TinyLinux-IoT-KernelLab: 从零实现一个可扩展 Linux 智能终端系统

虚拟机ubuntu 16.04 开发板：正点原子阿尔法imx6ull开发板 参考教程：正点原子【正点原子】I.MX6U嵌入式Linux驱动开发指南V1.81

①功能

1.设备驱动文件合集

Module	Skills
platfor LED 驱动	platform_device / device tree 匹配
按键INPUT子系统	INPUT子系统
PWM 驱动 + 用户态接口	控制背光
I2C 传感器驱动	AP3216C 三合一环境光传感器
SPI 传感器驱动	ICM-20608 六轴传感器
UART	tty driver wrapper
CAN 收发框架（使用 flexcan）	netdev / socketCAN
EEPROM 驱动 + sysfs	file operations, sysfs node

用统一的CLI工具访问设备

```
myctl led on
myctl can send 01 02 03
myctl i2c read accel
```

2.加 LLM API → “语音/命令控制开发板”

②开发流程

2025.11.04 加入LED功能

1. 在设备树种创建节点

在根节点/下添加节点，描述一个LED设备

```
    gpioled{
        compatible = "gpio-leds";
        pinctrl-names = "default";
        pinctrl-0 = <&pinctrl_led>;
```

```
        led-gpios   =  <&gpio1 3 GPIO_ACTIVE_LOW>;
        status = "okay";
    };
```

先编写一个基础的驱动框架来测试一下。

```
#include "asm/gpio.h"
#include "asm/string.h"
#include "linux/fs.h"
#include "linux/interrupt.h"
#include "linux/mm.h"
#include "linux/printk.h"
#include <linux/cdev.h>
#include <linux/delay.h>
#include <linux/device.h>
#include <linux/errno.h>
#include <linux/gpio.h>
#include <linux/ide.h>
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/string.h>
#include <linux/types.h>
#include <linux/uaccess.h>

#include <linux/jiffies.h>
#include <linux/of.h>
#include <linux/of_address.h>
#include <linux/of_gpio.h>
#include <linux/of_irq.h>
#include <linux/timer.h>
#include <linux/types.h>

#include <linux/irq.h>
#include <linux/irqreturn.h>
#include <linux/of_irq.h>
#include <linux/platform_device.h>

MOUDULE_LICENSE("GPL");
MOUDULE_AUTHOR("TATAROSE");


static int __init led_init(void){
    return 0;
}
static void __exit led_exit(void){

}

module_init(led_init);
module_exit(led_exit);
```

配置一下Cmake和Makefile,项目结构为

```
.
├── app
│   ├── CMakeLists.txt
│   └── myctl.c
├── build.sh
├── cmake
│   └── toolchains
│       └── Toolchain-arm-linux-gnueabihf.cmake
├── CMakeLists.txt
├── drivers
│   └── LED
│       ├── CMakeLists.txt
│       ├── leddriver.c
│       ├── Makefile
│       └── modules.order
├── Makefile
├── scripts
```

`app`下存放总应用程序，`Cmake`用户态即可编译出应用程序。`drivers`下存放驱动程序，由于使用`kbuild`来编译模块，所以Cmake无法替代`Makefile`，所以将`Makefile`添加到`drivers/LED`。