SR-SToF100测试系统开发手册

项目开始日期：2021.03.15

文档更新日期：2021.06.22

更新人员：产品中心-研发部-硬件组-白家伟

更新日志：

> 2021.06.20-V1.1：增加循环测试模式以及3D测试数据输出
> 2021.06.15-V1.0：完整的开发手册
> 2021.06.10-V0.2：增加异常点监控及输出异常率功能
> 2021.06.08-V0.1：软件基本功能

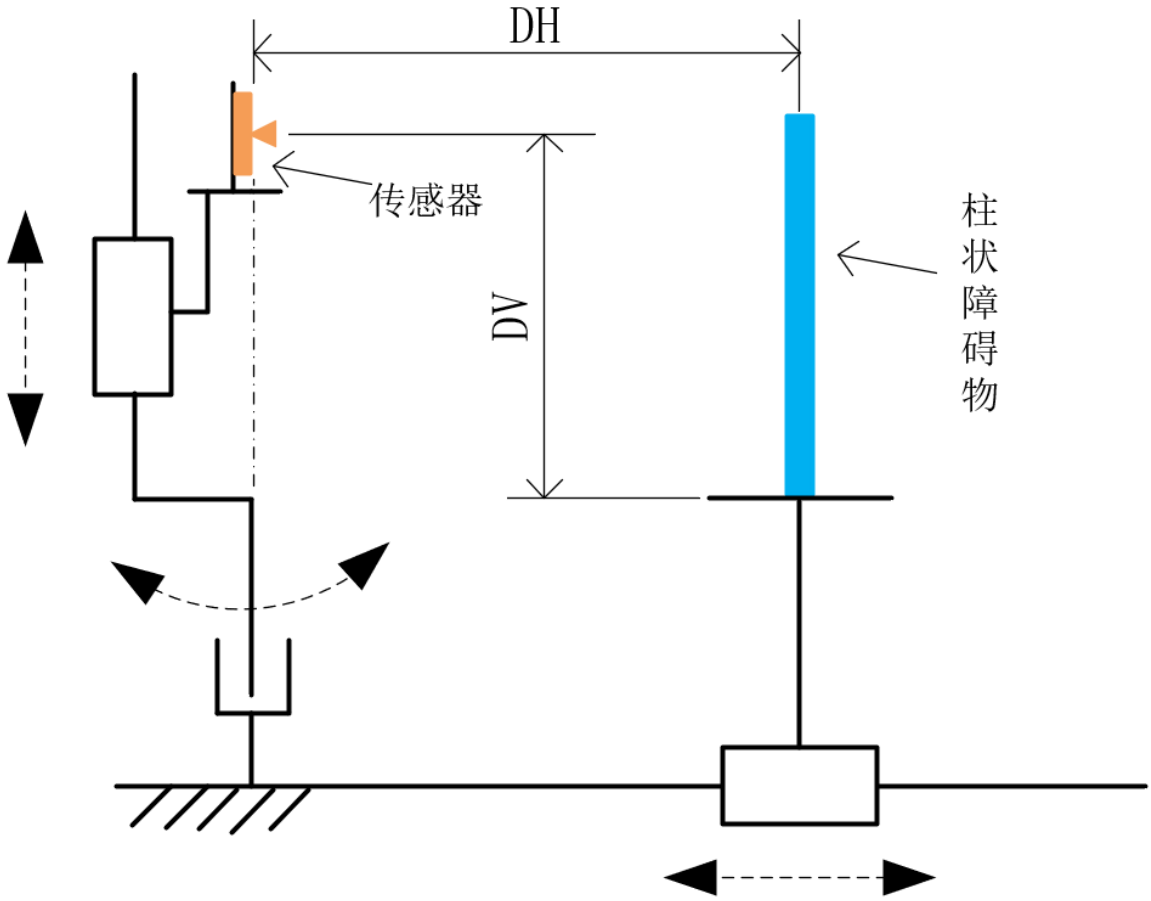
项目描述：

项目背景：

- 公司当前以及后续开发的大FOV测距类传感器需要一套测试治具，用于测试障碍物尺寸、表面材质等因素对传感器的FOV、精度以及稳定性的影响。

项目需求：

- 3个自由度，分别为水平X轴、垂直Z轴以及旋转R轴，如下图：
  - 水平X轴：0-1500mm（分辨率：1mm）
  - 垂直Z轴：0-500mm（分辨率：10mm）
  - 旋转R轴：-40°~40°（分辨率：0.25°）



- 将传感器的实时测量到的距离值以极坐标的方式绘制完整的轮廓曲线。
- 生成PDF格式的报告。

项目功能需求分析以及完成情况：

- [x] ~~系统规划梳理：~~
  - [x] ~~细化系统流程~~
  - [x] ~~完整的测试人员操作步骤~~
- [x] ~~检查电气设备以及机械结构的完整性，排查可能存在的故障：~~
  - [x] ~~X轴限位传感器损坏待更换~~
  - [x] ~~X轴模组机械卡顿，同步轮与电机轴滑丝~~
- [x] ~~重写EU100的嵌入式代码，配置外设IO与CAN驱动~~
- [x] ~~电机驱动器定义原点与正负限位~~
- [x] ~~GUI交互界面及各模块功能：使用PyQt5作为GUI框架~~
  - [x] ~~主界面：~~
    - [x] 标题栏
      - [x] ~~打开后台文件功能：包括CSV文件，最终测距图图片以及PDF报告~~
      - [x] ~~退出系统按钮~~
      - [x] ~~调试：包括电机单机调试以及SH200的强制开启与关闭功能~~
      - [x] ~~驱动：手动打开PCAN-USB驱动，建立上位机与EU100之间的通信~~
      - [x] ~~其他：Help可连接到在线帮助文档；Version显示软件版本以及作者信息~~
    - [x] ~~两个图像显示器（测试结果图和信号强度图），要求在每轮测试结束后更新测试结果图的迭代图以及信号强度图的本次测试信号强度反馈图~~
    - [x] ~~当前状态栏Text提示功能~~
    - [x] ~~主界面操作选项：~~
      - [x] ~~“开始测试”按钮：启动测试主程序，开启后全程自动测试直到输出PDF报告后结束~~
      - [x] ~~“停止测试”按钮：将除了后台输出文件外的信息重置为默认值，并确保数据完整性~~
      - [x] ~~“三轴回原点”按钮：自动连接PCAN-USB驱动，自动使能三轴电机，将设备三轴均至于通过驱动器配置好的原点处，并通过【绿灯】闪烁【3次】提示用户以回到原点~~
      - [x] ~~初始位置设置按钮及功能：为了扩大测试系统软件的适用范围，即可以在不同初始位置进行测试（除原点以外的零点标定功能）~~
      - [x] ~~后台数据实时显示按钮及功能：为了在测试过程中实时监控后台测试数据流，需要对后台测试数据进行实时抓取并显示，添加清空界面功能~~
      - [x] ~~生成PDF报告按年及功能：将测试的最终结果以PDF格式输出，命名为“PDFreport_测试ToF型号_测试ToF编号_时间.pdf”，保存在程序安装路径的PDFs文件夹中~~
  - [x] ~~子界面：~~
    - [x] ~~电机可单机调试的GUI界面：~~
      - [x] ~~XZR轴电机可以分别开关以及给定速度调试~~
    - [x] ~~初始位置设置操作的GUI界面：~~
      - [x] ~~XZ轴可根据实际需求进行零点标定~~
    - [x] ~~后台数据实时显示的GUI界面：~~
      - [x] ~~实时读取并解析SH200-ToF返回的CAN报文~~
      - [x] ~~重写数据格式并实时抓取显示在TextBrower中~~
      - [x] ~~清空TextBrower中的内容~~

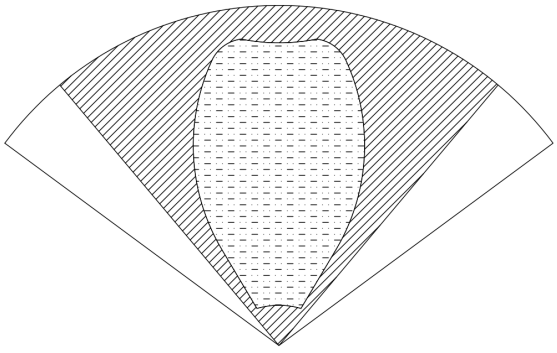
- [x] ~~Version信息的GUI界面：显示软件版本号，公司标志，版权声明，技术支持联系方式~~
- [x] ~~PCAN-USB驱动：实现上位机与EU100的通信，上位机可以收发CAN报文~~
- [x] ~~PCAN的集成和封装：实现继承PCANBasic类的DriveCAN，发送数据等帧类型可选，读取对应id的报文~~
- [x] ~~USB2SPI的集成和封装：实现主机配置、读取功能、写入功能、写读功能~~
- [x] ~~PDF测试报告自动生成，可自定义PDF格式（PDF样式、表格、字体、图片）~~
- [x] ~~循环测试~~
- [x] ~~3D数据输出~~
- [x] ~~系统稳定性的完善：~~
  - [x] ~~返回的测试数据异常点过多的问题~~
  - [x] ~~GUI主界面关闭后，软件开启的线程后台进行运行的问题~~
  - [x] ~~强制开关SH200后，后台数据跳变过大的问题~~
  - [x] ~~按下“停止测试”按钮后，软件闪退的问题~~
- [x] ~~将程序打包成.exe文件~~
- [ ] 附加功能：
  - [x] ~~增加主界面标题栏的快捷方式~~
  - [ ] 使用数据库管理后台数据（可选）

功能概述：

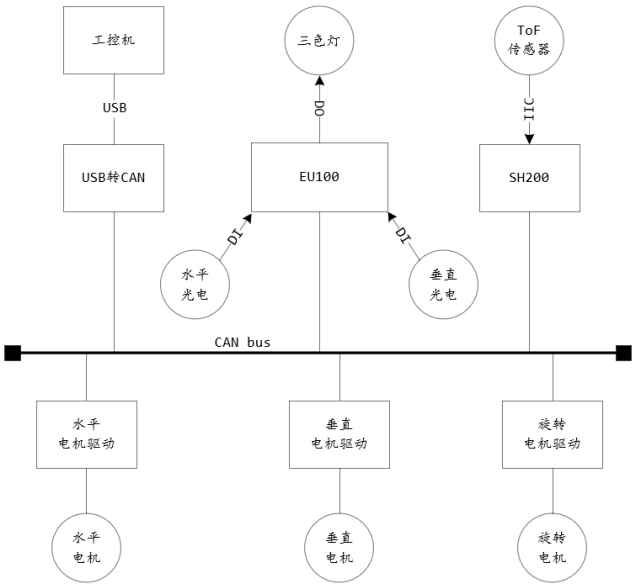
1. 指定测距区间（水平、竖直），且在测距区间内循环遍历测量
2. 实时观测后台测距数据，包括：无效点、异常点、正常点（个数，与中心线夹角，测距距离，信号强度）
3. 随时终止（且保证数据完整度）
4. 电机单机调试，SH200强制开关
5. 输出PDF格式的2D测距报告，包括单层平面图，异常率
6. 输出3D测距数据，由于目前使用的Python-matplotlib的限制，无法完美的生成柱坐标图，所以目前的方式只能讲3D的数据导出后通过MATLAB进行绘制
7. 后台数据csv文件导出，包括：Distance_data.csv；Stremgjt_data.csv；outliersRate.csv；Distance_image.png；Strenght_image.png；OutliersRate_image.png

SToF100-FOV测试方案：

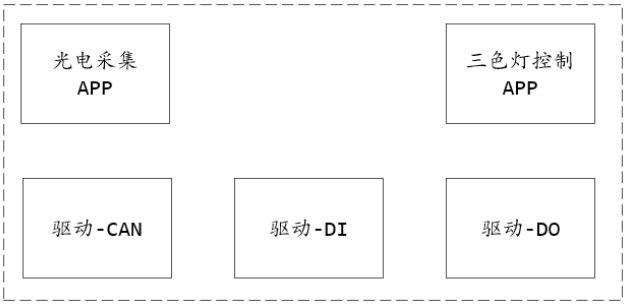
- 平台：X、Z、R三轴电机及其驱动器，X轴模组，Z轴丝杆滑台，R轴转台，限位传感器，EU100，SH200，LED三色灯，一路CAN，PCAN-USB，待检ToF
- 测试目标：以10mm为分辨率将垂直行程DV（500mm）离散为50个高度值。在各个高度值下遍历水平行程（1mm分辨率）和旋转行程（0.01度分辨率），将传感器的实时测量到的距离值以极坐标的方式苗点，最终获得完整的轮廓曲线。所得的理论图像如下图所示。



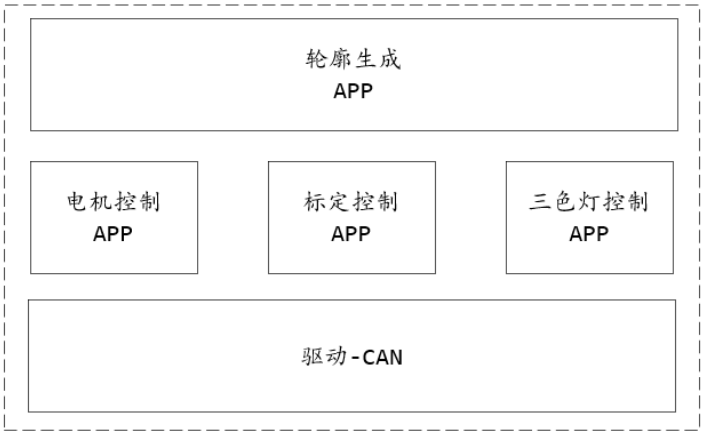
- 测试方案：
  - 电气拓扑图如下图所示：



  - 嵌入式开发：
    - 嵌入式开发主要针对EU100，主要完成光电传感器采集、3个轴的零位标定和限位、三色灯控制；



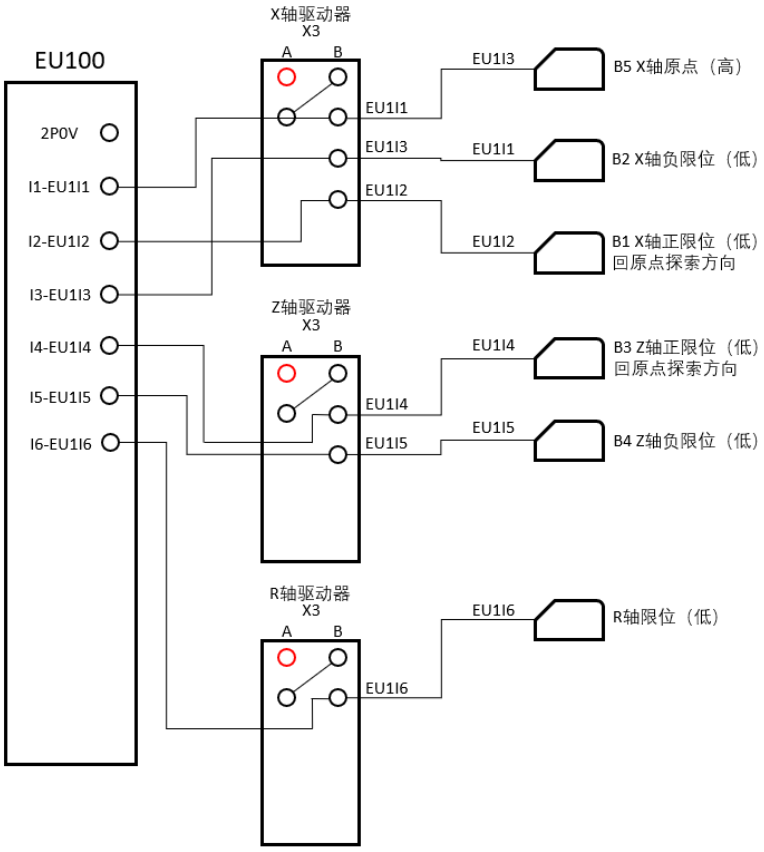
  - 上位机开发：
    - 工控机为治具的上位机，控制EU100、电机，采集传感器的测距值，生成轮廓。



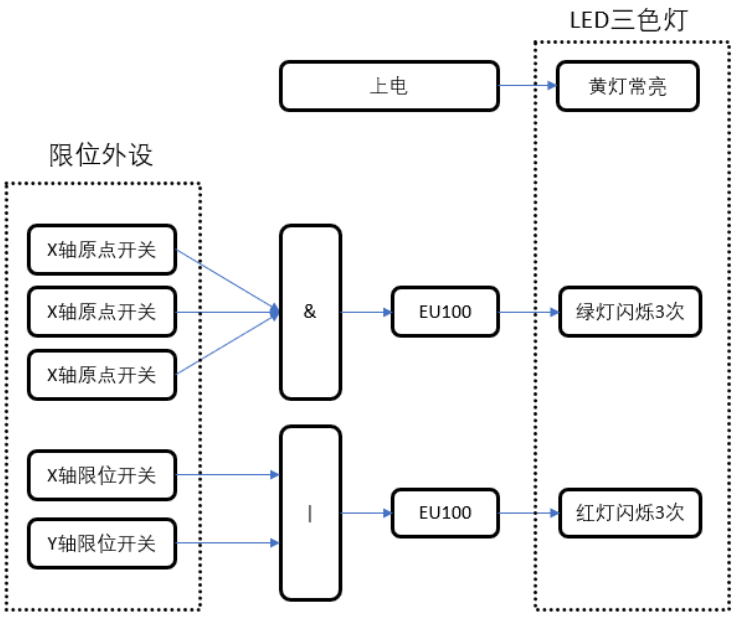
SToF100-FOV测试方案实施：

- 电气方面：

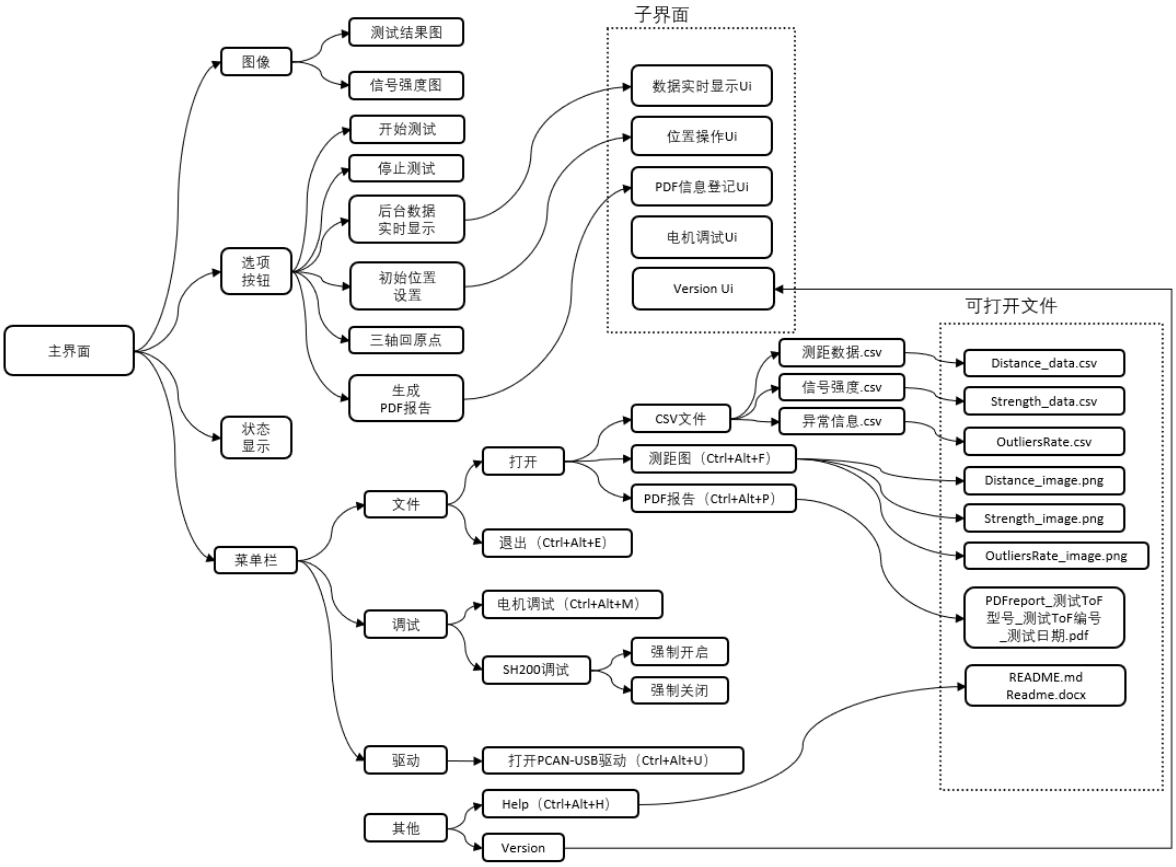
  在项目开始初期，由于原先限位传感器的损坏，同时考虑到后期电机驱动器进行原点及限位定义以及上位机控制的逻辑，在更换限位传感器的同时对电气接线做了一定的修改，即：将限位传感器的信号线分成两股，一股接电机驱动器，一股接EU100，拓扑图如下所示。



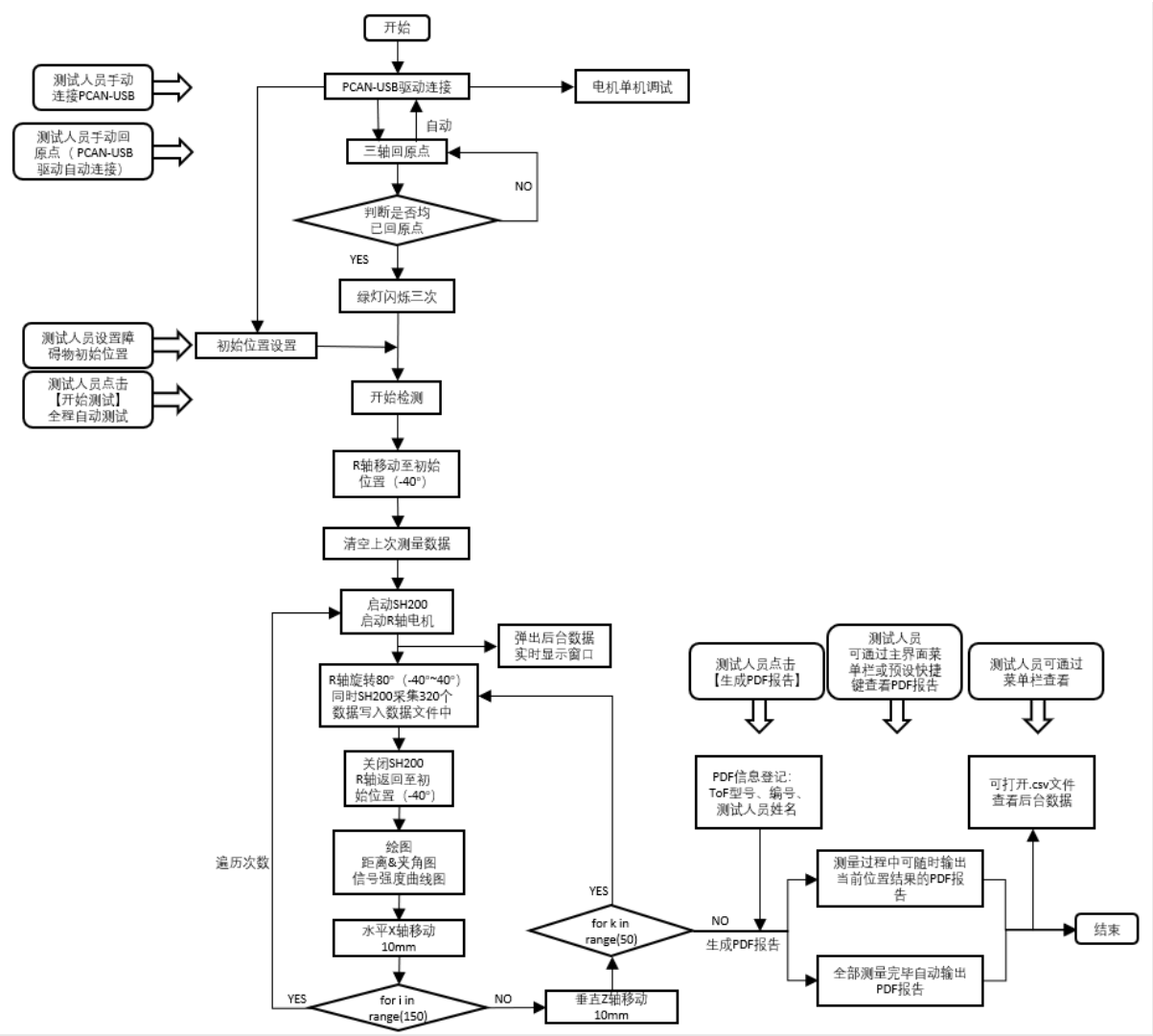
- 嵌入式方面：
  - EU100主要负责对外设IO口的控制以及CAN数据的收发以及完成规定的动作



- 上位机软件方面：
  - 上位机：SR-SToF100测试系统主要完成对EU100、测试流程的控制，对SH200-ToF发回的CAN报文进行解析抓取，绘制测试结果图以及对后台数据和文件的监控与保存。
  - 上位机各界面关系图如下所示：

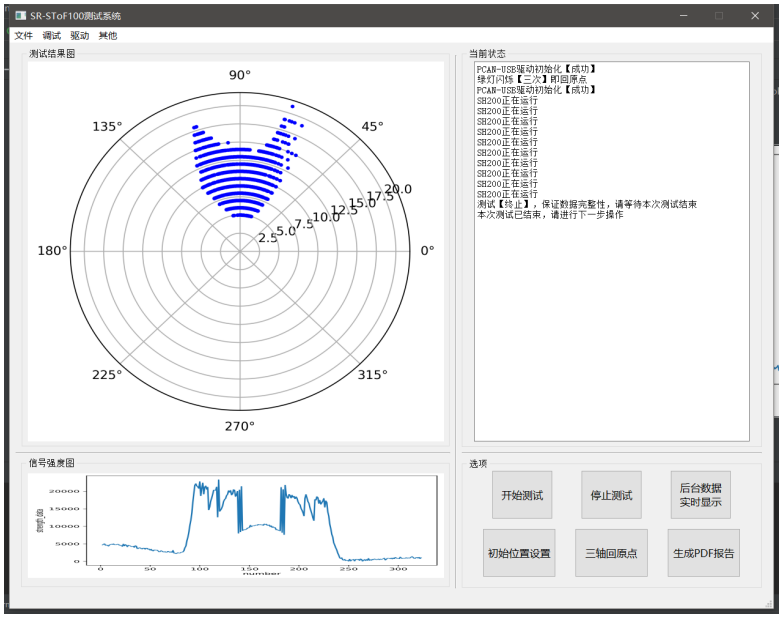


程序流程及测试人员操作流程示意图：

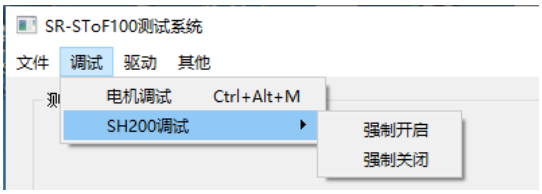


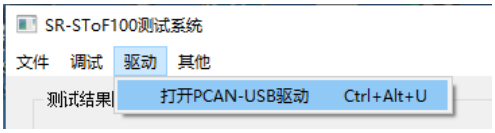
各UI 界面以及功能概述：

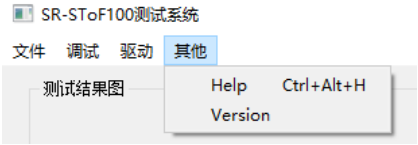
- tips：各UI界面的包含以及耦合关系如上位机各界面关系图所示。
- 主界面：分为菜单栏（包括：文件、调试、驱动及其他），测试结果图迭代显示，信号强度图试试显示，当前状态Text提示以及操作选项（开始测试、停止测试、后台数据实时显示、初始位置设置、三轴回原点和生成PDF报告），用户可通过该界面进行启停测试，初始位置设定，回原点，以及结果报告输出。



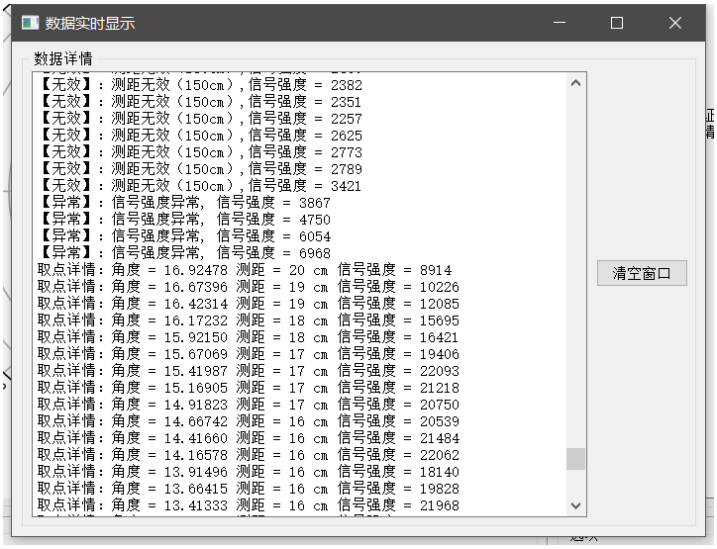
- 主界面的菜单栏：通过菜单栏可以进行后台数据文件的监控，部分硬件设备的调试，PCAN-USB驱动连接，打开帮助文档以及查看版本信息，以上操作均可通过键盘快捷键操作。





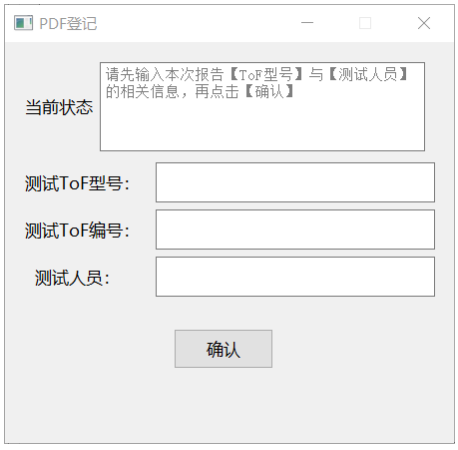


- 后台数据实时监测窗口：用户可以对测距过程中，上位机抓取到的后台数据进行监控和查看，程序中对SH200-ToF发回的CAN报文进行了解析和重构，同时为了测量结果的精确性，对数据进行合理性以及信号强度过滤，并标记【无效】、【异常】、【取点详情】
  - 【无效】：测距无效，即在ToF的测距范围内无障碍物，超过了150cm
  - 【异常】：信号强度异常，即在ToF的测距范围内检测返回的值与实际不符，信号偏弱，测距数据异常
  - 【取点详情】：测距有效且信号强度正常，即有笑点，输出数据格式：
    - 角度：在以ToF为原点，以ToF正对障碍物方向为0°，测距目标点到ToF与0°线的夹角；
    - 测距：测距目标点到原点的距离，单位：cm；
    - 信号强度：根据SH200上报的CAN报文格式，16进制转换为10进制后的数值；

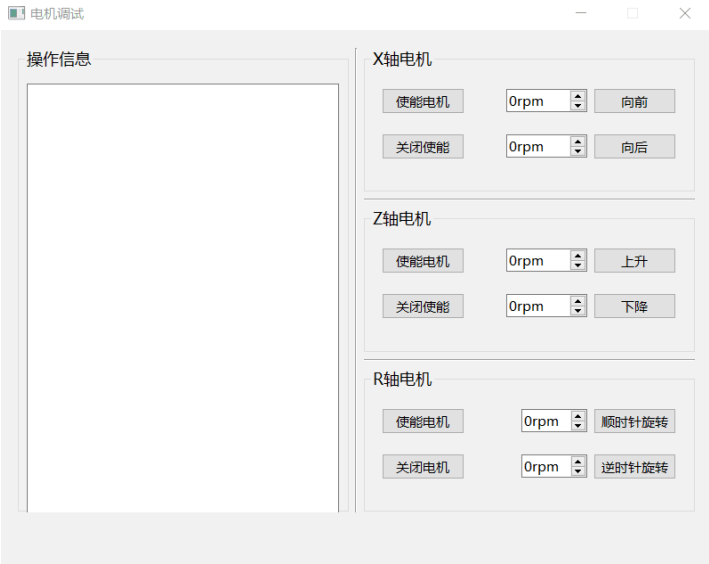


- 初始位置设置-位置操作窗口：测试人员可通过此窗口，对障碍物的水平X轴，垂直Z轴的零点进行标定（零点区别于原点，原点：XZR三轴初始点）以及对测距位置区间的设置

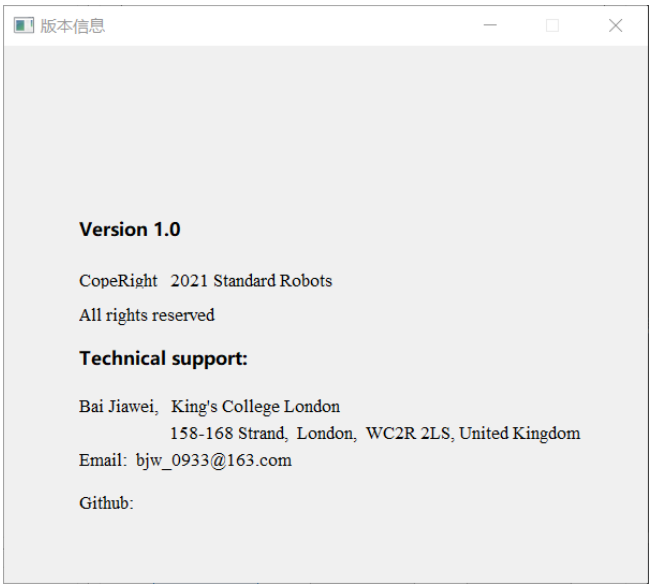
- 生成PDF报告信息登记-PDF登记窗口：测试人员在测试结束后，可将测试结果输出为PDF格式的报告，在生成报告前，需要测试人员在PDF登记窗口输入被测ToF的型号、编号以及本人姓名。
  - 可以通过快捷键：Ctrl+Alt+P，打开PDFs文件夹



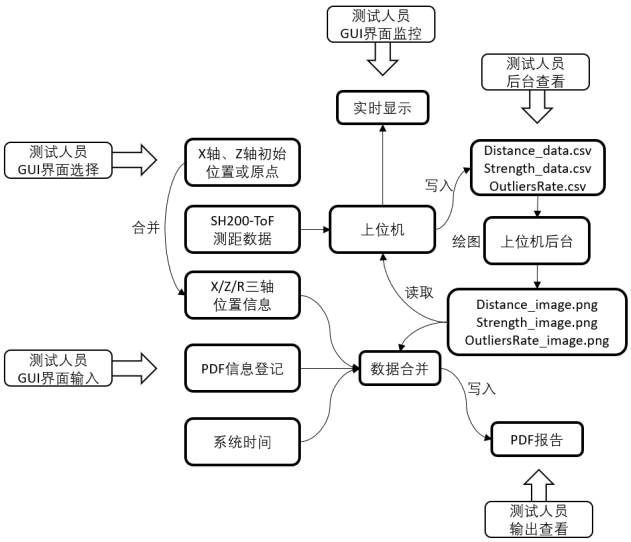
- 电机调试窗口：测试人员可以通过此窗口对单个电机进行调试



- Version窗口：测试人员可通过此窗口查看软件版本信息和技术支持



数据流示意图：



项目开发环境：

嵌入式开发环境：

- MDK5
- 芯片：STM32F105RB
- stm32f10x系列包

嵌入式项目文件：

- /app/main.c：主函数
- components/drivers/blink.c：EU100板载LED灯
- components/drivers/exti.c：外部中断的初始化配置以及外部中断函数
- components/drivers/sensor.c：外设IO口的初始化配置
- components/drivers/can_SToF100.c：CAN的初始化配置以及CAN的接收和发送中断函数
  - 包括XZR三轴电机回原点反馈以及SH200自动配置

```
//三轴回原点标志位：
extern unsigned char LED_X_Origin_flag;  //X轴回原点成功标志位
extern unsigned char LED_Z_Origin_flag;  //Z轴回原点成功标志位
extern unsigned char LED_R_Origin_flag;  //R轴回原点成功标志位

//SH200自动配置：
if((RxMessage.StdId == 0x511)&&(RxMessage.IDE == CAN_ID_STD)&&(RxMessage.DLC == 8) && (SH200_init_flag == 0))
    {
      switch(SH200_index){
        case 0x20010101:{
          SH200_transmit_flag = 1;
          SH200_active();
        };break;

        case 0x20010801:{
          SH200_OUTconfig_flag = 1;
          SH200_active();
        };break;

        case 0x20010202:{
          SH200_activate_flag = 1;
          SH200_active();
        };break;

        case 0x20110101:{
          LED_RED = !LED_RED;
          SH200_init_flag = 1;
        };
        default:break;
      }
    }
```

```
    else if((RxMessage.StdId == 0x511)&&(RxMessage.Data[1] == 0x04))
    {

    }
```

- components/drivers/motor_kinco.c：kinco驱动器-电机配置，包括：使能/关闭/模式选择/速度
- components/drivers/SH200.c：SH200驱动配置，包括：进入配置/使能主动上报/退出配置/激活设备
- components/drivers/led.c：外设LED三色灯动作：回原点绿灯三次闪烁，触碰限位红灯三次闪烁

上位机开发环境：

- anaconda3+spyder4.01：环境管理与IDE，以下模块都通过conda安装
- python 3.8.8：anaconda3自带的python版本
- PyQt5 5.9.2：GUI界面
- Qt designer：GUI界面设计
- PCANBasic API V4.4.1.413：PCAN官方库
- reportlab 3.5.26：生成PDF
- .ttf文件：生成PDF所用到的字体包
- ctype：python 3.8.8自带版本，C语言的接口，带有指针等功能
- re：自带版本，正则表达式匹配
- matplotlib 3.4.2：绘制极坐标图
- numpy 1.20.3：进行数组计算
- pandas 1.2.4：用于数据分析、时间序列和统计的数据结构
- csv 1.0.5：用于读写CSV数据文件
- math：Python自带的数学运算包
- time：自带版本，获取当前时间信息
- webbrowser：自带版本，调取网页
- threading：自带版本，线程控制

项目文件说明：

- SR_SToF100_system.py：项目的功能实现代码，为了使代码结构清晰，该代码文件调用以下代码
  - ProjectPath.py：获取当前程序运行目录
  - Main_code/CSV_Plot.py：读取CSV文件绘制测距结果图和信号强度图
  - Main_code/CSV_write.py：将SH200-ToF返回的CAN报文解析重组后，读写CSV文件
  - Main_code/PDF_1.py：测试PDF生成的旧版类和测试主函数（未用）
  - Ui/MainUi.py：主界面GUI界面的.ui文件生成的py可调用文件
  - Ui/Data_display.py：后台数据实时显示GUI界面的.ui文件生成的py可调用文件
  - Ui/Label.py：PDF登记GUI界面的.ui文件生成的py可调用文件
  - Ui/location.py：位置操作GUI界面的.ui文件生成的py可调用文件
  - Ui/MotorControl.py：电机调试GUI界面的.ui文件生成的py可调用文件
  - Ui/show_image.py：测试图像显示GUI界面的.ui文件生成的py可调用文件
  - Ui/Version.py：版本信息GUI界面的.ui文件生成的py可调用文件
  - Icon/：存放公司标识和标志
  - PDFs/：存放生成的PDF报告
  - Matplot_data/：存放测试过程中生成的测距结果图（迭代），信号强度图（实时），测距结果数据的CSV文件（迭代），信号强度的CSV文件（实时）

二次开发注意事项：

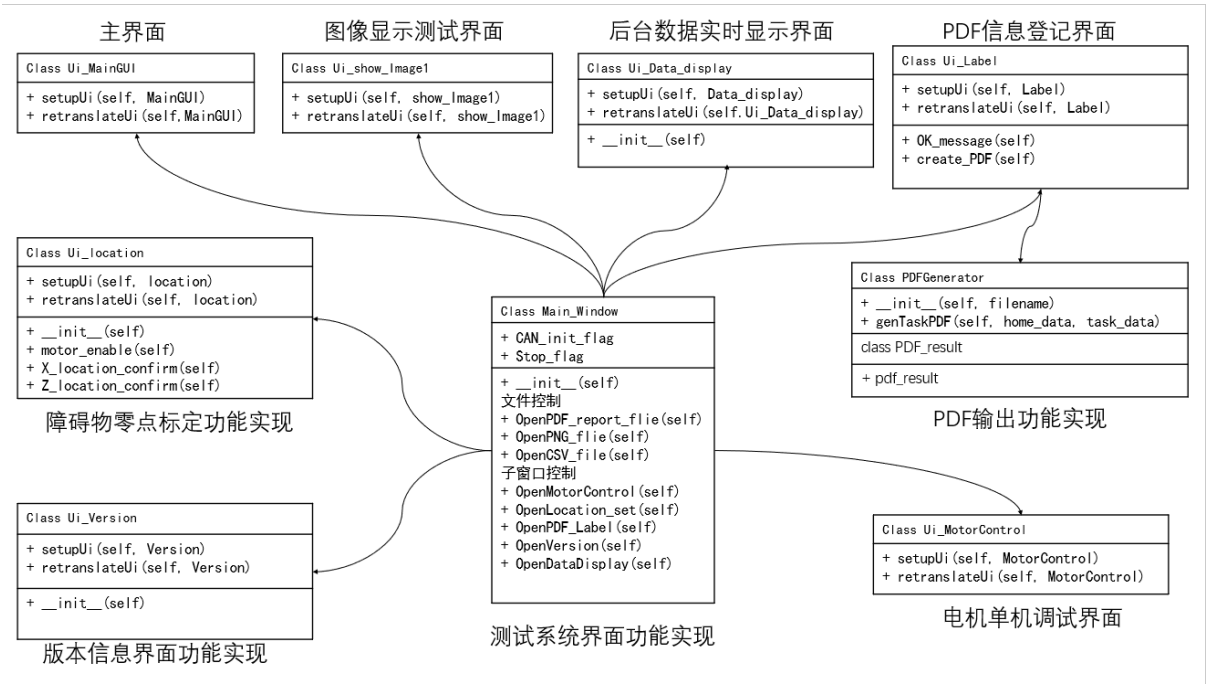
全局变量：

该上位机程序未使用数据库操作，因此定义了多个全局变量

全局变量中的标志位在每次动作时都会自动重置，X轴Z轴的测距变量在每轮测试完成后都会自加相应的数值，PDF类的全局变量中，X轴Z轴的初始位置会被直接引用与DATA_distance、DATA_vertical组合构成PDF报告中的测试范围

```
class global_value():
    '''
    全局变量
    '''
    CAN_init_flag = 0         #CAN初始化标志位
    SH200_work_flag = 0       #SH200工作标志位
    stop_flag = 0             #全机停止标志位
    resume_flag = 0           #恢复工作标志位（未用到）
    DATA_distance = 50        #水平X轴测距变量
    DATA_vertical = 0         #垂直Z轴测距变量
    X_init_location = 50      #X轴初始位置变量
    Z_init_location = 0       #Z轴初始位置变量
    X_goal_location = 50      #X轴目标位置变量
    Z_goal_location = 0       #Z轴目标位置变量
    X_bcak_completedFlag = 1  #X轴一次遍历返回初始位置完成标志位
    ###############  PDF类的全局变量#############
    PDF_DistanceData = 0      #PDF中测距范围变量
    ToF_type = 0              #PDF中ToF类型登记变量
    inspector_person = 0      #PDF中测试人员登记变量
    ToF_num = 0               #PDF中ToF编号登记变量
    outliers_num = 0          #异常点个数（尚未启用）
    normal_num = 0            #正常点个数（尚未启用）
    outliersRata_flag = 0     #PDF类异常点率的标志位
    PDF_normal_num = 0        #PDF类中的正常点个数
    PDF_outliers_num = 0      #PDF类中的错误点个数
```

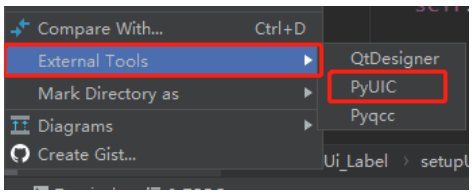
Ui继承关系UMl图：



GUI界面采用PyQt5，其中Ui_MainGUI、Ui_show_Image1、Ui_Data_display、Ui_Label、Ui_location、Ui_Version以及Ui_MotorControl是使用Qt Designer绘制.ui后生成的py类。

更改Ui界面的方法：

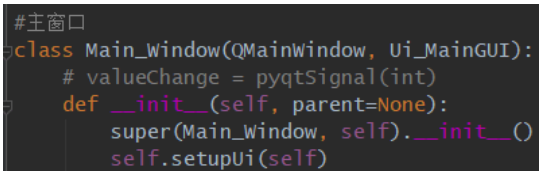
- PyUIC：通过Qt Designer绘制所需的.ui文件后，可以通过PyCharm的外部工具：External Tools→PyUIC可以直接将.ui文件转换为.py文件，如下图所示：



- 或者通过在conada中输入如下命令：

```
pyuic5 -o C:\Users\liuya\Desktop\SR_work\miangui.py C:\Users\liuya\Desktop\SR_SToF100_test_system\miangui.ui
```

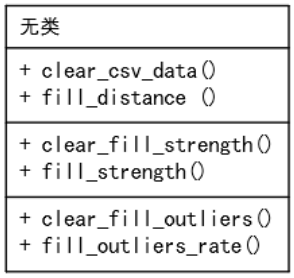
- .ui生成的.py文件不需要复制进工程文件夹，只需要复制生成的.py文件中的类，在ST_ToF_system.py中替换对应的类，如下图所示。



```
#主窗口
class Main_Window(QMainWindow, Ui_MainGUI):
    # valueChange = pyqtSignal(int)
    def __init__(self, parent=None):
        super(Main_Window, self).__init__()
        self.setupUi(self)
```

- import调用.ui的类的好处是方便在每次修改GUI界面布局后直接通过PyUIC自动修正代码，无需在主函数中重复复制修改

CSV文件的读写：




```
import csv
from ProjectPath import projectPath       #获取当前文件夹路径
distance_csv_path = (projectPath + "\\Matplot_data\\Distance_data.csv")
strength_csv_path = (projectPath + "\\Matplot_data\\Strength_data.csv")
outliersRate_csv_path = (projectPath + "\\Matplot_data\\outliersRate.csv")

# 将数据写入文件
def clear_csv_data():
    with open(distance_csv_path, "w") as cf:
        cf.truncate()
def fill_distance_strength(data1,data2,data3):
    with open(distance_csv_path, "a", newline="") as cf:
        w = csv.writer(cf)
        w.writerow([data1, data2, data3])
        cf.close()

#记录信号强度
def clear_fill_strength():
    with open(strength_csv_path, "w") as sf:
        sf.truncate()
def fill_strength(data1, data2):
    with open(strength_csv_path, "a", newline="") as sf:
        Strength = csv.writer(sf)
        Strength.writerow([data1, data2])
        sf.close()

#异常点记录
def clear_fill_outliers():
    with open(outliersRate_csv_path, "w") as rf:
```

```
        rf.truncate()
def fill_outliers_rate(data1,data2):
    with open(outliersRate_csv_path, "a", newline="") as rf:
        outliersRate = csv.writer(rf)
        outliersRate.writerow([data1, data2])
        rf.close()
```

其中，对于文件夹路径的获取，需要调用文件ProjectPath.py

```
import os

projectPath = os.path.split(os.path.abspath(__file__))[0]

if __name__ == "__main__":
    print(projectPath)
```

- 实现了对上位机抓取过滤的数据保存问csv文件，并可以在后台查看的功能
- 实现每次启动测试对前一次数据的擦除功能

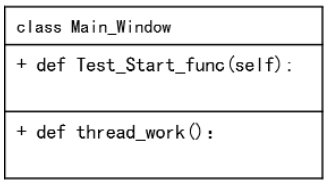
绘图功能可以通过读取csv文件中的数据进行绘图，其中测距结果和异常点的数据是迭代更新的，信号强度是每轮检测实时更新。

```
def DrawDataplot():
    # plt.ion()  #交互模式
    #取数据
    df = pd.read_csv(projectPath + "\\Matplot_data\\Distance_data.csv")
    alt = array(df)
    # print(alt)
    polar_angle = []
    distance = []
    for a in alt:
        polar_angle.append(a[0])
        distance.append(a[1])
    polar_angle = array(polar_angle)
    distance = array(distance)
    # print(x0)
    print(distance)

    #设置下面所需要的参数
    angle = polar_angle  #极角
    r = distance         #距离
    #polar表示绘制极坐标图，颜色=蓝色；线的类型=空；线宽=0；标志点样式="."；标志点的大小=4
    plt.polar(angle, r, color ="b", linestyle='None' ,marker=".", ms=4)
    #将生成的图片保存到/Matplot_data/文件夹下
    # PNGfileSave()
    Distance_img_path = (projectPath + "\\Matplot_data\\Distance_image" + ".png")   #文件名为Distance_image.png
    plt.savefig(Distance_img_path, dpi=500, bbox_inches='tight')

    #绘图展示
    # plt.show(block = False)
    plt.close()
```

主线程：

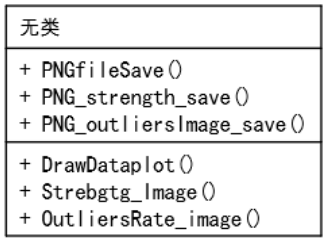


- 循环测试模式：

```
if global_value.stop_flag == 0:
   if X_back_DemandFlag == 1:
       self.X_TestRange_back()
       while global_value.X_bcak_completedFlag == 0:
          DriveCAN().can_knicoX_write_manual(
                  send_data=[0x40, 0x41, 0x60, 0x00, 0x00, 0x00, 0x00,
                  0x00])  # 读取驱动器状态字，判断X轴是否已经回到初始指定位置
          time.sleep(0.1)
          print('正在检查是否以回初始位置')
```

```
            DriveCAN().can_X_ReadStatusword()
        self.Z_Test_active()  # Z轴动作，向下遍历，分辨率为10mm
        global_value.DATA_vertical = global_value.DATA_vertical + 10
        global_value.DATA_distance = global_value.X_init_location
        X_back_DemandFlag = 0
    else:
        pass
else:
    pass
```

绘图功能实现：

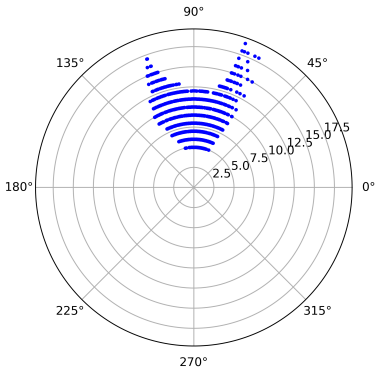


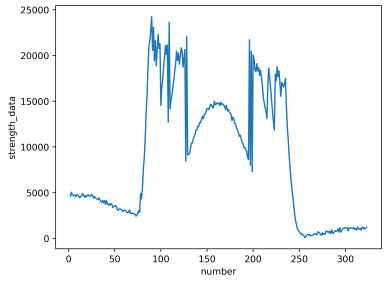
- 读取CSV文件中的数据
- 绘制极坐标下的散点图图，测距图像保存在/Matplot_data/Distance_data.csv
- 绘制平面坐标下的折线图，信号强度图像和异常率图像分别保存在/Matplot_data/Strength_data.csv和/Matplot_data/outliersRate.csv下

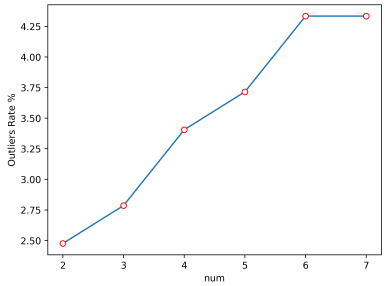
```
def DrawDataplot():
    # plt.ion()  #交互模式
    #取数据
    df = pd.read_csv(projectPath + "\\Matplot_data\\Distance_data.csv")
    alt = array(df)
    # print(alt)
    polar_angle = []
    distance = []
    for a in alt:
        polar_angle.append(a[0])
        distance.append(a[1])
    polar_angle = array(polar_angle)
    distance = array(distance)
    # print(x0)
    print(distance)

    #设置下面所需要的参数
    angle = polar_angle  #极角
    r = distance         #距离
    #polar表示绘制极坐标图，颜色=蓝色；线的类型=空；线宽=0；标志点样式="."；标志点的大小=4
    plt.polar(angle, r, color ="b", linestyle='None' ,marker=".", ms=4)
    #将生成的图片保存到/Matplot_data/文件夹下
    Distance_img_path = (projectPath + "\\Matplot_data\\Distance_image" + ".png")   #文件名为Distance_image.png
    plt.savefig(Distance_img_path, dpi=500, bbox_inches='tight')

    #绘图展示
    # plt.show(block = False)
    plt.close()
```







- 需要注意：在文件打包时，不能开启plt.show()函数，因为该函数是一个阻塞函数，在Pycharm环境下默认为交互（interactive）模式，但当打包后运行脚本，程序会默认为阻塞（block）模式，导致报错：如下代码，原因就是plt.show()函数将主线程阻塞

```
ERROR：signal only works in main thread
```

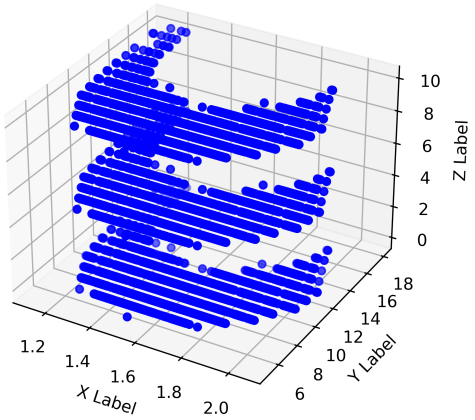
过滤方案：

- 根据实验检查发现，自研ToF存在一定的异常点，例如：实际与障碍物距离只有10cm以内，但返回的CAN报文经过解析后得到了一个远大于10cm但信号强度极小的值，因此认为这类点为异常点。为保证测试结果的合理性，在软件代码中将这些被认为是异常点的数值过滤并记录下来，最终在PDF报告上会显示本次测试所得的异常点率
- 过滤方案如下：
- 注意线程与global_value类中的变量

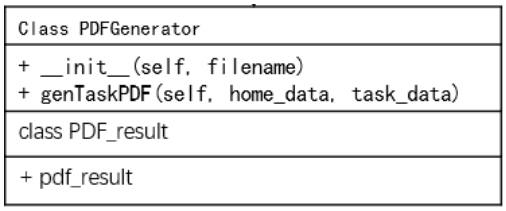
```
if readResult[1].DATA[0] == 0x05:
   distance = (readResult[1].DATA[2] << 8) | (readResult[1].DATA[1])
   signal_strength = (readResult[1].DATA[4] << 8) | (readResult[1].DATA[3])
      if distance != 0x96:
         num = num + 1
         print("num = %s" %(num))
         # signal_strength = (readResult[1].DATA[4]<<8)|(readResult[1].DATA[3])
         fill_strength(num, signal_strength)
         if signal_strength > 0x1B58:#信号强度过滤 > 7000
            global_value.normal_num = global_value.normal_num + 1  # 正常点个数
            fill_distance_strength(Polar_angle[num], distance, signal_strength)
            print("num = %s  极角 = %s  Distance = [%s]  信号强度 = %s  十六进制为 = %x"
                    % (num, Polar_angle[num],
                    distance, signal_strength, signal_strength))
            angle_data = 90 - math.degrees(Polar_angle[num])
            mainWindow.DataDisplay_Window.textBrowser.append("样点编号：%s  取点详情：角度 = %.5f 测距 = %s cm 信号强度 = %s"
                                                             % (num, angle_data, distance, signal_strength))
         else:
             global_value.outliers_num = global_value.outliers_num + 1  #异常点个数
             mainWindow.DataDisplay_Window.textBrowser.append("【异常】：信号强度异常, 信号强度 = %s"
                                                              %(signal_strength))
     else:
          num = num + 1
          print("num = %s" % (num))
          fill_strength(num, signal_strength)
          mainWindow.DataDisplay_Window.textBrowser.append("【无效】：测距无效（150cm）,信号强度 = %s"
                                                           %(signal_strength))
else:
     pass
```

3D测距图的生成：

由于目前使用的Python-matplotlib的限制，无法完美的生成柱坐标图（如下图所示），所以目前的方式只能讲3D的数据导出后通过MATLAB进行绘制



PDF的登记和生成：



读取绘图结果：测距图以及异常率图；读取并计算global value类中的变量

生成的PDF保存在/PDFs文件夹中，命名格式为：PDFreport_受检ToF型号_受检ToF编号_日期.pdf

例如：PDFreport_ToF100_1_20210611.pdf

- PDF报告结果生成的方法：

```
class PDF_result():
    # def __init__(self):
    def pdf_result():
        # 格式化成2016-03-20 11:45:39形式
        data = time.strftime("%Y-%m-%d %H:%M:%S", time.localtime())
        name_data = time.strftime("%Y%m%d", time.localtime())
        reportname = global_value.ToF_type + '_' + global_value.ToF_num + '_' + name_data
        """主程序"""
        #获得的产品数据
        home_data = {'report_code': reportname,
                   'task_name':global_value.ToF_type,
                   'report_date': data ,
                   'report_creator': global_value.inspector_person
                  }
        location_X = str(global_value.X_goal_location)
        init_X = str(global_value.X_init_location)
        init_Z = str(global_value.Z_init_location)
        location_Z = str(global_value.DATA_vertical)
        rate = str(((global_value.PDF_outliers_num)/(global_value.PDF_normal_num))*100)
        # rate = str(((global_value.PDF_outliers_num) / 2) * 100)  #无数据测试用，替换上面一句，否则会报错

        task_data = [
                        ['测试者：', global_value.inspector_person],
                        ['测试时间：', data],
                        ['受检ToF型号：', global_value.ToF_type],
                        ['受检ToF编号：', 'num_' + global_value.ToF_num],
                        ['异常率：', rate + '%'],
                        ['ToF水平方向检测范围（mm）', init_X + ' ~ ' + location_X],
                        ['ToF竖直方向移动范围（mm）', init_Z + ' ~ ' + location_Z],
                        ['ToF旋转扫描范围（度）', '-40° ~ 40°']
                    ]

        #PDF类，读取产品数据并生成PDF
```

```
        test = PDFGenerator("report_" + reportname)
        test.genTaskPDF(home_data,task_data)
```

- 在测试结果后，测试人员可以通过上位机输出本次测试结果的PDF报告，报告内包含：
  - 报告编号
  - 测试名称
  - 报告日期
  - 负责人
  - 测试数据：
    - 测试人员
    - 测试时间
    - 受检ToF型号
    - 受检ToF编号
    - 异常率
    - ToF水平方向检测范围（mm）
    - ToF竖直方向检测范围（mm）
    - ToF旋转扫描范围（度°）
  - SToF100测试结果图
  - SToF100异常率结果图

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/bb3cab20-6d24-49a2-97d6-3828b8c7497f/PDFreport_ToF100_1_20210622.pdf