Linux GPIO: Evolution and Current State of the User API

Embedded Linux Conference 2020

Bartosz Golaszewski

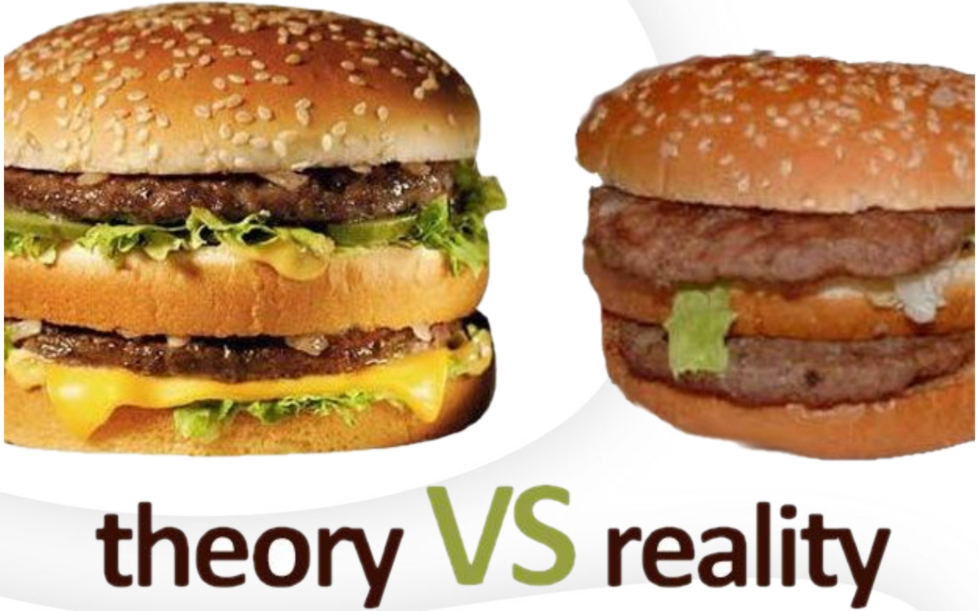
About us

- Embedded Linux Engineering Firm
- ~40 senior engineers, coming from the semiconductor world
- HW and SW products: from concept to manufacturing
- Upstream Linux kernel development and maintenance
- Founding developers of kernelCI.org project

About me

- 10 years experience
- Kernel and user-space developer
- Maintainer of libgpiod and co-maintainer of the GPIO kernel sub-system

A lot can change in a couple months...



A lot can change in a couple months...

The GPIO character device has been extended with new features in linux v5.5 and final new additions before declaring it feature-complete are planned for v5.6 & v5.7.

FALSE

A lot can change in a couple months...

The GPIO character device has been extended with new features in linux v5.5 but due to shortcomings in the first version of the ABI, the existing ioctl() calls are being retired and v2 of the ABI is being introduced aiming at a release as part of linux v5.9.

Linux GPIO: A Lesson in user API design

Embedded Linux Conference 2020

Bartosz Golaszewski

Agenda

1. Current state of user API
   a. sysfs = bad
   b. Character device
   c. Recently added features
   d. GPIO aggregator
2. Upcoming overhaul
   a. What’s wrong?
   b. What’s new?
   c. Advice on designing good uAPI
3. libgpiod
   a. what’s new
   b. Future

Current state of GPIO uAPI

GPIO in userspace

- Writing drivers for devices using GPIOs is encouraged wherever possible, but...
- Needed when no kernel device drivers provided/possible
  - Power switches
  - Relays
  - GPS
  - Bluetooth
- Certain users prefer to toggle GPIOs from user space
  - Intelligent home systems
  - Robotics

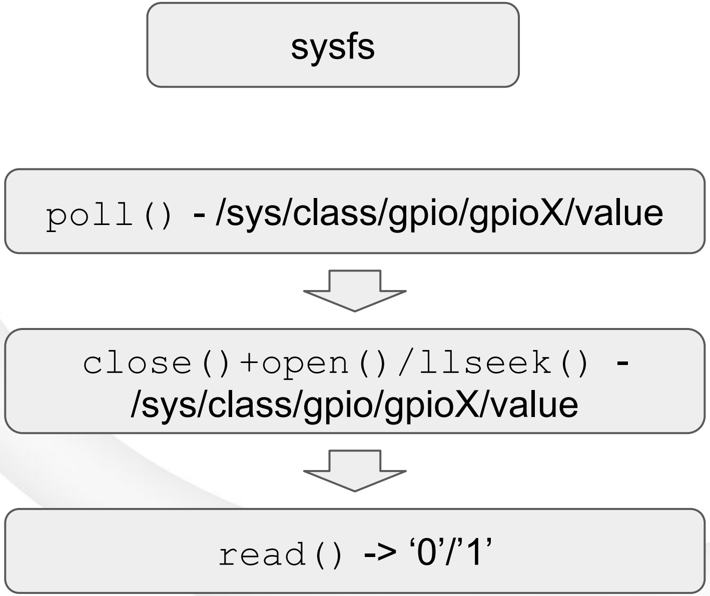
/sys/class/gpio - legacy user API

- d8f388d8 (“gpio: sysfs interface”)
- State not tied to process
- Concurrent access to sysfs attributes
- If process crashes, the GPIOs remain exported
- Cumbersome API
- Single sequence of GPIO numbers representing a two-level hierarchy - necessary to calculate the number of the GPIO, numbers not stable

GPIO character device - new user API

- Merged in linux v4.8
- One device file per gpiochip
  - /dev/gpiochip0, /dev/gpiochip1, /dev/gpiochipX…
- Similar to other kernel interfaces: `open() + ioctl() + poll() + read() + close()`
- Possible to request multiple lines at once (for reading/setting values)
- Possible to find GPIO lines and chips by name
- Open-source and open-drain flags, user/consumer strings, uevents
- Reliable polling

GPIO event polling - now reliable

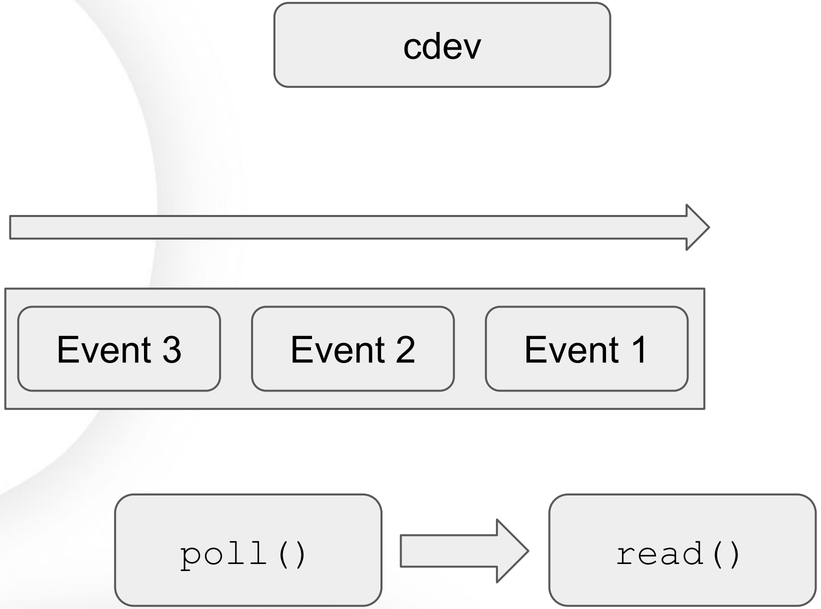


sysfs

`poll()` - /sys/class/gpio/gpioX/value

`close()+open()/llseek()` - /sys/class/gpio/gpioX/value

`read()` -> ‘0’/’1’



cdev

Event 3

Event 2

Event 1

`poll()`

`read()`

Events never get lost!

Unless kernel buffer overflows...

Character device:
new features

For more info on previous features:

https://www.youtube.com/watch?v=BK6gOLVRKuU

GPIO character device: set_config

- Allows to change the configuration of owned lines
  - Previously lines needed to be released, re-configured and re-requested -> this is racy
- Does not work for lines monitored for events

```
struct gpiohandle_config {
     __u32 flags;
     __u8 default_values[GPIOHANDLES_MAX];
     __u32 padding[4]; /* padding for future use */
};

#define GPIOHANDLE_SET_CONFIG_IOCTL _IOWR(0xB4, 0x0a, struct gpiohandle_config)
```

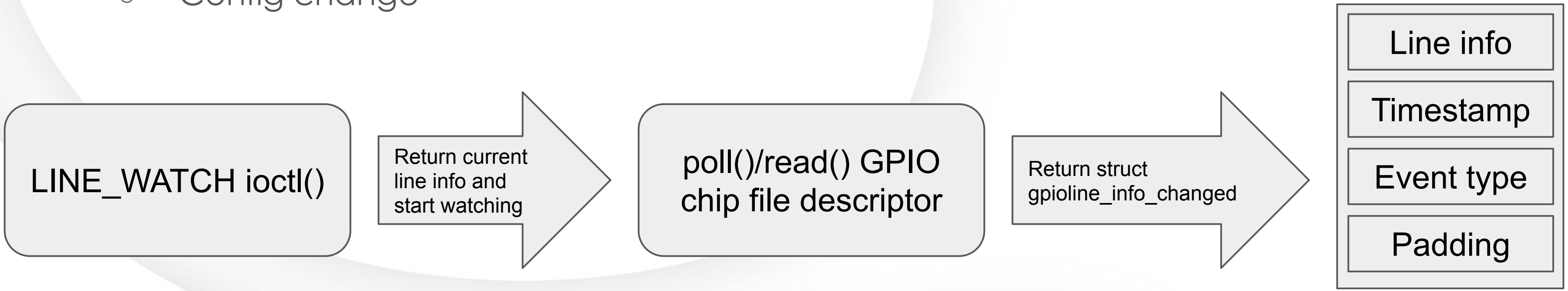
GPIO character device: bias settings

- Bias settings: pull-up/pull-down/disabled
  - Requested for a long time
  - RPi folks are very happy
  - Simple flags in struct gpioline_info and struct gpioline_handle

```
/* Linerequest flags */
#define GPIOHANDLE_REQUEST_INPUT             (1UL << 0)
#define GPIOHANDLE_REQUEST_OUTPUT            (1UL << 1)
#define GPIOHANDLE_REQUEST_ACTIVE_LOW        (1UL << 2)
#define GPIOHANDLE_REQUEST_OPEN_DRAIN        (1UL << 3)
#define GPIOHANDLE_REQUEST_OPEN_SOURCE       (1UL << 4)
#define GPIOHANDLE_REQUEST_BIAS_PULL_UP      (1UL << 5)
#define GPIOHANDLE_REQUEST_BIAS_PULL_DOWN    (1UL << 6)
#define GPIOHANDLE_REQUEST_BIAS_DISABLE      (1UL << 7)
```

GPIO character device: line watch

- Allows user-space to be notified about changes in line status
  - Line request
  - Line release
  - Config change



LINE_WATCH ioctl()

Return current line info and start watching

poll()/read() GPIO chip file descriptor

Return struct gpioline_info_changed

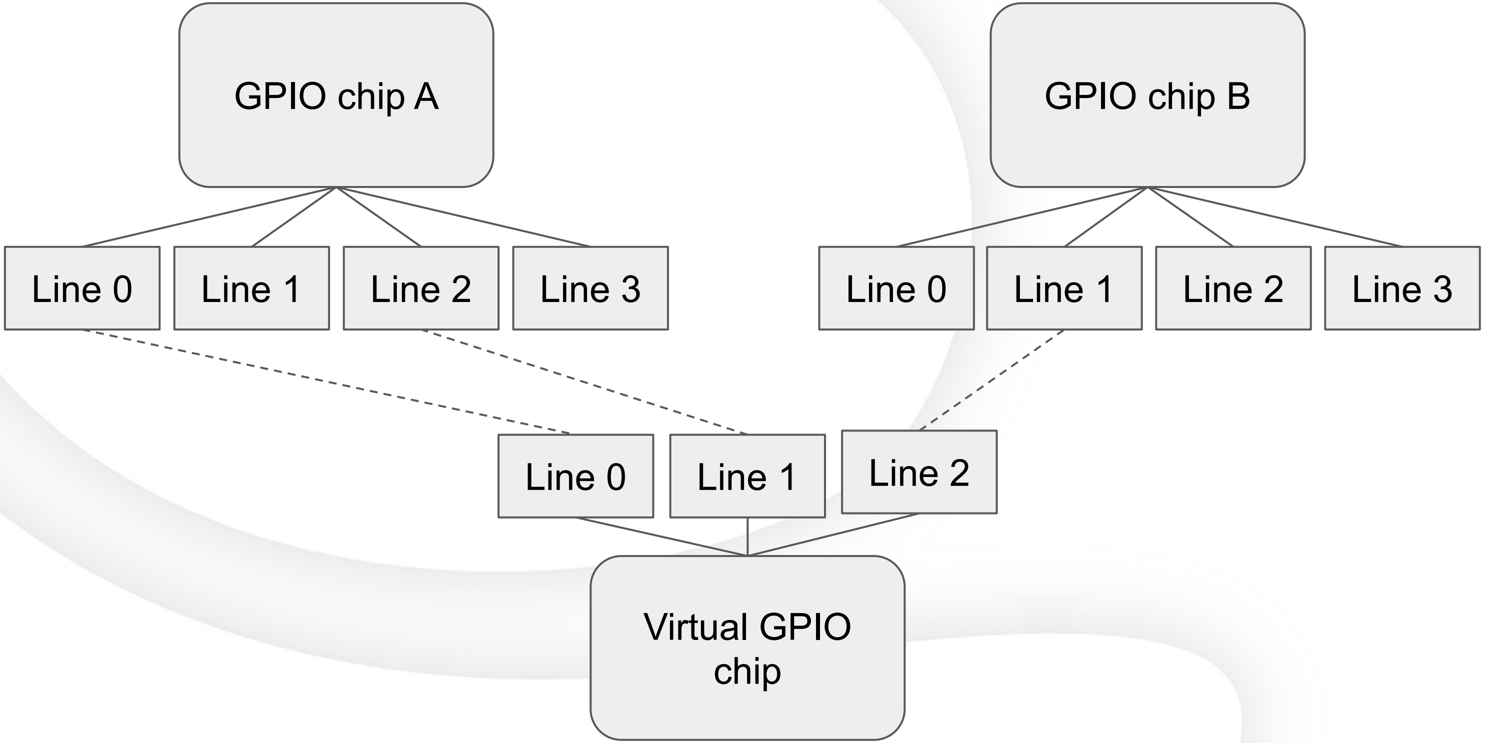
Line info
Timestamp
Event type
Padding

GPIO Aggregator

GPIO aggregator

- Access control for GPIO character devices is provided using unix persmissions
- All or nothing - either all lines are accessible to user or none are
- Sometimes it’s necessary to grant users access to certain GPIO lines
- “Aggregate” them via a virtual GPIO chip device

GPIO aggregator



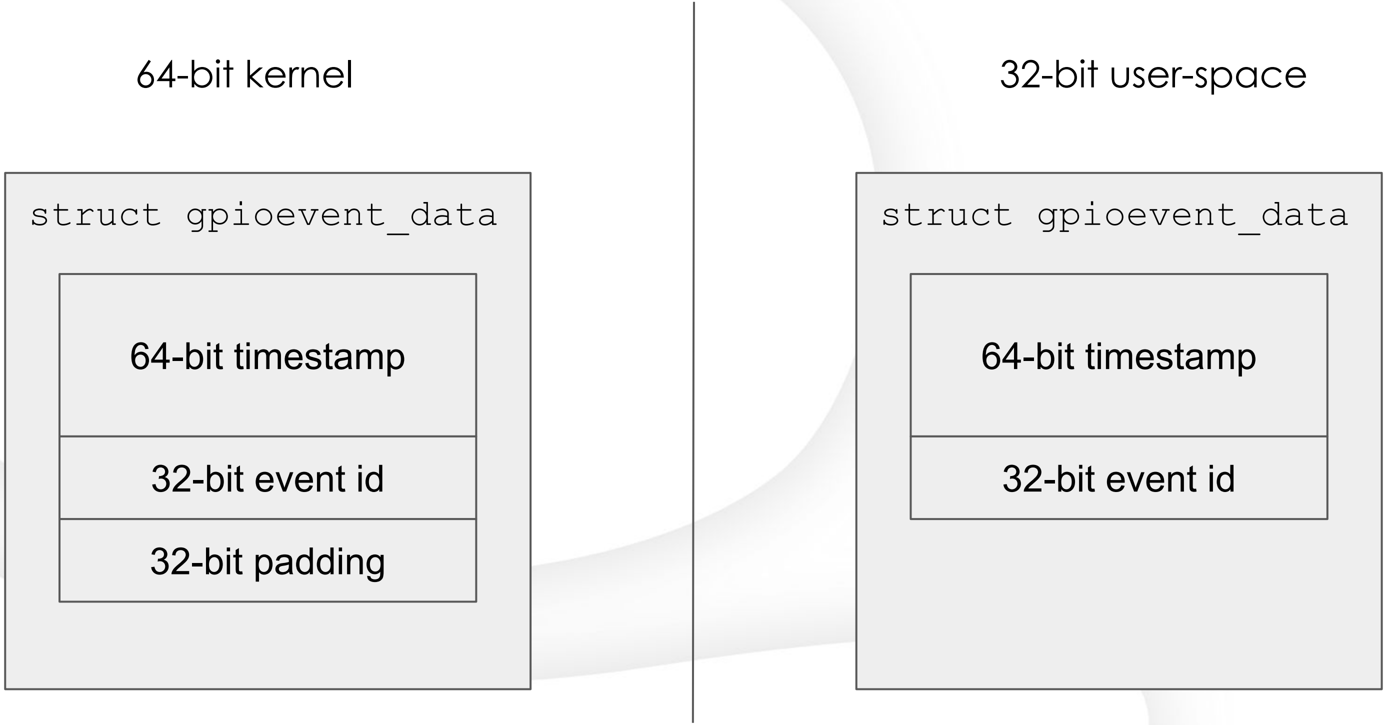
GPIO aggregator

```
$ cd /sys/bus/platform/drivers/gpio-aggregator
$ ls
bind delete_device module new_device uevent unbind
$ gpiodetect
gpiochip0 [gpio-mockup-A] (4 lines)
gpiochip1 [gpio-mockup-B] (4 lines)
$ echo "gpio-mockup-A 0,2 gpio-mockup-B 1" > new_device
$ ls
bind delete_device gpio-aggregator.2 module new_device uevent unbind
$ gpiodetect
gpiochip0 [gpio-mockup-A] (4 lines)
gpiochip1 [gpio-mockup-B] (4 lines)
gpiochip2 [gpio-aggregator.2] (3 lines)
$ gpioinfo gpiochip0
gpiochip0 - 4 lines:
      line   0:      unnamed "gpio-aggregator.2" input active-high [used]
      line   1:      unnamed       unused   input  active-high
      line   2:      unnamed "gpio-aggregator.2" input active-high [used]
      line   3:      unnamed       unused   input  active-high
$
```

Upcoming overhaul

Uh… so what’s wrong?

What’s wrong: 64-bit kernel with 32-bit user-space



64-bit kernel

`struct gpioevent_data`

64-bit timestamp

32-bit event id

32-bit padding

32-bit user-space

`struct gpioevent_data`

64-bit timestamp

32-bit event id

What’s wrong: no padding for future use

```
struct gpiohandle_request {
    __u32 lineoffsets[GPIOHANDLES_MAX];
    __u32 flags;
    __u8 default_values[GPIOHANDLES_MAX];
    char consumer_label[32];
    __u32 lines;
    int fd;
};
```

- ABI needs to remain stable
- No extension possible
- Extended config structure needed (debounce time etc.)

What’s wrong: using wrong clock for timestamps

- GPIO character device uses the REAL clock for timestamps
- It can go backwards…
- Need to use the mononic clock
- Switched to using the monotonic clock in commit `f8850206e160` `("gpio: Switch timestamps to ktime_get_ns()")`
- This breaks the ABI but we just bit the bullet...

Sensible requests for new features

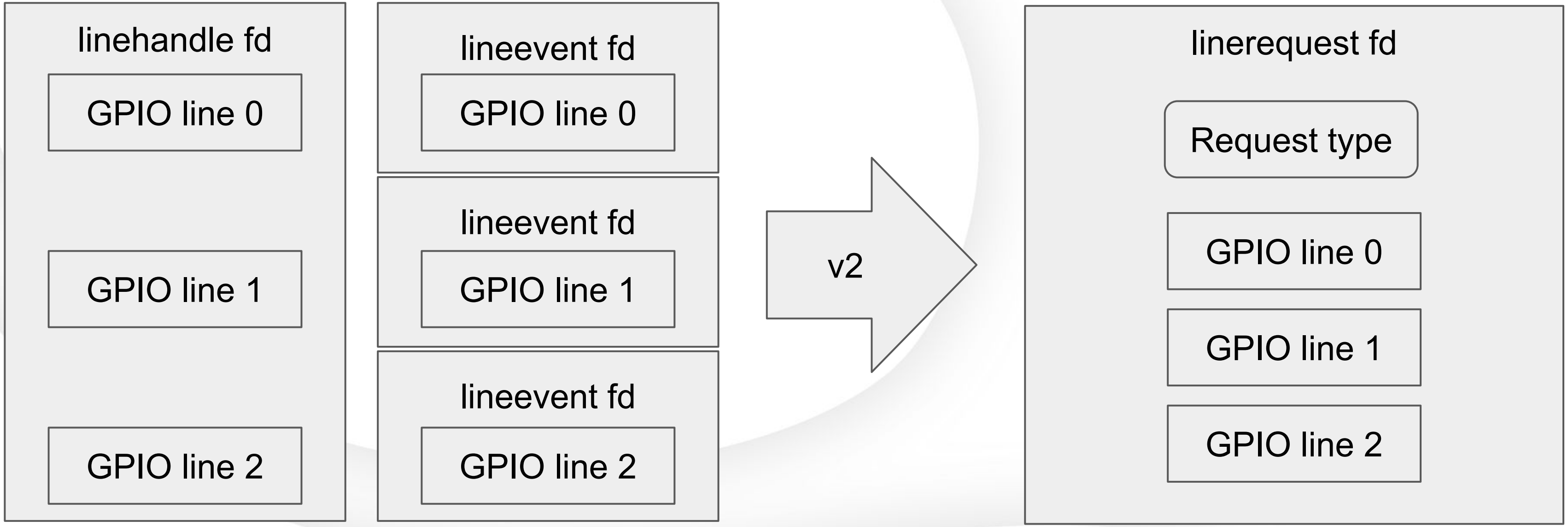
- Debounce time: for line events read from user-space
- Bias setting (pull-up/pull-down resistors controlled by drivers)
- Event sequence numbering
- All of the above can’t be implemented with current ioctl()s

GPIO chardev v2

GPIO chardev v2 - current status

- New uAPI proposed on linux-gpio
- Don’t break user-space - v1 ABI must remain stable
- Fix 64-bit kernel to 32-bit user-space issues
- Add padding to structures
- Rework flags
- Merge linehandle and lineevent requests

GPIO chardev v2 - merge linehandle and lineevent



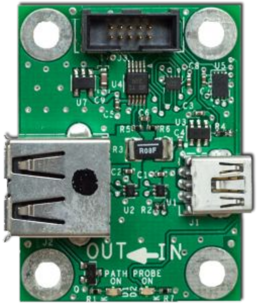
GPIO chardev v2 - rework flags

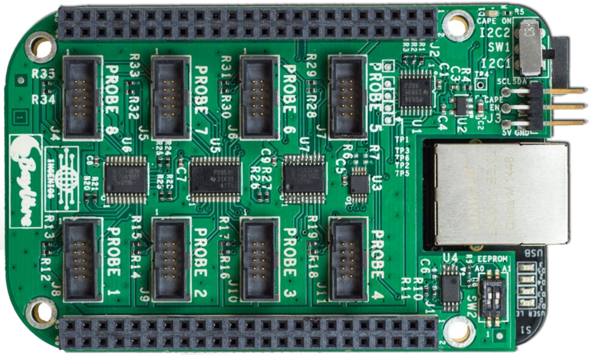
Direction	Drive	Bias	Edge
input	push-pull	disabled	none
output	open-drain	pull-up	falling
	open-source	pull-down	rising
			both

libgpiod

libgpiod – C library & tools for GPIO chardev

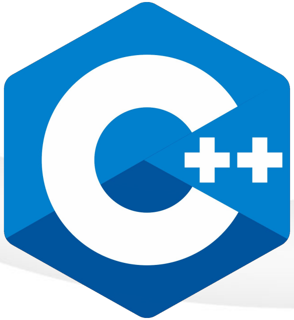
- History
  - Needed a solution for toggling power switches on BayLibre ACME
    - ~~IIO attributes~~
    - ~~Regulators controlled from user space~~
    - GPIO character device
  - Version 0.1 released on January 18th 2017
  - v1.0 released on February 7th 2018
  - Current stable version is 1.5.2
    - needs linux v5.5
  - v1.4.x series still supported for linux v5.4 (LTS)
  - v2.0 development starting soon (chardev v2)
  - v2.0 will **not** support chardev v1
  - v0.3.x support soon dropped

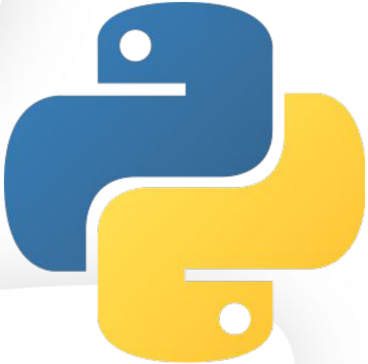




libgpiod – C library & tools for GPIO chardev

- Features
  - C API, fully documented in doxygen
  - Command-line tools: gpiodetect, gpioinfo, gpioset, gpioget, gpiofind & gpiomon
  - Custom test suite (working together with gpio-mockup kernel module and irq_sim)
- Language bindings





libgpiod – new features and future

- New features in v1.5
  - Support bias flags and set_config
  - Use existing testing frameworks for automated tests
    - GLib tests
    - Catch2
    - Python unittest
    - Bats
- Future (v2.x)
  - new API
  - GLib bindings
  - dbus bindings
  - gpiowatch

Q & A

Kudos to:

Geert Uytterhoeven <geert@linux-m68k.org>: for implementing the GPIO aggregator

Kent Gibson <warthog618@gmail.com>: for his work on v2 of GPIO uAPI