

Technische Universität Berlin  
Fakultät IV  
Institut für Energie- und Automatisierungstechnik  
Fachgebiet Lichttechnik



# **Entwicklung und Realisierung einer Messeinrichtung mit den Sensoren AS7261 und AS72651 von ams**

**Bachelorarbeit**

Vorgelegt von: Lennard Bödiger

Matrikelnr.: 363470

Studiengang: Technische Informatik

26. Dezember 2020

# 1 Abstract

The goal of this work is to develop a spectral sensor platform using the AS7261 and AS72651, which can be used modularly for different research purposes, but especially for daylight measurement. Since it is not possible to measure in several directions in the sky at the same time with only one of the AS7261 (or AS72651) sensors without mechanical components, it must be ensured that a large number of sensors can be combined into a sensor array in order to be able to measure directional resolution. The resulting sensor platform uses a Raspberry Pi 4 as central control unit, which communicates with external sensor boards via I2C. The sensor boards are designed to be connected in series to allow a variety of measurement configurations. The database and the web interface for displaying and querying the data runs locally on the Rasperrypi. The thesis describes the developed of all hardware and software components.

Ziel dieser Arbeit ist es, mithilfe des AS7261 und AS72651 eine Spektral-Sensor Plattform zu entwickeln, die modular für unterschiedliche Forschungszwecke, aber vor allem zu Tageslichtmessung, genutzt werden kann. Entscheidend dabei ist, dass das System portabel und kosteneffizient umgesetzt wird, um vielfache und flexible messaufbauten zu ermöglichen. Da es ohne mechanische Komponenten nicht möglich ist, mit nur einem der AS7261 (oder auch AS72651) Sensoren in mehrere Richtungen am Himmel gleichzeitig zu messen, muss es gewährleistet sein, eine Vielzahl der Sensoren zu einem Sensor Array zusammenzuschließen, um so Richtungsauf lösend messen zu können.

Die resultierende Sensorplattform nutzt einen Raspberry Pi 4 als zentrale Steuereinheit, welcher über I2C mit externen Sensnorboards kommuniziert. Die Sensnoroards sind darauf ausgelegt, in reihe geschaltet zu werden, um so eine Vielzahl von Messkonfigurationen zu ermöglichen.

Die Datenbank und das Webinterface zum darstellen und abfragen der daten laufen lokal auf dem Rasperrypi.

In der Arbeit sind die entwickelten hard und Softwarekomponente beschrieben.

## Inhaltsverzeichnis

1 Abstract	2
------------	---

<b>2</b>	<b>TODO Liste</b>	<b>5</b>
<b>3</b>	<b>Einleitung</b>	<b>6</b>
3.1	Aufbau des Messsystems . . . . .	6
<b>4</b>	<b>Technische Grundlagen</b>	<b>7</b>
4.1	Fotodioden . . . . .	7
4.2	I2C . . . . .	7
<b>5</b>	<b>Hardware Komponenten</b>	<b>8</b>
5.1	Sensoren . . . . .	9
5.1.1	AS7261 . . . . .	11
5.1.2	AS7265X . . . . .	12
5.2	Mikrocontroller . . . . .	13
5.3	I2C Address Translator LTC4316 . . . . .	15
5.4	Companion Flash . . . . .	17
<b>6</b>	<b>Platine</b>	<b>19</b>
6.1	System Topologie . . . . .	19
6.2	Status & Adapterboard . . . . .	19
6.3	Sensorboard . . . . .	20
<b>7</b>	<b>Datenbank &amp; Webinterface</b>	<b>24</b>
7.1	InfluxDB . . . . .	24
7.2	Grafana . . . . .	24
<b>8</b>	<b>C Code</b>	<b>25</b>
8.1	wiringPi . . . . .	25
8.1.1	wiringPil2CSetup . . . . .	25
8.1.2	wiringPil2CWriteReg8 . . . . .	26
8.1.3	wiringPi4l2CReadReg8 . . . . .	26
8.2	AS726X Library . . . . .	26
8.2.1	virtualWriteRegister . . . . .	28
8.2.2	virtualReadRegister . . . . .	30
8.2.3	MeasurementFromAdress . . . . .	32
8.2.4	takeMeasurements . . . . .	32
8.2.5	setMeasurementMode . . . . .	33

8.2.6	dataAvailable & clearDataAvailable . . . . .	34
8.2.7	Rohwerte des AS7261 Auslesen . . . . .	34
8.2.8	getChannel . . . . .	35
8.2.9	Rohwerte des AS7265X Auslesen . . . . .	35
8.2.10	getChannel_AS7265X . . . . .	36
8.2.11	selectDevice . . . . .	36
8.2.12	Enable/Disable Indicator . . . . .	37
8.2.13	softReset . . . . .	37
8.2.14	I2CScan . . . . .	38
8.2.15	getVersion . . . . .	39
8.2.16	scanAS7262 . . . . .	40
8.2.17	scanAS7263 . . . . .	40
8.2.18	setGain . . . . .	40
8.2.19	setIntegrationTime . . . . .	41
8.2.20	disableInterrupt . . . . .	41
8.3	influxDB Library writeToDatabase . . . . .	41
8.3.1	writeToDatabase . . . . .	41
8.4	main . . . . .	42
8.4.1	default_values.h . . . . .	42
8.5	measurement . . . . .	43
8.5.1	fixedGainMeasurementAS7261 & fixedGainMeasurementAS7265X . .	43
8.5.2	autoGainMeasurementAS7261 & autoGainMeasurementAS7265X . .	45
8.5.3	matchValueToMaxGain . . . . .	48
<b>9</b>	<b>Benutzerhandbuch</b> . . . . .	<b>48</b>
9.1	Hardware Setup . . . . .	48
9.2	Messung Starten . . . . .	48
9.3	Webinterface . . . . .	51
9.4	Messsoftware Neustarten . . . . .	52
9.5	IP Adress Scan . . . . .	52
9.6	Bedeutung der Status LEDs . . . . .	53
9.6.1	Rapsberry Pi . . . . .	53
9.6.2	Status & Adapterboard . . . . .	53
9.7	Hilfe . . . . .	53
9.8	Liste Der Verwendeten I2C Adressen & Translationbytes . . . . .	53

<b>10</b>	<b>Messungen</b>	<b>54</b>
<b>11</b>	<b>Zusammenfassung</b>	<b>57</b>

## **2 TODO Liste**

Beispielmessung  
CPU Auslastungs messung  
Speicher auslastung  
Zusammenfassung  
Fix Bib Tex  
FOTO CSV Export  
how to store  
maximus connectable devices

## 3 Einleitung

Es gibt viele wissenschaftliche und industrielle Anwendungen, die zuverlässige Daten über die Spektralezusammensetzung des menschlich sichtbaren Tageslichts erfordern.

Das menschlich sichtbare Licht besteht aus den Wellenlängen der elektromagnetischen Strahlung ( $380 - 780\text{nm}$ ), die ursprünglich von unserer Sonne emittiert werden und direkt oder indirekt die Erde erreichen. Um das Licht über den gesamten sichtbaren Bereich des Himmels zu erfassen, sind mehrere Sensoren zur Messung in verschiedenen Richtungen oder mechanische Komponenten zur Abtastung des Himmels in mehreren Punkten erforderlich.

Das Fachgebiet Lichttechnik benutzt bereits einen Messaufbau. Dieser verwendet einen Spektrometer der durch einen beweglichen Spiegel und eine rotierbare Basis in der Lage ist den Himmel abzutasten. Der Messaufbau kann somit mit einer sehr genauen spektralen Auflösung von 1 nm in 145 Himmelsbereichen messen.

Im Dauereinsatz haben sich mechanische Komponenten unter dem Einfluss äußerer Bedingungen als gravierender Schwachpunkt erwiesen.

Die Messeinrichtung ist aufgrund des hochwertigen Sensors in der Lage sehr genaue Messungen aufzuzeichnen, die Anschaffung ist jedoch mit erheblichen Kosten verbunden.

Aus diesem Grund wird am Fachgebiet für Lichttechnik der Technischen Universität Berlin an einer transportablen, spectral- und richtungsauf lösenden Messeinrichtung geforscht die ohne mechanische Komponenten auskommt und durch die Verwendung von weniger genauen Sensoren eine kostengünstige und robuste Alternative darstellt.

In dieser Arbeit wird mithilfe der von AMS entwickelten auf Fotodioden basierenden Lichtsensoren AS7261 und AS7265X eine Spektral-Sensor Plattform entwickelt, die Modular für unterschiedliche Forschungszwecke, aber vor allem zur Tageslichtmessung, genutzt werden kann.

### 3.1 Aufbau des Messsystems

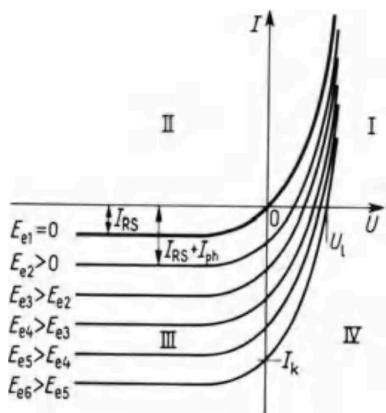
Ein Einplatinencomputer wird mit bis zu Zehn AS7261 und Zehn AS7265X Sensoren verbunden. Die Messungen werden über eine I<sub>2</sub>C Bus Verbindung gesteuert und ausgelesen. Auf dem Einplatinencomputer wird eine Datenbank mit den Messdaten gefüllt und über ein Webinterface welches auch auf dem Einplatinencomputer gehostet wird verfügbar gemacht.

# 4 Technische Grundlagen

## 4.1 Fotodioden

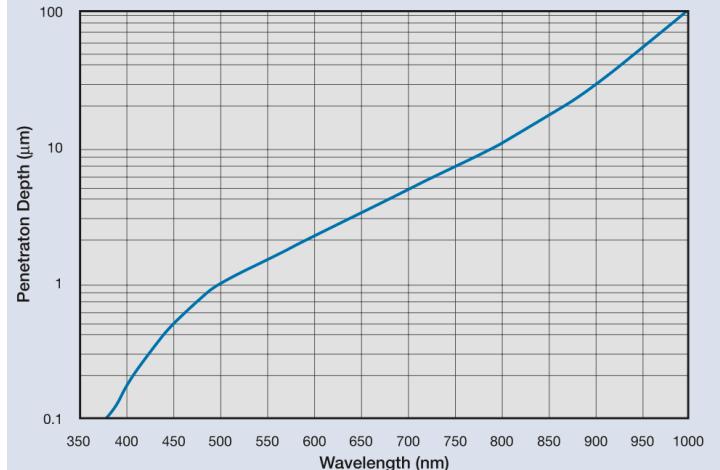
Photodioden sind beleuchtete pn-Übergänge. Im Kurzschlussbetrieb ( $U = 0$ ) fließt ein über einen Bereich von mehr als acht Zehnerpotenzen linear von der Beleuchtungsstärke abhängiger Kurzschlussstrom  $I_k$  (Abbildung 1a) [1]. Dieser setzt sich aus dem durch ein strahlendes Licht verursachten Photostrom und dem Temperaturabhängigen Dunkel Strom zusammen [2]. Allerdings hängt der Photostrom auch von der Eindringtiefe in das Silizium Substrat ab, welche wiederum von der Wellenlänge abhängt (Abbildung 1b) [3]

(a) Kennlinienfeld Photodiode



Kennlinienfeld  $I = f(U)$  der Photodiode mit der Bestrahlungsstärke  $E_e$  als Parameter  
Kurzschlussstrom  $I_k$  bei  $U = 0$

(b) Silizium-Eindringtiefe-Licht



Quelle: osiopptoelectronics.com

Damit Rückschlüsse über den Kurzschlussstrom  $I_K$  zur Lichtintensität zulässig sind, muss also die Temperatur konstant oder zumindest bekannt sein. Außerdem ist es wichtig, bei Tagessichtmessungen, das einfallende Licht auf eine möglichst begrenzten Wellenlängenbereich zu beschränken, da so ein möglichst akkurate Temperatur und Frequenzabhängiger kompensationsfaktor gewählt werden kann.

## 4.2 I2C

I2C ist ein simples und effizientes Busprotokoll. Es wurde ursprünglich von Phillips entwickelt, wird aber seit einigen Jahren von NXP weiterentwickelt. In seiner simpelsten Form ermöglicht

es einen Master mit bis zu 128 Slave-Geräten zu verbinden. Dafür werden nur 2 Leitungen benötigt, die SCL und SDA genannt werden. SCL ist die Takteleitung. Sie wird verwendet, um alle Datenübertragungen über den I2C-Bus zu synchronisieren. SDA ist die Datenleitung. Außerdem müssen alle Busteilnehmer mit dem gleichen GND potential verbunden sein um Stromfluss über die SDA und SCL Leitungen zu ermöglichen [4].

SCL und SDA benötigen einen Pull-Up-Widerstand das sie als "Open Drain" betrieben werden, was bedeutet, dass die Leitungen über Widerstände (Pull-Up-Widerstände) mit der Versorgungsspannung verbunden sind und somit in den High-Zustand gezogen werden. Die angeschlossenen Geräte ändern den Zustand einer Datenleitung auf Low, indem sie die Leitung über einen MOSFET mit GND verbinden. Um die Leitung wieder auf High zu heben, wird die Verbindung zu GND wieder getrennt und der Pull-Up-Widerstand sorgt dafür, dass die Leitung wieder im High-Zustand ist.

Die Clockleitung SCL wird nur vom Bus Master gesteuert so kann der Master den tackt der Datenübertragung bestimmen. Die SDA Leitung wird vom Master und Slave genutzt um Daten zu übertragen allerdings antworten die Slaves im Normalbetrieb nur nachdem sie vom Master auf ihrer Adresse eine Anfrage erhalten haben. Die Spezifikation des Protokolls [5] empfiehlt die SDA und SCL Leitung möglichst weit voneinander zu entfernen um so die Signalqualität zu verbessern.

## 5 Hardware Komponenten

## 5.1 Sensoren

Die Sensoren aus der AS726X Reihe sind in der Lage Licht, also elektromagnetische Strahlung zu messen. In jedem Sensor sind 6 Photodioden verbaut. Vor jeder Photodiode ist ein Silizium-Interferenzfilter montiert, welcher wie ein Bandpassfilter arbeitet, er ist nur für einen bestimmten Ausschnitt des Lichtspektrums durchlässig. Jeder Baustein enthält einen Analog-digital-Wandler mit 16 Bit Auflösung, der den Photostrom aus den Fotodioden integriert. Nach Abschluss einer Messung wird das integrierte Ergebnis in die entsprechenden RAW Data Register übertragen.

So kann über das beschriebene Sensorarray die farbliche Zusammensetzung des eingestrahlten Lichts erfasst werden.



Abbildung 2: AS726X [6]

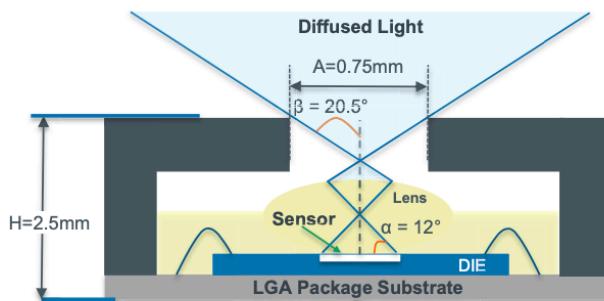


Abbildung 3: Seitenansicht AS726X [6]

Wie in Abb: 3 dargestellt, fällt das Licht durch die Öffnung in der Mitte des Sensors ein, eine intern verbaute Linse verteilt das Licht auf die Interferenzfilter. Die Genauigkeit der Filter verändert sich je nach Einstrahlwinkel, daher ist es wichtig vor den Sensor noch eine Streuscheibe zu montieren.

Die so gemessenen Daten können über UART oder I2C an einen Mikrocontroller übertragen werden, da über UART nur ein Gerät verbunden werden kann, eignet sich aber für diesen Anwendungsfall nur die I2C Schnittstelle.

Alle Sensoren erhalten vom Hersteller dieselbe, nicht veränderbare I2C Adresse: 0x49. Daher muss ein I2C Translator genutzt werden, welcher es ermöglicht mehrere Sensoren im gleichen Bussystem zu adressieren (siehe Abschnitt 5.3).

Die Modelle unterscheiden sich durch die verbauten Silizium-Interferenzfilter, also die unterschiedbaren Wellenlängenbereiche sowie in der benötigte Peripherie. Die grundsätzliche Messmethode ist aber immer gleich. Im Folgenden werden die verwendeten Sensoren AS7261 (siehe Abschnitt 5.1.1) sowie der AS7265X (siehe Abschnitt 5.1.2) beschrieben.

Intern im Sensor werden aus den RAW-Werten kalibrierte Werte berechnet, aber es wird keine ausreichende Dokumentation bereitgestellt. Daher konnte keine konsistente Korrelation zwischen RAW- und kalibrierten Werten ermittelt werden, weshalb sie im Messaufbau nicht verwendet werden.

### 5.1.1 AS7261

Das Sensorarray des AS7261 unterscheidet zwischen X, Y, Z, Clear, Dark und NIR. Wobei X, Y und Z nach CIE 1931 definiert sind. Clear ist eine Fotodiode ohne Filter. Dark ist eine abgedunkelte Fotodiode und wird nur verwendet um störende Einflüsse zu detektieren. NIR steht für Near-infrared (Nah-Infrarot).

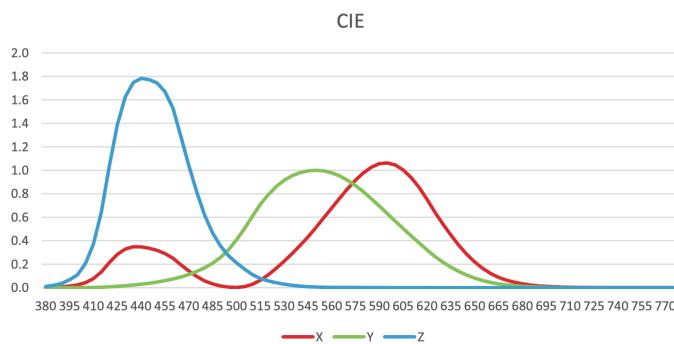


Abbildung 4: AS7261 Spektrale Empfindlichkeit von X,Y,Z [7]

#### Bank Modes:

Es gibt 3 sogenannte Bank Modes, in denen der Sensor arbeiten kann.

#### Bank Mode 0

Die Konvertierungen erfolgen kontinuierlich und Daten sind in den I2C-Registern X, Y, Z und NIR verfügbar.

#### Bank Mode 1

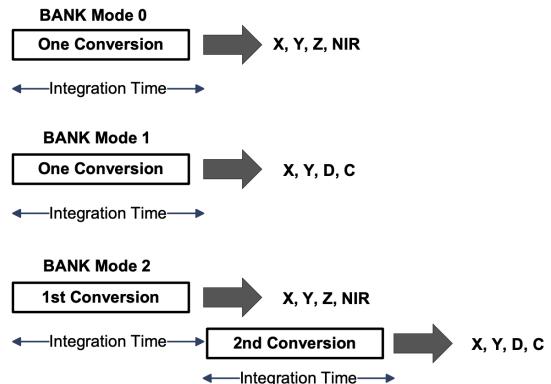
Die Konvertierungen erfolgen kontinuierlich und Daten sind in den I2C-Registern X, Y, D und C verfügbar.

#### Bank Mode 2

Die Konvertierungen erfolgen kontinuierlich, und Daten sind nach zwei Integrationsperioden in den Registern X, Y, Z, NIR, D und C verfügbar. In diesem Modus können auch die kalibrierten, korrigierten Werte aus den entsprechenden I2C-Registern abgerufen werden.

#### Bank Mode 3

Abbildung 5: AS7261-Bank Modes



Quelle: Datenblatt AS7261

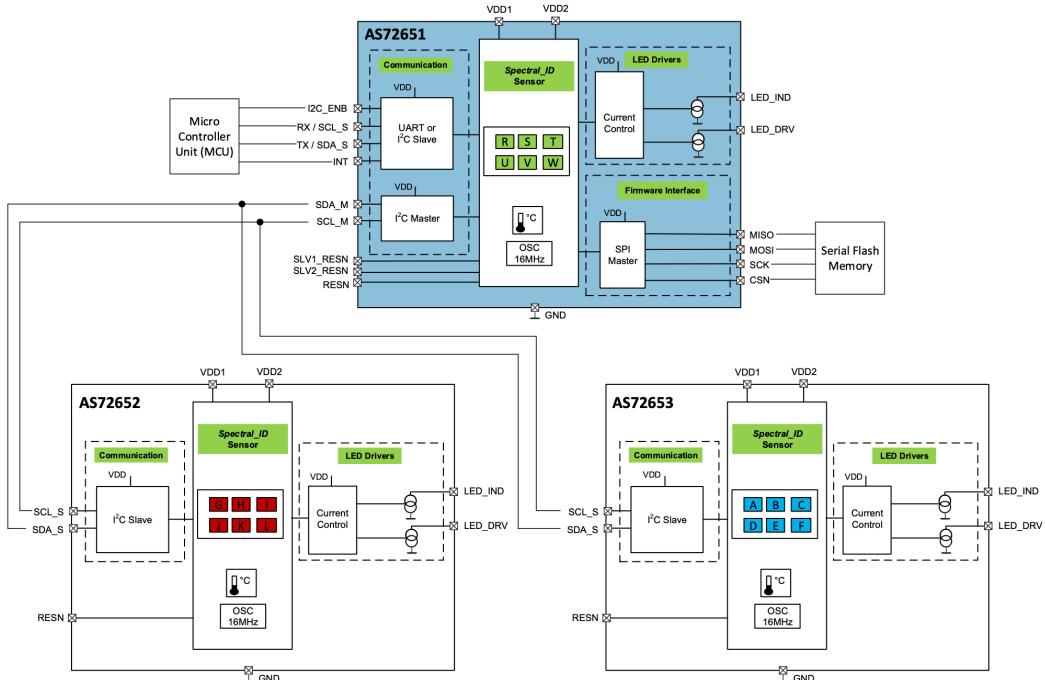
Die Konvertierungen erfolgen nur einmal, und Daten sind wie in Bank Mode 2 nach zwei Integrationsperioden in den Registern X, Y, Z, NIR, D und C verfügbar. Auch die kalibrierten, korrigierten Werte können aus den entsprechenden I2C-Registern abgerufen werden. Das DATA RDY-Bit wird auf 1 gesetzt, sobald Daten verfügbar sind.

Für den hier beschriebenen Messaufbau wird Bankmode 3 verwendet, da so an alle angeschlossenen Sensoren möglichst gleichzeitig eine Messung gestartet werden kann. Die Daten können nach abgeschlossener Messung an den Raspberry Pi übertragen werden.

### 5.1.2 AS7265X

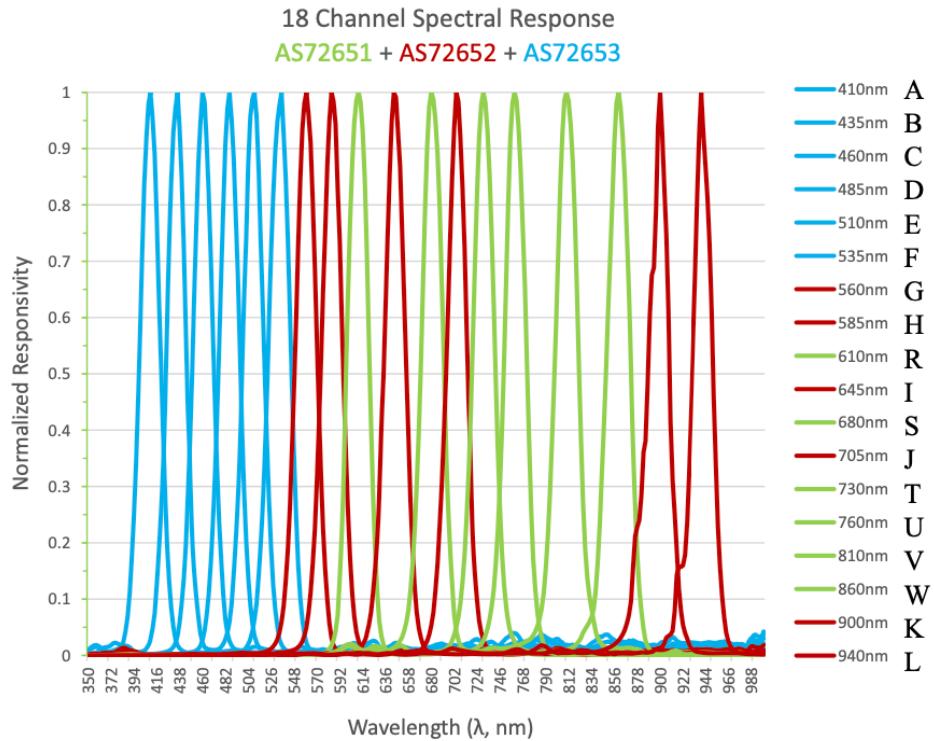
AS7265X beschreibt AS72651, AS72652 und AS72653 wobei der AS72651 als Master für AS72652 und AS72653 fungiert (Abbildung 6), indem er über einen weiteren separaten I2C Bus ihre Daten abfragt und ansonsten wie der AS7261 arbeitet. Der angeschlossene Mikrocontroller (Einplatinencomputer) kommuniziert nur mit dem AS72651.

Abbildung 6: S7265X-Scematic



Quelle: Datenblatt AS7265X

Abbildung 7: AS7261-Spectral Responsivity



Quelle: Datenblatt AS7265X

Die drei Sensoren messen in Kombination mit 18 Photodioden mit unterschiedlichen filtern, so können sie 18 unterschiedliche Frequenz Channel im Bereich zwischen 410 nm und 940 nm mit einer Halbwertsbreite von jeweils 20 nm erfassen. Die Frequenz Channel sind wie in Abbildung 7 zu sehen mit den Buchstaben A-L gekennzeichnet.

Wie der AS7261 kann der AS7265X in 3 Bankmodi betrieben werden, wobei auch für diesen Sensor der Bankmodus 3 verwendet wird, da alle Farbkanäle in einer manuell ausgelösten Messung aufgezeichnet werden sollen.

## 5.2 Mikrocontroller

Bei der Auswahl des Mikrocontrollers ist die kleine Bauform, ausreichend langlebiger Speicher sowie eine Netzwerkschnittstelle und I2C Anschluss entscheidend.

Die Abfrage der Messdaten, sowie die Messkonfiguration soll über einen Fernzugriff möglich sein. Die Daten sollen grafisch in einem Webinterface dargestellt werden ohne das eine weitere Server Instanz benötigt wird. Daher eignet sich ein Linux-Fähiger Einplatinencomputer

besser als ein simplerer Mikrocontroller. Außerdem ermöglicht ein Einplatinencomputer einfache nachträgliche Änderungen ohne das eine komplexe Entwicklungsumgebung eingerichtet werden muss.

Der Raspberry Pi 4 Model B erfüllt alle diese Anforderungen:

Abmessungen	85.6 mm × 56.5 mm
Speicher	eMMC Flash Module Socket
Anschlüsse	Gigabit Ethernet, USB, WLAN
GPIO	UART, I2C, IO
RAM	2 GB
CPU	Broadcom BCM2711 ARM Cortex A72
Preis	37€ (+20€ für Netzteil und 32GB SD-Karte)



Abbildung 8: Raspberry Pi 4 Model B [?]

An den SD-Karten slot kann bis zu 256 GB Flash speicher angeschlossen werden. TODO warum reichen 32 GB.

Die Internetverbindung kann über WLAN oder Ethernet hergestellt werden.

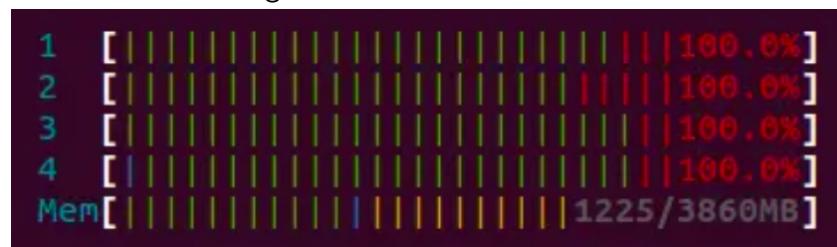
Da eine kabelgebundene Lösung mehr Zuverlässigkeit bietet, wird eine Netzwerkverbindung über Ethernet einer WLAN-Schnittstelle vorgezogen. Die GPIO des Raspebrry Pi arbeiten mit 3,3V. Der 3,3V Pin des Raspebrry Pi stellt nur 850mA am 3,3V Output bereit. Da jedes Sensorboard etwa 170 mA benötigt, reichen 850 mA nicht für die maximale Anforderung von 10 Boards aus. Der 5V-Pin ist direkt mit dem Netzteil verbunden und kann daher genauso viel Leistung wie das Netzteil liefern, mit Hilfe eines Abwärtswandlers ist es möglich, die Sensorplatten mit 3,3V zu betreiben. Die CPU ist für den Anwendungsfall weitaus ausreichend

dimensioniert. In Abbildung 9 ist ein Performancetest zu sehen, die Reihen 1-4 beschreiben die prozentuale CPU Auslastung der jeweiligen Prozessorkerne. Mem beschreibt die Auslastung des Arbeitsspeichers.

Für den Performancetest wurde eine Messung an 10 Sensor Boards mit minimalem Messintervall gestartet. Außerdem wurden gleichzeitig Daten über Grafana exportiert.

TODO: Richtiges Bild Performance Test

Abbildung 9: Screenshot: CPU-Performance



### 5.3 I2C Address Translator LTC4316

Da wie in Abschnitt 5.1 genannt, alle Sensoren unter derselben I2C Adresse erreichbar sind, wird ein I2C Translator genutzt, um für eine individuelle Adressierung zu sorgen.

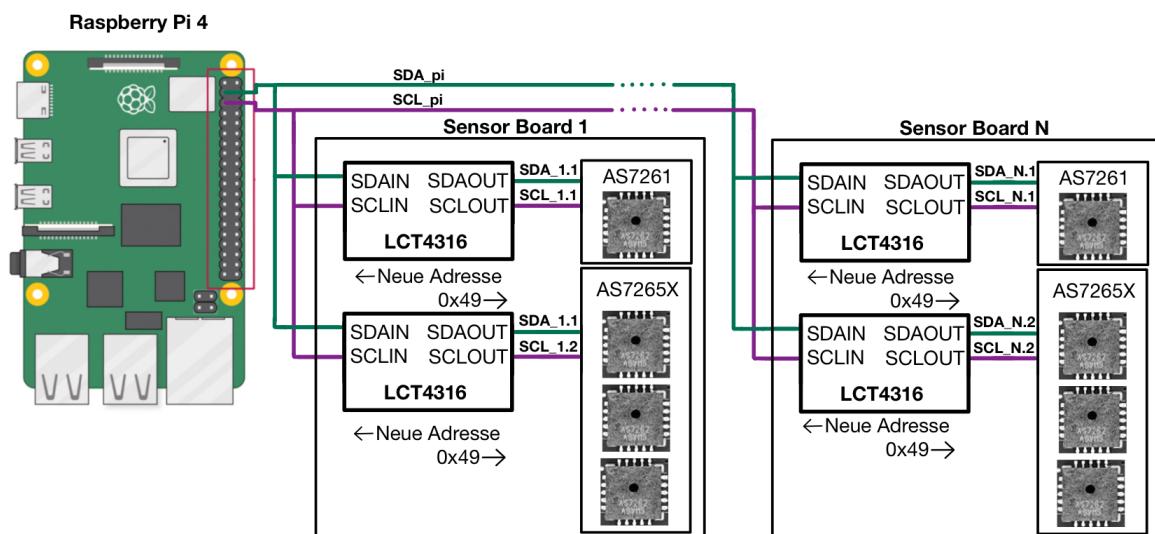


Abbildung 10: I2C-Bus im Messaufbau [8]

Wie in Abbildung ?? zu sehen wird für jeden Sensor ein LTC4316 an die Busschnittstelle des Raspberry Pi angeschlossen (SDAIN, SCLIN).

An jeden LTC4316 wird ein AS7261 oder AS72651 angeschlossen (SDAOUT, SCLOUT). Bei Kommunikation vom Raspberry Pi zum Sensor wird dann die I2C Adresse mit einem Faktor (Translation Byte), welcher mit diskreten Widerständen eingestellt wird mit Formel (1) verrechnet, um so die Adresse anzupassen (XORH,XORL). Um das Translation Byte einzustellen müssen die Widerstände  $R_{HT}$ ,  $R_{LT}$ ,  $R_{HB}$  und  $R_{LB}$  wie in Abbildung 11 am LTC4316 angeschlossen werden.

$$\text{SensorAdresse} \oplus \text{TranslationByte} = \text{NeueAdresse}^1 \quad (1)$$

### Beispiel Rechnung

$$0x49 \oplus 0x01 = 0x48$$

$$0x49 \oplus 0x02 = 0x4b$$

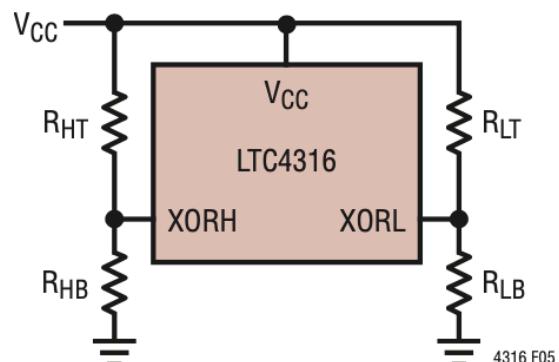
$$0x49 \oplus 0x05 = 0x4c$$

$$0x49 \oplus 0x06 = 0x4f$$

$$0x49 \oplus 0xA = 0x43$$

$$0x49 \oplus 0x49 = 0x00$$

Abbildung 11: Translation Byte [9]




---

<sup>1</sup>xor =  $\oplus$

Tabelle 1 enthält die erforderlichen Widerstandswerte für die diskreten Widerstände  $R_{LT}$  und  $R_{LB}$  und die daraus resultierenden unteren 4 Bits des Translationsbytes. Tabelle 2 enthält die erforderlichen Widerstandswerte für die diskreten Widerstände  $R_{HT}$  und  $R_{HB}$  und die daraus resultierenden oberen 4 Bits des Translationsbytes.

Durch das anlöten der jeweiligen Widerstände können alle 128 mögliche Kombinationen für das Translation byte erreicht werden. (128 Möglichkeiten ergeben sich da Bit a7 immer 0 ist.)

Im Handbuch (Abschnitt 9.8) ist aufgelistet welche Translation Bytes bereits verwendet werden. Wenn weitere Sensor-Boards angefertigt werden ist die Liste dort weiter zu pflegen.

Tabelle 1: Untere 4 Bit des Translation Byte

a3	a2	a1	a0	$R_{LT}[\Omega]$	$R_{LB}[\Omega]$
0	0	0	0	Open	Short
0	0	0	1	976	102
0	0	1	0	976	182
0	0	1	1	1000	280
0	1	0	0	1000	392
0	1	0	1	1000	523
0	1	1	0	1000	681
0	1	1	1	1000	887
1	0	0	0	887	1000
1	0	0	1	681	1000
1	0	1	0	523	1000
1	0	1	1	392	1000
1	1	0	0	280	1000
1	1	0	1	182	976
1	1	1	0	102	976
1	1	1	1	Short	Open

## 5.4 Companion Flash

Die Sensoren AS7261 und AS72651 benötigen einen Flash-Speicher von welchem sie ihre Firmware laden können. Die jeweilige Firmware von AMS wird mithilfe von Flashcat-USB, einem USB Memory Programmer über das SPI (Serial Peripheral Interface) auf den den Flash

Tabelle 2: Obere 4 Bit des Translation Byte

a7	a6	a5	a4	$R_{HT}[\Omega]$	$R_{HB}[\Omega]$
0	0	0	0	Open	Short
0	0	0	1	976	102
0	0	1	0	976	182
0	0	1	1	1000	280
0	1	0	0	1000	392
0	1	0	1	1000	523
0	1	1	0	1000	681
0	1	1	1	1000	887

Speicher übertragen. Ein Leitfaden für diesen Vorgang wird von AMS unter dem Titel “How to Program AS72xx Firmware with FlashCatUSB” bereitgestellt und ist im Anhang zu finden. Der AT25SF041-SSHD-B wurde aus der von AMS bereitgestellten Liste kompatibler Flash-Speichern<sup>2</sup> ausgewählt, da er am günstigsten ist. Für den Messaufbau werden folgende Versionen verwendet die .bin Files sind im Anhang zu finden, da die Firmware nur auf Nachfrage bei AMS erhältlich ist:

AS7261\_complete ??

AS726X\_AS7265\_complete\_moonlight\_v1 ??

---

<sup>2</sup>AS726x Design Considerations - 2.5 Flash Memory

# 6 Platine

## 6.1 System Topologie

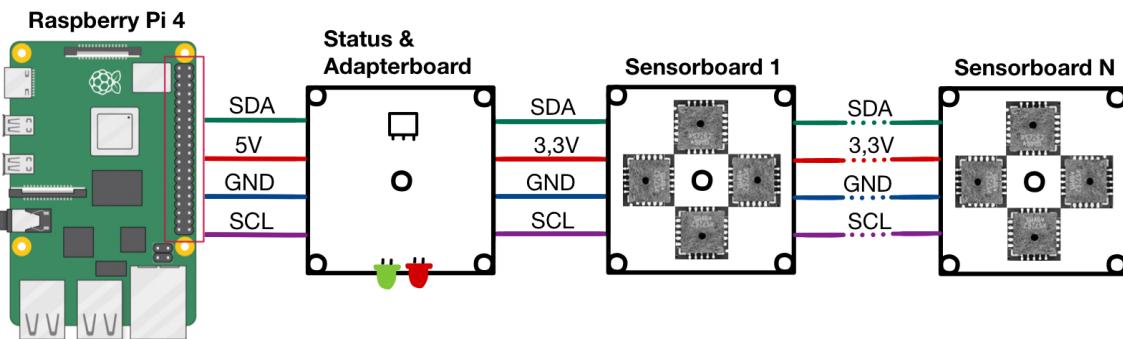


Abbildung 12: Verschaltung der der Platinen [8]

Der Messaufbau besteht aus einem Rapberry Pi 4 Model B welcher über eine Status und Adapterplatinene mit 1-10 Sensnorboards verbunden werden kann (Abbildung 12).

## 6.2 Status & Adapterboard

Da wie in Abschnitt 5.2 erklärt der Rapberry Pi nicht die benötigte 3,3V Stromversorgung bereitstellt wird eine Adapterboard (Abbildung 13) mit einem Spannungswandler (LM3940IT-3.3) verwendet. Auf dem Adapterboard findet der einheitliche Steckverbinder sowie die Pull Up widerstände des I2C Bus Platz, da das Adapterboard aufgrund seiner Bauform nicht falsch montiert werden kann so die Verpolung der Sensoren ausgeschlossen werden. Die Bedeutung der Status LED wird im Handbuch unter Abschnitt 9.6 erläutert.

Der überflüssige Platz auf dem Adapterboard wurde für einen Prototyping-Bereich genutzt.

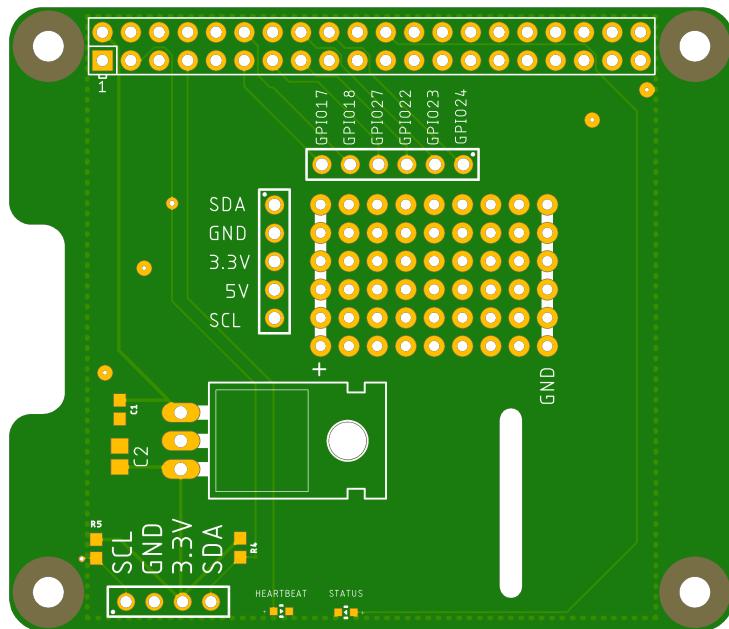


Abbildung 13: Platine des Status & Adapterboard

### 6.3 Sensorboard

Die Hauptaufgabe der Sensorboard Platine ist es den AS7261 und der AS7265X mit ihrem Companion Flash (Abschnitt 5.4) und über den I2C Translator (Abschnitt 5.3) dem I2C Bus zu verbunden. Außerdem werden verschiedene benötigte LEDs, Widerstände und Kondensatoren verbaut.

**Status LED:** Am AS7261 und AS7265X Befindet sich jeweils eine Rote Status LED wenn es ein Problem mit dem Companion Flash gibt fängt sie an zu blinken im normalbetrieb kann die LED Softwareseitig ein und ausgeschaltet werden. Während der Messung sollte sie ausgeschaltet werden da das rote Licht sonst die Messung verfälscht.

**Pull-up-Widerstände:** R7 und R8 sind die Pullup Widerstände des seperaten I2C Bus welcher die AS7265X Sensoren miteinander verbindet.

R12 und R11 sind die Pullup Widerstände des I2C Bus welcher den AS7261 mit seinem LCT4316 verbindet.

R4 und R5 sind die Pullup Widerstände des I2C Bus welcher den AS72651 mit seinem LCT4316 verbindet.

**I2C Enable Wiederstände:** Pin R6 ist mit 3.3V und Pin 8 (I2C Enable) des AS7261 verbunden. So wird I2C als Kommunikationsmodus des AS7261 ausgewählt. R3 erfüllt die gleiche Aufgabe für den AS72651.

**Translation Byte Wiederstände:** Die 8 Wiederstände auf der Rückseite R1\_XX und R2\_XX sind die in 5.3 beschriebenen Wiederstände welche das Translation Byte einstellen. Die Wiederstände R1\_XX bestimmen die Adresse des AS7261 und R2\_XX die Adresse des AS72651.

**Entstörkondensatoren:** Im Datenblatt der Sensoren wird empfohlen, für jeden Sensor 2 Parallele Entstörkondensatoren mit den Kapazitäten  $10\mu F$  und  $100nF$  so nah wie möglich am Sensor zwischen GND und VCC zu platzieren.

Kürzel	Wert	Sensor
C5	$10\mu F$	AS7261
C4	$100nF$	AS7261
C1	$10\mu F$	AS72651
C8	$100nF$	AS72651
C2	$10\mu F$	AS72652
C3	$100nF$	AS72652
C7	$10\mu F$	AS72653
C6	$100nF$	AS72653

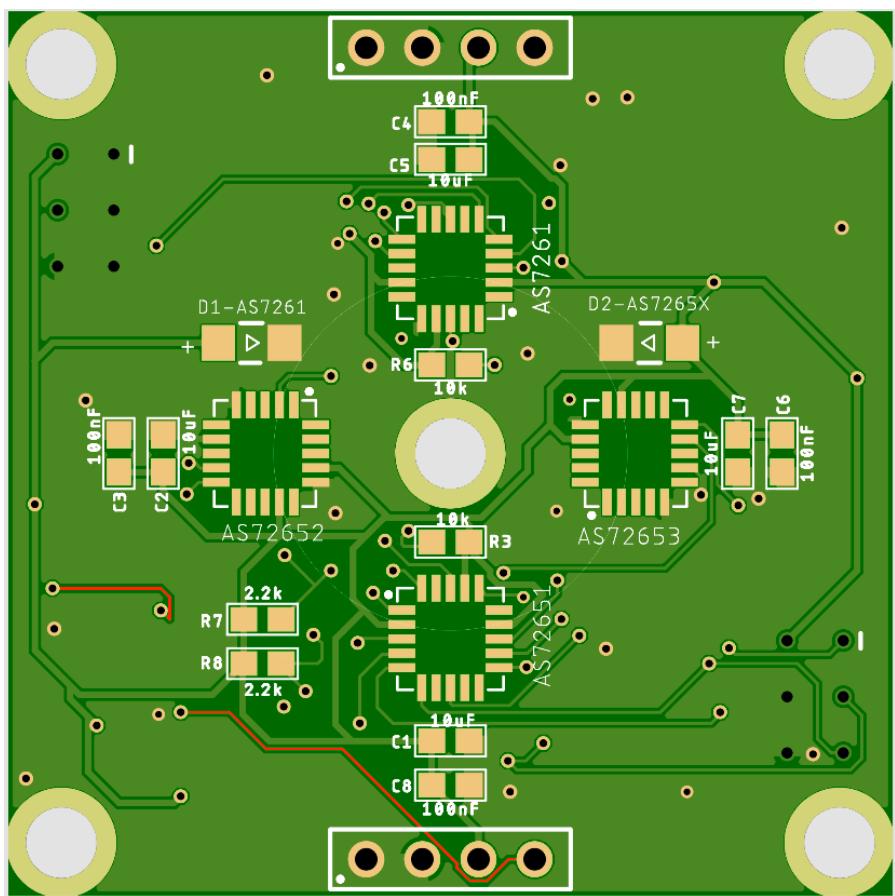


Abbildung 14: Vorderseite des Sensorboards I2C Leitungsverlauf in Rot

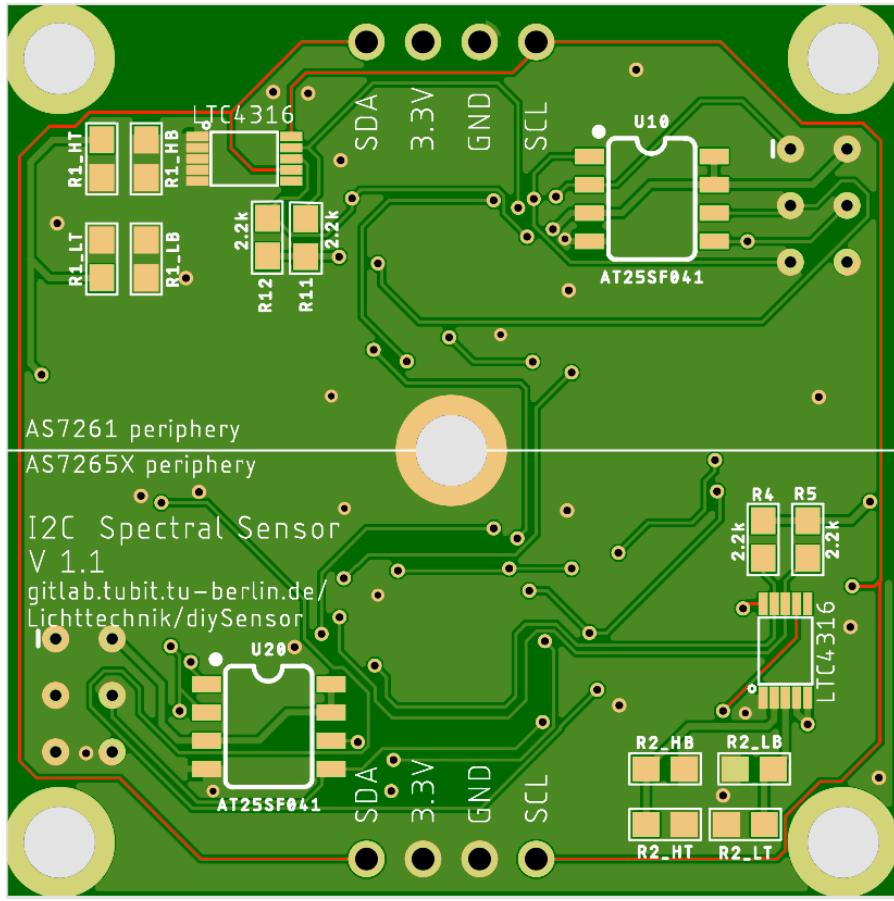


Abbildung 15: Rückseite des Sensorboards I2C Leitungsverlauf in Rot

**I2C Lanes:** Die I2C Leitungen haben ihren Ursprung auf der Adapterplatine und werden mithilfe der seitlichen Steckverbinder über die Sensorboards durchgeschleift. Die maximal mögliche Länge einer I2C Leitung hängt von der Länge der Leitungskapazität sowie äußeren Störeinflüssen ab. Die Data und die Clock Leitung sind möglichst weit von anderen Datenleitungen, also auch von einander entfernt platziert, da so Störeinflüsse durch elektromagnetische Felder minimiert werden. Außerdem wurde darauf geachtet, dass die beiden Leitungen auf den Platinen die gleiche Länge haben, da sich sonst die Differenzen der Leitungslänge mit jeder angeschlossenen Platine addieren und es zu Timing Differenzen zwischen der Daten und Clock Leitung kommt. Der Verlauf der Leitungen ist in Abbildung 15 und 14 Rot gekennzeichnet.

**Connector:** Es gibt 4 durchkontakte Lcher auf der rechten und linken Seite der Platine, hier knnen unterschiedliche Steckverbinder mit 2.54 mm Pitch montiert werden. Es empfiehlt sich, verpolungssichere Steckverbinder zu verwenden, um Hardware Schden vorzubeugen.

Für dieses Bauteil wurde keine SMD-Technik, sondern Durchsteckmontage gewählt, da so eine bessere mechanische Festigkeit erreicht wird. Außerdem kann so alternativ zu einem Steckverbinder direkt ein Flachbandkabel angelötet werden.

## 7 Datenbank & Webinterface

Die Messdaten müssen sicher, einfach zugänglich und möglichst Speichereffizient abgelegt werden. Dafür eignet sich InfluxDB<sup>3</sup> in Kombination mit einem Grafana<sup>4</sup> Webinterface

### 7.1 InfluxDB

InfluxDB ist eine von InfluxData entwickelte Open Source Time Series Database. Time Series Database sind darauf optimiert zusammengehörige Paare von Zeit(en) und Wert(en) zu verwalten und zu komprimieren. InfluxDB hat eine große Benutzer- und Entwicklerbasis, es gibt vielseitige Interfaces um Daten zu visualisieren, darunter Grafana. Eine Time Series Database eignet sich für den Messaufbau da die Spectrallmessdaten mit Zeitbezug erfasst werden.

### 7.2 Grafana

Grafana ist eine Open Source Anwendung zur grafischen Darstellung von Daten aus verschiedenen Datenbanken (InfluxDB, MySQL,...) in einem Webinterface. Das Webinterface ermöglicht sogenannte Dashboards (Abb. 16) anzulegen, in welchen die gespeicherten Daten vielseitig dargestellt werden können. Außerdem können die Daten im CSV-Format exportiert werden. Die Webschnittstelle eignet sich für den Messaufbau, da sie eine leicht verständliche Live-Überwachung der Messungen bietet.

---

<sup>3</sup>influxdata.com

<sup>4</sup>Grafana.com

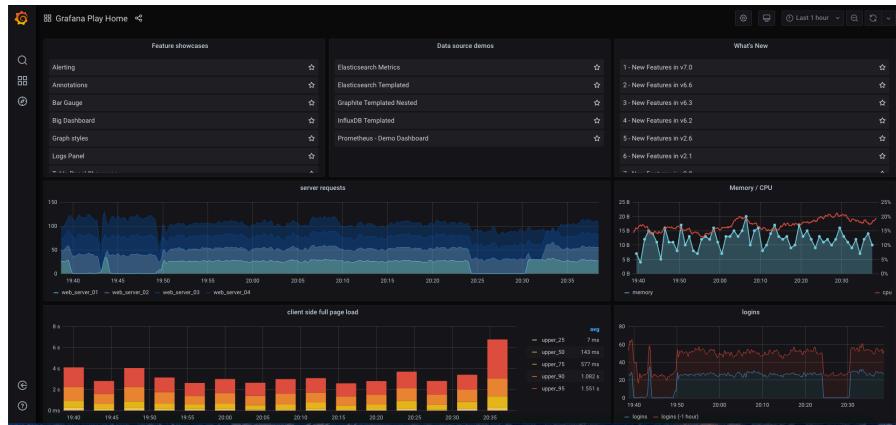


Abbildung 16: Beispiel Grafana Dashboard

## 8 C Code

Der Programmcode für den Messaufbau wurde in C geschrieben da es einfach möglich ist, auf die GPIO des Raspberry Pi zuzugreifen im Übrigen ist die Sprache am Fachgebiet für Lichttechnik weitgehend geläufig.

Das folgende Kapitel ist in mehrere Punkte unterteilt. Zunächst werden die Funktionen der grundlegenden I2C Schnittstelle wiringPi Library vorgestellt (Abschnitt 8.1.1). Anschließend werden alle notwendigen Funktionen für die verwendeten AMS-Sensoren (Abschnitt 8.2) und die Schnittstelle zur Influx-Datenbank (Abschnitt 8.2) erläutert. Abschließend wird der eigentliche Programmablauf aus der C-Datei main.c (Abschnitt 8.4) und measurement.c (Abschnitt 8.5) dargestellt.

Der Programmcode ist unter [todo:gitlab link](#) und im Anhang zu finden.

### 8.1 wiringPi

Auf dem Rapberry Pi 4 ist die wiringPi Libarry vorinstalliert. Die Funktionalität der drei verwendeten C-Funktionen wird im folgenden beschrieben.

#### 8.1.1 wiringPil2CSetup

wiringPil2CSetup baut eine I2C-Verbindung zu einem Slave-Gerät auf. Der Funktion wird beim Aufruf die I2C Adresse übergeben mit welcher eine Verbindung aufgebaut werden soll: `wiringPiI2CSetup(address)`.

Der Rückgabewert ist der Standard Linux File Descriptor oder -1, falls ein Fehler auftritt. Der File Descriptor wird im Programmcode mit fd abgekürzt. Unter Linux ist ein Dateideskriptor ein abstrakter Indikator, der für den Zugriff auf eine Datei oder eine andere Ein-/Ausgabe-Ressource, wie z.B. eine Pipe, einen network socket oder eine I2C Verbindung. Die Anzahl der File Descriptoren ist begrenzt daher muss die Verbindung mit der Funktion `close()` aus der Library: `unistd.h` geschlossen werden wenn sie nicht mehr benötigt wird.

### 8.1.2 `wiringPiI2CWriteReg8`

`wiringPiI2CWriteReg8` überträgt ein Byte an einen I2C-Slave. Der Funktion wird beim Aufruf der File Descriptor einer aktiven I2C-Verbindung sowie ein Zielregister und die zu schreibenden 8bit Daten übergeben:

`wiringPiI2CWriteReg8(fd, RegAdr, 8BitData).`

Wenn der Schreibzugriff vom I2C-Slave bestätigt wurde wird eine 0 zurück gegeben.

### 8.1.3 `wiringPiI2CReadReg8`

`wiringPiI2CReadReg8` überträgt ein Byte an einen I2C-Slave. Der Funktion wird beim Aufruf der File Descriptor einer aktiven I2C Verbindung sowie ein Zielregister übergeben:  
`wiringPiI2CReadReg8(fd, RegAdr).`

Der gelesene 8-Bit-Inhalt des Registers ist der Rückgabewert. Wenn das Lesen Fehlschlägt, bleibt das Programm in einer Endlosschleife hängen.

## 8.2 AS726X Library

Die WiringPi\_AS726X\_Library enthält alle Funktionen um den AS7261 und AS7265X zu steuern und auszulesen. Da der Beispiel Code und die sonstigen Angaben im Datenblatt in vielen Detailfragen ungenau und Fehlerhaft sind, wurde die Arduino OpenSource Libarrys von sparkfun SparkFun\_AS726X\_Arduino\_Library-master und SparkFun\_AS7265x\_Arduino\_Library als Implementierungsgrundlage verwendet. Im ersten Schritt der Entwicklung wurde sie für die I2C Schnittstelle, wiringPi Libarry des Rapberry Pi 4 umgeschrieben und anschließend in ihrer Funktionalität erweitert, um für das Messsystem mit mehreren Sensoren auf dem gleichen I2C-Bus nutzbar zu sein.

Im Folgenden Text werden die Registeradressen mit den gleichen Namen wie im Programmcode bezeichnet die numerische Adressen sowie ihre Aufgaben sind in den Abbildungen 17-22 zu finden.

STATUS Register	I <sup>2</sup> C slave interface STATUS register Read-only	Register Address = 0x00 Bit 1: TX_VALID 0 → New data may be written to WRITE register 1 → WRITE register occupied. Do NOT write. Bit 0: RX_VALID 0 → No data is ready to be read in READ register. 1 → Data byte available in READ register.
WRITE Register	I <sup>2</sup> C slave interface WRITE register Write-only	Register Address = 0x01 8-Bits of data written by the I <sup>2</sup> C Master intended for receipt by the I <sup>2</sup> C slave. Used for both virtual register addresses and write data.
READ Register	I <sup>2</sup> C slave interface READ register Read-only	Register Address = 0x02 8-Bits of data to be read by the I <sup>2</sup> C Master.

Abbildung 17: PhysicalRegister

Abbildung 18: AS726x\_CONTROL\_SETUP 0x04

Bit	Bit Name	Default	Access	Bit Description
7	RST	0	R/W	Soft Reset, Set to 1 for soft reset, goes to 0 automatically after the reset
6	INT	0	R/W	Enable interrupt pin output (INT), 1: Enable, 0: Disable
5:4	GAIN	10	R/W	Sensor Channel Gain Setting (all channels) 'b00=1x; 'b01=3.7x; 'b10=16x; 'b11=64x;
3:2	BANK	10	R/W	Data Conversion Type (continuous) 'b00=Mode 0: X, Y, Z and NIR 'b01=Mode 1: X, Y, D and C 'b10=Mode 2: X, Y, Z, NIR, D and C 'b11=Mode 3: One-Shot operation
1	DATA_RDY	0	R/W	1: Data Ready to Read, sets INT active if interrupt is enabled. Can be polled if not using INT.
0	RSVD	0	R	Reserved; Unused

Quelle: Datenblatt AS7261

Abbildung 19: AS726x\_INT\_T 0x05

Bit	Bit Name	Default	Access	Bit Description
7:0	INT_T	0xFF	R/W	Integration time = <value> * 2.8ms

Quelle: Datenblatt AS7261

Abbildung 20: AS726x\_DEVICE\_TEMP 0x06

Bit	Bit Name	Default	Access	Bit Description
7:0	Device_Temp	0xFF	R/W	Internal device temperature data byte (°C)

Quelle: Datenblatt AS7261

Abbildung 21: AS726x\_LED\_CONTROL 0x07

Bit	Bit Name	Default	Access	Bit Description
7:6	RSVD	0	R	Reserved
5:4	ICL_DRV	00	R/W	LED_DRV current limit 'b00=12.5mA; 'b01=25mA; 'b10=50mA; 'b11=100mA;
3	LED_DRV	0	R/W	Enable LED_DRV 1: Enabled; 0: Disabled
2:1	ICL_IND	00	R/W	LED_IND current limit 'b00=1mA; 'b01=2mA; 'b10=4mA; 'b11=8mA;
0	LED_IND	0	R/W	Enable LED_IND 1: Enabled; 0: Disabled

Quelle: Datenblatt AS7261

Abbildung 22: AS7265X\_DEV\_SELECT\_CONTROL 0x4F

Bit	Bit Name	Default	Access	Bit Description
5	AS72653_id	0	R	Second slave Available
4	AS72652_id	0	R	First slave available
1:0	SELECT DATA	00	R/W	0x00: Select master data 0x01: Select first slave data 0x02: Select second slave data

Quelle: Datenblatt AS7265X

### 8.2.1 virtualWriteRegister

Wie bei Embedded-Geräten üblich werden Einstellungen auf dem Sensor verändert indem verschiedene sogenannte Special Function Register mit Daten beschrieben werden.

Jedes Special Function Register ist 8Bit groß und hat eine Adresse. Jedes Bit des Register repräsentiert eine Einstellung. Beispielsweise ist 0x07 (Abbildung 21) das LED Control Register

Tabelle 3: Raw Data Registers

Name	Adresse
AS7261_X	0x08
AS7261_Y	0x0A
AS7261_Z	0x0C
AS7261_NIR	0x0E
AS7261_DARK	0x10
AS7261_CLEAR	0x12
AS7265X_R_G_A	0x08
AS7265X_S_H_B	0x0A
AS7265X_T_I_C	0x0C
AS7265X_U_J_D	0x0E
AS7265X_V_K_E	0x10
AS7265X_W_L_F	0x12

des Sensors. Bit 0 des Registers Beschreibt den Zustand der Status LED. Die Restlichen 7 Bit des Registers Beschreibt den Zustand anderer LEDs die für den Messaufbau aber irrelevant sind.

Wird Register 7 mit dem Dezimalwert 0 beschrieben sind alle LED aus, wird es mit dem Dezimalwert 1 beschrieben leuchtet nur die Status LED. Die Register Lassen sich aber nicht direkt beschreiben, stattdessen sind sie als sogenannte virtuelle Register implementiert. Das heißt, das nur Register 0x01 (WRITE Register Abbildung 17) beschrieben werden kann. Um Daten in eins der Special Funktion Register zu schreiben wird die C-Funktion virtualWriteRegister verwendet. Die Funktionsweise lässt sich in 4 Schritten zusammenfassen:

- Zeile 8 wartet bis das WRITE Register leer ist, was angezeigt wird indem das Bit AS72XX\_TX\_VALID im Register AS72XX\_STATUS\_REG den Wert 1 annimmt.
- Zeile 14 Schreibt die virtuelle Adresse in das WRITE Register und setzt zusätzlich Bit 8 des WRITE Register auf 1, um zu zeigen, dass es sich um einen schreibenden Zugriff auf das virtuelle Register handelt.
- Zeile 20 wartet erneut bis das WRITE Register leer ist.
- Zeile 26 schreibt die Daten in das WRITE Register

Der Sensor wird jetzt selber die übertragenen Daten aus dem WRITE Register in das angegebene virtuelle Register kopieren.

---

<sup>1</sup>

<sup>2</sup> //Write to a virtual register in the AS726x  
<sup>3</sup> void virtualWriteRegister(uint8\_t virtualAddr, uint8\_t ←

```

        dataToWrite, int fd){
4         uint8_t status;
5         //Wait for WRITE register to be empty
6         while (1){
7             status = wiringPiI2CReadReg8(fd, AS72XX_SLAVE_STATUS_REG);
8             if((status & AS72XX_SLAVE_TX_VALID) == 0) {
9                 break; // No inbound TX pending at slave. Okay to ←
10                write now.
11            }
12            delay(POLLING_DELAY);
13        }
14        // Send the virtual register address (setting bit 7 to ←
15        // indicate a write a register).
16        wiringPiI2CWriteReg8(fd, AS72XX_SLAVE_WRITE_REG, ←
17            (virtualAddr | 0x80));
18
19        //Wait for WRITE register to be empty
20        while (1)
21        {
22            status = wiringPiI2CReadReg8(fd, ←
23                AS72XX_SLAVE_STATUS_REG);
24            if ((status & AS72XX_SLAVE_TX_VALID) == 0){
25                break; // No inbound TX pending at slave. Okay to ←
26                write now.
27            }
28            delay(POLLING_DELAY);
29        }
30        // Send the data to complete the operation.
31        wiringPiI2CWriteReg8(fd, AS72XX_SLAVE_WRITE_REG, ←
32            dataToWrite);
33    }

```

---

### 8.2.2 virtualReadRegister

Die Unterschiedlichen Messdaten des Sensors werden in dedizierten Registern gespeichert. Es ist aber nur über den indirekten weg des AS72XX\_READ\_REG und der virtuellen Registeradressen möglich Daten auszulesen. Die Funktionsweise der zum Datenauslesen benötigten Funktion virtualReadRegister lässt sich wieder in 4 Schritte aufteilen.

- Das AS72XX\_READ\_REG wird ausgelesen ohne, dass die Daten verarbeitet werden.  
Dieser Schritt ist wie ein Reset des Registers zu verstehen.

- Zeile 12 schreibt die virtuelle Adresse in das WRITE Register und setzt zusätzlich Bit 8 des WRITE Register auf 0 um zu zeigen das es sich um einen Lesendenzugriff auf das virtuelle Register handelt.
  - Sobald das AS72XX\_STATUS\_REG den Wert AS72XX\_TX\_VALID annimmt sind die Daten aus dem angebenden virtuellen Register in das AS72XX\_READ\_REG kopiert worden.
  - Zeile 28 liest die Daten aus dem S72XX\_READ\_REG
- 

```

1 //Read a virtual register from the AS726x
2 uint8_t virtualReadRegister(uint8_t virtualAddr, int fd){
3     uint8_t status;
4     //Do a prelim check of the read register
5     status = wiringPiI2CReadReg8(fd, AS72XX_SLAVE_STATUS_REG);
6     if ((status & AS72XX_SLAVE_RX_VALID) != 0){ //There is data ←
7         to be read
8         /*uint8_t incoming = */wiringPiI2CReadReg8(fd, ←
9             AS72XX_SLAVE_READ_REG); //Read the uint8_t but do ←
10            nothing with it
11    }
12    //Wait for WRITE flag to clear
13    while (1) {
14        status = wiringPiI2CReadReg8(fd, ←
15            AS72XX_SLAVE_STATUS_REG);
16        if ((status & AS72XX_SLAVE_TX_VALID) == 0){
17            break; // If TX bit is clear, it is ok to write
18        }
19        delay(POLLING_DELAY);
20    }
21    // Send the virtual register address (bit 7 should be 0 to ←
22    // indicate we are reading a register).
23    wiringPiI2CWriteReg8(fd, AS72XX_SLAVE_WRITE_REG, ←
24        virtualAddr);
25
26    //Wait for READ flag to be set
27    while (1)
28    {
29        status = wiringPiI2CReadReg8(fd, ←
30            AS72XX_SLAVE_STATUS_REG);
31        if ((status & AS72XX_SLAVE_RX_VALID) != 0) break; // ←
32            Read data is ready.
33        delay(POLLING_DELAY);
34    }
35
36    //Read the data
37    incoming = wiringPiI2CReadReg8(fd, AS72XX_SLAVE_READ_REG);
38
39    //Clear the READ flag
40    wiringPiI2CWriteReg8(fd, AS72XX_SLAVE_WRITE_REG, ←
41        virtualAddr);
42
43    return incoming;
44}

```

```

26     }
27
28     uint8_t incoming = wiringPiI2CReadReg8(fd, ←
29         AS72XX_SLAVE_READ_REG);
30     return (incoming);
31 }
```

---

### 8.2.3 MeasurementFromAddress

Die Funktion baut einen I2C Verbindung zur übergebenen Bus-Adresse auf und ruft die Funktion takeMeasurements mit dem File Descriptor der aktiven I2C Verbindung auf. Nachdem die Funktion takeMeasurements durchlaufen ist wird die I2C Verbindung wieder geschlossen.

```

1 //Calls takeMeasurements on gives I2C Address
2 void MeasurementFromAddress(int address){
3     int fd = wiringPiI2CSetup(address);
4     if (fd == -1) {
5         printf("i2c failed");
6     }
7     takeMeasurements(fd); // takesMeasuremant Readings can now ←
8         be accessed via getX(), getY(), etc
9     close(fd);
10 }
```

---

### 8.2.4 takeMeasurements

Die Funktion takeMeasurements ruft die Funktion setMeasurementMode mit dem Parameter 3 auf das setzt den aus 5.1.1 bekannten Bank Mode der übergebenen I2C Verbindung (fd) auf Bank Mode 3. Die One-Shot Messung wird sofort gestartet, in Zeile 9 wird gewartet bis die Messung abgeschlossen ist. Um sicherzustellen das die Funktion DataAvailable richtig arbeitet muss vor der Messung das Flag DataAvailable auf 0 gesetzt werden (Zeile 3). Die Daten werden hier nicht ausgelesen daher gibt es keinen Rückgabewert.

```

1 //Tells IC to take measurements and polls for data ready flag
2 void takeMeasurements(int fd) {
3     clearDataAvailable(fd); //Clear DATA_RDY flag when using ←
4         Mode 3
5 }
```

---

```

5      //Goto mode 3 for one shot measurement of all channels
6      setMeasurementMode(3, fd);
7
8      //Wait for data to be ready
9      while (dataAvailable(fd) == 0) delay(POLLING_DELAY); ←
        //Potential TODO: avoid this to get faster nearly ←
        parallel measurements
10
11     //Readings can now be accessed via getViolet(), getBlue(), ←
          etc
12 }
```

---

### 8.2.5 setMeasurementMode

Mit der Funktion setMeasurementMode werden die Bank Mode Bits 2 und 3 des AS726x\_CONTROL\_SETUP Registers mit dem gewünschten Wert für den Bank Mode beschrieben.

Da die anderen Bits des Registers noch weitere Einstellungen repräsentieren welche nicht verändert werden sollen, muss das Register erst ausgelesen werden. Anschließend werden nur die Bank Mode Bits auf 0 gesetzt um im nächsten Schritt mit dem gewünschten neuen Bank Modewert beschrieben zu werden. Die Bedeutung der Bank Modes ist in 5.1.1 erläutert.

```

1 //Sets the measurement mode
2 //Mode 0: Continuous reading of VBGY (7262) / STUV (7263)
3 //Mode 1: Continuous reading of GYOR (7262) / RTUX (7263)
4 //Mode 2: Continuous reading of all channels (power-on default)
5 //Mode 3: One-shot reading of all channels
6 void setMeasurementMode(uint8_t mode, int fd) {
7     if (mode > 0b11) mode = 0b11;
8
9     //Read, mask/set, write
10    uint8_t value = virtualReadRegister(AS726x_CONTROL_SETUP, ←
11        fd); //Read
12    value &= 0b11110011; //Clear BANK bits
13    value |= (mode << 2); //Set BANK bits with user's choice
14    virtualWriteRegister(AS726x_CONTROL_SETUP, value, fd); ←
          //Write
15 }
```

---

### 8.2.6 dataAvailable & clearDataAvailable

Das Bit dataAvailable im Register AS726x\_Control\_Setup wird vom Sensor auf 1 gesetzt, wenn nach einer Messung neue Daten vorhanden sind, Interrupts müssen dafür ausgeschaltet sein. dataAvailable wird auf 0 gesetzt wenn Daten gelesen werden. Wenn eine One-Shot Messung im Bank Mode 3 durchgeführt wird muss das dataAvailable Bit mit der Funktion clearDataAvailable auf 0 gesetzt werden, da sonst nicht sicher gestellt ist das die Daten vor der Messung gelesen wurden besteht die Möglichkeit das sich das Bit fälschlicherweise noch im falsch positiven zustand befindet. Die Funktion dataAvailable gibt den Wert des DataAvailable Bit zurück.

### 8.2.7 Rohwerte des AS7261 Auslesen

Die in Abschnitt 5.1.1 beschrieben 6 Channel des AS7261 werden mit den folgenden Funktionen ausgelesen:

- getX\_CIE(fd)
- getY\_CIE(fd)
- getZ\_CIE(fd)
- getNIR(fd)
- getDark(fd)
- getClear(fd)

Der File Deskriptor einer I2C Verbindung mit einem AS7261 und das zu lesende Register(Abbildung 3) wird als Übergabe Parameter erwartet.

Um den Messwert auszulesen, wir die Funktion getchannel aufgerufen. Der Rückgabewert ist der 16-Bit Messwert aus dem jeweiligen Register vom Datentyp Integer.

---

```
1 //Get RAW AS7261 readings
2 int getX(int fd) { return(getChannel(AS7261_X, fd));}
3 int getY(int fd) { return(getChannel(AS7261_Y, fd));}
4 int getZ(int fd) { return(getChannel(AS7261_Z, fd));}
5 int getNIR(int fd) { return(getChannel(AS7261_NIR, fd));}
6 int getDark(int fd) { return(getChannel(AS7261_DARK, fd));}
7 int getClear(int fd) { return(getChannel(AS7261_CLEAR, fd));}
```

---

### 8.2.8 getChannel

Da die Messdaten 16 Bit groß sind, der Sensor aber nur über 8 Bit Register verfügt werden 2 aufeinander folgende Register ausgelen und im Big-Endian Format in einer 16 Bit Variable gespeichert . Die Funktion getChannel erwartet den File Descriptor einer I2C Verbindung zu einem Sensor und die Adressen des High Bytes eines Raw Data Registers. Der Rückgabewert ist der 16-Bit Messwert aus dem jeweiligen Register vom Datentyp Integer.

---

```
1 //A the 16-bit value stored in a given channel registerReturns
2 int getChannel(uint8_t channelRegister, int fd){
3     int colorData = virtualReadRegister(channelRegister, fd) ←
4         << 8; //High uint8_t
5     colorData |= virtualReadRegister(channelRegister + 1, fd); ←
6         //Low uint8_t
7     return(colorData);
8 }
```

---

### 8.2.9 Rohwerte des AS7265X Auslesen

Da beim Lesen des AS7265X, 3 Sensoren unter der gleichen Adresse erreichbar sind (Abbildung 3), muss zusätzlich zum File Descriptor und des Zielregisters der Device Identifier angeben werden. Die folgenden Funktionen übernehmen diese Aufgaben und können genutzt werden, um Rohdaten auslesen:

AS72651	AS72652	AS72653
getR(fd)	getG(fd)	getA(fd)
getS(fd)	getX(fd)	getB(fd)
getT(fd)	getI(fd)	getC(fd)
getU(fd)	getJ(fd)	getD(fd)
getV(fd)	getK(fd)	getE(fd)
getW(fd)	getL(fd)	getF(fd)

Um den Messwert auszulesen, wir die Funktion getChannel\_AS7265X aufgerufen.

Der Rückgabewert ist der 16-Bit Messwert aus dem jeweiligen Register vom Datentyp integer.

### 8.2.10 getChannel\_AS7265X

Die Funktion verarbeitet den aus Abschnitt 8.2.9 bekannten Device Identifier indem die Funktion selectDevice aufgerufen wird. Da es aber keine Möglichkeit gibt zu überprüfen ob der Gerätewechsel erfolgreich war muss vorher überprüft werden ob das jeweiligen Slavegerät AS72652 oder AS72653 vorhanden ist. Ist ein Slave nicht vorhanden wird der Wert -1 zurück gegeben. Ohne diese Überprüfung ist nicht sichergestellt, dass keine falschen Werte aus den gleich nummerierten Registern des AS72651 ausgelesen werden, obwohl Werte eines nicht vorhanden oder falsch aufgelöten Slavesensors (AS72652 / AS72653) erwartet werden. Das eigentliche Auslesen des 16-Bit Messwerts erfolgt mithilfe der Funktion getChannel, das Ergebnis ist der Rückgabewert der Funktion getChannel\_AS7265X.

---

```
1 // returns Color channel of AS7265X
2 // returns -1 if slave AS72651 or AS72652 is not available
3 int getChannel_AS7265X(int device, uint8_t channelRegister, int ←
4     fd){
5     selectDevice(AS72651_id, fd);    //select AS72651 to verify ←
6         presence of slave sensors
7     if (device == AS72651_id){
8         return getChannel(channelRegister, fd);
9     }
10    else if(device == AS72652_id && scan_AS7262(fd)){
11        selectDevice(device, fd);
12        return getChannel(channelRegister, fd);
13    }
14    else if (device == AS72653_id && scan_AS7263(fd)){
15        selectDevice(device, fd);
16        return getChannel(channelRegister, fd);
17    }
18    return -1;
19 }
```

---

### 8.2.11 selectDevice

Die Notwendigkeit für selectDevice wurde in Abschnitt 8.2.10 bereits erläutert. Laut Datenblatt sollen nur die Bits 0 und 1 des DEV\_SELECT\_CONTROL Registers beschrieben werden, das stimmt aber nicht.

In der Realität muss das gesamte 8-bit Register mit folgenden Werten beschrieben werden

um beim anschließenden Lesevorgang Daten vom jeweiligen Sensor zu erhalten.

DEV_SELECT_CONTROL	Sensor
0x00	AS72651
0x01	AS72652
0x02	AS72653

---

```
1 //Select which AS7265X Device is used
2 //AS72651_id or AS72652_id or AS72653_id
3 void selectDevice(uint8_t device, int fd) {
4     //Set the bits 0:1. Just overwrite whatever is there because ←
5     //masking in the correct value doesn't work.
6     virtualWriteRegister(AS7265X_DEV_SELECT_CONTROL, device, fd);
}
```

---

### 8.2.12 Enable/Disable Indicator

Mit der Funktion enableIndicator wird das 0 Bit des AS726x\_LED\_CONTROL Register auf 1 gesetzt, so wird die rote Status LED des jeweiligen Sensors auf dem Sensorboard angeschaltet.

Mit der Funktion disableIndicator wird das gleiche Bit auf 0 gesetzt, also die LED ausgeschaltet.

Als Übergabeparameter wird bei beiden Funktionen der File Descriptor einer aktiven I2C-Verbindung erwartet.

### 8.2.13 softReset

Wenn einzelne Messwerte außerhalb des Erwartungsbereichs liegen oder ähnliche Probleme auftreten, kann in einigen Fällen ein Soft-Reset das Problem beheben. Da bei der hier beschriebenen Implementierung keine Fehler auftreten, wird die Funktion nicht benötigt, kann aber in zukünftigen Versionen verwendet werden.

Die Funktion softReset setzt das 8 Bit des Registers CONTROL\_SETUP auf 1 um einen softReset auszulösen. Das Datenblatt gibt an, dass mindestens 1000 ms gewartet werden muss, bevor der softReset abgeschlossen ist und der Sensor wieder genutzt werden kann. Für beide Funktionen wird der File Descriptor einer aktiven I2C-Verbindung als Übertragungsparameter

erwartet.

### 8.2.14 I2CScan

Die Funktion I2CScan wird zu Beginn des Programms aufgerufen, um festzustellen, welche Sensoren an den I2C DataBus angeschlossen sind. Die gefundenen Sensoradressen und der Sensortyp werden in das Struct sensor\_list geschrieben.

Die Adressen werden außerdem im Terminal angezeigt, so dass der Benutzer überprüfen kann, ob alle erwarteten Sensoren erkannt werden. Als Übergabeparameter wird im call by reference style ein Pointer zu einem Struct vom Typ sensor\_list erwartet. Da die Daten direkt in das extern deklarierte Struct geschrieben werden gibt es keinen Rückgabewert. Um die angeschlossenen Sensoren zu detektieren wir zu jeder der  $2^8$  möglichen I2C Adressen eine Verbindung aufgebaut und ein Schreibversuch mithilfe der Funktion wiringPiI2CWriteReg8 vorgenommen. Wenn die Funktion wiringPiI2CWriteReg8 eine 0 zurück gibt war der Schreibversuch erfolgreich also muss ein Sensor unter dieser Adresse vorhanden sein. Das im Datenblatt nicht erwähnte Register mit der Adresse 0x05 wurde zufällig ausgewählt da es mit dem Wert 0x01 beschrieben werden kann ohne, dass der Sensor sein Verhalten verändert. Die Funktion getVersion wird genutzt um die Version des gefunden Sensors zu ermitteln. Wenn ein AS72651 erkannt wird, wird auch abgefragt, ob die Slavesensoren AS72652 und AS72653 vorhanden sind, diese Information wird nur im Terminal ausgegeben und nicht in das struct sensor\_list geschrieben, da es für den Programmablauf nicht notwendig ist.

---

```
1 // Scans for sensors on all 128 possible addresses
2 // input pointer to array of sensor_list struct size has to be 128
3 // writes sensor address and type to array of sensor_list struct
4 void I2C_Scan(sensor_list *const s){
5     //printf("test struct address in function %i is value %i\n", s[0].address, s[0].type );
6     printf("-----I2C Scan ----- \n");
7     uint8_t sensor_count = 0;
8     for (int address = 0; address < 128; ++address)
9     {
10         int fd = wiringPiI2CSetup(address);
11         if (fd != -1){
12             //try to write to some hopefully unused register(5)-> return value: 0 indicates that
```

```

        someone was listening
13     //this possibly writes to the Integration Value ←
        Register so make shure to properly set is ←
        before starting a measurement
14     if (wiringPiI2CWriteReg8 (fd, 5, 1) == 0){
15         int version = getVersion(fd);
16         if (version == SENSORTYPE_AS7261){
17             printf("Device at: 0x%X is ←
18                 AS7261\n",address);
19             s[sensor_count].address = address;
20             s[sensor_count].type = SENSORTYPE_AS7261;
21             sensor_count++;
22         }
23         else if(version == SENSORTYPE_AS72651){
24             printf("Device at: 0x%X is AS72651",address);
25             s[sensor_count].address = address;
26             s[sensor_count].type = SENSORTYPE_AS72651;
27             sensor_count++;
28             if (scan_AS7262(fd)){
29                 printf(", AS72652");
30             }
31             if (scan_AS7263(fd)){
32                 printf(", AS72653");
33             }
34             printf("\n");
35         }
36     }
37     close(fd);
38 }
39 printf ("-----\n");
40 }

```

---

### 8.2.15 getVersion

Die Funktion getVersion gibt den Inhalt der Registers AS726x\_HW\_VERSION zurück indem der zurückgegebene Wert mit den beiden SENSORTYPE Werten verglichen wird (SENSORTYPE\_AS7261 und SENSORTYPE\_AS72651) kann festgestellt werden um welchen Sensor es sich handelt. Als Übergabeparameter wird bei beiden Funktionen der File Descriptor einer aktiven I2C-Verbindung erwartet.

### **8.2.16 scanAS7262**

Als Übergabeparameter wird bei der Funktionen der File Descriptor einer aktiven I2C Verbindung mit einem AS72651 erwartet.

Die Funktion überprüft den Status des 4. Bit des Registers DEV\_SELECT\_CONTROL und gibt ihn zurück. Das Bit ist auf 1 gesetzt, wenn der Slavesensor AS7262 vorhanden ist, im Falle der Abwesenheit ist es 0.

Das AS7265X-Datenblatt gibt fälschlicherweise an, dass das 5. Bit geprüft werden muss.

### **8.2.17 scanAS7263**

Als Übergabeparameter wird bei der Funktionen der File Descriptor einer aktiven I2C Verbindung mit einem AS72651 erwartet.

Die Funktion überprüft den Status des 5. Bit des Registers DEV\_SELECT\_CONTROL und gibt ihn zurück. Das Bit ist auf 1 gesetzt wenn der Slavesensor AS7263 vorhanden ist, im Falle der Abwesenheit ist es 0.

### **8.2.18 setGain**

Die Messergebnisse der Sensoren können intern verstärkt werden, was Beispielsweise in dunklen Messumgebungen oder bei der Verwendung einer relativ lichtundurchlässiger Streuscheibe notwendig ist.

Außerdem kann das Speicherregister besser Ausgesteuert werden, um ein genauereres Messergebnis zu erhalten. Um den Verstärkungsfaktor einzustellen, wird die Funktion setGain benötigt.

Die Funktion setGain beschreibt das vierte und fünfte Bit des Registers CONTROL\_SETUP mit einem der vier möglichen Zustände welcher an die Funktion übergebenen wird.

Wert	Verstärkungsfaktor
0	1x
1	3.7x
2	16x
3	64x

Ist der übergebe Wert größer als 3 ist wird das Register auf den Wert 3 gesetzt.

$$\text{Registerwert} = \text{Integrierter Messwert} \cdot \text{Gain} \quad (2)$$

### 8.2.19 setIntegrationTime

Um die Integrationszeit der Messung einzustellen wird in das Register AS726x\_INT\_T ein Wert (integrationValue) zwischen 0 und 255 geschrieben, die Integrationszeit errechnet sich indem dieser Wert mit dem Faktor 2.8ms multipliziert wird. Die Funktion setIntegrationTime erwartet als Übergabeparameter das integrationValue und den File Descriptor sowie eine aktive I2C-Verbindung zu einem Sensor.

$$\text{Integrationszeit} = \text{integrationValue} \cdot 2.8\text{ms} \quad (3)$$

### 8.2.20 disableInterrupt

Die Funktion disableInterrupt setzt das INT Bit im Register AS726x\_CONTROL\_SETUP auf 0 um Interrupts auszuschalten. Da der Interrupt-Pin auf der Sensor-Platine nicht angeschlossen ist, können Messungen nur durchgeführt werden, wenn der Interrupt ausgeschaltet ist. Laut Datenblatt wird der Interrupt beim Systemstart ausgeschaltet.

## 8.3 influxDB Library writeToDatabase

Um die Messdaten der Sensoren in die aus 7.1 bekannte InfluxDB zu schreiben wurde die InfluxDB Library entwickelt, sie enthält nur eine einzige nach außen sichtbare Funktion: writeToDatabase.

### 8.3.1 writeToDatabase

Um die Datenbank zu beschreiben muss zuerst ein Socket geöffnet werden um sich mit der InfluxDB Server-Instanz zu verbinden. der InfluxDB Sever ist unter der Localhostaddress 127.0.0.1 am Port 8086 zu erreichen. Diese Werte können in der influxdb.h Datei angepasst werden falls in Zukunft die Notwendigkeit besteht den Server auf einem externen Gerät zu betreiben.

Die Kommunikation mit dem InfluxDB Server erfolgt über das HTTP basierte InfluxDB-Line-Protocol. In Zeile 68 wird der Datenbankname, Username, Passwort der Datenbank und die Größe des Bodys der Anfrage in den Header-Teil der HTTP Anfrage geschrieben. In Zeile 64 wird der Body-Teil der Anfrage im InfluxDB-Line-Protocol Format erzeugt, dieser enthält den Bezeichner des Messwerts (z.B. X oder Y), die I2C Adresse des jeweiligen Sensors von dem die Messdaten stammen, den Messwert selbst sowie den Zeitstempel der Messung in ms. In Zeile 85 wird der HTTP-Request bestehend aus Header und Body an den Server übertragen. Die Antwort vom Server wird gespeichert und anschließen auf die erwartete Rückmeldung **HTTP/1.1 204 No Content** untersucht, wenn der Schreibversuch fehlschlägt wird eine Meldung im Terminal angezeigt. Außerdem geht die Power-LED auf dem Status & Adapter-board aus.

## 8.4 main

Die main-Funktion führt den Benutzer durch den Auswahlprozess der Einstellungen (Integrationszeit, Verstärkungsfaktor und Intervall der Messpunkte) und startet dann die Messung.

### 8.4.1 default\_values.h

Alternativ können die Einstellungen in der Datei default\_values.h geändert werden. Hier ist es außerdem möglich, den zu verwendenden Sensortyp auszuwählen und die benutzerfreundliche Menüführung auszuschalten, um den Messaufbau im Plug-and-Play-Modus zu betreiben.

---

```
1 #define DEFAULT_INTEGRATION_VALUE 20 // [0:255] Integration ←
    Value * 2,8ms = Integration Time
2 #define DEFAULT_GAIN 4 // Set Gain [0:3] 0->1x 1->3.7x 2->16x ←
    3->64x 4->Auto Gain:
3 #define DEFAULT_Mesuremnt_Interval 1 // Set Measurement ←
    Interval in Minutes [1:65535]
4 #define USE_AS7261 1 // 1 = use AS7261 0 = ignore AS7261
5 #define USE_AS7265X 1 // 1 = use AS7265X 0 = ignore AS7265X
6 #define PLUG_AND_PLAY 0 // 1 = Start Measurement instantly 0 = ←
    Show Settings -Menu
```

---

## 8.5 measurement

Die Datei measurement.c enthält alle Funktionen, welche mithilfe der die beiden Librarys (influxDB Library 8.3, AS726X Library 8.2), Messungen mit den zuvor definierten Einstellungen durchführt und in der Datenbank zu speichert.

### 8.5.1 fixedGainMeasurementAS7261 & fixedGainMeasurementAS7265X

Bei der Messung mit konstanter Verstärkung werden alle Werte mit einer festen Integrationszeit und einem festen Gain erfasst. Es gibt also für jeden Kanal einen Wert, der unter dem gleichen Namen gespeichert wird. In der Datenbank sind sie unter den in Tabelle 4 und 5 beschriebenen Namen zu finden.

Fixed Gain
X
Y
Z
Clear
Dark
NIR

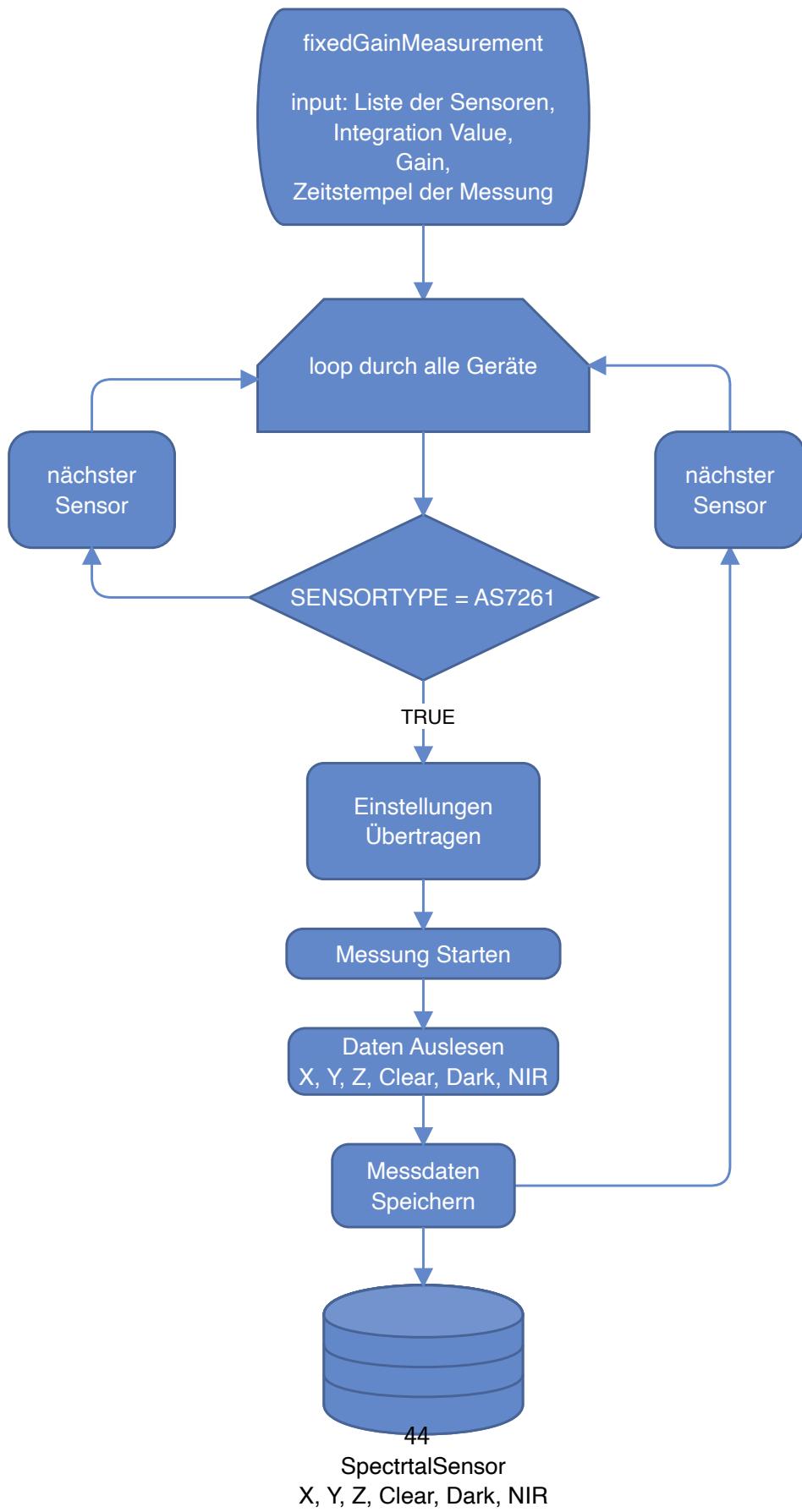
Tabelle 4: AS7261 Datenbank-ID bei festem Verstärkungsfaktor

Fixed Gain
R
S
T
U
V
W
G
X
I
J
K
L
A
B
C
D
E
F

Tabelle 5: AS7265X Datenbank-ID bei festem Verstärkungsfaktor

Das Flussdiagramm in Abbildung 23 beschreibt den Ablauf für fixedGainMeasurementAS7261, der Ablauf von fixedGainMeasurementAS7265X ist bis auf den SENSORTYP und die Namen der Channel identisch.

Abbildung 23: Flowchart fixedGainMeasurementAS7261



### 8.5.2 autoGainMeasurementAS7261 & autoGainMeasurementAS7265X

Bei der Auto-Gain-Messung wird für jeden der 4 möglichen Verstärkungsfaktoren eine Messung mit fester Integrationszeit durchgeführt. Die Messergebnisse werden in der Datenbank unter den in Tabelle 6 und 7 beschriebenen Namen gespeichert. Darüber hinaus wird bei jeder einzelnen Messung für jeden Channel individuell der Messwert ausgewählt, welcher das 16 Bit Register am besten aussteuert. Dieser Wert wird mit der Funktion `matchValueToMaxGain` (Abschnitt 8.5.3) an die maximale Verstärkung angepasst um einen sprungfreien Output-Plot zu ermöglichen (Datenbank-ID AutoGain). Zuletzt wird der verarbeitete Wert und der ursprünglich verwendete Gain in der Datenbank gespeichert.

Gain 0	Gain 1	Gain 2	Gain 3	AutoGain	UsedGain
X*0	X*1	X*2	X*3	X	gainX
Y*0	Y*1	Y*2	Y*3	Y	gainY
Z*0	Z*1	Z*2	Z*3	Z	gainZ
Clear*0	Clear*1	Clear*2	Clear*3	Clear	gainClear
Dark*0	Dark*1	Dark*2	Dark*3	Dark	gainDark
NIR*0	NIR*1	NIR*2	NIR*3	NIR	gainNIR

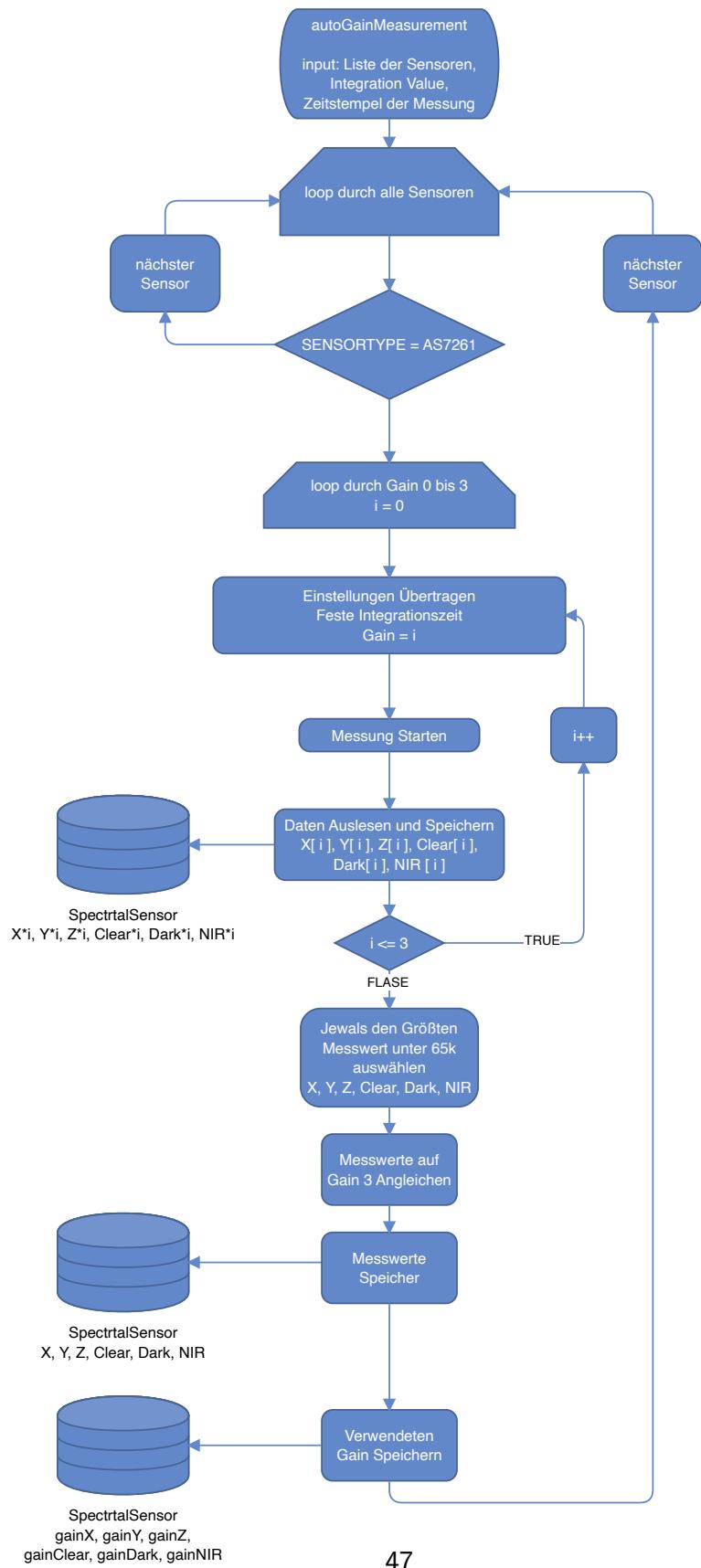
Tabelle 6: AS7261 Datenbank-ID bei automatischem Verstärkungsfaktor

Gain 0	Gain 1	Gain 2	Gain 3	AutoGain	UsedGain
A*0	A*1	A*2	A*3	A	gainA
B*0	B*1	B*2	B*3	B	gainB
C*0	C*1	C*2	C*3	C	gainC
D*0	D*1	D*2	D*3	D	gainD
E*0	E*1	E*2	E*3	E	gainE
F*0	F*1	F*2	F*3	F	gainF
G*0	G*1	G*2	G*3	G	gainG
H*0	H*1	H*2	H*3	H	gainH
R*0	R*1	R*2	R*3	R	gainR
I*0	I*1	I*2	I*3	I	gainI
S*0	S*1	S*2	S*3	S	gainS
J*0	J*1	J*2	J*3	J	gainJ
T*0	T*1	T*2	T*3	T	gainT
U*0	U*1	U*2	U*3	U	gainU
V*0	V*1	V*2	V*3	V	gainV
K*0	K*1	K*2	K*3	K	gainK
W*0	W*1	W*2	W*3	W	gainW
L*0	L*1	L*2	L*3	L	gainL

Tabelle 7: AS7265X Datenbank-ID bei automatischem Verstärkungsfaktor

Das Flussdiagramm in Abbildung 24 beschreibt den Ablauf für autoGainMeasurementAS7261, der Ablauf von autoGainMeasurementAS7265X ist bis auf den SENSORTYP und die Namen der Channel identisch.

Abbildung 24: Flowchart autoGainMeasurementAS7261



### **8.5.3 `matchValueToMaxGain`**

Die Funktion `matchValueToMaxGain` passt einen übergebenen Messwert mit dem verwendeten Verstärkungsfaktor von 1x, 3,7x, 16x, 64x, an die maximale Verstärkung von 64x an:

- $\text{Gain1}x \cdot 64 = 64$
- $\text{Gain3,7}x \cdot 17,2972972973 \approx 64$
- $\text{Gain16}x \cdot 4 = 64$
- $\text{Gain64}x \cdot 1 = 64$

## **9 Benutzerhandbuch**

### **9.1 Hardware Setup**

Der Raspberry Pi muss mit seiner Stromversorgung (USB C), Einem Netzwerk (Ethernet) und 1-10 Sensorboards verbunden werden.

### **9.2 Messung Starten**

Über das Terminal (Unter Windows cmd) muss eine SSH Verbindung zum Raspberry Pi aufgebaut werden.

**`ssh pi@[IP-ADRESSE]`**

Die Messsoftware wurde automatisch in einem Screen mit den Namen `SpectralSensor` gestartet, um die Messung zu starten muss sich mit diesem Screen verbunden werden.

**`screen -r`**

Abbildung 25: Terminal Output bei Start der Messsoftware

Zuerst muss im Terminaloutput überprüft werden ob der NTP Server synchronisiert ist. Wird eine Falsche Systemzeit angezeigt muss das Programm nach etwa 40 Sekunden Neugestartet werden, anschließend solle die Systemzeit richtig sein. (Mehr zum Programm Neustart: 9.4). Im I2C Scann sollten alle angeschlossenen Sensoren angezeigt werden. Die Messung wird mit den angezeigten Einstellungen gestartet, indem die Eingabeaufforderung mit **Y** bestätigt wird.

```
-----I2C Scan -----
Device at: 0x4B is AS7261
Please check if all expected devices are available.
-----Settings-----
Integration Value: 20 * 2.8ms = Integration Time
Gain: 4
Mesurement Intervall: 1 min
-----
Are The Settings Correct? Type y to continue, n to change Settings
y
--Starting Measurment Cycle--
0x4B Changed Gain to 0
0x4B getX: 37
0x4B getY: 76
0x4B getZ: 7
0x4B getClear: 719
0x4B getDark: 0
0x4B getNIR: 208
0x4B Changed Gain to 1
0x4B getX: 135
0x4B getY: 279
0x4B getZ: 28
0x4B getClear: 2645
0x4B getDark: 0
0x4B getNIR: 764
0x4B Changed Gain to 2
0x4B getX: 592
0x4B getY: 1228
0x4B getZ: 119
0x4B getClear: 11634
0x4B getDark: 2
0x4B getNIR: 3356
0x4B Changed Gain to 3
0x4B getX: 2186
0x4B getY: 4541
0x4B getZ: 433
0x4B getClear: 65535
0x4B getDark: 6
0x4B getNIR: 12150
X used gain: 3 matched value to gain 3: 2186
Y used gain: 3 matched value to gain 3: 4541
Z used gain: 3 matched value to gain 3: 433
Clear used gain: 2 matched value to gain 3: 46536
Dark used gain: 3 matched value to gain 3: 6
NIR used gain: 3 matched value to gain 3: 12150
0x4B saveX: 2186
0x4B saveY: 4541
0x4B saveZ: 433
0x4B getClear: 46536
0x4B getDark: 6
0x4B getNIR: 12150
Measurement duration: 1220 ms
-----
Next Mesurement: Mon Oct 12 00:07:00 2020
```

Abbildung 26: Terminal Output bei Start der Messung

Um die Einstellung zu verändern wird **N** ausgewählt.

```
-----  
Are The Settings Correct? Type y to continue, n to change Settings  
n  
  
Set Integration Value [0:255]:  
20  
Set Gain [0:3] 0->1x 1->3.7x 2->16x 3->64x 4->Auto Gain:  
4  
Set Measurement Interval in Minutes [1:65535]  
1
```

Abbildung 27: Beispielhafter Terminal Output bei Einstellungsänderungen

### 9.3 Webinterface

Das Grafana Webinterface ist unter folgender Adresse im Browser zu erreichen:  
[IP-ADRESSE]:3000

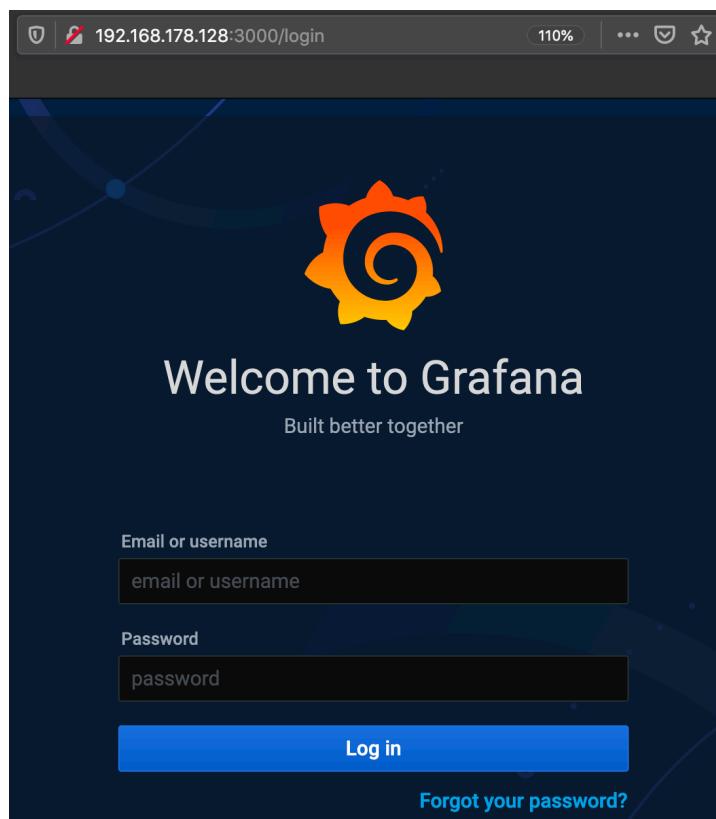


Abbildung 28: Grafana Login im Browser

Im Example Dashboard sind alle Sensordatenplots vorhanden, wenn weniger Plots benötigt

werden oder die Darstellung angepasst werden muss kann eine Kopie angefertigt und bearbeitet werden.

Um Daten im CSV-Format zu exportieren muss zuerst der gewünschte Zeitbereich ausgewählt werden. Anschließend wird wie in Abbildung 29 zu sehen im Kontextmenü (Klick auf den Namen eines Plots) unter Inspect die Option Data ausgewählt.



Abbildung 29: Grafana Login im Browser

Daten können nur pro Plot Exportiert werden um mehr Daten auf einmal zu exportieren müssen alle gewünschten Daten in einem Plot vereint werden.

## 9.4 Messsoftware Neustarten

Das Programm kann mit Control+C beendet werden.

Um es erneut zu starten sollte es neu kompiliert werden, da so immer alle eventuellen Einstellungsänderungen übernommen werden:

**sudo ./SpectralSensor make** TODO: pfad richtig?

## 9.5 IP Adress Scan

**Unter Unix:** sudo nmap -sn 192.168.1.0/24

**Unter Windows:** Mit Hilfe der Software "PortScan"  
<https://www.the-sz.com/products/portscan/>

## 9.6 Bedeutung der Status LEDs

### 9.6.1 Rapsberry Pi

Die rote PWR-LED leuchtet kontinuierlich bei stabiler 5 V Stromversorgung.  
Die grüne ACT-LED blinkt, wenn die SD-Karte korrekt arbeitet.

### 9.6.2 Status & Adapterboard

Die rote Power-LED zeigt an, dass die Messsoftware gestartet wurde.  
Die grüne Heartbeat-LED ändert ihren Status nach jedem Messzyklus.

## 9.7 Hilfe

### Nachdem starten der Messung gibt es keine ausgabe in Terminal:

Vermutlich sind keine Sensoren angeschlossen

### Die Systemzeit ist Falsch:

Keiner der NTP-Server aus der Liste TODO kann erreicht werden.

### Das Webinterface ist nicht Erreichbar:

Entweder wurde die Adresse falsch geschrieben oder der Grafana-Server ist abgestürzt / wird nicht mehr automatisch gestartet.

**TODO was passiert wenn InfluxDB aus ist?** Der InfluxDB-Server abgestürzt / wird nicht mehr automatisch gestartet.

## 9.8 Liste Der Verwendeten I2C Adressen & Translationbytes

Die Translation Bytes (Dezimal) 0- 59 Wurde bereits verwendet, sollten weitere Sensorboards angefertigt werden muss das hier notiert werden:

## 10 Messungen

Die Abbildungen 30-33 zeigen die gleiche Messung eines Wolkenfreien Dezembertags am 12.12.2020 in Berlin mit unterschiedlichen Verstärkungsfaktoren der Integrationswert ist auf 255 gesetzt. Es ist zu erkennen, dass die Auflösung der Y-Achse mit zunehmender Verstärkung steigt. Aus Abbildung 3 ist jedoch ersichtlich, dass der maximale Messwert von 65.000 mit einem hohen Verstärkungsfaktor erreicht werden kann und die Daten so ihre Aussagekraft verlieren (clipping).

Abbildung 34 zeigt die gleiche Messung im AutoGain-Modus, hier werden die Messungen mit den verschiedenen Verstärkungsfaktoren zusammengefasst und in einem Plot mit maximaler Auflösung, aber ohne Clipping dargestellt. Durch Auswahl der Integrationszeit und der Durchlässigkeit der Streuscheibe muss sichergestellt werden, dass bei Gain 0 kein Clipping auftritt.

Die Ausgabe-Daten der Messung sind nicht kalibriert. Außerdem haben sie keine Einheit bei festem Gain liegen sie zwischen 0 und 65000. Im Autogain-Modus liegen sie zwischen 0 und 4160000. Um die Daten nutzbar zu machen, müssen sie mit Hilfe eines anderen Messgerätes auf allgemein verständliche Candela-Werte kalibriert werden.

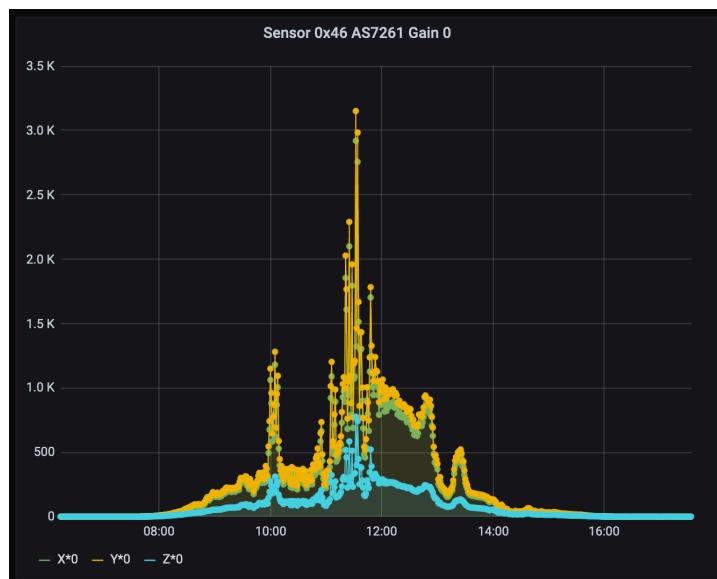


Abbildung 30: Gain 0 (1x)

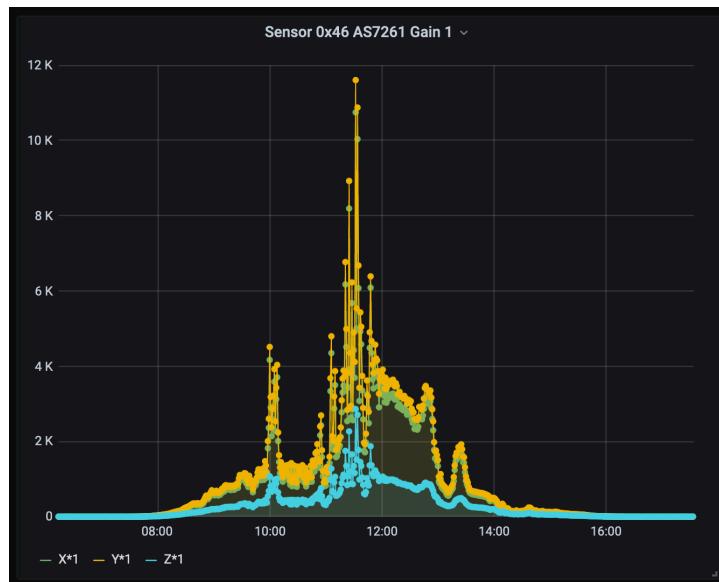


Abbildung 31: Gain 1 (3,7x)

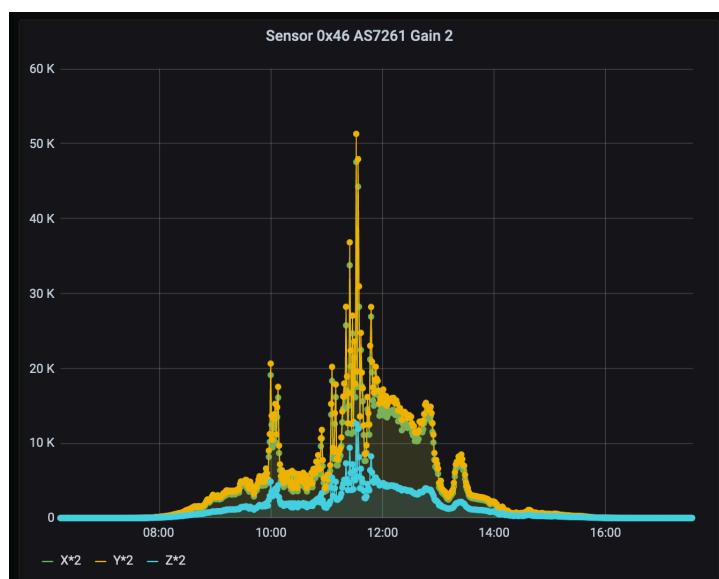


Abbildung 32: Gain 2 (16x)

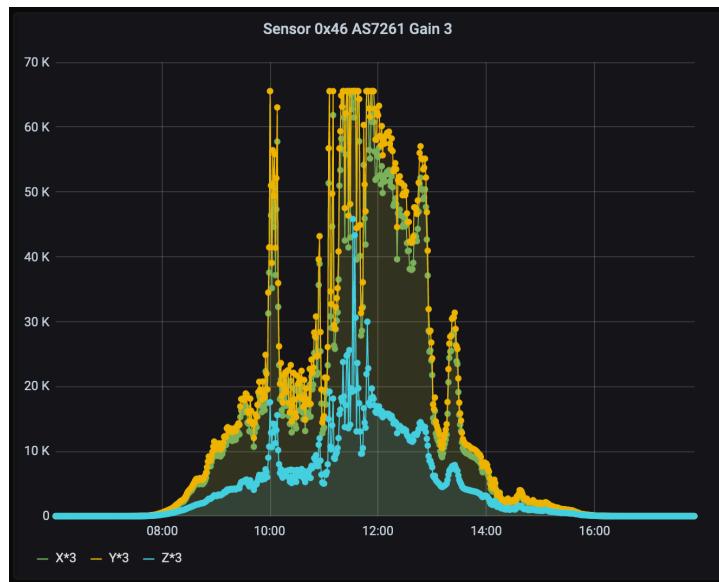


Abbildung 33: Gain 3 (64x)

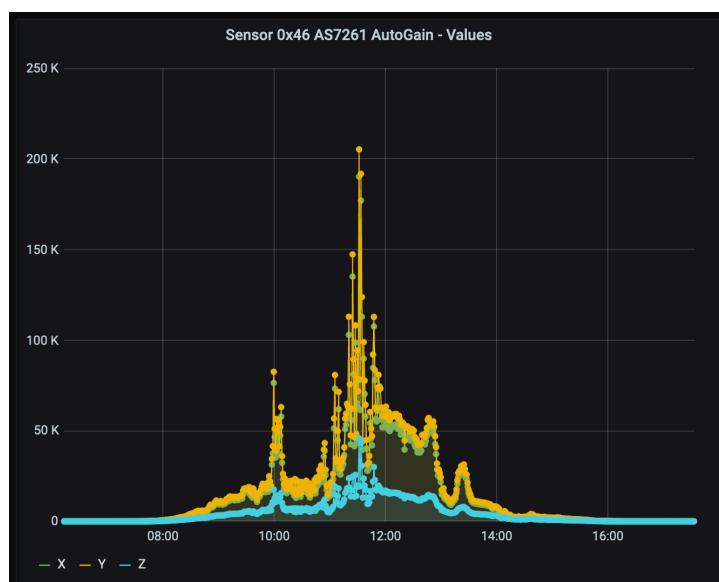


Abbildung 34: AutoGain

# 11 Zusammenfassung

## Literatur

- [1] L. Stiny, *Aktive elektronische Bauelemente*. Seite 168.
- [2] H. Göbel, *Bauelemente der Halbleiterelektronik*. Seite 184.
- [3] “osioptoelectronics.” <http://www.osioptoelectronics.com>.
- [4] “i2b-bus.org.” [www.i2b-bus.org](http://www.i2b-bus.org).
- [5] “nxp.com.” [www.nxp.com/docs/en/user-guide/UM10204.pdf](http://www.nxp.com/docs/en/user-guide/UM10204.pdf).
- [6] “Datenblatt as7265x.” [ams.com/as7265Xtab/documents](http://ams.com/as7265Xtab/documents).
- [7] “Datenblatt as7261.” [ams.com/as7261tab/documents](http://ams.com/as7261tab/documents).
- [8] “Pi<sub>4</sub>topview.” [projects.raspberrypi.org/en/projects/raspberry-pi-setting-up/1](http://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up/1).
- [9] “Datenblatt ltc4316.” <https://www.analog.com/en/products/ltc4316.htmlproduct-documentation>.