厦门大学

《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 12 月

第 8 次实验　Qt 与 Android 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）

2. 实验内容

（1） 必做实验：请从 10 个 Qt 设计实验（1-1、1-2、1-3、1-4、2-1、2-2、2-3、2-4、3-1、3-2）中选择 3 个，被控制设备不重复。具体即从 1-1，1-2，1-3，1-4 中选 1 个，2-1，2-2，2-3，2-4 中选 1 个，3-1，3-2 中选 1 个。

（2） 必做实验：请从 2 个 Android 设计实验（4-1、4-2）任选 1 个。

（3） 挑战实验：请从 2 个 Android 设计实验（5-1、5-2）任选 1 个。完成挑战实验的，本次实验成绩加 10 分。

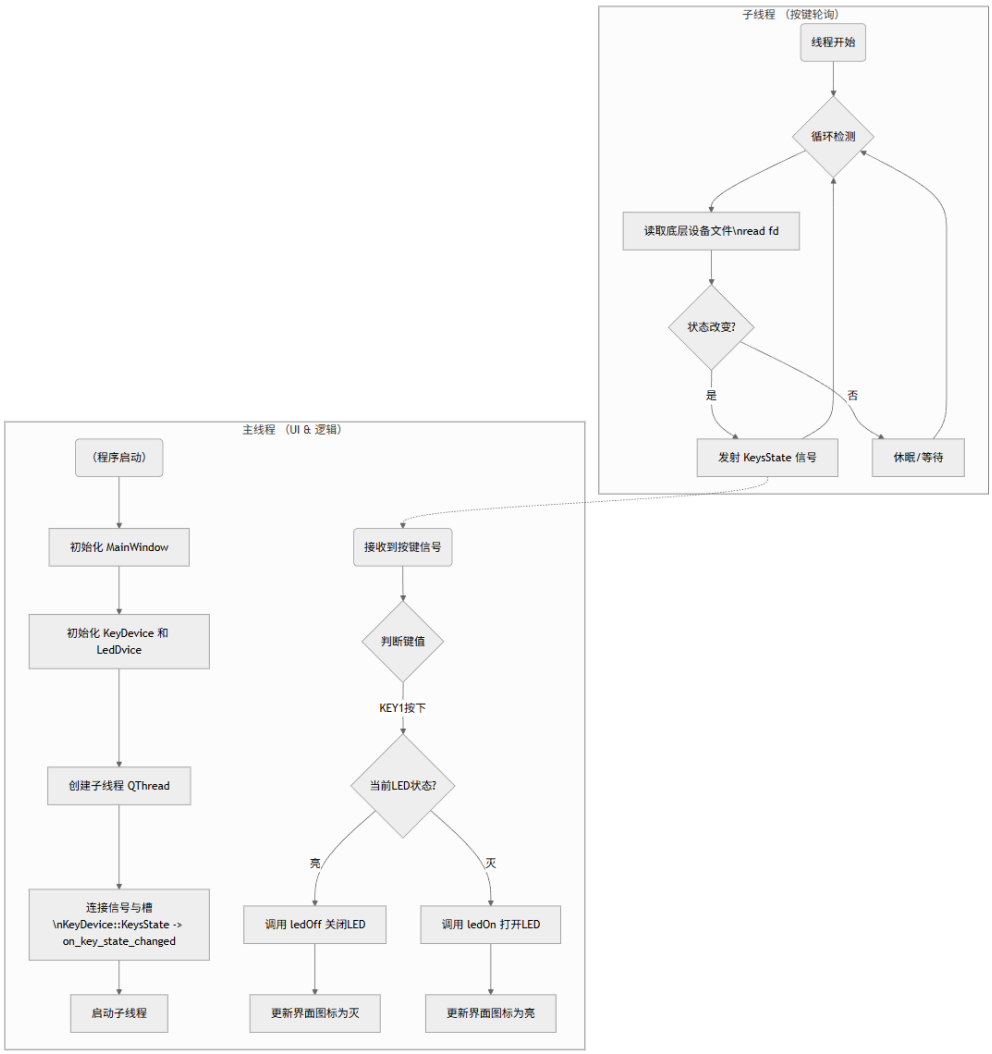
附：设计实验清单

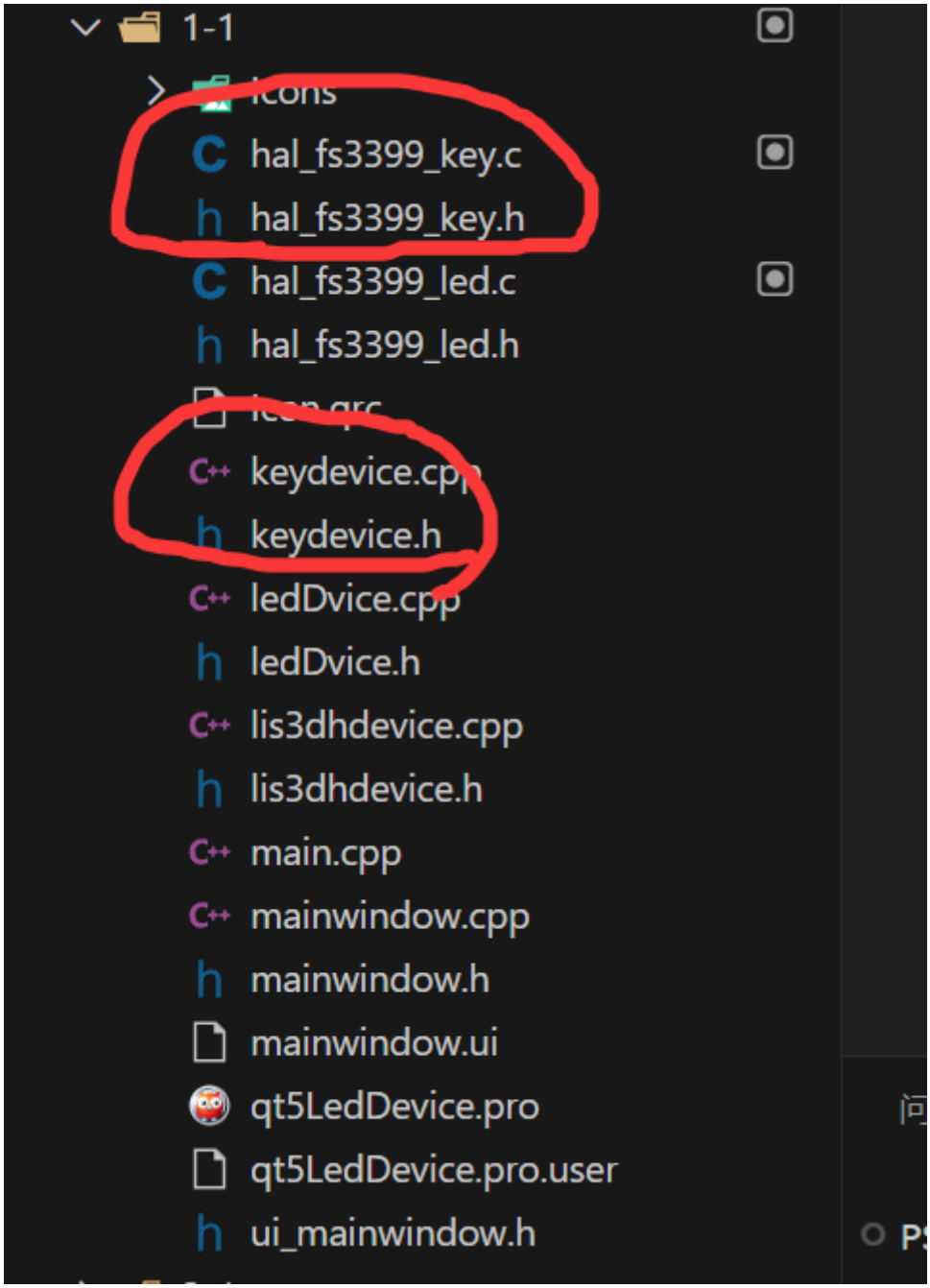
- 第一部分：Qt 设计实验
  - 设计实验 1-1：按键控制 LED 灯

设计思路

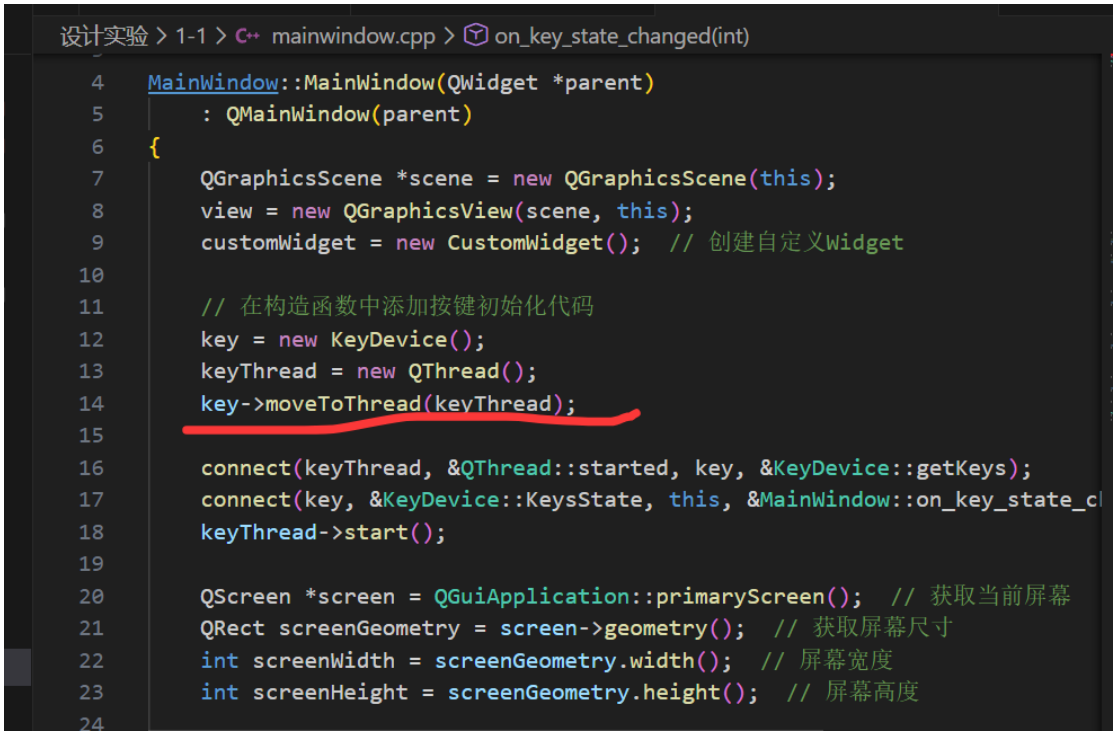
在原有的 qt5LedDevice 工程基础上，引入 qt5KeyDevice 工程中的按键驱动文件

流程图如下：

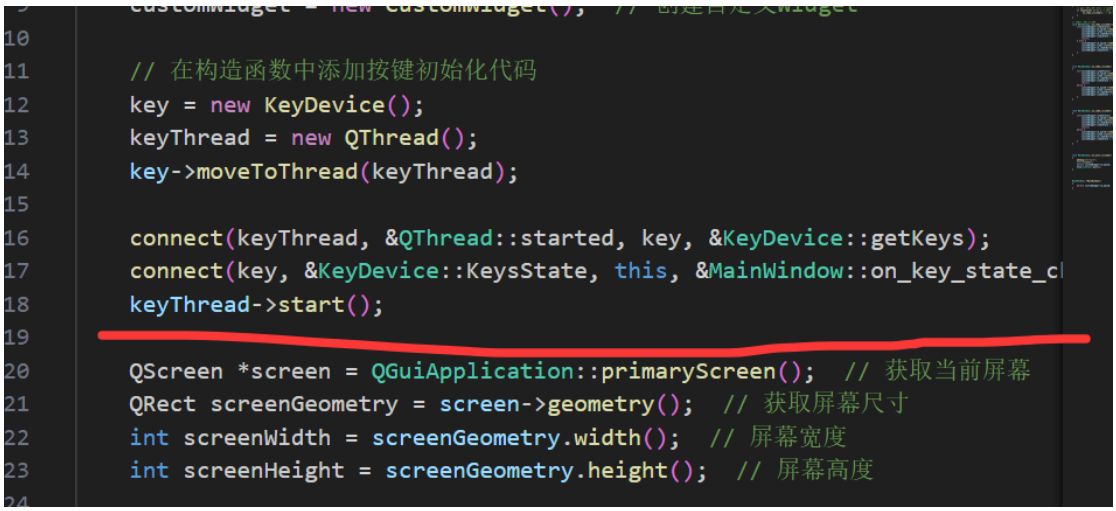

子线程 （按键轮询）
线程开始
循环检测
读取底层设备文件\nread fd
状态改变?
是
否
发射 KeysState 信号
休眠/等待
主线程 （UI & 逻辑）
（程序启动）
初始化 MainWindow
初始化 KeyDevice 和 LedDvice
创建子线程 QThread
连接信号与槽 \nKeyDevice::KeysState -> on_key_state_changed
启动子线程
接收到按键信号
判断键值
KEY1按下
当前LED状态?
亮
灭
调用 ledOff 关闭LED
调用 ledOn 打开LED
更新界面图标为灭
更新界面图标为亮



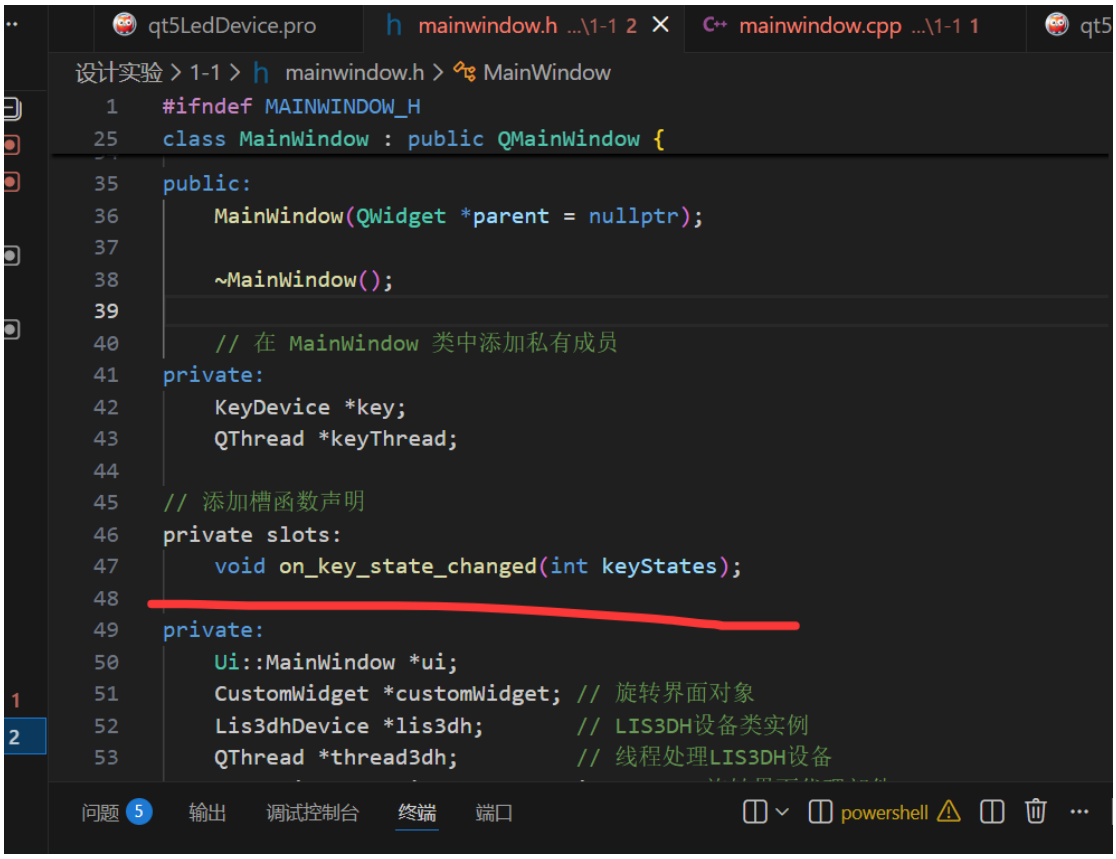
在 **mainwindow.cpp** 中加入按键的初始化



连接



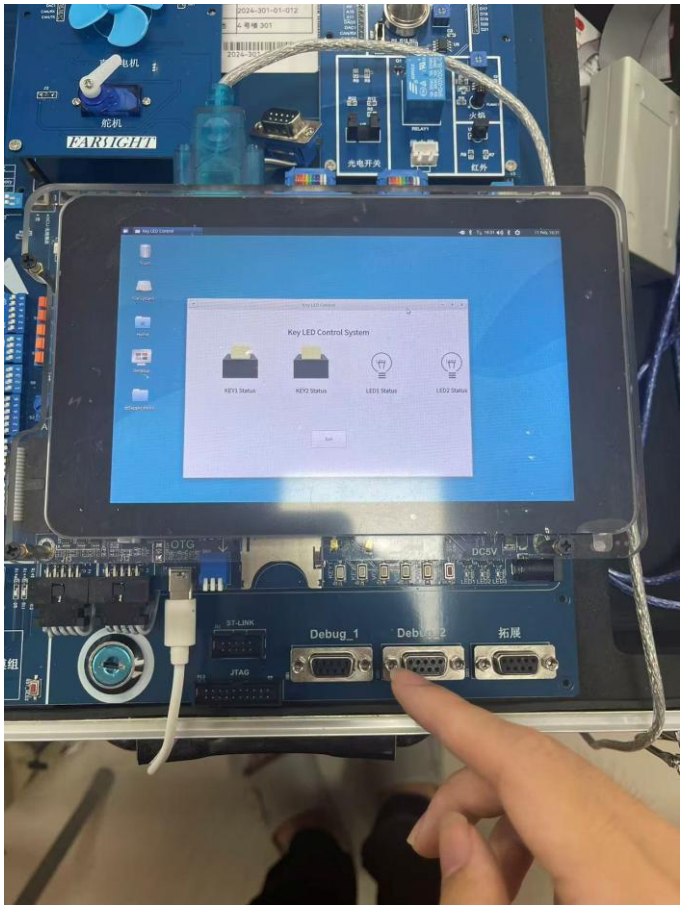
在 **mainwindow.h** 中添加按键槽函数声明

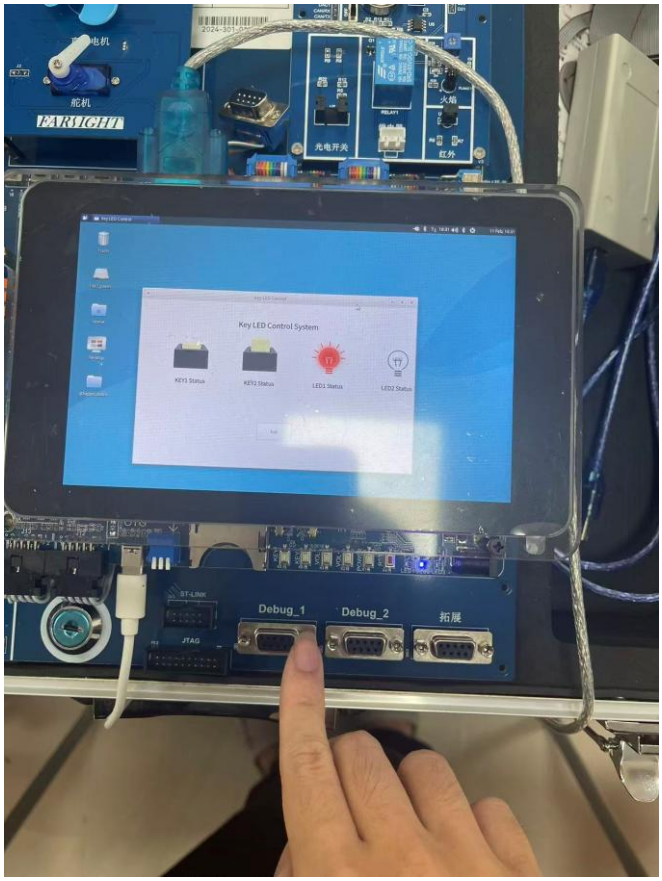


添加对应的槽函数实现

```
// 添加槽函数实现
void MainWindow::on_key_state_changed(int keyStates)
{
    // 假设 KEY1_ON 定义为 1 (具体值请参考 keydevice.h)
    if (keyStates == 1) { // KEY1 按下
        on_led1_clicked(); // 调用现有的LED切换函数
    }
}
```

实验结果如下


舵机
FARSIGHT
光电开关
红外
Key LED Control System
KEY1 Status
KEY2 Status
LED1 Status
LED2 Status
OTG
DC5V
ST-LINK
JTAG
Debug_1
Debug_2
拓展

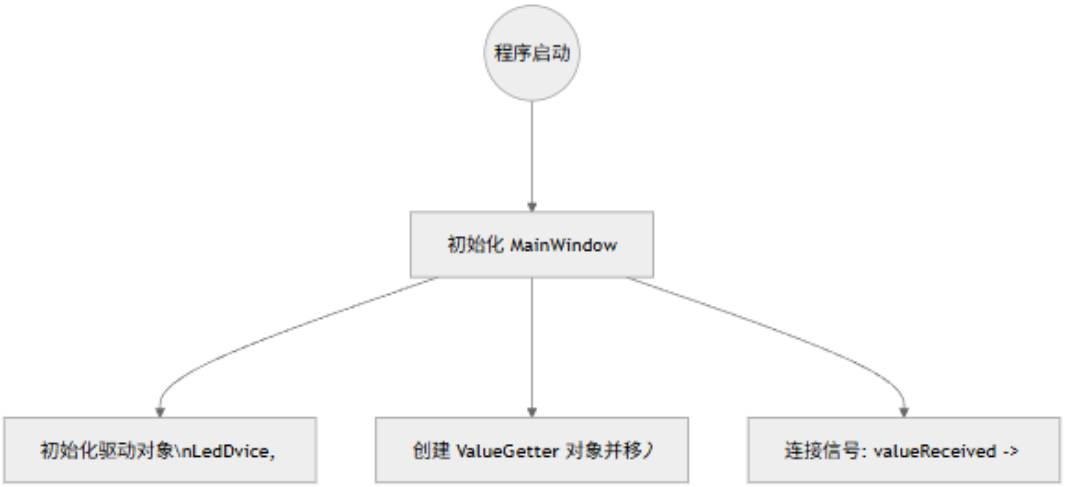

舵机
FARSIGHT
光电开关
火焰
红外
Key LED Control System
KEY1 Status
KEY2 Status
LED1 Status
LED2 Status
ST-LINK
JTAG
Debug_1
Debug_2
拓展

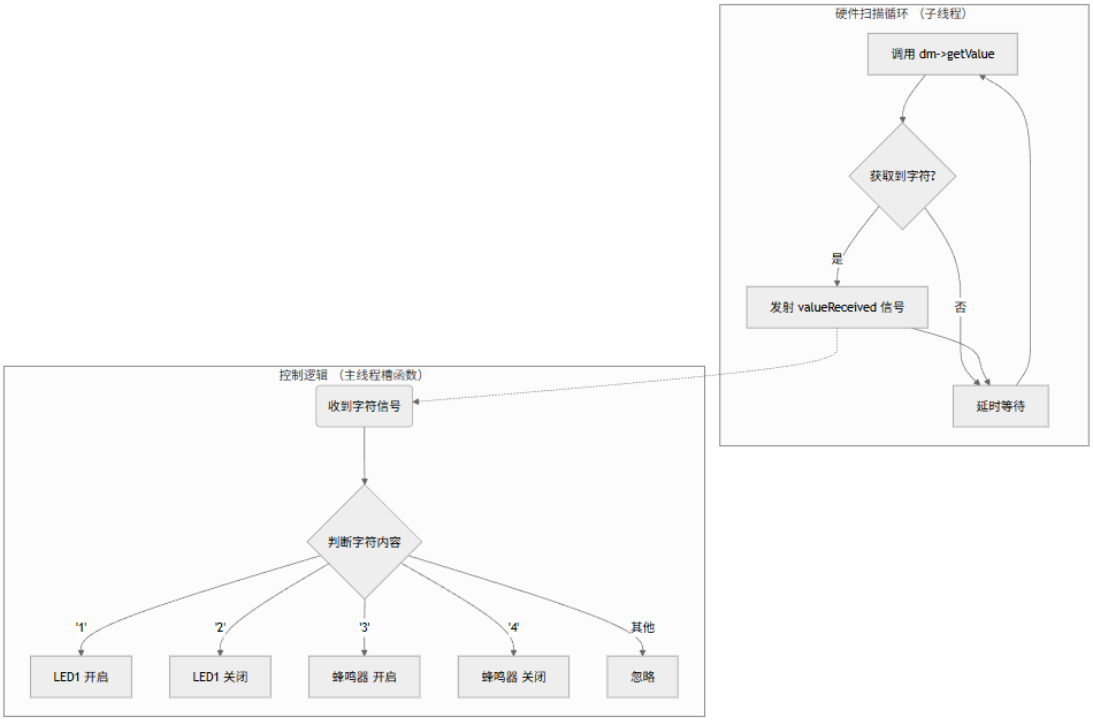
- **设计实验 1-2：按键控制蜂鸣器**
- **设计实验 1-3：按键控制直流电机**

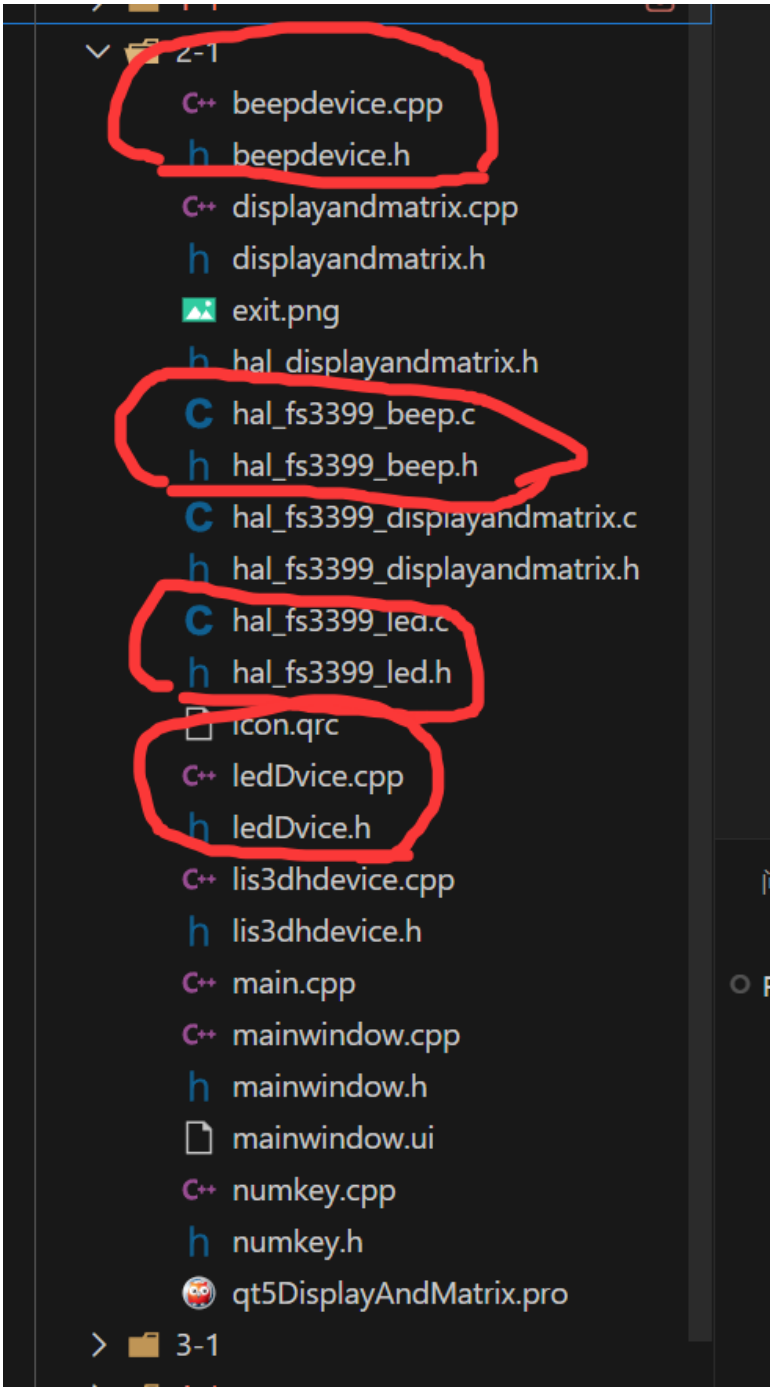
■ 设计实验 1-4：按键控制步进电机

■ 设计实验 2-1：小键盘控制 LED 灯和蜂鸣器

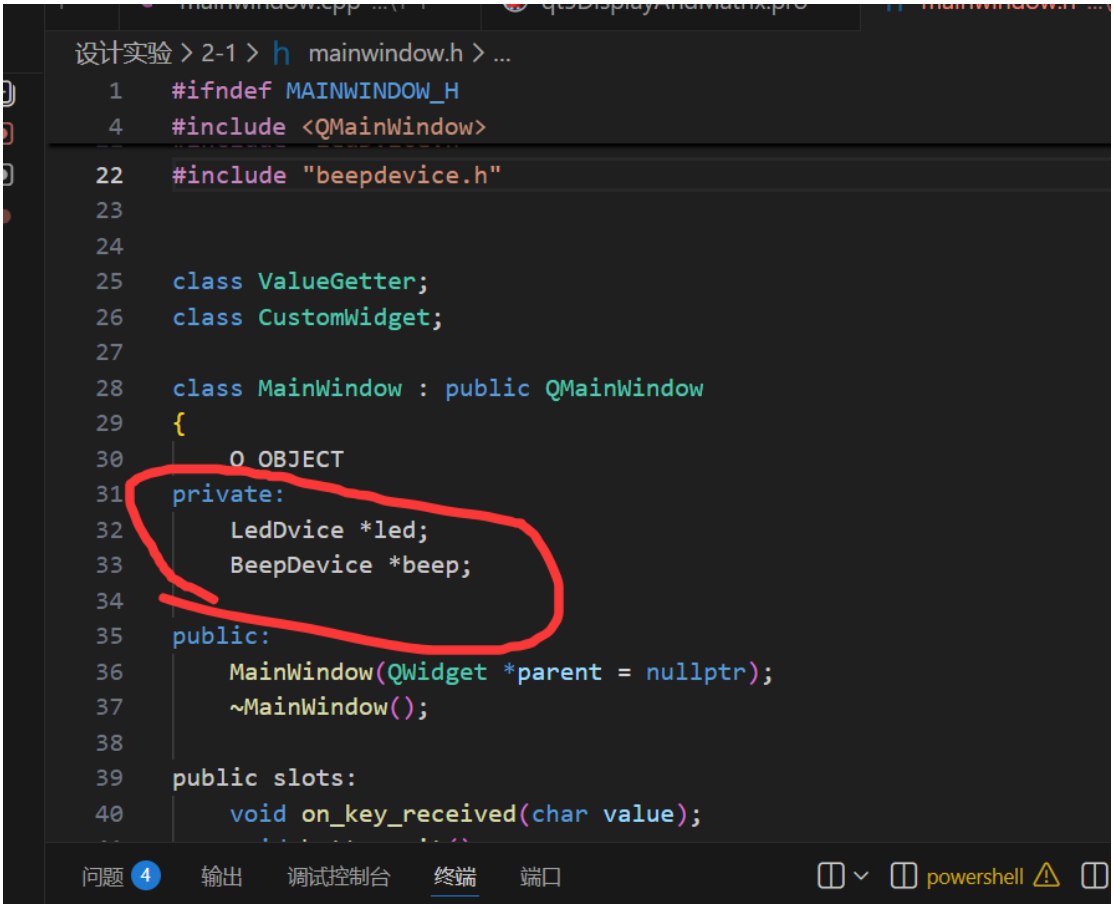
思路：在 qt5DisplayAndMatrix 项目中，利用数码管/矩阵键盘驱动读取键值，并引入 LED 和蜂鸣器驱动进行控制。
从 qt5LedDevice 复制： ledDvice.cpp, ledDvice.h, hal_fs3399_led.c, hal_fs3399_led.h
从 qt5BeepDevice 复制： beepdevice.cpp, beepdevice.h, hal_fs3399_beep.c, hal_fs3399_beep.h
放入 qt5DisplayAndMatrix 文件夹。
流程图如下



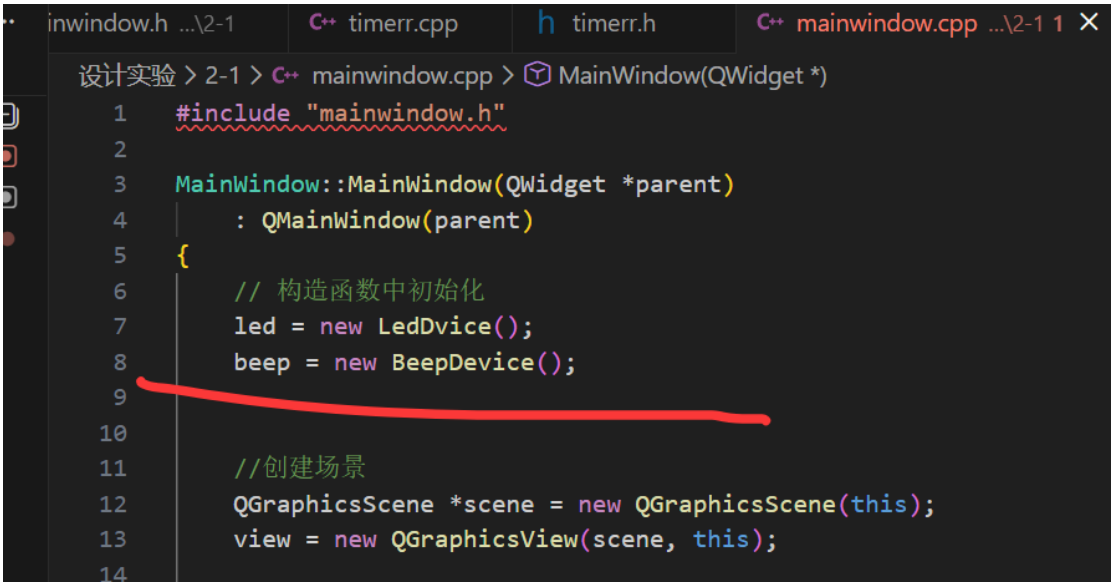

硬件扫描循环 （子线程）
调用 dm->getValue
获取到字符?
是
否
发射 valueReceived 信号
延时等待
控制逻辑 （主线程槽函数）
收到字符信号
判断字符内容
'1'
'2'
'3'
'4'
其他
LED1 开启
LED1 关闭
蜂鸣器 开启
蜂鸣器 关闭
忽略



在 Mainwindow 类中添加蜂鸣器和 led 灯成员



在 **mainwindow.cpp** 中初始化



在 **mainwindow.h** 中添加 **on_key_received** 槽函数声明

```
#ifndef MAINWINDOW_H
class MainWindow : public QMainWindow

public:
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

public slots:
    void on_key_received(char value);
    void buttonexit();

    void numberButtonClicked(int num);

    void letterButtonClicked();

    void stopGettingValues();

private:
    QLCDNumber *Lcd;
    DisplayandMatrix *dm;
    QPushButton *m_exitButton1;
    QString accumulatedStr;
```

实现对应的槽函数

```
void MainWindow::on_key_received(char value)
{
    // 根据接收到的按键值控制 LED 和 蜂鸣器
    // 假设 '1' 开灯, '2' 关灯, '3' 开蜂鸣器, '4' 关蜂鸣器
    {

    if (value == '1') {
        led->ledOn(LED1);
    } else if (value == '2') {
        led->ledOff(LED1);
    } else if (value == '3') {
        beep->beepOn();
    } else if (value == '4') {
        beep->beepOff();
    }
}   }
    this->close();
    QApplication::quit();
}
```

问题 3　输出　调试控制台　终端　端口　powershell

实验结果如下

- 设计实验 2-2：小键盘控制 LED 灯和蜂鸣器
- 设计实验 2-3：小键盘控制直流电机
- 设计实验 2-4：小键盘控制步进电机

- 设计实验 3-1：电子钟

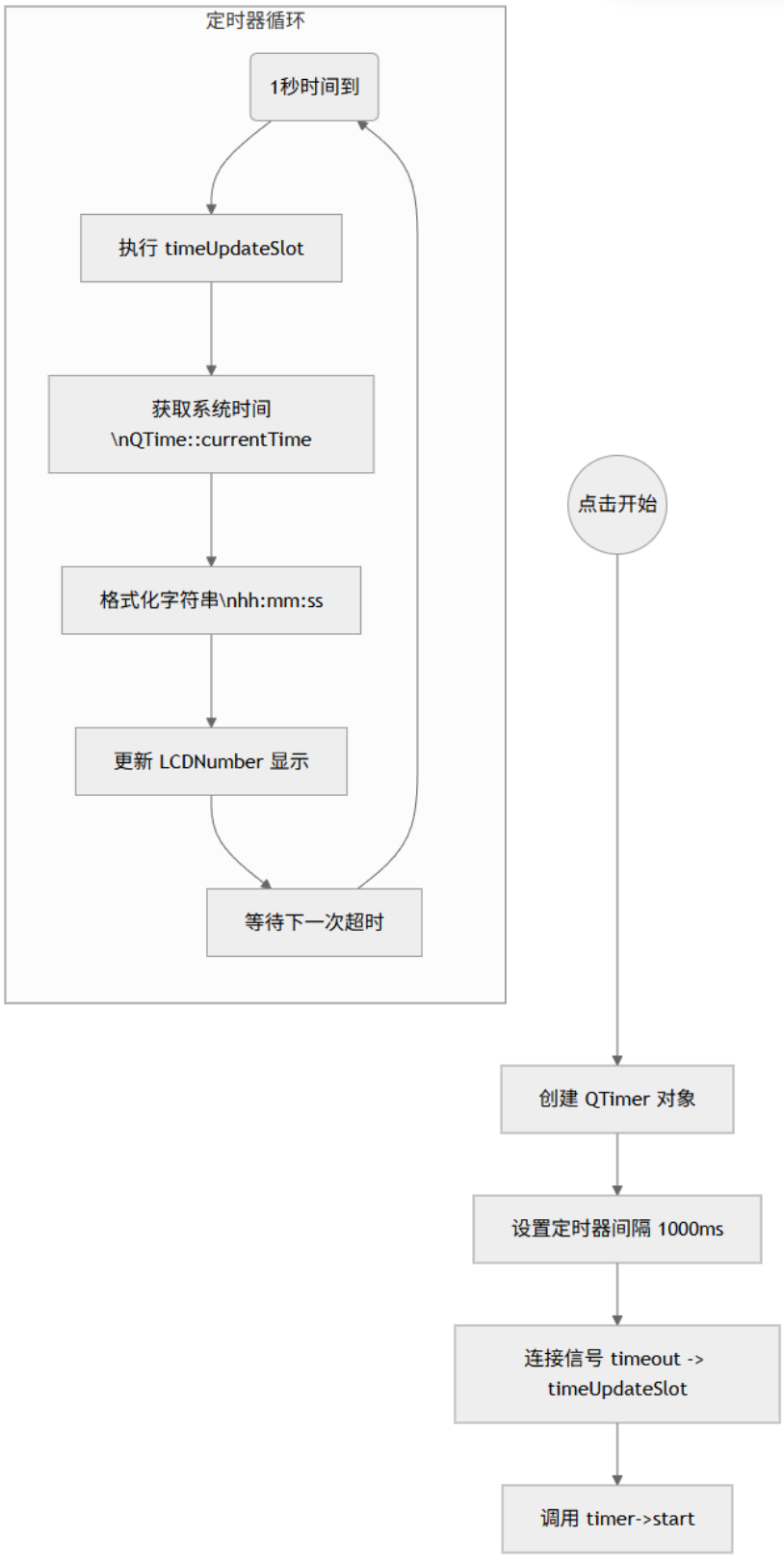
思路：修改 Stopwatch (第一部分) 项目，将其从计时器改为显示当前系统时间。

```
    delete ui;
}
void Timerr::start()
{
    timer = new QTimer(this);
    timer->setSingleShot(false);
    timer->start(1000);
    //超时发出信号
    connect(timer,SIGNAL(timeout()),this,SLOT(timeUpdateSlot()));
    timeUpdateSlot();
    ui->lcdNumber->show();
}
void Timerr::stop()
```

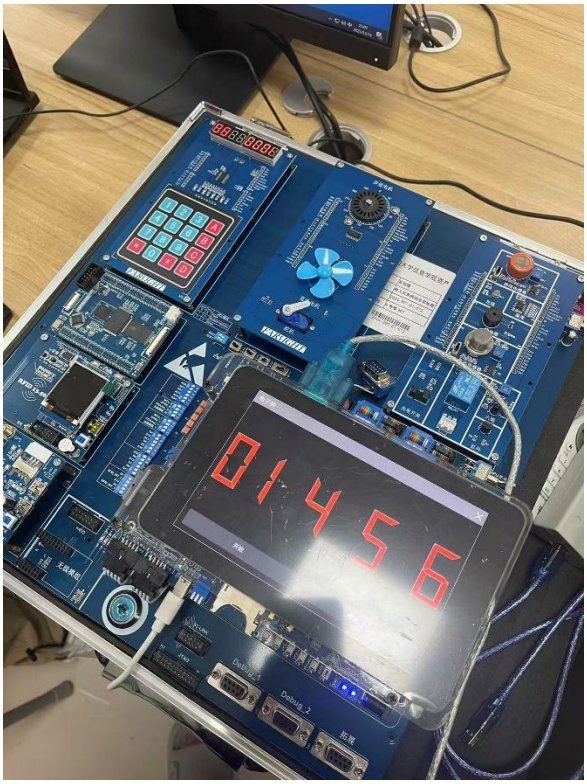
修改显示逻辑，将原来的手动计数逻辑替换为获取系统当前时间并格式化显示。

```
29  }
30  //超时处理槽函数
31  void Timerr::timeUpdateSlot()
32  {
33      // 获取当前系统时间
34      QTime currentTime = QTime::currentTime();
35      // 格式化为 HH:mm:ss
36      time = currentTime.toString("hh:mm:ss");
37
38      ui->lcdNumber->display(time);
39  }
40
```

流程图如下


定时器循环
1秒时间到
执行 timeUpdateSlot
获取系统时间
\nQTime::currentTime
格式化字符串\nhh:mm:ss
更新 LCDNumber 显示
等待下一次超时
点击开始
创建 QTimer 对象
设置定时器间隔 1000ms
连接信号 timeout ->
timeUpdateSlot
调用 timer->start

实验结果如下



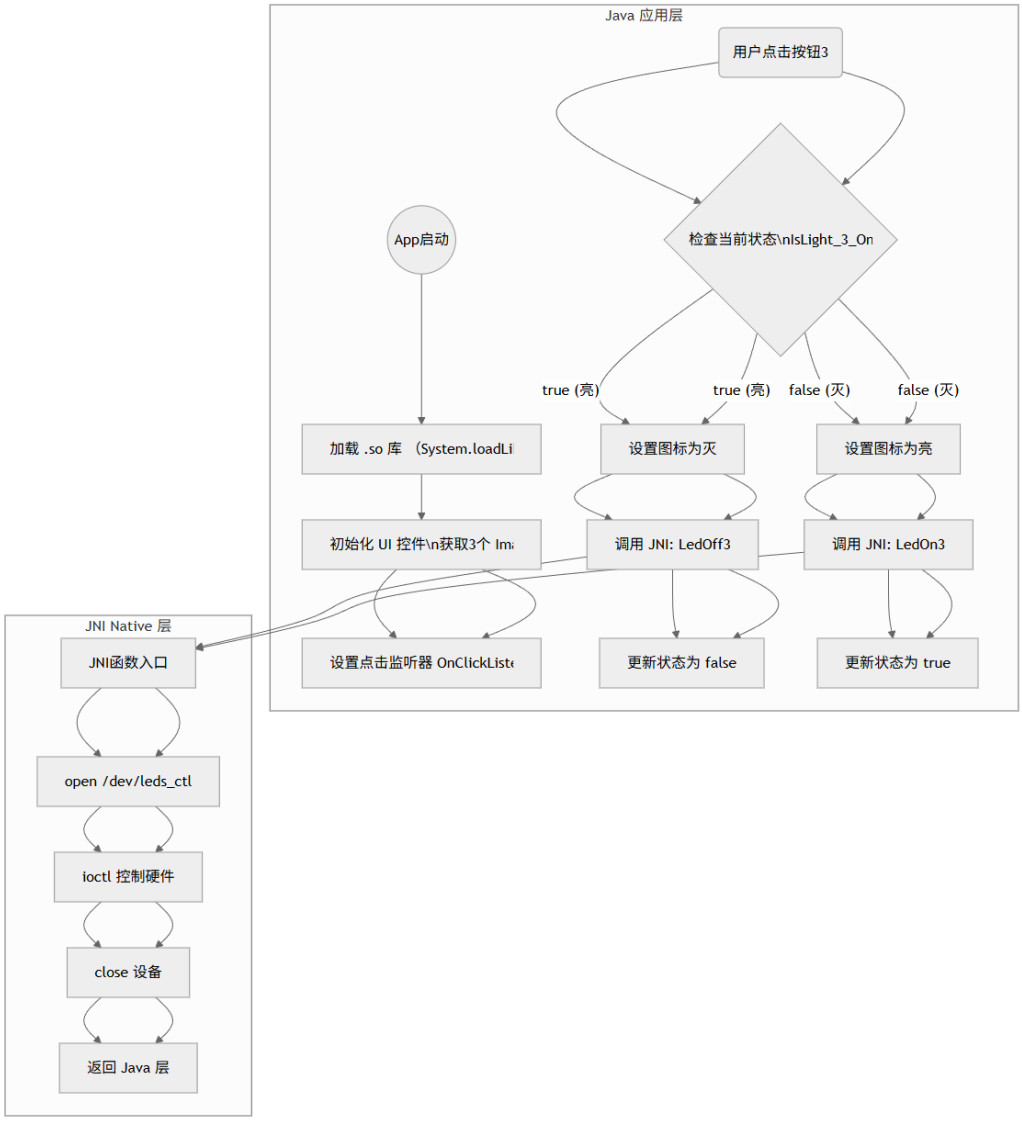
■ 设计实验 3-2：在数码管上显示 ADC 值

第二部分：Android 设计实验

■ 设计实验 4-1：3 个 LED 灯

设计思路：扩展原有的 Android LED 实验（仅控制 2 个灯），使其能控制 3 个灯

流程图如下



JNI 层修改 (native-lib.cpp)：

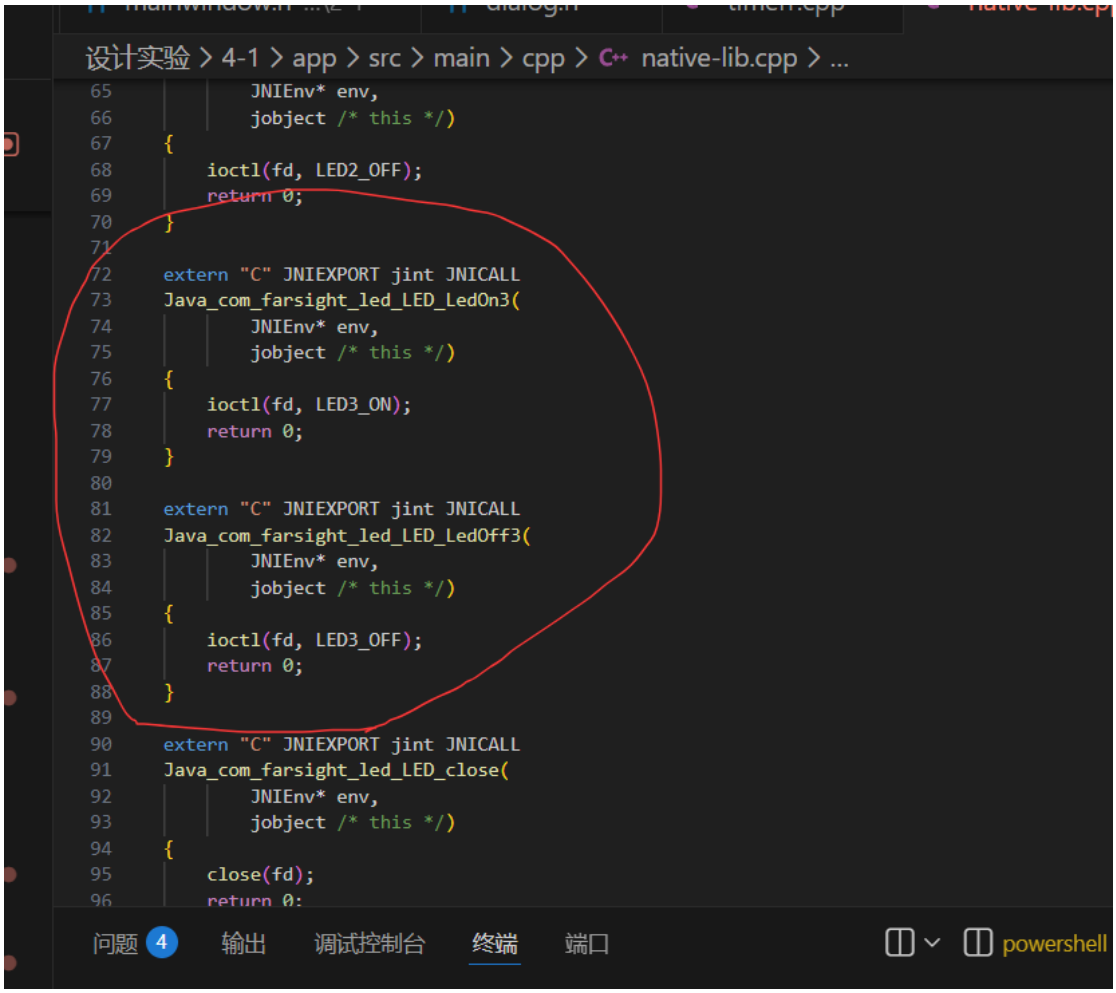
修改宏定义，正确映射硬件 GPIO：LED1('y'), LED2('z'), LED3('x')。

增加 LedOn3 和 LedOff3 的 C++ 实现函数，调用 ioctl 控制设备文件。

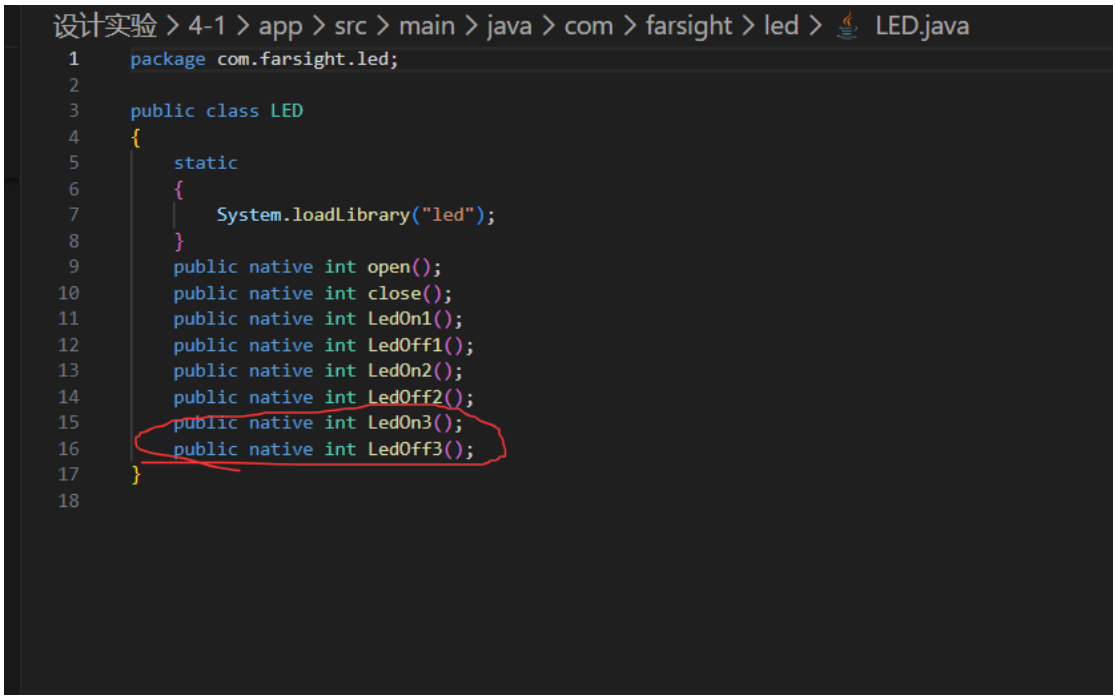
硬件映射

```
9
10  #define LED1_ON    _IO('y',1)
11  #define LED1_OFF   _IO('y',0)
12  #define LED2_ON    _IO('z',1)
13  #define LED2_OFF   _IO('z',0)
14  #define LED3_ON    _IO('x',1)
15  #define LED3_OFF   _IO('x',0)
16
```

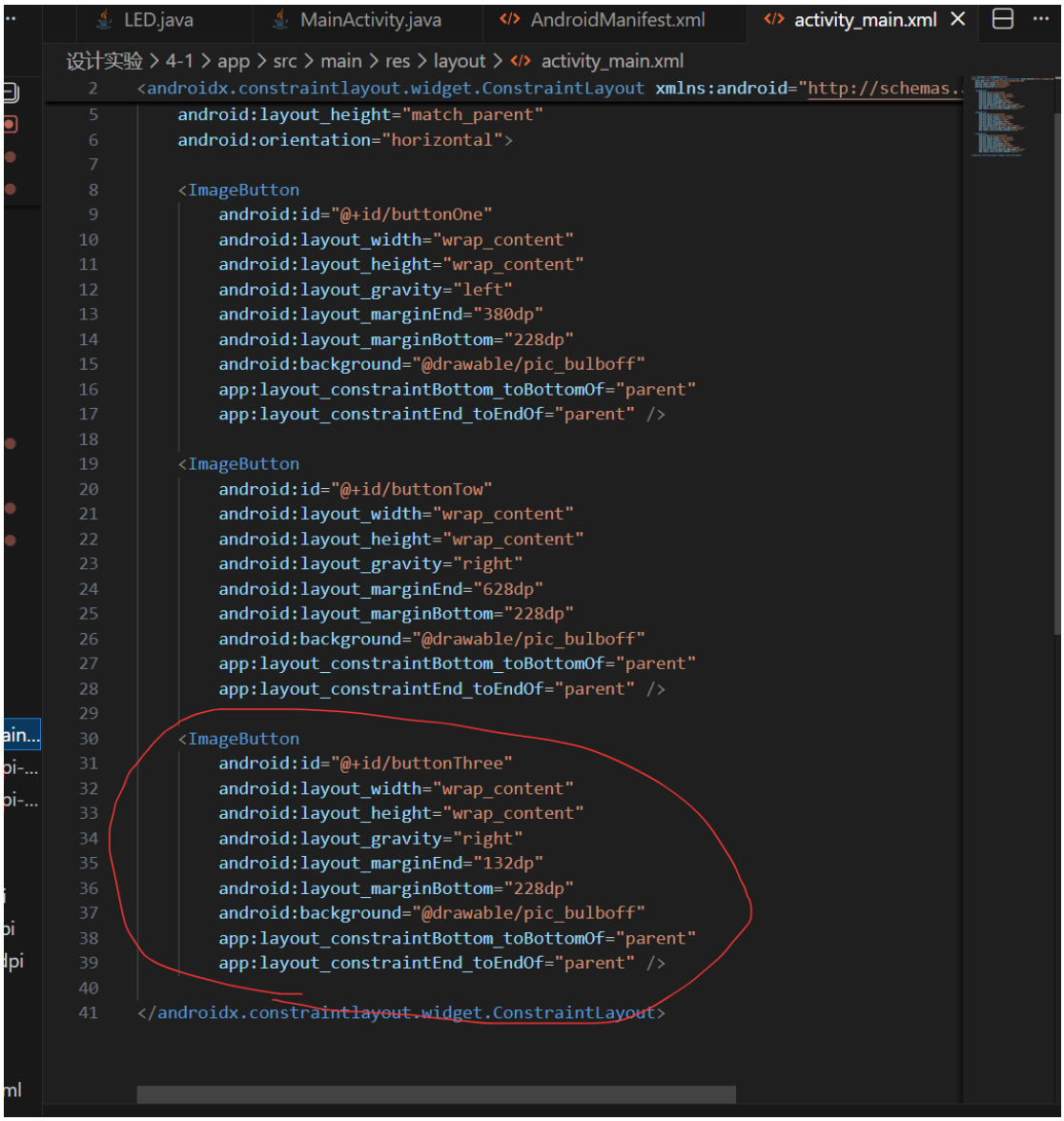
增加 led3 的实现函数



Java 层修改 (LED.java)：声明新的 Native 接口 LedOn3() 和 LedOff3()。



UI 层修改 (activity_main.xml)：在布局文件中增加第三个 ImageButton，设置 ID 和初始图片。



逻辑层修改 (MainActivity.java)：获取第三个按钮的实例，设置点击监听器。

在点击事件中调用 JNI 接口控制 LED3，并切换按钮的背景图片（亮/灭图标）。

```
import android.view.View;
import android.widget.ImageButton;
import com.farsight.led.databinding.ActivityMainBinding;

public class MainActivity extends AppCompatActivity
{
    LED led = new LED();
    boolean IsLight_1_On = false;
    boolean IsLight_2_On = false;
    boolean IsLight_3_On = false;
    private ImageButton LedButton_1;
    private ImageButton LedButton_2;
    private ImageButton LedButton_3;

    @Override
    protected void onCreate(Bundle savedInstanceState)
```

```
                led.LedOn2();
                led.close();
            }
            IsLight_2_On = !IsLight_2_On;
        }
    });

    LedButton_3 = (ImageButton)findViewById(R.id.buttonThree);

    LedButton_3.setOnClickListener(new View.OnClickListener()
    {
        @Override
        public void onClick(View view)
        {
            if (IsLight_3_On)
            {
                ((ImageButton)view).setBackground(getDrawable(R.drawable.pic_bulboff));
                led.open();
                led.LedOff3();
                led.close();
            }
            else
            {
                ((ImageButton)view).setBackground(getDrawable(R.drawable.pic_bulbon));
                led.open();
                led.LedOn3();
                led.close();
            }
            IsLight_3_On = !IsLight_3_On;
        }
    });

}
}
```

实验结果

详情见打包的视频

- 设计实验 4-2：通过图标控制蜂鸣器

- 设计实验 5-1：2 个按键
- 设计实验 5-2：采集 ADC 值

3. 实验报告具体要求

- 所选择的设计实验的具体设计思路（可利用文字、流程图等进行说明）
- 实验结果（截屏等方式体现）
- 设计实验遇到的问题，如何解决以及心得体会

遇到的问题及解决

问题：LED 灯的编号与控制引脚对应关系混乱，导致控制错位。

解决：查阅实验指导书或原理图，确认了 LED1 对应 GPIO 'y'，LED2 对应 'z'，LED3 对应 'x'，并在 native-lib.cpp 中修正了宏定义。

心得体会

本实验完整走通了 Android App 开发的 JNI 流程（Java Native Interface）。理解了上层 Java 代码是如何一步步穿透到 C++ 层，最终通过 Linux 设备驱动控制硬件的。这种“上层应用+底层驱动”的开发模式是嵌入式 Android 开发的核心技能