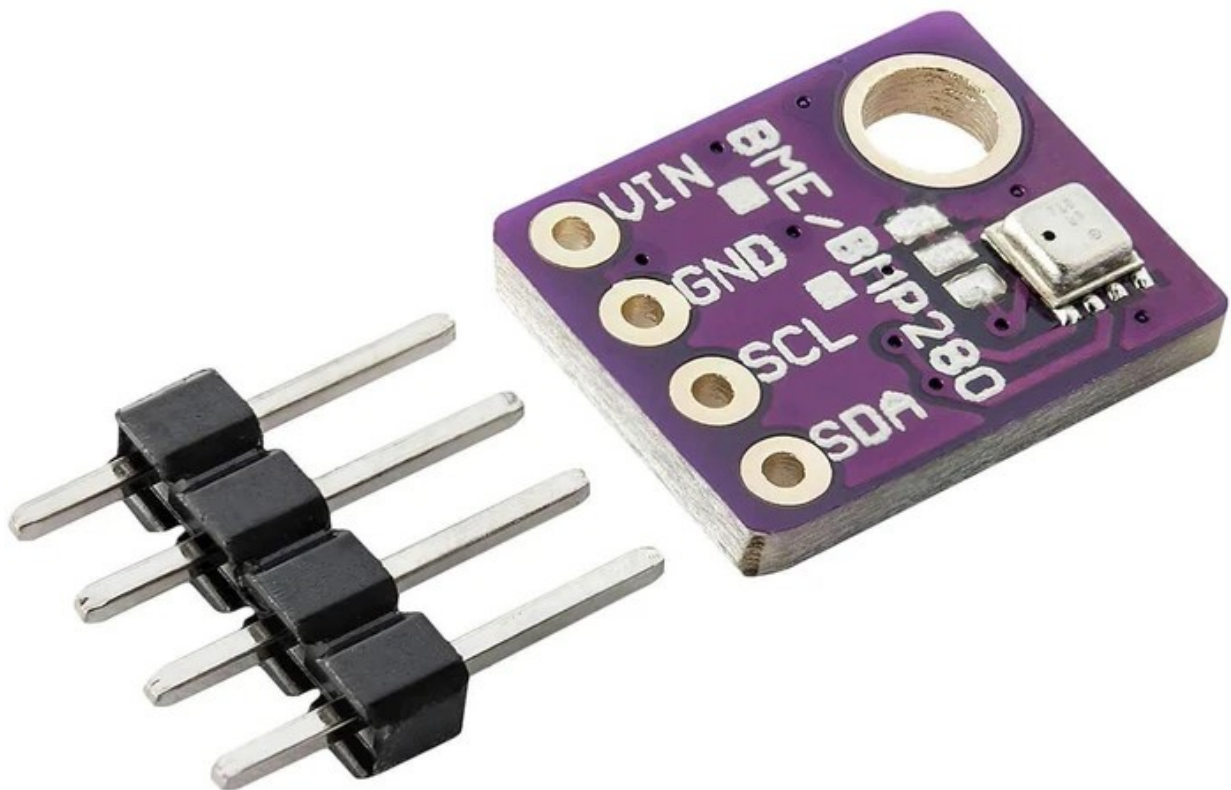


AZ-Delivery

Willkommen

Vielen Dank, dass Sie sich für unseren Temperatur-, Feuchtigkeits- und Luftdrucksensor AZ-Delivery BME280 entschieden haben. Auf den folgenden Seiten erfahren Sie, wie Sie dieses Gerät bedienen und einrichten können.

Viel Spaß!



Az-Delivery

Inhaltsverzeichnis

Einführung.....	3
Spezifikationen.....	4
Einrichten der Arduino IDE.....	5
Einrichten mit Raspberry Pi und Python.....	9
Pinbelegung.....	10
Verbinden des Moduls mit dem Mikrocontroller.....	11
Arduino IDE Bibliothek.....	12
Beispiel Sketch.....	14
Anschließen des Sensors mit Raspberry Pi.....	19
Freigeben der I2C Schnittstelle.....	20
Python Skripte.....	22

Einführung

Der BME280 ist ein digitaler barometrischer Sensor auf einer kleinen Platine. Der Sensor besteht aus Temperatur-, Feuchte- und Drucksensoren. Der BME280-Sensor kann in einer Vielzahl von Anwendungen eingesetzt werden, z. B. in der Hausautomatisierung für Heizung und Klimaanlage, in Geräten zur Gesundheitsüberwachung, in Navigationssystemen, Wetterstationen, Mobilgeräten, IoT und vielen anderen Anwendungen. Die kompakte Bauweise und der geringe Stromverbrauch sind vorteilhaft für Portabilität und batteriebetriebene Geräte. Hohe Genauigkeit und schnelle Reaktionszeit machen ihn zu einem perfekten Kandidaten für die Erweiterung der Funktionalität vieler anderer Geräte der Wahl.

Der BME280 unterstützt die serielle Schnittstelle I2C. Der Sensor hat die vordefinierte I2C-Adresse, 0x76. Die I2C-Adresse kann auf den Wert 0x77 geändert werden, was in diesem eBook nicht behandelt wird.

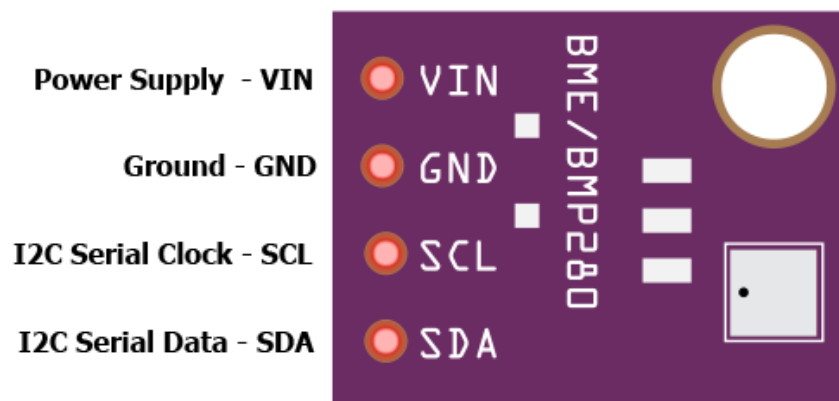
Der Stromverbrauch liegt unter 1 mA, wenn er sich im Messmodus befindet, und 5 μ A, wenn er sich im Leerlaufmodus befindet.

Spezifikationen

- » Betriebs-Spannungsbereich: von 3.3V bis 5V DC
- » Stromverbrauch: < 1mA
- » Temperaturbereich: von -40°C bis 85 °C
- » Temperaturgenauigkeit: $\pm 1.0^{\circ}\text{C}$
- » Druckbereich: von 300 bis 1100 hPa
- » Druckgenauigkeit: $\pm 1\text{hPa}$
- » Luftfeuchtigkeitsbereich: von 0 bis 100% RH
- » Genauigkeit der Feuchte: $\pm 3\%$
- » Abmessungen: 9 x 11 x 2mm

Pinbelegung

Der Sensor BME280 hat vier Pins. Die Pinbelegung ist in der folgenden Abbildung dargestellt:



Der Sensor hat einen on-board LM6206 3,3V Spannungsregler und einen Spannungspegelübersetzer. Die Pins des BME280-Sensors können mit Spannungen im Bereich von 3,3V bis 5V betrieben werden, ohne dass eine Gefahr für den Sensor selbst besteht.

Einrichten der Arduino IDE

Wenn die Arduino IDE nicht installiert ist, folgen Sie dem [link](#) und laden Sie die Installationsdatei für das Betriebssystem Ihrer Wahl herunter.

Download the Arduino IDE



The screenshot shows the Arduino IDE download page. On the left, there is a teal circle containing the Arduino logo (an infinity symbol with a minus sign on the left and a plus sign on the right). To the right of the logo, the text reads: **ARDUINO 1.8.9**. Below this, it says: "The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. It runs on Windows, Mac OS X, and Linux. The environment is written in Java and based on Processing and other open-source software. This software can be used with any Arduino board. Refer to the [Getting Started](#) page for Installation instructions." On the right side of the page, there is a teal sidebar with the following options: **Windows** Installer, for Windows XP and up; **Windows** ZIP file for non admin install; **Windows app** Requires Win 8.1 or 10 with a "Get" button; **Mac OS X** 10.8 Mountain Lion or newer; **Linux** 32 bits; **Linux** 64 bits; **Linux ARM** 32 bits; **Linux ARM** 64 bits; and links for Release Notes, Source Code, and Checksums (sha512).

Für Windows-Benutzer doppelklicken Sie auf die heruntergeladene .exe-Datei und folgen Sie den Anweisungen im Installationsfenster.

Az-Delivery

Für Linux-Benutzer laden Sie eine Datei mit der Erweiterung .tar.xz herunter, die extrahiert werden muss. Wenn sie entpackt ist, gehen Sie in das extrahierte Verzeichnis und öffnen Sie das Terminal in diesem Verzeichnis. Zwei .sh-Skripte müssen ausgeführt werden, das erste heißt arduino-linux-setup.sh und das zweite heißt install.sh.

Um das erste Skript im Terminal auszuführen, öffnen Sie das Terminal im extrahierten Verzeichnis und führen Sie den folgenden Befehl aus:

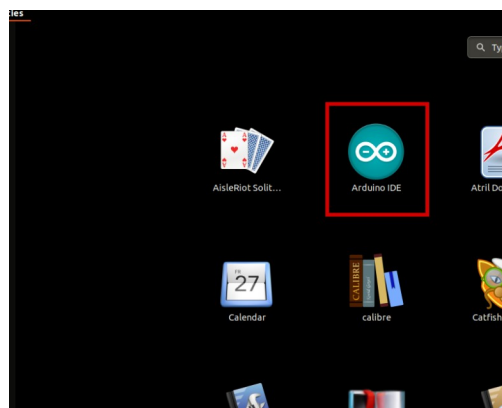
```
sh arduino-linux-setup.sh benutzer_name
```

user_name - ist der Name eines Superusers im Linux-Betriebssystem.

Beim Starten des Befehls muss ein Passwort für den Superuser eingegeben werden. Warten Sie ein paar Minuten, bis das Skript alles abgeschlossen hat.

Das zweite Skript namens install.sh muss nach der Installation des ersten Skripts verwendet werden. Führen Sie den folgenden Befehl im Terminal (entpacktes Verzeichnis) aus: sh install.sh

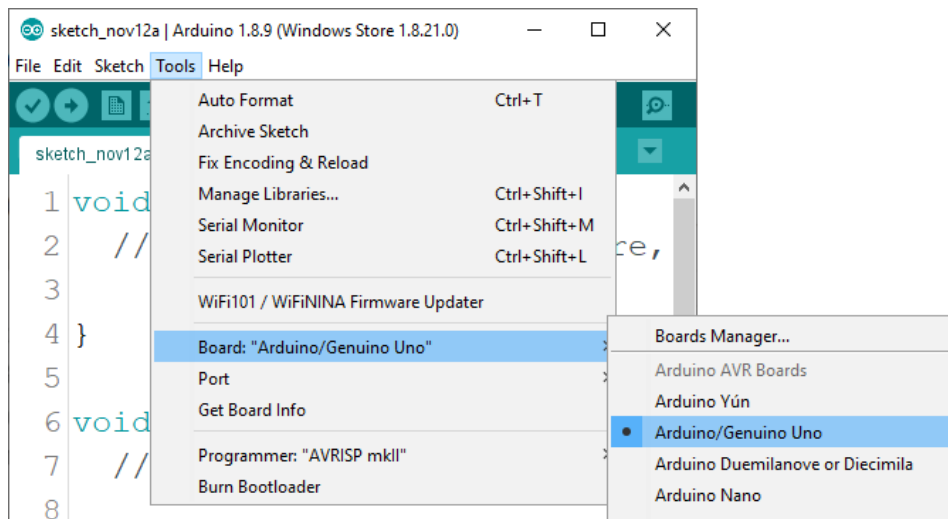
Nach der Installation dieser Skripte gehen Sie zu den All Apps, wo die Arduino IDE installiert ist.



Az-Delivery

Auf fast allen Betriebssystemen ist ein Texteditor vorinstalliert (z. B. Windows mit Notepad, Linux Ubuntu mit Gedit, Linux Raspbian mit Leafpad usw.). Alle diese Texteditoren sind für den Zweck des eBooks vollkommen in Ordnung.

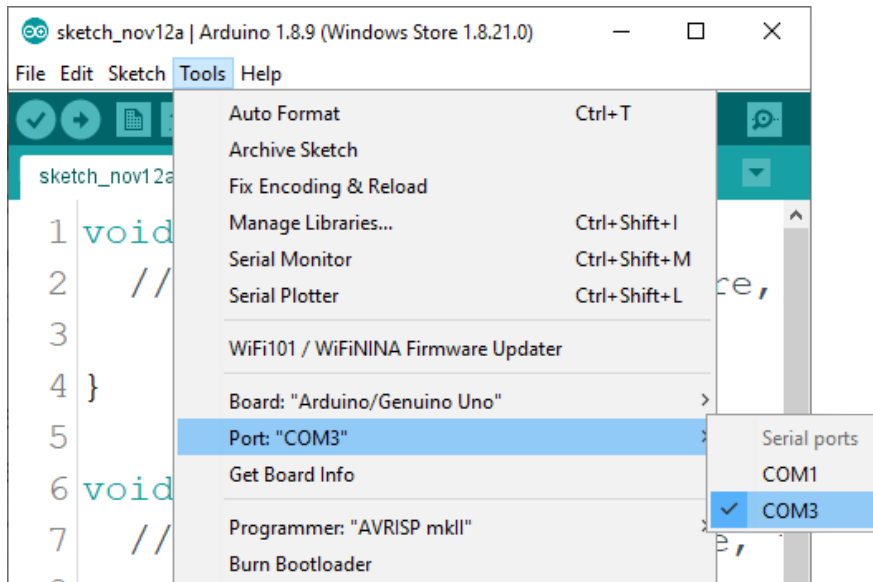
Als nächstes müssen Sie überprüfen, ob Ihr PC ein Arduino-Board erkennen kann. Öffnen Sie die frisch installierte Arduino-IDE, und gehen Sie zu: Werkzeuge > Board > {Ihr Boardname hier}
{Ihr Boardname hier} sollte das Arduino/Genuino Uno sein, wie es auf dem folgenden Bild zu sehen ist:



Der Port, an dem das Arduino-Board angeschlossen ist, muss ausgewählt werden. Gehen Sie zu: Tools > Port > {Portname geht hierhin} und wenn das Arduino-Board mit dem USB-Port verbunden ist, ist der Portname im Dropdown-Menü auf dem vorherigen Bild zu sehen.

Az-Delivery

Wenn die Arduino IDE unter Windows verwendet wird, lauten die Portnamen wie folgt:



Für Linux-Benutzer lautet der Name des Anschlusses beispielsweise /dev/ttyUSBx, wobei x eine ganze Zahl zwischen 0 und 9 darstellt.

Einrichten des Raspberry Pi und Python

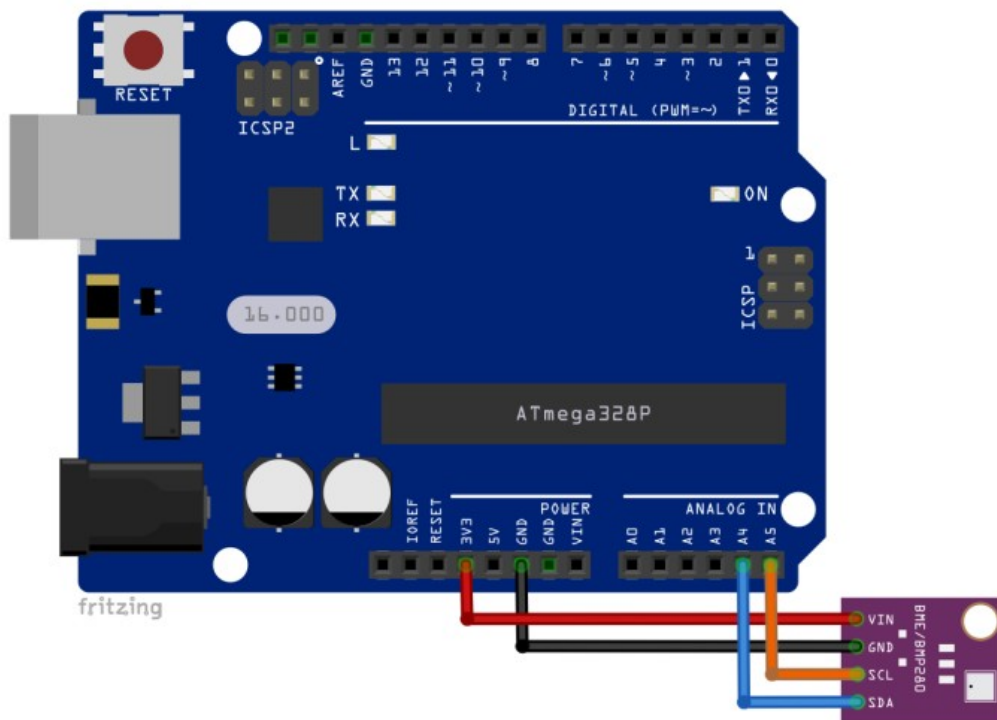
Für den Raspberry Pi muss zunächst das Betriebssystem installiert werden, dann muss alles so eingerichtet werden, dass er im Headless-Modus verwendet werden kann. Der Headless-Modus ermöglicht eine Remote-Verbindung zum Raspberry Pi, ohne dass ein PC-Bildschirm Monitor, Maus oder Tastatur benötigt wird. Die einzigen Dinge, die in diesem Modus verwendet werden, sind der Raspberry Pi selbst, die Stromversorgung und die Internetverbindung. All dies wird im kostenlosen eBook genau erklärt:

[Raspberry Pi Quick Startup Guide](#)

Auf dem Betriebssystem Raspbian ist Python vorinstalliert.

Verbinden des Moduls mit dem Mikrocontroller

Verbinden Sie den Sensor BME280 mit dem Mikrocontroller wie im folgenden Anschlussplan dargestellt:



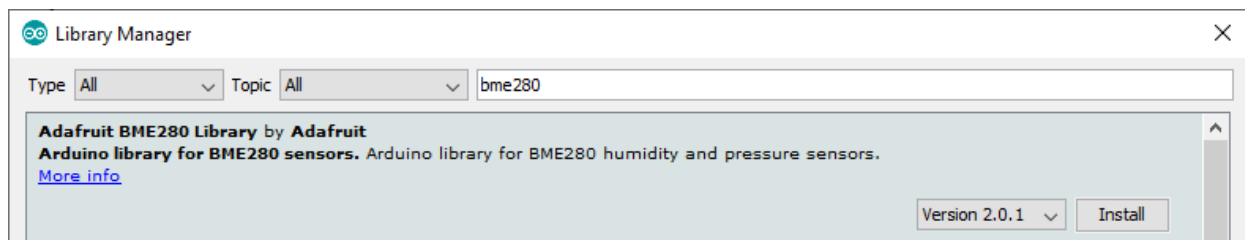
BME280 pin	MC pin	Wiring color
VIN	3.3V	Red wire
GND	GND	Black wire
SCL	A5	Orange wire
SDA	A4	Blue wire

Bibliothek für Arduino IDE

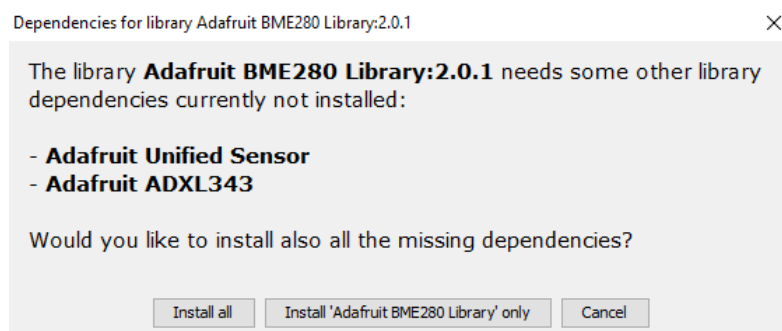
Um den Sensor mit einem Mikrocontroller zu verwenden, empfiehlt es sich, eine externe Bibliothek dafür herunterzuladen. Die Bibliothek, die in diesem eBook verwendet wird, heißt Adafruit BME280. Um sie herunterzuladen und zu installieren, öffnen Sie die Arduino IDE und gehen Sie zu:

Werkzeuge > Bibliotheken verwalten

Wenn sich ein neues Fenster öffnet, geben Sie BME280 in das Suchfeld ein und installieren Sie die Bibliothek namens Adafruit BME280 Library von Adafruit, wie auf dem folgenden Bild gezeigt:



Wenn die Schaltfläche Installieren angeklickt wird, wird die Aufforderung zur Installation einiger zusätzlicher Bibliotheken angezeigt, wie auf dem folgenden Bild:



Az-Delivery

Klicken Sie auf alle installieren, um die Installation der Adafruit BME280-Bibliothek abzuschließen.

AZ-Delivery

Beispiel Sketch

Das folgende Sketch-Beispiel ist ein modifizierter Sketch aus der Adafruit BME280-Bibliothek:

Datei > Beispiele > Adafruit BME280 Library > bme280test

```
#include <Wire.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
Adafruit_BME280 bme; // I2C

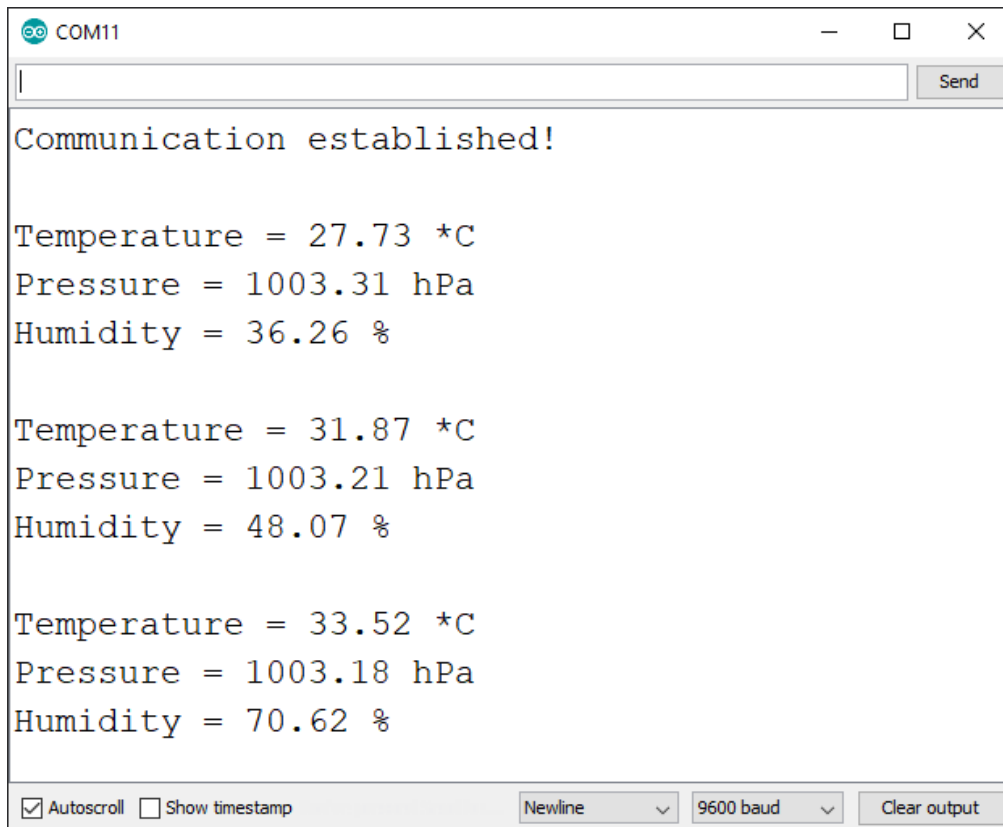
void setup() {
  Serial.begin(9600);
  // default address from library is 0x77
  // bool communication = bme.begin();
  bool communication = bme.begin(0x76);
  if (!communication) {
    Serial.println("Could not find a valid BME280 sensor");
    Serial.println("check wiring, address, sensor ID!");
    Serial.print("SensorID was: 0x");
    Serial.println(bme.sensorID(), 16);
    Serial.println("ID of 0xFF probably means a bad address\n");
    while (true) { };
    delay(10);
  }
  else {
    Serial.println("Communication established!\n");
  }
}
```

Az-Delivery

```
void loop() {  
  Serial.print("Temperature = ");  
  Serial.print(bme.readTemperature());  
  Serial.println(" *C");  
  Serial.print("Pressure = ");  
  Serial.print(bme.readPressure() / 100.0F);  
  Serial.println(" hPa");  
  Serial.print("Humidity = ");  
  Serial.print(bme.readHumidity());  
  Serial.println(" %\n");  
  delay(1000);  
}
```

Az-Delivery

Laden Sie den Sketch auf den Mikrocontroller hoch und öffnen Sie den Serial Monitor (Werkzeuge > Serial Monitor). Das Ergebnis sollte wie die Ausgabe auf dem folgenden Bild aussehen:



The screenshot shows the Serial Monitor window for COM11. The window title is "COM11" and it has standard window controls (minimize, maximize, close). At the top, there is a text input field and a "Send" button. The main area displays the following text:

```
Communication established!  
  
Temperature = 27.73 *C  
Pressure = 1003.31 hPa  
Humidity = 36.26 %  
  
Temperature = 31.87 *C  
Pressure = 1003.21 hPa  
Humidity = 48.07 %  
  
Temperature = 33.52 *C  
Pressure = 1003.18 hPa  
Humidity = 70.62 %
```

At the bottom, there are several controls: a checked "Autoscroll" checkbox, an unchecked "Show timestamp" checkbox, a "Newline" dropdown menu, a "9600 baud" dropdown menu, and a "Clear output" button.

Az-Delivery

Der Sketch beginnt mit der Einbindung von drei Bibliotheken: Wire, Adafruit_Sensor und Adafruit_BME280.

Als nächstes wird das Objekt namens bme mit der folgenden Codezeile erstellt: Adafruit_BME280 bme;

In der Funktion setup() wird die serielle Kommunikation mit der Baudrate von 9600bps gestartet.

Dann wird das bme-Objekt mit der folgenden Codezeile initialisiert:
bme.begin(0x76)
wobei 0x76 die I2C-Adresse des Sensors ist.

Die Funktion begin() gibt einen booleschen Wert zurück, der anzeigt, ob die Initialisierung erfolgreich war oder nicht. Dieser Wert wird in der Variablen mit dem Namen communication gespeichert, mit der folgenden Zeile des Codes:

```
bool communication = bme.begin(0x76);
```

Am Ende der Funktion setup() wird der Erfolg der Initialisierung geprüft. Wenn sie erfolgreich ist, wird im Serial Monitor die Meldung Communication established angezeigt. Wenn die Initialisierung nicht erfolgreich war, werden die Fehlerdaten im Serial Monitor angezeigt.

Az-Delivery

In der Funktion `loop()` werden die Temperatur-, Druck- und Feuchtigkeitsdaten mit den folgenden Codezeilen gelesen:

```
bme.readTemperature()  
bme.readPressure() / 100.0F  
bme.readHumidity()
```

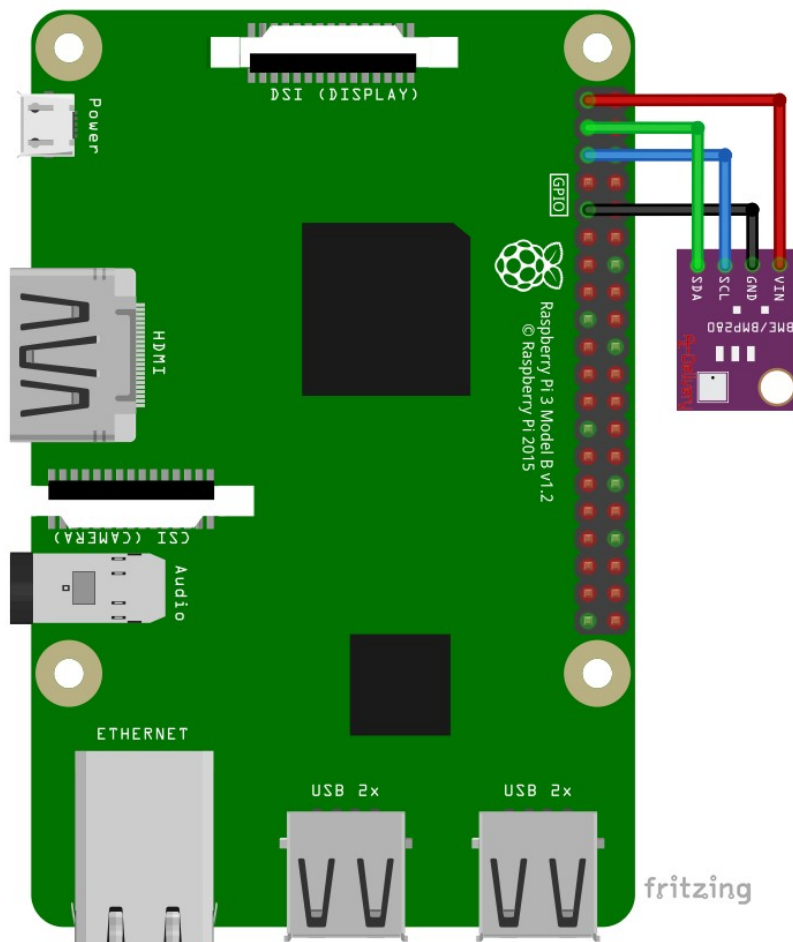
Danach werden die Daten im Serial Monitor mit folgenden Codezeilen angezeigt:

```
Serial.print(bme.readTemperature());  
Serial.print(bme.readPressure() / 100.0F);  
Serial.print(bme.readHumidity());
```

Es gibt eine Pause von einer Sekunde zwischen zwei Messungen am Ende der `loop()` Funktion: `delay(1000);`

Anschließen des Sensors mit Raspberry Pi

Verbinden Sie den BME280-Sensor mit dem Raspberry Pi wie auf dem folgenden Anschlussplan dargestellt:



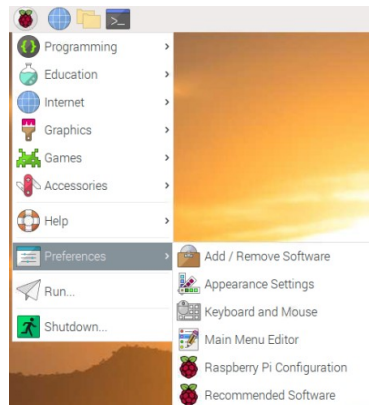
BME280 pin	Raspberry Pi pin	Physical pin No.	Wire color
GND	GND	9	Black wire
VIN	3V3	1	Red wire
SCL	GPIO3	5	Blue wire
SDA	GPIO2	3	Green wire

Az-Delivery

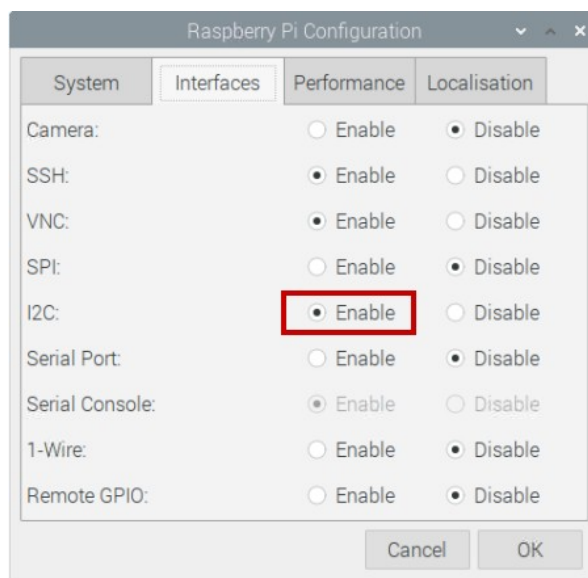
Freigeben der I2C Schnittstelle

Um das Modul mit dem Raspberry Pi verwenden zu können, muss die I2C-Schnittstelle aktiviert werden. Öffnen Sie folgendes Menü:

Application Menu > Preferences > Raspberry Pi Configuration



Aktivieren Sie im neuen Fenster auf der Registerkarte "Interfaces" das Optionsfeld "I2C", wie auf dem folgenden Bild gezeigt:



Az-Delivery

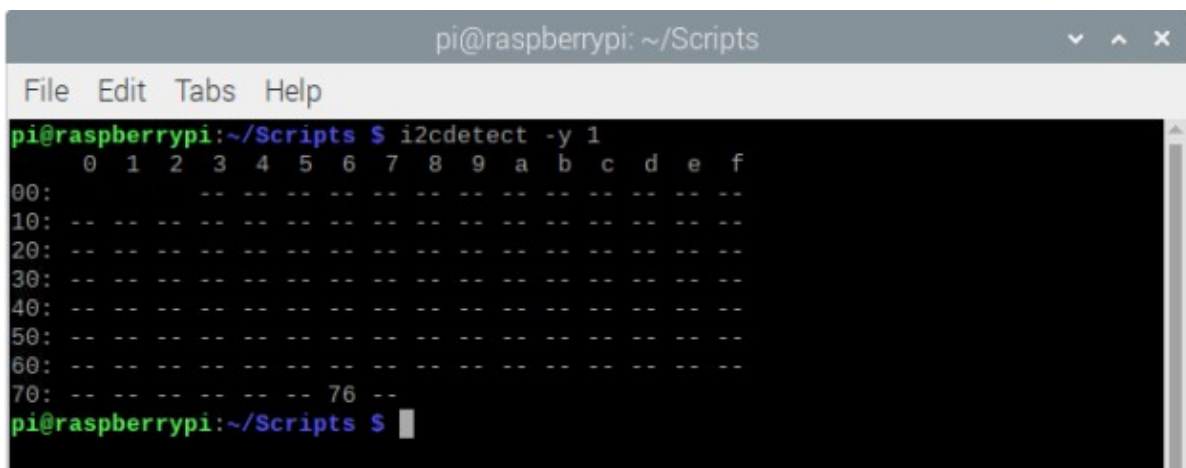
Um die I2C-Adresse des an der I2C-Schnittstelle des Raspberry Pi angeschlossenen Sensors zu ermitteln, muss das Tool `i2c-tools` installiert sein, falls nicht, öffnen Sie das Terminal und führen Sie den folgenden Befehl aus:

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

Um die I2C-Adresse zu ermitteln, öffnen Sie das Terminal und führen Sie den folgenden Befehl aus:

```
i2cdetect -y 1
```

Das Ergebnis sollte wie die Ausgabe auf dem folgenden Bild aussehen:



```
pi@raspberrypi: ~/Scripts
File Edit Tabs Help
pi@raspberrypi:~/Scripts $ 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: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- 76 -- -- -- -- -- -- --
pi@raspberrypi:~/Scripts $
```

Dabei ist `0x76` die I2C-Adresse des Sensors.

Wenn die I2C-Schnittstelle des Raspberry Pi nicht aktiviert ist und der vorherige Befehl ausgeführt wird, wird der folgende Fehler ausgelöst:



```
pi@raspberrypi: ~/Scripts
File Edit Tabs Help
pi@raspberrypi:~/Scripts $ i2cdetect -y 1
Error: Could not open file `/dev/i2c-1' or `/dev/i2c/1': No such file or directory
pi@raspberrypi:~/Scripts $
```

Python Skripte

Es werden zwei Skripte erstellt, eines für alle Funktionen und das andere für die Verwendung dieser Funktionen, wegen der besseren Lesbarkeit. Der Code für das erste Skript sieht wie folgt aus:

```
import smbus
import time
from ctypes import c_short
from ctypes import c_byte
from ctypes import c_ubyte

DEVICE = 0x76 # Default device I2C address
bus = smbus.SMBus(1) # Rev 2 Pi, Pi 2 & Pi 3 uses bus 1
                    # Rev 1 Pi uses bus 0

def getShort(data, index):
    # return two bytes from data as a signed 16-bit value
    return c_short((data[index+1] << 8) + data[index]).value

def getUShort(data, index):
    # return two bytes from data as an unsigned 16-bit value
    return (data[index + 1] << 8) + data[index]

def getChar(data, index):
    # return one byte from data as a signed char
    result = data[index]
    if result > 127:
        result -= 256
    return result
```

Az-Delivery

```
def getUChar(data, index):
    # return one byte from data as an unsigned char
    result = data[index] & 0xFF
    return result

def readBME280ID(addr=DEVICE):
    # Chip ID Register Address
    REG_ID = 0xD0
    (chip_id, chip_version) = bus.read_i2c_block_data(addr, REG_ID, 2)
    return (chip_id, chip_version)

def readBME280All(addr=DEVICE):
    # Register Addresses
    REG_DATA = 0xF7
    REG_CONTROL = 0xF4
    REG_CONFIG = 0xF5
    REG_CONTROL_HUM = 0xF2
    REG_HUM_MSB = 0xFD
    REG_HUM_LSB = 0xFE
    # Oversample setting
    OVERSAMPLE_TEMP = 2
    OVERSAMPLE_PRES = 2
    MODE = 1
    # Oversample setting for humidity register
    OVERSAMPLE_HUM = 2
    bus.write_byte_data(addr, REG_CONTROL_HUM, OVERSAMPLE_HUM)
    control = OVERSAMPLE_TEMP << 5 | OVERSAMPLE_PRES << 2 | MODE
    bus.write_byte_data(addr, REG_CONTROL, control)
    # Read blocks of calibration data from EEPROM
    cal1 = bus.read_i2c_block_data(addr, 0x88, 24)
    cal2 = bus.read_i2c_block_data(addr, 0xA1, 1)
    cal3 = bus.read_i2c_block_data(addr, 0xE1, 7)
```

Az-Delivery

```
# one tab
# Convert byte data to word values
dig_T1 = getUShort(cal1, 0)
dig_T2 = getShort(cal1, 2)
dig_T3 = getShort(cal1, 4)
dig_P1 = getUShort(cal1, 6)
dig_P2 = getShort(cal1, 8)
dig_P3 = getShort(cal1, 10)
dig_P4 = getShort(cal1, 12)
dig_P5 = getShort(cal1, 14)
dig_P6 = getShort(cal1, 16)
dig_P7 = getShort(cal1, 18)
dig_P8 = getShort(cal1, 20)
dig_P9 = getShort(cal1, 22)
dig_H1 = getUChar(cal2, 0)
dig_H2 = getShort(cal3, 0)
dig_H3 = getUChar(cal3, 2)
dig_H4 = getChar(cal3, 3)
dig_H4 = (dig_H4 << 24) >> 20
dig_H4 = dig_H4 | (getChar(cal3, 4) & 0x0F)
dig_H5 = getChar(cal3, 5)
dig_H5 = (dig_H5 << 24) >> 20
dig_H5 = dig_H5 | (getUChar(cal3, 4) >> 4 & 0x0F)
dig_H6 = getChar(cal3, 6)
# Wait in ms (Datasheet Appendix B: Measurement
# time and current calculation)
wait_time = 1.25 + (2.3 * OVERSAMPLE_TEMP) + ((2.3 *
    OVERSAMPLE_PRES) + 0.575) + ((2.3 * OVERSAMPLE_HUM) + 0.575)
time.sleep(wait_time / 1000) # Wait the required time
# Read temperature / pressure / humidity
data = bus.read_i2c_block_data(addr, REG_DATA, 8)
pres_raw = (data[0] << 12) | (data[1] << 4) | (data[2] >> 4)
```


Az-Delivery

```
# one tab
temp_raw = (data[3] << 12) | (data[4] << 4) | (data[5] >> 4)
hum_raw = (data[6] << 8) | data[7]
# Refine temperature
var1 = (((temp_raw >> 3) - (dig_T1 << 1))) * (dig_T2) >> 11
var2 = (((((temp_raw >> 4) - (dig_T1)) * ((temp_raw >> 4) -
          (dig_T1))) >> 12)*(dig_T3)) >> 14
t_fine = var1 + var2
temperature = float(((t_fine * 5) + 128) >> 8);
# Refine pressure and adjust for temperature
var1 = t_fine / 2.0 - 64000.0
var2 = var1 * var1 * dig_P6 / 32768.0
var2 = var2 + var1 * dig_P5 * 2.0
var2 = var2 / 4.0 + dig_P4 * 65536.0
var1 = (dig_P3 * var1 * var1 / 524288.0 + dig_P2 * var1) / 524288.0
var1 = (1.0 + var1 / 32768.0) * dig_P1
if var1 == 0:
    pressure = 0
else:
    pressure = 1048576.0 - pres_raw
    pressure = ((pressure - var2 / 4096.0) * 6250.0) / var1
    var1 = dig_P9 * pressure * pressure / 2147483648.0
    var2 = pressure * dig_P8 / 32768.0
    pressure = pressure + (var1 + var2 + dig_P7) / 16.0

# Refine humidity
humidity = t_fine - 76800.0
humidity = (hum_raw - (dig_H4 * 64.0 + dig_H5 / 16384.0 * humidity))
* (dig_H2 / 65536.0 * (1.0 + dig_H6 / 67108864.0 * humidity * (1.0 +
dig_H3 / 67108864.0 * humidity)))
humidity = humidity * (1.0 - dig_H1 * humidity / 524288.0)
```

Az-Delivery

```
# one tab
if humidity > 100:
    humidity = 100
elif humidity < 0:
    humidity = 0
return temperature / 100.0, pressure / 100.0, humidity
```

Speichern Sie das Skript unter dem Namen bme280.py. Der Skriptcode wird aus dem [script](#) modifiziert.

Az-Delivery

Im Folgenden finden Sie den Code für das Hauptskript:

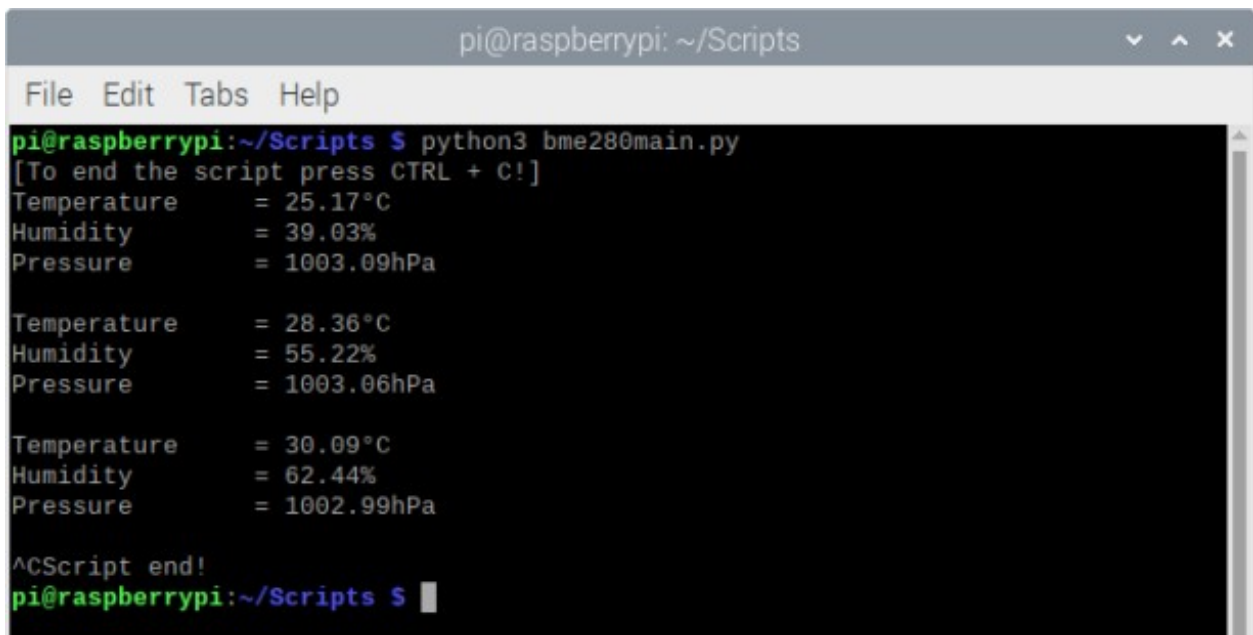
```
import bme280
from time import sleep
dgr = u'\xb0'
print('[Press CTRL + C to end the script!]' )
try:
    while(True):
        temperature,pressure,humidity = bme280.readBME280All()
        print('Temperature = {}{}C'.format(temperature, dgr))
        print('Humidity = {:.2f}%'.format(humidity))
        print('Pressure = {:.2f}hPa\n'.format(pressure))
        sleep(1)

except KeyboardInterrupt:
    print('Script end!')
```

Az-Delivery

Speichern Sie das Skript unter dem Namen `bme280main.py` in das gleiche Verzeichnis, in dem Sie das Skript `bme280.py` gespeichert haben. Um das Hauptskript auszuführen, öffnen Sie das Terminal in dem Verzeichnis, in dem die Skripte gespeichert sind, und führen Sie den folgenden Befehl aus:
python3 bme280main.py

Das Ergebnis sollte wie die Ausgabe auf dem folgenden Bild aussehen:



```
pi@raspberrypi: ~/Scripts
File Edit Tabs Help
pi@raspberrypi:~/Scripts $ python3 bme280main.py
[To end the script press CTRL + C!]
Temperature = 25.17°C
Humidity = 39.03%
Pressure = 1003.09hPa

Temperature = 28.36°C
Humidity = 55.22%
Pressure = 1003.06hPa

Temperature = 30.09°C
Humidity = 62.44%
Pressure = 1002.99hPa

^CScript end!
pi@raspberrypi:~/Scripts $
```

Um das Skript zu stoppen, drücken Sie die Tastenkombination STRG + C auf der Tastatur.

Das erste Skript wird in diesem eBook nicht erklärt.

Az-Delivery

Das Skript bme280main.py beginnt mit dem Import des Skripts bme280 und der Funktion sleep aus der Bibliothek time.

Dann wird die dgr-Variable erstellt, in der der UTF-Grad-Vorzeichenwert gespeichert wird.

Als nächstes wird der try-except-Codeblock erstellt. Im try-Codeblock wird der Endlosschleifenblock (while True:) erstellt. Innerhalb dieses Codeblocks wird die Funktion readBME280All() verwendet, um die Sensordaten zu lesen. Diese Funktion gibt ein Tupel aus drei Elementen zurück: Temperatur-, Druck- und Feuchtigkeitselemente. Anschließend werden die Daten im Terminal angezeigt. In der Ausgabe wird zum Runden der Fließkommazahl auf zwei Nachkommastellen die folgende Codezeile verwendet:

```
print('Humidity = {:.2f}%'.format(humidity))
```

Der except Codeblock wird ausgeführt, wenn STRG + C auf der Tastatur gedrückt wird. Dies wird als KeyboardInterrupt bezeichnet. Wenn dieser Blockcode ausgeführt wird, wird im Terminal die Meldung Script end! angezeigt.

Az-Delivery

Jetzt ist es an der Zeit, zu lernen und selbst Projekte zu erstellen. Das können Sie mit Hilfe von vielen Beispielskripten und anderen Tutorials tun, die Sie im Internet finden können.

Wenn Sie auf der Suche nach den qualitativ hochwertigen Produkten für Arduino und Raspberry Pi sind, sind Sie bei der AZ-Delivery Vertriebs GmbH genau richtig. Sie erhalten zahlreiche Anwendungsbeispiele, vollständige Installationsanleitungen, eBooks, Bibliotheken und Unterstützung durch unsere technischen Experten.

<https://az-delivery.de>

Viel Spaß!

Impressum

<https://az-delivery.de/pages/about-us>