Hardware Installation auf Raspberry Pi

Evil
m.stroh@softhome.net
http://evil.hn.vc

1. April 2013

Inhaltsverzeichnis

1 Vorwort

Diese Dokumentation ist eine Erweiterung der Dokumentationen 'Debian Server Installation auf Raspberry Pi'. Es wird deshalb von einem bestehenden Debian System ausgegangen. Das Dokument beschreibt die Installation von modifizierter oder gebastelter Hardware die mit einer Raspberry Pi verbunden werden kann.
Jegliche Haftung für Schäden, die direkt oder indirekt aus der Benutzung dieser Anleitung entstehen, sind ausgeschlossen!

2 Hardware

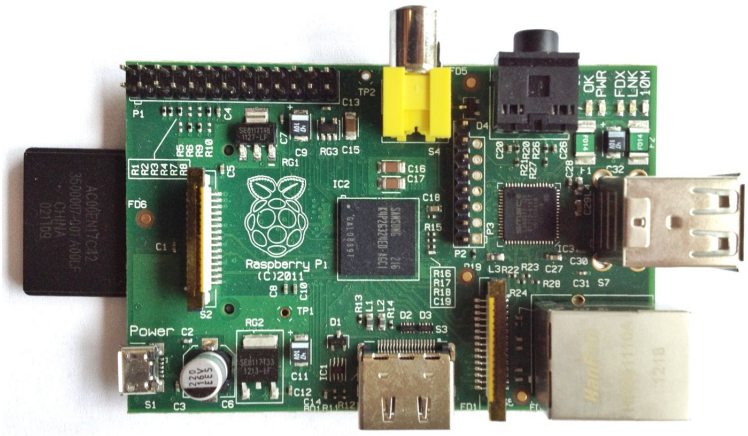

Abbildung 1: Raspberry Pi Board

3 GPIO

Nützliche Links:
GPIO Informationen: `http://elinux.org/RPi_Low-level_peripherals`

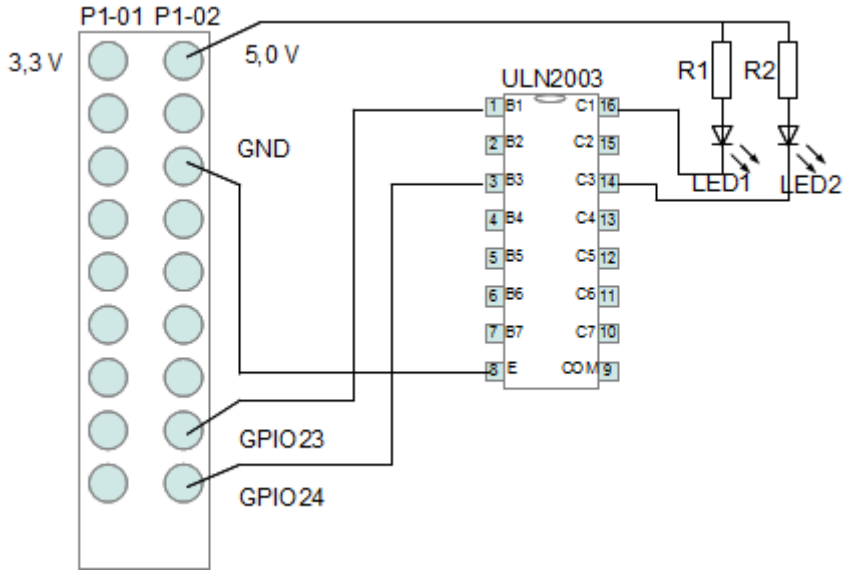

Abbildung 2: GPIO an ULN2003 an LED

$I_f = 10\ mA$
$U_f = 1{,}6\ V$
$U_{CE} = 1\ V$
$R_1 = R_2 = R_{LED}$
$R_{LED} = \frac{U - U_f - U_{CE}}{I_f} = \frac{5-1{,}6-1{,}0}{0{,}01} = 240\ \Omega$

Für R_1 und R_2 gewählt 200 Ω.

3.1 Dateisystem

```
sudo -i

echo "23" > /sys/class/gpio/export
echo "out" > /sys/class/gpio/gpio23/direction
echo "1" > /sys/class/gpio/gpio23/value
echo "0" > /sys/class/gpio/gpio23/value

echo "24" > /sys/class/gpio/export
echo "out" > /sys/class/gpio/gpio24/direction
echo "1" > /sys/class/gpio/gpio24/value
echo "0" > /sys/class/gpio/gpio24/value
```

3.2 wiringPi (Command Line Tool/C-Library)

Nützliche Links:
wiringpi: https://projects.drogon.net/raspberry-pi/wiringpi/download-and-install/
Library Funktionen: https://projects.drogon.net/raspberry-pi/wiringpi/functions/ Pin Bezeichnung für Library: https://projects.drogon.net/raspberry-pi/wiringpi/pins/

```
sudo apt-get install git-core
cd /usr/src
git clone git://git.drogon.net/wiringPi
cd wiringPi
git pull origin
./build
```

3.2.1 Ausgang

```
gpio -g mode 23 out
gpio -g write 23 1
gpio -g write 23 0

gpio -g mode 24 out
gpio -g write 24 1
gpio -g write 24 0

cd ../examples/
```

Listing 1: 'test.c' C Source Code

```
//
//      Simple test program to test the wiringPi functions
//      for GPIO23 und GPIO24 output
//
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
#include <signal.h>
#include <string.h>

static sig_atomic_t end = 0;
const int GPIO23=23;
const int GPIO24=24;

static void sighandler(int signo){
  end = 1;
}

int main (void){
  struct sigaction sa;

  printf("Raspberry Pi wiringPi test program (press ctrl+c to quit)\n");

  if (wiringPiSetupGpio() == -1){
    printf("wiringPiSetup failed\n\n");
    exit (1) ;
  }

  memset(&sa, 0, sizeof(struct sigaction));
  sa.sa_handler = sighandler;
  sigaction(SIGINT, &sa, NULL);
  sigaction(SIGQUIT,&sa, NULL);
  sigaction(SIGTERM,&sa, NULL);

  pinMode(GPIO23, OUTPUT);
  pinMode(GPIO24, OUTPUT);

  delay(500);

  printf("all off\n");
  digitalWrite(GPIO23, LOW);
  digitalWrite(GPIO24, LOW);
  delay(1000);
  printf("all on\n");
  digitalWrite(GPIO23, HIGH);
  digitalWrite(GPIO24, HIGH);
  delay(1000);

  printf("blinking...\n");
  while(!end){
    digitalWrite(GPIO23, HIGH);
```

```
    digitalWrite(GPIO24, LOW);
    delay(1000);
    digitalWrite(GPIO23, LOW);
    digitalWrite(GPIO24, HIGH);
    delay(1000);
  }
  printf("\nall off\n");
  digitalWrite(GPIO23, LOW);
  digitalWrite(GPIO24, LOW);

  return 0;
}
```

```
gcc -o test test.c -L/usr/local/lib -lwiringPi -I/usr/local/include -Wall
./test
```

```
Raspberry Pi wiringPi test program (press ctrl+c to terminate)
all off
all on
blinking...
^C
all off
```

3.2.2 PWM

pwm: Tastverhältnis in 0-1024 (256 = 1/4 High, 3/4 Low)
pwmc: Frequenz in 0-4096 (Berechnung $f = 9600\ kHz/pwmc$ bzw. $pwmc = 9600\ kHz/f$)

```
gpio pwm 1 0
gpio mode 1 pwm
gpio pwmc 300
gpio pwm 1 512
```

4 I2C

Tabelle 1: Sensoren

Bezeichung	**Funktion**	**Adresse(n)**
[1] HH10D	Luftfeuchtigkeit	0x51 (1)[1]
[2] HP03S	Luftdruck	0x50, 0x77 (1) [2]

[1] Adresse im Datenblatt falsch angegeben
[2] Adresse im Datenblatt mit 8 Bit angegeben (inkl. RW Bit 0)

[1] http://www.hoperf.com/upload/sensor/HH10D.pdf
[2] http://www.hoperf.com/upload/sensor/HP03S.pdf

Tabelle 2: ICs

Bezeichung	**Funktion**	**Adresse(n)**
[1] PCA9515	I2C Repeater (5 V tolerant)	-
[2] DS2482-100S	1-Wire Bus Master	0x18 - 0x1B (4)
[3] PCA8574	8-Bit I/O Extender	0x20 - 0x27 (8)
[4] MCP23017	16-Bit I/O Extender	0x20 - 0x27 (8)
[5] PCF8583	Clock, Calendar/Counter	0x50 - 0x51 (2)
[6] PCF8591	8-bit A/D and D/A converter	0x48 - 0x4F (8)

[1] http://www.nxp.com/documents/data_sheet/PCA9515.pdf
[2] http://datasheets.maximintegrated.com/en/ds/DS2482-100.pdf
[3] http://www.nxp.com/documents/data_sheet/PCA8574_PCA8574A.pdf
[4] http://ww1.microchip.com/downloads/en/devicedoc/21952b.pdf
[5] http://www.nxp.com/documents/data_sheet/PCF8583.pdf
[6] http://www.nxp.com/documents/data_sheet/PCF8591.pdf

4.1 Temperatursensor DS18B20 an DS2482-100

Nützliche Links:
Bestellung DS2482-100: `http://www.reichelt.de/Real-Time-Clock/DS-2482-100S/3//index.html?ACTION=3&GROUPID=2949&ARTICLE=69961`
Bestellung DS18B20: `http://www.reichelt.de/ICs-CA-ISD-/DS-18B20Z/3//index.html?ARTICLE=58170`
Datenblatt DS2482-100: `http://datasheets.maxim-ic.com/en/ds/DS2482-100.pdf`
Datenblatt DS18B20: `http://datasheets.maxim-ic.com/en/ds/DS18B20.pdf`
Anleitung: `http://raspberrypi.homelabs.org.uk/i2c-connected-1-wire-masters/`
Anleitung: `http://www.fischer-net.de/hausautomation/haustechnik/1-wire/40-1-wire-software-unter-linux-teil-2.html`

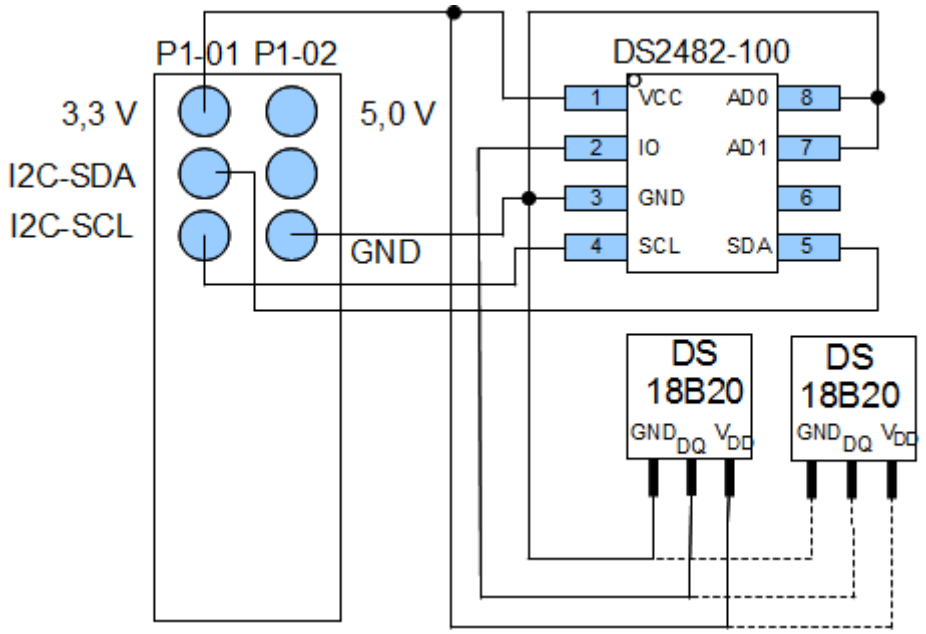

Abbildung 3: Temperatursensor DS18B20 an DS2482-100 an I2C Bus

Adresse am DS2482-100 Chip:
A0=GND, A1=GND ergibt Adresse 0x18.

```
sudo -i
modprobe i2c-bcm2708
modprobe i2c-dev
```

/etc/modules [-rw-r--r-- root root]

```
i2c-bcm2708
i2c-dev
```

```
dmesg | grep i2c
```

```
bcm2708_i2c bcm2708_i2c.0: BSC0 Controller at 0x20205000 (irq 79) (baudrate 100k)
bcm2708_i2c bcm2708_i2c.1: BSC1 Controller at 0x20804000 (irq 79) (baudrate 100k)
i2c /dev entries driver
```

```
apt-get install i2c-tools
i2cdetect -l
```

```
i2c-0   i2c             bcm2708_i2c.0                           I2C adapter
i2c-1   i2c             bcm2708_i2c.1                           I2C adapter
```

Alte Raspberry Pi (256 MB):

```
i2cdetect -y 0
```

Neue Raspberry Pi (512 MB):

```
i2cdetect -y 1
```

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- 18 -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

```
apt-get install owfs owserver
```

Einbinden des 1-Wire Buses ins Dateisystem:

```
mkdir /owfs
chmod 777 /owfs
owfs -m /owfs /dev/i2c-1
cat -T /owfs/bus.1/interface/settings/name && echo
```

```
DS2482-100
```

```
cat /owfs/28.A5F672020000/temperature && echo °C
```

```
24.69°C
```

```
fusermount -u /owfs
```

4.2 Zähler PCF8583

PCF8583 ist ein IC der zur Zeitmessung (mit einem 32,768 KHz Quarz) oder zur Ereigniszählung verwendet werden kann.
Die Adresse kann über den Pin A0 eingestellt werden, somit können maximal 2 ICs an einem Bus betrieben werden.
Adresse A0=Low: 0x50
Adresse A0=High: 0x51

4.2.1 Zeit

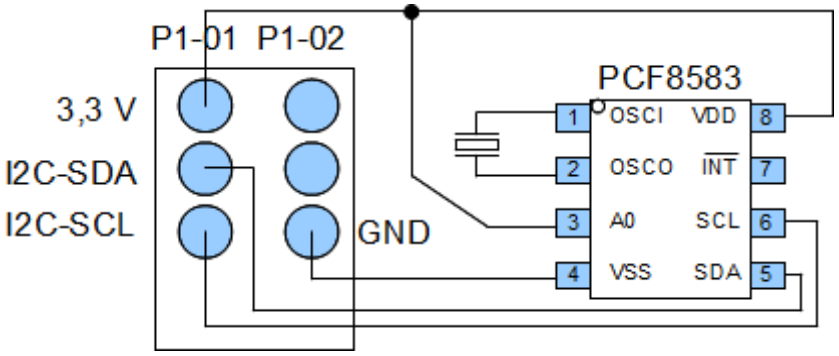

Abbildung 4: PCF8583 mit Quarz an I2C Bus

Adresse (A0=Gnd): 0x51

```
i2cdetect -y 1
```

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- 51 -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

```
i2cget -y 1 0x51 0x03 | cut -b 3-4 | tr -d '\n' && echo -n " min " && \
i2cget -y 1 0x51 0x02 | cut -b 3-4 | tr -d '\n' && echo " sec"
```

```
27 min 39 sec
```

Test 30 Sekunden Messung:

```
i2cget -y 1 0x51 0x03 | cut -b 3-4 | tr -d '\n' && echo -n " min " && \
i2cget -y 1 0x51 0x02 | cut -b 3-4 | tr -d '\n' && echo " sec" && \
sleep 30 && \
i2cget -y 1 0x51 0x03 | cut -b 3-4 | tr -d '\n' && echo -n " min " && \
i2cget -y 1 0x51 0x02 | cut -b 3-4 | tr -d '\n' && echo " sec"
```

```
28 min 36 sec
29 min 06 sec
```

Stopp/Pause:

```
i2cset -y 1 0x51 0x00 0x82
```

Reset:

```
i2cset -y 1 0x51 0x03 0x00
i2cset -y 1 0x51 0x02 0x00
i2cset -y 1 0x51 0x01 0x00
```

Start/Fortsetzen:

```
i2cset -y 1 0x51 0x00 0x02
```

4.2.2 Ereignis

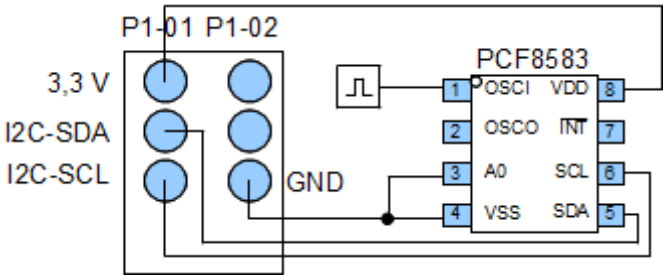

Abbildung 5: PCF8583 mit externen Takt an I2C Bus

Adresse (A0=Low): 0x50

```
i2cdetect -y 1

     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: 50 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Ereigniszähler setzen:

```
i2cset -y 1 0x50 0x00 0x20
```

Stopp/Pause:

```
i2cset -y 1 0x50 0x00 0xA0
```

Reset:

```
i2cset -y 1 0x50 0x03 0x00
i2cset -y 1 0x50 0x02 0x00
i2cset -y 1 0x50 0x01 0x00
```

Start:

```
i2cset -y 1 0x50 0x00 0x20
```

4.3 Luftfeuchtesensor HH10D mit Frequenzmessung über Zähler PCF8583

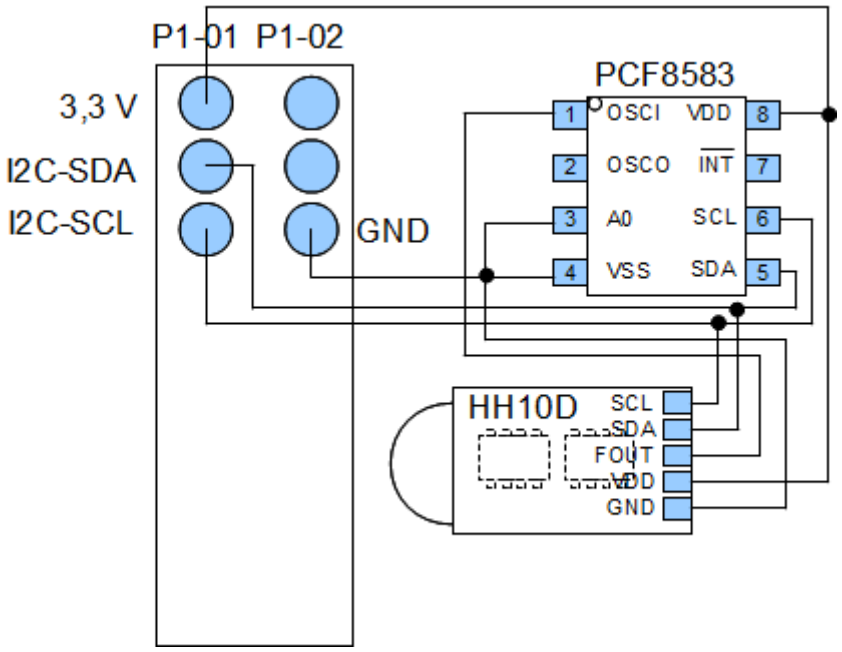

Abbildung 6: Luftfeuchtesensor HH10D an PCF8583 an I2C Bus

EEPROM Adresse: 0x51
Sens Adresse im EEPROM: 0x0A
Offset Adresse LSB im EEPROM: 0x0C
Offset Adresse MSB im EEPROM: 0x0D
Berechnung Relative Luftfeuchte: $(Offset - Frequenz) * Sens/2^{10})$

Der Sensor liefert den Luftfeuchtewert als Frequenz (Signalform: Rechteck 0-V_{DD}, 5-10 kHz). Da bei der Raspberry Pi kein Frequenzeingang vorhanden ist kann diese Aufgabe durch einen Zähler IC PCF8583 erledigt werden.
Achtung die EEPROM Adresse überschneidet sich mit dem IC PCF8583 wenn die Adressleitung A0 auf High gesetzt wird.
Es kann also der IC PCF8583 als Zähler für die Frequenzmessung nur mit A0 auf Low mit dem Luftfeuchtesensor kombiniert werden. (Alternativ können die Kalibierdaten auch nur einmal Ausgelesen werden und später der Bus nicht angeschlossen werden).
Der Zähler kann bis 999.999 zählen, das entspricht bei maximal 10 kHz ca. 1 Minute und 40 Sekunden ohne Überlauf. Ich habe mich für 10 Sekunden Messzeit entschieden.

```
i2cdetect -y 1

     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
```

```
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: 50 51 -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Kalibierdaten: **HH10D_const.sh [-rwxr-xr-x root root]**

```
#!/bin/bash
i2cget -y 1 0x51 0x0A
i2cget -y 1 0x51 0x0C | tr -d '\n'  && \
i2cget -y 1 0x51 0x0D | cut -b 3-5
```

```
./HH10D_const.sh | tee HH10D_const.txt
```

```
0x01
0x1d3b
```

Sens: 0x01
Offset: 0x1D3B

Frequenzmessung mit Ereigniszähler:

```
i2cset -y 1 0x50 0x00 0xA0 && \
i2cset -y 1 0x50 0x03 0x00 && \
i2cset -y 1 0x50 0x02 0x00 && \
i2cset -y 1 0x50 0x01 0x00 && \
i2cset -y 1 0x50 0x00 0x22 && \
sleep 10 && \
i2cset -y 1 0x50 0x00 0xA0 && \
i2cget -y 1 0x50 0x03 | cut -b 3-4 | tr -d '\n' && \
i2cget -y 1 0x50 0x02 | cut -b 3-4 | tr -d '\n' && \
i2cget -y 1 0x50 0x01 | cut -b 3-4 | tr -d '\n' && echo " (10*Hz)"
```

```
071274 (10*Hz)
```

Gemessen mit Fluke 199C Scopmeter 7,11 kHz.

```
apt-get install bc
```

RH.sh [-rwxr-xr-x root root]

```
#!/bin/bash
CNT_ADDRESS=0x50

SENSH=`cat $1 | head -n 1 | cut -b 3-4 | tr '[:lower:]' '[:upper:]'`
SENS=`echo "obase=10; ibase=16; $SENSH" | bc`
#echo "Sens: 0x$SENSH (${SENS})"

OFFSETH=`cat $1 | tail -n 1 | cut -b 3-6 | tr '[:lower:]' '[:upper:]'`
OFFSET=`echo "obase=10; ibase=16; $OFFSETH" | bc`
#echo "Offset: 0x$OFFSETH (${OFFSET})"

FREQ10=`i2cset -y 1 $CNT_ADDRESS 0x00 0xA0 &&
i2cset -y 1 $CNT_ADDRESS 0x03 0x00 &&
i2cset -y 1 $CNT_ADDRESS 0x02 0x00 &&
i2cset -y 1 $CNT_ADDRESS 0x01 0x00 &&
i2cset -y 1 $CNT_ADDRESS 0x00 0x22 &&
sleep 10 &&
```

```
i2cset -y 1 $CNT_ADDRESS 0x00 0xA0 &&
i2cget -y 1 $CNT_ADDRESS 0x03 | cut -b 3-4 | tr -d '\n' &&
i2cget -y 1 $CNT_ADDRESS 0x02 | cut -b 3-4 | tr -d '\n' &&
i2cget -y 1 $CNT_ADDRESS 0x01 | cut -b 3-4 | tr -d '\n'`
FREQ=`echo "${FREQ10}/10" | bc`
#echo "Frequency: $FREQ Hz"

RH=`echo "scale=3; ($OFFSET-$FREQ)*${SENS}/2^10*100" | bc`
echo "Realtiv humidity=$RH %"
```

./RH.sh HH10D_const.txt

```
Sens: 0x01 (1)
Offset: 0x1D3B (7483)
Frequency: 7146 Hz
Realtiv humidity: 32.900 %
```

4.4 Luftdrucksensor HP03S

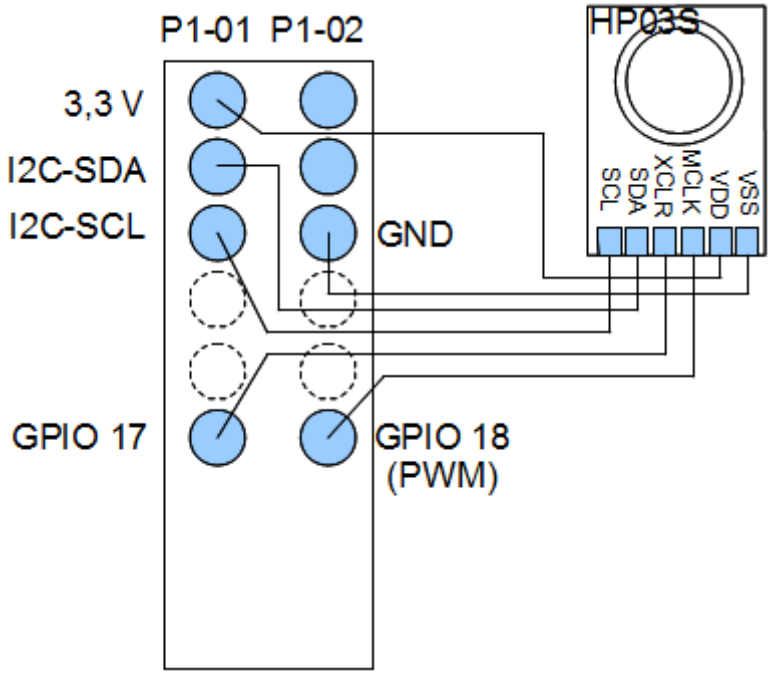

Abbildung 7: Luftdrucksensor HP03S an I2C Bus

EEPROM Adresse: 0x50
ADC Adresse: 0x77
GPIO 17 (wiringPi Pin 0): XCLR
GPIO 18 (wiringPi Pin 1): MCLK

XCLR muss auf Low gesetzt werden bevor das EEPROM gelesen wird.
XCLR muss auf High gesetzt werden und das 'Master Clock'-Signal (32 kHz) muss auf MCLK anliegen bevor der ADC gelesen wird.

Sequenz Luftdruck ADC-Wert D1 lesen:
Schreiben 0xFFF0
Warten 40 ms
Schreiben 0xFD
Lesen 2 Byte (D1 Wert)

Sequenz Temperatur ADC-Wert D2 lesen:
Schreiben: 0xFFE8
Warten 40 ms
Schreiben 0xFD
Lesen 2 Byte (D2 Wert)

Berechnung Luftdruck und Temperatur:

Tabelle 3: Parameter

Bezeichung	**Variable**	**Adresse(n)**
Sensity coefficient	C1	0x10 (MSB), 0x11 (LSB)
Offset coefficient	C2	0x12 (MSB), 0x13 (LSB)
Temperature soefficient of sensitivity	C3	0x14 (MSB), 0x15 (LSB)
Temperature soefficient of offset	C4	0x16 (MSB), 0x17 (LSB)
Reference Temperature	C5	0x18 (MSB), 0x19 (LSB)
Temperature Coefficient of Temperature	C6	0x1A (MSB), 0x1B (LSB)
Offset Fine Tuning	C7	0x1C (MSB), 0x1D (LSB)
Sensor Specific Parameter	A	0x1E
Sensor Specific Parameter	B	0x1F
Sensor Specific Parameter	C	0x20
Sensor Specific Parameter	D	0x21

Wenn $D2 >= C5$:
$dUT = D2 - C5 - ((D2 - C5)/2^7) * ((D2 - C5)/2^7) * A/2^C$
Wenn $D2 < C5$:
$dUT = D2 - C5 - ((D2 - C5)/2^7) * ((D2 - C5)/2^7) * B/2^C$
$OFF = (C2 + (C4 - 1024) * dUT/2^{14}) * 4$
$SENS = C1 + C3 * dUT/2^{10}$
$X = SENS * (D1 - 7168)/2^{14} - OFF$
$P = X * 10/2^5 + C7$
$T = 250 + dUT * C6/2^{16} - dUT/2^D$

```
i2cdetect -y 1

     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: 50 -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- 77
```

HP03S_const.sh [-rwxr-xr-x root root]

```
#!/bin/bash
gpio mode 0 out
gpio write 0 0
i2cget -y 1 0x50 0x10 |  tr -d '\n' &&  i2cget -y 1 0x50 0x11 | cut -b 3-5
i2cget -y 1 0x50 0x12 |  tr -d '\n' &&  i2cget -y 1 0x50 0x13 | cut -b 3-5
i2cget -y 1 0x50 0x14 |  tr -d '\n' &&  i2cget -y 1 0x50 0x15 | cut -b 3-5
i2cget -y 1 0x50 0x16 |  tr -d '\n' &&  i2cget -y 1 0x50 0x17 | cut -b 3-5
i2cget -y 1 0x50 0x18 |  tr -d '\n' &&  i2cget -y 1 0x50 0x19 | cut -b 3-5
i2cget -y 1 0x50 0x1A |  tr -d '\n' &&  i2cget -y 1 0x50 0x1B | cut -b 3-5
i2cget -y 1 0x50 0x1C |  tr -d '\n' &&  i2cget -y 1 0x50 0x1D | cut -b 3-5
i2cget -y 1 0x50 0x1E
i2cget -y 1 0x50 0x1F
```

```
i2cget -y 1 0x50 0x20
i2cget -y 1 0x50 0x21
```

```
./HP03S_const.sh | tee HP03S_const.txt
```

```
0x420a
0x0d80
0x0124
0x0388
0x82b1
0x1750
0x09c4
0x07
0x1e
0x06
0x09
```

Tabelle 4: Parameter

Bezeichung	**Variable**	**Wert**
Sensity coefficient	C1	0x420a (16906)
Offset coefficient	C2	0x0d80 (3456)
Temperature soefficient of sensitivity	C3	0x0124 (292)
Temperature soefficient of offset	C4	0x0388 (904)
Reference Temperature	C5	0x82b1 (33457)
Temperature Coefficient of Temperature	C6	0x1750 (5968)
Offset Fine Tuning	C7	0x09c4 (2500)
Sensor Specific Parameter A	A	0x07 (7)
Sensor Specific Parameter B	B	0x1e (30)
Sensor Specific Parameter C	C	0x06 (6)
Sensor Specific Parameter D	D	0x09 (9)
Luftdruck	D1	0xA87C (43132)
Temperatur	D2	0x80C7 (32967)

Listing 2: 'HP03S.c' C Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <errno.h>
#include <linux/i2c-dev.h>

unsigned short ReadAddress(int fd, char nAddress, const char* szDataName){
  char buf[3] = {0};

  buf[0] = 0xFF;
  buf[1] = nAddress;
  if (write(fd,buf,2) != 2) {
    fprintf(stderr,
    "Failed to write to the i2c bus (%s)\n",strerror(errno));
    return 0;
  }
  usleep(80000);
  buf[0] = 0xFD;
  if (write(fd,buf,1) != 1) {
    fprintf(stderr,
    "Failed to write to the i2c bus (%s)\n",strerror(errno));
    return 0;
  }

  unsigned short* pData=(unsigned short*)&buf[1];
  if (read(fd,buf,2) != 2) {
    fprintf(stderr,
    "Failed to read from the i2c bus (%s)\n",strerror(errno));
    return 0;
  } else {
    buf[2]=buf[0];
    //printf("%s data: 0x%X (%hu)\n", szDataName,*pData,*pData);
  }
  return *pData;
}


void main(int argc, char* argv[]){
  int fd;
  char device[20];
  int D1,D2;
  int nParameters[cnConstants];
  const int SlaveAddr   = 0x77;
  const int cnConstants = 11;
  const int cnMaxLine   = 10;

  if(argc>=2){
    FILE* file;
    const char* szFilename=NULL;
    const int cnMaxLine=10;
    char szBuffer[cnMaxLine+1];
```

```
  szFilename=argv[1];
  if(szFilename)
    file = fopen(szFilename,"rb");
  if(file){
    int nParameterCount;
    for(nParameterCount=0;nParameterCount<cnConstants;nParameterCount++){
      if(!fgets(szBuffer,cnMaxLine,file)){
        fprintf(stderr,
        "Error reading line %d from file 's'\n",nParameterCount+1,szFilename);
        exit(1);
      }
      if(sscanf(szBuffer,"%X",&nParameters[nParameterCount])!=1){
        fprintf(stderr,
        "Error scanning line %d from file 's'\n",nParameterCount+1,szFilename);
        exit(1);
      }
    }
    fclose(file);
  }
  else{
    fprintf(stderr,"Can't open file 's' for reading\n",szFilename);
    exit(1);
  }

}
else{
  fprintf(stderr,"Constant file parameter missing\n");
  exit(1);
}

sprintf(device,"/dev/i2c-1");
if ((fd = open(device,O_RDWR)) < 0) {
  fprintf(stderr,"Failed to open i2c bus '%s'\n",device);
      exit(1);
  }
  if (ioctl(fd,I2C_SLAVE,SlaveAddr) < 0) {
  fprintf(stderr,
  "Failed to acquire i2c bus access or talk to slave %X\n",SlaveAddr);
  close(fd);
  exit(1);
}

D1=ReadAddress(fd,0xF0,"Pressure");
D2=ReadAddress(fd,0xE8,"Temperature");
usleep(50000); //drop first reading
D1=ReadAddress(fd,0xF0,"Pressure");
D2=ReadAddress(fd,0xE8,"Temperature");
close(fd);
if(D1!=0 && D2!=0){
  int C1,C2,C3,C4,C5,C6,C7,A,B,C,D;
  int dUT,OFF,SENS,X,P,T;

  C1=nParameters[0];
```

```
    C2=nParameters[1];
    C3=nParameters[2];
    C4=nParameters[3];
    C5=nParameters[4];
    C6=nParameters[5];
    C7=nParameters[6];
    A=nParameters[7];
    B=nParameters[8];
    C=nParameters[9];
    D=nParameters[10];

    if(D2>=C5)
      dUT=D2-C5-((D2-C5)/128)*((D2-C5)/128)*A/(2<<C);
    else
      dUT=D2-C5-((D2-C5)/128)*((D2-C5)/128)*A/(2<<C);
    OFF=(C2+(C4-1024)*dUT/16384)*4;
    SENS=C1+C3*dUT/1024;
    X=SENS*(D1-7168)/16384-OFF;
    P=X*10/32+C7;
    T=250+dUT*C6/65536-dUT/(2<<D);

    printf("Pressure=%.1f hPa\tTemperature=%.1f C\n",P/10.0f,T/10.0f);
    exit(0);
  }
  else{
    fprintf(stderr,
    "ADC read error (i2c bus ok?, XCLR high?, MCLK=32kHz?\n");
    exit(1);
  }
}
```

```
gcc HP03S.c -o Pressure
```

Pressure.sh [-rwxr-xr-x root root]

```
#!/bin/bash
# MCLK 32 KHz, 1/1 duty cycle
gpio pwm 1 0
gpio mode 1 pwm
gpio pwmc 300
gpio pwm 1 512
gpio mode 0 out
# XCLR High
gpio write 0 1
./Pressure $1
# XCLR Low
gpio write 0 0
# MCLK off (Low)
gpio pwm 1 0
```

```
./Pressure.sh HP03S_const.txt

Pressure=967.7 hPa      Temperature=20.6 C
```