New GPIO interface for linux user space

SophiaConf 2021

Bartosz Golaszewski

Telecom Valley

SophiaConf

100% OPEN SOURCE - Conférences et Workshops

About us

- Embedded Linux Engineering Firm
- ~30 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 the linux kernel GPIO sub-system and libgpiod

Agenda

1. What are GPIOs?
2. GPIO sub-system in the kernel
3. Interacting with GPIOs from user-space
4. libgpiod
   a. What is it and what it improves
   b. Examples
   c. Bindings
   d. Future

GPIO – overview

- General-purpose input/output
- Generic pin
- Can be configured at run time
  - Input (readable)/output (writable)
  - Enabled/disabled
  - IRQs
- Provided by SoCs, expanders or multifunction devices (I2C, SPI, ...)

GPIO in the kernel

- Provider-consumer model
- Two co-existing interfaces
  - Based on GPIO numbers (legacy, deprecated)
  - Based on GPIO descriptors (recommended)
    - Easy access to GPIOs associated with devices
    - More fine-grained control
- GPIO chip drivers in drivers/gpio
- Consumers all over the place
  - Writing drivers for devices using GPIOs is encouraged wherever possible

GPIO in user space

- 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
  - Multiple attributes per GPIO: value, direction, active_low, edge
  - Single sequence of GPIO numbers representing a two-level hierarchy - necessary to calculate the number of the GPIO, numbers not stable
  - Polling possible but complicated: need to lseek() or reopen ‘value’ on events, need to open ‘value’ separately for every 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

Character device v2 – even newer user API

- *"Plan to throw one away"*
- Merged in linux v5.10
- Future-proof
- Per line config
- Line attributes overloading
- Edge detection decoupled from direction
- Bias flags, debounce period, event clock type
- Sequence numbers for events
- List status changes monitoring

Character device – user API (linux/gpio.h)

- Chip info
- Line info
- Line request for values
- Reading values
- Setting values
- Line request for events
- Polling for events
- Reading events

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 1.6.3
  - v2.0 coming soon

libgpiod – C library & tools for GPIO chardev

- Features
  - C API, fully documented in doxygen
  - Command-line tools: gpiodetect, gpioinfo, gpioset, gpioget, gpiofind & gpiomon (gpiowatch coming soon)
  - Custom test suite (working together with gpio-mockup kernel module and irq_sim)
    - Soon switching to gpio-sim
  - C++ bindings
  - Python 3 bindings

libgpiod tools - examples

```
$ gpiodetect
gpiochip0 [gpio-mockup-A] (8 lines)
gpiochip1 [gpio-mockup-B] (8 lines)
gpiochip2 [gpio-mockup-C] (8 lines)

$ gpioinfo gpiochip1
gpiochip1 - 8 lines:
        line   0: "gpio-mockup-B-0" unused output active-high
        line   1: "gpio-mockup-B-1" unused output active-high
        line   2: "gpio-mockup-B-2" unused output active-high
        line   3: "gpio-mockup-B-3" unused output active-high
        line   4: "gpio-mockup-B-4" unused output active-high
        line   5: "gpio-mockup-B-5" unused output active-high
        line   6: "gpio-mockup-B-6" unused output active-high
        line   7: "gpio-mockup-B-7" unused output active-high
```

libgpiod tools - examples

```
$ gpiofind gpio-mockup-B-3
gpiochip1 3

$ gpioget `gpiofind gpio-mockup-B-3`
0

$ gpioset gpiochip1 3=1
$ gpioget gpiochip1 1 2 3 4 5
0 0 1 0 0

$ gpioset --mode=wait gpiochip2 0=1

$ gpiomon gpiochip0 2
event:  RISING EDGE offset: 2 timestamp: [1508094667.935877214]

$ gpiomon --format="%o %e %s.%n" gpiochip0 2
2 1 1508094729.895930484
```

libgpiod – C++ bindings

- C API wrapped in C++17 classes
- RAII
- Fully documented in Doxygen
- Exception-safe
- Tools reimplemented in C++ as an example
- Many examples included

libgpiod – Python 3 bindings

- C API wrapped in a set of Python 3 classes
- Fully documented in pydoc
- Native Python3 module written in C
- Tools reimplemented in Python as an example
- Many examples included

libgpiod – dbus bindings (coming soon)

- Work-in-progress
- git@github.com:brgl/libgpiod.git topic/gpio-dbus
- Daemon written in C and based on GDBus and Gudev
- Chip and line objects
- Properties: name, label, offset etc.
- Methods: request, set_value, get_value etc.
- Signals: line events

libgpiod – future

- v2.0 coming soon
- entirely reworked C API
  - intuitive line config
  - GPIO chips and requests are decoupled
- reworked language bindings
- rust bindings
- glib bindings

libgpiod – C library & tools for GPIO chardev

- Where to get it:
  - Hosted at kernel.org
    - Source: https://git.kernel.org/pub/scm/libs/libgpiod/libgpiod.git/
    - Releases: https://www.kernel.org/pub/software/libs/libgpiod/
  - Available in meta-openembedded & buildroot
  - Packaged in Fedora, Arch, Debian linux and more
- Contributions & bug reports:
  - Send e-mails to linux-gpio@vger.kernel.org
  - Use [libgpiod] prefix

Q & A

THANK YOU!