GPIO as a character device
or Fedora on wheels
Aleksandra Fedorova

- CI Engineer at Red Hat
- Fedora Ambassador and member of FESCo
- https://quantum-integration.org
- Fedora User Group NRW @ Meetup.com

bookwar @ IRC
https://telegram.me/bookwar
bookwar@fedoraproject.org
Fedora 17 and 18 – 2013-2014:
- Raspberry Pi is based on ARMv6 which is not supported by upstream kernel
- Pidora – Fedora Remix with heavily modified kernel by Seneca College

since Fedora 27 – 2017:
- Fedora for ARM (ARMv7 and AArch64) supports Raspberry Pi 2 and 3

since Fedora 28 – 2018:
- Support for Raspberry Pi 3+

In Fedora 29 – 2018:
- Extended hardware support (wifi, hdmi, touchscreen..)

https://fedoraproject.org/wiki/Architectures/ARM/Raspberry_Pi
Getting “started”
(root)# dnf install python3-RPi.GPIO
(root)# python
> import RPIO

import RPIO._GPIO as _GPIO
SystemError: This module can only be run on a Raspberry Pi!

https://bugzilla.redhat.com/show_bug.cgi?id=1471731
attempt #2: sysfs interface to gpio

```
(root)# cd /sys/class/gpio
(root)# echo 4 > /sys/class/gpio/export
(root)# echo 1 > /sys/class/gpio/gpio4/value
```

But there is no /sys/class/gpio...
All available GPIO libraries

- have hardcoded Raspbian-specific parameters;
- use sysfs or /dev/mem interface to GPIO.

https://github.com/JamesBarwell/rpi-gpio.js/blob/master/rpi-gpio.js#L8
Turning on the light
• In kernel-4.6 sysfs interface for GPIO was deprecated, and it is now disabled in the default kernel.

• But there is a new interface: /dev/gpiochip0
  – Character device
  – Operated via ioctl calls
  – Manages multiple pins at once
#define GPIO_GET_CHIPINFO_IOCTL _IOR(0xB4, 0x01, struct gpiochip_info)
#define GPIO_GET_LINEINFO_IOCTL _IOWR(0xB4, 0x02, struct gpioline_info)
#define GPIO_GET_LINEHANDLE_IOCTL _IOWR(0xB4, 0x03, struct gpiohandle_request)
#define GPIO_GET_LINEEVENT_IOCTL _IOWR(0xB4, 0x04, struct gpioevent_request)
#define GPIOHANDLE_GET_LINE_VALUES_IOCTL _IOWR(0xB4, 0x08, struct gpiohandle_data)
#define GPIOHANDLE_SET_LINE_VALUES_IOCTL _IOWR(0xB4, 0x09, struct gpiohandle_data);
wrapper functions for ioctl calls

```c
int get_chipinfo(int fd, struct gpiochip_info* info){
    int status;
    status = ioctl(fd, GPIO_GET_CHIPINFO_IOCTL, info);
    return status;
};
...

int get_linehandle(int fd, struct gpiohandle_request *req) {
    int status;

    status = ioctl(fd, GPIO_GET_LINEHANDLE_IOCTL, req);
    return status;
};
```
def line_info(self, line):
    _info = _gpioline_info(line=line)

    status = libgpioctl.get_lineinfo(self.fd, ctypes.byref(_info))
    if status != 0:
        raise GPIOError("get_chipinfo call returned non-zero status")

    info = {
        "line": _info.line,
        "flags": _info.flags,
        "name": _info.name,
        "consumer": _info.consumer,
    }

    return info
from gpiodev import GPIOHandle
import time

RedBlueLED = GPIOHandle((26, 21), mode="out")

states = [
    (1, 0),  # red
    (0, 1),  # blue
    (1, 1),  # purple
]

for state in states:
    RedBlueLED.set_values(state)
    time.sleep(5)
Wheels?
Two DC motors
L293D
4 AA batteries
Fedora 29
And one GPIOHandle objects controls four GPIO pins:
left wheel direction, left wheel power, right wheel direction, right wheel power
and it works
Conclusion
• `pip install git+https://github.com/bookwar/python-gpiodev`
• 300 lines of code
• set values, get values, read event data
• doesn’t make any assumptions about the system, only needs `/dev/gpiochip0` to work
alternatives?

Libgpiod
- Written in C
- Provides cli tools (gpiodetect, gpioinfo, gpioget, gpioset..)
- Got Python bindings
takeaways

- Raspberry Pi ecosystem is in a dangerous state
  - It won’t survive after rebase to the latest kernel
  - Too many lower-level things are assumed and hardcoded in high-level libraries
- There is a lot of fun in reimplementing even most simple Raspberry Pi tutorials (blinking LED light, DC motor, sonar..) in a “new way”
- There is a lot of value in doing that too
Aleksandra Fedorova

- Fedora User Group NRW @ Meetup.com
- bookwar @ IRC
- https://telegram.me/bookwar
- bookwar@fedoraproject.org