APRS Wetterstation

Ich möchte hier kurz eine kleine Wetterstation beschreiben die die Daten eines BME280 Sensors auf die aprs.fi Webseite bringt

Hintergrund

Vor einiger Zeit hat Wolfgang, DF7PN einen Sensor im Holzturm installiert, der auf der APRS Seite die aktuelle Temperatur anzeigt. Soweit ich weiss hat er dazu ein NodeMCU Modul verwendet.

Ich habe jetzt lange nicht mehr mit NodeMCUs gearbeitet und der Aufwand mir wieder eine Arduino IDE zu installieren und mich wieder in das Thema reinzuarbeiten war mir einfach zu groß.

In Raspberry PI Computer muss ich mich nicht einarbeiten, also war klar, ich nehme einen Raspberry PI Zero dafür.

Ausserdem greife ich schon ein paar Jahre die Daten meines Feinstaubsensors ab und liefere Temperatur, Luftfeuchte und Luftdruck auf die APRS Seite. Dieses Script musste also nur *umgebaut* werden.

Benötigte Teile

Natürlich sollte das ganze wieder möglichst kostengümstig umzusetzen sein.

- Raspberry PI Zero W, besser Zero 2 W
- BME280 Sensor
- Ein paar Dupont Kabel
- Netzteil für den PI
- Gehäuse nach persönlichem Geschmack

Persönliche Designanforderungen

Ich mag die Unix Denkweise, ein Programm für einen Job. Das System sollte also aus zwei Komponenten bestehen

- Messdatenerfassung
- Datenübertragung zu APRSIS

Vorarbeiten

Wie ein PI aufgesetzt wird, ist ja bekannt und wurde auch von mir schon in einigen Vorträgen gezeigt. Ich entschied mich für die aktuelle Trixie Version ohne Desktop.

Nach der Installation des Grundsystems müssen noch ein paar Pakete nachinstalliert werden. Unix Gurus und Sicherheitsfanatiker werden mir meinen vereinfachten Ansatz verzeihen 😃

Aufsetzen der fehlenden Pakete

Ein paar der Pakete werden nicht wirklich gebraucht, aber sind meine persönliche Macke. Die hab ich einfach gerne auf jedem meiner Linux Systeme.

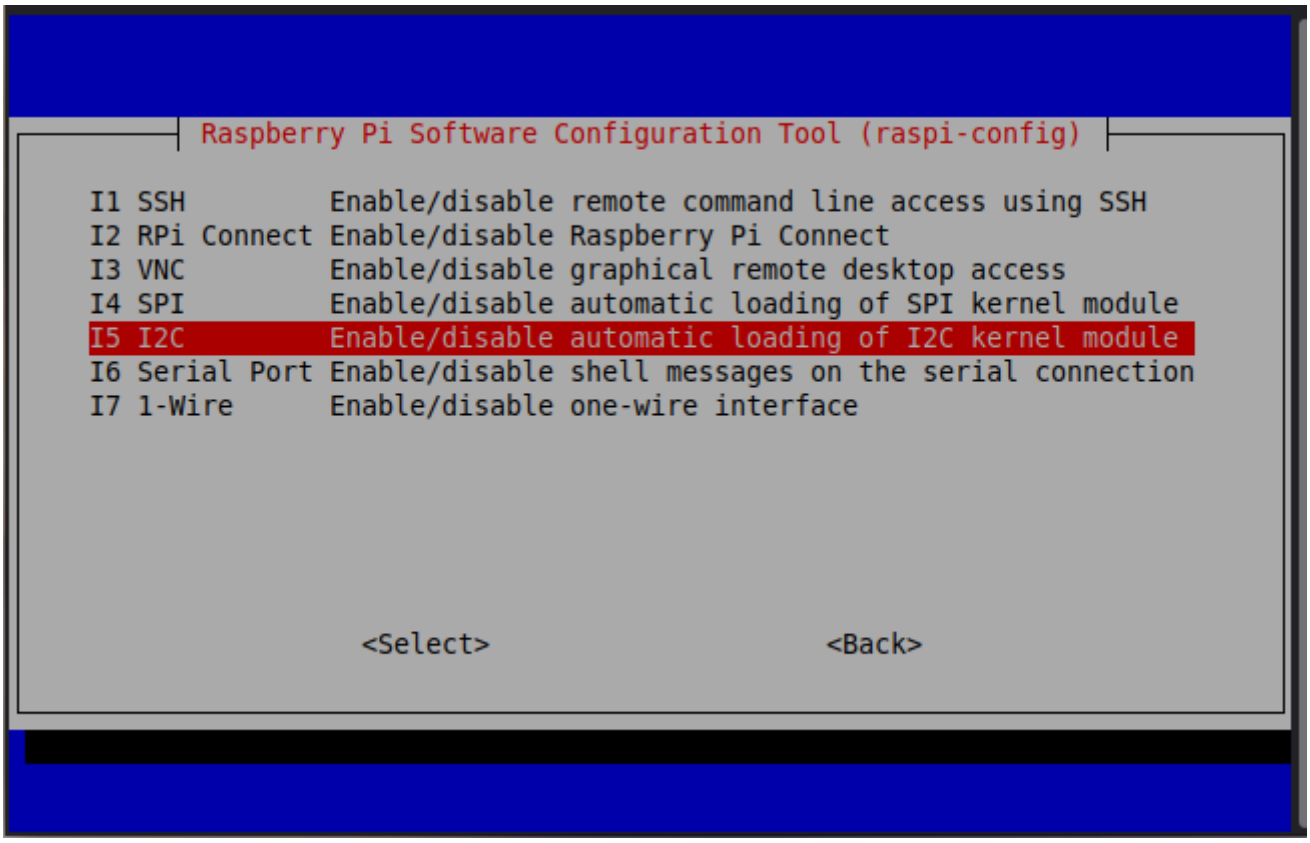
```
cd
sudo apt update && sudo apt upgrade -y
sudo apt install figlet python3-smbus i2c-tools telnet ftp dnsutils
python3-pip bc
figlet APRS | sudo tee /etc/motd
sudo pip3 install --break-system-packages adafruit-circuitpython-bme280
mkdir aprs-wx
```

Aktivieren der I2C Schnittstelle des PI

Dazu wird raspi-config verwendet:

```
sudo raspi-config
```

Hier wird in den *Interface Options* die i2c Schnittstelle aktiviert:



Nun den PI runterfahren:

```
sudo shutdown -h now
```

Jetzt den PI von der Spannungsversorgung trennen!

Anschliesen des Sensors

Der BME280 Sensor braucht nur 4 Kabel:

- VCC
- GND
- SDA
- SCL

Ich habe den wie folgt verkabelt:

Sensor	PI Zero
VCC	GPIO PIN 1
GND	GPIO PIN 6
SCL	GPIO PIN 5
SDA	GPIO PIN 3

Nach dem Verkabeln wird der PI wieder mit Spannung versorgt und startet neu.

Test ob der Sensor erreicht wird

Nachdem man sich wieder per ssh mit dem PI verbunden hat, wird geprüft ob der Sensor erkannt wird und welche ID er hat.

Zuerst wird geprüft ob auch die i2c Schnittstelle initialisiert ist:

```
lsmod | grep i2c_
```

da sollte dann sowas zu sehen sein:

```
i2c_bcm2835            16384  0
i2c_dev                20480  0
```

Nun wird geprüft welche ID der Sensor bekommen hat:

```
i2cdetect -y 1
```

Dabei sollte dann sowas hier rauskommen:

```
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f

00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- 76 --
```

In diesem Beispiel hat der Sensor die ID 76 (haben die fast immer). Diese Nummer nun merken. Brauchen wir für das kleine Python Script. Python Programmierer mögen mir verzeihen, ich bin ein wirklich guter Unix Admin aber ein echt schlechter Programmierer...

Python Script um den Sensor auszulesen

```
#!/usr/bin/python3
# messdaten.py
# Kleines Script um Daten eines BME280 auszulesen
import board
import busio
import time
from adafruit_bme280 import basic as adafruit_bme280

i2c = busio.I2C(board.SCL, board.SDA)
bme280 = adafruit_bme280.Adafruit_BME280_I2C(i2c, address=0x76)

bme280.mode = adafruit_bme280.MODE_NORMAL
time.sleep(1)

# Zum Testen kann man die # entfernen um die Daten "hübscher zu sehen"
#print("Temperature: %0.1f C" % bme280.temperature)
#print("Humidity: %0.1f %%" % bme280.humidity)
#print("relative Humidity: %0.1f %%" % bme280.relative_humidity)
#print("absolute Pressure: %0.1f hPa" % bme280.pressure)
#print("Altitude = %0.2f meters" % bme280.altitude)

# Daten so wie ich sie haben möchte durch ; getrennt, also
# Temperatur;Luftfeuchte;Luftdruck

print(f"{bme280.temperature:.2f}",f"{bme280.humidity:.2f}",f"
{bme280.pressure:.2f}",sep=";")

#EOF
```

Dieses Script am besten im Ordner *aprs-wx* ablegen den wir ja bereits angelegt haben. Danch muss es noch ausführbar gemacht werden und wir können es aufrufen:

```
cd
cd aprs-wx
chmod a+x messdaten.py
./messdaten.py
```

Da sollte dann sowas in der Art rauskommen:

```
24.37;35.71;1011.14
```

Also 24,37 °C, 35,71 % Feuchte und 1011,14 hPa

Bash Script

Dieses Script ruft das messdaten.py Script auf und sendet die Daten an aprsis. Die Unix Leute unter uns schauen jetzt nicht so genau hin. Das Script ist vor ein paar Jahren mal in einer dunklen Nacht und dem Einfluss von Osborne Veterano entstanden.

```
#!/bin/bash
#set -x
#------------------------------------------------------------------
#
# Zieht Daten vom Sensor. Bereitet auf und schiebt sie zu aprs.fi
#
# Stephan Forth, DF6PA      14.04.2021
# Huebsch gemacht               16.04.2021
# Luftdruck Korrektur           17.04.2021
# Führende Null bei Temp    20.04.2021
# Angepasst auf Pi Zero und BME280  24.12.2025
#
#------------------------------------------------------------------

# Variablen deklarieren ---------------------------------------
KORREKTUR=15            # Korrekturfaktor Luftdruck
APRSSERVER=dl.aprs2.net     # Willkürlich aus der Liste der Maschinen
APRSPORT=14580          # Port ist fix
APRSID=MYCALL-13        # ID, die auf der Karte auftaucht
APRSPWD=12345           # Password für aprs

# Ab hier nix ändern :-) --------------------------------------

# Daten aus der vom Sensor holen ------------------------------
# Feld 1 = Temeperatur
# Feld 3 = Luftdruck unkorrigiert
# Feld 2 = Luftfeuchte
#------------------------------------------------------------------

ZEILE=`/home/pi/aprs-wx/messdaten.py`
Temp=`echo $ZEILE| cut -d ";" -f 1`
Druck=`echo $ZEILE| cut -d ";" -f 3`
Feuchte=`echo $ZEILE| cut -d ";" -f 2`
KORDRUCK=`echo "scale=0;$Druck + $KORREKTUR" | bc`

# Zum testen die # entfernen und Ausgabe auf dem Bildschim geniesen
#echo "Temp: $Temp"
#echo "Druck: $Druck"
#echo "Kor. Druck: $KORDRUCK"
#echo "Feuchte: $Feuchte"

# Temp muss als Grad Fahrenheit eingeliefert werden, daher umbauen
Fahrenheit=`echo "scale=0;($Temp * 1.8 +32) / 1" | bc`
```

```
# Luftdruck wird nicht als hectopascal eingeliefert sondern als pascal
OutDruck=`echo "scale=0;($KORDRUCK * 10) / 1" | bc`

# Feuchte ohne Nachkommestellen
OutFeuchte=`echo "scale=0;($Feuchte) / 1" | bc`

# Datensatz zusammenbauen und in Datei schreiben
# Hier noch die Position und den Kommentar anpassen!
echo "user $APRSID pass $APRSPWD vers VERSION THETIS" > /tmp/aprs
echo "$APRSID>APRS,TCPIP*:=4958.61N/00814.38E&Test-Wetter JN49CX (v1.0)" >>
/tmp/aprs

# Wenn Fahrenheit nur 2 Stellen hat, eine führende Null unterbringen
COUNT=`expr length $Fahrenheit`
if [ $COUNT -le 2 ]
then
  echo "$APRSID>APRS,TCPIP*:=4958.61N/00814.38E_.../...g...\
t0$((Fahrenheit))r...p...P...h$((OutFeuchte))b$((OutDruck))" >> /tmp/aprs
else
  echo "$APRSID>APRS,TCPIP*:=4958.61N/00814.38E_.../...g...\
t$((Fahrenheit))r...p...P...h$((OutFeuchte))b$((OutDruck))" >> /tmp/aprs
fi

# Testausgabe
#echo "Druck vom Sensor:      $Druck"
#echo "Druck nach Korrektur:  $KORDRUCK"
#echo "Gesendeter Druck:      $OutDruck"
#echo "Gesendete Feuchte:     $OutFeuchte"
#echo "Temperatur vom Sensor: $Temp"

# Datei mit Werten in RAW TCP Verbindung schieben
# ACHTUNG: Geht nur mit der bash!!
cat /tmp/aprs > /dev/tcp/$APRSSERVER/$APRSPORT

# Aufräumen
rm /tmp/aprs

###EOF###
```

Anlegen einer crontab

Schön ist es jetzt natürlich wenn das Script regelmäßig läuft und alle x Minuten die Daten an aprsis sendet.

Dazu wird mit:

```
crontab -e
```

Eine crontab erzeugt und folgendes eingetragen:

```
# m h  dom mon dow   command
# APRS Karte befuettern
*/23 * * * * /home/pi/aprs-wx/send-wx.sh > /dev/null 2>&1
```

In diesem Fall wird das Script alle 23 Minuten aufgerufen. Bitte verweidet die üblichen *5, 10, 15, 30, 60* Minuten...das machen alle und dementsprechend geht dann im Netz die Post ab.

Damit habt ihr jetzt einen feinen, kleinen Sensor der Daten zu aprsis schiebt. Natürlich sind noch Erweiterungen wie GPS Empfang und bereitstellen eines Zeitservers für das eigene, lokale Netz denkbar. Kommt wenn ich mal wieder Langeweile habe.

Hinweis

Stephan Forth, DF6PA OV Mainz, K07 25.12.2025