项目重构思路备忘录 (WebSocket 架构)

状态： 数据库连接成功， `add_task` / `add_report_detail` / `insertreport` 功能已验证。 **截止日期：** 2025年11月8日 (星期六) **目标：** 11月7日 (星期五) 下班前完成所有功能并测试完毕。

模块一：地图交互与路径创建 (新核心任务)

挑战： `map.pcd` [cite: map.pcd] 是 3D 点云。 **策略：** 将其"压平"(Flatten) 为 2D 俯视图进行显示和交互。

任务分解：

1. **PCD 解析与 2D 渲染 (工作量: 高)**
    - **任务:** 编写一个函数，用于读取 `map.pcd` 文件。
    - **实现:**
        1. 打开文件，跳过 PCD 头信息（以 "DATA" 开头的行为止）。
        2. 遍历所有点，只提取 `x` 和 `y` 坐标（忽略 `z`）。
        3. 在遍历时，记录下 `x_min`, `x_max`, `y_min`, `y_max`（地图边界）。
        4. 使用 `QGraphicsScene` 作为地图容器。
        5. 设置 `QGraphicsScene::setSceneRect(x_min, y_min, x_max - x_min, y_max - y_min)`。
        6. 再次遍历所有点，将每个 `(x, y)` 坐标使用 `scene->addEllipse(x, y, 0.1, 0.1, ...)` (设置一个很小的半径) 添加到 `QGraphicsScene` 中。
        7. 将 UI 上的 `QGraphicsView` 控件的 `scene` 设置为这个 `QGraphicsScene`。
    - **时间：** 11月1日 (周六) - 11月3日 (周一) (共 3 天)
2. **地图交互与坐标转换 (工作量: 中)**
    - **任务:** 在地图上点击，获取真实的地图坐标，并创建路径点列表。
    - **实现:**
        1. 为 `QGraphicsView` 创建一个子类（例如 `MapGraphicsView`），并重写 `mousePressEvent(QMouseEvent *event)` 函数。
        2. 在事件中，通过 `mapToScene(event->pos())` 获取点击点的**地图坐标** (例如 `(15.2, -8.4)`)。
        3. 添加 "添加路径点" / "添加巡检点" 的 `QRadioButton` (UI)。
        4. 当用户点击时，检查 `RadioButton` 的状态，将地图坐标和点类型（"path" 或 "inspection"）存入一个 `QList<QVariantMap>` 成员变量 `m_pointList` 中。
    - **时间：** 11月4日 (周二) (共 1 天)

模块二：WebSocket 通信 (对接)

使用 `QWebSocket` (您已有的 `m_rosSocket` [cite: robot.h]) 与巡检车电脑通信。

1. **IP 输入界面 (工作量: 低)**

- **任务:** 创建一个 `QDialog` ，用于在启动时输入 IP 地址。
- **时间：** 10月30日 (周四) 晚上

2. **发送路径与接收数据 (工作量: 中)**
   - **任务:** 实现“发送任务”按钮，并将 `m_pointList` （来自模块一) 序列化为 JSON 发送。同时，实现 `onRosMessageReceived` 槽函数来解析巡检车返回的 JSON 数据。
   - **JSON 格式 (建议发送)：**

```
{
  "command": "set_path",
  "path_name": "path_from_gui",
  "points": [
    { "x": 1.2, "y": -0.5, "z": 0.0, "qw": 1.0, "point_type": "path" },
    { "x": 5.2, "y": -1.3, "z": 0.0, "qw": 1.0, "point_type": "inspection" }
  ]
}
```

   - **JSON 格式 (建议接收)：**

```
{
  "type": "inspection_result",
  "point_id": "巡检点 5",
  "data": "0.51MPa",
  "status": "正常",
  "image_url": "http://<robot_ip>/images/img_123.jpg",
  "video_url": null
}
```

   - **图片/视频：** 您的猜测是对的。巡检车会发回一个 URL 链接 (例如 `http://<robot_ip>/images/fault.jpg`)。您的程序**只需存储这个 URL 字符串**即可。
   - **时间：** 11月5日 (周三) (共 1 天)

模块三：数据记录与报告 (集成)

这部分功能已在 `dataBase` [cite: database.cpp (已更新):database.cpp] 中实现，现在只需在正确的时间点调用它们。

1. **集成测试 (工作量: 中)**
   - **任务:** 将所有模块串联起来，进行完整流程测试。
   - **流程：**
     1. (UI) 点击 "选择方案" (`on_Btn_PlanSelect_clicked`) -> 加载 JSON [cite: path_info.json] -> `InspectionPlanName` 显示方案名。
     2. (UI) 点击 "任务开始" (`on_Btn_missionStartORstop_clicked`) -> 调用 `dbase->add_task(...)` -> 将模块一的 `m_pointList` 发送给模块二（WebSocket)。
     3. (WebSocket) 收到巡检结果 JSON (`onRosMessageReceived`) -> 解析 JSON -> 调用 `dbase->add_report_detail(...)` 存入明细。

4. (WebSocket) 收到 "任务完成" 消息 (或用户点击 "结束任务") -> 调用 `dbase->insertreport(...)` 生成总结报告。
5. (UI) 切换到报告界面，调用 `showReport()` 和 `showRpmore()`，确认数据能正确显示。

- **时间：** 11月6日 (周四) (共 1 天)

2. **Buffer 与最终测试 (工作量: 可变)**
   - **任务:** 处理遗留 Bug、调整 UI 细节、与导航同事进行联调测试。
   - **时间：** 11月7日 (周五) (共 1 天)

总结

您最大的工作量是**模块一（PCD 地图）**。一旦地图可以显示和点击，后续的 WebSocket 通信和数据库记录都非常快，因为我们已经打通了数据库连接（ `add_task` ， `add_report_detail` 等）。这个计划时间很