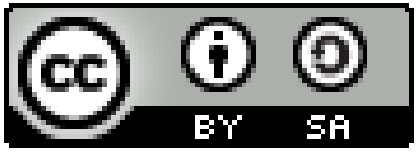
Industrial IO

Andreas Klinger

IT - Klinger
http://www.it-klinger.de
ak@it-klinger.de

OSADL COOL
30.04.2025



Author:
Andreas Klinger <ak@it-klinger.de>

Industrial IO — background information

- Abstraction of sensor and actor interface for userspace
- Sensors provide measured values from the environment
- Implementation as Linux kernel driver
- Fast data collection
- Use asynchronous and buffered with device node,
  synchronous in sysfs or
  triggered waiting for event

Use case: Bee scales

Measuring the weight of beehive using the scales

- Measuring food consumption (honey)
  → Is the bee colony hungry?
- Measure honey in store
  → When can the beekeeper extract honey?
- Identify bee swarm
  → Bee colony splits and sets forth
- Raid
  → Bee colony breaks into another bee colony and steals their honey

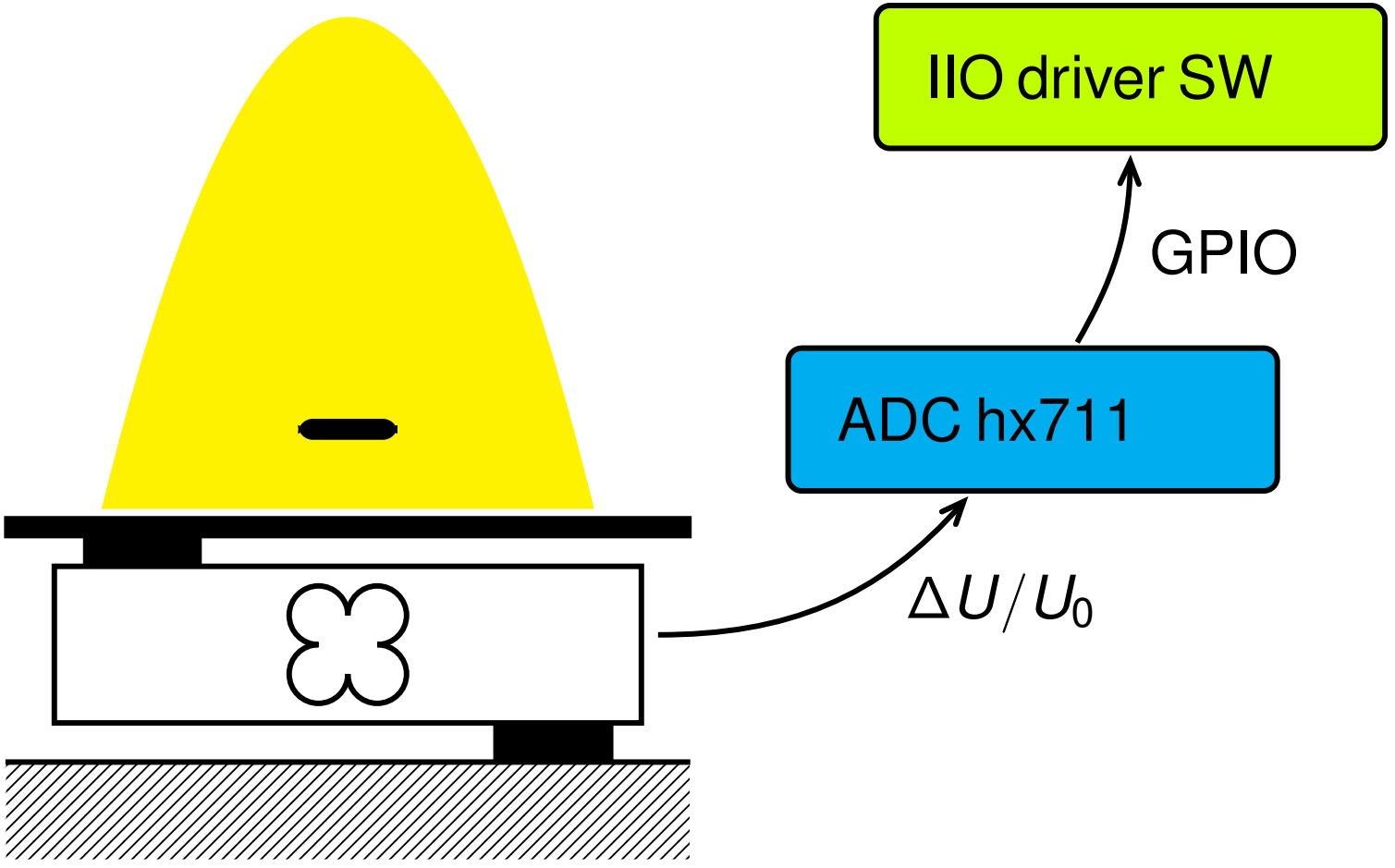
Measuring the temperature

→ Flight activity starting at 12°C

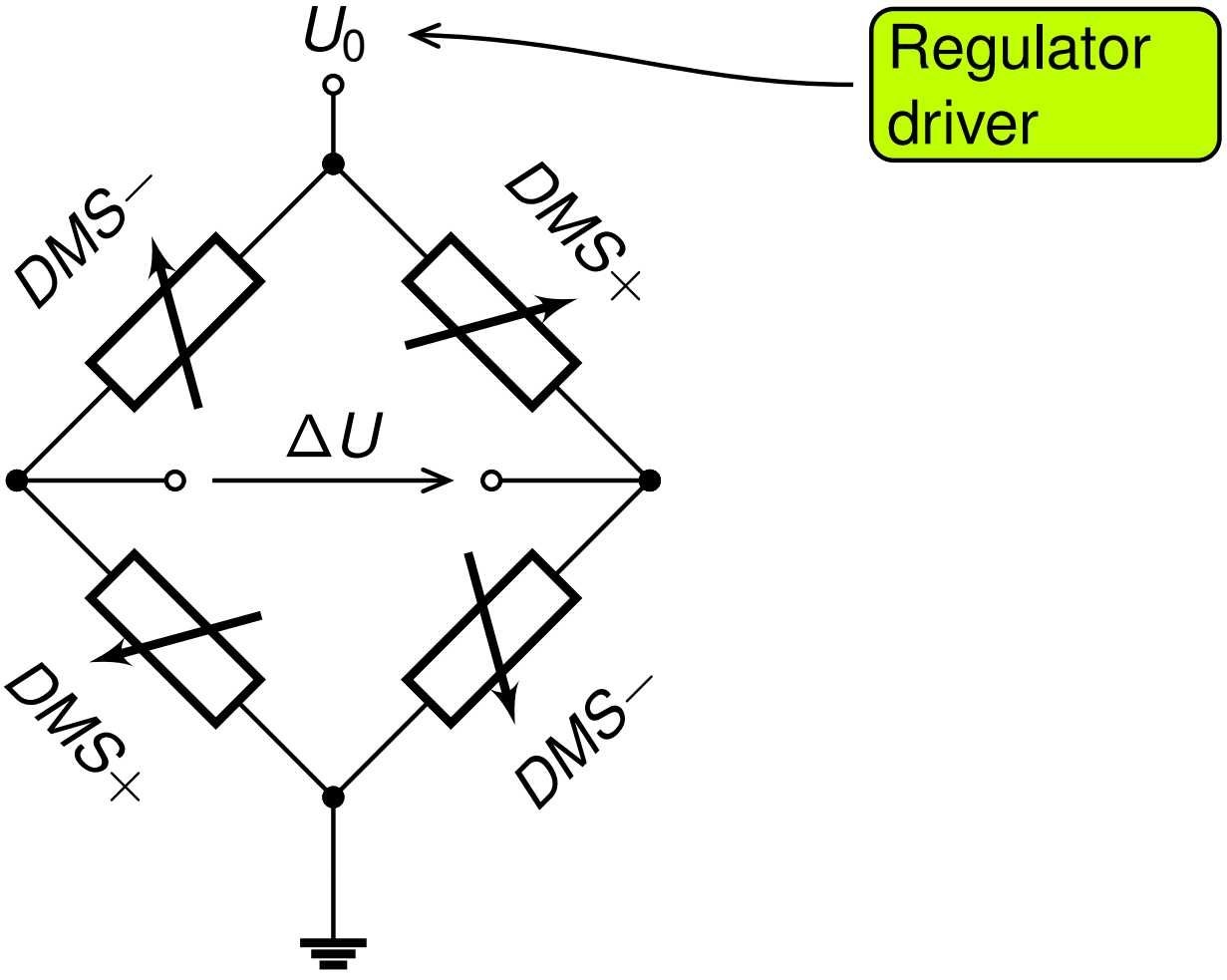
Measuring the air pressure

→ Bees are aggressive when there is a risk of thunderstorms (drop in air pressure)

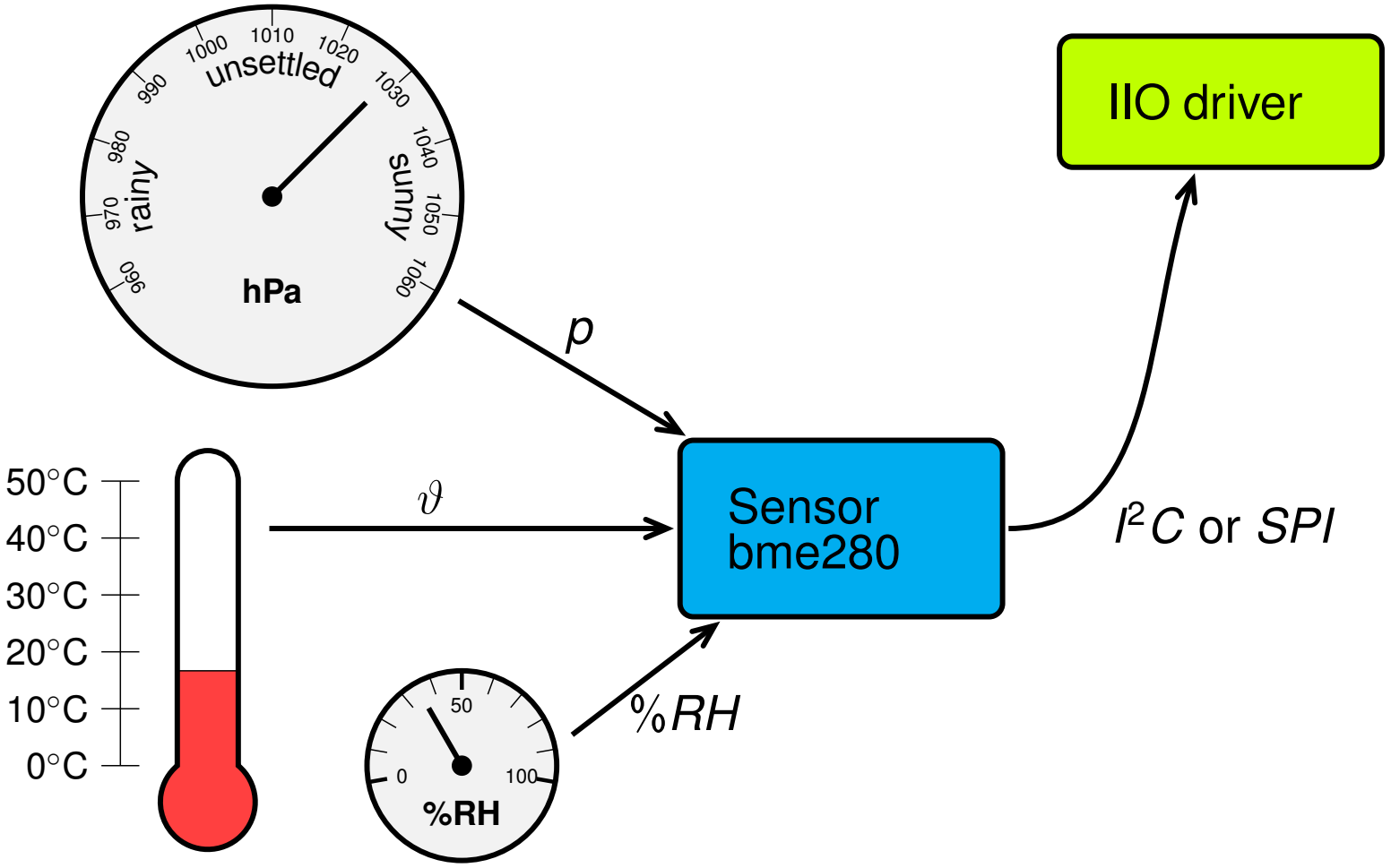
Example: Bee scales with IIO sensors



Weighing cell



Environmental data measurement with IIO sensors



Operation modes of industrial IO

direct mode

Read synchronously from sysfs attribute

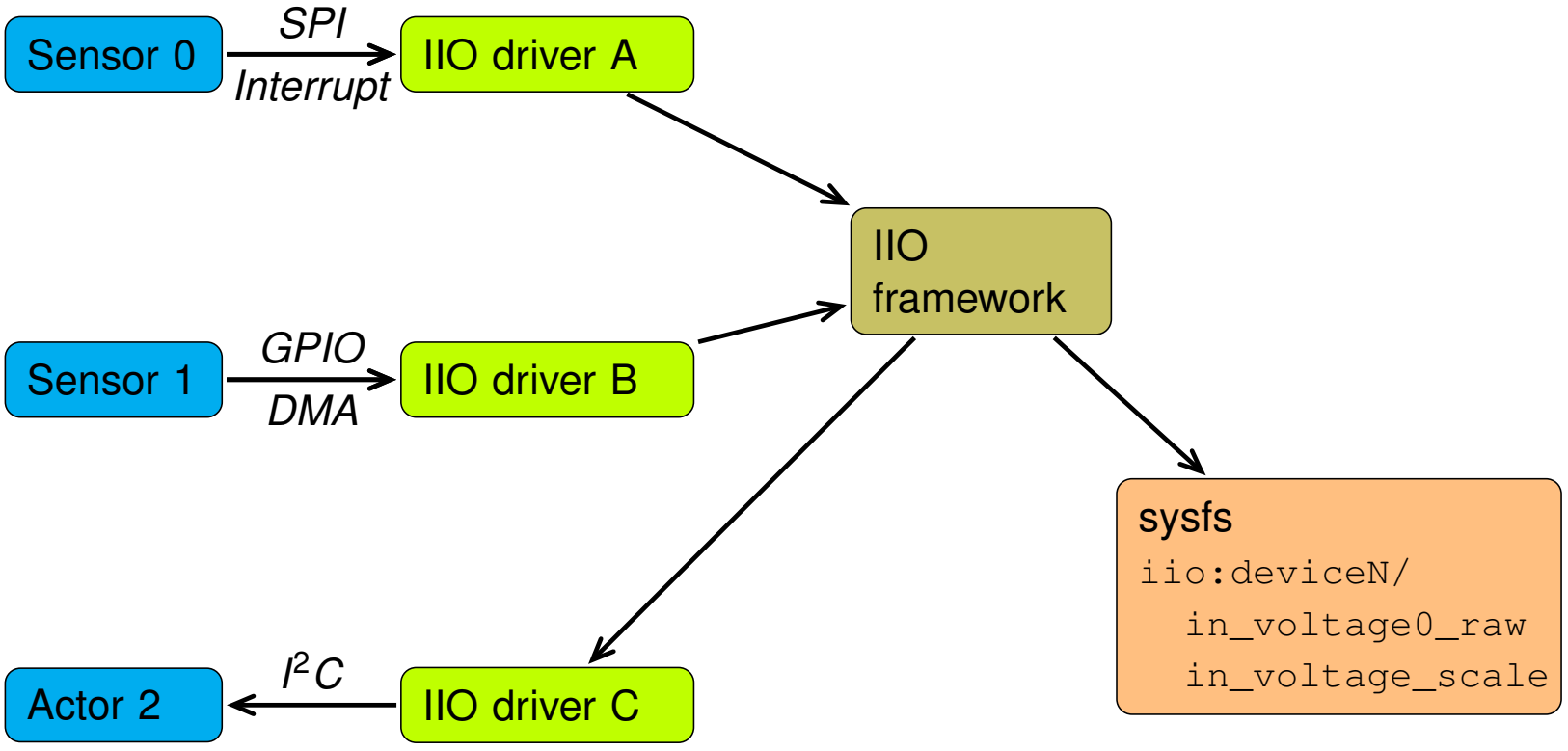
buffered mode

(hrtimer) trigger causes read process
data are written into buffer
→ asynchronous read from device node `/dev/iio:deviceN`

event triggered

Application waits (e. g. with `poll()`) on an event
it will be woken up by an interrupt which in turn is delivered by the sensor.
→ overflow or underflow of meassuring limits

Direct Mode


Sensor 0
SPI
Interrupt
IIO driver A
IIO framework
Sensor 1
GPIO
DMA
IIO driver B
sysfs
iio:deviceN/
in_voltage0_raw
in_voltage_scale
Actor 2
I²C
IIO driver C

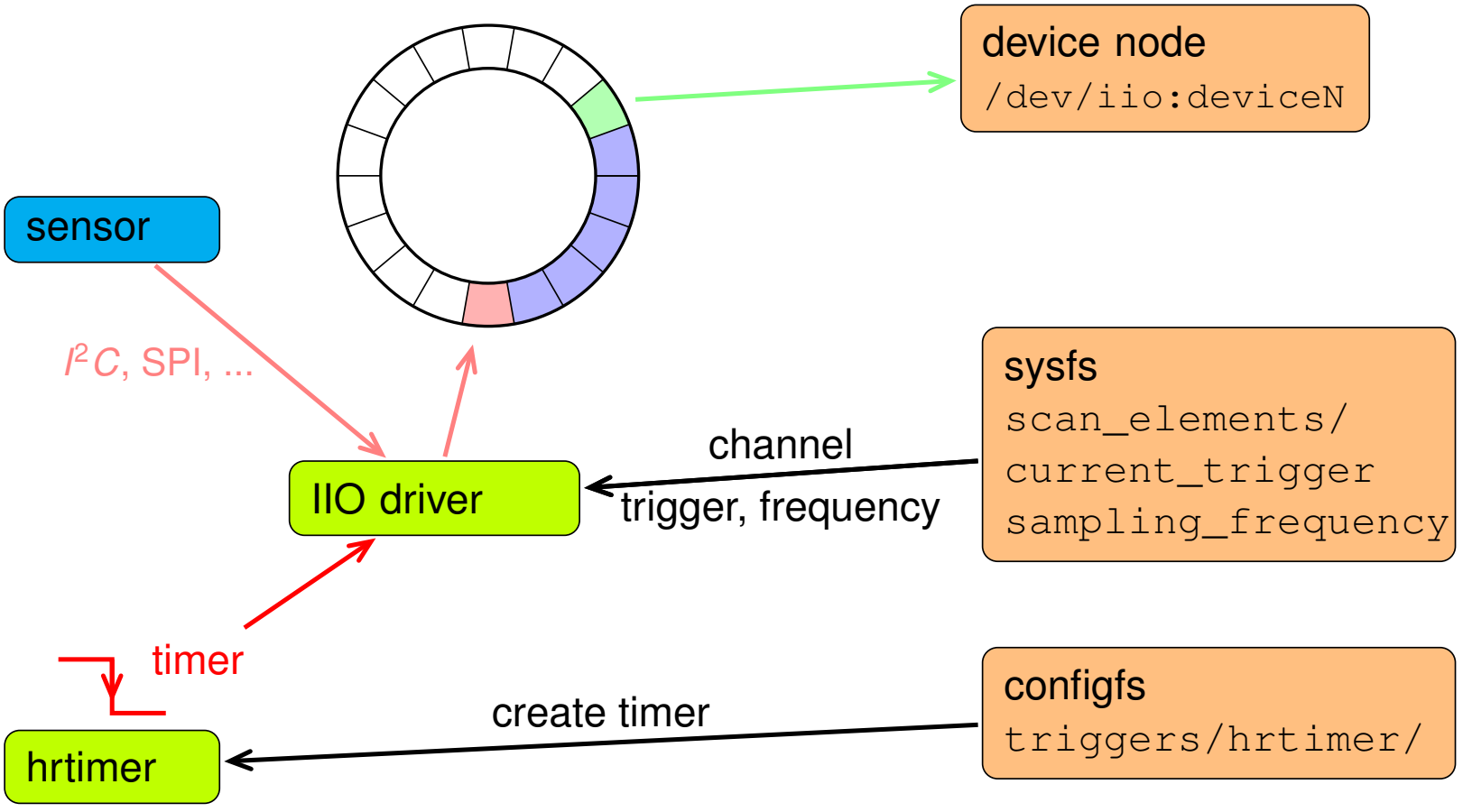
IIO device — directory structure I

```
$ tree /sys/bus/iio/devices/iio:device1
.
|-- buffer
|   |-- enable
|   |-- length
|   `-- watermark
|-- dev
|-- in_voltage0_raw
|-- in_voltage0_scale_available
|-- in_voltage1_raw
|-- in_voltage1_scale_available
|-- in_voltage_scale
|-- name
|-- of_node -> /firmware/devicetree/base/ocp/weight0
|-- scan_elements
|   |-- in_timestamp_en
```

IIO device — directory structure II

```
|    |-- in_timestamp_index
|    |-- in_timestamp_type
|    |-- in_voltage0_en
|    |-- in_voltage0_index
|    |-- in_voltage0_type
|    |-- in_voltage1_en
|    |-- in_voltage1_index
|    `-- in_voltage1_type
|-- subsystem -> ../../../../../bus/iio
`-- trigger
    `-- current_trigger
```

Buffered mode



Device tree of industrial IO device

```
$ cat /boot/am335x-wega-bw.dts

[...]

weight0 {
    compatible = "avia,hx711";
    pinctrl-names = "default";
    pinctrl-0 = <&weight0_pins>;
    sck-gpios = <&gpio0 26 GPIO_ACTIVE_HIGH>;
    dout-gpios = <&gpio1 13 GPIO_ACTIVE_HIGH>;
    avdd-supply = <&avdd>;
    clock-frequency = <100000>;
};
```

`lsiio` — list industrial IO devices

- IIO devices are identified by a number and a name
- With several devices of the same type the name is not unique
- The number is assigned in the order of registration
  → Can change after reboot
- Practical solution:
  Use device tree name for looking-up current device number

```
$ lsiio -v
[...]
Device 002: hx711
   in_voltage0_raw
   in_voltage1_raw

$ ls -l /dev/iio:device2
crw-rw---- 1 root root 250, 2 Apr 22 08:42 /dev/iio:device2
```

Device tree node name ↔ device number

- Get device number by iterating over all IIO devices

```
$ cat /sys/bus/iio/devices/iio:device2/uevent
MAJOR=250
MINOR=0
DEVNAME=iio:device0
DEVTYPE=iio_device
OF_NAME=weight0
OF_FULLNAME=/ocp/weight0
OF_COMPATIBLE_0=avia,hx711
OF_COMPATIBLE_N=1

$ cat /sys/bus/iio/devices/iio\:device*/uevent |                    \
                    grep -E '(DEVNAME|OF_NAME)' | xargs -n2 echo

DEVNAME=iio:device0 OF_NAME=adc
DEVNAME=iio:device1 OF_NAME=pressure
DEVNAME=iio:device2 OF_NAME=weight0
```

Create hrtimer trigger

- Creation of hrtimer (high resolution timer) trigger event in industrial IO is implemented as reaction on directory creation in configfs
- Setup sampling frequency to 1 Hz

```
$ mkdir /sys/kernel/config/iio/triggers/hrtimer/mytmr

$ lsiio -v
[...]
Trigger 000: mytmr

$ ls -l /sys/bus/iio/devices/trigger0/
-r--r--r-- 1 root root 4096 Apr 22 08:44 name
drwxr-xr-x 2 root root    0 Apr 22 09:37 power
-rw-r--r-- 1 root root 4096 Apr 22 08:44 sampling_frequency
lrwxrwxrwx 1 root root    0 Apr 22 09:37 subsystem -> ../../bus/iio
-rw-r--r-- 1 root root 4096 Apr 22 08:43 uevent

$ echo 1 > /sys/bus/iio/devices/trigger0/sampling_frequency
```

iio_generic_buffer — read buffered data

```
$ iio_generic_buffer -a -c 2 -N 2 -t mytmr

iio device number being used is 2
iio trigger number being used is 0

Enabling all channels
Enabling: in_voltage1_en
Enabling: in_timestamp_en
Enabling: in_voltage0_en

/sys/bus/iio/devices/iio:device2 mytmr
42.657402 40.615715 1745304439736372577
42.656483 40.615883 1745304440736376685

Disabling: in_voltage1_en
Disabling: in_timestamp_en
Disabling: in_voltage0_en
```

Setup channels for reading

- Channels can be enabled individually
- Index of channel in binary data stream as well as interpretation can be read

```
$ cd /sys/bus/iio/devices/iio:device2/scan_elements
$ ls
in_timestamp_en      in_voltage0_en      in_voltage1_en
in_timestamp_index   in_voltage0_index   in_voltage1_index
in_timestamp_type    in_voltage0_type    in_voltage1_type

$ cat in_voltage0_type
le:u24/32>>0
$ cat in_timestamp_type
le:s64/64>>0

$ echo 1 > in_timestamp_en
$ echo 1 > in_voltage0_en
```

Read buffered data manually

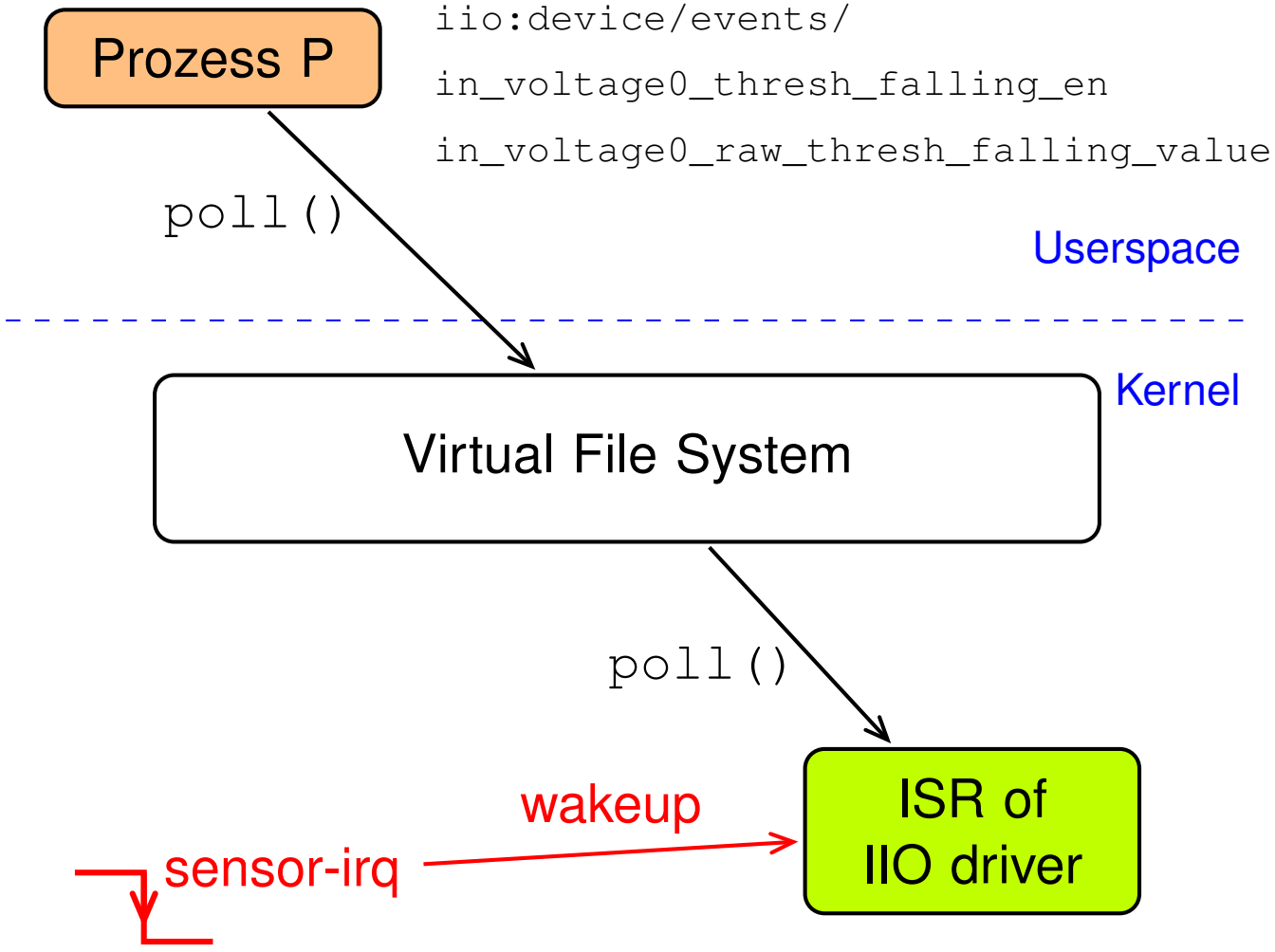
- Read binary data from character device node

```
$ cd /sys/bus/iio/devices/iio:device2/buffer
$ ls
data_available  direction  enable  length  watermark

$ echo 1 > enable

$ hexdump -X /dev/iio\:device2

00000000  3a c0 86 00 00 00 00 00 86 c8 d9 aa 33 95 38 18
00000010  10 c0 86 00 00 00 00 00 06 72 74 e6 33 95 38 18
00000020  f1 bf 86 00 00 00 00 00 fe 48 0f 22 34 95 38 18
00000030  af bf 86 00 00 00 00 00 9e 04 aa 5d 34 95 38 18
00000040  38 bf 86 00 00 00 00 00 d6 c8 44 99 34 95 38 18
```

Demonstration

- Setup and use buffered mode manually
- Make `hexdump` of data and verify content
- Use `iio_generic_buffer` instead

Event triggered



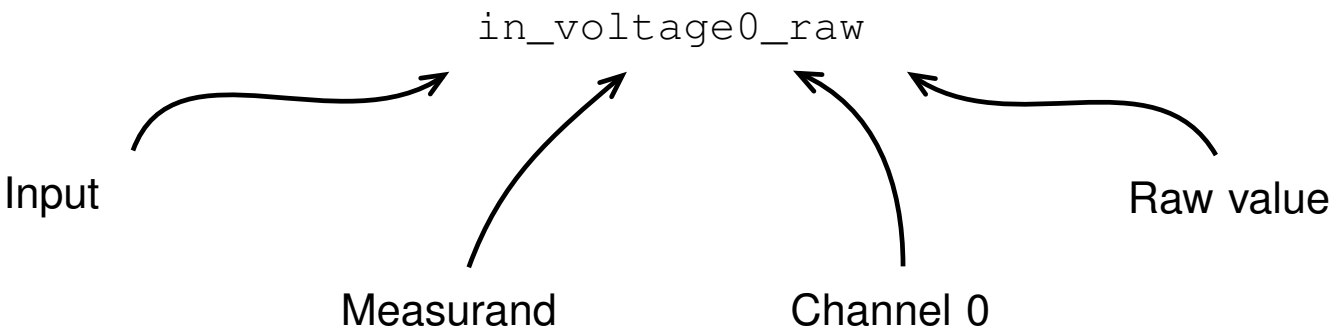
Industrial IO subsystem (IIO) I

- Support of access to sensors
- Standardized interface for kernel driver and userspace access
  $\Rightarrow$ userspace application can be used for different sensors
- Hardware interfaces:
  I^2C
  SPI
  GPIO
  etc.

Industrial IO subsystem (IIO) II

- Query and parameterization in sysfs:

```
/sys/bus/iio/devices/iio:device2/
```



- (SI) unit is defined for each measurand and identical for all sensors
- Attributes (amplification, scaling, offset, ...) mapped in standardized form in sysfs

Industrial IO subsystem (IIO) III

- Suitable for high query frequency
  e.g. zero copy up to userspace:
  - Data collection with DMA and buffer swap
  - Shown in userspace with `mmap()` und memory swap

Demonstration

- Linux kernel menuconfig for IIO
- Kernel sources of hx711.c
- Documentation of driver ABI
- Documentation of device tree

What sensors are mainline? I

Sensors (Stable v5.2, driver variants and staging not counted):

Sensor Type	Directory	Amount
Acceleration	`accel`	26
AD converter	`adc`	89
Amplifier	`amplifier`	1
Chem. sensors	`chemical`	8
DA converter	`dac`	36
Frequency synthesizer (PLL)	`frequency`	2
Gyro	`gyro`	12
Heart frequency, pulse	`health`	4
Humidity	`humidity`	7
Inertia	`imu`	6

What sensors are mainline? II

Lighting	`light`	42
Magnetic field	`magnetometer`	9
Inclination	`orientation`	2
Potentiometer	`potentiometer`	9
Potentiostat	`potentiostat`	1
Air pressure	`pressure`	13
Distance	`proximity`	9
Resolver	`resolver`	2
Temperature	`temperature`	9
Total		287

libiio — userspace library I

```
unsigned int nrdev, nratt, nrchan, i, j;
struct iio_context *ctx;
struct iio_device *dev;
struct iio_channel *chan;

ctx = iio_create_default_context();

nrdev = iio_context_get_devices_count(ctx);

for (i = 0; i < nrdev; i++) {

  dev = iio_context_get_device(ctx, i);

  nratt = iio_device_get_attrs_count(dev);

  for (j = 0; j < nratt; j++) {
    /* read device specific attributes */
  }
```

libiio — userspace library II

```
    nrchan = iio_device_get_channels_count(dev);
20
    for (j = 0; j < nrchan; j++) {

      chan = iio_device_get_channel(dev, j);

25    nratt = iio_channel_get_attrs_count(chan);

      /* get channel specific attributes */
    }
  }
```

Community — IIO development

- A lot of activity in mailing list linux-iio
- Initial author and maintainer is Jonathan Cameron
- Thorough and fast review
  → good input on how to do things differently
- Strong focus on unification:
  Into which available groups does a driver fit?
  Use of established interfaces rather than creating new ones
- Thanks to Lars-Peter Clausen for reviewing the slides

Links for industrial IO

Open source project bee scale (Bienenwaage)

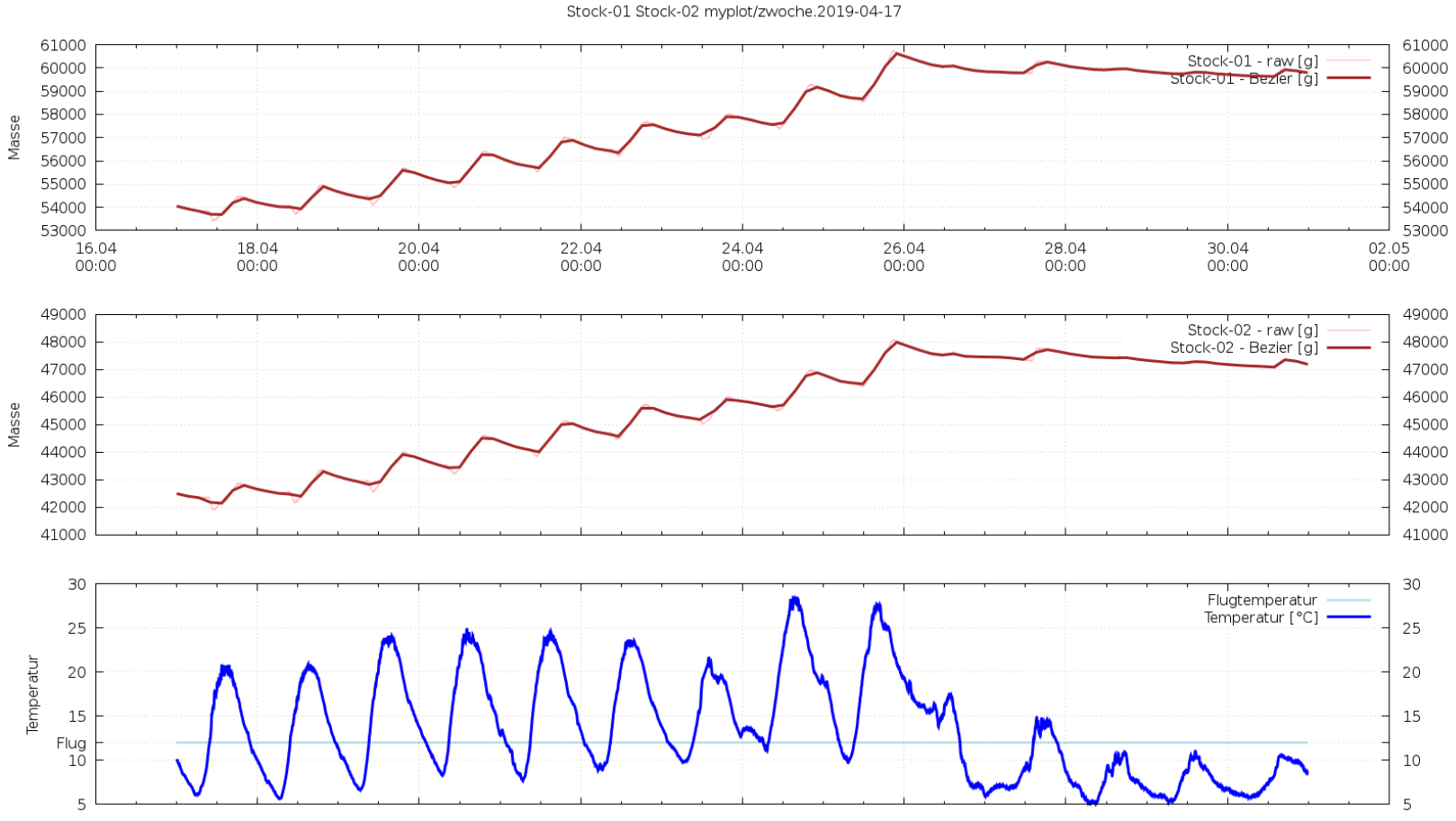
bw.grabenreith.de

Industrial IO documentation:

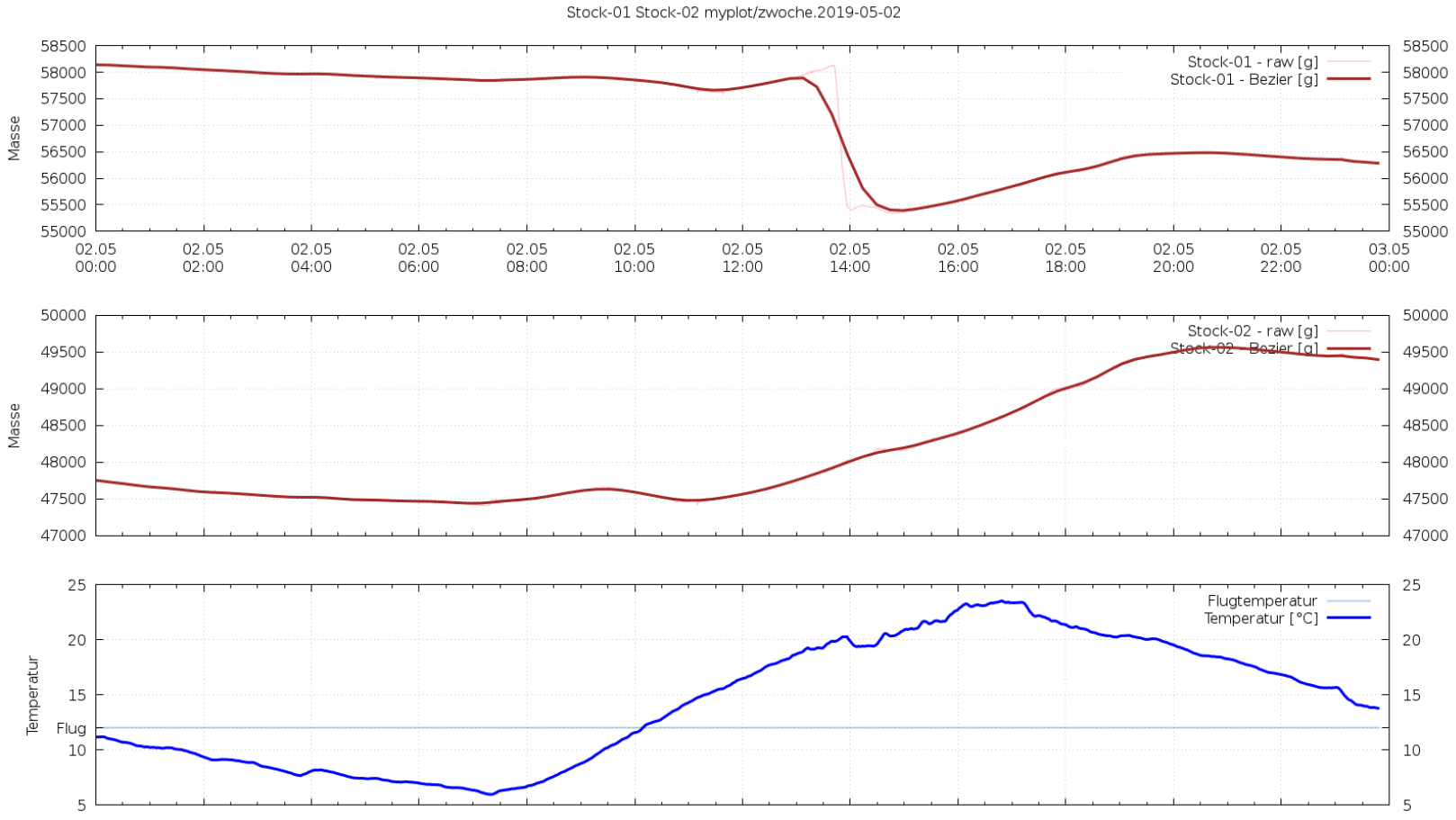
`linux-stable/Documentation/iio/`

`linux-stable/Documentation/ABI/testing/sysfs-bus-iio*`

Evaluation of bee scale with gnuplot I



Evaluation of bee scale with gnuplot II



Evaluation of bee scale with gnuplot III