Kernel- und Treiberprogrammierung – Folge 117

Schatzsuche

Das Industrial-IO-Subsystem bringt umfangreichen Treiberquellcode für unterschiedlichste Sensorik mit. Wir zeigen anhand der populären Sensoren BME280 und HC-SR04, wie Sie den Schatz heben. Eva-Katharina Kunst, Jürgen Quade

Klar, bei Linux kann man auch alles selbst machen. Das erfordert allerdings Know-how, kostet Zeit und ist fehleranfällig. Da erscheint es schon praktischer, auf Bestehendes und Funktionierendes zurückzugreifen. Durch das offene Linux-Ökosystem liegt die Wahrscheinlichkeit hoch, dass es irgendwo da draußen bereits die Lösung für das eigene IT-Problem gibt. Fragt sich nur: wo?

Wer sich in Sachen Treiberquellcode auf eine Google-Suche verlässt, setzt schnell aufs falsche Pferd, denn unprofessionellen Bastelcode gibt es an jeder Internet-Ecke, und die im Quellcode hinterlegten professionellen Treiber sind im WWW nur unzureichend referenziert. Der Blick in die Kernel-Quellen ist in jedem Fall sinnvoll.

Liegen die Sourcen bereits auf der eigenen Maschine, hilft bei der Suche nach verdächtigen Dateien ein unterhalb von `drivers/` gestartetes `find`, das Sie mit ein paar markanten Zeichen des Sensornamens parametrieren. Weitaus strukturierter geht es im Unterverzeichnis `Documentation/devicetree/bindings/` zu. Bindings beschreiben die Konfiguration einer Hardware. Die Namen der hier abgelegten Dateien bestehen aus dem Namen des Herstellers, durch ein Komma getrennt vom Hardwaretyp. Sie sind brav entsprechend ihrer Technologie, ihrer Systemankopplung oder ihrem Sensortyp in weitere Unterordner sortiert.

Für Geräte, für die es einen Industrial-IO-Treiber (siehe Kasten Industrial IO) gibt, existiert ein separates Unterverzeichnis (`iio/`). Dort tummeln sich unter anderem chemische Sensoren sowie solche, die Entfernung, Druck, Feuchtigkeit oder Lichtintensität messen. Inhaltlich beschreiben die Bindings allerdings nur die Konfiguration, nicht die Hardware selbst. Informationen über die zugehörigen Gerätetreiber und deren Schnittstellen finden sich hier ärgerlicherweise nicht.

Finden statt suchen

Haben Sie den Treiber über die Suche unterhalb des Verzeichnisses `drivers/` über die Bindings oder letztlich mit einer Volltextsuche gefunden, heißt das noch nicht, dass Sie sofort loslegen können. Tatsächlich gibt es gerade im Industrial-IO-Subsystem sehr umfangreichen Treiberquellcode, den standardmäßig weder Ubuntu noch im Fall des Raspberry Pi – Pi OS vorkonfiguriert ausliefert. Programmieren müssen Sie also zwar nicht, Hand anlegen aber schon.

Als einen von ganz wenigen IIO-Treibern liefert Pi OS den für den BME280 als Modul mit. Sie verbinden das von Bosch stammende Messequipment per I²C oder SPI mit einem Steuerrechner, beispielsweise einem Raspberry Pi. Der Hersteller hat die Adresse am I²C-Bus fest als `0x77` oder `0x76` vorgegeben. Richtig angesteuert, liefert der Sensor die relative Luftfeuchtigkeit, den Luftdruck und die Umgebungstemperatur.

Der BME280 ist, wie die Bezeichnung schon vermuten lässt, eng verwandt mit dem BMP280. Er erweitert Letzteren um die Messung der Luftfeuchtigkeit. Tatsächlich nutzt er auch den Treiber für den BMP280, der den Vorgängersensor BMP180, aber eben auch die Erweiterung BME280 unterstützt. Allerdings liegt diese Information in den Tiefen der Kernel-

Konfiguration und der Bindings-Datei `bmp085.yaml` verborgen. Es bleibt somit dem eigenen Fingerspitzengefühl überlassen, den Treiber zu finden.

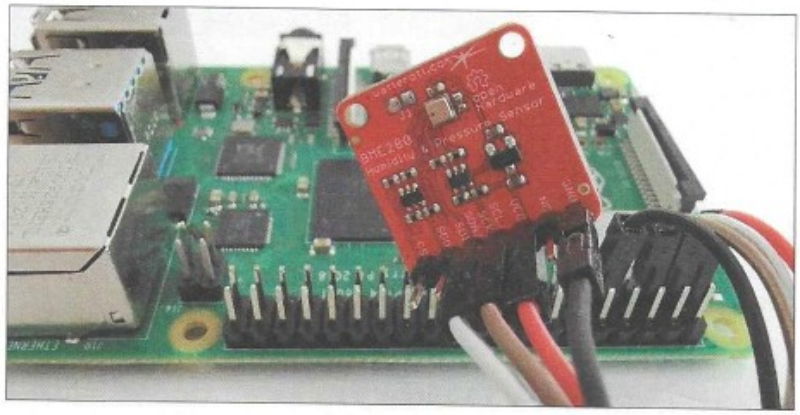
Voraussetzungen schaffen

Um auf dem RasPi 4 I²C-Hardware verwenden zu können, müssen Sie die Module `i2c_dev` und `i2c_bcm2835` laden – entweder über die Kommandozeile oder besser über das Tool `raspi_config`.

In Raspi_config wählen Sie unter *[Interface Options]* den Punkt *[I²C]* aus. Nach der Aktivierung lädt der RasPi beim Booten über die Datei `/boot/config.txt` das zum Nutzen von I²C notwendige Overlay `i2c-arm`. Außerdem entsteht in `/etc/modules` der Eintrag `i2c-dev`, sodass das entsprechende Modul automatisiert seinen Weg in den Kernel findet.

Darüber hinaus steht noch die Installation der I²C-Werkzeuge an. Wie gewohnt geht das bequem auf der Konsole über den Aufruf `sudo apt install i2c-tools`.

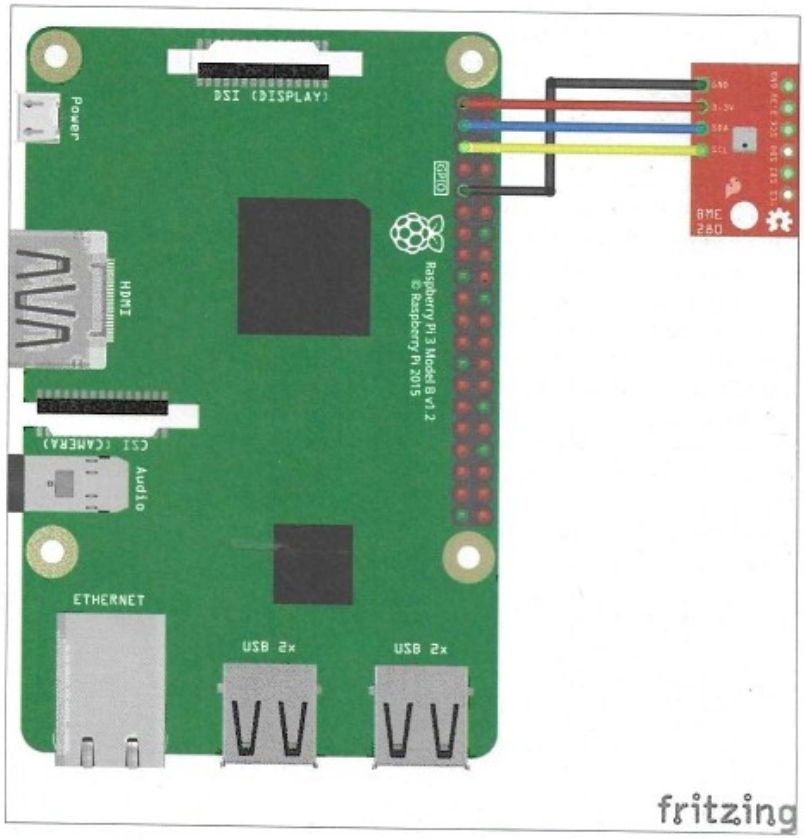
Hardwaretechnisch wird der BME280 über zwei Signalleitungen plus Strom und Masse mit dem Raspberry Pi verbunden.



1 Der Bosch-Sensor BME280 misst Luftdruck, Luftfeuchte und Temperatur.

Die Spannungsversorgung (Strom) erfolgt zum Beispiel über Pin 1 der Steckerleiste, Masse findet sich an Pin 9. Die beiden Signalleitungen *SCL* und *SDA* verbinden Sie mit Pin 3 und Pin 5 2.

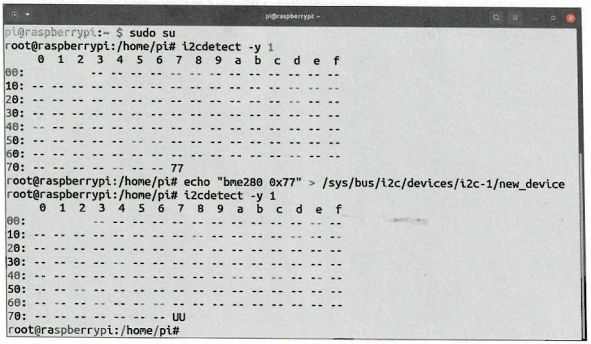
Haben Sie den Sensor korrekt mit dem Raspberry Pi gekoppelt, zeigt das Kommando `i2cdetect -y 1` ein Gerät an der Adresse `0x77` an (Abbildung 3, oben). Da der Kernel keine Ahnung hat, um welches Gerät es sich handelt, leisten wir Schützenhilfe und ordnen den Sensor der Adresse mithilfe des Kommandos aus Listing 1 zu (Abbildung 3, Mitte).



2 Der Schaltplan für den BME280 am Raspberry Pi (Verschaltung von vorn).

Industrial IO

Als Industrial IO bezeichnet man ein im Linux-Kernel integriertes Sensor-Interface, das einen professionellen Zugang zu Daten ermöglicht. Professionell bedeutet in diesem Zusammenhang unter anderem, dass die Sensorwerte standardisiert sind, beispielsweise bezüglich der Auflösung und Repräsentation (Ganzzahl, Float). Außerdem lassen sich, sofern das der jeweilige Treiber unterstützt, Sensorwerte automatisiert inklusive Auftrittszeitpunkt erfassen. Der Zugriff auf die Daten kann über das Sys-Filesystem erfolgen. Alternativ gibt es mit der Libiio eine Bibliothek, die Anschluss an die gängigen Programmiersprachen bietet. Mit den Libiio-Utils kommt eine Handvoll Werkzeuge hinzu, die den Zugriff bequemer gestalten. Mittlerweile unterstützt Industrial IO mehrere Hundert unterschiedliche Sensoren und zusätzlich auch noch einige Aktoren. Wir haben Industrial IO als solches sowie die Treiberunterstützung dafür in den letzten Folgen dieser Serie bereits thematisiert.



3 Den I²C-Sensor machen Sie Linux per Echo bekannt.

I2cdetect ändert jetzt die 77 in ein UU (Abbildung 3, unten). Die Bekanntgabe der I²C-Adresse genügt, damit Linux bereits alle benötigten Treiber automatisiert lädt: das Grundmodul `bmp280`, den zugehörigen I²C-Interface-Treiber `bmp280_i2c` und `industrialio`. Im Sys-Filesystem ist das neue IIO-Gerät angelegt worden und steht für den Zugriff bereit. Ein kurzer Blick ins Verzeichnis `/sys/bus/iio/devices/iio:device0/` zeigt eine ganze Reihe von Dateien (siehe Tabelle „Dateien in `iio:device0` (Auswahl)").

Ein Zugriff per `cat in_temp_input` liefert beispielsweise die Temperatur in Grad Celsius bei einer Auflösung in Milligrad Celsius zurück. `cat in_pressure_input` ergibt den Luftdruck in Kilopascal, `cat in_humidityrelative_input` die relative Luftfeuchtigkeit in Milliprozent.

Da der Treiber für den BME280 bereits vorkompiliert vorliegt und die Bekanntgabe der Sensoradresse über das Sys-Filesystem erfolgen kann, lässt sich der Sensor ohne großen Aufwand direkt nutzen.

Es geht komplizierter

Anders sieht es aus, wenn man den Gerätetreiber noch kompilieren muss und die Sensorkonfiguration komplexer ausfällt. Die dann notwendigen Schritte lassen sich anhand des weitverbreiteten Sensors HC-SR04 4 verdeutlichen.

Der Quellcode des passenden Treibers bildet zwar einen Teil des Linux-Kernels, ist aber standardmäßig in der Konfiguration abgewählt. Zur Generierung des Gerätetreibers bieten sich zwei Wege an. Der erste ist zeitaufwendiger, aber dafür etwas unkomplizierter. Der zweite führt schnell zum Kompilat, bedarf aber mehr Kenntnissen und auch mehr Handarbeit.

Für den ersten Weg holen Sie den zugehörigen Kernel-Quellcode per `apt install rpi-source` und nachfolgendem `rpi-source` auf den Raspberry Pi. Listing 2 zeigt alle dazu notwendigen Schritte. Rpi-source lädt den zur aktuellen Kernel-Version passenden Quellcode auf den Raspberry Pi. Es legt ihn jedoch nicht unter dem Standardpfad `/usr/src/linux/` ab, sondern unter `/root/linux/`.

Ein `make menuconfig` öffnet die Kernel-Konfiguration, in der Sie den Treiber aktivieren. Im Fall des HC-SR04 findet er sich unterhalb von *[Device Drivers] | [Industrial I/O support] | [Proximity and distance sensors] | [GPIO bitbanged ultrasonic ranger sensor (SRF04, MB1000)]*. Nach der Auswahl verlassen Sie das Konfigurationswerkzeug über *[Finish]*, wobei die Konfiguration in der Datei `.config` landet.

Das nachfolgende `make modules` dauert auf einem Raspberry Pi 4 über eine Stunde; hier ist also Geduld angebracht. Das `make modules_install` dagegen geht schnell von der Hand und installiert den Treiber an der korrekten Stelle.

Für Eilige

Für den zweiten, zeitlich kürzeren Weg, installieren Sie die passenden Kernel-Header, kopieren den Treiberquellcode in ein separates Verzeichnis, nehmen ein passendes Makefile und rufen `make` auf. Das dann hoffentlich fehlerfrei generierte Kernel-Objekt laden Sie anschließend per `insmod srf04.ko`.

Listing 1: Adresszuordnung

```
$ echo "bme280 0x77" > /sys/bus/i2c/devices/i2c-1/new_device
```

Dateien in `iio:device0` (Auswahl)

Datei	enthält
in_humidityrelative_input	skalierte Feuchtigkeit in Milliprozent
in_humidityrelative_oversampling_ratio	Anzahl der Luftfeuchtemessungen, über die für einen Feuchtewert gemittelt wird
in_pressure_input	Druck in Kilopascal
in_pressure_oversampling_ratio	Anzahl der Druckmessungen, über die für einen Druckwert gemittelt wird
in_temp_input	Temperatur in Milligrad Celsius
in_temp_oversampling_ratio	Anzahl der Temperaturmessungen, über die für einen Messwert gemittelt wird

Die Autoren

Eva-Katharina Kunst ist schon seit den Anfängen von Linux Fan von Open Source. **Jürgen Quade**, Professor an der Hochschule Niederrhein, bietet auch für Unternehmen Schulungen zu den Themen Treiberprogrammierung und Embedded Linux an.

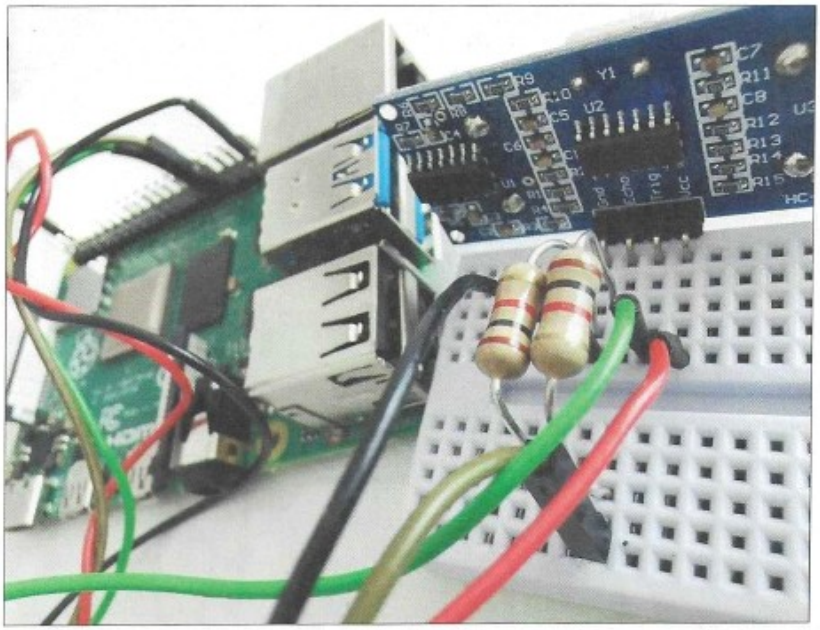
Da der Anschluss des Ultraschallsensors über frei wählbare GPIOs erfolgt und nicht über ein (serielles) Bussystem, hat Entwickler Andreas Klinger als Basis folgerichtig einen Plattformgerätetreiber gewählt. Dem Kernel müssen Sie allerdings sowohl die Existenz des für ihn nicht automatisiert erkennbaren Sensors als auch die Wahl der GPIOs bekanntgeben.

Devicetree selbst gemacht

Der einfachste Weg dazu – insbesondere, ohne Code schreiben zu müssen –, besteht darin, die Hardware per Devicetree zu beschreiben. Eine passende Devicetree-Beschreibung sehen Sie in Listing 3. Der Vorteil dieses Vorgehens: Die Konfiguration und Instanziierung einer Peripherie über den Devicetree sind sehr flexibel. Der Nachteil: Es erfordert Systemkenntnisse, um einen Devicetree-Eintrag zu erstellen. Eine erste Hilfestellung für das Erzeugen des hier benötigten Devicetree-Eintrags leistet die Kernel-Dokumentation mit den Devicetree-Bindings.

Stammt der Gerätetreiber aus der Hand eines einigermaßen versierten Entwicklers, findet sich für die zugehörige Hardware ein Dokument, das deren konfigurierbare Elemente, alle verfügbaren Optionen sowie die eigentliche Funktionalität beschreibt. Das Dokument im YAML-Format listet die unbedingt notwendigen Einstellungen auf und gibt klassischerweise ein Beispiel. Insbesondere die Angaben unter dem Tag `required:` sind für den Anwender relevant, ein Beispiel am Ende der Datei unter `examples:` hilft idealerweise beim Verständnis. Übrigens entspricht der Name der unterhalb von `iio/proximity/` zu findenden Bindings-Datei `devantech-srf04.yaml` für den HC-SR04 nicht exakt der eingangs erwähnten Namenskonvention: Statt des Kommas steht ein Bindestrich im Dateinamen.

Zur Integration ins System genügt das Kopieren der Beispielzeilen für den Devicetree-Code aus der Bindings-Datei noch nicht. Zum einen gilt es, einen Kopf (Listing 3, Zeilen 1 und 2) zu ergänzen. Zum anderen müssen Sie die Integration in den baumartig strukturierten Devicetree definieren. Hier kommt es insbesondere auf die Variable `target_path` an, die das Plattformgerät auf der obersten Ebene einhängt, quasi der Wurzelebene (/).



4 Mit dem HC-SR04 lassen sich Entfernungen messen. Hier die Verschaltung von hinten.

Den Sensor selbst schließen Sie über vier Leitungen an 5. Neben Stromversorgung (*5V*) und Erde (*GND*) benötigen Sie je eine Leitung für den Trigger und das Echo-Signal. Im vorliegenden Beispiel kommen dazu *GPIO5* (Echo) und *GPIO6* (Trigger) zum Einsatz. Verschalten Sie den Sensor über andere GPIOs, müssen Sie lediglich die entsprechenden Einträge in der Datei `srf04.dts` anpassen.

Die Schaltung erfordert einen Spannungsteiler, da der Sensor selbst mit 5 Volt betrieben wird und auch ein 5-Volt-Signal für Echo erzeugt. Der Raspberry Pi verwendet jedoch 3,3-Volt-Signale; ein 5-Volt-Eingangssignal würde ihn überfordern und eventuell zerstören. Da er aber alles zwischen 1,7 und 3,3 Volt an seinem Eingang als ein `true` interpretiert, trifft die durch den Teiler halbierte Spannung die goldene Mitte. Beachten Sie bitte, dass der Schaltplan den Sensor von vorn darstellt, Abbildung 4 die Verschaltung aber von hinten zeigt, sodass dort die Leitungen spiegelverkehrt erscheinen.

Das Kommando `dtc srf04.dts -o srf04.dtbo` übersetzt den in `srf04.dts` gespeicherten Devicetree-Quellcode (Listing 3) in den Blob `srf04.dtbo`, der sich per `dtcoverlay srf04.dtbo` schließlich in den Kernel injizieren lässt. Den Erfolg der Aktion vollziehen Sie bei Bedarf über das Verzeichnis `/sys/firmware/devicetree/base/` nach: Dort gibt es das neue Verzeichnis `proximity/`, das die im Overlay abgelegten Informationen in Form mehrerer Dateien sichtbar macht.

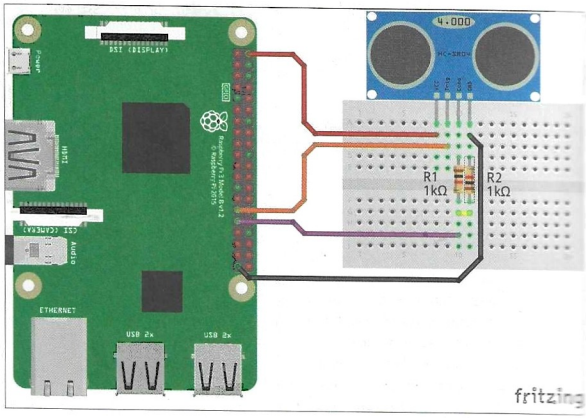
Beim Laden des Overlays erkennt der Kernel das neue Gerät, und RasPi OS lädt automatisiert die notwendigen Treiber.

Listing 2: Treiber für den HC-SR04 generieren

```
# apt install rpi-source
libncurses-dev
# rpi-source
# cd /root/linux/
# make menuconfig
### Menüpunkte:
### [Device Drivers] |
[Industrial I/O support] |
### [Proximity and distance
sensors] |
### [GPIO bitbanged ultrasonic
ranger sensor (SRF04, MB1000)] |
### [Finish]
# make modules
# make modules_install
```

Dass Linux das neue Device erkannt und den Treiber geladen hat, erkennen Sie am Inhalt des Verzeichnisses `/sys/bus/iio/devices/iio:device0/`. Wechseln Sie dorthin, um per Shell auf die Messdaten zuzugreifen. Statt der Null in `iio:device0` kann bei Ihnen auch eine höhere Zahl stehen, falls bereits andere Geräte über das Industrial-IO-Subsystem bedient werden.

Der HC-SR04-Sensor stellt nur einen Kanal zur Verfügung, `in_distance_raw`. Die Skalierung lässt sich über `in_distance_scale` ablesen und entspricht demnach einer Angabe in Millimetern. Der programmgesteuerte Zugriff kann über die Libiio erfolgen, die Sie über `apt install libiio0` installieren. Bei dieser Gelegenheit sollten Sie auch die gleich das Paket *libiio-utils* miteinrichten, sodass Sie per `iio_info` einen ausführlichen Überblick über die zur Verfügung stehenden Geräte und deren Attribute erhalten 6.



5 **Der Schaltplan für das Anbinden eines HC-SR04 am Raspberry Pi.**

Schon beim Booten

Kopieren Sie den Devicetree-Blob `srf04.dtbo` in das Verzeichnis `/boot/overlays/` des Raspberry Pi und ergänzen die Datei `/boot/config.txt` um den Eintrag `dtoverlay=srf04`, dann steht das IIO-Gerät direkt beim nächsten Reboot zugriffsfertig zur Verfügung. Auch der BME280-Sensor lässt sich übrigens anstelle des Echo-Befehls per Devicetree-Overlay konfigurieren und nach dem beschriebenen Schema beim Booten aktivieren.

Listing 3: `srf04.dts`

```
01 /dts-v1/;
02 /plugin/;
03
04 / {
05   compatible = "brcm,bcm2835",
   "brcm,bcm2711";
06
07   fragment@0 {
08     target-path = "/";
09
10       __overlay__ {
11           proximity {
12             compatible =
   "devantech,srf04";
13             status = "okay";
14             trig-gpios =
   < &gpio 5 0 >;
15             echo-gpios =
   < &gpio 6 0 >;
16           };
17       };
18   };
19 };
```

Wie Sie gesehen haben, erfordert es also etwas Aufwand, einem vom Kernel unterstützten Sensor frische Daten zu entlocken. Das ist der Preis, den es für den professionellen Code zu zahlen gilt, sieht man einmal von der eingangs erwähnten Quellcodesuche ab. Aber wie schon der Name des Kommandos Find andeutet: Bei Linux sucht man nicht, man findet. (*jlu*)

```
root@raspberrypi:/proc/device-tree/soc/gpio@7e200000# iio_info
Library version: 0.16 (git tag: v0.16)
Compiled with backends: local xml ip usb serial
IIO context created with local backend.
Backend version: 0.16 (git tag: v0.16)
Backend description string: Linux raspberrypi 5.10.17-v7l+ #1414 SMP Fri Apr 30 13:20
:47 BST 2021 armv7l
IIO context has 1 attributes:
        local,kernel: 5.10.17-v7l+
IIO context has 2 devices:
        iio:device0: bme280
                3 channels found:
                        humidityrelative:  (input)
                        2 channel-specific attributes found:
                                attr  0: input value: 53224
                                attr  1: oversampling_ratio value: 16
                        pressure:  (input)
                        2 channel-specific attributes found:
                                attr  0: input value: 101.437328125
                                attr  1: oversampling_ratio value: 16
                        temp:  (input)
                        2 channel-specific attributes found:
                                attr  0: input value: 27190
                                attr  1: oversampling_ratio value: 2
        iio:device1: srf04
                1 channels found:
                        distance:  (input)
                        2 channel-specific attributes found:
                                attr  0: raw value: 1401
                                attr  1: scale value: 0.001000
root@raspberrypi:/proc/device-tree/soc/gpio@7e200000#
```

6 **Ein Aufruf von `iio_info` gibt einen ersten Überblick über die Geräte und Attribute.**