BÀI 22: GPIO DRIVER

Điều khiển phần cứng từ Kernel

🎯 Mục tiêu bài học

1. **Lịch sử:** Legacy GPIO API (số nguyên) vs Modern GPIO Descriptor (cấu trúc).
2. **Legacy API:** `gpio_request`, `gpio_direction_output`, `gpio_set_value`.
3. **Thực hành:** Viết Driver nhấp nháy LED khi User ghi lệnh.

1. Legacy GPIO API (Cũ nhưng dễ)

Trong các Kernel cũ (hoặc code đơn giản), GPIO được quản lý bằng số hiệu (Integer).

- `gpio_request(int gpio, const char *label)` : Xin quyền sử dụng.
- `gpio_free(int gpio)` : Trả lại quyền.
- `gpio_direction_input(int gpio)` : Cài đặt Input.
- `gpio_direction_output(int gpio, int value)` : Cài đặt Output & giá trị đầu.
- `gpio_set_value(int gpio, int value)` : Ghi 0 hoặc 1.
- `gpio_get_value(int gpio)` : Đọc trạng thái.

2. Quy trình viết LED Driver

1. **Init Module:**
   - Xin cấp phát GPIO (`gpio_request`).
   - Kiểm tra xem GPIO có hợp lệ không (`gpio_is_valid`).
   - Set hướng ra (`gpio_direction_output`).
2. **Hàm Write:**
   - Nhận dữ liệu từ User (0 hoặc 1).
   - Gọi `gpio_set_value` để bật/tắt LED.
3. **Exit Module:**
   - Tắt LED.
   - Giải phóng GPIO (`gpio_free`).

3. Modern API (Giới thiệu)

- Kernel mới khuyến khích dùng `gpiod_*` (GPIO Descriptor).
- Nó gắn liền với **Device Tree**. Thay vì hardcode số `60`, ta lấy thông tin từ Node trong file `.dts`.
- *(Chúng ta sẽ học kỹ phần này ở Bài 25).*

🛠️ PHẦN THỰC HÀNH (LAB 22)

Driver bật tắt LED

Yêu cầu

1. Tìm số hiệu GPIO nối với LED trên board (VD: GPIO 60 trên BeagleBone).
2. Viết Driver `led_driver.c`:
    - Tạo file `/dev/my_led`.
3. Test:

```
# Bật đèn
echo 1 > /dev/my_led
# Tắt đèn
echo 0 > /dev/my_led
```

4. **Nâng cao:** Tự động nhấp nháy LED khi Driver được nạp (Sử dụng Kernel Timer - sẽ học sau, hoặc dùng `mdelay` trong vòng lặp - *Không khuyến khích treo máy*).

Q & A

Hẹn gặp lại ở Bài 23: Xử lý Ngắt (Interrupt)!