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

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

Bachlorarbeit

Vorgelegt von: Lennard Bödiger
Matrikelnr.: 363470
Studiengang: Technische Informatik

18. September 2020

1 Einleitung

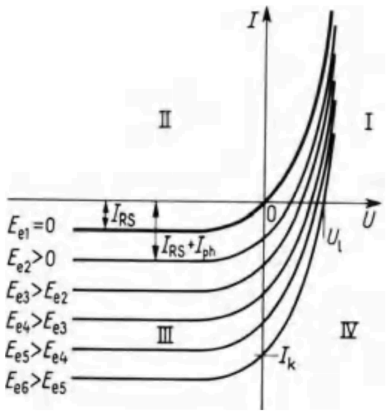
2 Technische Grundlagen

2.1 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 (Abb. 1a). Allerdings hängt der Kurzschlussstrom auch von der Eindringtiefe in das Silizium Substrat ab, welche wiederum von der Wellenlänge abhängt (Abb 1b). Außerdem ändert sich der sich der Absorptionskoeffizient von Silizium mit der Temperatur.

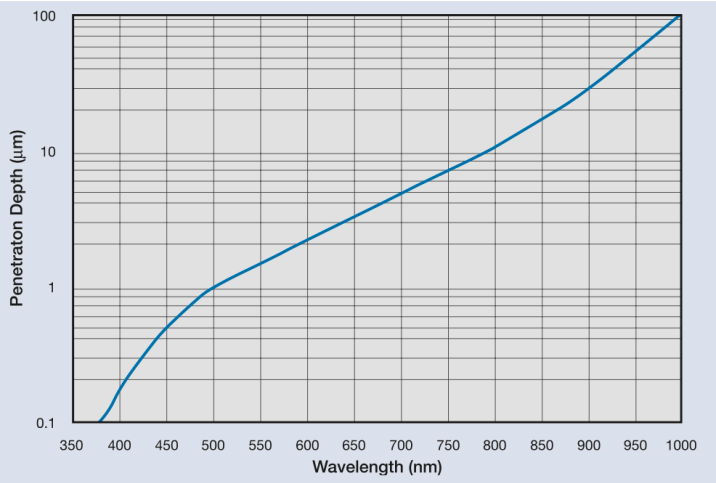
Ich habe die Sensoren mit Luft gelötet, aber da dieses QFN Package einfach ein Pain zum Löten ist, brauche ich immer mal wider mehrere versuche, und früher oder später läuft mir Flussmittel in den Sensor/ ich brenne ihn an weil ich zu lange auf einer Stelle bleibe. Das sind also alles Probleme, die mit etwas Kapton Tape auf der Sensor Öffnung und mehr Übung zu lösen sind, aber es wäre super mal ein sauberes referenboard zu haben.

(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: osioptoelectronics.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 Tageslichtmessungen, dass einfallende Licht auf eine möglichst begrenzten Wellenlängenbereich zu beschränken, da so ein möglichst akkurater Temperatur und Frequenzabhäniger kompensations faktor gewählt werden kann.

2.2 I2C

I2C ist ein siples und effizentes Busprotokoll. Es wurde ursprünglich von phillips entwikelt, wird aber seit einigen jahren von NPX weiterentwikelt. In seiner simpelsten form ermöglicht es einen Master mit bis zu 128 Slave geräten zu verbinden. Dafür werden nur 2 leitungen benötigt, die die SCL und SDA genannt werden. SCL ist die Taktleitung. Sie wird verwendet, um alle Datenübertragungen über den I2C-Bus zu synchronisieren. SDA ist die Datenleitung. Außerdem müssen alle Bus teilnehmer mit dem gleichen GND potential verbunden sein um stromfluss über SDA und SCL leitungen zu ermöglichen.

Da SCL und SDA als "open drain" betrieben werden, was bedeutet das die Bus teilnehmer den output low aber nicht high setzen können, muss ein Pull up Wiederstand zur versorgungspannung verwendet werden.

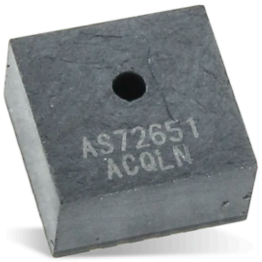
Die Clock leitung SCL wird nur vom Bus Master gesteuert. Die SDA leitung wird vom Master und Slave genutzt allerdings antworten die Slaves im normalbetrieb nur nachdem sie vom master auf iherer Adresse eine einfage erhalten haben. Die Spezifikation des Protokolls empfiehlt die SDA und SCL Leitung möglichst weit voneinander zu entfernen um so die Signalqualität zu verbessern.

3 Hardware Komponenten

3.1 Sensoren

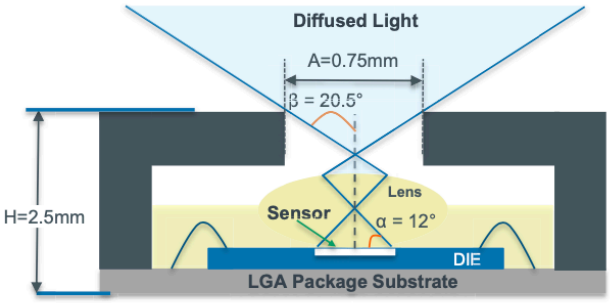
Die Sensoren aus der AS726X Reihe sind in der lage Licht, also elektromagnetische Strahlung zu messen. In jedem Sensor sind 6 Photodioden verbaut. Vor jeder Photodiode ist ein Silizium-Interferenzfilter montiert, welcher wie ein Bandpassfilter arbeitet, er ist nur für einen bestimmten Ausschnitt des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

Abbildung 2: AS726X



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

Abbildung 3: Seitenasicht AS726X



Quelle: Datenblatt AS7261

Wie in Abb:18 dargestellt, fällt das Licht durch die Öffnung in der Mitte des Sensors ein, eine intern verbaute Linse verteilt das Lich auf die Interferenzfilter. Die Genauigkeit der Filter verändert sich je nach Einstrahlwinkel, daher ist es wichtig, vor den Sensor noch eine Streuscheibe zu montieren.

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.

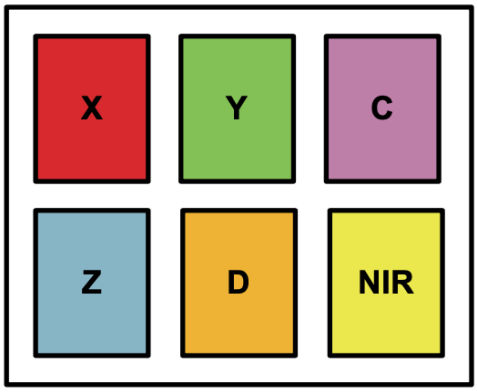
Alle Sensoren erhalten vom Hersteller dieselbe nicht veränderbare I2C Adresse: 0x49. Daher muss ein I2C Translator genutzt werden, welcher es ermöglicht, mehrere Sensoren im gleichen Bussystem zu adressieren(siehe Abschnitt 3.3).

Die Modelle unterscheiden sich durch die verbauten Silizium-Interferenzfilter, also die unterscheidbaren Wellenlängenbereiche sowie in der benötigten Peripherie. Die grundsätzliche Messmethode ist aber immer gleich. Im Folgenden werden die verwendeten Sensoren AS7261(3.1.1) sowie der AS7265X(3.1.2) beschrieben.

3.1.1 AS7261

Das Sensorarray des AS7261 unterscheidet zwischen X,Y,Z,C,D und NIR.

Abbildung 4: AS7261-Sensor Array



Quelle: Datenblatt AS7261

Die Ergebnisse des ADC werden direkt, ohne das der Temperaturdrift der Photodioden berücksichtigt wird in die in Tabelle 1 beschrieben Register geschrieben.

Tabelle 1: Your caption.

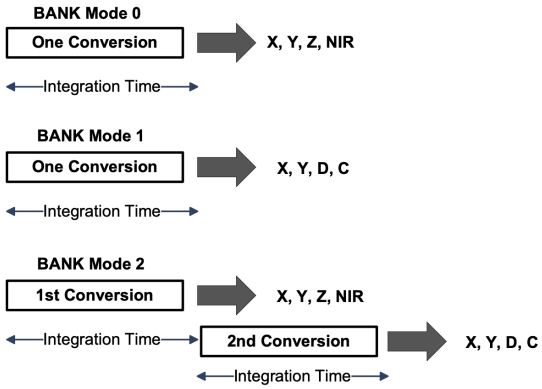
kürzel	Bezeichnung	Register	Filtereigenschaften
C	Clear	0x12-0x13	kein Filter
D	Dark	0x10-0x11	Lichtdurchlässigkeit des Gehäuses
NIR	nahes Infrarot	0x0E-0x0F	?????
X	X (CIE 1931)	0x08-0x09	????
Y	X (CIE 1931)	0x0A-0x0B	????
Z	X (CIE 1931)	0x0C-0x0D	?????

Da die Messergebnisse der unterschiedlichen Wellenlängenbereiche den in **??** genannten Verzerrungen unterliegen, werden auch noch berichtigte X,Y,Z-Werte bereitgestellt.

kürzel	Register	Berichtigungs Formel
Cal-X	0x14:0x17	????
Cal-Y	0x18:0x1B	????
Cal-Z	00x1C:0x1F	?????

Es gibt 3 sogenannte Bank Modes in denen der Sensor Arbeiten Kann.

Abbildung 5: AS7261-Bank Modes



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 dauch die kalibrierten, korrigierten Werte auch aus den entsprechenden I2C-Registern abgerufen werden.

Bank Mode 3

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 auch aus den entsprechenden I2C-Registern können abgerufen werden. Das DATA RDY-Bit wird auf 1 gesetzt, sobald Daten verfügbar sind.

Für diesen Anwendungsfall 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 NanoPi übertragen werden.

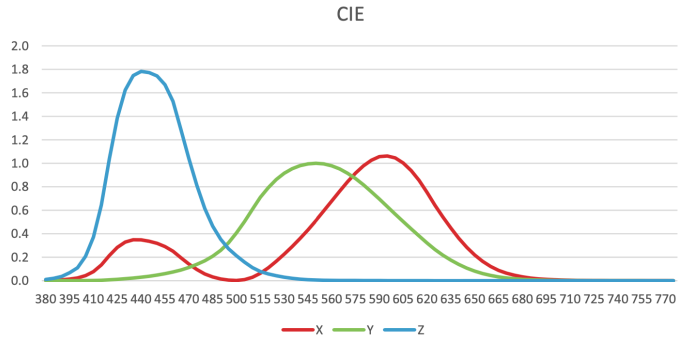


Abbildung 6: AS7261-Sensor Array

3.1.2 AS7265X

AS7265X beschreibt AS72651, AS72652 und AS72653 wobei der AS72651 als master für AS72652 und AS72653 fungiert indem er über einen weiteren separaten I2C Bus ihre Daten abfragt und ansonsten wie der AS7261 arbeitet.

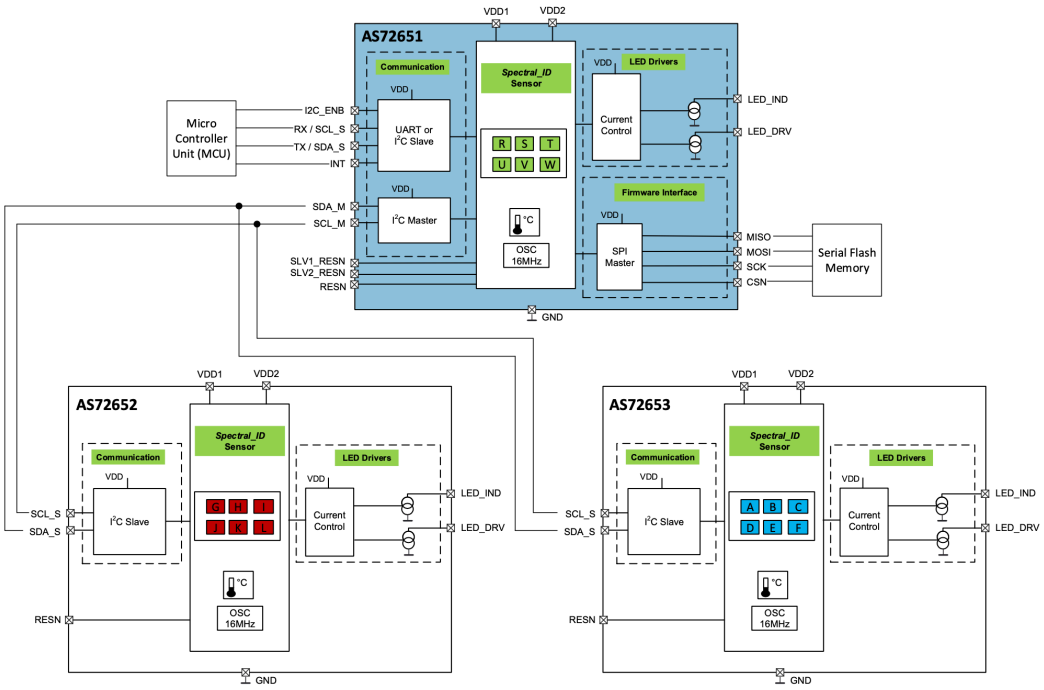


Abbildung 7: S7265X-Scematic

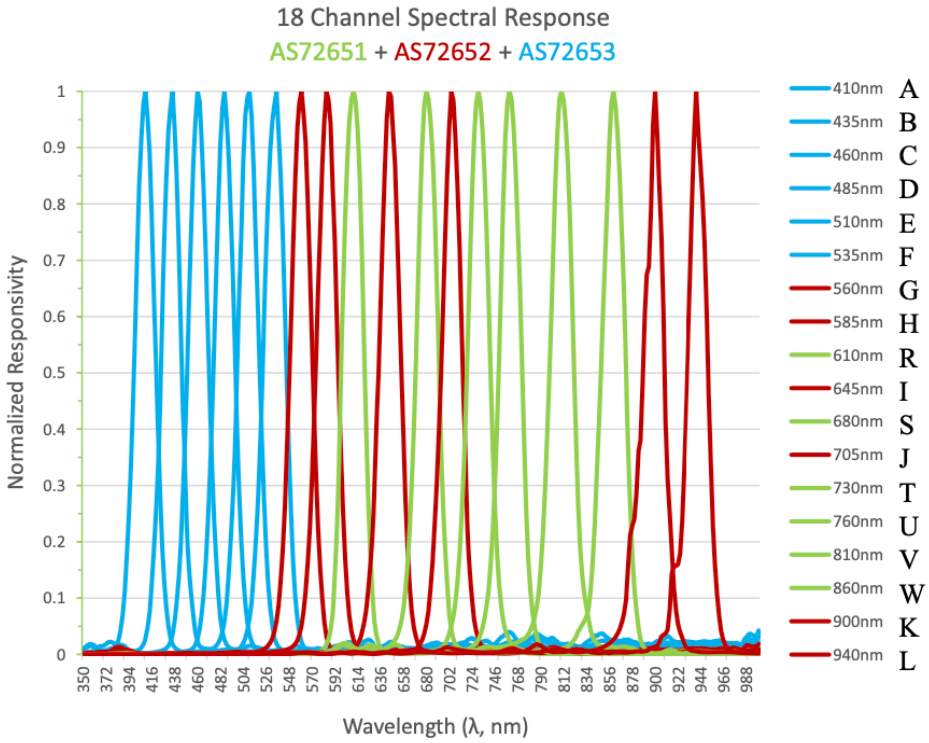


Abbildung 8: AS7261-Spectral Responsivity

Die drei Sensoren messen in Kombination mit 18 unterschiedliche Photodioden, so können 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 Abb: TODO zu sehen mit den Buchstaben A-L gekennzeichnet.

3.2 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 NanoPi NEO2 Black erfüllt alle diese Anforderungen:

Abmessungen	4 cm x 4 cm
Speicher	eMMC Flash Module Socket
Anschlüsse	10/100/1000M Ethernet
GPIO	UART, I2C, IO
RAM	1 GB
CPU	Allwinner H5 Quad-Core Cortex-A53
Preis	TODO

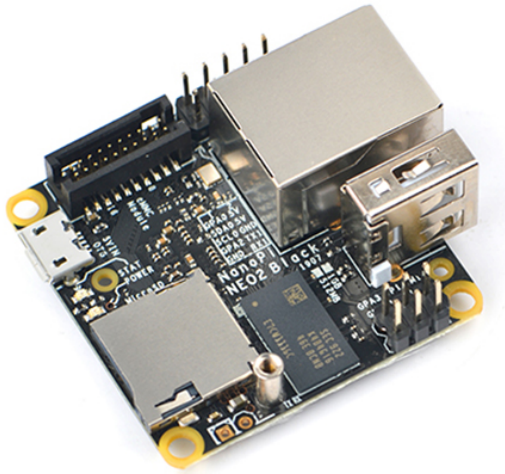
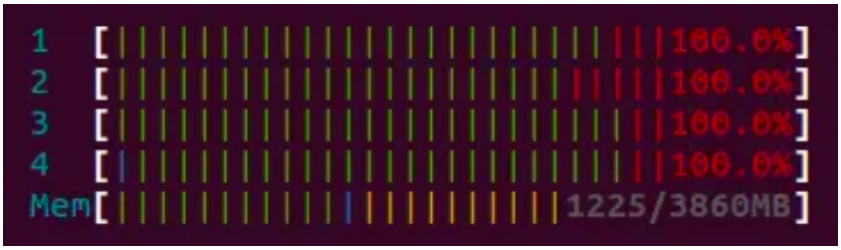


Abbildung 9: NanoPi NEO2 Black

An den eMMc Socket kann bis zu 128 GB Flash speicher angeschlossen werden. TODO warum reichen 32 GB.
Da eine kabelgebundene Lösung mehr Zuverlässigkeit bietet, wird eine Netzwerkverbindung über Ethernet einer WLAN-Schnittstelle vorgezogen. Es besteht aber die Möglichkeit, einen externen USB WLAN-Adapter nachzurüsten.
Die I2C Schnittstelle des NanoPi arbeitet mit 3.3V allerdings wird nur ein 5V und kein 3,3V Output bereitgestellt, daher muss ein steppdown Wandler verwendet werden, um die Sensorboards mit Strom zu versorgen.
Die CPU ist für den Anwendungsfall weitaus ausreichend dimensioniert. In Abbildung 10 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

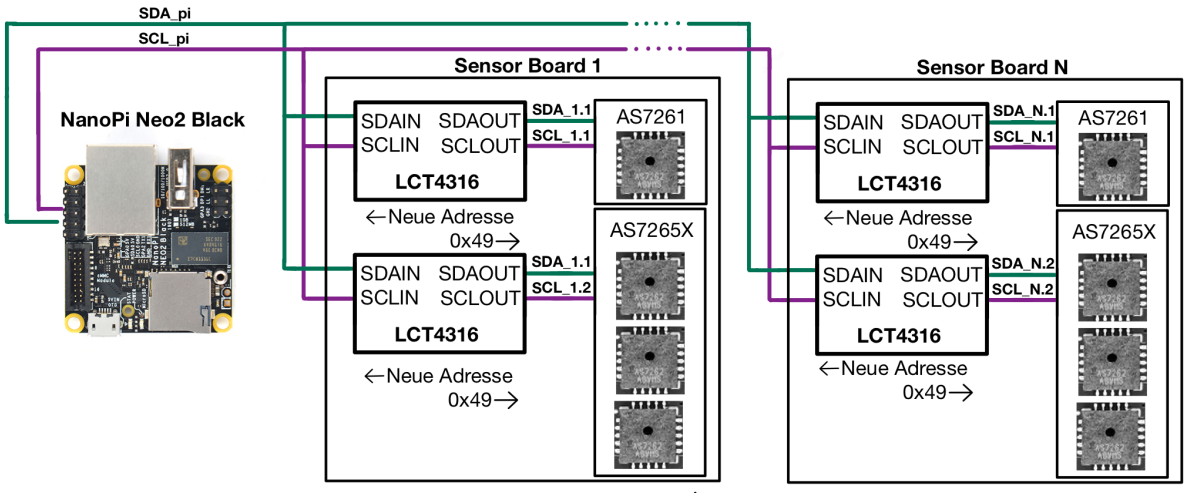
Abbildung 10: CPU-Performance



3.3 I2C Address Translator LTC4316

Da, wie in Abschnitt 3.1 genannt alle Sensoren unter derselben I2C Addresse erreichbar sind, wird ein I2C Translator genutzt, um für eine individuelle Adressierung zu sorgen.

Abbildung 11: I2C-Bus im Messaufbau



Wie in Abb 11 zu sehen wird für jeden Sensor ein LTC4316 an die Busschnittstelle des NanoPi angeschlossen.(SDAIN, SCLIN).

An jeden LTC4316 wird ein AS7261 oder AS72651 angeschlossen.(SDAOUT, SCLOUT) Bei Kommunikation vom Nano-Pi zum Sensor wird dann die I2C Adresse mit einem Faktor (Translation Byte) welcher mit diskreten Widerständen eingestellt wird mit Fomel 1 verrechnet, um so die Adresse anzupassen.(XORH,XORL). Um das Translation Byte einzustellen müssen die Widerstände bla,bla und bla wie in Abb **??** am LTC4316 angeschlossen werden.

$$SensorAdresse \oplus TranslationByte = NeueAdresse \tag{1}$$

Beispiel Rechnung

$0x49 \oplus 0x01 = 0x48$
$0x49 \oplus 0x02 = 0x4b$
$0x49 \oplus 0x05 = 0x4c$
$0x49 \oplus 0x06 = 0x4f$
$0x49 \oplus 0x0A = 0x43$
$0x49 \oplus 0x49 = 0x00$

Abbildung 12: Translation Byte

In Tabelle 2 und 3 sind die unterschiedlichen Konfigurationen des Translation Bytes aufgelistet.
Im Handbuch **??** ist eine Liste zu finde welche Translation Bytes bereits verwendet werden. Wenn weitere Sensor Boards angefertigt werden ist die Liste dort weiter zu pflegen.

Tabelle 2: Untere 4 Bit des Translation Byte

a3	a2	a1	a0	R_{LT}	R_{LB}
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

Tabelle 3: Obere 3 Bit des Translation Byte

a6	a5	a4	R_{LT}	R_{LB}
0	0	0	Open	Short
0	0	1	976	102
0	1	0	976	182
0	1	1	1000	280
1	0	0	1000	392
1	0	1	1000	523
1	1	0	1000	681
1	1	1	1000	887

3.4 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 Protokoll auf den den Flash Speicher übertragen. Der AT25SF041-SSHD-B wurde aus der von AMS bereitgestellten liste kompatiebler Flash speichern ausgewählt da er am günstigsten ist. Da die Firmware nur auf nachfrage bei AMS erhältlich. Für den Messaufbau werder folgende versionen verwendet die .bin files sind im anhang zu finden.
AS7261_complete **??**
AS726X_AS7265_complete_moonlight_v1 **??**

4 Platine

4.0.1 System Topologie

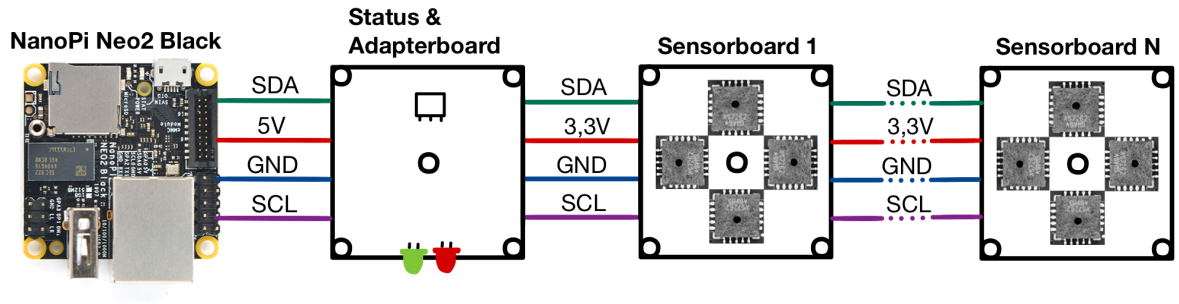


Abbildung 13: Verschaltung der der Platienen

Der Msssaufbau besteht aus einem NanoPi Neo2 Black welcher über eine status und Adapterplatinene mit 1-10 Sensnorboards verbunden werden kann.

4.0.2 Status & Adapterboard

Da der NanoPi nicht die benötigte $3,3V$ Stromversorgung bereitstellt wird eine Adapter-Shield mit einem Spannungswandler(LM3940IT-3.3) verwendet. Sollte der vom NanoPi bereitgestellte Strom nicht ausreichen kann eine externe Stromversorgung an connector j3 auf der adapterplatine angeschlossen werden. Auf dem Adapter-Shield findet der einheitliche Steckverbinder Platz, da das Adapter-Shield aufgrund seiner Bauform nicht falsch montiert werden kann so die Verpolung der Sensoren ausgeschlossen werden. Die Bedeutung der Status LED wird im Handbuch **??** erläutert.

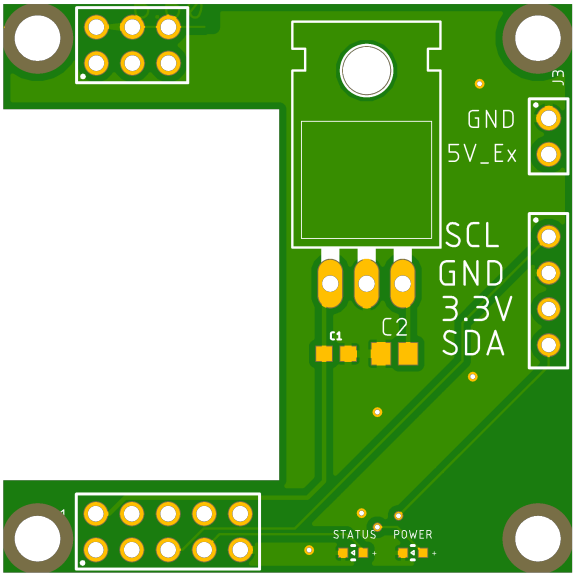


Abbildung 14: Verschaltung der der Platienen

4.0.3 Sensorboard

Die hauptaufgabe der Sensorboard Platine ist es den AS7261 und der AS7265X mit ihrem companion flash (**??**) und über den I2C Translator (**??**) dem I2C Bus verbunden. Außerdem werden verschiednene LED, Wiederstände und Kondensatoren verbaut.

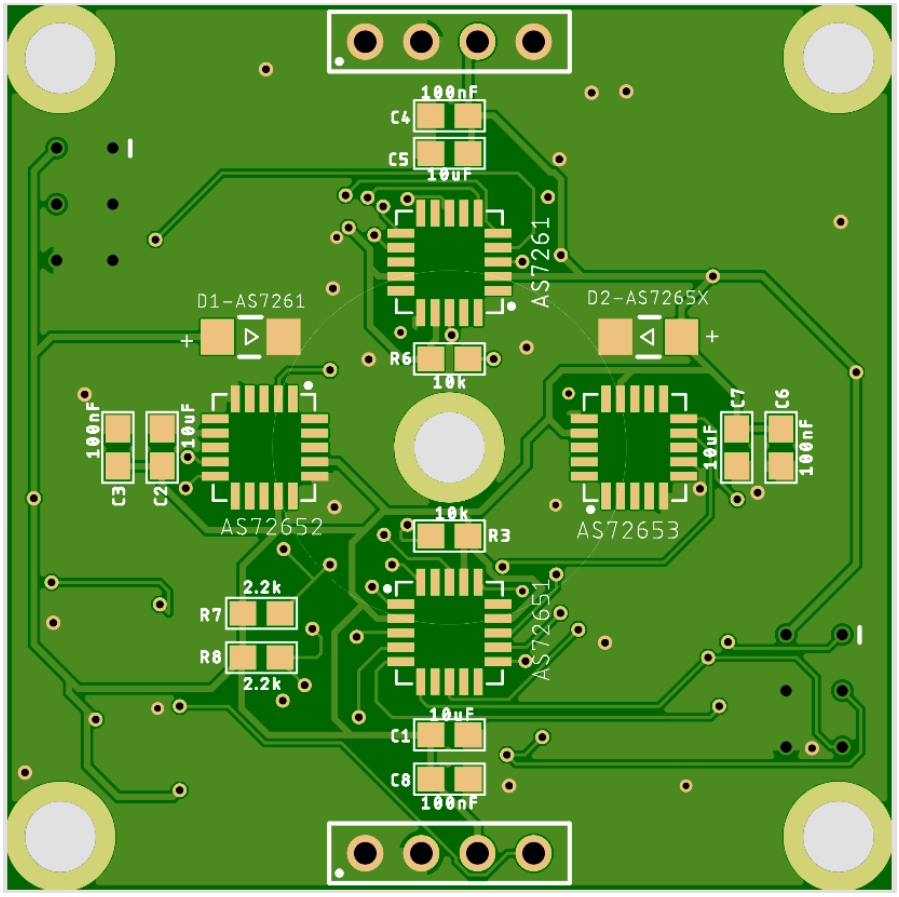


Abbildung 15: Verschaltung der der Platienen

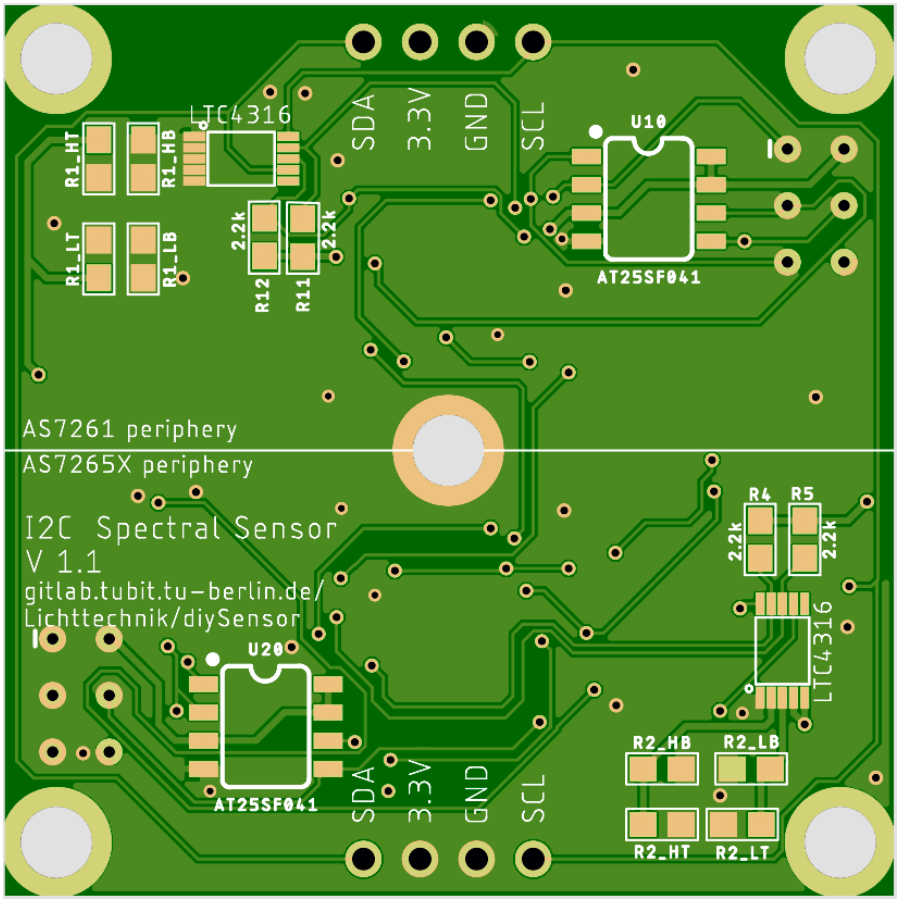


Abbildung 16: Verschaltung der der Platienen

Status LED: Am AS7261 und AS7265X Befindet sich jemals eine Rote Status LED wenn es ein problem mit dem Compainin Flash gibt fängt sie an zu blinken im normalbetrieb kann die LED Softwareseitg ein und ausgeschaltet werden. Wärend der Messubng solle 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 der AS7261 in den I2C modus gebracht. 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 **??** beschriebenen wiederstände welche das Translation Byte einstellen. Die wiederstände R1_XX bestimmen die Addresse des AS7261 und R1_XX die Addresse des AS72651.

Entstörkondensatoren: Das Datenblatt der Sensoren emphielt für jeden Sensor 2 Entstörkondensatoren möglichst nah am Sensor zwischen GND und VCC anzubringen.

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

I2C Lanes: Die I2C lanes haben ihren ursprung auf der Adapterplatinene und werden mithilfe der seitlichen connectoren über die Sensnsor boards durschgeschleift. Die maximal mögliche länge einer I2C Leitung hängt von der länge der Leitungskapazietät 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 eletromagitisches felder minimiert werden. Außerdem wurde darauf geachtet das die leitungen auf den platinen die gleiche länge habe da sich sonst die differenzen der Leitungs länge mit jeder angeschlossenen Platine addiert und es zu Timing differnezn zwischen der Daten und clock Leitung kommt. Der Verlauf der Leitungen ist in Abb**??** Rot gekennzeichnet.

Connector: An den Seiten der platiene befinden sich 4 durchkontaktierte Löcher, hier können unterschiedliche connectoren mit 2.54 mm pitch montiert werden. Es empfiehlt sich verpolungssichere connectoren zu verwenden, um hardware schäden vorzubeugen. Für dieses

Bauteil wurde keine SMD-Technik sonder durchkontaktierten Anschlussdrähten gewählt da so eine bessere mechanische Festigkeit erreicht wird.

Der Connector ist ein 2.54 mm pitch through hole socket. Er wird verwendet da es viele

5 Software

5.1 wiringPi

Der NanoPi Neo2 Black kommt mit einem vorinstallierten Fork von der wiringPi Libarry https://github.com/friendlyarm/WiringNP/blob/master/wiringPi/wiringPi.c Die funktionalität der drei verwendetetn C-Funktionen werden im folgenden beschrieben.

5.1.1 wiringPiI2CSetup

Der Funktion wird beim audfruf die I2C Adresse übergeben mit welcher eine verbindung aufgebaut werden soll: wiringPiI2CSetup(address).
Der Rückgabewert ist der Standard Linux File Disciptor oder -1, falls ein Fehler auftritt. Die Anzahl der File Desciptoren ist begrenzt daher muss die verbindung mit der Funktion close() aus der Library: unistd.h geschlossen werden wenn sie nicht mehr benötigt wird.

5.1.2 wiringPiI2CWriteReg8

Der Funktion wird bei aufruf der File Desciptor einer I2C verbindung sowie ein ziel register und die zu schreibenden 8bit Daten übergeben: wiringPiI2CWriteReg8 (fd, RegAdr, 8-Bit-Data). Wenn der Schreibzugriff vom I2C gerät bestätigt wurde wird eine 0 zurück gegeben.

5.1.3 wiringPiI2CReadReg8

Der Funktion wird bei aufruf der File Desciptor einer I2C verbindung sowie ein ziel register übergeben: wiringPiI2CReadReg8(fd, RegAdr). Der ausgelsene 8-Bit inhalt des Register ist der rüchgabewert. Wenn das Lesen Fehlschlägt bleibt das Programm in einer Endlosschleife hängen.

5.2 AS726X Libary

Die WiringPi_AS726X_Libary enthält alle funktionen um den AS7261 und AS7265X zu steuern und auszulesen. Da die Beispielimplementierungen im Datenblatt in vielen detaifragen ungenau und Fehlerhaft ist wurde die Arduino OpenSource Libarrys von sparkfun SparkFun_AS726X_Arduino_Library-master und SparkFun_AS7265x_Arduino_Library als implementierungs grundlage verwendet. Im ersten schritt der entwirklung wurde sie für die I2C schnittschelle WiringPi des NanoPi umgeschrieben und anschließen in ihrer Funktionalität erweitert um für das messystem mit mehreren sensoren auf dem gleichen bus nutzbar zu sein.

Im Folgenden Text werden die Register Adressen und mit den gleichen Namen wie im source code bezeichnet die Numerische Adressen sind in Tabelle **??** aufgelistet. TODO table

STATUS Register	I²C 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	I²C slave interface WRITE register Write-only	Register Address = 0x01 8-Bits of data written by the I²C Master intended for receipt by the I²C slave. Used for both virtual register addresses and write data.
READ Register	I²C slave interface READ register Read-only	Register Address = 0x02 8-Bits of data to be read by the I²C Master.

Abbildung 17: PysicalRegister

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

Abbildung 18: AS726x_CONTROL_SETUP

5.2.1 virtualWriteRegister

Wie bei Embeddet geräten üblich werden Einstellungen auf dem sensor verändern indem verschiedene sogenannte Special function registers mit daten beschrieben werden.
Jedes Special function register ist 8Bit groß und hat eine adresse und jedes Bit des Register representiert eine einstellung. Beispielsweise ist 0x07 das LED Control Register 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 Dezimal wert 0 beschrieben sind alle LED aus, wird es mit dem Dezimal wert 1 beschrieben leuchtet ist nur die status LED. Die Register Lassen sich aber nicht direkt Beschreiben, stattdesswen sind sie als sogennate Virtuelle Register Implementiert.

Der Sensor arbeitet mit virtuellenRegistern. Das heißt das nur Register 0x01 (WRITE Register) beschrieben werden kann. Um daten in eins der Special Funktions Register zu schreiben wird die funktion virtualWriteRegister verwendet. Die Funtionsweise lässt sich in 4 schritten zusammnefassen:

- Zeile TODO Warten 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 TODO Schreibe die Virtuelle Adresse in das WRITE Register und Setze zusätzlich Bit 8 des WRITE Register auf 1 um zu zeigen das es sich um einen Schreibenden zugriff auf das Virtuelle Register handelt.

- Warte erneut bis das WRITE Register leer ist.
- Schreibe die Daten in das WRITE Register

Der Sensor wird jetzt selber die übertragenen Daten aus dem WRITE Register angebene Virtuelle Register kopieren.

```
1
2 //Write to a virtual register in the AS726x
3 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 ↩
                  write now.
10        }
11        delay(POLLING_DELAY);
12    }
13    // Send the virtual register address (setting bit 7 to ↩
         indicate a  write a register).
14    wiringPiI2CWriteReg8(fd, AS72XX_SLAVE_WRITE_REG, ↩
         (virtualAddr | 0x80));
15
16    //Wait for WRITE register to be empty
17    while (1)
18    {
19        status = wiringPiI2CReadReg8(fd, ↩
             AS72XX_SLAVE_STATUS_REG);
20        if ((status & AS72XX_SLAVE_TX_VALID) == 0){
21            break; // No inbound TX pending at slave. Okay to ↩
                  write now.
22        }
23        delay(POLLING_DELAY);
24    }
25    // Send the data to complete the operation.
26    wiringPiI2CWriteReg8(fd, AS72XX_SLAVE_WRITE_REG, ↩
         dataToWrite);
27 }
```

5.2.2 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 Register Adressen möglich daten auszulesen. Die Funktionsweise der zum Daten auslesen benötigten funktion virtualReadRegister lässt sich wieder in 4 schritte aufteilen.

- Das AS72XX_READ_REG wird ausgelesen ohne das die daten verabeitet werden. Dieser schritt ist wie ein Reset des Registers zu verstehen.
- Zeile TODO Schreibe die Virtuelle Adresse in das WRITE Register und Setze zusätzlich Bit 8 des WRITE Register auf 0 um zu zeigen das es sich um einen Lesenden zugriff auf das Virtuelle Register handelt.
- Sobald das AS72XX_STATUS_REG den wert AS72XX_TX_VALID annimmt sind die Daten aus dem angebenen Virtuellen Register in das AS72XX_READ_REG kopiert worden.
- Lese 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 ↩
           to be read
7          /*uint8_t incoming = */wiringPiI2CReadReg8(fd, ↩
               AS72XX_SLAVE_READ_REG); //Read the uint8_t but do ↩
               nothing with it
8      }
9      //Wait for WRITE flag to clear
10     while (1) {
11         status = wiringPiI2CReadReg8(fd, ↩
               AS72XX_SLAVE_STATUS_REG);
12         if ((status & AS72XX_SLAVE_TX_VALID) == 0){
13             break; // If TX bit is clear, it is ok to write
14         }
15         delay(POLLING_DELAY);
16     }
17     // Send the virtual register address (bit 7 should be 0 to ↩
           indicate we are reading a register).
18     wiringPiI2CWriteReg8 (fd, AS72XX_SLAVE_WRITE_REG, ↩
```

```
        virtualAddr);
19
20      //Wait for READ flag to be set
21      while (1)
22      {
23          status = wiringPiI2CReadReg8(fd, ↩
                AS72XX_SLAVE_STATUS_REG);
24          if ((status & AS72XX_SLAVE_RX_VALID) != 0) break; // ↩
                Read data is ready.
25          delay(POLLING_DELAY);
26      }
27
28      uint8_t incoming = wiringPiI2CReadReg8(fd, ↩
            AS72XX_SLAVE_READ_REG);
29      return (incoming);
30  }
```

5.2.3 MeasurementFromAdress

Die Funktion baut einen I2C Verbindung zur übergebenen Bus-Adresse auf und ruft die Funktion takeMeasurments 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 Adrress
2  void MeasurementFromAdress(int address){
3      int fd =  wiringPiI2CSetup(address);
4      if (fd == -1) {
5          printf("i2c failed");
6      }
7      takeMeasurements(fd);  // takesMeasurmant Readings can now ↩
           be accessed via getX(), getY(), etc
8      close(fd);
9  }
```

5.2.4 takeMeasurements

Die Funktion takeMeasurements ruft die Funktion setMeasurementMode mit dem Parameter 3 auf das setzt den aus **??** bekannten Bankmode der übergebenen I2C Verbindung (fd) auf Bank Mode 3. Die On-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 ↩
          Mode 3
4
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 ↩
          parralel mesurments
10
11    //Readings can now be accessed via getViolet(), getBlue(), ↩
          etc
12 }
```

5.2.5 setMeasurementMode

Mit der Funktion setMeasurementMode werden die Bankmode Bits 2 und 3 des AS726x_CONTROL_SETUP Registers mit dem gewünschten wert für den Bankmode beschrieben.
Da die anderen Bits des Registers noch weitere einstellungen repräsentieren welche nicht verändert werden sollen, muss das register erst ausgelsen werden. Anschließend werden die Bankmode Bits auf 0 gesetzt um im nächsten schritt mit dem gewünschten neuen Bankmode wert beschrieben zu werden. Die Bedeutung der Bankmodes ist in **??** erleutert.

```
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, ↩
            fd); //Read
11      value &= 0b11110011; //Clear BANK bits
12      value |= (mode << 2); //Set BANK bits with user's choice
13      virtualWriteRegister(AS726x_CONTROL_SETUP, value, fd); ↩
            //Write
14  }
```

5.2.6 dataAvailable & clearDataAvailable

Das Bit dataAvailble im register todo wird vom Sensor auf 1 gesetzt wenn nach einer Messung neue daten vorhanden sind, Intterupts müssen dafür ausgeschaltet sein. dataAvailble wird auf 0 gesetzt wenn daten gelsen werden. Wenn eine One-Shot messung im Bankmode 3 durchgeführt wird sollte das dataAvailble bit mit der Funktion clearDataAvailable auf 0 gesetzt werden, da nicht sicher gestellt ist das die daten vor der Messung gelsen wurden. Es besteht die möglichkeit das sich das bit fälschicherweise noch im falsch positiven zustand befindet. Die Funktion dataAvailable gibt den wert des DataAvailable Bit zurück.

```
1  //Checks to see if DRDY flag is set in the control setup register
2  //TODO: was bool test retuned 2
3  uint8_t dataAvailable(int fd) {
4      uint8_t value = virtualReadRegister(AS726x_CONTROL_SETUP, ↩
           fd);
5      return (value & (1 << 1)); //Bit 1 is DATA_RDY
6  }
7
8  //Clears the DRDY flag
9  //Normally this should clear when data registers are read
10 void clearDataAvailable(int fd) {
11     uint8_t value = virtualReadRegister(AS726x_CONTROL_SETUP, ↩
           fd);
12     value &=  << 1); //Set the DATA_RDY bit
13     virtualWriteRegister(AS726x_CONTROL_SETUP, value, fd);
14 }
```

5.3 Rohwerte des AS7261 Auslesen

Die in **??** 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 Diskritor einer I2C verbindung mit einem AS7261 wird als übergabe parameter erwatet.
Um den Messert auszulesen wir die Funktion getchannel aufgerufen. Der Rückgabewer ist der 16-Bit Messwert aus dem jewaligen Register vom Datentyp Integer.

5.4 getChannel

Da die Messdaten 16-Bit groß sind, der Sensor aber nur über 8 Bit register verfügt werden 2 aufeinader folgende register ausgelsen und im Big-Andian Format aneinader geheftet. Die Funktion getChannel erwatet einen File Disciptor einer I2C verbindung zu einem Sensor und die adressen des High Bytes Raw Data Registers.

5.5 disableInterrupt

Die Funktion disableInterrupt setzt das INT Bit im register AS726x_CONTROL_SETUP auf 0 um Interrupts aus zu schalten. Da Der Interrupt auf der Sensor Paltiene nicht verbinden ist muss .. toto muss dass wirklich?