sensor Top*

Versuche integriert und wird noch nicht weiter betrachtet.
Das Layout ist so ausgelegt, dass die analogen und digitalen Anteile jeweils ihre eigene Stromversorgung und Masse haben. Die Umweltsensoren sind so auf der Unterseite platziert, dass sie sich im Inneren der optischen Kammer befinden. Die optischen Detektoren sind in einem Winkel von 180° gegenüber angeordnet, um zum Beispiel ein mal den Direktstrahl der Lichtquellen messen zu können. Das Layout ist in Abbildung 5.14 dargestellt und in Abbildung 5.15 ist die produzierte, unbestückte Platine zu sehen

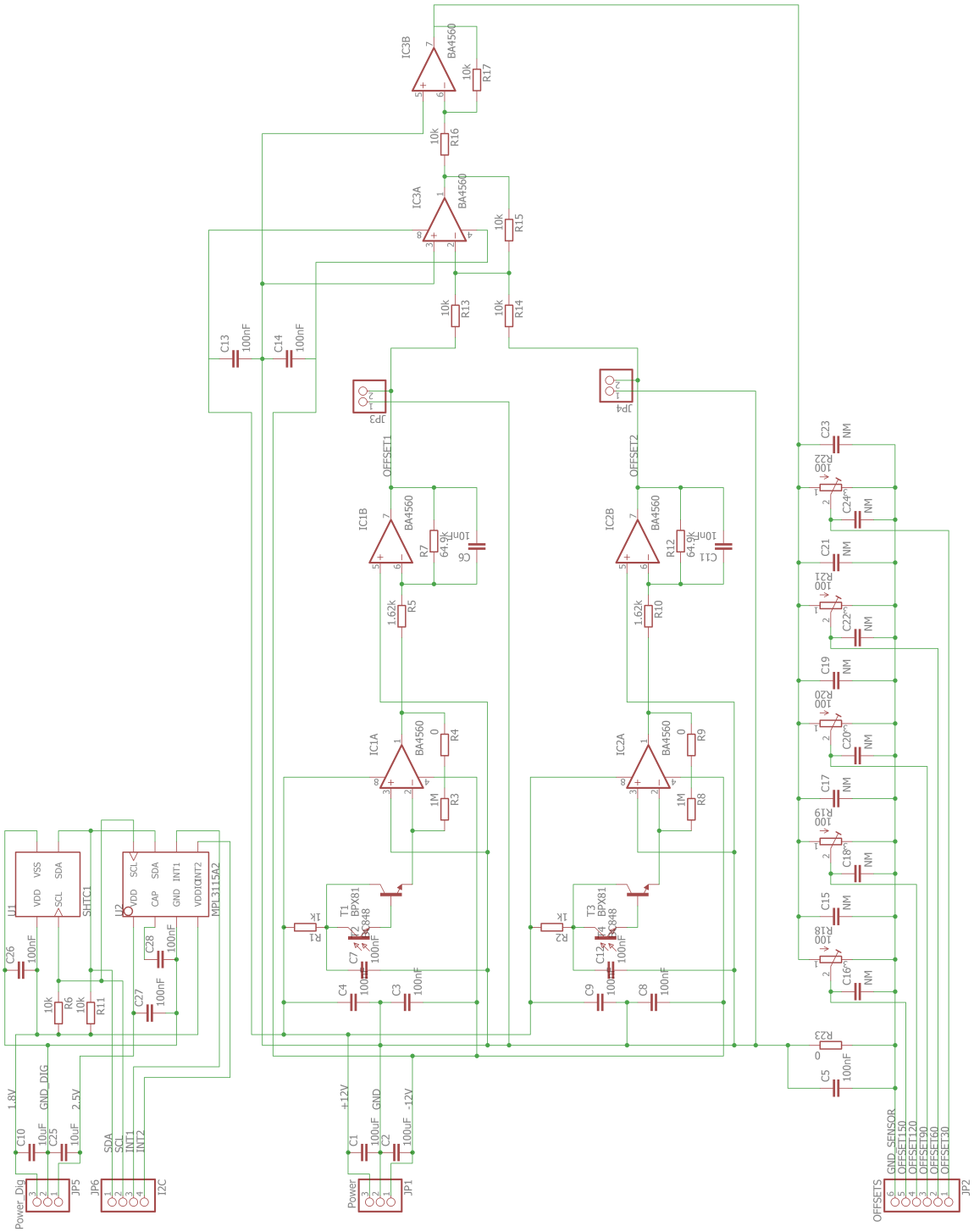


Abbildung 5.13.: Schaltplan der Platine *Sensor Bottom*

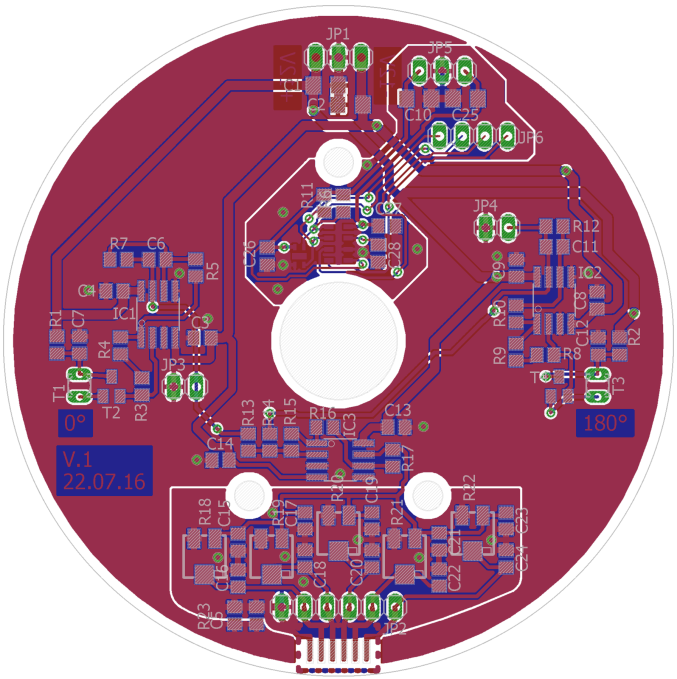


Abbildung 5.14.: Layout der Platine *Sensor Bottom*

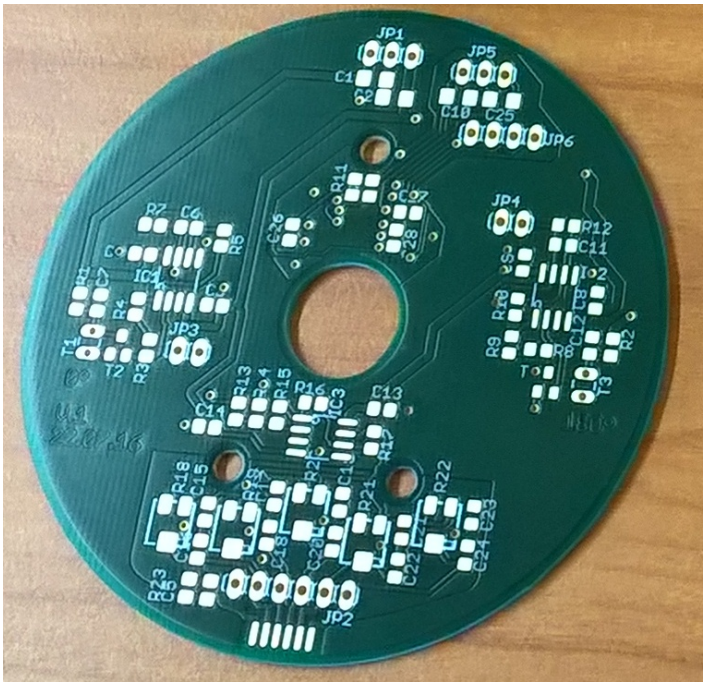


Abbildung 5.15.: unbestückte Platine *Sensor Bottom*

## 5.3. Optische Kammer

Die optische Kammer hat folgende Aufgaben:

- Die mechanische Aufnahme der Elektronik zur Partikelmessung
- Abschirmung des Messraumes vom Umgebungslicht
- Führung des Luftstroms
- Führung der erzeugten Lichtimpulse, so dass sie nicht am Material der Kammer in ungewünschte Richtungen reflektiert werden

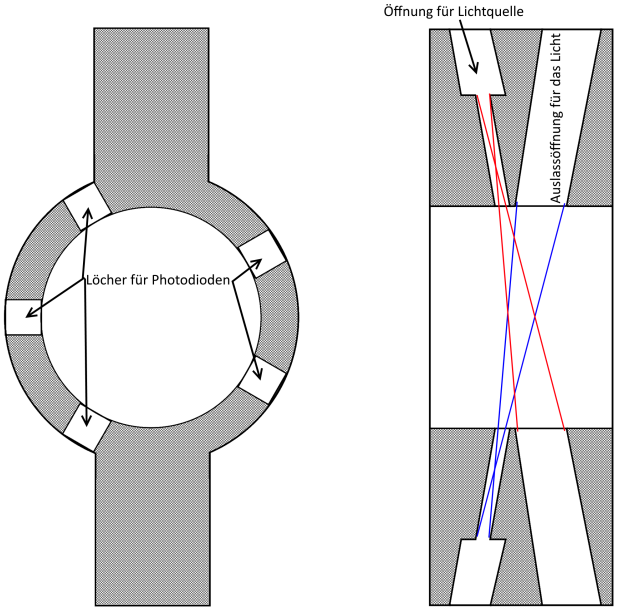


Abbildung 5.16.: Skizze der optischen Kammer; links horizontaler Schnitt, rechts vertikaler Schnitt in der Längsachse

Die optische Kammer und die Platinen der Messelektronik sind so auf einander abgestimmt, dass die Öffnungen in der Kammer für die Phototransitohren genau unterhalb der Anschlüssen auf der Platine liegen. Des Weiteren sind Bohrungen in den Platinen so gesetzt, dass sie mit der Kammer verschraubt werden können.
Die Kammer des Prototypen wurde aus milchig weißem Kunststoff gefertigt, deshalb ist sie mit deckendem matten Lack gestrichen worden, um zum Einen das Eindringen von Umgebungslicht zu verhindern und zum Anderen die Reflexionen im Inneren zu minimieren.

Die Lichtimpulse werden über 2 sich vom 6 mm auf 2 mm Durchmesser verjüngende Bohrungen in die Messkammer geführt und auf der gegen überliegenden Seite wieder durch 2 Löcher mit 7 mm Durchmesser aus der Kammer entlassen. In Abbildung 5.16 sind die Stahlenverläufe in rot und blau angedeutet.
Damit die Luft nur von oben nach unten durch die Kammer strömt, werden die Löcher der Lichtauslässe mit schwarzem Isolierklebeband verschlossen. Alle anderen seitlichen Öffnungen sind durch die Phototransistoren und Leuchtdioden verschlossen.
Damit von Hinten kein Licht in die Phototransistoren fallen kann, wird nach der Montage der Platinen die gesamte Seitenfläche mit schwarzem Isolierband umwickelt.

## 5.4. Software

### 5.4.1. Mikrocontroller

Die Software für das *Tiva C Series TM4C123G LaunchPad Evaluation Board* wird mit der Software *Code Composer Studio 6.1.2* von Texas Instruments geschrieben.

Der Mikrocontroller hat 3 große Funktionen:

- Synchronisierte Ansteuerung der Lichtquelle und Analog-Digital-Wandlung der Partikelmessung
- Auslesen und Konfigurieren der zusätzlichen Umweltsensoren über einen I2C-Bus
- Seine Messungen über I2C-Bus konfigurieren lassen und die Messwerte über denselben Bus verschicken

Es wird das Firmware-Paket *TivaWare* Version 2.1.2.111 für das *Tiva C Series TM4C123G LaunchPad Evaluation Board* verwendet. Das *Hello*-Beispielprojekt daraus wird als Ausgangspunkt genutzt. Dort ist schon eine UART-Schnittstelle zum Debuggen eingerichtet. Es wird so programmiert, dass es immer ein gleichnamiges Paar aus Codedatei, mit der Endung ".c", und Headerdatei, mit der Endung ".h", gibt. In den Codedateien sind die Funktionen geschrieben und in den Headerdateien sind die Deklarationen von Variablen und der Funktionen, sowie die Definitionen vom Makros. Alle Dateien mit dem Namensanteil "_pin"konfigurieren die Ausgangspins für das entsprechende Modul.
Der Controller ist so konfiguriert, dass er nach den Systemstart in Wartestellung geht und auf Befehle von Einplatinenrechner erwartet.

### Synchrone Ansteuerung der Lichtquelle und Analog-Digital-Wandlung der Partikelmessung

Die zeitliche Abfolge bei einer Messung ist in Abbildung 5.17 dargestellt.

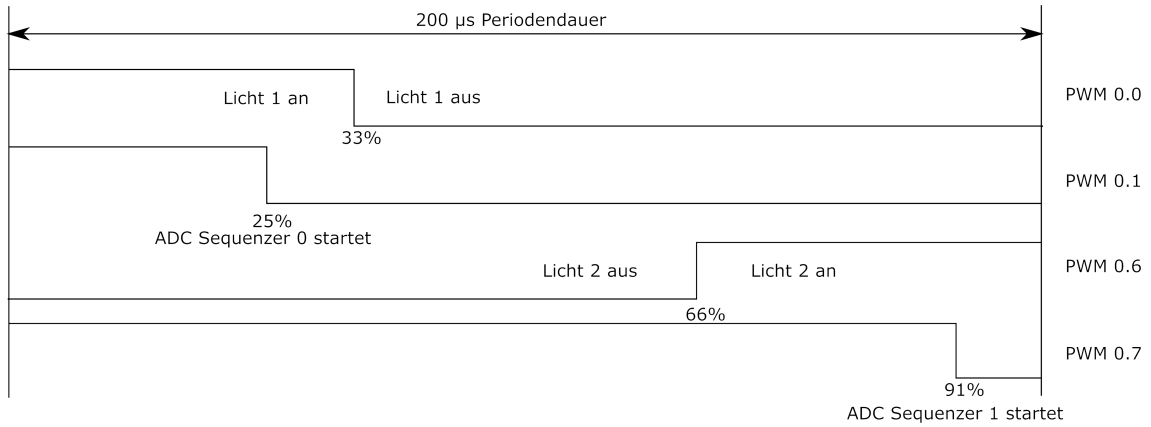


Abbildung 5.17.: Zeitlicher Aublauf einer Messperiode; die Prozentangaben sind der relative Zeitpunkt innerhalb der Messperiode

Das PWM-Modul des Controllers bietet die Möglichkeit bis zu 8 synchronisierte PWM-Signale zu Erzeugen. Diese Signale können nicht nur extern als Spannungssignal genutzt werden, sondern auch intern als Auslöser für die ADCs. Das PWM- und ADC-Modul werden entsprechend Abbildung 5.17 eingestellt. wobei eine Periodendauer von 200 µs gewählt wird, da bei dieser Frequenz die Signale bis zum Messzeitpunkt eingeschwungen sind. Dieses wurde im Vorfeld untersucht. Diese Messfrequenz ist fest und unabhängig von durch den Datenlogger vorgegebenen Messperiode. Diese wird erreicht in dem solange Messwerte aufaddiert werden bis die vorgegeben Messzeit erreicht ist. Da die ADCs mit 12 Bit auflösen und native auf dem Controller in 32 Bit breite Werte gearbeitet wird, kann aus die Art bei 5 kHz Abtastrate eine Messzeit von 209 Sekunden erreicht werden, ohne dass ein Überlauf der Variablen riskiert wird, gefordert waren nur 60 s. Die entsprechenden Quellcodes dazu finden sich in den Anhängen B.1, B.2, B.20, B.21, B.1, B.2, B.22 und B.23.

### Auslesen und Konfigurieren der zusätzlichen Umweltsensoren über einen I2C-Bus

Die Umweltsensoren werden an das I2C-Modul 0 angeschlossen und im Master-Modus genutzt. Die Sensoren werden nach den Spezifikationen aus ihren Datenblätten angesprochen

und ihre Messwerte zwischengespeichert. Diese Funktionalität ist noch nicht abschließend umgesetzt. Die entsprechenden Quellcodes dazu finden sich in den Anhängen B.9, B.10, B.13, B.14, B.18, B.19, B.26 und B.27.

#### Messungen über I2C-Bus konfigurieren lassen und die Messwerte über denselben Bus verschicken

Hier für wird das I2C-Modul 1 im Slave-Modus genutzt. Das heißt, es wird nur aktiv, wenn ein anderes Gerät am Bus Daten verschickt oder anfragt. Es ist so programmiert, dass immer eine Interrupt-Routine ausgelöst wird, wenn Daten an seine Busadresse geschickt wurden oder Daten von ihm angefordert werden. Für den Datenaustausch auf diese Weise wurde ein Array angelegt, dessen Felder in Tabelle 5.6 zu finden sind. Dabei ist jedes Feld nur 8 Bit groß, da über den I2C-Bus immer nur 1 Byte an Stück verschickt werden kann.

Um Daten in ein Feld zu schreiben, muss der Master an die Busadresse des Mikrocontrollers erst ein Byte schicken, mit dem er an gibt, in welches Feld er schreiben will und dann das Datenbyte. Mehrere Datenbytes können nicht am Stück zu gesendet werden. Wird versucht in ein *nur-Lesen*-Feld oder Feld mit einen Index, der außerhalb des Arrays liegt, zu schreiben, wird die Anfrage ignoriert. Zum Auslesen eines Feldes wird auch wieder ein Byte an die Adresse des Controllers geschickt, welches wieder als Index des zu lesenden Feldes gilt, gefolgt von einer Sendeaufforderung, auf die hin der Mikrocontroller das entsprechend Feld verschickt. Folgt danach eine weitere Sendeaufforderung, wird das Feld mit dem nächsten Index verschickt, so lassen sich einfacher alle Messdaten im Stück abfragen. Wenn nach einem Index außerhalb des Array gefragt wird, wird immer der Index auf 0 gesetzt und Feld 0 als Fehlermeldung verschickt. Die entsprechenden Quellcodes dazu finden sich in den Anhängen B.11, B.12, B.15 und B.16.

#### Koordination der drei Aufgaben

Über den Systick-Timer wird jede 1/4 Sekunde eine Interrupt-Routine ausgelöst, mit dieser wird zwischen den drei Prozessen kommuniziert. Sie erfüllt folgende Aufgaben:

- Meldet neue Messdaten vom ADC an das I2C-Slave-Array weiter, und trägt diese dort ein
- Generiert die Status meldung im I2C-Slave-Array

Tabelle 5.6.: I2C-Slave Array

| Feldnummer | Inhalt | Zugriffsart über den Bus |
|---|---|---|
| 0 | wird bei Fehlern in der Übertragung ausgewählt; fest = 0xFF | nur Lesen |
| 1 | Status; Bit 0 = Messung im Gange, Bit 1 = Neuer Messwert bereit, Bit 3 = Modus ( 0 = Einzelmessung, 1 = wiederholte Messung ) | nur Lesen |
| 2 | Sensor ID; fest = 10 | nur Lesen |
| 3 | Messperiode; Wert + 1 in Sekunden | Lesen und Schreiben |
| 4 | Start/Stop; Wert 1 startet und Wert 2 stopt eine Messung | Lesen und Schreiben |
| 5 | Modus; 0 = Einzelmessung, 1 = Wiederholte Messung bis Abbruch | Lesen und Schreiben |
| 6-15 | nicht genutzt | nur Lesen |
| 16 - 19 | ADC-Messwert bei Beleuchtung mit Licht 1 unter 30° | nur Lesen |
| 20 - 23 | ADC-Messwert bei Beleuchtung mit Licht 1 unter 60° | nur Lesen |
| 24 - 27 | ADC-Messwert bei Beleuchtung mit Licht 1 unter 90° | nur Lesen |
| 28 - 31 | ADC-Messwert bei Beleuchtung mit Licht 1 unter 120° | nur Lesen |
| 32 - 35 | ADC-Messwert bei Beleuchtung mit Licht 1 unter 150° | nur Lesen |
| 36 - 38 | Druck | nur Lesen |
| 39 - 40 | Temperatur gemessen mit Drucksensor | nur Lesen |
| 41 - 43 | relative Luftfeuchtigkeit | nur Lesen |
| 44 - 46 | Temperatur gemessen mit Luftfeuchtigkeitssensor | nur Lesen |
| 47 - 50 | Temperatur des Controllers mit Sequenzer 0 gemessen | nur Lesen |
| 51 - 54 | Messwerte an ADC - AIN10 - Port B Pin 4 Sequenzer 0 | nur Lesen |
| 51 - 54 | Messwerte an ADC - AIN11 - Port B Pin 5 Sequenzer 0 | nur Lesen |
| 55 - 58 | ADC-Messwert bei Beleuchtung mit Licht 2 unter 30° | nur Lesen |
| 59 - 62 | ADC-Messwert bei Beleuchtung mit Licht 2 unter 60° | nur Lesen |
| 63 - 66 | ADC-Messwert bei Beleuchtung mit Licht 2 unter 90° | nur Lesen |
| 67 - 70 | ADC-Messwert bei Beleuchtung mit Licht 2 unter 120° | nur Lesen |
| 71 - 74 | ADC-Messwert bei Beleuchtung mit Licht 2 unter 150° | nur Lesen |
| 75 - 78 | Temperatur des Controllers mit Sequenzer 1 gemessen | nur Lesen |
| 79 - 82 | Messwerte an ADC - AIN10 - Port B Pin 4 Sequenzer 1 | nur Lesen |
| 83 - 86 | Messwerte an ADC - AIN11 - Port B Pin 5 Sequenzer 1 | nur Lesen |

- Startet und stoppt die Messwertaufnahme bei Eintreffen des entsprechenden Befehls über das I2C-Slave-Array im vorgegeben Modus
- Passt die Samplelänge des ADCs bei Änderung der Sampleperiode im I2C-Slave-Array entsprechend an
- Schaltet den Lüfter
- Schaltet das *neuer-Messwert-vorhanden*-Interrupt-Signal

Die entsprechenden Quellcodes dazu finden sich in den Anhängen B.28 und B.29.

### 5.4.2. Einplatinencomputer

Hier wird nur als Nachweis, dass die sensitive Einheit als Sensor für Datenlogger funktioniert, ein einfaches Datenlogging-Skript erstellt. Es ist in Python, einer Skript-Sprache, geschrieben, da es hierfür sehr einfach zu benutzende Bibliotheken für den Raspberry Pi gibt. Das Skript startet eine wiederholte Messung, mit einer Samplelänge von 1 Sekunde. Mit einer Interrupt-Routine werden bei jeder steigenden Flanke des *neuer-Messwert-vorhanden*-Interrupt-Signal alle ADC-Werte abgerufen, in reale Spannungswerte umgerechnet und in eine Textdatei geschrieben. Die Textdatei ist als Tabelle mit Semikolons als Trennzeichen formatiert. Nach 60 Sekunden wird die Messung beendet. Das Skript findet sich in Anhang B.30.

# 6. Test und Bewertung

Es wurde zum Test eine komplette Einheit aufgebaut und ihre Komponenten erst einzeln und dann als Verbund getestet.

## 6.1. Interfaceplatine

Der Verstärkungsfaktor wurde mit Anlegen von 0,1 V an die Eingänge und Messen der Ausgangsspannung mit einem Multimeter überprüft, dabei wird je nach Position des Potentiometers eine Ausgangsspannung zwischen 0 V und 1 V erwartet. Diese Werte wurden auf allen Ausgängen auf +/- 0,03 V erreicht. Dieses ist völlig ausreichend für die Funktion der Platine.
Die Schutzschaltung gegen positive Überspannung wird getestet, indem alle Verstärker auf ihre maximal Verstärkung gestellt werden und dann 3,3 V jeweils auf einen Eingang gelegt wird, so das der Verstärker übersteuert und annähernd die positive Versorgungsspannung an dem entsprechenden Ausgang erwartet wird. Nun wird die tatsächliche Ausgangsspannung gemessen. Die übersteigt die erwarteten 3,7 V bei weitem, sie liegt etwa bei 5 V, das ist keine sichere Spannung für den ADC. Außerdem wird die Versorgungsspannung des Mikrocontrollers durch den über die Schutzdiode fließenden Strom auf 3,8 V erhöht. Auch das ist außerhalb der Spezifikation des Mikrocontrollers.
Eine Lösungsansatz könnte sein, die Schutzdiode gegen Masse durch eine Zenerdiode mit einer Durchbruchsspannung von 3 V zu ersetzten. Damit nicht die gesamte Überspannung Richtung der Versorgungsspannung abfließt, sondern sich aufteilt auf Versorgungsspannung und Masse.

## 6.2. optischer Messaufbau

Da die Elektronik zur Partikelmessung nicht ohne die optische Kammer getestet werden kann, wird beides zusammen ausprobiert. Dabei wird festgestellt, dass das von der Kammer

in Zusammenspiel mit den LEDs erzeugte Streulicht die optischen Messeingänge übersteuert. Dieses muss erst abgestellt werden, bevor wie angedacht gemessen werden kann.

## 6.3. Datenlogging

Das Testskript (B.30) hat einige Testläufe erfolgreich abgeschlossen und dabei wie erwartet in den 60 Sekunden Messdauer 60 Messwerte erfasst. Bei diesen Testläufen waren nur das Raspberry Pi und der Mikrocontroller beteiligt und über drei Steckleitungen, die I2C-Daten und Tacktleitung sowie das *neue-Messung-vorhanden*-Interruptsignal, verbunden. Das analog Eingangssignal wurde am Mikrocontroller mit den PWM-Ausgängen für die Lichtsteuerung simuliert. Damit konnte auch gezeigt werden, dass die ADCs zum richtigen Zeitpunkt ausgelöst wurden, da nur die Messwerte der PWM die zur der entsprechenden Lampe gehören auf 3 V lagen.
Allerdings wurde nachdem ersten Einbau in das Gehäuse und Verbinden des Mikrocontrollers mit der Interfaceplatine festgestellt, dass das I2C-Modul des Raspberry Pis auf einmal nicht mehr funktionierte, ebenso wie mit anderen Fertigmodulen am I2C-Bus. Unverständlicher Weise funktionierte auch nach dem Austausch des Raspberry Pis und sogar des Mikrocontrollerbords, dieser Testlauf nicht mehr. Bei weiteren Versuchen mit den Befehl *sudo i2cdetect -y 1* werden andere Fertigmodule zuverlässig erkannt, das Mikrocontrollerbord nur noch in etwa 50% der Fälle.

## 6.4. Gesamtsystem

Das Gesamtsystem konnte auf Grund der Vielzahl an Problemen noch keinem Kompletttest unterzogen werden. Der Einbau aller Systemelemente in das Gehäuse war erfolgreich.

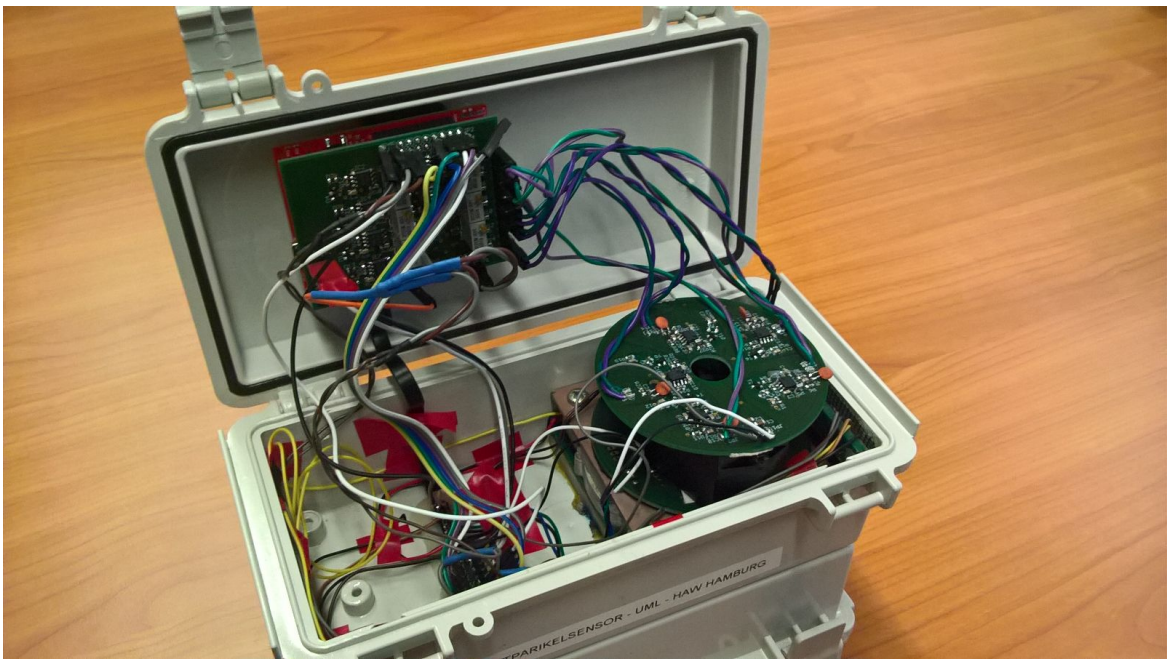


Abbildung 6.1.: Einbau sensitive Einheit

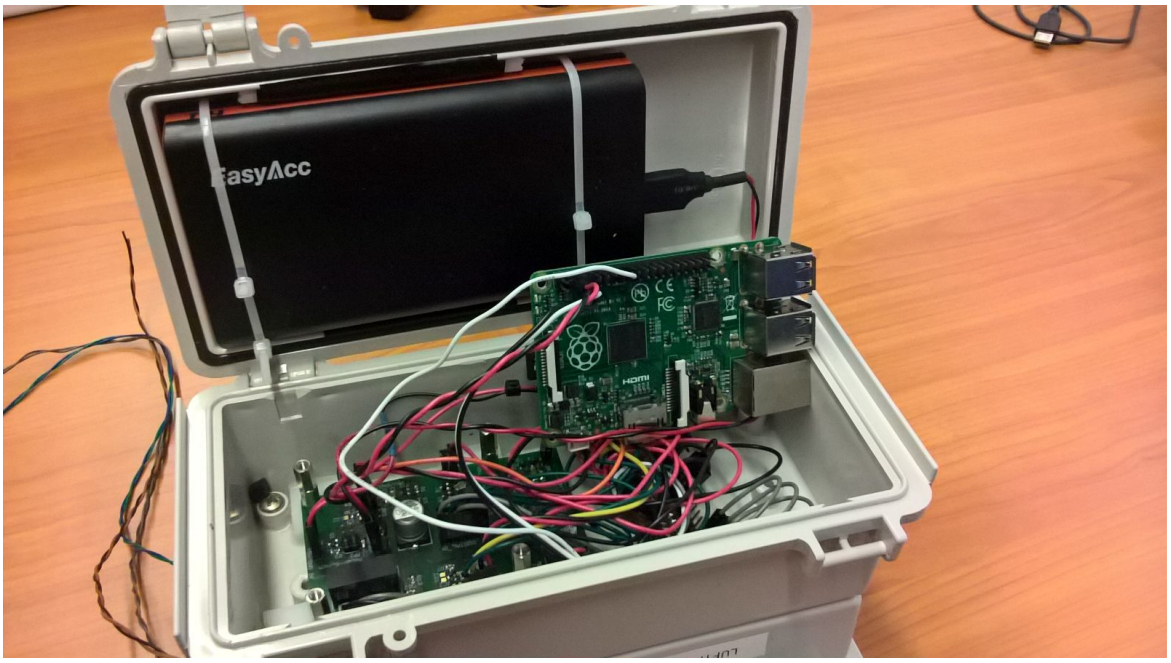


Abbildung 6.2.: Einbau Spannungsversorgung und Datenspeicherung

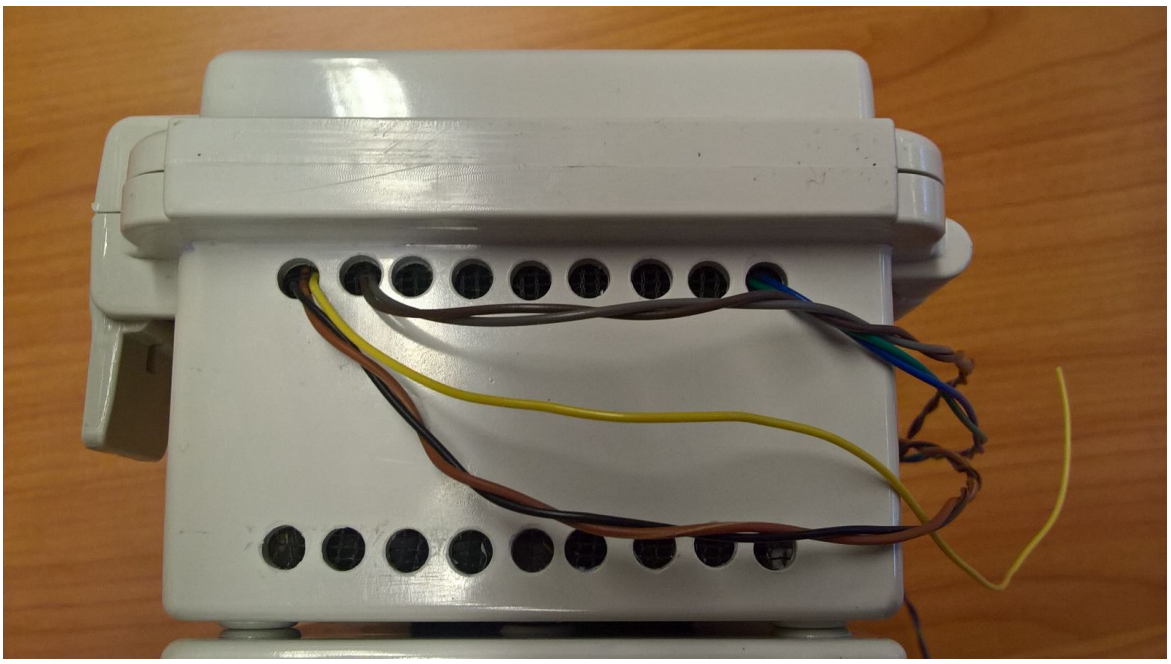

Abbildung 6.3.: Ansicht Luftöffnungen mit Insektenschutz

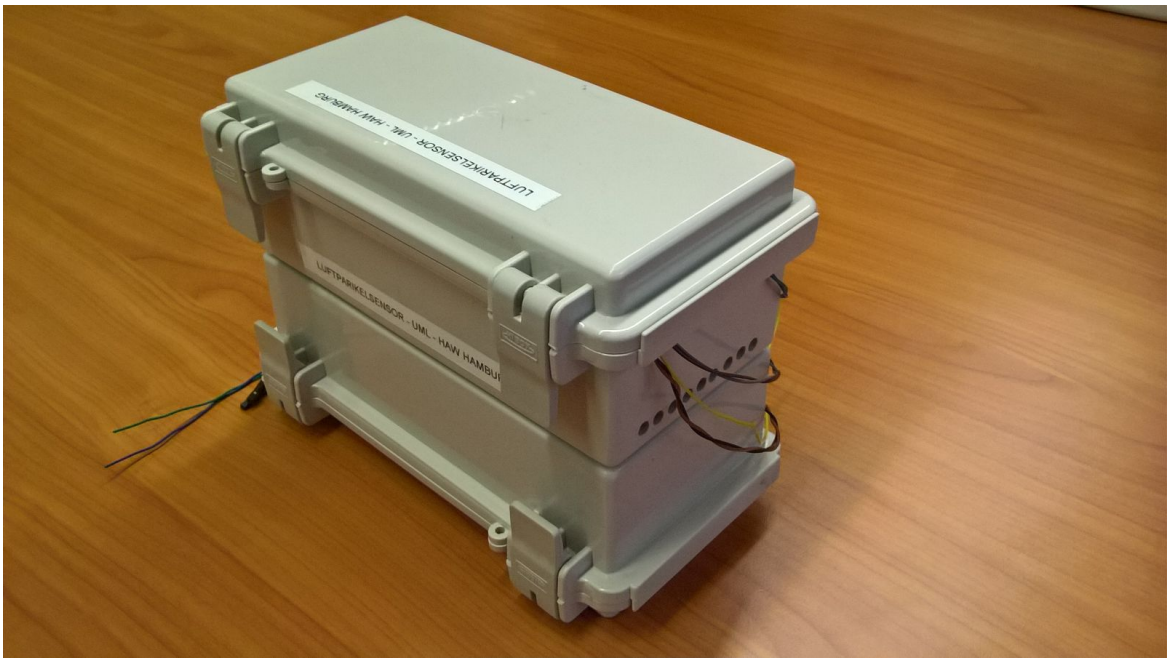


Abbildung 6.4.: Ansicht Aufbau schräg von Oben

# 7. Fazit

Die Realisierung der Idee eines kleinem portablen Gerätes, das zur Bestimmung der Partikelgröße die Eigenschaften der Mie-Streuung nutzt hat mich sehr gereizt.
Die Arbeit hat mir gezeigt, wie komplex ein auf den ersten Blick einfaches System, wie ein Datenlogger ist.
Leider ergaben sich trotz umfangreicher Nachforschung, Planung und erfolgreichen Tests der Einzelkomponenten unvorhersehbare Probleme im Zusammenspiel des Gesamtsystems, auch der gewählte Ansatz zur Messung der Streustrahlung ist Überdenkenswert.
Die Idee, die Eigenschaften der Mie-Streuung zur Bestimmung der Partikelgröße in einem kleinen portablem Gerät zu nutzen, ist in meinen Augen auch weiterhin interessant und ausbaufähig.
Auch wenn das Gesamtsystem nicht den gewünschten Erfolg hatte, ergaben sich duch meine Arbeit wichtige Erkentnisse über die Möglichkeiten eine solche mobile Messstation der Anforderungsanalyse entsprechend zu bauen.

# Literaturverzeichnis


[Pollin-1 2016] : *Pollin Electronic: Laptop-CPU-Kï¿½hler UDQFYZH07C, abgerufen 10.07.2016*. 2016. – URL http://www.pollin.de/shop/dt/NzU2OTc2OTk-/Bauelemente_Bauteile/Luefter/DC_Luefter/Laptop_CPU_Kuehler_UDQFYZH07C.html

[Bud-Industries ] BUD-INDUSTRIES: *Datenblatt: NBF SERIES PLASTIC NEMA BOX, Revision Date 9/1/2010; auf CD enthalten*

[EasyAcc 2016] EASYACC: *EasyAcc Brilliant 15000 mAh Power Bank, abgerufen 10.07.2016*. 2016. – URL https://www.easyacc.com/496-easyacc-brilliant-15000-mah-power-bank.html

[Fairchild ] FAIRCHILD: *Datenblatt: FDN306P P-Channel 1.8V Specified PowerTrench MOSFET, Rev D; auf CD enthalten*

[ICDC 2010] ICDC, Integrated Climate Data C.: *DWD Klimadaten fï¿½r Deutschland aufgerufen am 20.06.2016*. 2010. – URL http://icdc.cen.uni-hamburg.de/daten/atmosphere/dwd-station.html

[MPL3115A2 2013] MPL3115A2: *Datenblatt: Freescale Semiconductor - Xtrinsic MPL3115A2 I2C Precision Altimeter, Rev 3.0, 12/2013; auf CD enthalten*. 2013

[NXP ] NXP: *Datenblatt: MMBT3904 NPN switching transistor, Rev Date 2004 Feb 03; auf CD enthalten*

[SHTC1 2013] SHTC1: *Datenblatt: Sensirion - Data Sheet SHTC1 Humidity and Temperature Sensor IC, Version 3 - May 2014; auf CD enthalten*. 2013

[TI 2016] TI, Texas I.: *WEBENCH Design Center*. 2016. – URL http://www.ti.com/lsds/ti/analog/webench/overview.page

[Wegner 2015] WEGNER, Mario: *Entwicklung eines autonom arbeitenden GPS-Datenloggers mit hoher Updatefrequenz fï¿½r die Anwendung in Nahverkehrsbussen*, Hochschule für Angewandte Wissenschaften Hamburg, Bachelorthesis, 2015. – URL nichtveri\T1\textquestiondown\protect$\relax\protect{\begingroup1\endgroup\@@over2}$ffentlicht

# A. Stücklisten

## A.1. Spannungsversorgungseinheit

| Qty | Value | Device | Package | Parts | Description |
|---|---|---|---|---|---|
| 6 | MMBT3904 | BC846 | SOT23 | T1, T2, T3, T4, T5, T6 | NPN TRANSISTOR |
| 1 | +/-12V/GND OUT Sensor | PINHD-1X3 | 1X03 | JP11 | PIN HEADER |
| 1 | +5V/0V OUT3 Fan | PINHD-1X2 | 1X02 | JP5 | PIN HEADER |
| 1 | +5V/0V OUT4 Laser1 | PINHD-1X2 | 1X02 | JP7 | PIN HEADER |
| 1 | +5V/0V OUT5 Laser2 | PINHD-1X2 | 1X02 | JP8 | PIN HEADER |
| 1 | +5V/GND IN | PINHD-1X2 | 1X02 | JP1 | PIN HEADER |
| 1 | +5V/GND/+1.8V/+2.5V OUT2 | PINHD-1X4 | 1X04 | JP4 | PIN HEADER |
| 1 | +5V/GND/+3.3V OUT1 | PINHD-1X3 | 1X03 | JP3 | PIN HEADER |
| 2 | 1000uF | CPOL-EU150CLZ-1014 | 150CLZ-1014 | C1, C11 | POLARIZED CAPACITOR, European symbol |
| 5 | 100nF | C-EUC1206 | C1206 | C2, C4, C12, C14, C16 | CAPACITOR, European symbol |
| 5 | 100uF | C-EUC1206 | C1206 | C3, C13, C15, C17, C19 | CAPACITOR, European symbol |
| 2 | 100uH | L-EUMCSD54 | MCSD54-2023 | L2, L3 | INDUCTOR, European symbol |
| 6 | 10k | R-EU_R0805 | R0805 | R1, R4, R5, R7, R8, R12 | RESISTOR, European symbol |
| 1 | 10nF | C-EUC0805 | C0805 | C10 | CAPACITOR, European symbol |
| 1 | 10uF | C-EUC1206 | C1206 | C18 | CAPACITOR, European symbol |
| 3 | 1k | R-EU_R0805 | R0805 | R3, R6, R13 | RESISTOR, European symbol |
| 5 | 1uF | C-EUC1206 | C1206 | C5, C6, C7, C8, C9 | CAPACITOR, European symbol |

| | | | | | |
|---|---|---|---|---|---|
| 1 | 4.7uH | L-EUTYS2520 | TYS2520 | L4 | INDUCTOR, European symbol |
| 1 | 470pF or 0 Ohms | C-EUC1206 | C1206 | C20 | CAPACITOR, European symbol |
| 4 | 500 | R-EU_R0805 | R0805 | R2, R9, R10, R11 | RESISTOR, European symbol |
| 1 | AAT3221IGV-1.8-T1 | TS520525 | SOT25 | IC2 | 150mA Low Noise LDO Voltage Regulator with Enable |
| 1 | DIGITAL IN | PINHD-1X6 | 1X06 | JP6 | PIN HEADER |
| 1 | DR127 | DR127 | DR127 | L1 | High Power Density, High Efficiency, Shielded Inductors |
| 4 | FDN306P | FDN306P | SOT23 | Q1, Q2, Q3, Q4 | FDN306P |
| 1 | MIC5504-3.3YM5-TR | TS520525 | SOT25 | IC1 | 150mA Low Noise LDO Voltage Regulator with Enable |
| 2 | NM | C-EUC0805 | C0805 | C21, C22 | CAPACITOR, European symbol |
| 1 | Power-Switch | PINHD-2X2 | 2X02 | JP2 | PIN HEADER |
| 2 | PowerOut | PINHD-2X2 | 2X02 | JP9, JP10 | PIN HEADER |
| 1 | RECOM_RKZ-0512D | RECOM_RKZ-0512D | 919-RKZ-0512D | U1 | |
| | 1 | SIP21106DT-25-E3 | TS520525 | SOT25 | IC3 |

## A.2. Interfaceplatine

| Qty | Value | Device | Package | Parts | Description |
|---|---|---|---|---|---|
| 4 | | PINHD-1X10/OCT | 1X10/OCT | JP1, JP2, JP3, JP4 | PIN HEADER |
| 1 | | PINHD-1X14 | 1X14 | JP5 | PIN HEADER |
| 1 | | PINHD-1X3 | 1X03 | JP6 | PIN HEADER |
| 1 | 0 | R-EU_R0805 | R0805 | R30 | RESISTOR, European symbol |
| 7 | 100k | R-TRIMM64Y | RTRIM64Y | R1, R3, R5, R7, R9, R11, R13 | Trimm resistor |
| 7 | 100n | C-EUC0805 | C0805 | C1, C2, C3, C4, C5, C6, C7 | CAPACITOR, European symbol |
| 7 | 10k | R-EU_R0805 | R0805 | R2, R4, R6, R8, R10, R12, R14 | RESISTOR, European symbol |

| | | | | | |
|---|---|---|---|---|---|
| 2 | 10u | C-EUC1210 | C1210 | C8, C9 | CAPACITOR, European symbol |
| 14 | 1k | R-EU_R0805 | R0805 | R15, R16, R17, R18, R19, R20, R21, R22, R23, R24, R25, R26, R27, R28 | RESISTOR, European symbol |
| 7 | BA4560 | TLC272D | SO08 | IC1, IC2, IC3, IC4, IC5, IC6, IC7 | OP AMP |
| 14 | BAS70 | BAS70 | SOT23 | D1, D2, D3, D4, D5, D6, D7, D8, D9, D10, D11, D12, D13, D14 | Silicon Schottky Diodes |

## A.3. Sensor Top

| Qty | Value | Device | Package | Parts | Description |
|---|---|---|---|---|---|
| 5 | 0 | R-EU_R0805 | R0805 | R4, R9, R15, R21, R27 | RESISTOR, European symbol |
| 5 | 1.62k | R-EU_R0805 | R0805 | R5, R10, R16, R22, R28 | RESISTOR, European symbol |
| 15 | 100nF | C-EUC0805 | C0805 | C3, C4, C7, C8, C9, C12, C13, C14, C17, C18, C19, C22, C23, C24, C27 | CAPACITOR, European symbol |
| 2 | 100uF | C-EUC1206 | C1206 | C1, C2 | CAPACITOR, European symbol |
| 10 | 10nF | C-EUC0805 | C0805 | C5, C6, C10, C11, C15, C16, C20, C21, C25, C26 | CAPACITOR, European symbol |
| 5 | 178 | R-EU_R0805 | R0805 | R6, R11, R17, R23, R29 | RESISTOR, European symbol |
| 5 | 1M | R-EU_R0805 | R0805 | R3, R8, R14, R20, R26 | RESISTOR, European symbol |
| 5 | 1k | R-EU_R0805 | R0805 | R1, R2, R13, R19, R25 | RESISTOR, European symbol |
| 5 | 64.9k | R-EU_R0805 | R0805 | R7, R12, R18, R24, R30 | RESISTOR, European symbol |
| 5 | BC848 | BC848 | SOT23 | T2, T4, T6, T8, T10 | NPN TRANSISTOR |
| 5 | BPX81 | BPX81 | BPX81 | T1, T3, T5, T7, T9 | PHOTO TRANSISTOR |

| | | | | | |
|---|---|---|---|---|---|
| 5 | OPA2650U | OPA2650U | SO08 | IC1, IC2, IC3, IC4, IC5 | Dual Wideband, Low Power Voltage Feedback Operational Amplifier |
| 1 | OUT120Â° | PINHD-1X2 | 1X02 | JP5 | PIN HEADER |
| 1 | OUT150Â° | PINHD-1X2 | 1X02 | JP6 | PIN HEADER |
| 1 | OUT30Â° | PINHD-1X2 | 1X02 | JP2 | PIN HEADER |
| 1 | OUT60Â° | PINHD-1X2 | 1X02 | JP3 | PIN HEADER |
| 1 | OUT90Â° | PINHD-1X2 | 1X02 | JP4 | PIN HEADER |
| 1 | Power | PINHD-1X3 | 1X03 | JP1 | PIN HEADER |

## A.4. Sensor Bottom

| Qty | Value | Device | Package | Parts | Description |
|---|---|---|---|---|---|
| 2 | | PINHD-1X2 | 1X02 | JP3, JP4 | PIN HEADER |
| 3 | 0 | R-EU_R0805 | R0805 | R4, R9, R23 | RESISTOR, European symbol |
| 2 | 1.62k | R-EU_R0805 | R0805 | R5, R10 | RESISTOR, European symbol |
| 5 | 100 | R-TRIMM5X | RTRIM5X | R18, R19, R20, R21, R22 | Trimm resistor |
| 12 | 100nF | C-EUC0805 | C0805 | C3, C4, C5, C7, C8, C9, C12, C13, C14, C26, C27, C28 | CAPACITOR, European symbol |
| 2 | 100uF | C-EUC1206 | C1206 | C1, C2 | CAPACITOR, European symbol |
| 7 | 10k | R-EU_R0805 | R0805 | R6, R11, R13, R14, R15, R16, R17 | RESISTOR, European symbol |
| 2 | 10nF | C-EUC0805 | C0805 | C6, C11 | CAPACITOR, European symbol |
| 2 | 10uF | C-EUC1206 | C1206 | C10, C25 | CAPACITOR, European symbol |
| 2 | 1M | R-EU_R0805 | R0805 | R3, R8 | RESISTOR, European symbol |
| 2 | 1k | R-EU_R0805 | R0805 | R1, R2 | RESISTOR, European symbol |
| 2 | 64.9k | R-EU_R0805 | R0805 | R7, R12 | RESISTOR, European symbol |

| | | | | | |
|---|---|---|---|---|---|
| 3 | BA4560 | OPA2650U | SO08 | IC1, IC2, IC3 | Dual Wideband, Low Power Voltage Feedback Operational Amplifier |
| 2 | BC848 | BC848 | SOT23 | T2, T4 | NPN TRANSISTOR |
| 2 | BPX81 | BPX81 | BPX81 | T1, T3 | PHOTO TRANSISTOR |
| 1 | I2C | PINHD-1X4 | 1X04 | JP6 | PIN HEADER |
| 1 | MPL3115A2 | MPL3115A2 | MPL3115A2-LGA-8 | U2 | |
| 10 | NM | C-EUC0805 | C0805 | C15, C16, C17, C18, C19, C20, C21, C22, C23, C24 | CAPACITOR, European symbol |
| 1 | OFFSETS | PINHD-1X6 | 1X06 | JP2 | PIN HEADER |
| 1 | Power | PINHD-1X3 | 1X03 | JP1 | PIN HEADER |
| 1 | Power_Dig | PINHD-1X3 | 1X03 | JP5 | PIN HEADER |
| 1 | SHTC1 | SHTC1 | SHTC1-DFN-2X2X0.75MM | U1 | |

# B. Quellcode

## B.1. Microcontroller

Listing B.1: Mikrocontroller: adc.c

```
1  /*
2   * adc.c
3   *
4   *  Created on: 27.05.2015
5   *      Author: Torben
6   */
7
8  #include "adc.h"
9
10
11 void AdcConfig()
12 {
13   int i;
14   // Init ADC-Ports
15   AdcPortFunctionInit();
16
17   // init ADC0
18   ADC0_ACTSS_R &= 0xF0;          // all sequencers off
19   ADC0_SSMUX0_R |= 0x0000b920;   // Sequencer 0 channel  only
20   ADC0_SSCTL0_R |= 0x00002000;   // Sequencer 0 END3 set sequence Length= 4
21   ADC0_SSMUX1_R |= 0x0000b920;   // Sequencer 1 channel  only
22   ADC0_SSCTL1_R |= 0x00006000;   // Sequencer 1 END3 set sequence Length= 4 and
       Interrupt set
23   ADC0_CTL_R = 0x10;        // 3 V internal V_ref
24   ADC0_PC_R = 0x7;      // Set Samplerate to 1Msps
25   ADC0_SAC_R = 0x0;         // Set HW-Average to 1; 64 Times = 0x6
26   ADC0_EMUX_R = 0x96;        // Trigger   sequencer 0 on Gen 0 and sequencer 1
       onGen 3
27   ADC0_TSSEL_R = 0x0;       // Trigger  sequencer 0 + 1 sample on PWM0_event:
28   ADC0_IM_R |= 0x00000002; // interrupt from sequencer 1 send to interrupt contr
       .;
29   ADC0_ACTSS_R |= 0x03;     // sequencers 0 + 1 on
30
31   // init ADC1
32   ADC1_ACTSS_R &= 0xF0;          // all sequencers off
```

```
33    ADC1_SSMUX0_R |= 0x00000a81;   // Sequencer 0 channel  only
34    ADC1_SSCTL0_R |= 0x0000a000;   // Sequencer END3 set sequence Length= 4 and TS3
          set
35    ADC1_SSMUX1_R |= 0x00000a81;   // Sequencer 1 channel  only
36    ADC1_SSCTL1_R |= 0x0000a000;   // Sequencer END3 set sequence Length= 4 and TS3
          set
37    ADC1_CTL_R = 0x10;         // 3 V internal V_ref
38    ADC1_PC_R = 0x7;       // Set Samplerate to 1Msps
39    ADC1_SAC_R = 0x0;          // Set HW-Average to 1; 64 Times = 0x6
40    ADC1_EMUX_R = 0x96;         // Trigger   sequencer 0 on Gen 0 and sequencer 1
          onGen 3
41    ADC1_TSSEL_R = 0x0;        // Trigger sequencer 0 + 1 sample on PWM0_event:
42    //ADC1_IM_R |= 0x00000001; // interrupt from sequencer 0 send to interrupt
          contr.;
43    ADC1_ACTSS_R |= 0x03;      // sequencers 0 + 1 on
44
45    // Register Interrupthandler to Sequenzer 1
46    ADCIntRegister(ADC0_BASE, 1, AdcIntHandler);
47
48    // for counting ADC-Interrupts
49    ui32AdcIntCount = 0;
50    for (i = 0; i < 8; i++)
51    {
52      ui32AdcSamples[0][0][i] = 0;
53      ui32AdcSamples[0][1][i] = 0;
54      ui32AdcSamples[1][0][i] = 0;
55      ui32AdcSamples[1][1][i] = 0;
56    }
57    ui32AdcSampleLengh = 5000;
58    ui32AdcSampleActive = 0;
59    ui32AdcSampleReading = 1;
60    ui32AdcNewSample = 0;
61  //  ui32AdcLaser = 0; not used anymore
62  }
63
64  void AdcIntHandler()
65  {
66    ADCIntClear(ADC0_BASE, 1); // clear Interupt
67    // debug
68  //  GPIO_PORTF_DATA_R &= ~(0x10); //fallene Flanke
69
70    if (ui32AdcIntCount < ui32AdcSampleLengh)
71    { // adding Samples Up
72      ui32AdcSamples[0][ui32AdcSampleActive][0] += ADC0_SSFIFO0_R;
73      ui32AdcSamples[0][ui32AdcSampleActive][1] += ADC1_SSFIFO0_R;
74      ui32AdcSamples[0][ui32AdcSampleActive][2] += ADC0_SSFIFO0_R;
75      ui32AdcSamples[0][ui32AdcSampleActive][3] += ADC1_SSFIFO0_R;
76      ui32AdcSamples[0][ui32AdcSampleActive][4] += ADC0_SSFIFO0_R;
```

```
77      ui32AdcSamples[0][ui32AdcSampleActive][5] += ADC1_SSFIFO0_R;
78      ui32AdcSamples[0][ui32AdcSampleActive][6] += ADC0_SSFIFO0_R;
79      ui32AdcSamples[0][ui32AdcSampleActive][7] += ADC1_SSFIFO0_R;
80
81      ui32AdcSamples[1][ui32AdcSampleActive][0] += ADC0_SSFIFO1_R;
82      ui32AdcSamples[1][ui32AdcSampleActive][1] += ADC1_SSFIFO1_R;
83      ui32AdcSamples[1][ui32AdcSampleActive][2] += ADC0_SSFIFO1_R;
84      ui32AdcSamples[1][ui32AdcSampleActive][3] += ADC1_SSFIFO1_R;
85      ui32AdcSamples[1][ui32AdcSampleActive][4] += ADC0_SSFIFO1_R;
86      ui32AdcSamples[1][ui32AdcSampleActive][5] += ADC1_SSFIFO1_R;
87      ui32AdcSamples[1][ui32AdcSampleActive][6] += ADC0_SSFIFO1_R;
88      ui32AdcSamples[1][ui32AdcSampleActive][7] += ADC1_SSFIFO1_R;
89
90      ui32AdcIntCount++; //count Interrupts
91    }
92    else
93    {
94      ui32AdcSampleReady = ui32AdcSampleActive;
95
96      switch (ui32AdcSampleActive)
97      {
98      case 0:
99        if (ui32AdcSampleReading == 1)
100       {
101         ui32AdcSampleActive = 2;
102       }
103       else
104       {
105         ui32AdcSampleActive = 1;
106       }
107       break;
108     case 1:
109       if (ui32AdcSampleReading == 0)
110       {
111         ui32AdcSampleActive = 2;
112       }
113       else
114       {
115         ui32AdcSampleActive = 0;
116       }
117       break;
118     case 2:
119       if (ui32AdcSampleReading == 0)
120       {
121         ui32AdcSampleActive = 1;
122       }
123       else
124       {
```

```
125             ui32AdcSampleActive = 0;
126           }
127         }
128
129         ui32AdcSamples[0][ui32AdcSampleActive][0] = ADC0_SSFIFO0_R;
130         ui32AdcSamples[0][ui32AdcSampleActive][1] = ADC1_SSFIFO0_R;
131         ui32AdcSamples[0][ui32AdcSampleActive][2] = ADC0_SSFIFO0_R;
132         ui32AdcSamples[0][ui32AdcSampleActive][3] = ADC1_SSFIFO0_R;
133         ui32AdcSamples[0][ui32AdcSampleActive][4] = ADC0_SSFIFO0_R;
134         ui32AdcSamples[0][ui32AdcSampleActive][5] = ADC1_SSFIFO0_R;
135         ui32AdcSamples[0][ui32AdcSampleActive][6] = ADC0_SSFIFO0_R;
136         ui32AdcSamples[0][ui32AdcSampleActive][7] = ADC1_SSFIFO0_R;
137
138         ui32AdcSamples[1][ui32AdcSampleActive][0] = ADC0_SSFIFO1_R;
139         ui32AdcSamples[1][ui32AdcSampleActive][1] = ADC1_SSFIFO1_R;
140         ui32AdcSamples[1][ui32AdcSampleActive][2] = ADC0_SSFIFO1_R;
141         ui32AdcSamples[1][ui32AdcSampleActive][3] = ADC1_SSFIFO1_R;
142         ui32AdcSamples[1][ui32AdcSampleActive][4] = ADC0_SSFIFO1_R;
143         ui32AdcSamples[1][ui32AdcSampleActive][5] = ADC1_SSFIFO1_R;
144         ui32AdcSamples[1][ui32AdcSampleActive][6] = ADC0_SSFIFO1_R;
145         ui32AdcSamples[1][ui32AdcSampleActive][7] = ADC1_SSFIFO1_R;
146
147         ui32AdcIntCount = 1;
148         ui32AdcNewSample = 1;
149       }
150
151       // Toggle ui32AdcLaser     not used any more
152       //if (ui32AdcLaser) ui32AdcLaser = 0; else ui32AdcLaser = 1;
153
154       // Debug
155
156     //   if( GPIO_PORTF_DATA_R & 0x10 )
157     //   {
158     //     GPIO_PORTF_DATA_R &= ~(0x10); //fallene Flanke
159     //   }
160     //   else
161     //   {
162     //     GPIO_PORTF_DATA_R |= 0x10; //steigende Flange
163     //   }
164
165   }
```

Listing B.2: Mikrocontroller: acd.h

```
1  /*
2   * adc.h
3   *
4   *  Created on: 27.05.2015
5   *      Author: Torben
```

```
6   */
7
8   #ifndef ADC_H_
9   #define ADC_H_
10
11  #include <stdint.h>
12  #include <stdbool.h>
13
14  #include "driverlib/interrupt.h"
15  #include "driverlib/adc.h"
16
17  #include "inc/tm4c123gh6pm.h"
18  #include "inc/hw_memmap.h"
19
20  #include "adc_pin.h"
21  #include "i2c_slave.h"
22  #include "pwm.h"
23
24  // to count ADCInterrupts
25  volatile uint32_t ui32AdcIntCount;
26  volatile uint32_t ui32AdcSamples[2][3][8];
27  volatile uint32_t ui32AdcSampleLengh;
28  volatile uint32_t ui32AdcSampleActive;
29  volatile uint32_t ui32AdcSampleReading;
30  volatile uint32_t ui32AdcSampleReady;
31  volatile uint32_t ui32AdcNewSample;
32  volatile uint32_t ui32AdcMode;   // 0 == cont. , 1 == single Mode
33  //volatile uint32_t ui32AdcLaser; // 0 == Signal from Laser1; 1 == Signal from
        Laser2
34
35  void AdcConfig();
36  void AdcIntHandler();
37
38
39  #endif /* ADC_H_ */
```

Listing B.3: Mikrocontroller: adc_pin.c

```
1   //*****************************************************************************
2   // Copyright (c) 2014 Texas Instruments Incorporated.  All rights reserved.
3   // Software License Agreement
4   //
5   //   Redistribution and use in source and binary forms, with or without
6   //   modification, are permitted provided that the following conditions
7   //   are met:
8   //
9   //   Redistributions of source code must retain the above copyright
10  //   notice, this list of conditions and the following disclaimer.
11  //
```

```
12 //   Redistributions in binary form must reproduce the above copyright
13 //   notice, this list of conditions and the following disclaimer in the
14 //   documentation and/or other materials provided with the
15 //   distribution.
16 //
17 //   Neither the name of Texas Instruments Incorporated nor the names of
18 //   its contributors may be used to endorse or promote products derived
19 //   from this software without specific prior written permission.
20 //
21 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
22 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
23 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
24 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
25 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
26 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
27 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
28 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
29 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
30 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
31 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
32 //
33 // This file was automatically generated by the Tiva C Series PinMux Utility
34 // Version: 1.0.4
35 //
36 //*****************************************************************************
37
38 #include "adc_pin.h"
39
40 #include <stdint.h>
41 #include <stdbool.h>
42 #include "inc/hw_types.h"
43 #include "inc/hw_memmap.h"
44 #include "inc/hw_gpio.h"
45 #include "driverlib/sysctl.h"
46 #include "driverlib/pin_map.h"
47 #include "driverlib/rom_map.h"
48 #include "driverlib/gpio.h"
49
50 //*****************************************************************************
51
52 //*****************************************************************************
53 // Init for analog IN-Pins
54 // only using AINs on Port E
55 // AINs:   0,  1,  2,  8,  9, 10, 11
56 // GPIOs: E3, E2, E1, E5, E4, B4, B5
57 //*****************************************************************************
58 void
59 AdcPortFunctionInit(void)
```

```
60  {
61      //
62      // Enable Peripheral Clocks
63      //
64      MAP_SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC0);
65      MAP_SysCtlPeripheralEnable(SYSCTL_PERIPH_ADC1);
66      MAP_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
67      //MAP_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
68      MAP_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
69
70      //
71      // Enable pin PE0 for ADC AIN3
72      //
73      //MAP_GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_0);
74
75      //
76      // Enable pin PB4 for ADC AIN10
77      //
78      MAP_GPIOPinTypeADC(GPIO_PORTB_BASE, GPIO_PIN_4);
79
80      //
81      // Enable pin PD1 for ADC AIN6
82      //
83      //MAP_GPIOPinTypeADC(GPIO_PORTD_BASE, GPIO_PIN_1);
84
85      //
86      // Enable pin PD0 for ADC AIN7
87      //
88      //MAP_GPIOPinTypeADC(GPIO_PORTD_BASE, GPIO_PIN_0);
89
90      //
91      // Enable pin PE3 for ADC AIN0
92      //
93      MAP_GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_3);
94
95      //
96      // Enable pin PE1 for ADC AIN2
97      //
98      MAP_GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_1);
99
100     //
101     // Enable pin PD3 for ADC AIN4
102     //
103     //MAP_GPIOPinTypeADC(GPIO_PORTD_BASE, GPIO_PIN_3);
104
105     //
106     // Enable pin PE4 for ADC AIN9
107     //
```

```
108     MAP_GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_4);
109
110     //
111     // Enable pin PD2 for ADC AIN5
112     //
113     //MAP_GPIOPinTypeADC(GPIO_PORTD_BASE, GPIO_PIN_2);
114
115     //
116     // Enable pin PE5 for ADC AIN8
117     //
118     MAP_GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_5);
119
120     //
121     // Enable pin PB5 for ADC AIN11
122     //
123     MAP_GPIOPinTypeADC(GPIO_PORTB_BASE, GPIO_PIN_5);
124
125     //
126     // Enable pin PE2 for ADC AIN1
127     //
128     MAP_GPIOPinTypeADC(GPIO_PORTE_BASE, GPIO_PIN_2);
129 }
130 //*****************************************************************************
```

Listing B.4: Mikrocontroller: adc_pin.h

```
1  //*****************************************************************************
2  // Copyright (c) 2014 Texas Instruments Incorporated.  All rights reserved.
3  // Software License Agreement
4  //
5  //   Redistribution and use in source and binary forms, with or without
6  //   modification, are permitted provided that the following conditions
7  //   are met:
8  //
9  //   Redistributions of source code must retain the above copyright
10 //   notice, this list of conditions and the following disclaimer.
11 //
12 //   Redistributions in binary form must reproduce the above copyright
13 //   notice, this list of conditions and the following disclaimer in the
14 //   documentation and/or other materials provided with the
15 //   distribution.
16 //
17 //   Neither the name of Texas Instruments Incorporated nor the names of
18 //   its contributors may be used to endorse or promote products derived
19 //   from this software without specific prior written permission.
20 //
21 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
22 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
23 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
```

```
24 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
25 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
26 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
27 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
28 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
29 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
30 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
31 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
32 //
33 // This file was automatically generated by the Tiva C Series PinMux Utility
34 // Version: 1.0.4
35 //
36 //*****************************************************************************
37
38 #ifndef __ADC_PIN_H__
39 #define __ADC_PIN_H__
40
41 extern void AdcPortFunctionInit(void);
42
43 #endif //  __ADC_PIN_H__
```

Listing B.5: Mikrocontroller: gpio.c

```
1 /*
2  * gpio.c
3  *
4  *  Created on: 22.08.2016
5  *      Author: goeth
6  */
7
8 #include "gpio.h"
9
10 void GpioConfig()
11 {
12   GpioPortAPortFunctionInit();
13 }
```

Listing B.6: Mikrocontroller: gpio.h

```
1 /*
2  * gpio.h
3  *
4  *  Created on: 22.08.2016
5  *      Author: goeth
6  */
7
8 #ifndef GPIO_H_
9 #define GPIO_H_
10
```

```
11 #include "gpio.h"
12 #include "gpio_port_a.h"
13
14 void GpioConfig();
15
16 #endif /* GPIO_H_ */
```

Listing B.7: Mikrocontroller: gpio_port_a.c

```
1  //*****************************************************************************
2  // Copyright (c) 2014 Texas Instruments Incorporated.  All rights reserved.
3  // Software License Agreement
4  //
5  //   Redistribution and use in source and binary forms, with or without
6  //   modification, are permitted provided that the following conditions
7  //   are met:
8  //
9  //   Redistributions of source code must retain the above copyright
10 //   notice, this list of conditions and the following disclaimer.
11 //
12 //   Redistributions in binary form must reproduce the above copyright
13 //   notice, this list of conditions and the following disclaimer in the
14 //   documentation and/or other materials provided with the
15 //   distribution.
16 //
17 //   Neither the name of Texas Instruments Incorporated nor the names of
18 //   its contributors may be used to endorse or promote products derived
19 //   from this software without specific prior written permission.
20 //
21 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
22 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
23 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
24 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
25 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
26 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
27 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
28 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
29 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
30 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
31 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
32 //
33 // This file was automatically generated by the Tiva C Series PinMux Utility
34 // Version: 1.0.4
35 //
36 //*****************************************************************************
37
38 #include <stdint.h>
39 #include <stdbool.h>
40 #include "gpio_port_a.h"
```

```
41 #include "inc/hw_types.h"
42 #include "inc/hw_memmap.h"
43 #include "inc/hw_gpio.h"
44 #include "driverlib/sysctl.h"
45 #include "driverlib/pin_map.h"
46 #include "driverlib/rom_map.h"
47 #include "driverlib/gpio.h"
48
49 //*****************************************************************************
50 void
51 GpioPortAPortFunctionInit(void)
52 {
53     //
54     // Enable Peripheral Clocks
55     //
56     MAP_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
57
58     //
59     // Enable pin PA2 for GPIOOutput
60     // For Fan Control
61     MAP_GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_2);
62
63     //
64     // Enable pin PA4 for GPIOOutput
65     //
66     MAP_GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_4);
67
68     //
69     // Enable pin PA3 for GPIOOutput
70     //
71     MAP_GPIOPinTypeGPIOOutput(GPIO_PORTA_BASE, GPIO_PIN_3);
72 }
```

Listing B.8: Mikrocontroller: gpio_port_a.h

```
1  //*****************************************************************************
2  // Copyright (c) 2014 Texas Instruments Incorporated.  All rights reserved.
3  // Software License Agreement
4  //
5  //   Redistribution and use in source and binary forms, with or without
6  //   modification, are permitted provided that the following conditions
7  //   are met:
8  //
9  //   Redistributions of source code must retain the above copyright
10 //   notice, this list of conditions and the following disclaimer.
11 //
12 //   Redistributions in binary form must reproduce the above copyright
13 //   notice, this list of conditions and the following disclaimer in the
14 //   documentation and/or other materials provided with the
```

```
15 //    distribution.
16 //
17 //    Neither the name of Texas Instruments Incorporated nor the names of
18 //    its contributors may be used to endorse or promote products derived
19 //    from this software without specific prior written permission.
20 //
21 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
22 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
23 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
24 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
25 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
26 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
27 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
28 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
29 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
30 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
31 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
32 //
33 // This file was automatically generated by the Tiva C Series PinMux Utility
34 // Version: 1.0.4
35 //
36 //*****************************************************************************
37
38 #ifndef __GPIO_PORT_A_H__
39 #define __GPIO_PORT_A_H__
40
41 extern void GpioPortAPortFunctionInit(void);
42
43 #endif //  __GPIO_PORT_A_H__
```

Listing B.9: Mikrocontroller: i2c_master.c

```
1  /*
2   * i2c_master.c
3   *
4   *  Created on: 22.04.2015
5   *      Author: Torben
6   *
7   *  ToDo: I2C_MasterSend, Multi Byte Send/Reseive
8   *
9   */
10
11
12 #include "i2c_master.h"
13
14
15 // Configures I2C-Modul 0 to Normal-Mode
16 // on Ports PB2 = SCL and PB3 = SDA
17 void I2C_MasterConfig(void)
```

```
18 {
19     //enable I2C module 0
20     SysCtlPeripheralEnable(SYSCTL_PERIPH_I2C0);
21
22     //reset module
23     SysCtlPeripheralReset(SYSCTL_PERIPH_I2C0);
24
25   // Initialize I2C0-Pins
26   I2C0PortFunctionInit();
27   //
28   // Initialize Master and Slave
29   //
30   I2CMasterInitExpClk(I2C0_BASE, ROM_SysCtlClockGet(), false);
31   //
32   // Set Timeout to 20ms
33   I2CMasterTimeoutSet(I2C0_BASE, 0x7d);
34 }
35
36
37 //sends an I2C command to the specified slave
38 // SOURCE: https://eewiki.net/display/microcontroller/I2C+Communication+with+the+
      TI+Tiva+TM4C123GXL
39 void I2CSend(uint8_t slave_addr, uint8_t num_of_args, ...)
40 {
41   uint8_t i;
42
43     // Tell the master module what address it will place on the bus when
44     // communicating with the slave.
45     I2CMasterSlaveAddrSet(I2C0_BASE, slave_addr, false);
46
47     //stores list of variable number of arguments
48     va_list vargs;
49
50     //specifies the va_list to "open" and the last fixed argument
51     //so vargs knows where to start looking
52     va_start(vargs, num_of_args);
53
54     //put data to be sent into FIFO
55     I2CMasterDataPut(I2C0_BASE, va_arg(vargs, uint32_t));
56
57     //if there is only one argument, we only need to use the
58     //single send I2C function
59     if(num_of_args == 1)
60     {
61         //Initiate send of data from the MCU
62         I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_SINGLE_SEND);
63
64         // Wait until MCU is done transferring.
```

```
65          while(I2CMasterBusy(I2C0_BASE));
66
67          //"close" variable argument list
68          va_end(vargs);
69      }
70
71      //otherwise, we start transmission of multiple bytes on the
72      //I2C bus
73      else
74      {
75          //Initiate send of data from the MCU
76          I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_START);
77
78          // Wait until MCU is done transferring.
79          while(I2CMasterBusy(I2C0_BASE));
80
81          //send num_of_args-2 pieces of data, using the
82          //BURST_SEND_CONT command of the I2C module
83          for(i = 1; i < (num_of_args - 1); i++)
84          {
85              //put next piece of data into I2C FIFO
86              I2CMasterDataPut(I2C0_BASE, va_arg(vargs, uint32_t));
87              //send next data that was just placed into FIFO
88              I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_CONT);
89
90              // Wait until MCU is done transferring.
91              while(I2CMasterBusy(I2C0_BASE));
92          }
93
94          //put last piece of data into I2C FIFO
95          I2CMasterDataPut(I2C0_BASE, va_arg(vargs, uint32_t));
96          //send next data that was just placed into FIFO
97          I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_FINISH);
98          // Wait until MCU is done transferring.
99          while(I2CMasterBusy(I2C0_BASE));
100
101         //"close" variable args list
102         va_end(vargs);
103     }
104 }
105
106
107
108 // sends 1 byte of data to slave register reg, with Slave-Address slaveAddr
109 void I2C_MasterSend(uint8_t slaveAddr, uint8_t reg, uint8_t data)
110 {
111   while(I2CMasterBusBusy(I2C0_BASE) | I2CMasterBusy(I2C0_BASE));
112   I2CMasterSlaveAddrSet(I2C0_BASE, slaveAddr, SEND);
```

```
113   I2CMasterDataPut(I2C0_BASE, reg);
114   I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_START);
115   while(I2CMasterBusy(I2C0_BASE));
116   I2CMasterDataPut(I2C0_BASE, data);
117   I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_FINISH);
118   while(I2CMasterBusy(I2C0_BASE));
119
120 }
121
122 // Slave Debug
123 void I2C_MasterSendSlaveDebug(uint8_t slaveAddr, uint8_t reg, uint8_t data)
124 {
125   while(I2CMasterBusBusy(I2C0_BASE) | I2CMasterBusy(I2C0_BASE));
126   I2CMasterSlaveAddrSet(I2C0_BASE, slaveAddr, SEND);
127   I2CMasterDataPut(I2C0_BASE, reg);
128   I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_SINGLE_SEND);
129 //   while(I2CMasterBusy(I2C0_BASE));
130 //   I2CMasterDataPut(I2C0_BASE, data);
131 //   I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_FINISH);
132 //   while(I2CMasterBusBusy(I2C0_BASE) | I2CMasterBusy(I2C0_BASE));
133
134 }
135
136 // reseives 1 byte of data from Slave-Register reg, with Slave-Address slaveAddr
137 uint8_t I2C_MasterReseive(uint8_t slaveAddr, uint8_t reg)
138 {
139   // Safty, when Bus not ready
140   while(I2CMasterBusBusy(I2C0_BASE) | I2CMasterBusy(I2C0_BASE));
141
142   // Set Register to read from Slave
143   I2CMasterSlaveAddrSet(I2C0_BASE, slaveAddr, SEND);
144   I2CMasterDataPut(I2C0_BASE, reg);
145   I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_SINGLE_SEND);
146   while(I2CMasterBusBusy(I2C0_BASE) | I2CMasterBusy(I2C0_BASE));
147
148   // Read Register
149   I2CMasterSlaveAddrSet(I2C0_BASE, slaveAddr, RECEIVE);
150   I2CMasterDataPut(I2C0_BASE, reg);  // only Dummy for Delay
151   I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_SINGLE_RECEIVE);
152   while(I2CMasterBusBusy(I2C0_BASE) | I2CMasterBusy(I2C0_BASE));
153   return I2CMasterDataGet(I2C0_BASE);
154 }
155
156 // reseives 1 byte of data from 16bit Slave-Register, with Slave-Address
      slaveAddr
157 uint16_t I2C_MasterReseive16bitReg(uint8_t slaveAddr, uint8_t reg, uint8_t data)
158 {
159   uint16_t received = 0;
```

```
160
161     // Safty, when Bus not ready
162     while(I2CMasterBusBusy(I2C0_BASE) | I2CMasterBusy(I2C0_BASE));
163
164     // Set Register to read from Slave
165     I2CMasterSlaveAddrSet(I2C0_BASE, slaveAddr, SEND);
166     I2CMasterDataPut(I2C0_BASE, reg);
167     I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_START);
168     while(I2CMasterBusy(I2C0_BASE));
169     I2CMasterDataPut(I2C0_BASE, data);
170     I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_SEND_FINISH);
171     while(I2CMasterBusy(I2C0_BASE));
172
173     // Read Register
174     I2CMasterSlaveAddrSet(I2C0_BASE, slaveAddr, RECEIVE);
175     I2CMasterDataPut(I2C0_BASE, reg);   // only Dummy for Delay
176     I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_RECEIVE_START);
177     while(I2CMasterBusy(I2C0_BASE));
178     received = I2CMasterDataGet(I2C0_BASE) << 8; // MSB
179
180     I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_RECEIVE_FINISH);
181     while(I2CMasterBusBusy(I2C0_BASE) | I2CMasterBusy(I2C0_BASE));
182     return received + I2CMasterDataGet(I2C0_BASE);
183 }
184
185 // reseives data from Slave-Register, with Slave-Address slaveAddr
186 uint32_t I2C_MasterReseiveMulti(uint8_t slaveAddr, uint8_t reg)
187 {
188     uint32_t temp;
189     // Safty, when Bus not ready
190     while(I2CMasterBusBusy(I2C0_BASE) | I2CMasterBusy(I2C0_BASE));
191
192     // Set Register to read from Slave
193     I2CMasterSlaveAddrSet(I2C0_BASE, slaveAddr, SEND);
194     I2CMasterDataPut(I2C0_BASE, reg);
195     I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_SINGLE_SEND);
196     while(I2CMasterBusBusy(I2C0_BASE) | I2CMasterBusy(I2C0_BASE));
197
198     // Read Register
199     I2CMasterSlaveAddrSet(I2C0_BASE, slaveAddr, RECEIVE);
200     I2CMasterDataPut(I2C0_BASE, reg);   // only Dummy for Delay
201     I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_RECEIVE_START);
202     while(I2CMasterBusBusy(I2C0_BASE) | I2CMasterBusy(I2C0_BASE));
203     temp = I2CMasterDataGet(I2C0_BASE); // First Byte
204
205     I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_RECEIVE_CONT);
206     while(I2CMasterBusBusy(I2C0_BASE) | I2CMasterBusy(I2C0_BASE));
207     temp += ( I2CMasterDataGet(I2C0_BASE) << 8); //Second and First Byte
```

```
208
209     I2CMasterControl(I2C0_BASE, I2C_MASTER_CMD_BURST_RECEIVE_FINISH);
210     while(I2CMasterBusBusy(I2C0_BASE) | I2CMasterBusy(I2C0_BASE));
211     return temp + ( I2CMasterDataGet(I2C0_BASE) << 16); //Third and Second and
            First Byte
212 }
213
214 // wait until I2C-Bus free
215 inline void I2C_WaitForBusFree(uint32_t busBaseAddr)
216 {
217     //
218     // Delay until transmission completes
219     //
220     while(I2CMasterBusBusy(I2C0_BASE) | I2CMasterBusy(I2C0_BASE));
221 }
```

Listing B.10: Mikrocontroller: i2c_master.h

```
1  /*
2   * i2c_master.h
3   *
4   *  Created on: 22.04.2015
5   *      Author: Torben
6   *
7   *  ToDo: I2C_MasterSend, Multi Byte Send/Reseive
8   *
9   */
10
11 #ifndef I2C_MASTER_H_
12 #define I2C_MASTER_H_
13
14 #define RECEIVE true
15 #define SEND false
16
17 #define I2C_ADDR_PRESSURE_SENSOR_FREESCALE 0x60 // 7bit address 0x60, 8bit
       address 0xC0
18 #define I2C_ADDR_RTC 0x68 // Addresse of RTC-Clock
19 #define I2C_ADDR_MPU9250 0x69 // Addresse of MPU9250 Acc/Gyro/Mag
20 #define I2C_ADDR_PRESSURE_SENSOR_BMP180 0x77 // Addresse of Bosch BMP180 Pressure
       Sensor
21 #define I2C_ADDR_ACC_SENSOR_LSM303DLHC 0x19 // Acc-Addresse of ST LSM303DLHC Acc/
       Mag Sensor
22 #define I2C_ADDR_MAG_SENSOR_LSM303DLHC 0x1e // Mag-Addresse of ST LSM303DLHC Acc/
       Mag Sensor
23
24
25 #include <stdint.h>
26 #include <stdbool.h>
27
```

```
28 #include <stdarg.h>
29
30 #include "i2c0_pin.h"
31
32 // Specific Sensor Headers
33 #include "mpl3115_my.h"
34 #include "shtc1.h"
35
36 #include "inc/hw_types.h"
37 #include "inc/hw_memmap.h"
38 #include "driverlib/i2c.h"
39 #include "driverlib/sysctl.h"
40 #include "driverlib/rom.h"
41
42 void I2C_MasterConfig(void);
43
44 void I2CSend(uint8_t slave_addr, uint8_t num_of_args, ...);
45
46 void I2C_MasterSend(uint8_t slaveAddr, uint8_t reg, uint8_t data);
47 void I2C_MasterSendSlaveDebug(uint8_t slaveAddr, uint8_t reg, uint8_t data);
48 uint8_t I2C_MasterReseive(uint8_t slaveAddr, uint8_t reg);
49 uint16_t I2C_MasterReseive16bitReg(uint8_t slaveAddr, uint8_t reg, uint8_t data);
50 inline void I2C_WaitForBusFree(uint32_t busBaseAddr);
51
52 // reseives 2 bytes of data from Slave-Register, with Slave-Address slaveAddr
53 uint32_t I2C_MasterReseiveMulti(uint8_t slaveAddr, uint8_t reg);
54
55
56 #endif /* I2C_MASTER_H_ */
```

Listing B.11: Mikrocontroller: i2c_slave.c

```
1  /*
2   * i2c_master.c
3   *
4   *  Created on: 22.04.2015
5   *      Author: Torben
6   *
7   *  ToDo: I2C_MasterSend, Multi Byte Send/Reseive
8   *
9   */
10
11
12 #include "i2c_slave.h"
13
14
15 // Configures I2C-Modul 1 to Slave-Mode
16 // on Ports PA6 = SCL and PA7 = SDA with adresse 0x42
17 void I2C_SlaveConfig(void)
```

```
18 {
19   int i = 0;
20
21     //enable I2C module 1
22     SysCtlPeripheralEnable(SYSCTL_PERIPH_I2C1);
23
24     //reset module
25     SysCtlPeripheralReset(SYSCTL_PERIPH_I2C1);
26
27   // Initialize I2C1-Pins
28   I2C1PortFunctionInit();
29
30   // Initialize I2C1 as slave with 7bit adresse 0x42
31   I2CSlaveInit( I2C1_BASE, 0x42);
32
33   // Register Interrupt Handler
34   I2CIntRegister( I2C1_BASE, I2C_Slave_IntHandler );
35   I2CSlaveIntClear( I2C1_BASE );
36
37   I2CSlaveEnable( I2C1_BASE );
38
39   /*ui32I2C_SlaveCount = 0;
40   ui32I2C_SlaveData = 0;
41   ui32I2C_SlaveIntStatus[0] = 0;
42   ui32I2C_SlaveIntStatus[1] = 0;
43 */
44   // Initialize I2C Slave Registers
45   ui8I2CRegister[0] = 0xFF; // not used
46   ui8I2CRegister[1] = 0x04; // Status
47   ui8I2CRegister[2] = 0x0A; // ID
48   for (i = 3; i < 0x5b; i++)
49   {
50     ui8I2CRegister[i] = 0;
51   }
52   ui8I2CRegister[5] = 0x01; // Mode
53   ui8I2CRegisterIndex = 0;
54   ui32I2C_SlaveFirstByte = FIRSTBYTE;
55 }
56
57 void I2C_Slave_IntHandler(void)
58 {
59   ui32I2C_SlaveStatus = I2CSlaveStatus( I2C1_BASE );
60   I2CSlaveIntClear( I2C1_BASE );
61
62   switch (ui32I2C_SlaveStatus)
63   {
64     case I2C_SLAVE_ACT_RREQ:      ;
65
```

```
66      case I2C_SLAVE_ACT_RREQ_FBR:   if ( ui32I2C_SlaveFirstByte == FIRSTBYTE)
67                          {    // when firstbyte of sequenz, set register to work on
68                            ui8I2CRegisterIndex = I2CSlaveDataGet( I2C1_BASE );
69                            ui32I2C_SlaveFirstByte = NOT_FIRSTBYTE;
70                          }
71                          else if ( ( ui8I2CRegisterIndex > 2 ) && (
                                ui8I2CRegisterIndex < 51 ) )
72                          {    // when in writable register range, write reseived byte
                                to register
73                            ui8I2CRegister[ ui8I2CRegisterIndex ] = I2CSlaveDataGet(
                                  I2C1_BASE );
74                            ui8I2CRegisterIndex = 0;
75                            ui32I2C_SlaveFirstByte = FIRSTBYTE;
76                          }
77                          else
78                          {    // if index is out of range, reset index to 0 and set
                                First Byte Flag
79                            ui8I2CReseived = I2CSlaveDataGet( I2C1_BASE );
80                            ui8I2CRegisterIndex = 0;
81                            ui32I2C_SlaveFirstByte = FIRSTBYTE;
82                          }
83                          break;
84      case I2C_SLAVE_ACT_TREQ:     if ( ui8I2CRegisterIndex < 0x5b ) // when in
            register range, send register
85                          {
86                            I2CSlaveDataPut( I2C1_BASE, ui8I2CRegister[
                                  ui8I2CRegisterIndex ]);
87                            ui8I2CRegisterIndex++;
88                          }
89                          else  // when out of range, reset to index 0 and send reg 0
                                x00
90                          {
91                            ui8I2CRegisterIndex = 0;
92                            I2CSlaveDataPut( I2C1_BASE, ui8I2CRegister[
                                  ui8I2CRegisterIndex ]);
93                          }
94                          ui32I2C_SlaveFirstByte = FIRSTBYTE;
95                          ui8I2CRegister[1] &= ~0x02; // reset New Sample Bit in Status
                                Reg.
96                          GPIO_PORTF_DATA_R &= ~0x04; // reset New Sample Pin F2
97                          break;
98      default:;
99    }
100 }
```

Listing B.12: Mikrocontroller: i2c_slave.h

```
1  /*
2   * i2c_master.h
```

```
3   *
4   *  Created on: 22.04.2015
5   *      Author: Torben
6   *
7   *  ToDo: I2C_MasterSend, Multi Byte Send/Reseive
8   *
9   */
10
11  #ifndef I2C_SLAVE_H_
12  #define I2C_SLAVE_H_
13
14  #define RECEIVE true
15  #define SEND false
16  #define FIRSTBYTE 0
17  #define NOT_FIRSTBYTE 1
18
19  #include <stdint.h>
20  #include <stdbool.h>
21
22  #include "inc/tm4c123gh6pm.h"
23  #include "inc/hw_types.h"
24  #include "inc/hw_memmap.h"
25  #include "driverlib/i2c.h"
26  #include "driverlib/sysctl.h"
27  #include "driverlib/rom.h"
28  #include "driverlib/interrupt.h"
29
30  #include "i2c1_pin.h"
31
32  /*volatile uint32_t ui32I2C_SlaveCount;
33  volatile uint32_t ui32I2C_SlaveData;
34  volatile uint32_t ui32I2C_SlaveIntStatus[2];*/
35
36  volatile uint32_t ui32I2C_SlaveStatus;
37
38  volatile uint8_t ui8I2CRegister[0x5b]; // all Registers used by I2C-Slave
39  volatile uint8_t ui8I2CReseived; // last Byte reseived
40  volatile uint8_t ui8I2CRegisterIndex; // i2c Register to use
41  volatile uint32_t ui32I2C_SlaveFirstByte; // Type of last action of I2C-Slave, 1
        = reseive and 0 = send
42
43  void I2C_SlaveConfig(void);
44
45  void I2C_Slave_IntHandler(void);
46
47  #endif /* I2C_SLAVE_H_ */
```

Listing B.13: Mikrocontroller: i2c0_pin.c

```
1  //*****************************************************************************
2  // Copyright (c) 2014 Texas Instruments Incorporated.  All rights reserved.
3  // Software License Agreement
4  //
5  //   Redistribution and use in source and binary forms, with or without
6  //   modification, are permitted provided that the following conditions
7  //   are met:
8  //
9  //   Redistributions of source code must retain the above copyright
10 //   notice, this list of conditions and the following disclaimer.
11 //
12 //   Redistributions in binary form must reproduce the above copyright
13 //   notice, this list of conditions and the following disclaimer in the
14 //   documentation and/or other materials provided with the
15 //   distribution.
16 //
17 //   Neither the name of Texas Instruments Incorporated nor the names of
18 //   its contributors may be used to endorse or promote products derived
19 //   from this software without specific prior written permission.
20 //
21 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
22 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
23 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
24 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
25 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
26 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
27 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
28 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
29 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
30 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
31 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
32 //
33 // This file was automatically generated by the Tiva C Series PinMux Utility
34 // Version: 1.0.4
35 //
36 //*****************************************************************************
37
38 #include <stdint.h>
39 #include <stdbool.h>
40 #include "i2c0_pin.h"
41 #include "inc/hw_types.h"
42 #include "inc/hw_memmap.h"
43 #include "inc/hw_gpio.h"
44 #include "driverlib/sysctl.h"
45 #include "driverlib/pin_map.h"
46 #include "driverlib/rom_map.h"
47 #include "driverlib/gpio.h"
```

```
48
49 //*****************************************************************************
50 void
51 I2C0PortFunctionInit(void)
52 {
53     //
54     // Enable Peripheral Clocks
55     //
56     MAP_SysCtlPeripheralEnable(SYSCTL_PERIPH_I2C0);
57     MAP_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
58
59     //
60     // Enable pin PB2 for I2C0 I2C0SCL
61     //
62     MAP_GPIOPinConfigure(GPIO_PB2_I2C0SCL);
63     MAP_GPIOPinTypeI2CSCL(GPIO_PORTB_BASE, GPIO_PIN_2);
64
65     //
66     // Enable pin PB3 for I2C0 I2C0SDA
67     //
68     MAP_GPIOPinConfigure(GPIO_PB3_I2C0SDA);
69     MAP_GPIOPinTypeI2C(GPIO_PORTB_BASE, GPIO_PIN_3);
70 }
```

Listing B.14: Mikrocontroller: i2c0_pin.h

```
1  //*****************************************************************************
2  // Copyright (c) 2014 Texas Instruments Incorporated.  All rights reserved.
3  // Software License Agreement
4  //
5  //   Redistribution and use in source and binary forms, with or without
6  //   modification, are permitted provided that the following conditions
7  //   are met:
8  //
9  //   Redistributions of source code must retain the above copyright
10 //   notice, this list of conditions and the following disclaimer.
11 //
12 //   Redistributions in binary form must reproduce the above copyright
13 //   notice, this list of conditions and the following disclaimer in the
14 //   documentation and/or other materials provided with the
15 //   distribution.
16 //
17 //   Neither the name of Texas Instruments Incorporated nor the names of
18 //   its contributors may be used to endorse or promote products derived
19 //   from this software without specific prior written permission.
20 //
21 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
22 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
23 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
```

```
24 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
25 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
26 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
27 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
28 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
29 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
30 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
31 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
32 //
33 // This file was automatically generated by the Tiva C Series PinMux Utility
34 // Version: 1.0.4
35 //
36 //*****************************************************************************
37
38 #ifndef __I2C0_PIN_H__
39 #define __I2C0_PIN_H__
40
41 extern void I2C0PortFunctionInit(void);
42
43 #endif //  __I2C0_PIN_H__
```

Listing B.15: Mikrocontroller: i2c1_pin.c

```
1  //*****************************************************************************
2  // Copyright (c) 2014 Texas Instruments Incorporated.  All rights reserved.
3  // Software License Agreement
4  //
5  //   Redistribution and use in source and binary forms, with or without
6  //   modification, are permitted provided that the following conditions
7  //   are met:
8  //
9  //   Redistributions of source code must retain the above copyright
10 //   notice, this list of conditions and the following disclaimer.
11 //
12 //   Redistributions in binary form must reproduce the above copyright
13 //   notice, this list of conditions and the following disclaimer in the
14 //   documentation and/or other materials provided with the
15 //   distribution.
16 //
17 //   Neither the name of Texas Instruments Incorporated nor the names of
18 //   its contributors may be used to endorse or promote products derived
19 //   from this software without specific prior written permission.
20 //
21 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
22 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
23 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
24 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
25 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
26 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
```

```
27 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
28 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
29 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
30 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
31 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
32 //
33 // This file was automatically generated by the Tiva C Series PinMux Utility
34 // Version: 1.0.4
35 //
36 //*****************************************************************************
37
38 #include <stdint.h>
39 #include <stdbool.h>
40 #include "i2c1_pin.h"
41 #include "inc/hw_types.h"
42 #include "inc/hw_memmap.h"
43 #include "inc/hw_gpio.h"
44 #include "driverlib/sysctl.h"
45 #include "driverlib/pin_map.h"
46 #include "driverlib/rom_map.h"
47 #include "driverlib/gpio.h"
48
49 //*****************************************************************************
50 void
51 I2C1PortFunctionInit(void)
52 {
53     //
54     // Enable Peripheral Clocks
55     //
56     MAP_SysCtlPeripheralEnable(SYSCTL_PERIPH_I2C1);
57     MAP_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
58
59     //
60     // Enable pin PA7 for I2C1 I2C1SDA
61     //
62     MAP_GPIOPinConfigure(GPIO_PA7_I2C1SDA);
63     MAP_GPIOPinTypeI2C(GPIO_PORTA_BASE, GPIO_PIN_7);
64
65     //
66     // Enable pin PA6 for I2C1 I2C1SCL
67     //
68     MAP_GPIOPinConfigure(GPIO_PA6_I2C1SCL);
69     MAP_GPIOPinTypeI2CSCL(GPIO_PORTA_BASE, GPIO_PIN_6);
70 }
```

Listing B.16: Mikrocontroller: i2c1_pin.h

```
1 //*****************************************************************************
2 // Copyright (c) 2014 Texas Instruments Incorporated.  All rights reserved.
```

```
3  // Software License Agreement
4  //
5  //   Redistribution and use in source and binary forms, with or without
6  //   modification, are permitted provided that the following conditions
7  //   are met:
8  //
9  //   Redistributions of source code must retain the above copyright
10 //   notice, this list of conditions and the following disclaimer.
11 //
12 //   Redistributions in binary form must reproduce the above copyright
13 //   notice, this list of conditions and the following disclaimer in the
14 //   documentation and/or other materials provided with the
15 //   distribution.
16 //
17 //   Neither the name of Texas Instruments Incorporated nor the names of
18 //   its contributors may be used to endorse or promote products derived
19 //   from this software without specific prior written permission.
20 //
21 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
22 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
23 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
24 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
25 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
26 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
27 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
28 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
29 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
30 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
31 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
32 //
33 // This file was automatically generated by the Tiva C Series PinMux Utility
34 // Version: 1.0.4
35 //
36 //*****************************************************************************
37
38 #ifndef __I2C1_PIN_H__
39 #define __I2C1_PIN_H__
40
41 extern void I2C1PortFunctionInit(void);
42
43 #endif //  __I2C1_PIN_H__
```

Listing B.17: Mikrocontroller: main.c

```
1  //*****************************************************************************
2  //
3  // blinky.c - Simple example to blink the on-board LED.
4  //
5  // Copyright (c) 2012-2014 Texas Instruments Incorporated.  All rights reserved.
```

```
6  // Software License Agreement
7  //
8  // Texas Instruments (TI) is supplying this software for use solely and
9  // exclusively on TI's microcontroller products. The software is owned by
10 // TI and/or its suppliers, and is protected under applicable copyright
11 // laws. You may not combine this software with "viral" open-source
12 // software in order to form a larger program.
13 //
14 // THIS SOFTWARE IS PROVIDED "AS IS" AND WITH ALL FAULTS.
15 // NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT
16 // NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
17 // A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. TI SHALL NOT, UNDER ANY
18 // CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL, OR CONSEQUENTIAL
19 // DAMAGES, FOR ANY REASON WHATSOEVER.
20 //
21 // This is part of revision 2.1.0.12573 of the EK-TM4C123GXL Firmware Package.
22 //
23 //*****************************************************************************
24
25 #include <stdint.h>
26 #include "inc/tm4c123gh6pm.h"
27
28 //#include "clockset.h"
29 #include "uart.h"
30 #include "systick.h"
31 #include "i2c_master.h"
32 #include "i2c_slave.h"
33 #include "pwm.h"
34 #include "adc.h"
35 #include "gpio.h"
36
37
38 //*****************************************************************************
39 //
40 //! \addtogroup example_list
41 //! <h1>Blinky (blinky)</h1>
42 //!
43 //! A very simple example that blinks the on-board LED using direct register
44 //! access.
45 //
46 //*****************************************************************************
47
48 //*****************************************************************************
49 //
50 // Blink the on-board LED.
51 //
52 //*****************************************************************************
53
```

```
54  int
55  main(void)
56  {
57      volatile uint32_t ui32Loop = 0, ui32Old = 0, ui32Reseived = 0, ui32alive = 0;
58      char test[30];
59
60
61      //
62      // Set the clocking to run directly from the crystal.
63      //
64      //ROM_SysCtlClockSet(SYSCTL_SYSDIV_1 | SYSCTL_USE_OSC | SYSCTL_XTAL_16MHZ |
65      //                   SYSCTL_OSC_MAIN);
66
67      //
68      // Setup the system clock to run at 40 MHz from PLL with crystal reference
69      //
70      //ROM_SysCtlClockSet(SYSCTL_SYSDIV_5 | SYSCTL_USE_PLL | SYSCTL_XTAL_16MHZ |
71      //                   SYSCTL_OSC_MAIN);
72
73      //
74      // Set the system clock to run at 80Mhz off PLL with external crystal as
75      // reference.
76      //
77      ROM_SysCtlClockSet(SYSCTL_SYSDIV_2_5 | SYSCTL_USE_PLL | SYSCTL_XTAL_16MHZ |
78                         SYSCTL_OSC_MAIN);
79
80
81      //
82      // Enable the GPIO port that is used for the on-board LED.
83      //
84      SYSCTL_RCGC2_R = SYSCTL_RCGC2_GPIOF;
85
86      //
87      // Do a dummy read to insert a few cycles after enabling the peripheral.
88      //
89      ui32Loop = SYSCTL_RCGC2_R;
90
91      //
92      // Enable the GPIO pin for the RGB-LED (PF1-3) + Pins F0 + F4.  Set the
            direction as output, and
93      // enable the GPIO pin for digital function.
94      //
95      // PF1 I2C New Sample Interrupt
96      GPIO_PORTF_DIR_R = 0x08+0x04+0x02+0x10+0x01;
97      GPIO_PORTF_DEN_R = 0x08+0x04+0x02+0x10+0x01;
98
99      //
100     // Initialize the UART.
```

```
101         //
102         ConfigureUART();
103
104         //
105         // Hello!
106         //
107         UARTprintf("Hello, world!\n");
108         UARTprintf("MPU running at %dMHz\n", ROM_SysCtlClockGet());
109
110         //
111         // Configer GPIOs for Power Control
112         // Fan = PA2, I2C Sensors = PA3, Analog Interface = PA4
113         //
114         GpioConfig();
115
116         GPIO_PORTA_DATA_R |= 0x04; // start Fan
117         GPIO_PORTA_DATA_R |= 0x08; // Enable I2C-Sensors
118         GPIO_PORTA_DATA_R |= 0x10; // Enable Analog Interface
119
120         //
121         // Set up and enable the SysTick timer.  It will be used as a reference
122         // for delay loops in the interrupt handlers.  The SysTick timer period
123         // will be set up for 1/4 second.
124         //
125         SysTickConfig();
126
127         //
128         // I2C Slave Setup
129         // Ports: PA6 = SCL, PA7 = SDA
130         I2C_SlaveConfig();
131         //
132         // I2C Master Setup
133         // Ports: PB2 = SCL, PB3 = SDA, 100000kHz
134         I2C_MasterConfig();
135
136         //
137         // PWM0 Setup
138         // Ports: PB6 = PWM0.0 PC4 = PWM0.6
139         Pwm0Config();
140
141         //
142         // ADC Setup
143         // AINs:   0,  1,  2,  8,  9, 10, 11
144         // GPIOs: E3, E2, E1, E5, E4, B4, B5
145         // and internal Temperature Sensor
146         AdcConfig();
147
148         //
```

```
149       // Start PWM0
150       // Pwm0Start();
151
152       //
153       // Systick, ADC; I2C1_Slave and
154       // Master Interrupt Enable
155       I2CSlaveIntEnable(I2C1_BASE);
156       ROM_ADCIntEnable(ADC0_BASE, 0);
157       ROM_SysTickIntEnable();
158       ROM_IntMasterEnable();
159
160
161       // Debug
162       GPIO_PORTF_DATA_R = 0x00;
163
164  //     Pwm0Start();
165
166      // I2C_MasterSend(0x42,0x05,0x00); // single shot
167      // I2C_MasterSend(0x42,0x03,100); // sample periode
168      // I2C_MasterSend(0x42,0x04,0x01); // Start
169
170
171
172
173
174
175
176       //
177       // Loop forever.
178       //
179       while(1)
180       {
181
182         if (ui32alive)
183         {
184           ui32alive = 0;
185           UARTprintf("alive\nA\n");
186         }
187         else
188         {
189           ui32alive = 1;
190           UARTprintf("alive\nB\n");
191         }
192       /*  //
193           // Turn on the LED.
194           //
195           GPIO_PORTF_DATA_R &= ~(0x02);
196           GPIO_PORTF_DATA_R |= 0x08;
```

```
197
198             //
199             // Delay for a bit.
200             //
201     //        for(ui32Loop = 0; ui32Loop < 2000000; ui32Loop++)
202     //        {
203     //        }
204
205             //
206             // Turn off the LED.
207             //
208             GPIO_PORTF_DATA_R &= ~(0x08);
209             GPIO_PORTF_DATA_R |= 0x04;
210
211             //
212             // Delay for a bit.
213             //
214     //        for(ui32Loop = 0; ui32Loop < 2000000; ui32Loop++)
215     //        {
216     //        }
217
218             //
219             // Turn off the LED.
220             //
221             GPIO_PORTF_DATA_R &= ~(0x04);
222             GPIO_PORTF_DATA_R |= 0x02;
223
224             //
225             // Delay for a bit.
226             //
227     //        for(ui32Loop = 0; ui32Loop < 2000000; ui32Loop++)
228     //        {
229     //        }*/
230             // Debug SystickTimer
231             //ui32Old = ui32SysTickIntCount;
232             //UARTprintf("SysTickInterrupts: \%d\n", ui32Old);
233
234     /*        // Debug Pressure_Sensor
235             ui32Reseived = I2C_MasterReseive(I2C_ADDR_PRESSURE_SENSOR, 0x0C); // ask
                    the senor for its ID
236             UARTprintf("Pressure Sensor ID: \%x\n", ui32Reseived);
237             ui32Reseived = I2C_MasterReseive(I2C_ADDR_PRESSURE_SENSOR, 0x13); // ask
                    the senor for its ID
238             UARTprintf("Pressure Sensor Param: \%x\n", ui32Reseived);
239     */
240             // Trigger
241             // GPIO_PORTF_DATA_R |= 0x10; // Trigger steigende Flanke
242
```

```
243         //I2C_MasterSendDebug(I2C_ADDR_RTC, 0x01);   // Debug I2C Send
244         //UARTprintf("Test Send I2C to I2C_ADDR_PRESSURE_SENSOR\n");
245         //I2C_MasterSendDebug(0x00, 'g');   // Debug I2C Send
246         //UARTprintf("Test Send I2C Broadcast\n");
247
248         //for(ui32Loop = 0; ui32Loop < 500; ui32Loop++);
249         //I2C_MasterSendDebug(I2C_ADDR_RTC, 0x02);
250         // for(ui32Loop = 0; ui32Loop < 500; ui32Loop++);
251         //I2C_MasterSend(0x11, 0x22, 0x44 );
252         //ui32Reseived = I2C_MasterReseive(I2C_ADDR_PRESSURE_SENSOR_BMP180,0xD0);
                //Read Sensor ID
253
254         //ui32Reseived = I2C_MasterReseive(I2C_ADDR_MPU9250,117); //Read Sensor
              ID
255
256         //Trigger
257         // GPIO_PORTF_DATA_R &= ~(0x10); // Trigger fallende Flanke
258
259         UARTprintf("L1 AN0  PE3: %5d mV\n",((ui32AdcSamples[0][
              ui32AdcSampleReading][0]/(ui32AdcSampleLengh))*3000)/(4095));
260         UARTprintf("L1 AN1  PE2: %5d mV\n",((ui32AdcSamples[0][
              ui32AdcSampleReading][1]/(ui32AdcSampleLengh))*3000)/(4095));
261         UARTprintf("L1 AN2  PE1: %5d mV\n",((ui32AdcSamples[0][
              ui32AdcSampleReading][2]/(ui32AdcSampleLengh))*3000)/(4095));
262         UARTprintf("L1 AN8  PE5: %5d mV\n",((ui32AdcSamples[0][
              ui32AdcSampleReading][3]/(ui32AdcSampleLengh))*3000)/(4095));
263         UARTprintf("L1 AN9  PE4: %5d mV\n",((ui32AdcSamples[0][
              ui32AdcSampleReading][4]/(ui32AdcSampleLengh))*3000)/(4095));
264         UARTprintf("L1 AN10 PB4: %5d mV\n",((ui32AdcSamples[0][
              ui32AdcSampleReading][5]/(ui32AdcSampleLengh))*3000)/(4095));
265         UARTprintf("L1 AN11 PB5: %5d mV\n",((ui32AdcSamples[0][
              ui32AdcSampleReading][6]/(ui32AdcSampleLengh))*3000)/(4095));
266         UARTprintf("L1      Temp: %5d °C\n", 148 - ( ( 75 * 3 * (ui32AdcSamples
              [0][ui32AdcSampleReading][7]/(ui32AdcSampleLengh))) / (4096) ) );
267
268         UARTprintf("L2 AN0  PE3: %5d mV\n",((ui32AdcSamples[1][
              ui32AdcSampleReading][0]/(ui32AdcSampleLengh))*3000)/(4095));
269         UARTprintf("L2 AN1  PE2: %5d mV\n",((ui32AdcSamples[1][
              ui32AdcSampleReading][1]/(ui32AdcSampleLengh))*3000)/(4095));
270         UARTprintf("L2 AN2  PE1: %5d mV\n",((ui32AdcSamples[1][
              ui32AdcSampleReading][2]/(ui32AdcSampleLengh))*3000)/(4095));
271         UARTprintf("L2 AN8  PE5: %5d mV\n",((ui32AdcSamples[1][
              ui32AdcSampleReading][3]/(ui32AdcSampleLengh))*3000)/(4095));
272         UARTprintf("L2 AN9  PE4: %5d mV\n",((ui32AdcSamples[1][
              ui32AdcSampleReading][4]/(ui32AdcSampleLengh))*3000)/(4095));
273         UARTprintf("L2 AN10 PB4: %5d mV\n",((ui32AdcSamples[1][
              ui32AdcSampleReading][5]/(ui32AdcSampleLengh))*3000)/(4095));
```

```
274             UARTprintf("L2 AN11 PB5: %5d mV\n",((ui32AdcSamples[1][
                    ui32AdcSampleReading][6]/(ui32AdcSampleLengh))*3000)/(4095));
275             UARTprintf("L2      Temp: %5d °C\n", 148 - ( ( 75 * 3 * (ui32AdcSamples
                    [1][ui32AdcSampleReading][7]/(ui32AdcSampleLengh))) / (4096) ) );
276
277     //
278     //
279             UARTprintf("ADC count: %d\n",ui32AdcIntCount);
280
281
282     //          ui32Reseived = I2C_MasterReseive(I2C_ADDR_PRESSURE_SENSOR_BMP180,0xd0);
283     //          UARTprintf("I2C id: %x\n",ui32Reseived);
284     //          ui32Reseived = I2C_MasterReseive(I2C_ADDR_PRESSURE_SENSOR_BMP180,0xf4);
285     //          UARTprintf("I2C Con: %x\n",ui32Reseived);
286     //          ui32Reseived = I2C_MasterReseive(I2C_ADDR_PRESSURE_SENSOR_BMP180,0xf6);
287     //          UARTprintf("I2C MSB: %x\n",ui32Reseived);
288     //          ui32Reseived = I2C_MasterReseive(I2C_ADDR_PRESSURE_SENSOR_BMP180,0xf7);
289     //          UARTprintf("I2C LSB: %x\n",ui32Reseived);
290     //          ui32Reseived = I2C_MasterReseive(I2C_ADDR_ACC_SENSOR_LSM303DLHC,0x20);
291     //          UARTprintf("I2C ACC Reg1: %x\n",ui32Reseived);
292     //          ui32Reseived = ( I2C_MasterReseive(I2C_ADDR_ACC_SENSOR_LSM303DLHC,0x29)
            ) << 8;
293     //          ui32Reseived += I2C_MasterReseive(I2C_ADDR_ACC_SENSOR_LSM303DLHC,0x28);
294     //          UARTprintf("I2C ACC X: %d\n",ui32Reseived);
295
296
297     //          ui32Old++;
298     //      UARTprintf("I2C Slave masked Int Status: %x\n",ui32I2C_SlaveIntStatus[0]);
299     //      UARTprintf("I2C Slave unmasked Int Status: %x\n",ui32I2C_SlaveIntStatus[1])
            ;
300     //      UARTprintf("I2C Slave Reseived Bytes Count: %d\n",ui32I2C_SlaveCount);
301     //      UARTprintf("I2C Slave Reseived Byte: %d\n",ui32I2C_SlaveData);
302     //
303     //      ui32Reseived = I2C_MasterReseive(MPL3115_IIC_ADDRESS, 0xc4);
304     //      UARTprintf("MPL3115_IIC_ADDRESS: %x\n",ui32Reseived);
305     //      ui32Reseived = I2C_MasterReseive16bitReg(I2C_ADDR_HUMIDITY_SENSOR_SHTC1,
            SHTC1_ID_REG_MSB, SHTC1_ID_REG_LSB);
306     //      UARTprintf("SHTC1_ID_REG_LSB: %x\n",ui32Reseived);
307     //      ui32Reseived = I2C_MasterReseive(0x42,0x01);
308     //      UARTprintf("I2C Slave sended Byte: %x\n",ui32Reseived);
309     //      ui32Reseived = I2C_MasterReseive(0x42,0x10);
310     //      UARTprintf("I2C Slave sended Byte: %x\n",ui32Reseived);
311
312     //          if (ui32Old == 0)
313     //          {
314     //            ui32Old = 1;
315     //            Pwm0Stop();
316     //          }
```

```
317 //          else
318 //          {
319 //            ui32Old = 0;
320 //            Pwm0Start();
321 //          }
322 //          UARTprintf("Status %x\n", ui8I2CRegister[0x01]);
323           for(ui32Loop = 0; ui32Loop < 5000000L; ui32Loop++);
324           // // Stop
325       }
326 }
```

Listing B.18: Mikrocontroller: mpl3115_my.c

```
1  /*
2   * mpl3115_my.c
3   *
4   *  Created on: 03.08.2016
5   *      Author: goeth
6   */
7
8  #include "mpl3115_my.h"
9
10 //
11 // returns the Pressure in Pascals/2^2
12 inline uint32_t Mpl3115PressureConversion( uint8_t pressureMsb, uint8_t
       pressureCsb, uint8_t pressureLsb )
13 {
14   return ( ( (uint32_t) pressureMsb ) << 12 ) | ( ( (uint32_t) pressureCsb ) << 4
         ) | ( ( (uint32_t) pressureLsb ) >> 4 );
15 }
16
17 //
18 // returns the Temperatur in °C/2^4
19 inline int32_t Mpl3115TemperatureConversion( uint8_t temperatureMsb, uint8_t
       temperatureLsb )
20 {
21   return (int32_t) ( (int16_t) ( ( (uint16_t) temperatureMsb ) << 8 ) | ( ( (
         uint16_t) temperatureLsb ) ) >> 4 );
22 }
```

Listing B.19: Mikrocontroller: mpl3115_my.h

```
1  /*
2   * mpl3115_my.h
3   *
4   *  Created on: 03.08.2016
5   *      Author: goeth
6   */
7
```

```
8  #ifndef MPL3115_MY_H_
9  #define MPL3115_MY_H_
10
11 #include <stdint.h>
12 #include "mpl3115.h"
13
14
15 inline uint32_t Mpl3115PressureConversion( uint8_t pressureMsb, uint8_t
       pressureCsb, uint8_t pressureLsb );
16 inline int32_t Mpl3115TemperatureConversion( uint8_t temperatureMsb, uint8_t
       temperatureLsb );
17
18 #endif /* MPL3115_MY_H_ */
```

Listing B.20: Mikrocontroller: pwm.c

```
1  /*
2   * pwm.c
3   *
4   *  Created on: 27.05.2015
5   *      Author: Torben
6   */
7
8  #include "pwm.h"
9
10
11 void Pwm0Config()
12 {
13    int pwm0Period = 0;
14    //
15    // Configure P
16    Pwm0PortFunctionInit();
17
18
19    //
20    // Set PWMClock
21    ROM_SysCtlPWMClockSet(SYSCTL_PWMDIV_64);
22
23    //
24    // Configure the PWM generator 0 for count down mode with immediate updates
25    // to the parameters.
26    //
27    PWMGenConfigure(PWM0_BASE, PWM_GEN_0, PWM_GEN_MODE_DOWN | PWM_GEN_MODE_NO_SYNC)
          ;
28    PWMGenConfigure(PWM0_BASE, PWM_GEN_3, PWM_GEN_MODE_DOWN | PWM_GEN_MODE_NO_SYNC)
          ;
29
30    //
31    // Set the period. For a 100 KHz frequency, the period = 1/100,000, or 10
```

```
32      // microseconds. For a 20 MHz clock, this translates to 200 clock ticks.
33      // Use this value to set the period.
34      //
35      pwm0Period = ROM_SysCtlClockGet() / (PWM0_FREQUENCY * 64);
36      PWMGenPeriodSet(PWM0_BASE, PWM_GEN_0, pwm0Period);
37      PWMGenPeriodSet(PWM0_BASE, PWM_GEN_3, pwm0Period);
38
39      //
40      // Set the pulse width of PWM0.0 for a 33.3% duty cycle.
41      //
42      PWMPulseWidthSet(PWM0_BASE, PWM_OUT_0, pwm0Period / PWM0_DUTYCYCLE_DIV );
43
44      //
45      // Set the pulse width of PWM0.1 for a 25% duty cycle.
46      // This is used as delayed Trigger for the ADC
47      //
48      PWMPulseWidthSet(PWM0_BASE, PWM_OUT_1, pwm0Period / PWM0_DUTYCYCLE_DIV_ADC );
49
50      //
51      // Set the pulse width of PWM0.6 for a 66.6% duty cycle.
52      //
53      PWMPulseWidthSet(PWM0_BASE, PWM_OUT_6, (pwm0Period * ( PWM0_DUTYCYCLE_DIV - 1 )
            ) / PWM0_DUTYCYCLE_DIV );
54
55      //
56      // Set the pulse width of PWM0.7 for a 91.6% duty cycle.
57      // This is used as delayed Trigger for the ADC
58      //
59      PWMPulseWidthSet(PWM0_BASE, PWM_OUT_7, ( ( pwm0Period * ( PWM0_DUTYCYCLE_DIV -
            1 ) ) / PWM0_DUTYCYCLE_DIV ) + ( pwm0Period / PWM0_DUTYCYCLE_DIV_ADC ) );
60
61
62      //
63      // Invert output of PWM0.6
64      //
65      PWMOutputInvert(PWM0_BASE, PWM_OUT_0_BIT, false);
66      PWMOutputInvert(PWM0_BASE, PWM_OUT_1_BIT, false);
67      PWMOutputInvert(PWM0_BASE, PWM_OUT_6_BIT, true);
68      PWMOutputInvert(PWM0_BASE, PWM_OUT_7_BIT, false);
69
70
71      //
72      // Set Timer Count B (PWM0.1 and PWM0.7) to trigger ADC
73      //
74      PWMGenIntTrigEnable(PWM0_BASE, PWM_GEN_0, PWM_TR_CNT_BD);
75      PWMGenIntTrigEnable(PWM0_BASE, PWM_GEN_3, PWM_TR_CNT_BD);
76  }
77
```

```
78  void Pwm0Start()
79  {
80    //
81    // Start the timer in generator 0 + 3.
82    //
83    PWMGenEnable(PWM0_BASE, PWM_GEN_0);
84    PWMGenEnable(PWM0_BASE, PWM_GEN_3);
85
86    //
87    // Syncing generators 0 + 3
88    //
89    PWMSyncTimeBase(PWM0_BASE, ( PWM_GEN_0_BIT | PWM_GEN_3_BIT ) );
90
91    //
92    // Enable the outputs.
93    //
94    PWMOutputState(PWM0_BASE, (PWM_OUT_0_BIT | PWM_OUT_6_BIT), true); //
          PWM_OUT_1_BIT |   | PWM_OUT_7_BIT
95  }
96
97  void Pwm0Stop()
98  {
99    //
100   // Stop the timer in generator 0 + 3.
101   //
102   PWMGenDisable(PWM0_BASE, PWM_GEN_0);
103   PWMGenDisable(PWM0_BASE, PWM_GEN_3);
104   //
105   // Enable the outputs.
106   //
107   PWMOutputState(PWM0_BASE, (PWM_OUT_0_BIT | PWM_OUT_6_BIT), false); // |
          PWM_OUT_1_BIT   | PWM_OUT_7_BIT
108 }
109
110 void Pwm0IntHandler()
111 {
112   // not used at the moment
113 }
114
115 void Pwm1Config()
116 {
117   int pwm1Period = 0;
118   //
119   // Configure P
120   Pwm1PortFunctionInit();
121   //
122   // Configure the PWM generator 0 for count down mode with immediate updates
123   // to the parameters.
```

```
124     //
125     PWMGenConfigure(PWM1_BASE, PWM_GEN_0, PWM_GEN_MODE_DOWN | PWM_GEN_MODE_NO_SYNC)
            ;
126
127     //
128     // Set the period. For a 100 KHz frequency, the period = 1/100,000, or 10
129     // microseconds. For a 20 MHz clock, this translates to 200 clock ticks.
130     // Use this value to set the period.
131     //
132     pwm1Period = ROM_SysCtlClockGet() / PWM1_FREQUENCY;
133     PWMGenPeriodSet(PWM1_BASE, PWM_GEN_0, pwm1Period);
134
135     //
136     // Set the pulse width of PWM1.0 to ~0.32ms.
137     //
138     PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, pwm1Period / PWM1_DUTYCYCLE_DIV );
139
140     //
141     // Set the pulse width of PWM1.1 to ~28ms.
142     // This is used as delayed Trigger for the ADC
143     //
144     PWMPulseWidthSet(PWM1_BASE, PWM_OUT_1, pwm1Period / PWM1_DUTYCYCLE_DIV_ADC );
145
146     //
147     // Set Timer Count B (PWM1.1) to trigger ADC
148     //
149     PWMGenIntTrigEnable(PWM1_BASE, PWM_GEN_0, PWM_TR_CNT_BD);
150 }
151
152 void Pwm1Start()
153 {
154     //
155     // Start the timer in generator 0.
156     //
157     PWMGenEnable(PWM1_BASE, PWM_GEN_0);
158     //
159     // Enable the outputs.
160     //
161     PWMOutputState(PWM1_BASE, (PWM_OUT_0_BIT | PWM_OUT_1_BIT), true);
162 }
163
164 void Pwm1Stop()
165 {
166     //
167     // Stop the timer in generator 0.
168     //
169     PWMGenDisable(PWM1_BASE, PWM_GEN_0);
170     //
```

```
171     // Enable the outputs.
172     //
173     PWMOutputState(PWM1_BASE, (PWM_OUT_0_BIT | PWM_OUT_1_BIT), false);
174 }
175
176 void Pwm1IntHandler()
177 {
178     // not used at the moment
179 }
```

Listing B.21: Mikrocontroller: pwm.h

```
1  /*
2   * pwm.h
3   *
4   *  Created on: 27.05.2015
5   *      Author: Torben
6   */
7
8  #ifndef PWM_H_
9  #define PWM_H_
10
11 #define PWM0_FREQUENCY 5000 // Frequency of PWM0 in Hz
12 #define PWM0_DUTYCYCLE_DIV 3 // Dutycycle as Fraction of full period: 100% = 1,
       50% = 2, 25% = 4
13 #define PWM0_DUTYCYCLE_DIV_ADC 4 // Dutycycle as Fraction of full period: 100% =
       1, 50% = 2, 25% = 4
14
15 #define PWM1_FREQUENCY 100 // Frequency of PWM1 in Hz
16 #define PWM1_DUTYCYCLE_DIV 31 // Dutycycle as Fraction of full period: ~0.32ms
17 #define PWM1_DUTYCYCLE_DIV_ADC 35 // Dutycycle as Fraction of full period: ~0.28
       ms
18
19 #include <stdint.h>
20 #include <stdbool.h>
21
22 #include "inc/tm4c123gh6pm.h"
23
24 #include "inc/hw_types.h"
25 #include "inc/hw_memmap.h"
26 #include "driverlib/pwm.h"
27 #include "driverlib/sysctl.h"
28 #include "driverlib/rom.h"
29
30
31 #include "pwm0_pin.h"
32 #include "pwm1_pin.h"
33
34 void Pwm0Config();
```

```
35 void Pwm0Start();
36 void Pwm0Stop();
37
38 void Pwm1Config();
39 void Pwm1Start();
40 void Pwm1Stop();
41
42 #endif /* PWM_H_ */
```

Listing B.22: Mikrocontroller: pwm0_pin.c

```
1  //*****************************************************************************
2  // Copyright (c) 2014 Texas Instruments Incorporated.  All rights reserved.
3  // Software License Agreement
4  //
5  //   Redistribution and use in source and binary forms, with or without
6  //   modification, are permitted provided that the following conditions
7  //   are met:
8  //
9  //   Redistributions of source code must retain the above copyright
10 //   notice, this list of conditions and the following disclaimer.
11 //
12 //   Redistributions in binary form must reproduce the above copyright
13 //   notice, this list of conditions and the following disclaimer in the
14 //   documentation and/or other materials provided with the
15 //   distribution.
16 //
17 //   Neither the name of Texas Instruments Incorporated nor the names of
18 //   its contributors may be used to endorse or promote products derived
19 //   from this software without specific prior written permission.
20 //
21 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
22 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
23 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
24 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
25 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
26 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
27 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
28 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
29 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
30 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
31 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
32 //
33 // This file was automatically generated by the Tiva C Series PinMux Utility
34 // Version: 1.0.4
35 //
36 //*****************************************************************************
37
38 #include <stdint.h>
```

```
39 #include <stdbool.h>
40 #include "pwm0_pin.h"
41 #include "inc/hw_types.h"
42 #include "inc/hw_memmap.h"
43 #include "inc/hw_gpio.h"
44 #include "driverlib/sysctl.h"
45 #include "driverlib/pin_map.h"
46 #include "driverlib/rom_map.h"
47 #include "driverlib/gpio.h"
48
49 //*****************************************************************************
50
51 //*****************************************************************************
52 //
53 // only using PWM0.0 on GPIO B6, PWM0.6 on GPIO C4 and PWM0.1 to debug on GPIO B7
54 //
55 //*****************************************************************************
56 void
57 Pwm0PortFunctionInit(void)
58 {
59     //
60     // Enable Peripheral Clocks
61     //
62     MAP_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM0);
63     MAP_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);
64     MAP_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC);
65     //MAP_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
66     //MAP_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);
67
68     //
69     // Enable pin PB4 for PWM0 M0PWM2
70     //
71     //MAP_GPIOPinConfigure(GPIO_PB4_M0PWM2);
72     //MAP_GPIOPinTypePWM(GPIO_PORTB_BASE, GPIO_PIN_4);
73
74     //
75     // Enable pin PC5 for PWM0 M0PWM7
76     //
77     MAP_GPIOPinConfigure(GPIO_PC5_M0PWM7);
78     MAP_GPIOPinTypePWM(GPIO_PORTC_BASE, GPIO_PIN_5);
79
80     //
81     // Enable pin PD2 for PWM0 M0FAULT0
82     //
83     //MAP_GPIOPinConfigure(GPIO_PD2_M0FAULT0);
84     //MAP_GPIOPinTypePWM(GPIO_PORTD_BASE, GPIO_PIN_2);
85
86     //
```

```
87      // Enable pin PC4 for PWM0 M0PWM6
88      //
89      MAP_GPIOPinConfigure(GPIO_PC4_M0PWM6);
90      MAP_GPIOPinTypePWM(GPIO_PORTC_BASE, GPIO_PIN_4);
91
92      //
93      // Enable pin PE5 for PWM0 M0PWM5
94      //
95      //MAP_GPIOPinConfigure(GPIO_PE5_M0PWM5);
96      //MAP_GPIOPinTypePWM(GPIO_PORTE_BASE, GPIO_PIN_5);
97
98      //
99      // Enable pin PB6 for PWM0 M0PWM0
100     //
101     MAP_GPIOPinConfigure(GPIO_PB6_M0PWM0);
102     MAP_GPIOPinTypePWM(GPIO_PORTB_BASE, GPIO_PIN_6);
103
104     //
105     // Enable pin PB5 for PWM0 M0PWM3
106     //
107     //MAP_GPIOPinConfigure(GPIO_PB5_M0PWM3);
108     //MAP_GPIOPinTypePWM(GPIO_PORTB_BASE, GPIO_PIN_5);
109
110     //
111     // Enable pin PE4 for PWM0 M0PWM4
112     //
113     //MAP_GPIOPinConfigure(GPIO_PE4_M0PWM4);
114     //MAP_GPIOPinTypePWM(GPIO_PORTE_BASE, GPIO_PIN_4);
115
116     //
117     // Enable pin PB7 for PWM0 M0PWM1
118     //
119     MAP_GPIOPinConfigure(GPIO_PB7_M0PWM1);
120     MAP_GPIOPinTypePWM(GPIO_PORTB_BASE, GPIO_PIN_7);
121 }
```

Listing B.23: Mikrocontroller: pwm0_pin.h

```
1  //*****************************************************************************
2  // Copyright (c) 2014 Texas Instruments Incorporated.  All rights reserved.
3  // Software License Agreement
4  //
5  //   Redistribution and use in source and binary forms, with or without
6  //   modification, are permitted provided that the following conditions
7  //   are met:
8  //
9  //   Redistributions of source code must retain the above copyright
10 //   notice, this list of conditions and the following disclaimer.
11 //
```

```
12 //   Redistributions in binary form must reproduce the above copyright
13 //   notice, this list of conditions and the following disclaimer in the
14 //   documentation and/or other materials provided with the
15 //   distribution.
16 //
17 //   Neither the name of Texas Instruments Incorporated nor the names of
18 //   its contributors may be used to endorse or promote products derived
19 //   from this software without specific prior written permission.
20 //
21 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
22 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
23 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
24 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
25 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
26 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
27 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
28 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
29 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
30 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
31 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
32 //
33 // This file was automatically generated by the Tiva C Series PinMux Utility
34 // Version: 1.0.4
35 //
36 //*****************************************************************************
37
38 #ifndef __PWM0_PIN_H__
39 #define __PWM0_PIN_H__
40
41 extern void Pwm0PortFunctionInit(void);
42
43 #endif //  __PWM0_PIN_H__
```

Listing B.24: Mikrocontroller: pwm1_pin.c

```
1  //*****************************************************************************
2  // Copyright (c) 2014 Texas Instruments Incorporated.  All rights reserved.
3  // Software License Agreement
4  //
5  //   Redistribution and use in source and binary forms, with or without
6  //   modification, are permitted provided that the following conditions
7  //   are met:
8  //
9  //   Redistributions of source code must retain the above copyright
10 //   notice, this list of conditions and the following disclaimer.
11 //
12 //   Redistributions in binary form must reproduce the above copyright
13 //   notice, this list of conditions and the following disclaimer in the
14 //   documentation and/or other materials provided with the
```

```
15 //   distribution.
16 //
17 //   Neither the name of Texas Instruments Incorporated nor the names of
18 //   its contributors may be used to endorse or promote products derived
19 //   from this software without specific prior written permission.
20 //
21 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
22 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
23 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
24 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
25 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
26 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
27 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
28 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
29 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
30 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
31 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
32 //
33 // This file was automatically generated by the Tiva C Series PinMux Utility
34 // Version: 1.0.4
35 //
36 //*****************************************************************************
37
38 #include <stdint.h>
39 #include <stdbool.h>
40 #include "pwm1_pin.h"
41 #include "inc/hw_types.h"
42 #include "inc/hw_memmap.h"
43 #include "inc/hw_gpio.h"
44 #include "driverlib/sysctl.h"
45 #include "driverlib/pin_map.h"
46 #include "driverlib/rom_map.h"
47 #include "driverlib/gpio.h"
48
49 //*****************************************************************************
50 // using only M1PWM0
51 void Pwm1PortFunctionInit(void)
52 {
53     //
54     // Enable Peripheral Clocks
55     //
56     MAP_SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);
57 //    MAP_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOA);
58     MAP_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
59 //    MAP_SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
60
61     //
62     // Enable pin PF0 for PWM1 M1PWM4
```

```
63      // First open the lock and select the bits we want to modify in the GPIO
            commit register.
64      //
65  //    HWREG(GPIO_PORTF_BASE + GPIO_O_LOCK) = GPIO_LOCK_KEY;
66  //    HWREG(GPIO_PORTF_BASE + GPIO_O_CR) = 0x1;
67
68      //
69      // Now modify the configuration of the pins that we unlocked.
70      //
71  //    MAP_GPIOPinConfigure(GPIO_PF0_M1PWM4);
72  //    MAP_GPIOPinTypePWM(GPIO_PORTF_BASE, GPIO_PIN_0);
73
74      //
75      // Enable pin PA6 for PWM1 M1PWM2
76      //
77  //    MAP_GPIOPinConfigure(GPIO_PA6_M1PWM2);
78  //    MAP_GPIOPinTypePWM(GPIO_PORTA_BASE, GPIO_PIN_6);
79
80      //
81      // Enable pin PA7 for PWM1 M1PWM3
82      //
83  //    MAP_GPIOPinConfigure(GPIO_PA7_M1PWM3);
84  //    MAP_GPIOPinTypePWM(GPIO_PORTA_BASE, GPIO_PIN_7);
85
86      //
87      // Enable pin PD1 for PWM1 M1PWM1
88      //
89  //    MAP_GPIOPinConfigure(GPIO_PD1_M1PWM1);
90  //    MAP_GPIOPinTypePWM(GPIO_PORTD_BASE, GPIO_PIN_1);
91
92      //
93      // Enable pin PF3 for PWM1 M1PWM7
94      //
95  //    MAP_GPIOPinConfigure(GPIO_PF3_M1PWM7);
96  //    MAP_GPIOPinTypePWM(GPIO_PORTF_BASE, GPIO_PIN_3);
97
98      //
99      // Enable pin PD0 for PWM1 M1PWM0
100     //
101     MAP_GPIOPinConfigure(GPIO_PD0_M1PWM0);
102     MAP_GPIOPinTypePWM(GPIO_PORTD_BASE, GPIO_PIN_0);
103
104     //
105     // Enable pin PF2 for PWM1 M1PWM6
106     //
107 //    MAP_GPIOPinConfigure(GPIO_PF2_M1PWM6);
108 //    MAP_GPIOPinTypePWM(GPIO_PORTF_BASE, GPIO_PIN_2);
109
```

```
110         //
111         // Enable pin PF1 for PWM1 M1PWM5
112         //
113 //      MAP_GPIOPinConfigure(GPIO_PF1_M1PWM5);
114 //      MAP_GPIOPinTypePWM(GPIO_PORTF_BASE, GPIO_PIN_1);
115
116         //
117         // Enable pin PF4 for PWM1 M1FAULT0
118         //
119 //      MAP_GPIOPinConfigure(GPIO_PF4_M1FAULT0);
120 //      MAP_GPIOPinTypePWM(GPIO_PORTF_BASE, GPIO_PIN_4);
121 }
```

Listing B.25: Mikrocontroller: pwm1_pin.h

```
1  //*****************************************************************************
2  // Copyright (c) 2014 Texas Instruments Incorporated.  All rights reserved.
3  // Software License Agreement
4  //
5  //   Redistribution and use in source and binary forms, with or without
6  //   modification, are permitted provided that the following conditions
7  //   are met:
8  //
9  //   Redistributions of source code must retain the above copyright
10 //   notice, this list of conditions and the following disclaimer.
11 //
12 //   Redistributions in binary form must reproduce the above copyright
13 //   notice, this list of conditions and the following disclaimer in the
14 //   documentation and/or other materials provided with the
15 //   distribution.
16 //
17 //   Neither the name of Texas Instruments Incorporated nor the names of
18 //   its contributors may be used to endorse or promote products derived
19 //   from this software without specific prior written permission.
20 //
21 // THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
22 // "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
23 // LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
24 // A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
25 // OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
26 // SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
27 // LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
28 // DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
29 // THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
30 // (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
31 // OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
32 //
33 // This file was automatically generated by the Tiva C Series PinMux Utility
34 // Version: 1.0.4
```

```
35 //
36 //*****************************************************************************
37
38 #ifndef __PWM1_PIN_H__
39 #define __PWM1_PIN_H__
40
41 extern void Pwm1PortFunctionInit(void);
42
43 #endif //  __PWM1_PIN_H__
```

Listing B.26: Mikrocontroller: shtc1.c

```
1  /*
2   * shtc1.c
3   *
4   *  Created on: 02.08.2016
5   *      Author: goeth
6   */
7
8  #include "shtc1.h"
9
10 // Temperature in °C/2^16
11 inline int32_t Shtc1_Temperature(uint8_t tempMsb, uint8_t tempLsb)
12 {
13   return ( -45 << 16 ) + 175 * ((((int32_t) tempMsb) << 8) + (int32_t) tempLsb);
14 }
15
16 // relative humidity in %RH/2^16
17 inline int32_t Shtc1_Humidity(uint8_t humiMsb, uint8_t humiLsb)
18 {
19   return 100 * ((((int32_t) humiMsb) << 8) + humiLsb);
20 }
```

Listing B.27: Mikrocontroller: shtc1.h

```
1  /*
2   * shtc1.h
3   *
4   * Specific Sensor Header for Sensirion Humidity and Temperature Sensor IC SHTC1
5   *
6   *  Created on: 02.08.2016
7   *      Author: Torben Hensel
8   */
9
10 #ifndef SHTC1_H_
11 #define SHTC1_H_
12
13 #include <stdint.h>
14
```

```
15 // 7 bit I2C-bus-address
16 #define I2C_ADDR_HUMIDITY_SENSOR_SHTC1 0x70
17
18 // Measurement Start Commands ----------------------------------------------
19 // with Clock Stretching enabled
20 #define SHTC1_CLK_STR_EN_READ_TEMP_FIRST_MSB 0x7c
21 #define SHTC1_CLK_STR_EN_READ_TEMP_FIRST_LSB 0xA2
22
23 #define SHTC1_CLK_STR_EN_READ_HUMI_FIRST_MSB 0x5c
24 #define SHTC1_CLK_STR_EN_READ_HUMI_FIRST_LSB 0x24
25
26 // with Clock Stretching disabled
27 #define SHTC1_CLK_STR_DIS_READ_TEMP_FIRST_MSB 0x78
28 #define SHTC1_CLK_STR_DIS_READ_TEMP_FIRST_LSB 0x66
29
30 #define SHTC1_CLK_STR_DIS_READ_HUMI_FIRST_MSB 0x58
31 #define SHTC1_CLK_STR_DIS_READ_HUMI_FIRST_LSB 0xe0
32
33 //-------------------------------------------------------------------------
34
35 // Soft Reset
36 #define SHTC1_SOFT_RESET_MSB 0x80
37 #define SHTC1_SOFT_RESET_LSB 0x5d
38
39 // ID-Register
40 #define SHTC1_ID_REG_MSB 0xef
41 #define SHTC1_ID_REG_LSB 0xc8
42
43 // CRC-8 Polynomial
44 #define SHTC1_CRC_POLYNOM 0x31 // x^8 + x^5 + x^4 + 1
45
46 // Sensor Output Conversion ------------------------------------------------
47
48 inline int32_t Shtc1_Temperature(uint8_t tempMsb, uint8_t tempLsb);
49 inline int32_t Shtc1_Humidity(uint8_t humiMsb, uint8_t humiLsb);
50
51 //-------------------------------------------------------------------------
52
53 #endif /* SHTC1_H_ */
```

Listing B.28: Mikrocontroller: systick.c

```
1 /*
2  * systick.c
3  *
4  *  Created on: 22.04.2015
5  *      Author: Torben
6  */
7
```

```
8  #include "systick.h"
9
10 // configers SysTick
11 void SysTickConfig(void)
12 {
13     //
14     // Set up and enable the SysTick timer.  It will be used as a reference
15     // for delay loops in the interrupt handlers.  The SysTick timer period
16     // will be set up for 1/4 second.
17     //
18     ROM_SysTickPeriodSet(ROM_SysCtlClockGet()/4);
19     ROM_SysTickEnable();
20
21     // Systick Interrupt Handler Register
22     //
23     ui32SysTickIntCount = 0;
24     SysTickIntRegister(SysTickIntHandler);
25
26     ui32SysTickStat = 0x04;
27     ui32SysTickSamplePeriode = 0; // to save old Adc Sample Periode
28
29 }
30 // SysTick Interrupt Handler
31 void SysTickIntHandler(void)
32 {
33   ui32SysTickIntCount = ui32SysTickIntCount + 1; // counts SysTickInterrupts
34   // new data
35   if (ui32AdcNewSample)
36   {
37     // get index for new samples and block it
38     ui32AdcSampleReading = ui32AdcSampleReady;
39
40     // from Laser 1
41     // ADC AIN0 E3 - 30° LSB - MSB
42     ui8I2CRegister[0x13] = ui32AdcSamples[0][ui32AdcSampleReading][0] & 0
           x000000FF;
43     ui8I2CRegister[0x12] = (ui32AdcSamples[0][ui32AdcSampleReading][0] >> 8) & 0
           x000000FF;
44     ui8I2CRegister[0x11] = (ui32AdcSamples[0][ui32AdcSampleReading][0] >> 16) & 0
           x000000FF;
45     ui8I2CRegister[0x10] = (ui32AdcSamples[0][ui32AdcSampleReading][0] >> 24) & 0
           x000000FF;
46     //  ADC AIN1 E2 - 60° LSB - MSB
47     ui8I2CRegister[0x17] = ui32AdcSamples[0][ui32AdcSampleReading][1] & 0
           x000000FF;
48     ui8I2CRegister[0x16] = (ui32AdcSamples[0][ui32AdcSampleReading][1] >> 8) & 0
           x000000FF;
```

```
49      ui8I2CRegister[0x15] = (ui32AdcSamples[0][ui32AdcSampleReading][1] >> 16) & 0
            x000000FF;
50      ui8I2CRegister[0x14] = (ui32AdcSamples[0][ui32AdcSampleReading][1] >> 24) & 0
            x000000FF;
51      //  ADC AIN2 E1 - 90° LSB - MSB
52      ui8I2CRegister[0x1b] = ui32AdcSamples[0][ui32AdcSampleReading][2] & 0
            x000000FF;
53      ui8I2CRegister[0x1a] = (ui32AdcSamples[0][ui32AdcSampleReading][2] >> 8) & 0
            x000000FF;
54      ui8I2CRegister[0x19] = (ui32AdcSamples[0][ui32AdcSampleReading][2] >> 16) & 0
            x000000FF;
55      ui8I2CRegister[0x18] = (ui32AdcSamples[0][ui32AdcSampleReading][2] >> 24) & 0
            x000000FF;
56      //  ADC AIN8 E5 - 120° LSB - MSB
57      ui8I2CRegister[0x1f] = ui32AdcSamples[0][ui32AdcSampleReading][3] & 0
            x000000FF;
58      ui8I2CRegister[0x1e] = (ui32AdcSamples[0][ui32AdcSampleReading][3] >> 8) & 0
            x000000FF;
59      ui8I2CRegister[0x1d] = (ui32AdcSamples[0][ui32AdcSampleReading][3] >> 16) & 0
            x000000FF;
60      ui8I2CRegister[0x1c] = (ui32AdcSamples[0][ui32AdcSampleReading][3] >> 24) & 0
            x000000FF;
61      //  ADC AIN9 E4 - 150° LSB - MSB
62      ui8I2CRegister[0x23] = ui32AdcSamples[0][ui32AdcSampleReading][4] & 0
            x000000FF;
63      ui8I2CRegister[0x22] = (ui32AdcSamples[0][ui32AdcSampleReading][4] >> 8) & 0
            x000000FF;
64      ui8I2CRegister[0x21] = (ui32AdcSamples[0][ui32AdcSampleReading][4] >> 16) & 0
            x000000FF;
65      ui8I2CRegister[0x20] = (ui32AdcSamples[0][ui32AdcSampleReading][4] >> 24) & 0
            x000000FF;
66      //  ADC AIN10 B4 - reserve LSB - MSB
67      ui8I2CRegister[0x36] = ui32AdcSamples[0][ui32AdcSampleReading][5] & 0
            x000000FF;
68      ui8I2CRegister[0x35] = (ui32AdcSamples[0][ui32AdcSampleReading][5] >> 8) & 0
            x000000FF;
69      ui8I2CRegister[0x34] = (ui32AdcSamples[0][ui32AdcSampleReading][5] >> 16) & 0
            x000000FF;
70      ui8I2CRegister[0x33] = (ui32AdcSamples[0][ui32AdcSampleReading][5] >> 24) & 0
            x000000FF;
71      //  ADC AIN11 B5 - reserve LSB - MSB
72      ui8I2CRegister[0x3a] = ui32AdcSamples[0][ui32AdcSampleReading][6] & 0
            x000000FF;
73      ui8I2CRegister[0x39] = (ui32AdcSamples[0][ui32AdcSampleReading][6] >> 8) & 0
            x000000FF;
74      ui8I2CRegister[0x38] = (ui32AdcSamples[0][ui32AdcSampleReading][6] >> 16) & 0
            x000000FF;
```

```
75      ui8I2CRegister[0x37] = (ui32AdcSamples[0][ui32AdcSampleReading][6] >> 24) & 0
            x000000FF;
76      //  ADC internal Temperature LSB - MSB
77      ui8I2CRegister[0x32] = ui32AdcSamples[0][ui32AdcSampleReading][7] & 0
            x000000FF;
78      ui8I2CRegister[0x31] = (ui32AdcSamples[0][ui32AdcSampleReading][7] >> 8) & 0
            x000000FF;
79      ui8I2CRegister[0x30] = (ui32AdcSamples[0][ui32AdcSampleReading][7] >> 16) & 0
            x000000FF;
80      ui8I2CRegister[0x2f] = (ui32AdcSamples[0][ui32AdcSampleReading][7] >> 24) & 0
            x000000FF;
81
82      // from Laser 2
83      // ADC AIN0 E3 - 30° LSB - MSB
84      ui8I2CRegister[0x3e] = ui32AdcSamples[1][ui32AdcSampleReading][0] & 0
            x000000FF;
85      ui8I2CRegister[0x3d] = (ui32AdcSamples[1][ui32AdcSampleReading][0] >> 8) & 0
            x000000FF;
86      ui8I2CRegister[0x3c] = (ui32AdcSamples[1][ui32AdcSampleReading][0] >> 16) & 0
            x000000FF;
87      ui8I2CRegister[0x3b] = (ui32AdcSamples[1][ui32AdcSampleReading][0] >> 24) & 0
            x000000FF;
88      //  ADC AIN1 E2 - 60° LSB - MSB
89      ui8I2CRegister[0x42] = ui32AdcSamples[1][ui32AdcSampleReading][1] & 0
            x000000FF;
90      ui8I2CRegister[0x41] = (ui32AdcSamples[1][ui32AdcSampleReading][1] >> 8) & 0
            x000000FF;
91      ui8I2CRegister[0x40] = (ui32AdcSamples[1][ui32AdcSampleReading][1] >> 16) & 0
            x000000FF;
92      ui8I2CRegister[0x3f] = (ui32AdcSamples[1][ui32AdcSampleReading][1] >> 24) & 0
            x000000FF;
93      //  ADC AIN2 E1 - 90° LSB - MSB
94      ui8I2CRegister[0x46] = ui32AdcSamples[1][ui32AdcSampleReading][2] & 0
            x000000FF;
95      ui8I2CRegister[0x45] = (ui32AdcSamples[1][ui32AdcSampleReading][2] >> 8) & 0
            x000000FF;
96      ui8I2CRegister[0x44] = (ui32AdcSamples[1][ui32AdcSampleReading][2] >> 16) & 0
            x000000FF;
97      ui8I2CRegister[0x43] = (ui32AdcSamples[1][ui32AdcSampleReading][2] >> 24) & 0
            x000000FF;
98      //  ADC AIN8 E5 - 120° LSB - MSB
99      ui8I2CRegister[0x4a] = ui32AdcSamples[1][ui32AdcSampleReading][3] & 0
            x000000FF;
100     ui8I2CRegister[0x49] = (ui32AdcSamples[1][ui32AdcSampleReading][3] >> 8) & 0
            x000000FF;
101     ui8I2CRegister[0x48] = (ui32AdcSamples[1][ui32AdcSampleReading][3] >> 16) & 0
            x000000FF;
```

```
102     ui8I2CRegister[0x47] = (ui32AdcSamples[1][ui32AdcSampleReading][3] >> 24) & 0
            x000000FF;
103     //  ADC AIN9 E4 - 150° LSB - MSB
104     ui8I2CRegister[0x4e] = ui32AdcSamples[1][ui32AdcSampleReading][4] & 0
            x000000FF;
105     ui8I2CRegister[0x4d] = (ui32AdcSamples[1][ui32AdcSampleReading][4] >> 8) & 0
            x000000FF;
106     ui8I2CRegister[0x4c] = (ui32AdcSamples[1][ui32AdcSampleReading][4] >> 16) & 0
            x000000FF;
107     ui8I2CRegister[0x4b] = (ui32AdcSamples[1][ui32AdcSampleReading][4] >> 24) & 0
            x000000FF;
108     //  ADC AIN10 B4 - reserve LSB - MSB
109     ui8I2CRegister[0x56] = ui32AdcSamples[1][ui32AdcSampleReading][5] & 0
            x000000FF;
110     ui8I2CRegister[0x55] = (ui32AdcSamples[1][ui32AdcSampleReading][5] >> 8) & 0
            x000000FF;
111     ui8I2CRegister[0x54] = (ui32AdcSamples[1][ui32AdcSampleReading][5] >> 16) & 0
            x000000FF;
112     ui8I2CRegister[0x53] = (ui32AdcSamples[1][ui32AdcSampleReading][5] >> 24) & 0
            x000000FF;
113     //  ADC AIN11 B5 - reserve LSB - MSB
114     ui8I2CRegister[0x5a] = ui32AdcSamples[1][ui32AdcSampleReading][6] & 0
            x000000FF;
115     ui8I2CRegister[0x59] = (ui32AdcSamples[1][ui32AdcSampleReading][6] >> 8) & 0
            x000000FF;
116     ui8I2CRegister[0x58] = (ui32AdcSamples[1][ui32AdcSampleReading][6] >> 16) & 0
            x000000FF;
117     ui8I2CRegister[0x57] = (ui32AdcSamples[1][ui32AdcSampleReading][6] >> 24) & 0
            x000000FF;
118     //  ADC internal Temperature LSB - MSB
119     ui8I2CRegister[0x52] = ui32AdcSamples[1][ui32AdcSampleReading][7] & 0
            x000000FF;
120     ui8I2CRegister[0x51] = (ui32AdcSamples[1][ui32AdcSampleReading][7] >> 8) & 0
            x000000FF;
121     ui8I2CRegister[0x50] = (ui32AdcSamples[1][ui32AdcSampleReading][7] >> 16) & 0
            x000000FF;
122     ui8I2CRegister[0x4f] = (ui32AdcSamples[1][ui32AdcSampleReading][7] >> 24) & 0
            x000000FF;
123
124     // set New Sample I2c Flag
125     ui8I2CRegister[0x01] |= 0x02;
126     //  in single Shot Mode, reset Running Flag and stop PWM
127     if ( ( ui8I2CRegister[0x01] & 0x04 ) == 0x0)
128       {
129       ui8I2CRegister[0x01] &= ~0x01;
130       Pwm0Stop(); // stop PWM
131       }
132
```

```
133       // set new Sample Flag in I2C Register
134       ui8I2CRegister[0x01] |= 0x02;
135
136       // set new Sample Interrupt Pin F2
137       GPIO_PORTF_DATA_R |= 0x04;
138
139       // reset New Sample Flag
140       ui32AdcNewSample = 0;
141     } // end new ADC-Sample
142
143
144
145     if (ui8I2CRegister[0x05] != ( ui8I2CRegister[0x01] >> 2 ) ) // something new in
          Mode Register
146     {
147       switch ( ui8I2CRegister[0x05] )
148       {
149       case 0x00:
150         ui8I2CRegister[0x01] &= ~0x04;
151         break;
152       case 0x01:
153         ui8I2CRegister[0x01] |= 0x04;
154       }
155
156     } // end Mode Register
157
158     if ( ui8I2CRegister[0x03] != ui32SysTickSamplePeriode ) // to save old Adc
          Sample Periode
159     {
160       if ( ui8I2CRegister[0x03] < 0xD2 )
161       {
162         ui32SysTickSamplePeriode = ui8I2CRegister[0x03]; // Sample Periode in
              Seconds - 1s
163         ui32AdcSampleLengh = ( 5000 + ( ( (uint32_t) ui8I2CRegister[0x03] ) * 5000
              ) );  // Sample Periode in Samples
164       }
165       else // if value too big, cut it to maximum
166       {
167         ui32SysTickSamplePeriode = 0xD1; // Sample Periode 209 Seconds
168         ui32AdcSampleLengh = 209 * 5000; // Sample Periode in Samples
169       }
170     } // end Sample Period
171
172     if (ui8I2CRegister[0x04]) // something in Start/Stop Register
173     {
174       // Start ADC, if it is not Running
175       if ( (ui8I2CRegister[0x04] == 0x01) &&  ( ( ui8I2CRegister[0x01] & 0x01 ) ==
            0 ) )
```

```
176     {
177       // start Fan, PortA Pin 2
178       GPIO_PORTA_DATA_R |= 0x04;
179 
180       // reset ADC
181       ui32AdcSamples[0][ui32AdcSampleActive][0] = 0;
182       ui32AdcSamples[0][ui32AdcSampleActive][1] = 0;
183       ui32AdcSamples[0][ui32AdcSampleActive][2] = 0;
184       ui32AdcSamples[0][ui32AdcSampleActive][3] = 0;
185       ui32AdcSamples[0][ui32AdcSampleActive][4] = 0;
186       ui32AdcSamples[0][ui32AdcSampleActive][5] = 0;
187       ui32AdcSamples[0][ui32AdcSampleActive][6] = 0;
188       ui32AdcSamples[0][ui32AdcSampleActive][7] = 0;
189 
190       ui32AdcSamples[1][ui32AdcSampleActive][0] = 0;
191       ui32AdcSamples[1][ui32AdcSampleActive][1] = 0;
192       ui32AdcSamples[1][ui32AdcSampleActive][2] = 0;
193       ui32AdcSamples[1][ui32AdcSampleActive][3] = 0;
194       ui32AdcSamples[1][ui32AdcSampleActive][4] = 0;
195       ui32AdcSamples[1][ui32AdcSampleActive][5] = 0;
196       ui32AdcSamples[1][ui32AdcSampleActive][6] = 0;
197       ui32AdcSamples[1][ui32AdcSampleActive][7] = 0;
198 
199       ui32AdcIntCount = 0;
200       ui32AdcNewSample = 0;
201       //ui32AdcLaser = 0; not used anymore
202 
203       Pwm0Start(); // Start PWM
204 
205       ui8I2CRegister[0x01] &= ~0x02; // reset new Sample Flag
206       GPIO_PORTF_DATA_R &= ~0x04; // reset new Sample Pin F2
207       ui8I2CRegister[0x01] |= 0x01; // set running Flag
208     }
209     // Stop ADC, if running
210     if ( (ui8I2CRegister[0x04] == 0x02) && ( ui8I2CRegister[0x01] & 0x01 ) )
211     {
212       Pwm0Stop(); // Stop PWM
213 
214       GPIO_PORTA_DATA_R &= ~0x04; // stop fan, PortA Pin 2
215 
216       ui8I2CRegister[0x01] &= ~0x01; // reset running Flag
217     }
218 
219     ui8I2CRegister[0x04] = 0; // reset register
220   } // end Start/Stop register
221 
222 }
```

Listing B.29: Mikrocontroller: systick.h

```
/*
 * systick.h
 *
 *  Created on: 22.04.2015
 *      Author: Torben
 */

#ifndef SYSTICK_H_
#define SYSTICK_H_

#include <stdint.h>
#include <stdbool.h>
//#include <math.h>
//#include <time.h>
#include "inc/tm4c123gh6pm.h"
#include "inc/hw_types.h"
#include "inc/hw_memmap.h"
//#include "inc/hw_hibernate.h"
//#include "driverlib/fpu.h"
#include "driverlib/gpio.h"
//#include "driverlib/hibernate.h"
#include "driverlib/interrupt.h"
#include "driverlib/pin_map.h"
#include "driverlib/rom.h"
#include "driverlib/sysctl.h"
#include "driverlib/systick.h"
//#include "driverlib/uart.h"
//#include "utils/uartstdio.h"
//#include "utils/cmdline.h"
//#include "drivers/buttons.h"

#include "adc.h"
#include "i2c_slave.h"

extern volatile uint32_t ui32SysTickIntCount;

void SysTickIntHandler(void);
void SysTickConfig(void);

volatile uint32_t ui32SysTickStat;
volatile uint32_t ui32SysTickSamplePeriode; //Sample Periode in Seconds - 1s, to
    save old Adc Sample Periode

// to count SysTickInterrupts
volatile uint32_t ui32SysTickIntCount;


```

```
47 #endif /* SYSTICK_H_ */
```

## B.2. Einplatinencomputer

Listing B.30: Raspberry Pi: psense.py

```
1  import smbus
2  import RPi.GPIO as GPIO
3  import time
4
5  GPIO.setmode(GPIO.BCM)
6  GPIO.setup(23, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
7  GPIO.setup(24, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
8
9  bus = smbus.SMBus(1)
10
11 ADDRESS_PSENSE = 0x42
12 REG_ID = 0x02
13 REG_STATUS = 0x01
14 REG_MODE = 0x05
15 REG_SAMPLE_PERIODE = 0x03
16 REG_START_STOP = 0x04
17
18 REG_ADC1_30_MSB = 0x10
19 REG_ADC1_60_MSB = 0x14
20 REG_ADC1_90_MSB = 0x18
21 REG_ADC1_120_MSB = 0x1c
22 REG_ADC1_150_MSB = 0x20
23
24 REG_ADC1_TEMP_MSB = 0x2f
25 REG_ADC1_RESERVE1_MSB = 0x33
26 REG_ADC1_RESERVE2_MSB = 0x37
27
28 REG_ADC2_30_MSB = 0x3b
29 REG_ADC2_60_MSB = 0x3f
30 REG_ADC2_90_MSB = 0x43
31 REG_ADC2_120_MSB = 0x47
32 REG_ADC2_150_MSB = 0x4b
33
34 REG_ADC2_TEMP_MSB = 0x4f
35 REG_ADC2_RESERVE1_MSB = 0x53
36 REG_ADC2_RESERVE2_MSB = 0x57
37
38 file = open('./data/'+ time.strftime("%Y_%m_%d_%H_%M_%S", time.gmtime()) + '
       _psense.csv', 'a')
39
40 def pSenseNewSample(channel):
```

```
41      # all from light 1
42      #read 30 Degree from Light1
43      ADC1_30 = bus.read_byte_data(ADDRESS_PSENSE, REG_ADC1_30_MSB)
44      ADC1_30 = (ADC1_30 << 8) + bus.read_byte(ADDRESS_PSENSE)
45      ADC1_30 = (ADC1_30 << 8) + bus.read_byte(ADDRESS_PSENSE)
46      ADC1_30 = (ADC1_30 << 8) + bus.read_byte(ADDRESS_PSENSE)
47      #read 60 Degree from Light1
48      ADC1_60 = bus.read_byte(ADDRESS_PSENSE)
49      ADC1_60 = (ADC1_60 << 8) + bus.read_byte(ADDRESS_PSENSE)
50      ADC1_60 = (ADC1_60 << 8) + bus.read_byte(ADDRESS_PSENSE)
51      ADC1_60 = (ADC1_60 << 8) + bus.read_byte(ADDRESS_PSENSE)
52      #read 90 Degree from Light1
53      ADC1_90 = bus.read_byte(ADDRESS_PSENSE)
54      ADC1_90 = (ADC1_90 << 8) + bus.read_byte(ADDRESS_PSENSE)
55      ADC1_90 = (ADC1_90 << 8) + bus.read_byte(ADDRESS_PSENSE)
56      ADC1_90 = (ADC1_90 << 8) + bus.read_byte(ADDRESS_PSENSE)
57      #read 120 Degree from Light1
58      ADC1_120 = bus.read_byte(ADDRESS_PSENSE)
59      ADC1_120 = (ADC1_120 << 8) + bus.read_byte(ADDRESS_PSENSE)
60      ADC1_120 = (ADC1_120 << 8) + bus.read_byte(ADDRESS_PSENSE)
61      ADC1_120 = (ADC1_120 << 8) + bus.read_byte(ADDRESS_PSENSE)
62      #read 150 Degree from Light1
63      ADC1_150 = bus.read_byte(ADDRESS_PSENSE)
64      ADC1_150 = (ADC1_150 << 8) + bus.read_byte(ADDRESS_PSENSE)
65      ADC1_150 = (ADC1_150 << 8) + bus.read_byte(ADDRESS_PSENSE)
66      ADC1_150 = (ADC1_150 << 8) + bus.read_byte(ADDRESS_PSENSE)
67      # all from light 2
68      #read 30 Degree from Light2
69      ADC2_30 = bus.read_byte_data(ADDRESS_PSENSE, REG_ADC2_30_MSB)
70      ADC2_30 = (ADC2_30 << 8) + bus.read_byte(ADDRESS_PSENSE)
71      ADC2_30 = (ADC2_30 << 8) + bus.read_byte(ADDRESS_PSENSE)
72      ADC2_30 = (ADC2_30 << 8) + bus.read_byte(ADDRESS_PSENSE)
73      #read 60 Degree from Light2
74      ADC2_60 = bus.read_byte(ADDRESS_PSENSE)
75      ADC2_60 = (ADC2_60 << 8) + bus.read_byte(ADDRESS_PSENSE)
76      ADC2_60 = (ADC2_60 << 8) + bus.read_byte(ADDRESS_PSENSE)
77      ADC2_60 = (ADC2_60 << 8) + bus.read_byte(ADDRESS_PSENSE)
78      #read 90 Degree from Light2
79      ADC2_90 = bus.read_byte(ADDRESS_PSENSE)
80      ADC2_90 = (ADC2_90 << 8) + bus.read_byte(ADDRESS_PSENSE)
81      ADC2_90 = (ADC2_90 << 8) + bus.read_byte(ADDRESS_PSENSE)
82      ADC2_90 = (ADC2_90 << 8) + bus.read_byte(ADDRESS_PSENSE)
83      #read 120 Degree from Light2
84      ADC2_120 = bus.read_byte(ADDRESS_PSENSE)
85      ADC2_120 = (ADC2_120 << 8) + bus.read_byte(ADDRESS_PSENSE)
86      ADC2_120 = (ADC2_120 << 8) + bus.read_byte(ADDRESS_PSENSE)
87      ADC2_120 = (ADC2_120 << 8) + bus.read_byte(ADDRESS_PSENSE)
88      #read 150 Degree from Light2
```

```
89      ADC2_150 = bus.read_byte(ADDRESS_PSENSE)
90      ADC2_150 = (ADC2_150 << 8) + bus.read_byte(ADDRESS_PSENSE)
91      ADC2_150 = (ADC2_150 << 8) + bus.read_byte(ADDRESS_PSENSE)
92      ADC2_150 = (ADC2_150 << 8) + bus.read_byte(ADDRESS_PSENSE)
93      # MPU temperature
94      MPUtemp = bus.read_byte_data(ADDRESS_PSENSE, REG_ADC1_TEMP_MSB)
95      MPUtemp = (MPUtemp << 8) + bus.read_byte(ADDRESS_PSENSE)
96      MPUtemp = (MPUtemp << 8) + bus.read_byte(ADDRESS_PSENSE)
97      MPUtemp = (MPUtemp << 8) + bus.read_byte(ADDRESS_PSENSE)
98      # write data to file
99      file.write( str(ADC1_30) + ';' + str(ADC1_60) + ';' + str(ADC1_90) + ';' + str(
            ADC1_120) +
100       ';' + str(ADC1_150) + ';' + str(ADC2_30) + ';' + str(ADC2_60) + ';' + str(
              ADC2_90) +
101       ';' + str(ADC2_120) + ';' + str(ADC2_150) + ';' + str(MPUtemp) + ';\n' )
102     print(str(ADC1_30*3000/(5000*4095)) + ' ; ' + str(ADC1_60*3000/(5000*4095)) + '
            ; ' +
103       str(ADC1_90*3000/(5000*4095)) + ' ; ' + str(ADC1_120*3000/(5000*4095)) + ' ;
              ' +
104       str(ADC1_150*3000/(5000*4095)) + ' ;\n' + str(ADC2_30*3000/(5000*4095)) + ' ;
              ' +
105       str(ADC2_60*3000/(5000*4095)) + ' ; ' + str(ADC2_90*3000/(5000*4095)) + ' ; '
              +
106       str(ADC2_120*3000/(5000*4095)) + ' ; ' + str(ADC2_150*3000/(5000*4095)) + ' ;
              ' +
107       str(148-(75*3*MPUtemp/(5000*4095))) + ' ;\n' )
108
109
110   # Config Sensor
111   def pSenseConfig():
112     #stop sensor, just for safty
113     bus.write_byte_data(ADDRESS_PSENSE, REG_START_STOP, 2)
114     #wait a while for sensor to react to stop signal
115     time.sleep(.3)
116     #set sample periode to 1 second
117     bus.write_byte_data(ADDRESS_PSENSE, REG_SAMPLE_PERIODE, 0)
118     #set mode to continus sampling
119     bus.write_byte_data(ADDRESS_PSENSE, REG_MODE, 1)
120
121   # Start Sampling
122   def pSenseStart():
123     #open file and write header
124     file.write('pSense Messung;\n' + time.strftime("%a, %d %b %Y %H:%M:%S +0000",
            time.gmtime()) + ';\n' +
125       'ADC1_30;ADC1_60;ADC1_90;ADC1_120;ADC1_150;ADC2_30;ADC2_60;ADC2_90;ADC2_120;
              ADC2_150;MPUtemp;\n')
126     #start sampling on Sensor
127     bus.write_byte_data(ADDRESS_PSENSE, REG_START_STOP, 1)
```

```
128     #register interrupt on GPIO23 Rising
129     GPIO.add_event_detect(23, GPIO.RISING, callback=pSenseNewSample, bouncetime
            =200)
130
131 # Stop Sampling
132 def pSenseStop():
133     #stop sampling on Sensor
134     bus.write_byte_data(ADDRESS_PSENSE, REG_START_STOP, 2)
135     #stop interrupt on GPIO23
136     GPIO.remove_event_detect(23)
137
138 def pSenseManual():
139     # all from light 1
140     #read 30 Degree from Light1
141     ADC1_30 = bus.read_byte_data(ADDRESS_PSENSE, REG_ADC1_30_MSB)
142     ADC1_30 = (ADC1_30 << 8) + bus.read_byte(ADDRESS_PSENSE)
143     ADC1_30 = (ADC1_30 << 8) + bus.read_byte(ADDRESS_PSENSE)
144     ADC1_30 = (ADC1_30 << 8) + bus.read_byte(ADDRESS_PSENSE)
145     #read 60 Degree from Light1
146     ADC1_60 = bus.read_byte(ADDRESS_PSENSE)
147     ADC1_60 = (ADC1_60 << 8) + bus.read_byte(ADDRESS_PSENSE)
148     ADC1_60 = (ADC1_60 << 8) + bus.read_byte(ADDRESS_PSENSE)
149     ADC1_60 = (ADC1_60 << 8) + bus.read_byte(ADDRESS_PSENSE)
150     #read 90 Degree from Light1
151     ADC1_90 = bus.read_byte(ADDRESS_PSENSE)
152     ADC1_90 = (ADC1_90 << 8) + bus.read_byte(ADDRESS_PSENSE)
153     ADC1_90 = (ADC1_90 << 8) + bus.read_byte(ADDRESS_PSENSE)
154     ADC1_90 = (ADC1_90 << 8) + bus.read_byte(ADDRESS_PSENSE)
155     #read 120 Degree from Light1
156     ADC1_120 = bus.read_byte(ADDRESS_PSENSE)
157     ADC1_120 = (ADC1_120 << 8) + bus.read_byte(ADDRESS_PSENSE)
158     ADC1_120 = (ADC1_120 << 8) + bus.read_byte(ADDRESS_PSENSE)
159     ADC1_120 = (ADC1_120 << 8) + bus.read_byte(ADDRESS_PSENSE)
160     #read 150 Degree from Light1
161     ADC1_150 = bus.read_byte(ADDRESS_PSENSE)
162     ADC1_150 = (ADC1_150 << 8) + bus.read_byte(ADDRESS_PSENSE)
163     ADC1_150 = (ADC1_150 << 8) + bus.read_byte(ADDRESS_PSENSE)
164     ADC1_150 = (ADC1_150 << 8) + bus.read_byte(ADDRESS_PSENSE)
165     # all from light 2
166     #read 30 Degree from Light2
167     ADC2_30 = bus.read_byte_data(ADDRESS_PSENSE, REG_ADC2_30_MSB)
168     ADC2_30 = (ADC2_30 << 8) + bus.read_byte(ADDRESS_PSENSE)
169     ADC2_30 = (ADC2_30 << 8) + bus.read_byte(ADDRESS_PSENSE)
170     ADC2_30 = (ADC2_30 << 8) + bus.read_byte(ADDRESS_PSENSE)
171     #read 60 Degree from Light2
172     ADC2_60 = bus.read_byte(ADDRESS_PSENSE)
173     ADC2_60 = (ADC2_60 << 8) + bus.read_byte(ADDRESS_PSENSE)
174     ADC2_60 = (ADC2_60 << 8) + bus.read_byte(ADDRESS_PSENSE)
```

```
175     ADC2_60 = (ADC2_60 << 8) + bus.read_byte(ADDRESS_PSENSE)
176     #read 90 Degree from Light2
177     ADC2_90 = bus.read_byte(ADDRESS_PSENSE)
178     ADC2_90 = (ADC2_90 << 8) + bus.read_byte(ADDRESS_PSENSE)
179     ADC2_90 = (ADC2_90 << 8) + bus.read_byte(ADDRESS_PSENSE)
180     ADC2_90 = (ADC2_90 << 8) + bus.read_byte(ADDRESS_PSENSE)
181     #read 120 Degree from Light2
182     ADC2_120 = bus.read_byte(ADDRESS_PSENSE)
183     ADC2_120 = (ADC2_120 << 8) + bus.read_byte(ADDRESS_PSENSE)
184     ADC2_120 = (ADC2_120 << 8) + bus.read_byte(ADDRESS_PSENSE)
185     ADC2_120 = (ADC2_120 << 8) + bus.read_byte(ADDRESS_PSENSE)
186     #read 150 Degree from Light2
187     ADC2_150 = bus.read_byte(ADDRESS_PSENSE)
188     ADC2_150 = (ADC2_150 << 8) + bus.read_byte(ADDRESS_PSENSE)
189     ADC2_150 = (ADC2_150 << 8) + bus.read_byte(ADDRESS_PSENSE)
190     ADC2_150 = (ADC2_150 << 8) + bus.read_byte(ADDRESS_PSENSE)
191     # MPU temperature
192     MPUtemp = bus.read_byte_data(ADDRESS_PSENSE, REG_ADC1_TEMP_MSB)
193     MPUtemp = (MPUtemp << 8) + bus.read_byte(ADDRESS_PSENSE)
194     MPUtemp = (MPUtemp << 8) + bus.read_byte(ADDRESS_PSENSE)
195     MPUtemp = (MPUtemp << 8) + bus.read_byte(ADDRESS_PSENSE)
196     # write data to file
197     file.write( str(ADC1_30) + ';' + str(ADC1_60) + ';' + str(ADC1_90) + ';' + str(
            ADC1_120) +
198       ';' + str(ADC1_150) + ';' + str(ADC2_30) + ';' + str(ADC2_60) + ';' + str(
              ADC2_90) +
199       ';' + str(ADC2_120) + ';' + str(ADC2_150) + ';' + str(MPUtemp) + ';\n' )
200
201
202
203
204 pSenseStart()
205
206 time.sleep(60)
207
208 pSenseStop()
209 GPIO.cleanup()
210 file.close()
```

# Versicherung über die Selbstständigkeit

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

Hamburg, 14. September 2016
Ort, Datum

Unterschrift