一、整体代码规范

1. 命名方式：
   - 不使用命名空间（using namespace），直接 std:: …
   - 命名要能体现该变量/函数结构体/类的作用
   - 变量名：img_src; （小写，_分隔）
   - 函数名：getTempData; （首字母小写，后续字母大写）
   - 结构体\类名：WidgetDevice; （全部字母大写）

2. 模块化：
   - 每个 UI 界面单独一个模块。
   - 同时每个 UI 需要预留一个 init()函数接口。

```
class MainWindow : public QMainWindow
{
    Q_OBJECT

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

    bool init();
```

3. 注释：
   尽量多注释，后续我们需要调用接口。

4. 编译环境：
   MSVC，win10

5. 布局：
   页面大小在 1500*900 左右。不能太小，但也不能超出 1080p 屏幕大小（1920*1080）。可先固定大小，后续如果可以做伸缩更好。

6. 时间节点：
   完成时间大约在 10 月 27 日左右，如果来不及，可优先完成功能，样式可后续完善
   界面开发顺序：设备调试——首页——温度检测——检测结果——数据库管理

7. 入参：你从 UI 控件读取数据给我
   出参：我给你数据拿去 UI 显示
   Qss 可以借鉴 resource 文件夹里面

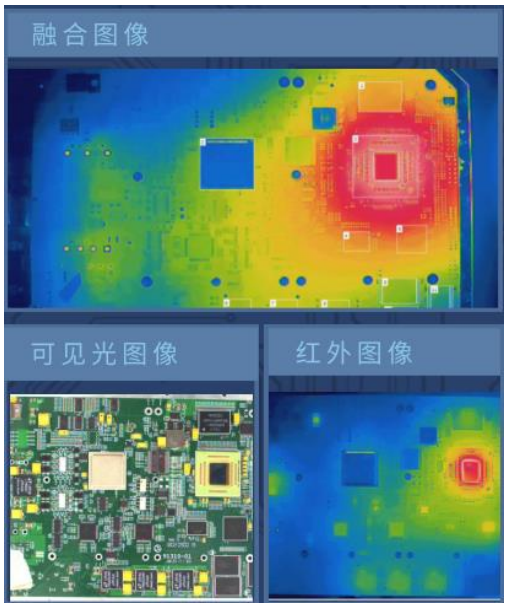
8. Github 仓库：
   https://github.com/XCmafu/pcb_ui.git
   每增加一个新 UI，就上传到一个新的分支，分支命名“update-月份-日期-新增 ui 名”

二、首页

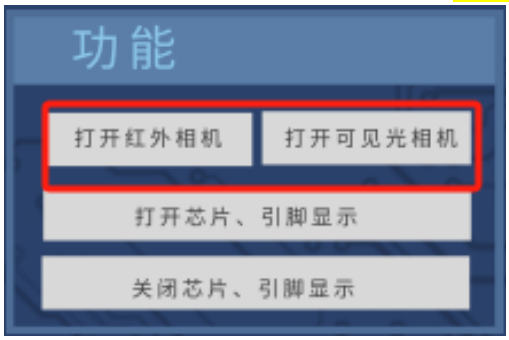
1. 三个图像显示部分：

原始图像均为 opencv cv::Mat 格式，可以转化成其他格式后（QImage）显示

原始图像接口每一次只能获取一张图像，所以该模块开启显示后，应该需要在循环中调用接口不断更新显示。



通过按钮进行打开和关闭图像显示，融合图像开始显示按钮是“开始故障检测”：



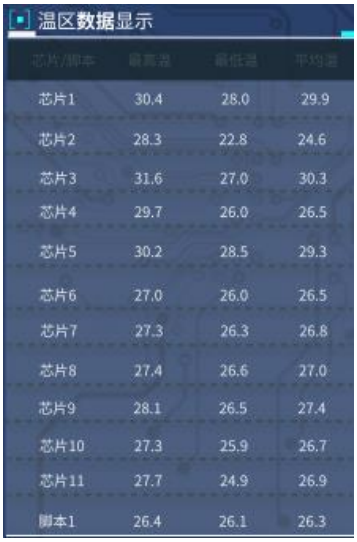
打开/关闭芯片引脚显示（可以换成勾选框）：区别是调用的后端接口不一样，但图像数据都是 cv::Mat 格式，所以需要给我一个标记让我知道调哪个接口。

2. 故障检测部分：

入参：时间和电路板 id（这里电路板 id 示意图画错了，不是 combobox，应该是 spinbox）

点击两个按钮都需要得到以上入参

3. 温区数据显示：



入参：故障检测部分输入数据。

出参：需要调用数据库接口：

```
bool selectICTempDataTable(int& id_pcb, int& id_ic, const QStringList& fields, QMap<QString, QVariantList>& result);
```

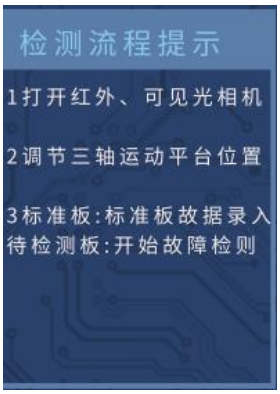
返回的数据类型是 QMap<QString, QVariantList>，解析数据流程如下：

```
    // TablePinTempData查询数据，需要指定id_pcb, id_pin和字段fields_temp
    QMap<QString, QVariantList> result_temp_data;
    QStringList fields_temp = {"temp_max", "temp_min", "temp_avg", "temp_center"}; // 入参
//    QStringList fields_temp = {"time_enable", "time_duration", "temp_max", "temp_min", "temp_avg", "temp_center"};
    sp_db->selectPinTempDataTable(id_pcb, id_pin, fields_temp, result_temp_data);
    // 打印查询到的数据
//    std::cout << fields_temp[0].toStdString() << ":  " << result_temp_data[fields_temp[0]][0].toString().toStdString() << std::endl;
//    std::cout << fields_temp[1].toStdString() << ":  " << result_temp_data[fields_temp[1]][0].toDouble() << std::endl;
    // or
    for (const QString& field : fields_temp)
    {
        qDebug() << "字段: " << field;
        foreach (const QVariant &value, result_temp_data[field])
        {
            qDebug() << "值: " << value;
        }
        qDebug() << "------";
    }
```

注：如果 QMap<QString, QVariantList>不好弄，可以先直接创建 double 类型的数据显示上去。

4. 检测流程部分：

文字显示即可

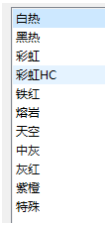


三、设备调试

1. 红外图像控制

① 切换伪彩

Combobox：顺序为：白热、黑热、彩虹、彩虹 HC、铁红、熔岩、天空、中灰、灰红、紫橙、特殊。（序号从 0 开始，默认是 2 彩虹）

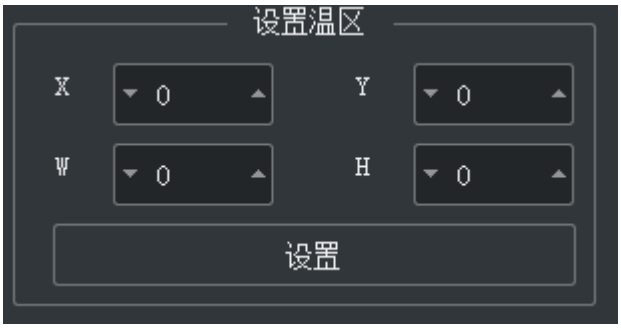


② 镜头：

三个按钮槽函数各自调用不同后端接口。

③ 设置温区

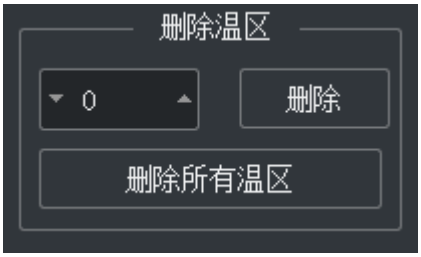
入参：X 范围为（0~640）、Y 范围为（0~512）、W 范围为（0~640-X）、H 范围为（0~512-X）。



④ 删除温区

单个温区删除入参：温区编号

删除所有温区不需要入参



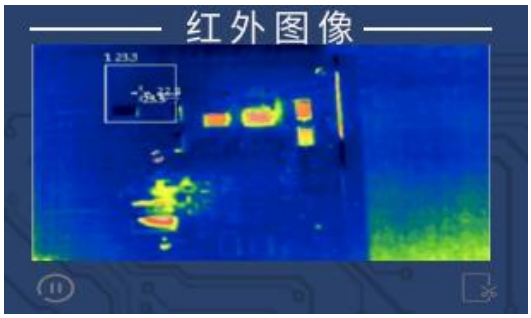
⑤ 读取温区温度：

入参：温区编号

出参：四个温度（double 类型）

```
bool readRegionTemp(int& index, float& temp_max, float& temp_min, float& temp_avg, float& temp_center);  //读取区域温度
```

2. 红外图像显示



① 播放按钮：

原始图像均为 opencv cv::Mat 格式，请转化成相应格式后显示

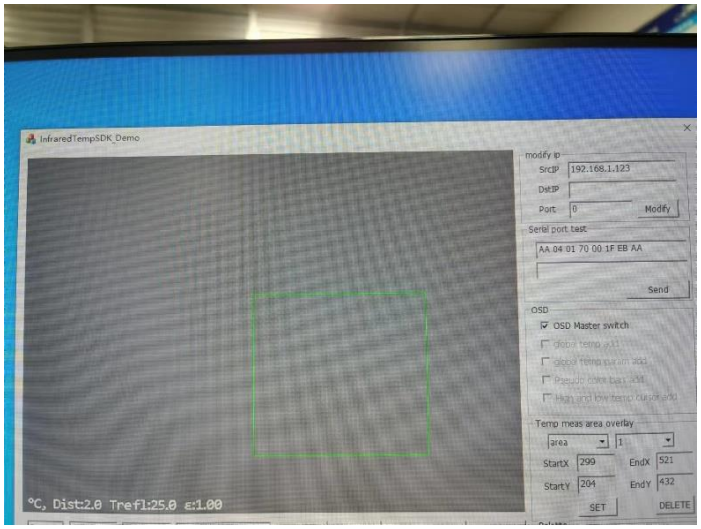
原始图像接口每一次只能获取一张图像，所以该模块开启显示后，应该需要在循环中调用接口不断更新显示。

② 截图按钮：

③ 双击放大：

放大后的大小为原始图像（640*512）大小

注：放大后需要有鼠标拖动框选功能，如下例子

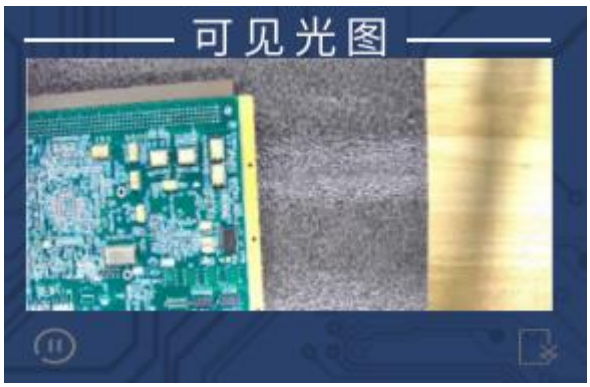


能够获取鼠标框选的大小，也即是上面设置温区的四个入参，并有其功能。

3. 可见光图像控制

四个模块，能有槽函数即可，我来调相应的后端接口
曝光时间那 spinbox 和 horizontalScrollBar 同时变化。范围为（100-5000）

4. 可见光图像显示



① 播放按钮：
原始图像均为 opencv cv::Mat 格式，请转化成相应格式后显示
原始图像接口每一次只能获取一张图像，所以该模块开启显示后，应该需要在循环中调用接口不断更新显示。

② 截图按钮：

③ 双击放大：
放大后的大小为原始图像（3840*2160）大小，需要等比例缩小

5. 运动平台

① Jop 模式：

X、y、z 三轴，鼠标点击就一直不断调用某个接口，直到鼠标释放再调一个接口。

例如 x 轴：

```
void WidgetDevice::on_bt_axis_x_up_pressed()
{
    int axis = 2;
    sp_global->sp_sport_console->jog_sport_up(axis);
}
void WidgetDevice::on_bt_axis_x_up_released()
{
    int axis = 2;
    sp_global->sp_sport_console->stopSport(axis);
}
```

② 点位模式：

入参：

轴（x 轴、y 轴、z 轴）、

位置（x 轴范围 0~300，y 轴范围 0~385，z 轴范围 0~400）

四、数据库管理

表结构：

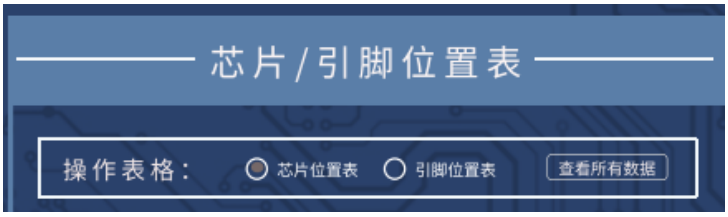
位置表：

```
"id INTEGER PRIMARY KEY,"
"id_pcb INTEGER,"
"id_ic INTEGER,"
"x INTEGER,"
"y INTEGER,"
"width INTEGER,"
"height INTEGER,"
"UNIQUE (id_pcb, id_ic))";
```

温度数据表：

```
"id INTEGER PRIMARY KEY,"
"id_pcb INTEGER,"
"id_ic INTEGER,"
"time_enable VARCHAR(14),"
"time_duration INTEGER,"
"temp_max FLOAT,"
"temp_min FLOAT,"
"temp_avg FLOAT,"
"temp_center FLOAT,"
"feature FLOAT,"
"UNIQUE (id_pcb, id_ic, time_enable, time_duration))"
```

1. 选择表，需要给我一个标记，知道是选的哪个表



2. 查看数据

调用接口如下

```
bool selectICPosTable(int& id_pcb, int& id_ic, const QStringList& fields, QMap<QString, QVariant>& result);
```

位置表入参：pcb_id（必填）、ic_id（必填）、位置 x（可选）、位置 y（可选）、高度 h（可选）、宽度 w（可选），结构可如下：

```
QStringList fields_pos = {"x", "y", "width", "height"};
```

位置表出参：类型为 QMap<QString, QVariant>& result，解析过程如下：

```
    QMap<QString, QVariant> result_pos_data;
    QStringList fields_pos = {"x", "y", "width", "height"};
    sp_db->selectPinPosTable(id_pcb, id_pin, fields_pos, result_pos_data);
    // 打印查询到的数据
    for (const QString& field : fields_pos)
    {
        qDebug() << field << ":" << result_pos_data.value(field);
        // or
//        std::cout << field.toStdString() << ":" << result_pos_data.value(field).toInt() << std::endl;
    }
```

注：如果 QMap<QString, QVariant>不好弄，可以先直接创建相应类型的数据显示上去。

注：查询后，点击某条记录可将该记录的数据填充到修改数据和删除数据框中。

3. 修改数据：

入参：pcb_id（必填不可修改）、ic_id（必填不可修改）、位置 x（必填可修改）、位置 y（必填可修改）、高度 h（必填可修改）、宽度 w（必填可修改）

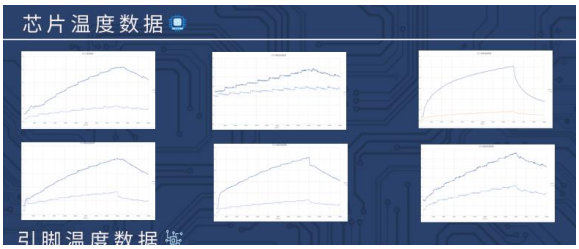
这里示意图的 Combobox 可能不正确，需要改成 spinbox

4. 删除数据：

入参：pcb_id（必填）、ic_id（必填）

五、温度监测

1. 芯片温度数据



入参：两个 std::vector<std::vector<std::pair<int, double>>>对应两条线。横坐标为时间 int 类型，纵坐标为温度 double 类型。

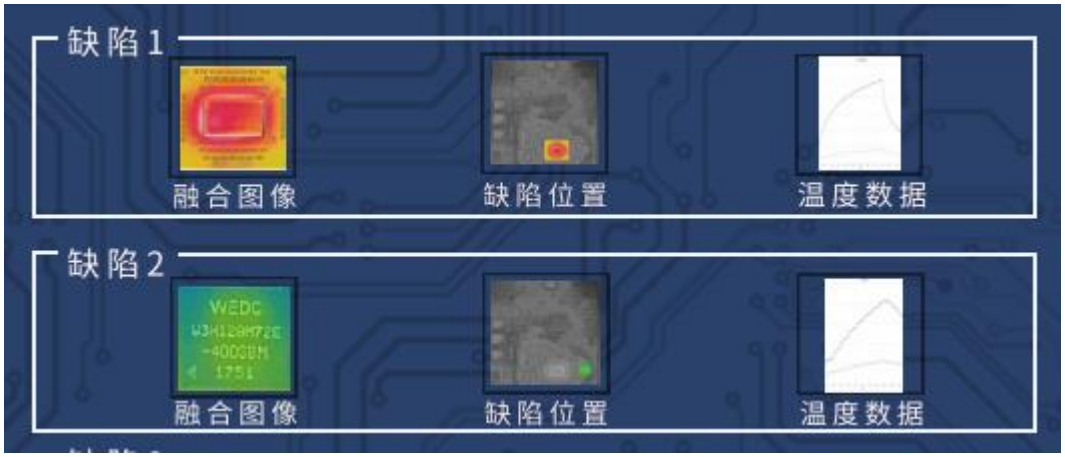
一行只展示 2~4 张表。

2. 引脚温度数据

同上。

六、检测结果

1. 缺陷部分：



入参：目前暂定，std::vector<std::vector<std::vector<cv::Mat>>>

2. 缺陷结果部分：

入参：同上

导出结果：把缺陷和相应的图导出到本地 word 文档里面


缺陷检测结果
缺陷数量:4
缺　陷1:芯片3
缺　陷2:芯片1
缺　陷3:芯片6
缺　陷4:引脚1
导出检测结果文件成功。
导出检测结果文件