COOL-WEB

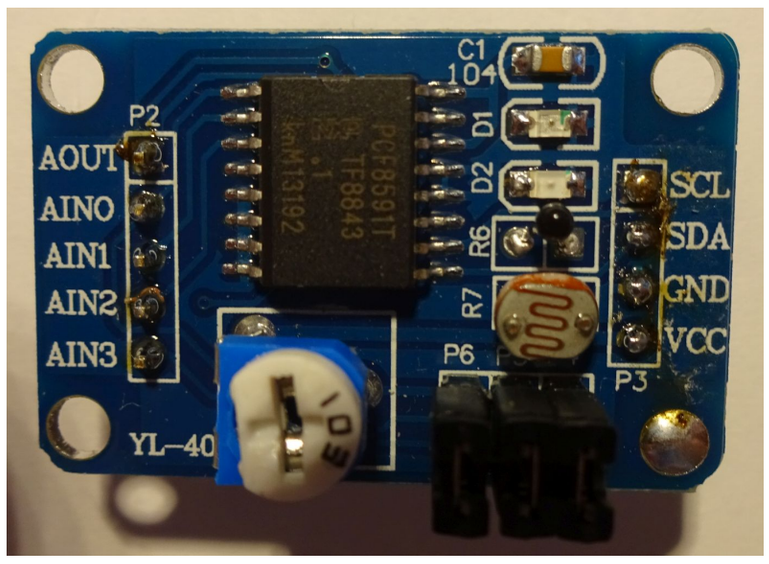
Über den I2C-Bus des Raspberry Pi einen Analog-Digital-Wandler (PCF8591) ansteuern

Der I^{2}C (sprich: I-Quadrat-C), auch I2C- oder SM-Bus ist eine Erfindung der frühen 1980er-Jahre von Philips (heute NXP Semiconductors), um auf kurzen Strecken, wie innerhalb eines Gerätes oder auf einer Platine Daten zu übertragen.

Dazu benutzt es eine Datenleitung (SDA, DA=Data) und eine Taktleitung (SCL, CL=Clock), auf der ein festgelegtes, byteweises Übertragungsprotokoll läuft. Außerdem benötigt es natürlich noch Strom (VCC und GND).

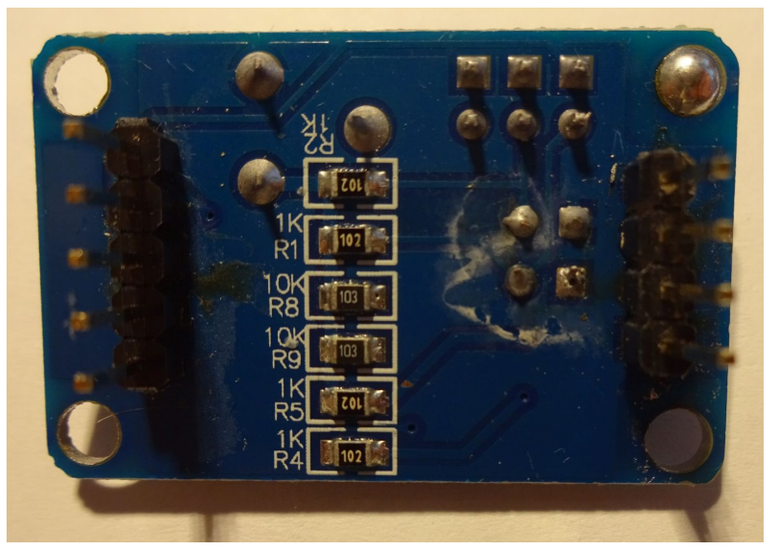
Das PCF8591-Modul mit eingebauten Sensoren

Dieses finden wir auf dem PCF8591-Modul wieder, dass ich mir besorgt habe. Die vier I2C-Leitungen schließen wir an die entsprechenden GPIO-Ports an Pin Nr. 3 und 5 (BCM 2 und 3), sowie GND und 3.3V an.



Der Raspi sorgt automatisch für die nötigen Pullup-Widerstände an den Anschlüssen. Ein sogenannter Bus-Master gibt den Takt und den Betriebsmodus vor und kann bis zu 116 Geräte (Slaves genannt) über ihre Adresse ansprechen.

Die Slaves nehmen Daten vom Master entgegen und schicken nach Ausführung eine Antwort zurück, passend zum Kommando. Welches Byte welches Kommando darstellt, ist vom angesprochenen Bauteil abhängig und aus dem entsprechenden Datenblatt ersichtlich.



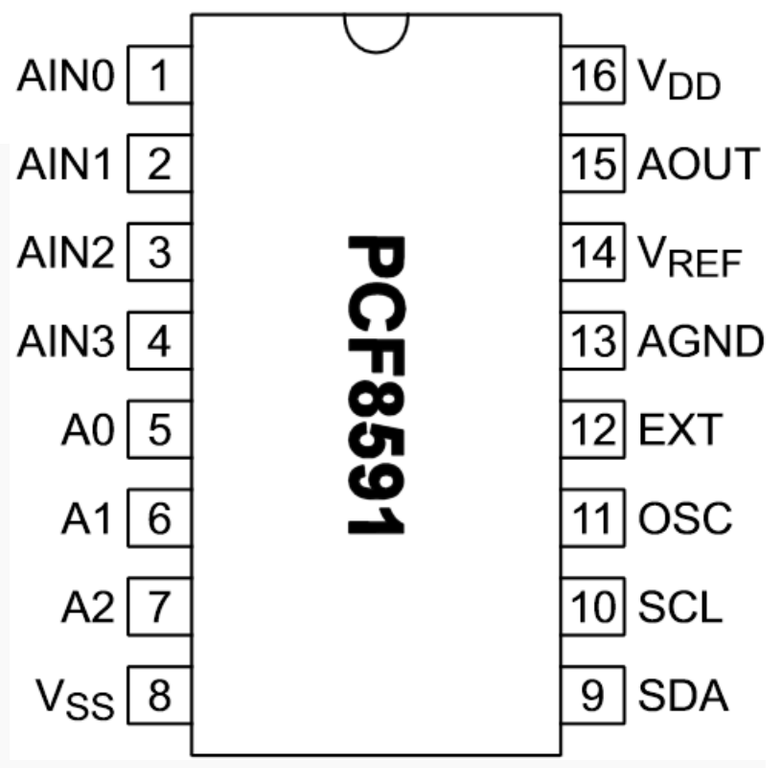
Auf der anderen Seite des Moduls finden wir die Anschlüsse für die 4 Analog-Eingänge (AIN0 bis AIN3) und einen Analog-Ausgang (AOUT). Außerdem befinden sich auf meinem Board die Jumper P4, P5 und P6, mit denen die drei eingebauten Sensoren auf die Analog-Eingänge geschaltet werden können.

Die Rückseite des Moduls ist eher uninteressant.

Ich habe die Header, die vorher zum Aufstecken von Kabel konzipiert waren, umgelötet und kann das Modul jetzt direkt aufs Breadboard stecken. Allerdings ist es jetzt so lang, dass ich dafür ein zweites Breadboard brauche, welches ich auch noch um die schmale Stromschiene an der Seite kürzen muss. Das ist nicht so optimal.

Der AD / DA - Wandler PCF8591

Wird ein PCF8501 ohne Modul verwendet sind die Anschlüsse:




```
Pin 1 bis 4: AIN0 bis AIN3: Analog Eingang 0 bis 3
Pin 5 bis 7: A0 bis A3:     Hardware Slave Adresse
Pin 8:       VSS:           GND
Pin 9:       SDA:           I2C Data
Pin 10:      SCL:           I2C Clock
Pin 11:      OSC:           Taktgeber input/output
Pin 12:      EXT:           Extern/Intern Schalter für
Taktgeber Input
                            intern = mit VSS verbinden
                            extern = mit VDD verbinden
Pin 13:      AGND:          Analog GND Versorgung
Pin 14:      VREF:          Referenzspannung
Pin 15:      AOUT:          Analog Ausgang
Pin 16:      VDD:           Versorgungsspannung (2.5-6V)
```

Da es sich beim PCF8591 um einen 8-bit-Wandler handelt, kann uns dieser ein angelegtes Analog-Signal in 256 Stufen (2 hoch 8) zurückliefern, je nachdem wie hoch die anliegende Spannung im Bereich von 0V bis 3.3V (unserer angelegten Versorgungsspannung) ist.

1.65V anliegende Spannung würde also bei 3.3V Bestriebsspannung zu einem Wert von 0x80 (128) führen.

```
Aufbau eines Steuerbytes für den PCF 8591:

0x80 0x40 0x20 0x10 0x08 0x04 0x02 0x01   Bit-Wertigkeit
  7    6    5    4    3    2    1    0    Bit-Nr.
  0    ^    ^----^    0    ^    ^----^--- Auswahl des Kanals
       |       |           |              00=K.0, 01=K.1, 10=K.2, 11=K.3
       |       |           `------------- wenn high: auto-increment
       |       `------------------------- Prog. d. Analog-Kanäle (s. Datenblatt)
       |                                  00=4 separate Kanäle
       |                                  01=3 verbundene Eingänge (verbindet Minuspol
       |                                       von Kanal 0 bis 2) (Differenzbetrieb)
       |                                  10=1x2 verbundene Eingänge (AIN2=(+) AIN3=(-))
       |                                  11=2x2 verbundene Eingänge (AIN0=(+) AIN1=(-),
       |                                         AIN2=(+) AIN3=(-))
       `--------------------------------- wenn high: Output aktiv
```

Den Raspberry Pi für I2C konfigurieren

Um I2C auf dem Raspi benutzen zu können, müssen wir erst einmal den zugehörigen Kernal-Treiber einrichten. Dies tun wir, indem wir das Raspi-Konfigurationsprogramm starten

```
sudo raspi-config
```

und unter *Interfacing Options* den Eintrag *I2C enable* wählen.

```
ls /dev/i*
```

sollte dann anzeigen:

```
/dev/i2c-1  /dev/initctl
```

I2C-Tools für die Kommandozeile

Jetzt können wir die I2C-Tools installieren:

```
sudo apt-get install i2c-tools
```

Danach steht uns das Tool i2cdetect auf der Kommandozeile zur Verfügung. Geben wir einmal *i2cdetect 1* ein (*i2cdetect -y 1*, um die Abfrage zu überspringen).

```
pi@raspberrypi:~ $ i2cdetect 1
WARNING! This program can confuse your I2C bus, cause data loss and worse!
I will probe file /dev/i2c-1.
I will probe address range 0x03-0x77.
Continue? [Y/n] y
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- 48 -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Hier wird uns nun eine Tabelle der Geräte unter den Adressen 0x03 (3) bis 0x77 (119) angezeigt, also insgesamt für 116 mögliche Geräte. Das Gerät an Adresse 0x48 meldet sich zurück.

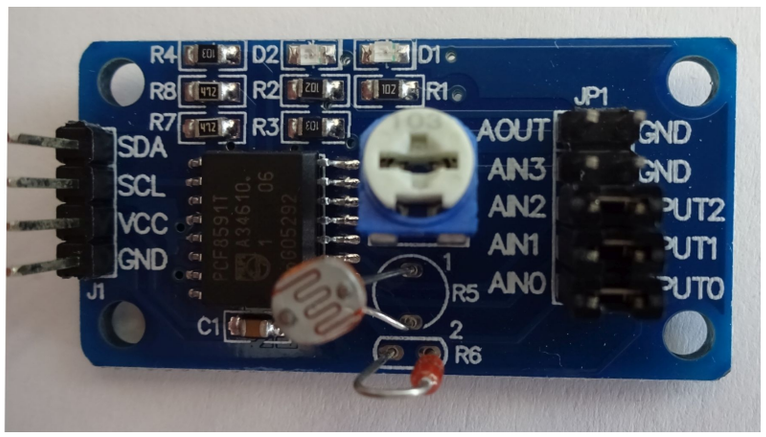
Das muss der Analog-Digital-Wandler sein. Einen allerersten Überblick können wir uns mit dem Tool i2cdump verschaffen:

```
pi@raspberrypi:~ $ i2cdump -y 1 0x48 b
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f    0123456789abcdef
00: 80 00 ff 7b 88 61 ff 6f 88 44 ff 8f 88 64 ff 69    ?..{?a.o?D.??d.i
10: 88 c0 78 1e dc ab 79 df b7 9b 78 15 d7 a9 77 ec    ??x???y???x???w?
20: c1 21 ff 04 cf 2c ff f0 c4 1d ff 05 cf 29 ff ee    ?!.??,.???.??).?
30: bf 80 86 82 80 80 80 80 80 80 80 80 80 80 80 80    ????????????????
40: 80 00 ff 54 88 51 ff 21 88 1a ff 57 88 50 ff 29    ?..T?Q.!??.W?P.)
50: 88 88 78 cb 86 80 78 8d 80 80 77 c1 84 80 77 90    ??x???x???w???w?
60: 80 00 ff bf 80 00 ff 93 83 00 ff c4 84 00 ff 85    ?..??..??..??..?
70: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80    ????????????????
80: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80    ????????????????
90: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80    ????????????????
a0: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80    ????????????????
b0: 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80    ????????????????
c0: 80 00 ff 59 88 50 ff 40 88 27 ff 65 88 4f ff 2f    ?..Y?P.@?'.e?O./
d0: 88 99 78 d8 93 82 77 a8 89 80 77 ce 89 80 77 a8    ??x???w???w???w?
e0: 8a 00 ff c9 8f 01 ff a2 8a 00 ff cc 8a 01 ff aa    ?..???.??..???.?
f0: 8e 80 80 80 80 80 80 80 80 80 80 80 80 80 80 80    ????????????????
```

Nur welcher Wert entspricht jetzt was? Die Beschreibung auf eBay zu dem Modul verspricht:

```
The INPUT3 analog input signal interface 3
INPUT2 thermistor signal input port is connected to
The INPUT1 has photoresistor signal input port connected to theINPUT0 is connected to the
signal input port of the potentiometer
```

Bei dem schlechten Englisch kann ich mir nur zusammenreimen, auch anhand der auf dem Modul befindlichen Bauteile, dass man einen Thermistor, also einen Wärmewiderstand und einen Photoresistor, also einen Lichtwiderstand messen und ausgeben kann, außerdem ist noch ein Poti verlötet, dass wahrscheinlich auch irgendeinen Wert zurückliefert kann. Nur wie rankommen?



Nachdem ich ein zweites, etwas anders aufgebautes Modul von einem anderen Hersteller bekommen habe, wurde mir klar, wozu die Jumper sind. Damit kann man die ersten drei der vier, ansonsten freien Analog-Eingänge, mit den auf dem Modul verbauten Bauteilen verbinden und so dann diese internen Komponenten auslesen. Ungejumpert bleiben die Eingänge dann zur freien Verfügung.

Kein gescheites Datenblatt? Dann hilft nur die alte Weisheit "Probieren geht über studieren!". Dazu nehmen wir das I2C-Tool ***i2cget*** zur Hilfe. Dieses erwartet die Nr. des

I2C-Bus (bei dem Raspi immer 1), die Speicheradresse des Gerätes (0x48 für unseren Wandler) und den Kanal des Analogeingangs (gleich Sensor im Moduls wenn gejumpert).

```
i2cget [-f] [-y] i2cbus chip-address [data-address [mode]]
```

Nach ein bisschen Rumprobieren komme ich zu dem Ergebnis:

```
i2cget -y 1 0x48 0  ->  Poti    0x00 = ganz links ... 0xFF = ganz rechts
i2cget -y 1 0x48 1  ->  Licht   0x00 = ganz hell  ... 0xFF = ganz dunkel
i2cget -y 1 0x48 2  ->  Wärme   0x00 = ganz heiß  ... 0xFF = ganz kalt
```

Das mit der Temperatur scheint noch eine ziemliche Rechnerei zu werden. Der Verlauf sieht nicht linear aus. Ob da zum Schluss ein genauer Wert bei herauskommt?

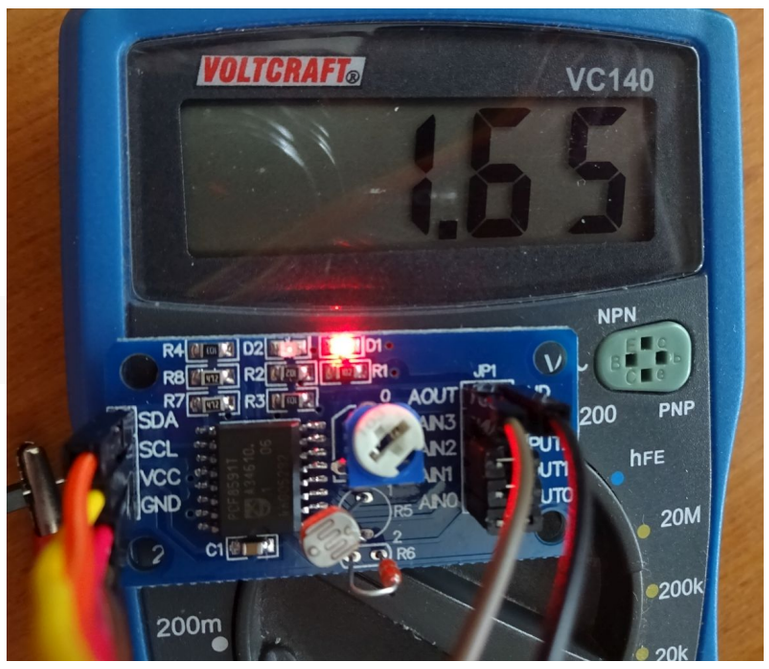
Das Gegenstück zum Auslesen auf der Kommandozeile mit *i2cget* ist übrigens ***i2cset***. Die Syntax dafür ist ganz ähnlich. Nur müssen wir hier beim Steuerbyte 0x40 zu der Kanaladresse addieren (siehe obige Tabelle), um den Ausgang zu aktivieren.

```
i2cset [-f] [-y] [-r] i2cbus chip-address data-address [value] ... [mode]
```

Wobei *-f* wieder für Force steht, also den erzwungenen Zugriff auf das Gerät, auch wenn es bereits belegt ist (nicht empfehlenswert). *-r* macht einen Verify und liest die Speicherstelle nach dem Beschreiben wieder aus und vergleicht, ob der eben übertragene Wert auch drinsteht. Das hat bei mir aber nicht so recht geklappt, obwohl richtig geschrieben worden ist.

Wenn unser Kanal für den Analog-Ausgang 0 ist, und wir wollten einen Spannungswert von 1.65V (die halbe Versorgungsspannung) an den analogen Ausgang bringen, dann müssten wir den Wert 0x80 (die Hälfte von 0x100) mit einem Steuerbyte von 0x40 schreiben:

```
i2cset -y -r 1 0x48 0x40 0x80
```



Passt! Witzigerweise leuchtet die 2. LED auf dem Modul in der Intensitivität des Ausgangs, sprich bei 0x00 ist sie aus und bei 0xFF maximal hell. So kann man einen Schreibversuch auch optisch verifizieren, ohne erst ein Voltmeter anzuschließen.

Ansteuerung mit Python

Jetzt bleibt nur noch die Frage: Wie spreche ich die Sensoren über Python in meinem Programm an? Zuerst müssen wir einmal die entsprechende Library für Python installieren:

```
sudo apt-get install -y python-smbus
```

Danach können wir unseren Python-Code erweitern:

```
...
import smbus                # für den I2C-Bus
import math
...

# Definitionen für den AD/DA-Wandler
i2cAdda = 0x48              # I2C-Adresse des Wandlers
addaPoti =   0x00           # Sensoren
addaLicht =  0x01
addaWaerme = 0x02
addaOut =    0x40
```

```
i2c = smbus.SMBus(1)     # Erzeugen und Öffnen einer I2C-Instanz

...

def readI2c(i2cDeviceAddr, channel):
   wert=i2c.read_byte_data (i2cDeviceAddr, channel)  # erster Zugriff weist Bauteil an, neu zu
lesen
   sleep (0.01)
   wert=i2c.read_byte_data (i2cDeviceAddr, channel)  # erst zweiter Zugriff liefert aktuellen
Wert
   return wert

def writeI2c(i2cDeviceAddr, channel, value):
   i2c.write_byte_data (i2cDeviceAddr, 0x40+channel, value)

...
```

Die eigentliche Abfrage des Sensors über den I2C-Bus erfolgt in der Zeile *wert=i2c.read_byte_data (GeräteAdresse, Kanal)*. Dabei ist eine Kleinigkeit zu beachten: Ein Read gibt den aktuellen im Speicher des Wandlers gehaltenen Wert zurück und holt dann einen neuen vom Sensorf. Will man also den aktuellen Wert des Sensors, so muss man zweimal hintereinander lesen.

Geschrieben kann entsprechend mit *write_byte_data (GeräteAdresse, (Kanal + 0x40), ByteWert)*.
