Industrial-IO unter Linux

Einbindung von Sensoren und Aktoren mit Industrial-IO in Linux

Andreas Klinger, IT-Klinger

Seit 2011 gibt es im Linux-Kernel das Industrial-Input-Output-Subsystem, kurz IIO. Inzwischen wurden von der recht eifrigen Community rund um die Mailingliste linux-iio beinahe 248 IIO-Treiber (stable v4.13, ohne Variationen und Staging) mainline gebracht. Beispiele sind AD- und DA-Wandler, Beschleunigungssensoren, Licht-, Feuchte-, Luftdruck-, Temperaturmessung, usw.
Was sind die Besonderheiten von IIO-Treibern und wie kann ich diese in meinem Projekt verwenden? Genau darüber handelt dieser Artikel.
Damit es nicht zu theoretisch sondern anschaulich wird, wurde als Anschauungsobjekt ein konkretes Projekt in etwas vereinfachter Darstellung gewählt.

1 Was ist Industrial-IO in Linux?

Das Industrial-IO-Subsystem im Linux-Kernel dient dazu, den Zugriff auf Sensoren und Aktoren zu vereinheitlichen. Bei klassischen Character Devices wird für jedes Gerät ein eigener Treiber mit seinen eigenen Funktionen und seiner oft recht individuellen Schnittstelle zum Userspace (z. B. ioctl(), sysfs-Attribute, Verwendung von Timern) implementiert.
Dies bedeutet oftmals, dass der Zugriff auf unterschiedliche Treiber eigens implementiert werden muss. Selbst bei Treibern der gleichen Art, wie beispielsweise Temperatursensoren ist es unwahrscheinlich, dass zwei Treiber das gleiche Interface anbieten. Wird nun ein Sensor durch einen anderen ersetzt ist die Anwendung an den neuen Treiber anzupassen.
Hier setzt Industrial-IO an mit dem Ziel sowohl die Schnittstelle für Treiber, welche Daten liefern als auch für die Verwendung der Daten im Userspace zu standardisieren.

1.1 Anwendungsbeispiel: Bienenwaage

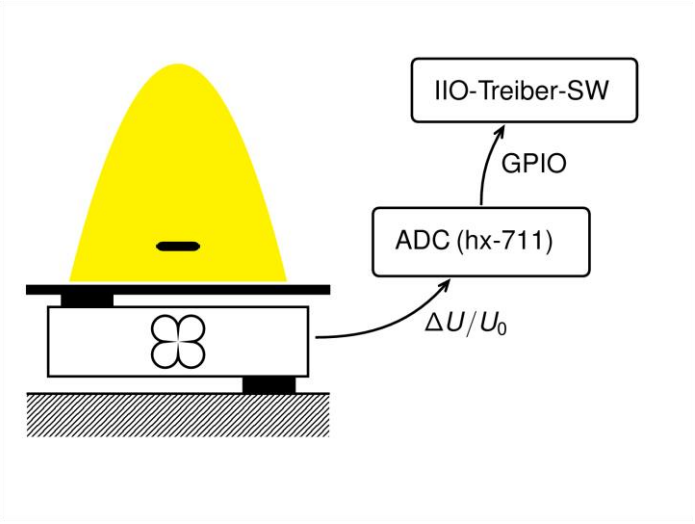

Abb. 1: Gewichtsmessung mit AD-Wandler

In dem für dieses Thema gewählten Projekt soll das Gewicht von Bienenstöcken gemessen werden. Dafür gibt es mehrere Use-Cases. Die Masse wird erfasst, um folgende Fragen zu klären:

- Messung des Futterverbrauches (Honig) bei der Überwinterung mit Erkennung von hungernden Völkern
- Verfolgung des Nektareintrages und damit des angesammelten Honigs, um ihn zu schleudern
- Erkennung von Schwärmen als natürliche Volksteilung mit der Möglichkeit diese rechtzeitig einzufangen und ein neues Volk zu gründen Umweltdaten werden am Bienenstand erfaßt mit der Zielsetzung:
- Umgebungstemperatur über 12° C führt zu nennenswerter Flugaktivität
- bei Abfall des Luftdrucks sind die Bienen oftmals stechlustiger

Die Skizze in Abbildung 1 erläutert den Informationsfluss von der Wägezelle, auf welcher der Bienenstock steht über den AD-Wandler (hx711) bis zum IIO-Sensor im Linux-Kernel, angebunden mittels GPIO-Bitbanging.
Temperatur, Luftdruck und Luftfeuchte können mit dem Sensor BME280 gemessen werden, wie in Abbildung 2 dargestellt. Dieser wiederum lässt sich mittels I2C oder SPI ansprechen und durch einen IIO-Sensor auswerten.

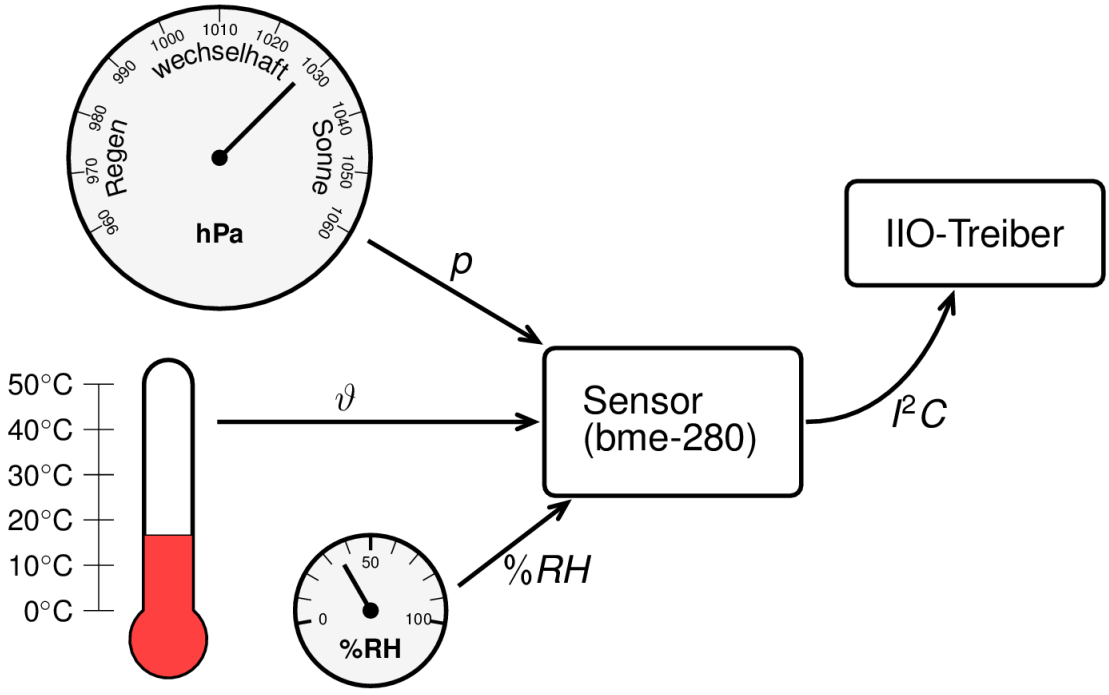

Abb. 2: Messung von Temperatur, Luftdruck und Luftfeuchtigkeit mit IIO-Sensor

1.2 Hardwareschnittstellen

Direkt durch das Framework unterstützt werden

- I 2 C
- SPI
- GPIO

Für diese Schnittstellen ist die Implementierung von Treibern besonders einfach, da eine ganze Reihe an Hilfsfunktionen zur Verfügung steht. Andere Hardwareschnittstellen sind nicht ausgeschlossen und können ebenso genutzt werden. Gegebenenfalls ist etwas mehr an Implementierung vonnöten.

Ganz ohne Telegrammverkehr kann der Datenaustausch auch mittels DMA erfolgen. Hierbei werden die Daten vom Gerät direkt in einen Speicherbereich geschrieben, ohne dass ein expliziter Telegrammverkehr notwendig ist.

Die Benachrichtigung über geschriebene Daten erfolgt dann mit einem Interrupt.
Die Beziehungen sowohl nach unten zur Hardware, als auch nach oben zum Userspace sind in Abbildung 3 beispielhaft dargestellt.

1.3 Schnittstellen zum Userspace

Aus dem Userspace heraus können die Sensoren über ein vereinheitlichtes Interface abgefragt und eingestellt werden. Dazu existiert ein eigenes Bussystem namens iio.

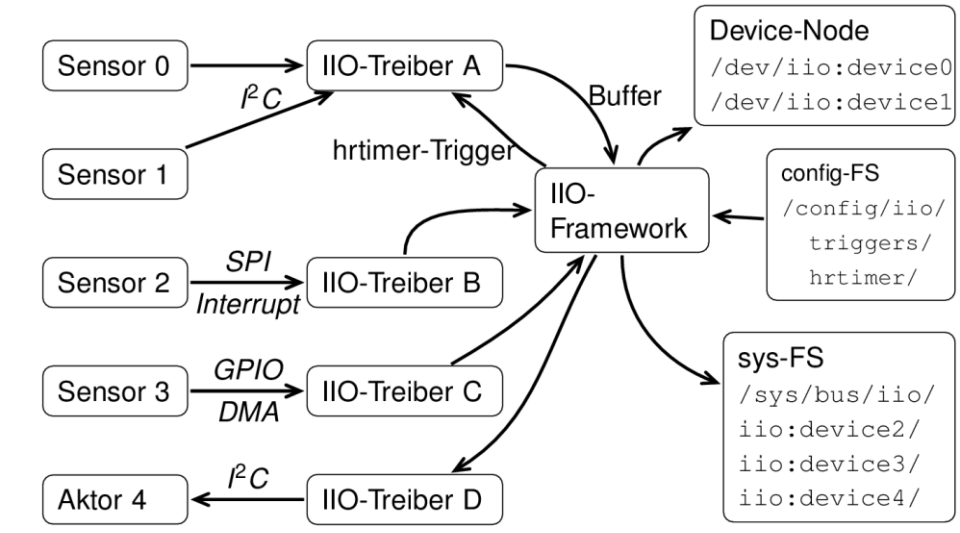

Abb. 3: Beziehungen des IIO-Frameworks zu den Treibern und zum Userspace

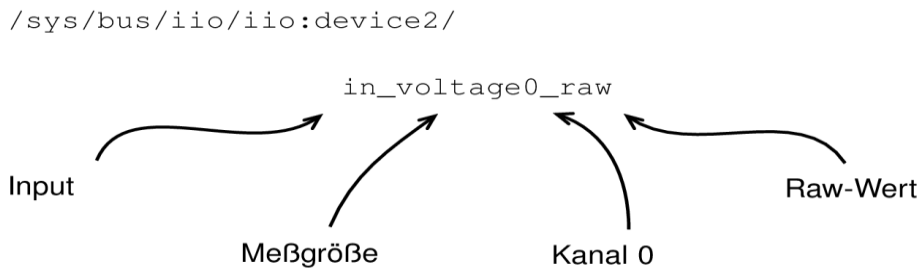

Abb. 4: Erläuterung der Dateinamenskonventionen an einem Beispiel

Dementsprechend sind die registrierten Devices in Unterverzeichnissen von /sys/bus/iio/devices auffindbar.

In Abbildung 4 ist ein Beispiel für einen eingelesenen Spannungswert eines AD-Wandlers zu sehen. Die einzelnen Namensbestandteile werden systematisch zusammengebaut und sind daher gut durch einen Algorithmus verwendbar.
Mit dem Tool lsiio aus dem Verzeichnis tools/iio/ der Kernelsourcen kann abgefragt werden, welche IIO-Devices erkannt wurden und welche Nummer diese bekommen haben:

root@waage: lsiio -v

Device 000: bme280
in_temp_input
in_humidityrelative_input
in_pressure_input

Device 001: hx711
in_voltage0_raw
in_voltage1_raw

In diesem Beispiel existieren zwei IIO-Devices. Das IIO-Device 000 implementiert den Temperatur-, Luftdruck- und Luftfeuchtesensor BME280 mit seinen drei Messgrößen.
Der AD-Wandler hx711 liefert den eingelesenen Rohwert beider verfügbarer Kanäle jeweils als Spannungswert. Die Umrechnung in das Gewicht erfolgt durch die Userspace-Anwendung.

1.4 Triggerung durch hrtimer-Event

Das IIO-Framework bietet Unterstützung für das Einlesen von Sensordaten beim Auftreten eines Ereignisses, wie zum Beispiel eines Interrupts. Die dann eingelesenen Werte können in einen Speicherbereich eingestellt und asynchron mit einem Device-Node abgefragt werden.

Die am häufigsten eingesetzte Triggerung ist der hrtimer-Event. Hierbei wird der Hardware-Timer durch das hrtimer-Framework auf einen Zeitpunkt in der Zukunft programmiert. Tritt der Zeitpunkt ein, liefert der Hardware-Timer einen Interrupt, welcher als hrtimer-Event zur Verfügung steht. Eine Callback-Funktion im Industrial-IO-Treiber wird aufgerufen und der Sensor abgefragt.

Die Daten können in einen Puffer eingestellt werden. Dieser muss im Treiber eingerichtet werden und dient der Zwischenspeicherung der Daten, bis diese mithilfe des Device-Nodes (/dev/iio:device<N>) abgefragt werden. Das IIO-Framework kümmert sich darum, dass der Timer erneut programmiert wird. Dadurch entsteht ein zyklischer Timer mit konstanter Abtastfrequenz.
Für die Abfrage der Daten existiert im Verzeichnis tools/iio der Linux-Sourcen ein Hilfsprogramm namens iio_generic_buffer. Diesem wird mitgeteilt, von welchem Device mit welchem Trigger welche Daten wie häufig abgefragt werden.

Nachfolgendes Beispiel zeigt, wie man hrtimer-Events aus dem Userspace heraus verwendet:

```
root@silo: cd /sys/kernel/config/iio/triggers/hrtimer
root@silo: mkdir mytmr
root@silo: cd /sys/bus/iio/devices/trigger0
root@silo: echo 1 > sampling_frequency
root@silo: iio_generic_buffer -a -c3 -N0 -T0

[...]

/sys/bus/iio/devices/iio:device0 mytmr

0.080000 1502965687594511120
0.230000 1502965688594510720
0.240000 1502965689594510840
```

Zunächst wird der hrtimer-Trigger generiert. Dies erfolgt im configfs durch Anlegen eines neuen Verzeichnisses. Als Name dafür wurde im Beispiel mytmr gewählt. Handelt es sich um den ersten angelegten Trigger, dann hat dieser die Nummer 0.

Im sysfs ist ein neues Device-Verzeichnis mit dem Namen trigger<T> entstanden, wobei T die Nummer des Triggers ist. In diesem Verzeichnis kann der Trigger parametrisiert werden. In obigem Beispiel wird die Frequenz auf 1 Hz eingestellt.

Die Datenabfrage erfolgt mit dem Programm iio_generic_buffer. Es werden alle Kanäle (-a) 3 mal (-c3) vom Device mit der Nummer 0 (-N0) unter Verwendung des Triggers mit der Nummer 0 (-T0) abgefragt.

1.5 Userspace-Libraries

Das IIO-Subsystem ist, wie oben ausgeführt sehr systematisch nach bekannten und dokumierten Regeln aufgebaut. Daß dies so bleibt, darüber wachen die Reviewer und der Maintainer. Bevor neue Definitionen verwendet werden, wird versucht, neue Treiber in die bestehenden Interfaces zu integrieren.

Am Framework angemeldete IIO-Geräte werden in der Reihenfolge der Anmeldung durchnummeriert. Dies erfordert vom Userspace-Entwickler, dass er den Zusammenhang zwischen Device-Nummer und dem gewünschten Treiber herstellen kann. Er muss die Devices im sysfs durchlaufen und anhand des Attributes name den betreffenden Treiber erkennen. Analog dazu funktioniert die Erkennung von Trigger-Events.
Dies ist mühsame Routine und kann aufgrund der Systematik von IIO gut auf eine Library ausgelagert werden. Die libiio ist genau dafür erschaffen worden. Mit ihr kann gut automatisiert werden:

- Verbindung zu lokalem oder entferntem IIO-Subsystem, auf welchem der iiod-Dämon läuft
- Abfrage der vorhanden Devices
- Kanäle und Attribute durchiterieren
- Lesen und Schreiben von Attributen

2 Verweise

Sourcen der im Artikel verwendeten Treiber:

- linux-stable/drivers/iio/adc/hx711.c
- linux-stable/drivers/iio/pressure/bmp280*.c

Autor

Andreas Klinger ist selbständiger Trainer und Entwickler. Seit Abschluss des Studiums der Elektrotechnik im Jahre 1998 arbeitet er im Bereich der systemnahen Softwareentwicklung mit den Schwerpunkten Treiberentwicklung, Embedded Linux und Echtzeit. Als Spezialist für Linux beschäftigt er sich mit dem internen Aufbau des Kernels, den Systemmechanismen sowie vor allem mit deren Einsatz in Embedded Systemen.

Kontakt

Internet: http://www.it-klinger.de
E-Mail: ak@it-klinger.de