



**《嵌入式系统》**

**课程实验报告**

姓名：苏一涵

学院：信息学院

系：软件工程

专业：软件工程

学号：36720232204041

2025年11月

**第5次实验 Linux驱动与应用设计实验**

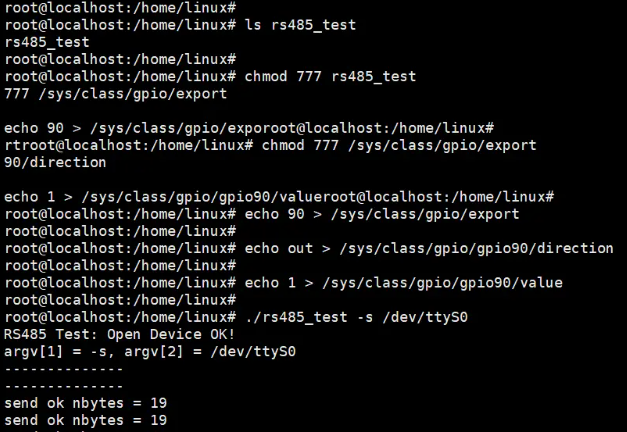
1. **实验设备**

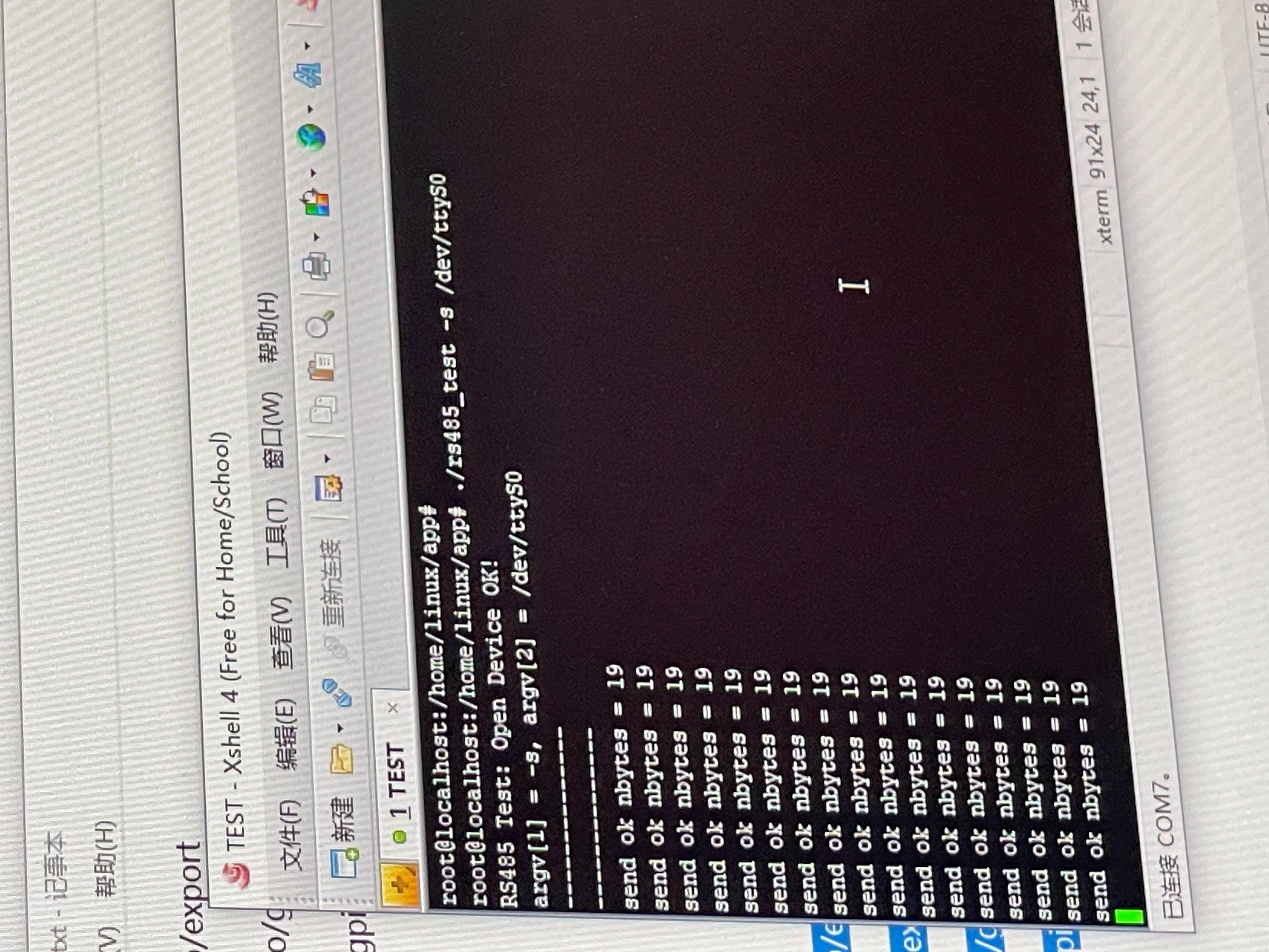
**（1）PC微机**

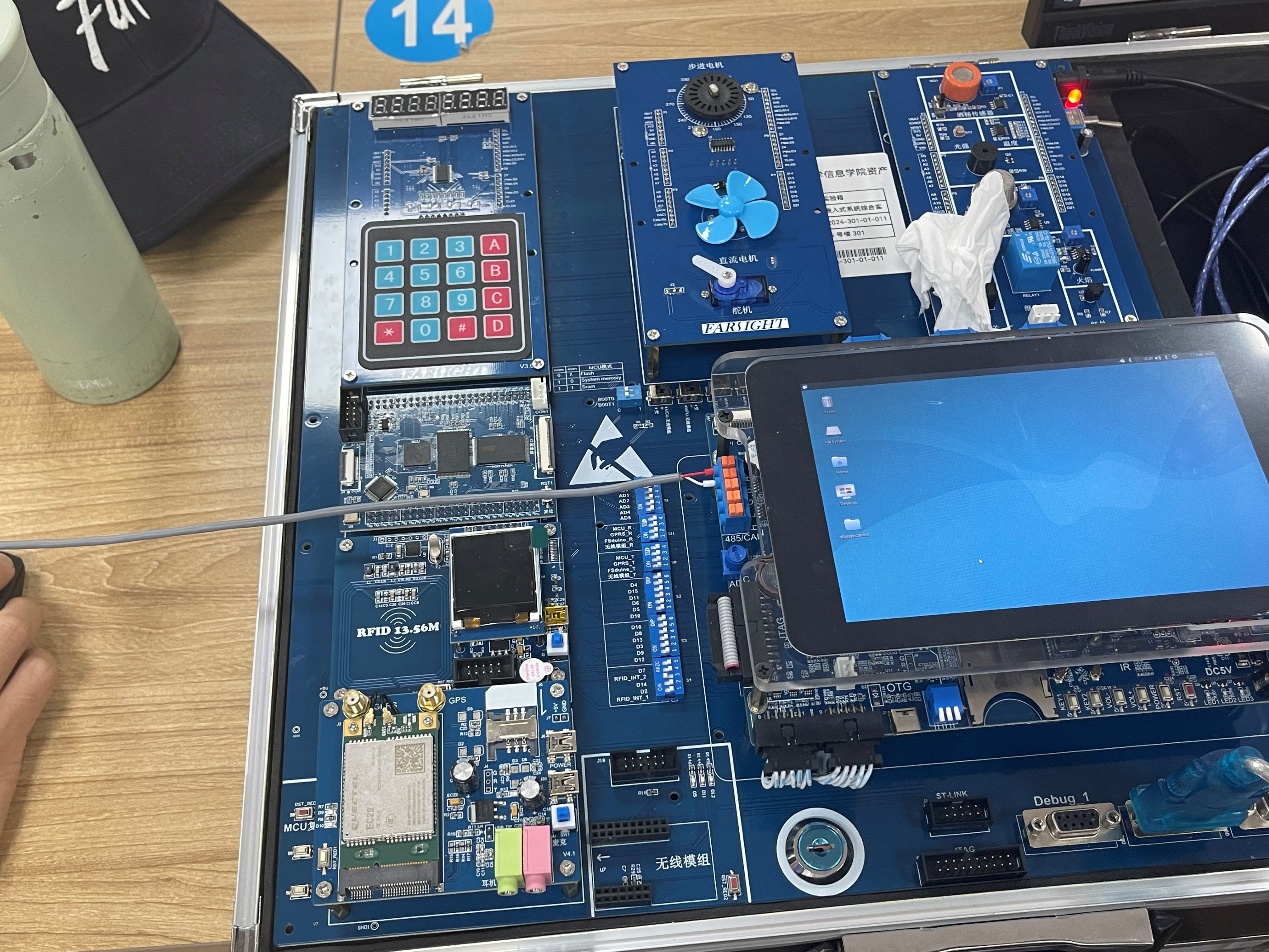
**（2）嵌入式系统综合实验箱（FS3399M4）**

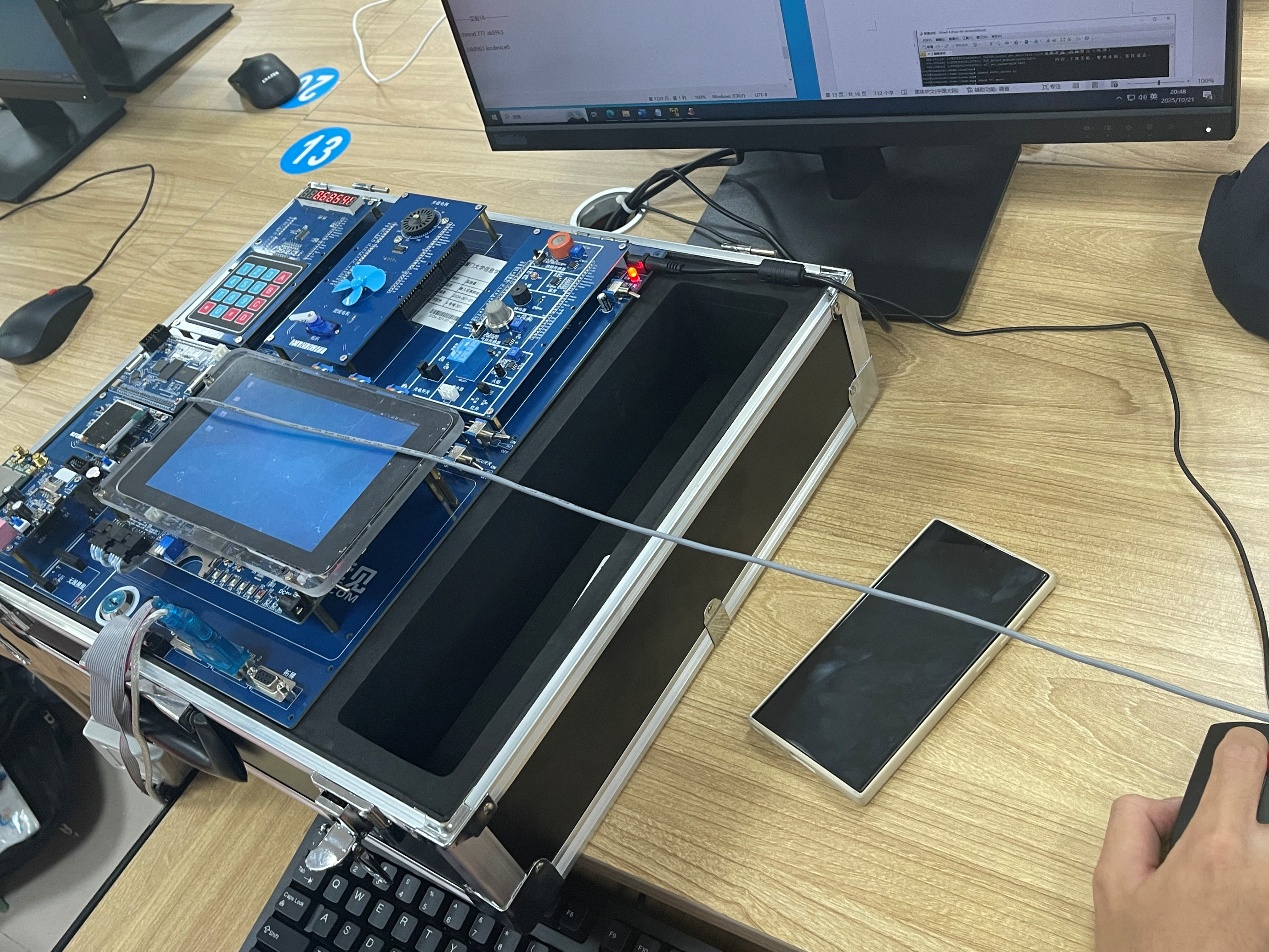
1. **实验内容**
   1. **实验要求**
   2. **补充完成第4次实验中的双机通信验证性实验**

* **实验19：RS-485（不需要驱动程序）**
* **实验20：CAN总线（不需要驱动程序）** 
  1. **请从设计实验1-1～1-7中选择一个（按键控制）**
  2. **请从设计实验2-1～2-7中选择一个（小键盘控制）**
  3. **请从设计实验3-1～3-7中选择一个（红外遥控器控制）**
  4. **上述21个实验中，**
* **必须选择1个控制步进电机的实验；由于步进电机涉及多线程编程，为进一步理解多线程编程的应用场景，需针对步进电机采用两种方式编程：多线程和单线程（即正常方式）**
* **所选择的3个实验必须是控制不同的设备；**
  1. **请从设计实验4-1～4-3中任选1个（数码管显示）**
  2. **挑战实验：实验5-1、5-2、6-1、6-2中任选一个（双机通信），完成挑战实验的加10分**
* **实验19：RS-485（不需要驱动程序）**

**在Ubuntu的“终端”上执行： cd /home/linux/workdir/fs3399/application/rs485aarch64-linux-gnu-gcc rs485\_test.c -o rs485\_testcp rs485\_test /mnt/hgfs/share/在Xshell上将电脑share目录中的rs485\_test文件上传到2个实验箱的MPU开发板中在实验箱1的在Xshell上运行（接收）：chmod 777 rs485\_testchmod 777 /sys/class/gpio/exportecho 90 > /sys/class/gpio/exportecho out > /sys/class/gpio/gpio90/directionecho 1 > /sys/class/gpio/gpio90/value****./rs485\_test -r /dev/ttyS0**

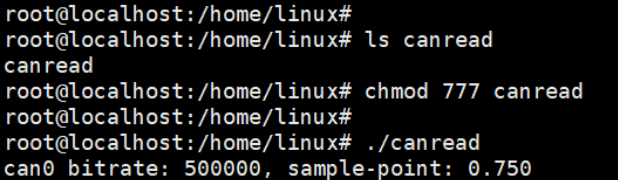
****

****

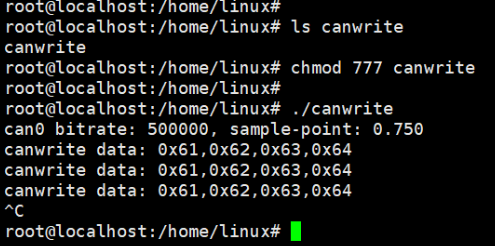
****

* **实验20：CAN总线（不需要驱动程序）**

**在电脑的Ubuntu“终端”上执行： cd /home/linux/workdir/fs3399/application/canaarch64-linux-gnu-gcc canwrite.c libsocketcan.c -o canwriteaarch64-linux-gnu-gcc canread.c libsocketcan.c -o canreadcp canwrite /mnt/hgfs/share/cp canread /mnt/hgfs/share/在Xshell上将电脑share目录中的canwrite文件上传到第1个实验箱的MPU开发板中，将电脑share目录中的canread文件上传到第2个实验箱的MPU开发板中在实验箱1的在Xshell上运行（接收）：chmod 777 canread./canread**



**在实验箱2的在Xshell上运行（发送）：chmod 777 canwrite./canwrite**



**实验结果**

**写入**

****

**读出**

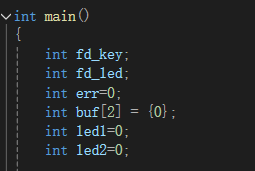
****

****

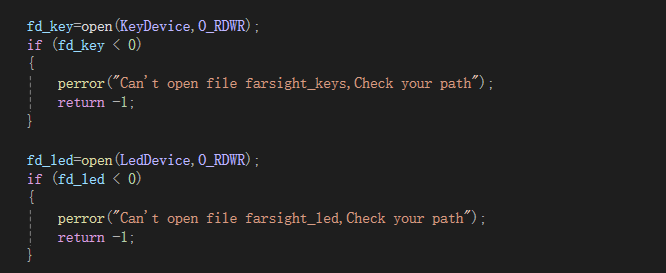
* **设计实验1-1：按键控制LED灯**

**程序设计思路如下**

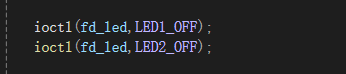
**首先初始化各种状态，文件描述符（用于操作设备）、缓冲区（存储按键数据）、LED 状态标志（记录当前亮灭状态）**



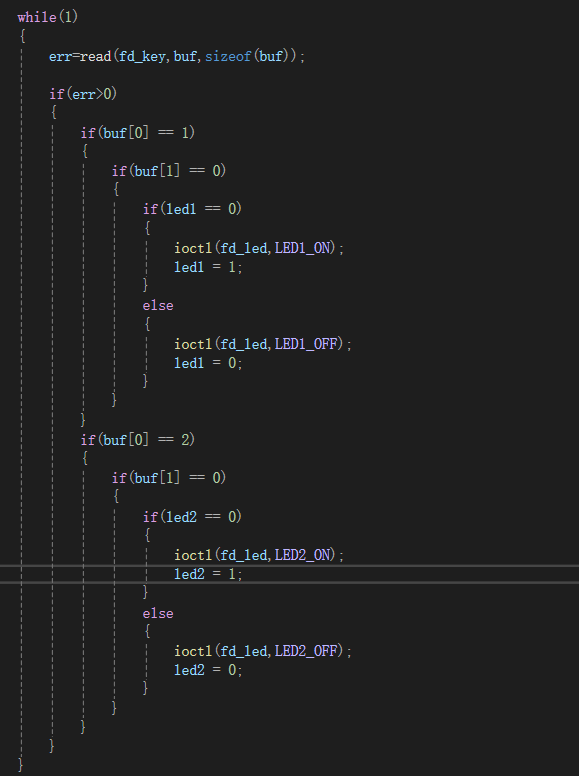
**然后打开设备文件，打开失败就打印错误信息**



**初始化led灯状态**

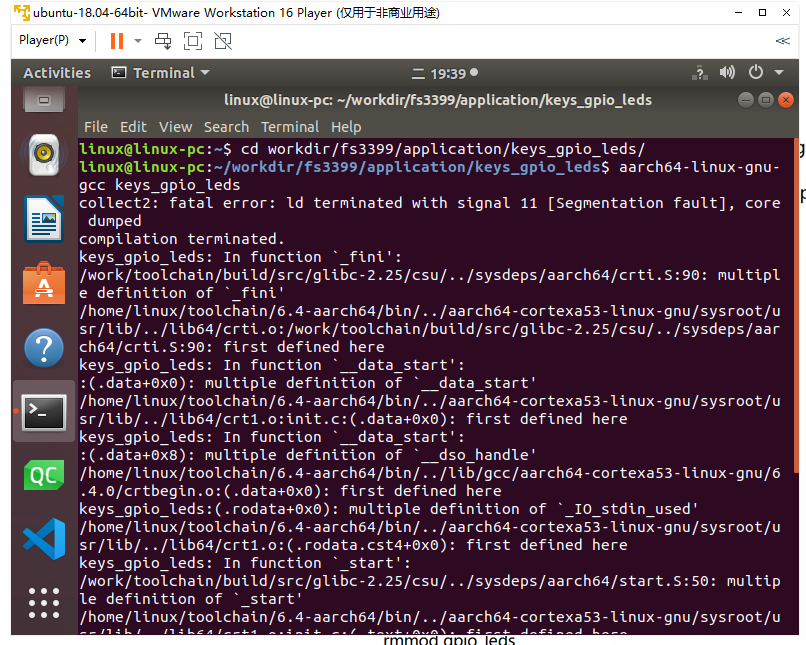


**最后就是主循环，包括按键检测与 LED 控制**



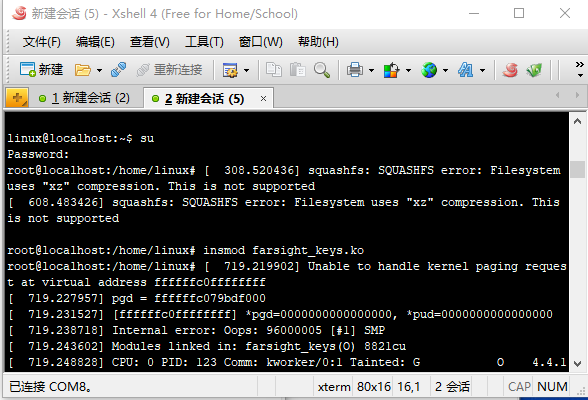
**通过检测 KEY1 和 KEY2 的按下事件，分别切换 LED1 和 LED2 的亮灭状态（按一次开，再按一次关）**

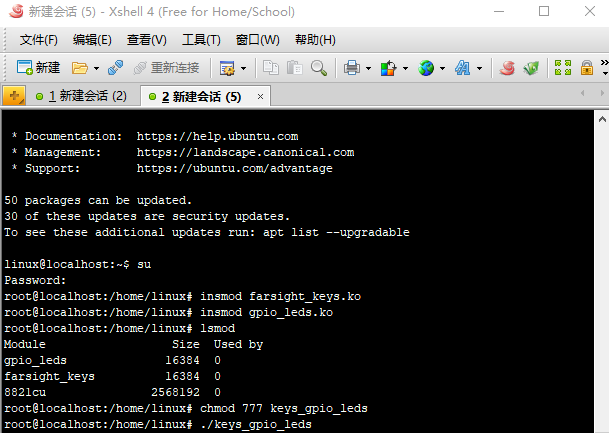
**在虚拟机上编译文件**

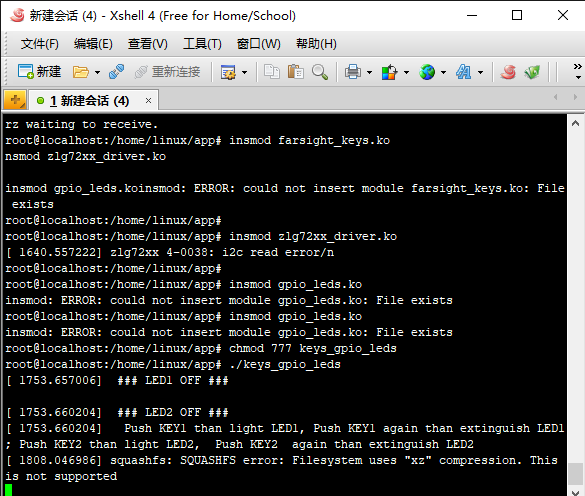
****

****

**再在xshell上运行**

****

****

****

**实验结果如下**

****

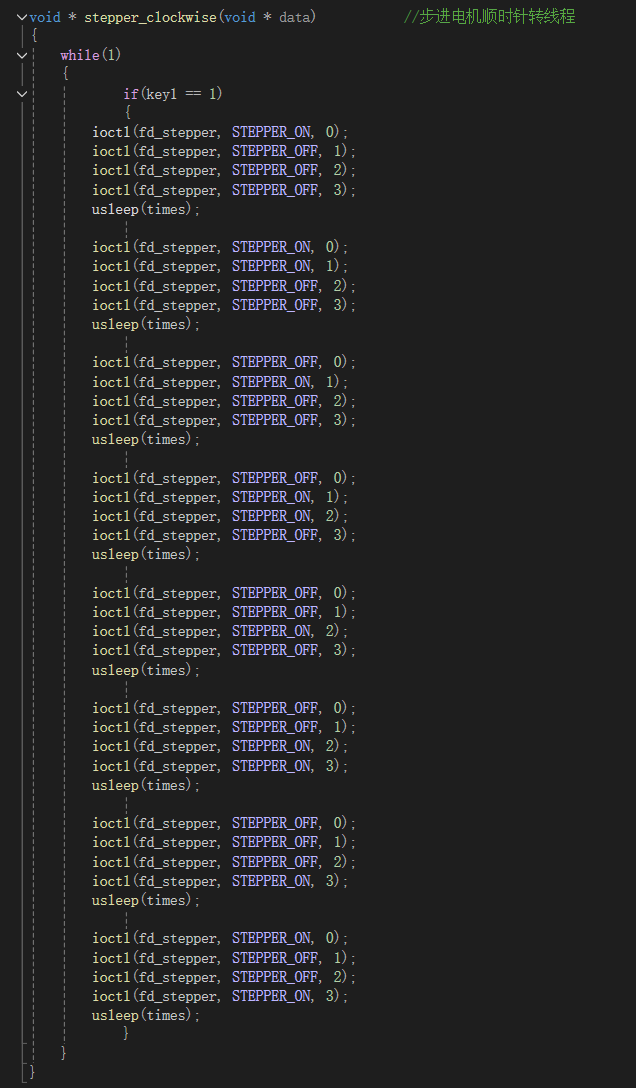
****

* **设计实验1-2：按键控制蜂鸣器**
* **设计实验1-3：按键控制蜂鸣器（底板）**
* **设计实验1-4：按键控制步进电机**
* **设计实验1-5：按键控制直流电机**
* **设计实验1-6：按键控制陀机**
* **设计实验1-7：按键控制继电器**
* **设计实验2-1：小键盘控制LED灯**
* **设计实验2-2：小键盘控制蜂鸣器**
* **设计实验2-3：小键盘控制蜂鸣器（底板）**
* **设计实验2-4：小键盘控制步进电机**

**多线程：**

**核心功能是实现步进电机顺时针和逆时针转动的控制线程**

**顺时针转动线程 stepper\_clockwise：**

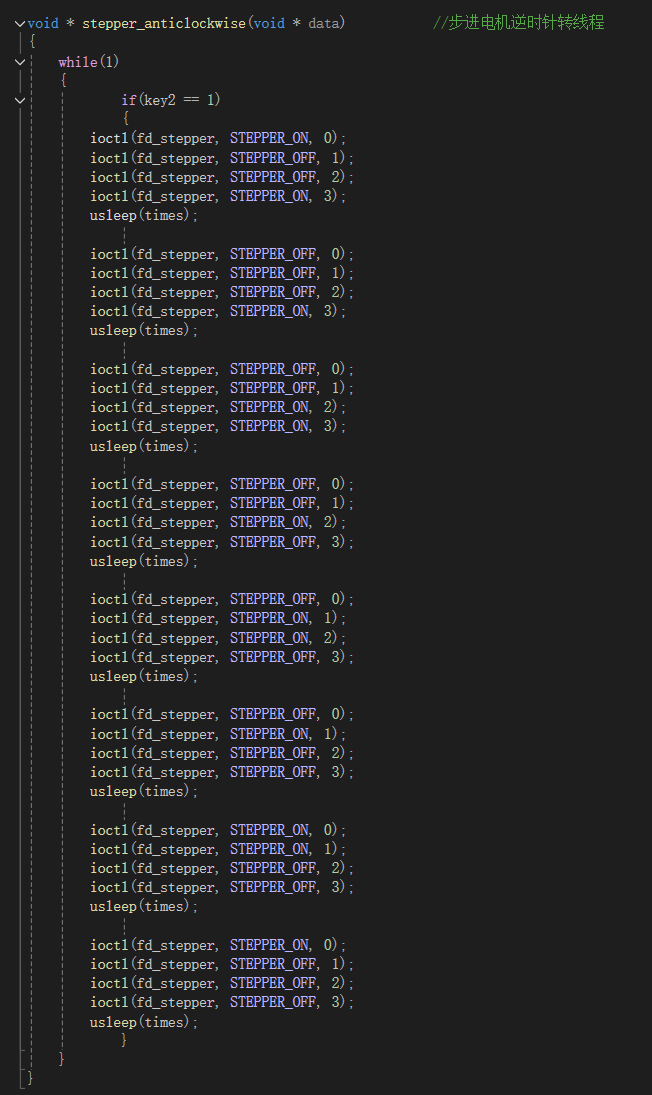


**死循环中判断key1是否为 1（若为 1 则执行转动逻辑）。**

**通过 8 组ioctl命令依次控制 4 个引脚的STEPPER\_ON/STEPPER\_OFF状态，每组操作后通过usleep(times)延时，形成连续转动。**

**每组状态对应步进电机的一个 “步”，按固定顺序切换实现顺时针转动。**

**逆时针转动线程 stepper\_anticlockwise：逻辑与顺时针类似，但引脚状态切换的顺序相反，实现逆时针转动**

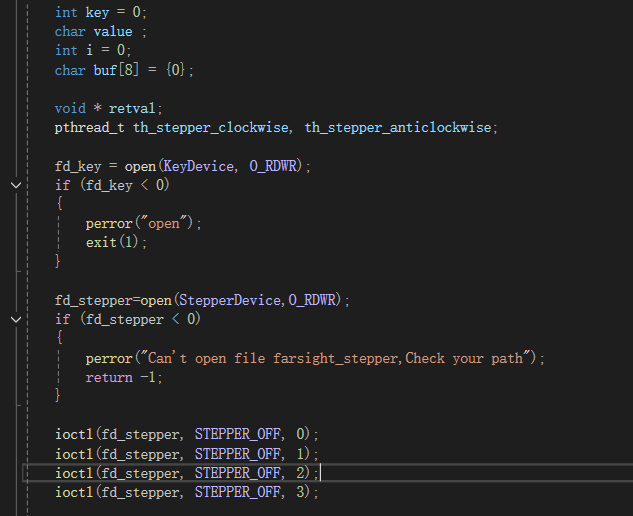


**主函数设计如下：**

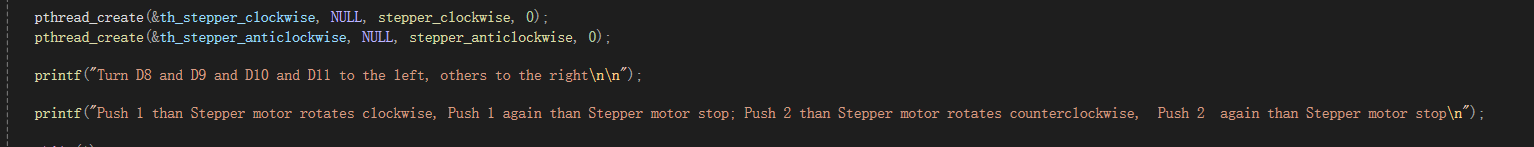
**首先是设备初始化**

**包括打开按键设备（/dev/zlg72xx）和步进电机设备（/dev/stepper），若打开失败则报错退出。**

**初始化步进电机：通过ioctl将 4 个引脚均设为STEPPER\_OFF（初始状态关闭）**



**然后创建两个线程：th\_stepper\_clockwise（顺时针）和th\_stepper\_anticlockwise（逆时针），分别执行对应转动逻辑**



**最后是按键输入处理逻辑**

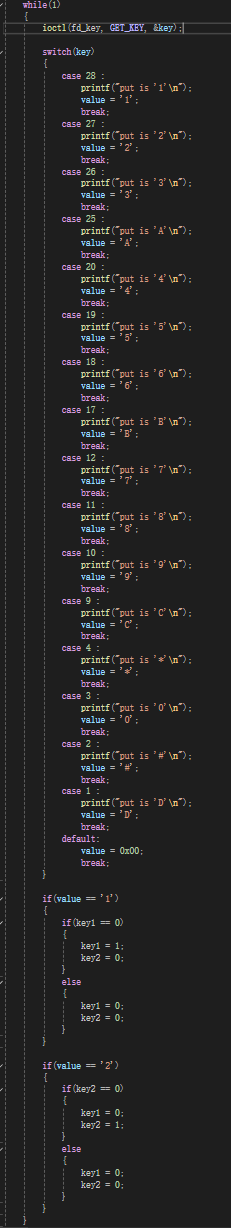
**死循环中通过ioctl(fd\_key, GET\_KEY, &key)获取按键设备的输入值。**

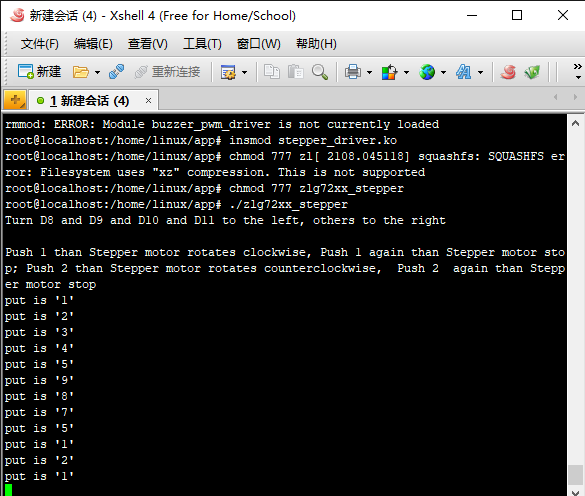
**用switch语句将按键值映射为实际字符（如key=28对应按键'1'，key=27对应'2'等）。**

**根据按键字符控制key1/key2的状态：**

**按下'1'：切换key1（0→1 或 1→0），同时重置key2=0（确保转向唯一）。**

**按下'2'：切换key2（0→1 或 1→0），同时重置key1=0**



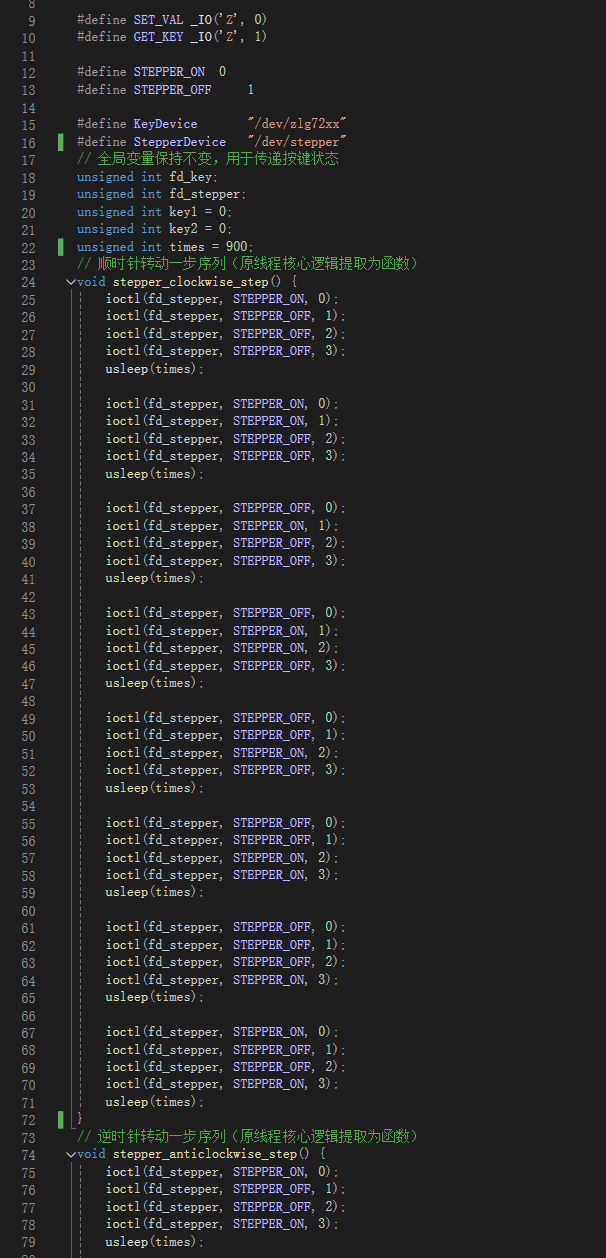
**实验结果如下**

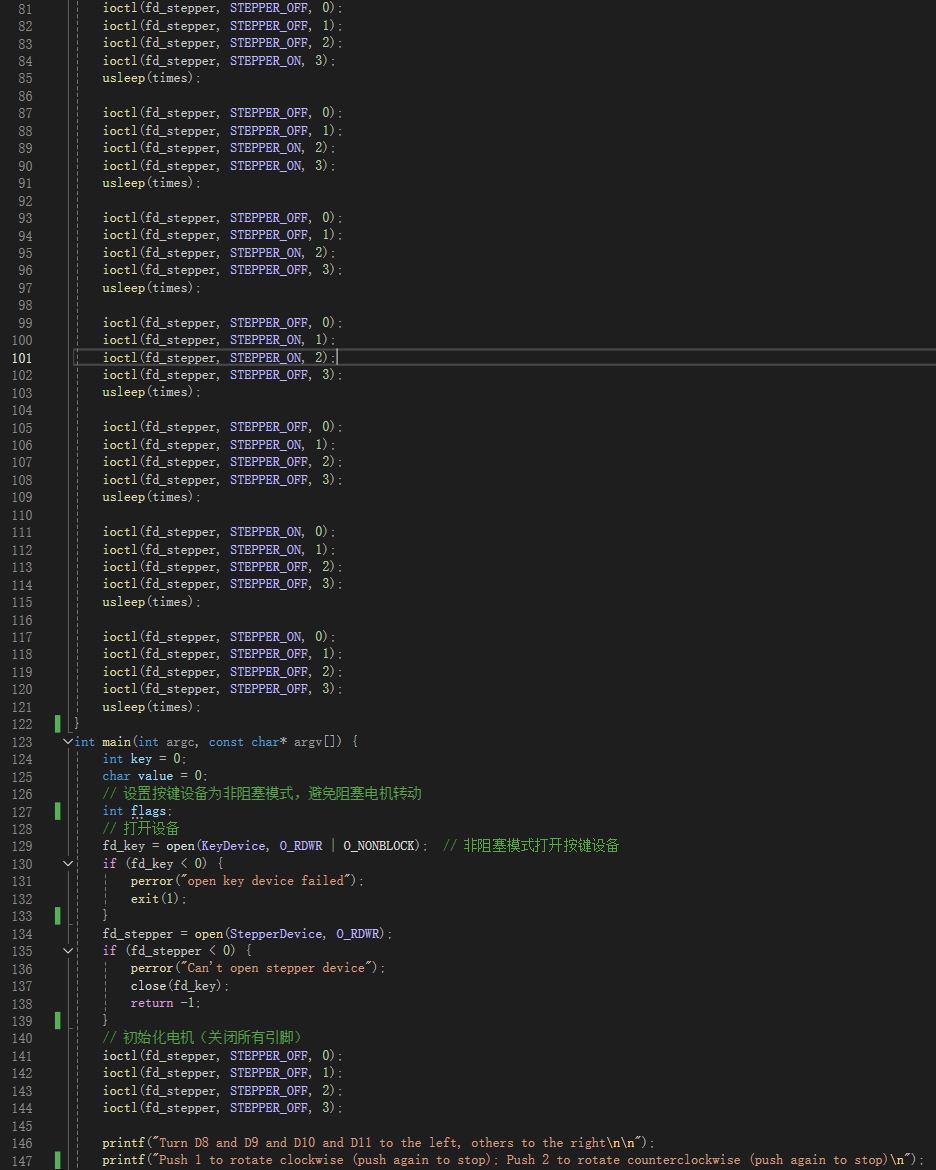
****

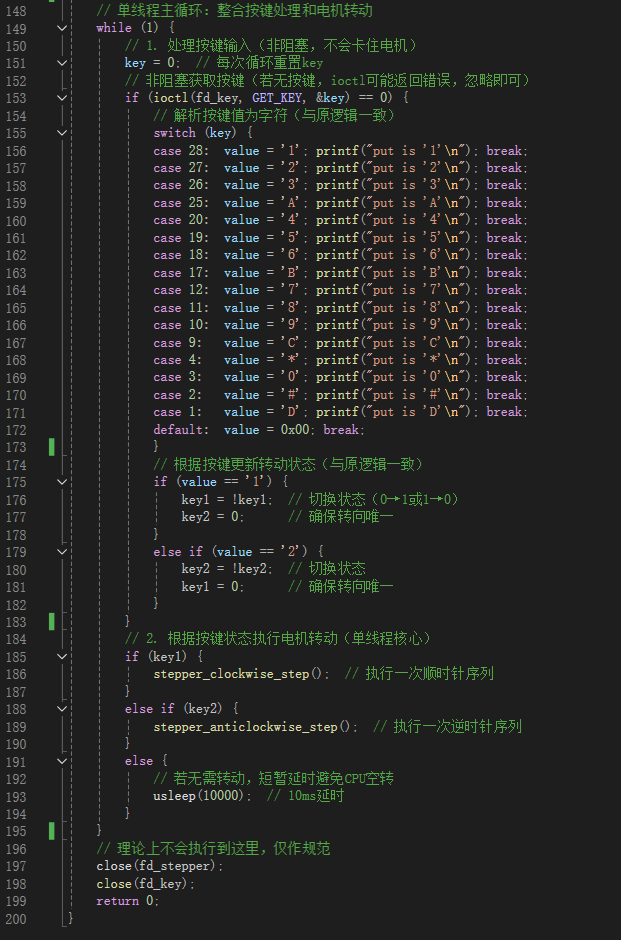
****

**单线程：**

**代码设计如下**







**移除多线程相关代码：删除了pthread库依赖、线程创建 / 连接函数，将原线程函数的核心转动逻辑提取为stepper\_clockwise\_step和stepper\_anticlockwise\_step函数（分别对应一次完整的顺 / 逆时针步序）。**

**单线程主循环设计：主循环中依次执行两项任务：**

**按键处理：以非阻塞方式读取按键（通过O\_NONBLOCK标志），避免因等待按键而阻塞电机转动。**

**电机转动：根据key1/key2的状态，调用对应的步序函数执行转动（若无需转动则短暂延时，降低 CPU 占用）。**

**保持功能一致性：**

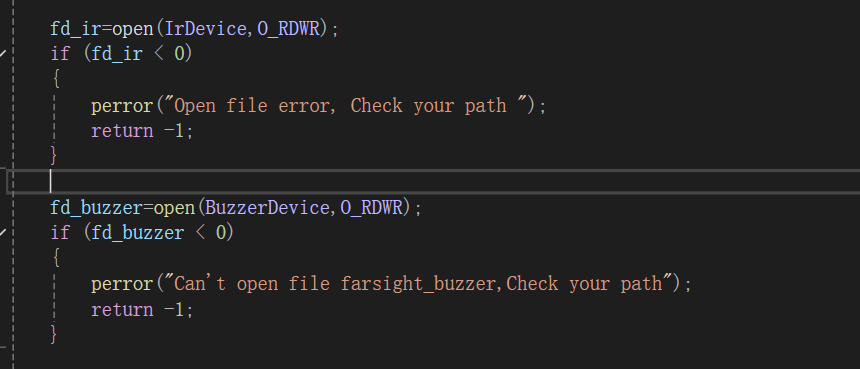
**按键控制逻辑（key1/key2的切换）与原程序完全一致。**

**电机转动的步序、延时时间（times）与原线程逻辑完全相同，确保转动效果一致。**

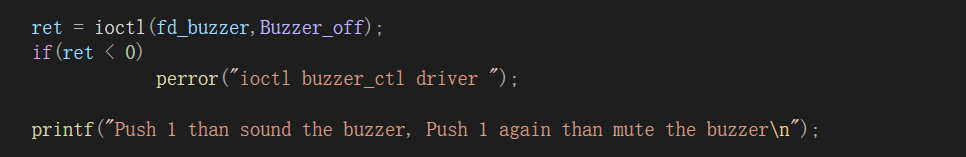
**实验结果与多线程一致**

* **设计实验2-5：小键盘控制直流电机**
* **设计实验2-6：小键盘控制陀机**
* **设计实验2-7：小键盘控制继电器**
* **设计实验3-1：红外遥控器控制LED灯**
* **设计实验3-2：红外遥控器控制蜂鸣器**

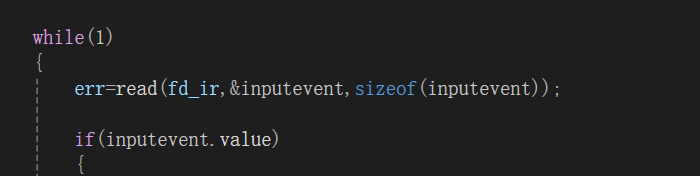
**打开红外设备（fd\_ir = open(IrDevice, O\_RDWR)）和蜂鸣器设备（fd\_buzzer = open(BuzzerDevice, O\_RDWR)），若打开失败则通过perror打印错误信息并退出。**



**初始化蜂鸣器为关闭状态（ioctl(fd\_buzzer, Buzzer\_off)），ioctl用于向设备发送控制命令。**



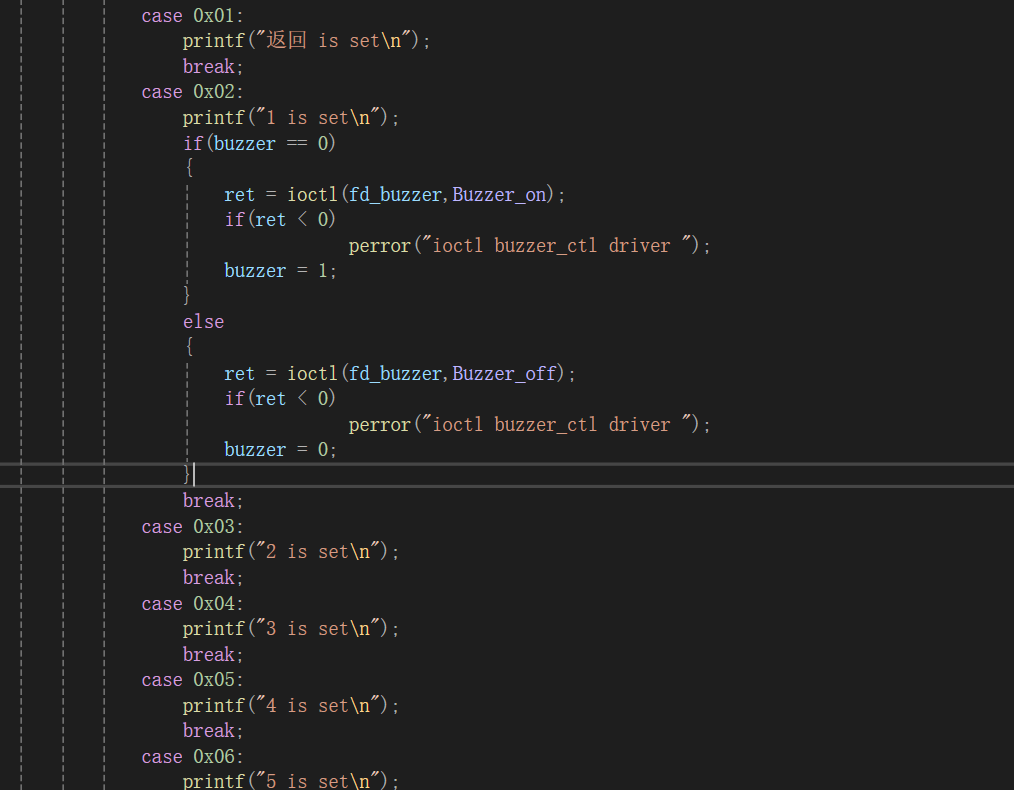
**程序进入无限循环（while(1)），通过read(fd\_ir, &inputevent, sizeof(inputevent))持续读取红外设备的输入事件。**



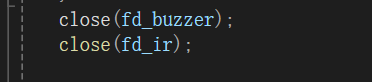
**当inputevent.value非 0 时（表示有按键按下，Linux 输入事件中value=1通常代表按键按下，value=0代表松开），根据inputevent.code（按键对应的编码）执行不同操作：**

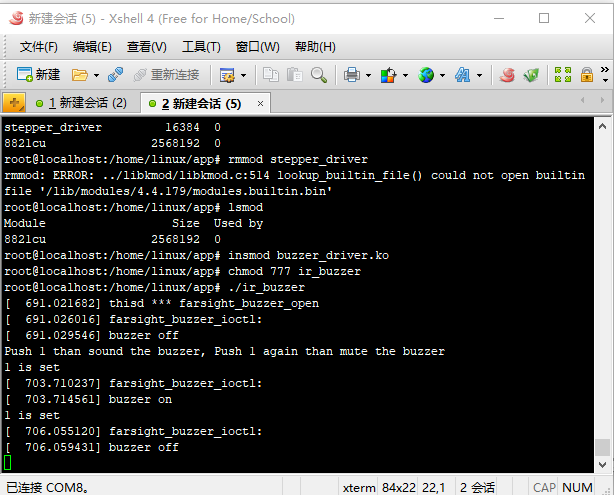
**按键1（代码0x02）：控制蜂鸣器切换状态（开→关或关→开），通过ioctl发送Buzzer\_on或Buzzer\_off命令，并更新buzzer变量记录当前状态。**

**其他按键（如0x74对应 “开关”、0x72对应 “音量减”、0x03对应 “2” 等）：仅打印按键触发信息（如printf("2 is set\n")），无其他控制逻辑。**



**理论上因无限循环不会执行到close语句，但代码仍包含关闭设备文件的逻辑（close(fd\_buzzer)和close(fd\_ir)），用于规范资源释放**



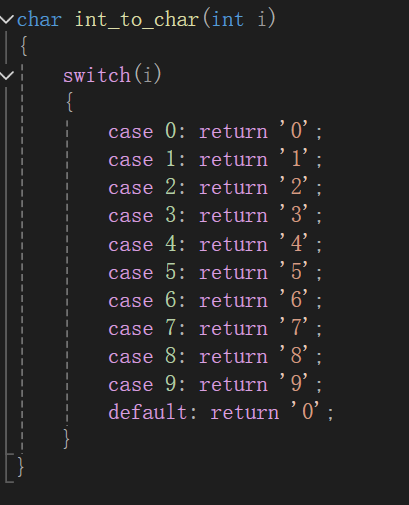
****

**实验结果见打包的视频**

* **设计实验3-3：红外遥控器控制蜂鸣器（底板）**
* **设计实验3-4：红外遥控器控制步进电机**
* **设计实验3-5：红外遥控器控制直流电机**
* **设计实验3-6：红外遥控器控制陀机**
* **设计实验3-7：红外遥控器控制继电器**
* **设计实验4-1：电子钟**

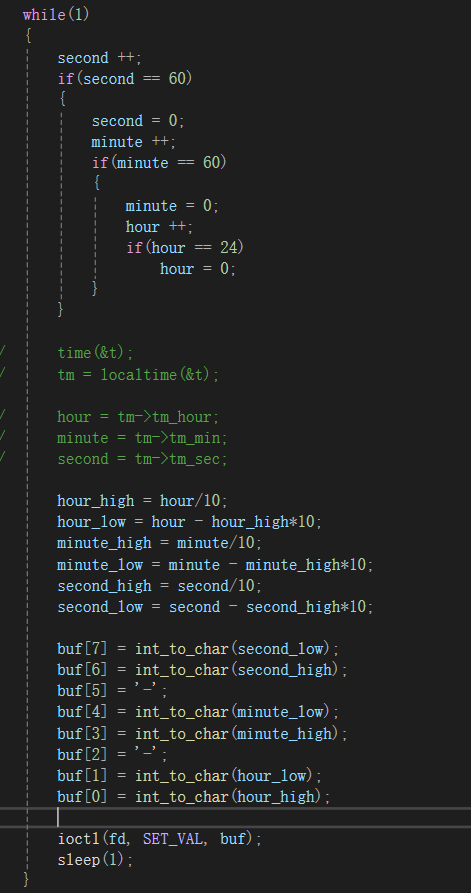
**主要包含计数和字符转化这两个功能**

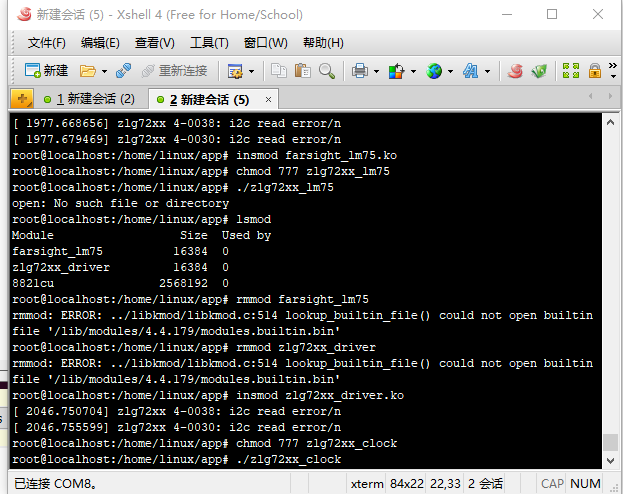
**函数intToChar将整数（0-9）转换为对应的字符（'0'-'9'），默认返回'0'（处理非 0-9 的输入）**



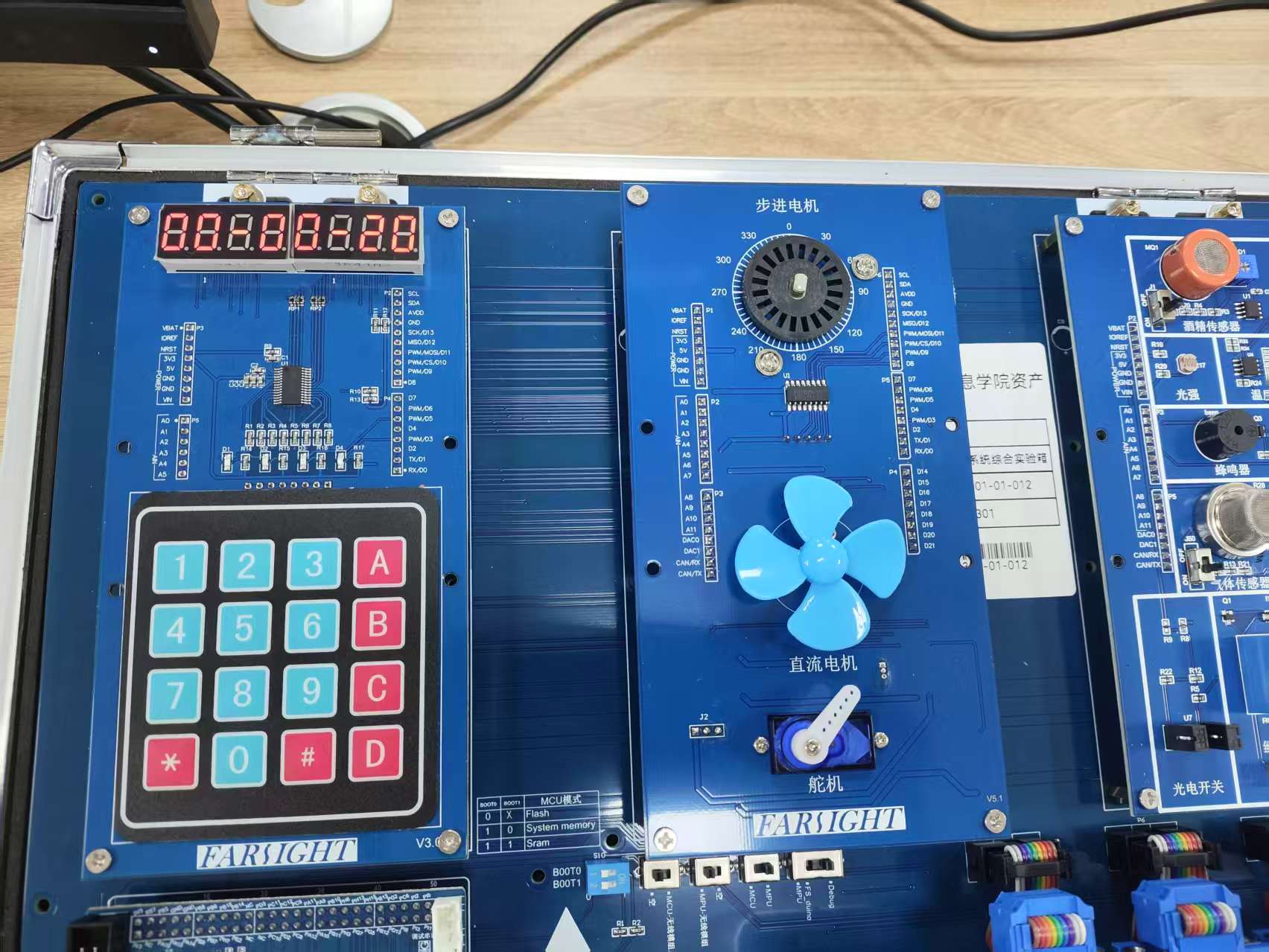
**时间模拟与显示逻辑：主循环**

**用buf[8]存储格式化的时间字符，格式为"HH-MM-SS"（8 个字符）**



****

**实验结果如下**



* **设计实验4-2：数码管显示ADC值**
* **设计实验4-3：数码管显示温度值**
* **挑战实验5-1：RS-485双机通信（1）**
* **挑战实验5-2：RS-485双机通信（2）**
* **挑战实验6-1：CAN总线双机通信（1）**
* **挑战实验6-2：CAN总线双机通信（2）**
  1. **报告提交包含：**
* **双机通信验证实验的结果展示（拍照或现场录制，并附加一定的文字说明）**
* **所完成的设计实验（挑战实验）情况（实验效果，可以通过拍照或录制视频的方式），并附上该设计实验（挑战实验）的主要代码。**
* **针对步进电机实验，需针对多线程和单线程（正常方式）进行实验结果比对展示（拍照或录制）以及分析（文字说明）**
* **将设计实验相关实验源码文件按照实验PPT的要求命名，并打包上传到平台，压缩包命名：姓名+学号+第5次设计实验。**
* **对于步进电机实验，采用单线程方式的源文件命名为XX\_stepper\_single.c，** 
  + **例如：实验1-4：按键控制步进电机，多线程源文件按照PPT命名为keys\_stepper.c, 单线程则命名为keys\_stepper\_single.c，其他实验以此类推。**