****

**J I A N G S U U N I V E R S I T Y**

《物联网系统实践3》

感知与控制

课程设计报告



学院名称： 计算机学院

专业班级： 物联网工程2303

学生姓名： 邱佳亮

学生学号： 3230611072

指导教师姓名： 李峰

2025年 6月

目录

[1. 实验目的和要求 1](#_Toc13962)

[1.1实验目标 1](#_Toc18366)

[1.2 实验内容 1](#_Toc24696)

[2. 实验设计 2](#_Toc12409)

[2.1 实验分工 2](#_Toc13404)

[2.2 UI设计 2](#_Toc17839)

[2.3 发送模块 4](#_Toc4272)

[2.4 接收模块 4](#_Toc28912)

[2.5 主模块 5](#_Toc4568)

[2.6 开关快递柜模块 6](#_Toc6616)

[2.7 温度调整模块 6](#_Toc29346)

[2.8 压缩机控制模块 7](#_Toc17174)

[2.9 绘图模块 7](#_Toc31152)

[2.10 API模块 7](#_Toc29871)

[2.11 网页 8](#_Toc9016)

[3. 实验结果 8](#_Toc1341)

[3.1 开关柜 8](#_Toc29120)

[3.2 修改温度 10](#_Toc8759)

[3.3 启动压缩机 10](#_Toc26046)

[3.4 绘制温度变化 11](#_Toc14226)

[3.5 API服务 11](#_Toc23371)

[3.6 Web 12](#_Toc6448)

[4. 实验总结 12](#_Toc17988)

[5. 实验代码 13](#_Toc9461)

[5.1 main.py 13](#_Toc18568)

[5.2 send.py 17](#_Toc2698)

[5.3 receive.py 20](#_Toc15171)

[5.4 com\_init.py 22](#_Toc3592)

[5.5 Dialog\_mach.py 22](#_Toc19293)

[5.6 Dialog\_open.py 25](#_Toc1312)

[5.7 Dialog\_pic.py 28](#_Toc2691)

[5.8 Dialog\_temp.py 30](#_Toc2694)

[5.9 global\_var.py 32](#_Toc8554)

[5.10 api\_server.py 32](#_Toc419)

[5.11 main\_ui.py 33](#_Toc30011)

[5.12 style.qss 41](#_Toc5335)

现场快递柜状态采集与控制系统

# 实验目的和要求

## ·1.1实验目标

（1）了解现场网感知与控制的基本方法。

（2）掌握基于 RS232 的协议设计与通信方法，在此基础上实现对现场设备状态数据的

采集、对设备的控制。

（3）设计实现一个对现场快递柜状态数据采集（包括当前温度和设定温度、各抽屉开

关状态、压缩机运行状态和系统运行状态等）、显示、参数设置、抽屉打开、保鲜控制等功

能软件系统。

## 1.2 实验内容

（1）设计实现快递柜控制板仿真软件，实现对快递柜控制板状态数据的采集与显示，包括当前温度、设定温度（控制温度）、压缩机状态（压缩机开始工作时，快递柜将降温；压缩机停止工作时，快递柜将逐步升温，因此通过压缩机的开停来控制温度再设定值附近变化）、10 个抽屉的开关状态。其中当前温度可以设置某种变化规律，例如压缩机启动后，每秒钟当前温度降 0.2 度，但不能超过控制下限值（设定值-指定阈值），例如-10度，如果温度降到低于控制下限值，则压缩机停止工作；当压缩机停止工作后，当前温度将每秒上升 0.1 度，但不能超过控制上限值（设定值+指定阈值），例如 60 度等。

（2）在快递柜控制板仿真软件中，依据控制温度和压缩机的启停控制，实现对快递柜

控制板温度的控制，控制精度为 1 度。

1. 理解快递柜控制板仿真软件的通信协议（见附件），并设计实现，进而实现与快递柜控制板仿真软件（见附件）的通信（对于控制命令要考虑通过握手机制实现可靠传输）
2. 设计实现现场快递柜状态采集与控制系统软件，如图 1 所示。实现对所控制快递柜控制板状态数据的采集与显示，包括当前温度、设定温度（控制温度）、压缩机工作状态、10 个抽屉的开关状态等；同时，实现对开关指定抽屉、启停温度控制（压缩机制冷控制）、设置控制温度、以及设置系统参数等设置操作。
3. 现场快递柜状态采集与控制系统软件中，可以以曲线方式显示 1 小时内的当前温度和设定温度的变化趋势，控制板的温度采集间隔为 10 秒。
4. 在此基础上，有兴趣的同学可以进一步完成扩展部分，即设计基于 B/S 的在现场快递柜监控网站中，完成对各现场柜的远程监控。

# 实验设计

## 实验分工

邱佳亮：后端控制软件开发

陈默霖：前端网页开发

## UI设计

本实验我们使用PyQt6界面开发，具体而言，使用Qt Designer设计基本框架，将.ui文件编译得到python程序，最后利用设计QSS样式表美化界面。

我们设计的主界面如下：



图 1 主界面

主界面左侧呈现了相关的系统参数和控制按钮，包括开关柜、控制压缩机、控制温度和绘制温度曲线。右侧上方是发出控制命令的日志，下方更新快递柜状态。

四个子窗口如下：

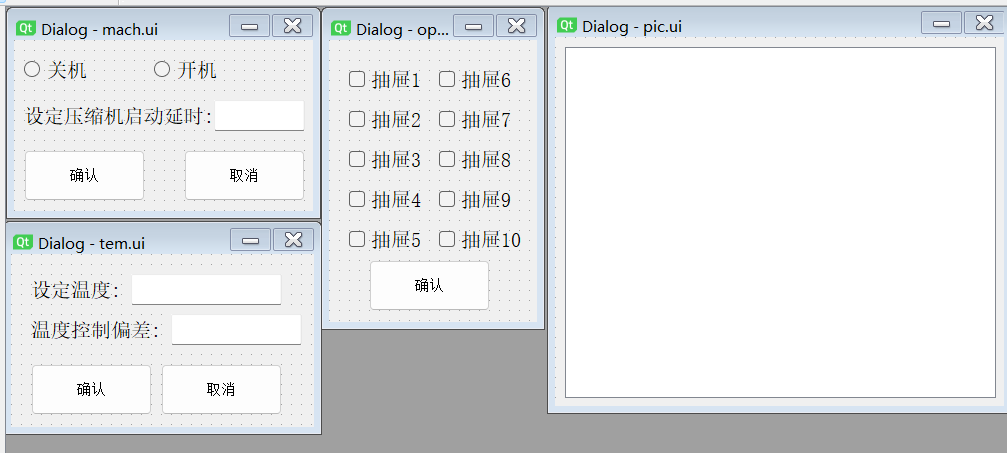


图 2 子窗口

在Dialog中可以控制压缩机开关、启动延时、设定温度和温度控制偏差、对每个快递柜进行开关，以及展示温度变化趋势。

最后我们通过QSS样式表对UI进行了美化，最终的呈现结果如下：



图 3 最终界面

## 发送模块

发送模块负责将控制指令按照通信协议封装成帧，并通过串口发送到下位机（控制板）。我们设计了如下函数：

packet\_data\_to\_frame：该函数是数据封装的核心，负责将数据按照协议规定的帧格式进行封装。它接收设备地址、功能号和数据作为输入，生成完整的帧数据。其中将数据封装成帧的部分使用了CRC16校验算法，我们通过crcmod中的mkCrcFun函数实现。

query\_frame：用于生成查询帧，功能号为1，无数据部分。该帧用于向上位机请求当前控制板的状态信息。

compressor\_control\_frame：生成启停压缩机控制帧，功能号为2。通过输入的布尔值决定是启动还是停止压缩机。

box\_control\_frame：生成开锁帧，功能号为3。根据输入的抽屉开关状态列表，控制指定抽屉的开锁或关锁操作。

set\_temperature\_frame：生成设置温度帧，功能号为4。将设定的温度值转换为特定字节格式，并封装到数据帧中。

set\_parameter\_frame：生成设置参数帧，功能号为5。根据输入的参数字典，组装并发送设备参数设置帧。

set\_temperature\_control\_deviation\_frame：生成设置温度控制偏差帧，功能号为6。用于设置温度控制的偏差值。

set\_device\_address\_frame：生成设置设备地址帧，功能号为9。根据输入的设备编码和新地址，更新控制板的设备地址。

temp\_convert：将温度值转换为特定的字节格式，以便在设置温度帧中正确表示。

init\_serial：初始化串口通信，设置端口、波特率和超时时间，为后续的数据发送做准备。

send\_frame：通过串口发送封装好的帧数据，并可将发送的帧信息发送给用户界面进行显示。

## 接收模块

接收模块负责从串口接收数据帧，并根据通信协议解析帧内容，更新系统状态或触发相应的控制动作。通过持续监听串口，接收从下位机发送来的数据。对接收到的数据进行解析，提取出完整的帧数据，并根据帧的格式和内容进行相应的处理。根据帧的长度和功能号识别不同类型的帧（如状态帧、参数帧、确认帧等），并对每种类型的帧执行特定的处理逻辑，如更新系统参数、控制压缩机等。并将解析出的帧内容更新到系统的全局状态中，以便其他模块（如用户界面）能够获取最新的设备状态信息。

我们设计了如下的函数和类：

SerialReceiver 类：继承自 QThread，用于在后台线程中持续监听串口数据。其 run 方法在一个循环中检查串口是否有可读数据，接收到数据后调用 receive\_frame 函数进行帧解析，并通过 frame\_received 信号将解析后的帧发送给其他模块。stop 方法用于停止接收线程。

receive\_data 函数：虽然未在提供的代码中直接使用，但通常用于接收数据并通过回调函数处理接收到的帧。

process\_frame 函数：根据帧的长度识别帧类型，并执行相应的处理逻辑。对于状态帧，提取设备编码、系统状态、压缩机状态、温度设定值、当前温度和抽屉状态等信息，并更新 control\_table\_status 字典；对于参数帧，提取设备参数并更新 control\_table\_attributes 字典；对于确认帧，记录状态信息。

get\_temp 函数：将温度字节数据转换为浮点数，考虑符号位、整数部分和小数部分。

compressor\_control 函数：根据当前温度与设定温度的偏差，控制压缩机的启停。如果当前温度低于设定温度减去偏差，则关闭压缩机；如果当前温度高于设定温度加上偏差，则打开压缩机。

receive\_frame 函数：从接收到的原始数据中提取完整帧。通过识别帧头（0xFFFF）和帧尾（0xFFF7），从数据流中分割出完整的帧，并返回这些帧的列表。

## 主模块

主模块是智能保鲜柜控制系统的入口点，负责初始化和协调各个功能模块，包括用户界面、数据接收和发送、以及Web API服务。首先，启动Flask API服务，初始化串口通信，并设置主窗口的用户界面。使用多线程确保数据接收和用户界面响应的实时性。我们定义了主窗口界面布局，处理用户交互事件，管理对话框的显示和隐藏。

MainWindow：主窗口类，继承自 QtWidgets.QMainWindow 和 Ui\_MainWindow。负责初始化主窗口的用户界面，连接按钮事件，并管理对话框的显示。

op：打开对话框的函数。根据传入的对话框类，创建对话框实例并显示。根据对话框类型，连接相应的按钮点击事件到对应的处理函数。

update\_state：更新抽屉状态显示。根据抽屉状态列表，更新表格中的文本显示为“开启”或“关闭”。

handle\_received\_frame：处理接收到的数据帧。根据帧长度判断帧类型，解析帧内容并更新界面显示，如温度、压缩机状态、抽屉状态等。

start\_monitor：启动数据接收监控。创建 SerialReceiver 实例并启动，用于监听串口数据。

log：记录帧信息。将帧的关键信息插入到界面表格中，用于显示通信日志。

## 开关快递柜模块

开关快递柜模块负责控制各抽屉的开关状态，通过用户界面接收用户的操作指令，并将这些指令按照通信协议发送到下位机（控制板）。在用户修改抽屉状态后，模块会将新的状态信息发送到控制板，并等待控制板的响应以更新本地状态显示，确保界面显示与实际设备状态一致。其中包括了如下的函数：

在 init\_state 函数中，根据全局变量中的当前抽屉状态设置复选框的初始勾选状态，确保界面显示与设备实际状态一致。

用户可以通过勾选或取消勾选复选框来改变抽屉的开关状态。在 update\_statu 函数中，收集所有复选框的当前状态，构造控制数据，并调用 box\_control\_frame 函数生成控制帧。

将生成的控制帧通过 send\_frame 函数发送到控制板，以实现对抽屉的远程控制。

## 温度调整模块

温度调整模块负责设定智能保鲜柜的目标温度和温度控制偏差，并将这些设置发送到控制板以实现精确的温度控制。模块提供一个简洁的对话框界面，包含两个输入框，分别用于设定温度和温度控制偏差。用户可以输入新的温度值和偏差值，并通过按钮提交设置。根据用户输入的设置，生成相应的控制指令帧，并通过串口发送到控制板。指令帧的格式遵循之前定义的通信协议。

set\_temp：处理用户的温度设置操作。获取输入框中的设定温度和温度控制偏差，更新全局数据字典，并生成相应的控制帧发送到控制板。

cancel：关闭对话框，取消当前操作。

## 压缩机控制模块

压缩机控制模块负责控制智能保鲜柜压缩机的启停，并设置压缩机启动延时参数。在界面加载时，根据当前系统状态初始化单选按钮和输入框的值，确保界面显示与设备实际状态一致。处理用户的输入和选择，更新系统状态，并生成相应的控制指令帧发送到控制板。

init\_mach：初始化界面元素状态，根据全局变量中的压缩机状态设置单选按钮和输入框的初始值，确保界面显示与设备实际状态一致。

set\_mach：处理用户的压缩机控制操作。获取用户输入的压缩机启动延时和选择的压缩机状态，更新全局数据字典，并生成相应的控制帧发送到控制板。

cancel：关闭对话框，取消当前操作。

## 绘图模块

绘图模块负责实时显示智能保鲜柜的温度变化趋势，以曲线方式直观展示当前温度和设定温度随时间的变化。我们使用 Matplotlib 库创建温度变化曲线图，横轴表示时间步，纵轴表示温度值。通过定时器定期更新曲线图，确保显示的温度数据是最新的。

setupUi：初始化对话框界面，创建绘图组件并将其添加到对话框布局中。设置定时器，定期调用 update\_plot 函数更新曲线图。

on\_close：在对话框关闭时调用，停止定时器并清理绘图组件，释放资源。

update\_plot：定期更新温度变化曲线图。清除之前的绘图内容，使用全局变量中的温度数据重新绘制曲线图，设置绘图样式、标题、标签、图例等，并刷新绘图组件显示。

## API模块

API 模块的设计旨在提供一个RESTful接口，用于获取智能保鲜柜的实时温度数据和其他系统状态信息。使用 Flask 框架创建一个简单的 Web 服务器，定义一个API端点，用于返回当前温度、设定温度、温度控制偏差、机器状态、温度历史记录等信息。通过访问全局变量，实时获取最新的温度数据和系统状态，并将其返回给请求者。将系统状态数据封装为JSON格式，便于前端或其他系统进行访问和解析。为了服务web端，我们配置了CORS以允许不同源的客户端访问API，确保前端界面能够顺利获取数据。

get\_temperature：该函数是一个路由处理函数，用于处理对/api/temperature端点的GET请求。它从全局变量中获取当前温度、设定温度、温度控制偏差、机器状态、温度历史记录和抽屉状态等信息，并将这些信息封装成JSON格式返回给客户端。

run\_flask\_app：用于启动Flask应用。它配置了应用的运行参数，如禁用调试模式和重新加载，以确保应用在多线程环境中稳定运行。

## 网页

网页模块的设计旨在提供一个基于Web的远程监控界面，使用户能够通过浏览器访问智能保鲜柜的状态信息。通过定时器定期从后端API获取最新数据，确保显示的信息是最新的。

fetch\_device\_status：后台线程函数，定时从本地API获取设备状态数据，并更新全局变量device\_status。

index：处理根路由请求，渲染包含HTML、CSS和JavaScript的监控页面模板。

status：定义一个API端点/status，返回当前设备状态的JSON数据，供前端页面定时请求获取最新状态。

main：启动后台线程以获取设备状态，然后启动Flask Web服务。

# 实验结果

## 开关柜

首先打开虚拟串口驱动，新建虚拟串口COM1和COM2：

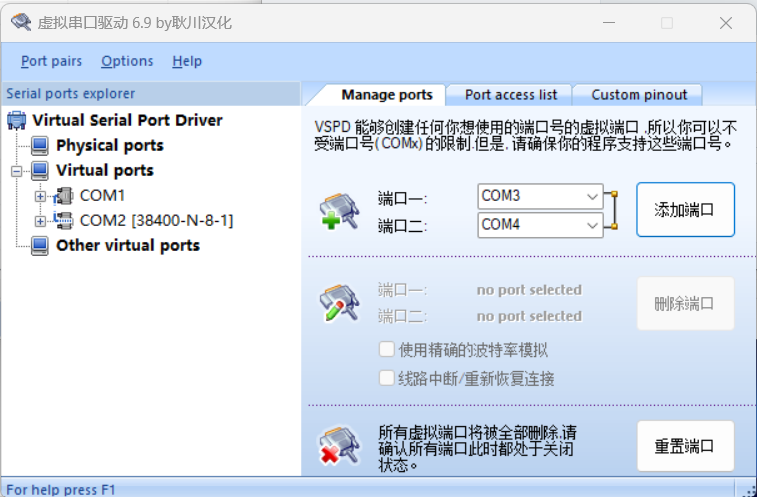


图 4 设置虚拟串口

运行程序和RCP\_ET\_exe，发现控制软件收到了快递柜发送的状态帧，设定温度等参数已经与快递柜同步：



图 5 打开软件

此时，点击开关柜按钮，选择打开1 2 3号柜，确定。可以看到，在表格中记录了发送的控制帧的信息，模拟程序中抽屉1 2 3已打开。软件根据ACK帧更新了快递柜状态表：



图 6 开关柜

## 修改温度

点击温度控制按钮，输入设定温度和温度偏差，确定。可以看到表格中记录了发送的控制帧，模拟软件的系统参数中设定温度和偏差更新为1和1，软件中相应的参数也更新完成：

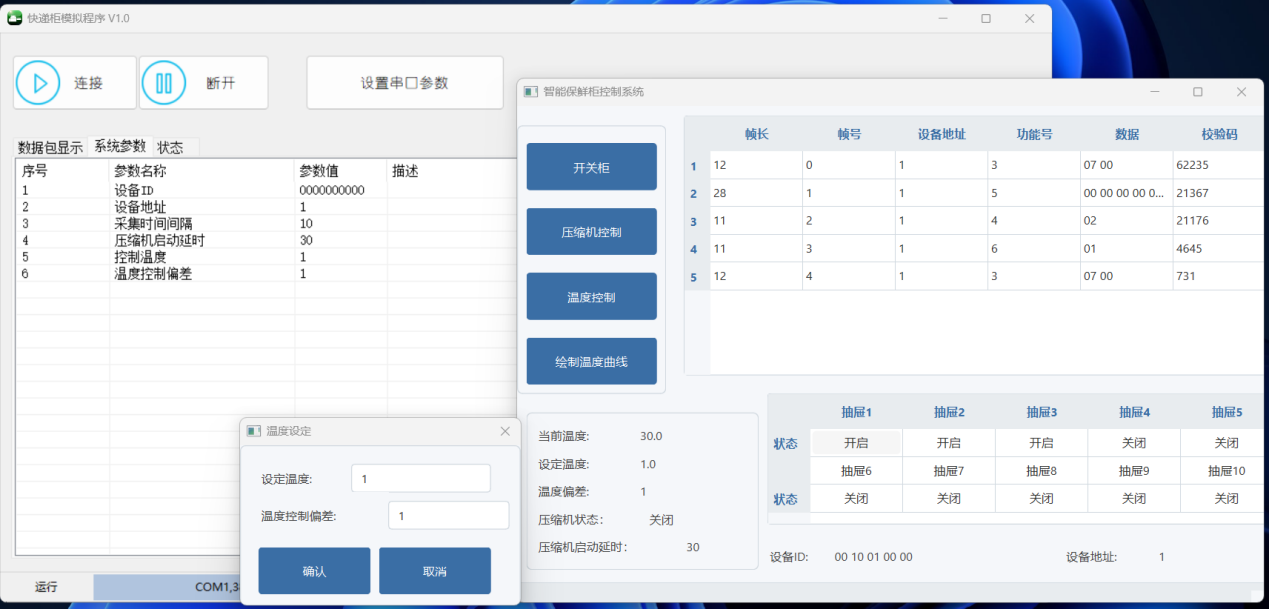


图 7 控制温度

## 启动压缩机

点击压缩机控制按钮，点击打开并将时延设置为5s，确定。此时模拟软件系统参数中压缩机启动延时参数修改完毕，显示压缩机进入预启动状态，5s后将启动：

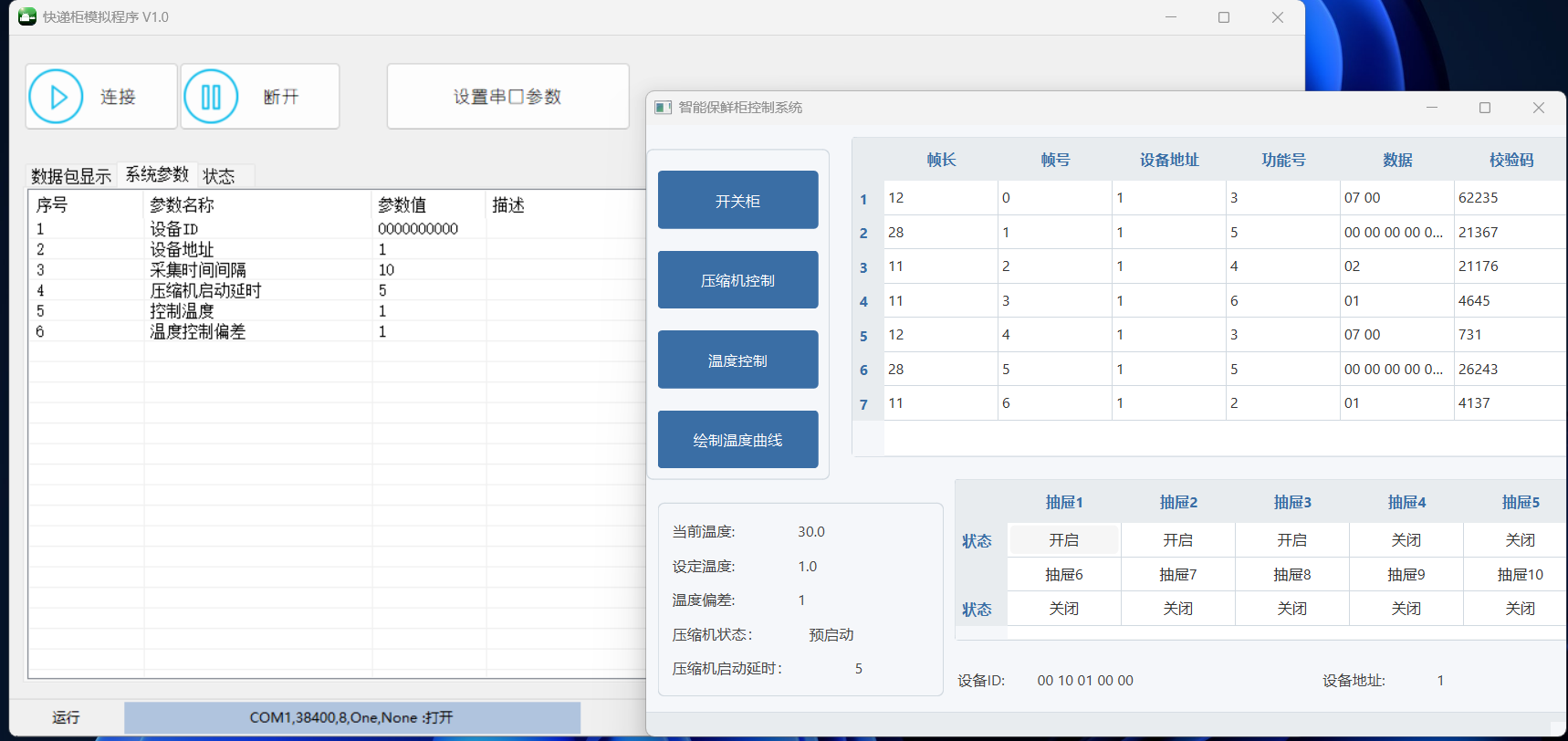


图 8 控制压缩机

## 绘制温度变化

随着温度下降，点击绘制温度曲线按钮可以查看温度变化趋势：

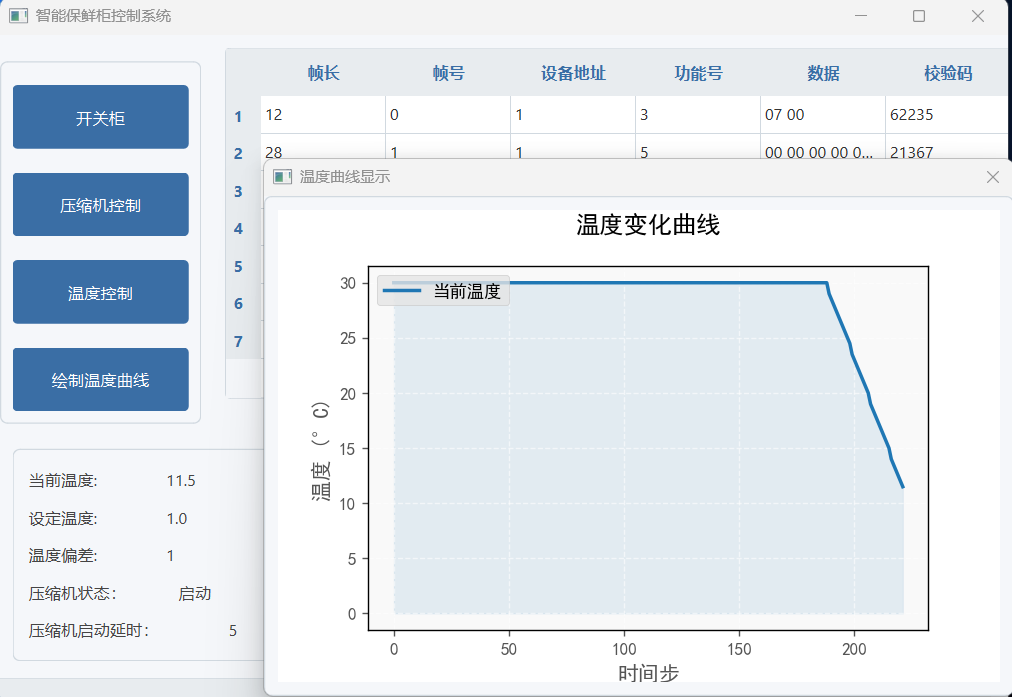


图 9 绘制温度变化

## API服务

我们设计了Restful API接口，软件运行时api服务器自动启动，访问http://127.0.0.1:5000/api/temperature，就可以看到当前温度、压缩机状态、快递柜状态等信息：

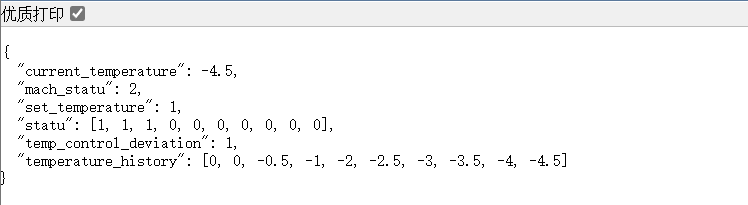


图 10 api服务

## Web

在软件运行同时启动web服务，就可以打开监控页面，查看到通过api获取的快递柜信息：



图 11 web服务

# 实验总结

通过本次实验，我成功设计并实现了一个智能保鲜柜的监控系统，取得了显著的效果。系统具备远程监控功能，用户可以随时随地通过Web界面查看设备的实时状态，包括温度、抽屉开关状态等关键信息。同时，系统实现了数据可视化，能够以直观的图表形式展示温度变化趋势，帮助用户更好地理解设备的运行情况。此外，系统采用了模块化设计，将用户界面、数据接收与发送、API服务等功能分离，提高了代码的可维护性和可扩展性。

在实验过程中，我显著提升了自己在多个方面的能力。首先，编程技能得到了极大锻炼，尤其是对Python、Flask和PyQt等技术的深入理解和灵活应用。其次，在系统设计方面，学会了如何从整体上规划和组织一个复杂的项目，考虑各个模块之间的交互和数据流动。最后，问题解决能力也得到了提升，通过不断调试和优化代码，成功解决了许多技术难题。

然而，实验中也遇到了一些问题和不足。例如，部分代码存在冗余，可优化的空间较大，未来可以进一步提高代码的效率和可读性。在异常处理方面，系统对于网络异常、设备断开等特殊情况的处理还不够完善，需要增强系统的稳定性和健壮性。此外，用户界面的交互设计还可以进一步改进，以提供更友好、更直观的用户体验。

总体而言，本次实验不仅让我掌握了智能监控系统的核心技术，还积累了宝贵的项目开发经验。通过总结经验教训，我明确了后续改进的方向，为未来更复杂的项目开发奠定了坚实的基础。

# 实验代码

## main.py

1. from PyQt6 import QtWidgets
2. import sys
3. from send import \*
4. from receive import \*
5. from Dialog\_open import Ui\_Dialog\_open
6. from Dialog\_pic import Ui\_Dialog\_pic
7. from Dialog\_mach import Ui\_Dialog\_mach
8. from Dialog\_temp import Ui\_Dialog\_temp
9. from mian\_ui import Ui\_MainWindow
10. import threading
11. from api\_server import run\_flask\_app
12. from com\_init import com
13. import global\_var
14. *# 主窗口类，继承自QtWidgets.QMainWindow和Ui\_MainWindow*
15. class MainWindow(QtWidgets.QMainWindow, Ui\_MainWindow):
16. def \_\_init\_\_(self):
17. super().\_\_init\_\_()
18. *# 调用setupUi方法初始化界面*
19. self.setupUi(self)
20. *# 打开样式表文件并设置样式*
21. with open(r"style.qss", "r",encoding='utf-8') as f:
22. self.setStyleSheet(f.read())
23. self.setWindowTitle("智能保鲜柜控制系统")
24. *# 手动连接按钮事件*
25. self.pushButton.clicked.connect(lambda: self.op(Ui\_Dialog\_open))
26. self.pushButton\_2.clicked.connect(lambda: self.op(Ui\_Dialog\_mach))
27. self.pushButton\_3.clicked.connect(lambda: self.op(Ui\_Dialog\_temp))
28. self.pushButton\_4.clicked.connect(lambda: self.op(Ui\_Dialog\_pic))
29. def op(self,window):
30. dialog =QtWidgets.QDialog()
31. *# 根据传入的窗口类初始化对应的对话框界面*
32. self.open\_form = window(main\_window=self)
33. self.open\_form.setupUi(dialog)
34. *# 根据不同的对话框类型，连接对应的按钮点击事件*
35. if isinstance(self.open\_form,Ui\_Dialog\_open):
36. self.open\_form.pushButton.clicked.disconnect()
37. self.open\_form.pushButton.clicked.connect(lambda: (
38. self.open\_form.update\_statu(),
39. dialog.accept(),  *# 关闭对话框*
40. ))
42. if isinstance(self.open\_form,Ui\_Dialog\_mach):
43. self.open\_form.pushButton.clicked.disconnect()
44. self.open\_form.pushButton.clicked.connect(lambda: (
45. self.open\_form.set\_mach(),
46. dialog.accept(),  *# 关闭对话框*
47. ))
49. if isinstance(self.open\_form,Ui\_Dialog\_temp):
50. self.open\_form.pushButton.clicked.disconnect()
51. self.open\_form.pushButton.clicked.connect(lambda: (
52. self.open\_form.set\_temp(),
53. dialog.accept(),  *# 关闭对话框*
54. ))
55. dialog.exec()
56. def update\_state(self,status):
57. *# 定义抽屉到表格行的映射关系*
58. drawer\_to\_row = {
59. 0: 0, 1: 0, 2: 0, 3: 0, 4: 0,
60. 5: 2, 6: 2, 7: 2, 8: 2, 9: 2
61. }
62. *# 遍历抽屉状态，更新表格显示*
63. for i in range(10):
64. row = drawer\_to\_row.get(i,0)
65. col = i % 5
66. state = "开启" if status[i] else "关闭"
67. item = self.tableWidget\_2.item(row, col)
68. if item is not None:
69. item.setText(state)
71. *# 处理接收到的数据帧的函数*
72. def handle\_received\_frame(self, frame):
73. if len(frame) == 44:  *# 状态帧*
74. current\_temp\_byte = frame[33]
75. set\_temp\_byte = frame[32]
76. temp\_p = frame[18]
77. temp\_now = get\_temp(current\_temp\_byte)
78. temp\_set = get\_temp(set\_temp\_byte)
80. self.label\_5.setText(f"{temp\_set:.1f}")
81. self.label\_4.setText(f"{temp\_now:.1f}")  *# 更新当前温度显示*
82. self.label\_8.setText(str(temp\_p))
83. global\_var.temp\_list.append(temp\_now)
84. print("温度已更新:", temp\_now)
86. *# 更新压缩机状态*
87. compressor\_status = frame[31]
88. delay=frame[14]
89. if compressor\_status==0:
90. self.label\_7.setText("关闭")
91. elif compressor\_status==1:
92. self.label\_7.setText("预启动")
93. elif compressor\_status==2:
94. self.label\_7.setText("启动")
95. else:
96. self.label\_7.setText("故障")
97. global\_var.mach\_statu = compressor\_status
99. self.label\_10.setText(str(delay))
101. device=frame[23:28]
102. self.label\_13.setText(" ".join(f"{b:02X}" for b in device))
104. addr = frame[11]
105. self.label\_14.setText(str(addr))
107. status\_byte= frame[36:38]
108. status=[]
109. for byte in status\_byte:
110. for i in range(8):
111. status.append((byte >> i) & 1)
112. self.update\_state(status[:10])
113. global\_var.statu=status
114. elif len(frame) == 28:  *# 参数帧*
115. temp\_set\_byte = frame[17]
116. temp\_set = get\_temp(temp\_set\_byte)
117. set = temp\_set
118. self.label\_5.setText(str(set))  *# 更新设定温度*
120. elif len(frame) == 14:
121. status\_byte= frame[8:10]
122. status=[]
123. for byte in status\_byte:
124. for i in range(8):
125. status.append((byte >> i) & 1)
126. s = status[:10]
127. self.update\_state(status[:10])

130. def start\_monitor(self,com):
131. *# 创建SerialReceiver对象并启动线程*
132. self.receiver=SerialReceiver(com)
133. self.receiver.frame\_received.connect(self.handle\_received\_frame)
134. self.receiver.start()

137. def log(self,frame\_info):
138. *# 在界面表格中插入帧信息*
139. row\_pos=self.tableWidget.rowCount()
140. self.tableWidget.insertRow(row\_pos)
141. headers = ["length", "frame\_number", "dev\_addr", "func\_code", "data", "checksum"]
142. for col, key in enumerate(headers):
143. self.tableWidget.setItem(row\_pos, col, QtWidgets.QTableWidgetItem(str(frame\_info[key])))
145. if \_\_name\_\_ == "\_\_main\_\_":
146. *# 启动 Flask 在单独线程中运行*
147. flask\_thread = threading.Thread(target=run\_flask\_app)
148. flask\_thread.daemon = True
149. flask\_thread.start()
150. *# 创建Qt应用实例并显示主窗口*
151. app = QtWidgets.QApplication(sys.argv)
152. window = MainWindow()
153. window.show()
154. try:
155. *# 启动数据接收监控*
156. window.start\_monitor(com)
157. sys.exit(app.exec())
158. finally:
159. if hasattr(window, 'receiver'):
160. window.receiver.stop()
161. com.close()

## send.py

1. from struct import pack
2. import serial
3. import crcmod
4. *# 全局变量*
5. frame\_num = 0  *# 帧号*
6. *# CRC16 校验算法*
7. crc16 = crcmod.mkCrcFun(0x18005, initCrc=0xFFFF, xorOut=0x0000)
8. *# 数据封装成帧*
9. def packet\_data\_to\_frame(addr, func\_num, data):
10. global frame\_num
11. frame\_head = pack('2B', 255, 255)  *# 0xFFFF 帧头，用于标识帧的开始*
12. frame\_length = pack('B', len(data) + 10)  *# 帧长，包括固定部分和数据部分的总长度*
13. num = pack('B', frame\_num)  *# 帧号，用于标识帧的顺序*
14. address = pack('B', addr)  *# 设备地址，目标控制板的地址*
15. fc\_num = pack('B', func\_num)  *# 功能码，表示该帧的功能（如查询、控制等）*
16. *# 计算 CRC 校验值，确保数据传输的可靠性*
17. crc\_check\_bytes = frame\_length + num + address + fc\_num + data
18. crc\_frame = pack('H', crc16(crc\_check\_bytes))
19. frame\_tail = pack('2B', 255, 247)  *# 0xFFF7 帧尾，用于标识帧的结束*
20. *# 组装完整的帧*
21. frame\_packet = frame\_head + frame\_length + num + address + fc\_num + data + crc\_frame + frame\_tail
22. frame\_num = (frame\_num + 1) % 256  *# 帧号循环递增*
23. return frame\_packet
24. *# 1. 查询帧(10Byte)*
25. def query\_frame(addr):
26. return packet\_data\_to\_frame(addr, 1, b'')  *# 功能号 1 表示查询帧，无数据部分*
27. *# 2. 启停压缩机控制帧(11Byte)*
28. def compressor\_control\_frame(addr, compressor\_control\_data):
29. control\_byte = pack('B', 1 if compressor\_control\_data else 0)  *# 1 表示启动压缩机，0 表示停止*
30. return packet\_data\_to\_frame(addr, 2, control\_byte)  *# 功能号 2 表示压缩机控制帧*
31. *# 3. 开锁帧(12Byte)*
32. def box\_control\_frame(addr, box\_control\_data):
33. byte1 = 0
34. byte2 = 0
35. *# 将抽屉开关状态转换为两个字节的数据*
36. for i in range(8):
37. byte1 += box\_control\_data[i] \* (2 \*\* i)
38. for i in range(8, 12):
39. byte2 += box\_control\_data[i] \* (2 \*\* (i - 8))
40. return packet\_data\_to\_frame(addr, 3, pack('2B', byte1, byte2))  *# 功能号 3 表示开锁帧*
41. *# 4. 设置温度帧(11Byte)*
42. def set\_temperature\_frame(addr, set\_temperature\_data):
43. temp = temp\_convert(set\_temperature\_data)  *# 将温度值转换为特定的字节格式*
44. return packet\_data\_to\_frame(addr, 4, pack('B', temp))  *# 功能号 4 表示设置温度帧*
45. *# 5. 设置参数帧(28Byte)*
46. def set\_parameter\_frame(addr, set\_parameter\_data):
47. *# 按照参数格式定义组装参数数据*
48. dev\_id = int(set\_parameter\_data['dev\_code'], 16).to\_bytes(5, 'big')
49. dev\_addr = pack('B', set\_parameter\_data['dev\_addr'])
50. time\_gap = pack('B', set\_parameter\_data['time\_gap'])
51. compressor\_delay = pack('B', set\_parameter\_data['compressor\_delay'])
52. set\_temp = pack('B', temp\_convert(set\_parameter\_data['set\_temp']))
53. temp\_control = pack('B', set\_parameter\_data['temp\_control'])
54. *# 功能号 5 表示设置参数帧*
55. data = dev\_id + dev\_addr + b'\x00' + time\_gap + compressor\_delay + b'\x00\x00' + set\_temp + temp\_control + b'\xFF\xFF\xFF\xFF\x00'
56. return packet\_data\_to\_frame(addr, 5, data)
57. *# 6. 设置温度控制偏差帧(11Byte)*
58. def set\_temperature\_control\_deviation\_frame(addr, set\_temperature\_control\_deviation\_data):
59. return packet\_data\_to\_frame(addr, 6, pack('B', set\_temperature\_control\_deviation\_data))  *# 功能号 6 表示设置温度控制偏差帧*
60. *# 7. 设置设备地址帧(16Byte)*
61. def set\_device\_address\_frame(addr, dev\_code\_data):
62. dev\_code\_bytes = int(dev\_code\_data, 16).to\_bytes(5, 'big')  *# 将设备编码转换为字节*
63. new\_addr = pack('B', addr)  *# 新的设备地址*
64. return packet\_data\_to\_frame(addr, 9, dev\_code\_bytes + new\_addr)  *# 功能号 9 表示设置设备地址帧*
65. *# 温度值处理*
66. def temp\_convert(set\_temperature\_data):
67. set\_temperature\_data = float(set\_temperature\_data)
68. s = str(set\_temperature\_data).split('.')
69. add\_bit1 = '0'  *# 整数部分的符号位*
70. add\_bit2 = '0'  *# 小数部分的符号位*
71. if int(s[0]) < 0:
72. add\_bit1 = '1'
73. num1 = abs(int(s[0]))  *# 整数部分取绝对值*
74. num2 = int(s[1]) if len(s) > 1 else 0  *# 小数部分，默认为 0*
75. if num2 >= 5:
76. add\_bit2 = '1'
77. *# 将整数部分转换为二进制字符串并补齐到 6 位*
78. res = bin(num1)[2:] if num1 != 0 else '0'
79. res = '0' \* (6 - len(res)) + res
80. res = add\_bit1 + res + add\_bit2  *# 组合符号位和数值位*
81. return int(res, 2)  *# 转换为整数*
82. *# 初始化串口*
83. def init\_serial(port='COM2', baudrate=38400):
84. return serial.Serial(port, baudrate, timeout=1)  *# 初始化串口，设置端口、波特率和超时时间*
85. *# 发送帧*
86. def send\_frame(frame, com, ui=None):
87. com.write(frame)  *# 通过串口发送帧*
88. *# 提取帧的关键信息用于日志记录*
89. frame\_info = {
90. 'length': len(frame),  *# 帧的总长度*
91. 'frame\_number': frame[3],  *# 帧号位于第 5 个字节*
92. 'dev\_addr': frame[4],  *# 设备地址位于第 6 个字节*
93. 'func\_code': frame[5],  *# 功能码位于第 7 个字节*
94. *# 数据段（去掉头部、帧长、帧号、地址、功能码和 CRC 校验部分）*
95. 'data': ' '.join(f'{byte:02X}' for byte in frame[6:-4]),
96. *# CRC 校验位计算范围为帧长 + 帧号 + 地址 + 功能码 + 数据部分*
97. 'checksum': int.from\_bytes(frame[-4:-2], byteorder='big')
98. }
99. print(f"帧已发送: {frame}")  *# 打印发送的完整帧用于调试*
100. if ui is not None:  *# 如果提供了用户界面对象，则将帧信息发送给界面进行显示*
101. ui.log(frame\_info)

## receive.py

1. from send import \*
2. from com\_init import com
3. from PyQt6.QtCore import QThread, pyqtSignal
4. *# 定义状态帧字典*
5. control\_table\_attributes = {}
6. control\_table\_status = {}
7. ack\_status = {}
8. class SerialReceiver(QThread):
9. frame\_received = pyqtSignal(object)  *# 定义一个信号，用于发送接收到的帧*
10. def \_\_init\_\_(self, com, parent=None):
11. super().\_\_init\_\_(parent)
12. self.com = com
13. self.running = True
14. def run(self):
15. while self.running:
16. if self.com.in\_waiting >= 1:  *# 判断是否有数据可读*
17. com\_data = self.com.read(self.com.in\_waiting)
18. frames = receive\_frame(com\_data)
19. for frame in frames:
20. self.frame\_received.emit(frame)  *# 发射信号，将接收到的帧传递给其他模块*
21. self.msleep(100)  *# 避免 CPU 占满*
22. def stop(self):
23. self.running = False
24. self.quit()
25. self.wait()
26. *# 数据接收*
27. def receive\_data(com, callback=None):
28. while True:
29. com\_data = com.read(80)  *# 读取数据*
30. if com\_data:
31. frames = receive\_frame(com\_data)  *# 解析帧*
32. for frame in frames:
33. process\_frame(frame)  *# 处理每个帧*
34. if callback:
35. callback(frame)
36. def process\_frame(frame):
37. if len(frame) == 44:  *# 状态帧*
38. control\_table\_status['dev\_code'] = frame[6:11]
39. control\_table\_status['sys\_status'] = frame[29]
40. control\_table\_status['compressor\_status'] = frame[31]
41. control\_table\_status['temp\_set'] = frame[32]
42. control\_table\_status['current\_temp'] = frame[33]
43. control\_table\_status['box\_status'] = frame[36:38]
44. print(f"控制表状态: {control\_table\_status}")
45. temp\_now = get\_temp(control\_table\_status['current\_temp'])
46. print(f"当前温度: {temp\_now}")
47. compressor\_control(temp\_now)  *# 根据当前温度控制压缩机*
48. elif len(frame) == 28:  *# 参数帧*
49. control\_table\_attributes['dev\_code'] = frame[6:11]
50. control\_table\_attributes['addr'] = frame[11]
51. control\_table\_attributes['time\_gap'] = frame[13]
52. control\_table\_attributes['compressor\_delay'] = frame[14]
53. control\_table\_attributes['temp\_set'] = frame[17]
54. control\_table\_attributes['temp\_control'] = frame[18]
55. print(f"控制表属性: {control\_table\_attributes}")
56. elif len(frame) == 14:
57. ack\_status['status'] = frame[8:10]
58. print(f"{ack\_status}")
59. def get\_temp(temp\_byte):
60. *# 第7位是符号位*
61. sign = -1 if (temp\_byte >> 7) & 1 else 1
62. *# 第6~1位是整数部分（取 bit6 到 bit1）*
63. integer\_part = (temp\_byte >> 1) & 0b111111
64. *# 第0位是小数部分*
65. decimal\_part = 0.5 if temp\_byte & 1 else 0.0
66. return sign \* (integer\_part + decimal\_part)
67. def compressor\_control(temp\_now):
68. temp\_set = get\_temp(control\_table\_status['temp\_set'])
69. temp\_control = control\_table\_attributes['temp\_control']
70. if temp\_now <= temp\_set - temp\_control:
71. compressor\_control\_frame\_data = compressor\_control\_frame(addr=0x01, compressor\_control\_data=False)
72. send\_frame(compressor\_control\_frame\_data,com)  *# 关闭压缩机*
73. print("关闭压缩机")
74. elif temp\_now >= temp\_set + temp\_control:
75. compressor\_control\_frame\_data = compressor\_control\_frame(addr=0x01, compressor\_control\_data=True)
76. send\_frame(compressor\_control\_frame\_data,com)  *# 打开压缩机*
77. print("打开压缩机")
78. def receive\_frame(hex\_datas):
79. res\_frames = []
80. i = 0
81. while i < len(hex\_datas) - 1:
82. if hex\_datas[i] == 0xFF and hex\_datas[i + 1] == 0xFF:
83. for j in range(i + 2, len(hex\_datas) - 1):
84. if hex\_datas[j] == 0xFF and hex\_datas[j + 1] == 0xF7:
85. frame = hex\_datas[i:j + 2]
86. if len(frame) in [14, 44]:
87. res\_frames.append(frame)
88. i = j + 2  *# 跳过已处理的帧*
89. break
90. i += 1
91. return res\_frames

## com\_init.py

1. from send import init\_serial
2. com = init\_serial(port="COM2",baudrate=38400)

## Dialog\_mach.py

1. from PyQt6 import QtCore, QtGui, QtWidgets
2. from send import \*
3. import global\_var
4. from com\_init import com
5. import time
6. class Ui\_Dialog\_mach(object):
7. def \_\_init\_\_(self, main\_window=None):
8. self.main\_window = main\_window  *# 用于与主窗口交互*
9. def init\_mach(self):
10. *# 初始化界面元素状态，根据全局变量中的压缩机状态设置单选按钮和输入框的初始值*
11. if global\_var.mach\_statu in [2, 1]:
12. self.radioButton.setChecked(True)  *# 开机状态*
13. else:
14. self.radioButton\_2.setChecked(True)  *# 关机状态*
15. self.lineEdit\_2.setText(str(global\_var.data["compressor\_delay"]))  *# 设置压缩机启动延时的初始值*
16. def set\_mach(self):
17. *# 获取用户输入的压缩机启动延时*
18. delay\_str = self.lineEdit\_2.text()
19. delay = int(delay\_str) if delay\_str else None
20. *# 获取用户选择的压缩机状态*
21. mach\_statu = self.radioButton.isChecked()
22. *# 如果输入有效，更新全局数据字典中的压缩机启动延时*
23. if delay is not None:
24. global\_var.data["compressor\_delay"] = delay
25. *# 生成并发送设置参数帧*
26. data\_p = set\_parameter\_frame(addr=0x01, set\_parameter\_data=global\_var.data)
27. send\_frame(data\_p, com, ui=self.main\_window)
28. time.sleep(1)
29. *# 生成并发送启停压缩机控制帧*
30. compressor\_control\_frame\_data = compressor\_control\_frame(addr=0x01, compressor\_control\_data=mach\_statu)
31. send\_frame(compressor\_control\_frame\_data, com, ui=self.main\_window)
32. time.sleep(1)
33. def cancel(self):
34. *# 关闭对话框*
35. self.Dialog.close()
36. def setupUi(self, Dialog):
37. self.Dialog = Dialog
38. Dialog.setObjectName("Dialog")
39. Dialog.resize(299, 171)
40. self.radioButton\_2 = QtWidgets.QRadioButton(parent=Dialog)
41. self.radioButton\_2.setGeometry(QtCore.QRect(10, 20, 115, 19))
42. font = QtGui.QFont()
43. font.setPointSize(12)
44. self.radioButton\_2.setFont(font)
45. self.radioButton\_2.setObjectName("radioButton\_2")
46. self.radioButton = QtWidgets.QRadioButton(parent=Dialog)
47. self.radioButton.setGeometry(QtCore.QRect(140, 20, 115, 19))
48. font = QtGui.QFont()
49. font.setPointSize(12)
50. self.radioButton.setFont(font)
51. self.radioButton.setObjectName("radioButton")
52. self.pushButton\_2 = QtWidgets.QPushButton(parent=Dialog)
53. self.pushButton\_2.setGeometry(QtCore.QRect(170, 110, 121, 51))
54. self.pushButton\_2.setObjectName("pushButton\_2")
55. self.lineEdit\_2 = QtWidgets.QLineEdit(parent=Dialog)
56. self.lineEdit\_2.setGeometry(QtCore.QRect(200, 60, 91, 31))
57. font = QtGui.QFont()
58. font.setPointSize(12)
59. self.lineEdit\_2.setFont(font)
60. self.lineEdit\_2.setObjectName("lineEdit\_2")
61. self.pushButton = QtWidgets.QPushButton(parent=Dialog)
62. self.pushButton.setGeometry(QtCore.QRect(10, 110, 121, 51))
63. self.pushButton.setObjectName("pushButton")
64. self.label\_2 = QtWidgets.QLabel(parent=Dialog)
65. self.label\_2.setGeometry(QtCore.QRect(10, 50, 191, 51))
66. font = QtGui.QFont()
67. font.setPointSize(12)
68. self.label\_2.setFont(font)
69. self.label\_2.setObjectName("label\_2")
70. self.retranslateUi(Dialog)
71. QtCore.QMetaObject.connectSlotsByName(Dialog)
72. with open(r"style.qss", "r", encoding='utf-8') as f:
73. Dialog.setStyleSheet(f.read())
74. Dialog.setWindowTitle("压缩机控制")
75. self.init\_mach()  *# 初始化界面状态*
76. def retranslateUi(self, Dialog):
77. \_translate = QtCore.QCoreApplication.translate
78. Dialog.setWindowTitle(\_translate("Dialog", "Dialog"))
79. self.radioButton\_2.setText(\_translate("Dialog", "关机"))
80. self.radioButton.setText(\_translate("Dialog", "开机"))
81. self.pushButton\_2.setText(\_translate("Dialog", "取消"))
82. self.pushButton.setText(\_translate("Dialog", "确认"))
83. self.label\_2.setText(\_translate("Dialog", "设定压缩机启动延时:"))
84. *# 连接确认按钮的点击事件到设置压缩机函数*
85. self.pushButton.clicked.connect(self.set\_mach)
86. *# 连接取消按钮的点击事件到关闭对话框函数*
87. self.pushButton\_2.clicked.connect(self. Cancel)

## Dialog\_open.py

1. from PyQt6 import QtCore, QtGui, QtWidgets
2. from send import \*
3. import global\_var
4. from com\_init import com
5. import time
6. class Ui\_Dialog\_open(object):
7. def \_\_init\_\_(self, main\_window=None):
8. self.main\_window = main\_window  # 用于与主窗口交互
9. def init\_state(self):
10. # 初始化界面元素状态，根据全局变量中的抽屉状态设置复选框的勾选状态
11. self.checkBox.setChecked(bool(global\_var.statu[0]))
12. self.checkBox\_2.setChecked(bool(global\_var.statu[1]))
13. self.checkBox\_3.setChecked(bool(global\_var.statu[2]))
14. self.checkBox\_4.setChecked(bool(global\_var.statu[3]))
15. self.checkBox\_5.setChecked(bool(global\_var.statu[4]))
16. self.checkBox\_10.setChecked(bool(global\_var.statu[5]))
17. self.checkBox\_6.setChecked(bool(global\_var.statu[6]))
18. self.checkBox\_7.setChecked(bool(global\_var.statu[7]))
19. self.checkBox\_8.setChecked(bool(global\_var.statu[8]))
20. self.checkBox\_9.setChecked(bool(global\_var.statu[9]))
21. def update\_statu(self):
22. # 收集各个复选框的当前状态，构造控制数据
23. statu = [
24. int(self.checkBox.isChecked()),     # 抽屉1
25. int(self.checkBox\_2.isChecked()),   # 抽屉2
26. int(self.checkBox\_3.isChecked()),   # 抽屉3
27. int(self.checkBox\_4.isChecked()),   # 抽屉4
28. int(self.checkBox\_5.isChecked()),   # 抽屉5
29. int(self.checkBox\_10.isChecked()),  # 抽屉6
30. int(self.checkBox\_6.isChecked()),   # 抽屉7
31. int(self.checkBox\_7.isChecked()),   # 抽屉8
32. int(self.checkBox\_8.isChecked()),   # 抽屉9
33. int(self.checkBox\_9.isChecked()),   # 抽屉10
34. 0, 0  # 保留两位，未使用的抽屉状态
35. ]
36. box\_control\_data = statu
37. unlock\_frame\_data = box\_control\_frame(addr=0x01, box\_control\_data=box\_control\_data)
38. send\_frame(unlock\_frame\_data, com, ui=self.main\_window)
39. time.sleep(1)
40. def setupUi(self, Dialog):
41. Dialog.setObjectName("Dialog")
42. Dialog.resize(208, 282)
43. self.pushButton = QtWidgets.QPushButton(parent=Dialog)
44. self.pushButton.setGeometry(QtCore.QRect(50, 220, 121, 51))
45. self.pushButton.setObjectName("pushButton")
46. self.checkBox = QtWidgets.QCheckBox(parent=Dialog)
47. self.checkBox.setGeometry(QtCore.QRect(20, 30, 91, 19))
48. font = QtGui.QFont()
49. font.setPointSize(12)
50. self.checkBox.setFont(font)
51. self.checkBox.setObjectName("checkBox")
52. self.checkBox\_2 = QtWidgets.QCheckBox(parent=Dialog)
53. self.checkBox\_2.setGeometry(QtCore.QRect(20, 70, 91, 19))
54. font = QtGui.QFont()
55. font.setPointSize(12)
56. self.checkBox\_2.setFont(font)
57. self.checkBox\_2.setObjectName("checkBox\_2")
58. self.checkBox\_3 = QtWidgets.QCheckBox(parent=Dialog)
59. self.checkBox\_3.setGeometry(QtCore.QRect(20, 110, 91, 19))
60. font = QtGui.QFont()
61. font.setPointSize(12)
62. self.checkBox\_3.setFont(font)
63. self.checkBox\_3.setObjectName("checkBox\_3")
64. self.checkBox\_4 = QtWidgets.QCheckBox(parent=Dialog)
65. self.checkBox\_4.setGeometry(QtCore.QRect(20, 150, 91, 19))
66. font = QtGui.QFont()
67. font.setPointSize(12)
68. self.checkBox\_4.setFont(font)
69. self.checkBox\_4.setObjectName("checkBox\_4")
70. self.checkBox\_5 = QtWidgets.QCheckBox(parent=Dialog)
71. self.checkBox\_5.setGeometry(QtCore.QRect(20, 190, 91, 19))
72. font = QtGui.QFont()
73. font.setPointSize(12)
74. self.checkBox\_5.setFont(font)
75. self.checkBox\_5.setObjectName("checkBox\_5")
76. self.checkBox\_6 = QtWidgets.QCheckBox(parent=Dialog)
77. self.checkBox\_6.setGeometry(QtCore.QRect(110, 70, 91, 19))
78. font = QtGui.QFont()
79. font.setPointSize(12)
80. self.checkBox\_6.setFont(font)
81. self.checkBox\_6.setObjectName("checkBox\_6")
82. self.checkBox\_7 = QtWidgets.QCheckBox(parent=Dialog)
83. self.checkBox\_7.setGeometry(QtCore.QRect(110, 110, 91, 19))
84. font = QtGui.QFont()
85. font.setPointSize(12)
86. self.checkBox\_7.setFont(font)
87. self.checkBox\_7.setObjectName("checkBox\_7")
88. self.checkBox\_8 = QtWidgets.QCheckBox(parent=Dialog)
89. self.checkBox\_8.setGeometry(QtCore.QRect(110, 150, 91, 19))
90. font = QtGui.QFont()
91. font.setPointSize(12)
92. self.checkBox\_8.setFont(font)
93. self.checkBox\_8.setObjectName("checkBox\_8")
94. self.checkBox\_9 = QtWidgets.QCheckBox(parent=Dialog)
95. self.checkBox\_9.setGeometry(QtCore.QRect(110, 190, 91, 19))
96. font = QtGui.QFont()
97. font.setPointSize(12)
98. self.checkBox\_9.setFont(font)
99. self.checkBox\_9.setObjectName("checkBox\_9")
100. self.checkBox\_10 = QtWidgets.QCheckBox(parent=Dialog)
101. self.checkBox\_10.setGeometry(QtCore.QRect(110, 30, 91, 19))
102. font = QtGui.QFont()
103. font.setPointSize(12)
104. self.checkBox\_10.setFont(font)
105. self.checkBox\_10.setObjectName("checkBox\_10")
106. self.retranslateUi(Dialog)
107. QtCore.QMetaObject.connectSlotsByName(Dialog)
108. with open(r"style.qss", "r", encoding='utf-8') as f:
109. Dialog.setStyleSheet(f.read())
110. Dialog.setWindowTitle("抽屉控制")
111. self.init\_state()  # 初始化界面状态
112. def retranslateUi(self, Dialog):
113. \_translate = QtCore.QCoreApplication.translate
114. Dialog.setWindowTitle(\_translate("Dialog", "Dialog"))
115. self.pushButton.setText(\_translate("Dialog", "确认"))
116. self.checkBox.setText(\_translate("Dialog", "抽屉1"))
117. self.checkBox\_2.setText(\_translate("Dialog", "抽屉2"))
118. self.checkBox\_3.setText(\_translate("Dialog", "抽屉3"))
119. self.checkBox\_4.setText(\_translate("Dialog", "抽屉4"))
120. self.checkBox\_5.setText(\_translate("Dialog", "抽屉5"))
121. self.checkBox\_6.setText(\_translate("Dialog", "抽屉7"))
122. self.checkBox\_7.setText(\_translate("Dialog", "抽屉8"))
123. self.checkBox\_8.setText(\_translate("Dialog", "抽屉9"))
124. self.checkBox\_9.setText(\_translate("Dialog", "抽屉10"))
125. self.checkBox\_10.setText(\_translate("Dialog", "抽屉6"))
126. # 连接确认按钮的点击事件到更新状态函数
127. self.pushButton.clicked.connect(self.update\_statu)

## Dialog\_pic.py

1. from PyQt6 import QtCore, QtWidgets
2. from global\_var import temp\_list
3. from matplotlib.backends.backend\_qtagg import FigureCanvasQTAgg as FigureCanvas
4. from matplotlib.figure import Figure
5. import matplotlib.pyplot as plt
6. plt.rcParams['font.sans-serif'] = ['SimHei'] *# 正常显示中文标签*
7. plt.rcParams['axes.unicode\_minus'] = False *# 正常显示负号*
8. class MplCanvas(FigureCanvas):
9. def \_\_init\_\_(self, parent=None, width=5, height=4, dpi=100):
10. fig = Figure(figsize=(width, height), dpi=dpi)
11. self.axes = fig.add\_subplot(111)
12. super(MplCanvas, self).\_\_init\_\_(fig)
14. class Ui\_Dialog\_pic(object):
16. def \_\_init\_\_(self,main\_window=None):
17. self.main\_window = main\_window
18. def setupUi(self, Dialog):
19. Dialog.setObjectName("Dialog")
20. Dialog.resize(600, 400)
21. self.canvas = MplCanvas(Dialog, width=5, height=4, dpi=100)
22. layout = QtWidgets.QVBoxLayout()
23. layout.addWidget(self.canvas)
24. Dialog.setLayout(layout)
25. *# 初始化定时器，每秒更新一次曲线图*
26. self.timer = QtCore.QTimer()
27. self.timer.timeout.connect(self.update\_plot)
28. self.timer.start(500)
29. *# 监听对话框关闭事件，用于清理资源*
30. Dialog.finished.connect(self.on\_close)
31. with open(r"style.qss", "r", encoding='utf-8') as f:
32. Dialog.setStyleSheet(f.read())
33. Dialog.setWindowTitle("温度曲线显示")
35. def on\_close(self):
36. *# 停止定时器并清理绘图组件*
37. if self.timer and self.timer.isActive():
38. self.timer.stop()
39. if self.canvas:
40. self.canvas.deleteLater()
41. def update\_plot(self):
42. self.canvas.axes.clear()
43. *# 应用 matplotlib 内置样式*
44. plt.style.use('ggplot')
45. line\_color = '#1f77b4'  *# 更优雅的蓝色*
47. *# 绘制折线图*
48. self.canvas.axes.plot(temp\_list, label='当前温度', color=line\_color, linewidth=2)
49. *# 添加阴影填充区域*
50. self.canvas.axes.fill\_between(range(len(temp\_list)), temp\_list, color=line\_color, alpha=0.1)
51. *# 设置标题和标签*
52. self.canvas.axes.set\_title('温度变化曲线', fontsize=14, pad=20)
53. self.canvas.axes.set\_xlabel('时间步', fontsize=12)
54. self.canvas.axes.set\_ylabel('温度 (°C)', fontsize=12)
55. *# 设置图例*
56. self.canvas.axes.legend(loc='upper left', fontsize=10)
57. *# 设置网格和背景*
58. self.canvas.axes.grid(True, linestyle='--', alpha=0.6)
59. self.canvas.axes.set\_facecolor('#f9f9f9')
60. self.canvas.draw()

## Dialog\_temp.py

1. from PyQt6 import QtCore, QtGui, QtWidgets
2. from send import \*
3. from global\_var import \*
4. from com\_init import com
5. import time
6. class Ui\_Dialog\_temp(object):
7. def \_\_init\_\_(self, main\_window=None):
8. self.main\_window = main\_window  *# 用于与主窗口交互*
9. def set\_temp(self):
10. *# 获取用户输入的设定温度和温度控制偏差*
11. set\_temp\_str = self.lineEdit.text()
12. set\_temp = float(set\_temp\_str) if set\_temp\_str else None
13. temp\_control\_str = self.lineEdit\_2.text()
14. temp\_control = int(temp\_control\_str) if temp\_control\_str else None
15. *# 更新全局数据字典中的温度设置*
16. data["set\_temp"] = set\_temp if set\_temp is not None else data["set\_temp"]
17. data["temp\_control"] = temp\_control if temp\_control is not None else data["temp\_control"]
18. *# 如果输入有效，生成并发送设置参数帧*
19. if set\_temp is not None and temp\_control is not None:
20. data\_p = set\_parameter\_frame(addr=0x01, set\_parameter\_data=data)
21. send\_frame(data\_p, com, ui=self.main\_window)
22. time.sleep(0.5)
23. *# 如果输入有效，生成并发送设置温度帧*
24. if set\_temp is not None:
25. set\_temperature\_frame\_data = set\_temperature\_frame(addr=0x01, set\_temperature\_data=set\_temp)
26. send\_frame(set\_temperature\_frame\_data, com, ui=self.main\_window)
27. time.sleep(0.5)
28. *# 如果输入有效，生成并发送设置温度控制偏差帧*
29. if temp\_control is not None:
30. set\_temperature\_control\_deviation\_frame\_data = set\_temperature\_control\_deviation\_frame(addr=0x01, set\_temperature\_control\_deviation\_data=temp\_control)
31. send\_frame(set\_temperature\_control\_deviation\_frame\_data, com, ui=self.main\_window)
32. time.sleep(0.5)
33. def cancel(self):
34. *# 关闭对话框*
35. self.Dialog.close()
36. def setupUi(self, Dialog):
37. self.Dialog = Dialog
38. Dialog.setObjectName("Dialog")
39. Dialog.resize(303, 173)
40. self.lineEdit = QtWidgets.QLineEdit(parent=Dialog)
41. self.lineEdit.setGeometry(QtCore.QRect(120, 20, 151, 31))
42. font = QtGui.QFont()
43. font.setPointSize(12)
44. self.lineEdit.setFont(font)
45. self.lineEdit.setObjectName("lineEdit")
46. self.label = QtWidgets.QLabel(parent=Dialog)
47. self.label.setGeometry(QtCore.QRect(20, 10, 91, 51))
48. font = QtGui.QFont()
49. font.setPointSize(12)
50. self.label.setFont(font)
51. self.label.setObjectName("label")
52. self.pushButton = QtWidgets.QPushButton(parent=Dialog)
53. self.pushButton.setGeometry(QtCore.QRect(20, 110, 121, 51))
54. self.pushButton.setObjectName("pushButton")
55. self.pushButton\_2 = QtWidgets.QPushButton(parent=Dialog)
56. self.pushButton\_2.setGeometry(QtCore.QRect(150, 110, 121, 51))
57. self.pushButton\_2.setObjectName("pushButton\_2")
58. self.label\_4 = QtWidgets.QLabel(parent=Dialog)
59. self.label\_4.setGeometry(QtCore.QRect(20, 50, 141, 51))
60. font = QtGui.QFont()
61. font.setPointSize(12)
62. self.label\_4.setFont(font)
63. self.label\_4.setObjectName("label\_4")
64. self.lineEdit\_2 = QtWidgets.QLineEdit(parent=Dialog)
65. self.lineEdit\_2.setGeometry(QtCore.QRect(160, 60, 131, 31))
66. font = QtGui.QFont()
67. font.setPointSize(12)
68. self.lineEdit\_2.setFont(font)
69. self.lineEdit\_2.setObjectName("lineEdit\_2")
70. self.retranslateUi(Dialog)
71. QtCore.QMetaObject.connectSlotsByName(Dialog)
72. with open(r"style.qss", "r", encoding='utf-8') as f:
73. Dialog.setStyleSheet(f.read())
74. Dialog.setWindowTitle("温度设定")
75. def retranslateUi(self, Dialog):
76. \_translate = QtCore.QCoreApplication.translate
77. Dialog.setWindowTitle(\_translate("Dialog", "Dialog"))
78. self.label.setText(\_translate("Dialog", "设定温度:"))
79. self.pushButton.setText(\_translate("Dialog", "确认"))
80. self.pushButton\_2.setText(\_translate("Dialog", "取消"))
81. self.label\_4.setText(\_translate("Dialog", "温度控制偏差:"))
82. *# 连接确认按钮的点击事件到设置温度函数*
83. self.pushButton.clicked.connect(self.set\_temp)
84. *# 连接取消按钮的点击事件到关闭对话框函数*
85. self.pushButton\_2.clicked.connect(self.cancel)

## global\_var.py

1. mach\_statu = 0
2. temp\_list=[]
3. statu=[0,0,0,0,0,0,0,0,0,0,0,0]
4. data = {
5. 'dev\_code': '0000000000',
6. 'dev\_addr': 1,
7. 'time\_gap': 10,
8. 'compressor\_delay': 30,
9. 'set\_temp': 5.5,
10. 'temp\_control': 2
11. }

## api\_server.py

1. from flask import Flask, jsonify
2. from flask\_cors import CORS
3. import global\_var
4. app = Flask(\_\_name\_\_)
5. CORS(app)  *# 启用跨源资源共享*
6. @app.route('/api/temperature', methods=['GET'])
7. def get\_temperature():
8. *# 从全局变量中获取当前温度和其他系统状态信息*
9. current\_temp = global\_var.temp\_list[-1] if global\_var.temp\_list else None
10. return jsonify({
11. "current\_temperature": current\_temp, *# 当前温度*
12. "set\_temperature": global\_var.data.get('set\_temp'), *# 设定温度*
13. "temp\_control\_deviation": global\_var.data.get('temp\_control'), *# 温度控制偏差*
14. "mach\_statu": global\_var.mach\_statu, *# 机器状态*
15. "temperature\_history": global\_var.temp\_list[-10:],  *# 返回最近10次温度记录*
16. "statu": global\_var.statu[:10] *# 抽屉状态*
17. })
18. def run\_flask\_app():
19. *# 启动 Flask 应用，设置 debug 模式为 False，禁用重新加载以避免线程问题*
20. app.run(debug=False, use\_reloader=False)

## main\_ui.py

1. from PyQt6 import QtCore, QtGui, QtWidgets
2. class Ui\_MainWindow(object):
3. def setupUi(self, MainWindow):
4. MainWindow.setObjectName("MainWindow")
5. MainWindow.resize(806, 536)
6. sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Policy.Fixed, QtWidgets.QSizePolicy.Policy.Fixed)
7. sizePolicy.setHorizontalStretch(0)
8. sizePolicy.setVerticalStretch(0)
9. sizePolicy.setHeightForWidth(MainWindow.sizePolicy().hasHeightForWidth())
10. MainWindow.setSizePolicy(sizePolicy)
11. self.centralwidget = QtWidgets.QWidget(parent=MainWindow)
12. self.centralwidget.setObjectName("centralwidget")
13. self.tableWidget = QtWidgets.QTableWidget(parent=self.centralwidget)
14. self.tableWidget.setGeometry(QtCore.QRect(180, 10, 761, 281))
15. sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Policy.Minimum, QtWidgets.QSizePolicy.Policy.Minimum)
16. sizePolicy.setHorizontalStretch(0)
17. sizePolicy.setVerticalStretch(0)
18. sizePolicy.setHeightForWidth(self.tableWidget.sizePolicy().hasHeightForWidth())
19. self.tableWidget.setSizePolicy(sizePolicy)
20. font = QtGui.QFont()
21. font.setStyleStrategy(QtGui.QFont.StyleStrategy.PreferAntialias)
22. self.tableWidget.setFont(font)
23. self.tableWidget.setObjectName("tableWidget")
24. self.tableWidget.setColumnCount(6)
25. self.tableWidget.setRowCount(0)
26. item = QtWidgets.QTableWidgetItem()
27. self.tableWidget.setHorizontalHeaderItem(0, item)
28. item = QtWidgets.QTableWidgetItem()
29. self.tableWidget.setHorizontalHeaderItem(1, item)
30. item = QtWidgets.QTableWidgetItem()
31. self.tableWidget.setHorizontalHeaderItem(2, item)
32. item = QtWidgets.QTableWidgetItem()
33. self.tableWidget.setHorizontalHeaderItem(3, item)
34. item = QtWidgets.QTableWidgetItem()
35. self.tableWidget.setHorizontalHeaderItem(4, item)
36. item = QtWidgets.QTableWidgetItem()
37. self.tableWidget.setHorizontalHeaderItem(5, item)
38. self.tableWidget\_2 = QtWidgets.QTableWidget(parent=self.centralwidget)
39. self.tableWidget\_2.setGeometry(QtCore.QRect(270, 310, 667, 142))
40. sizePolicy = QtWidgets.QSizePolicy(QtWidgets.QSizePolicy.Policy.Fixed, QtWidgets.QSizePolicy.Policy.Fixed)
41. sizePolicy.setHorizontalStretch(0)
42. sizePolicy.setVerticalStretch(0)
43. sizePolicy.setHeightForWidth(self.tableWidget\_2.sizePolicy().hasHeightForWidth())
44. self.tableWidget\_2.setSizePolicy(sizePolicy)
45. self.tableWidget\_2.setObjectName("tableWidget\_2")
46. self.tableWidget\_2.setColumnCount(5)
47. self.tableWidget\_2.setRowCount(3)
48. item = QtWidgets.QTableWidgetItem()
49. self.tableWidget\_2.setVerticalHeaderItem(0, item)
50. item = QtWidgets.QTableWidgetItem()
51. self.tableWidget\_2.setVerticalHeaderItem(1, item)
52. item = QtWidgets.QTableWidgetItem()
53. self.tableWidget\_2.setVerticalHeaderItem(2, item)
54. item = QtWidgets.QTableWidgetItem()
55. self.tableWidget\_2.setHorizontalHeaderItem(0, item)
56. item = QtWidgets.QTableWidgetItem()
57. self.tableWidget\_2.setHorizontalHeaderItem(1, item)
58. item = QtWidgets.QTableWidgetItem()
59. self.tableWidget\_2.setHorizontalHeaderItem(2, item)
60. item = QtWidgets.QTableWidgetItem()
61. self.tableWidget\_2.setHorizontalHeaderItem(3, item)
62. item = QtWidgets.QTableWidgetItem()
63. self.tableWidget\_2.setHorizontalHeaderItem(4, item)
64. item = QtWidgets.QTableWidgetItem()
65. item.setTextAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
66. self.tableWidget\_2.setItem(0, 0, item)
67. item = QtWidgets.QTableWidgetItem()
68. item.setTextAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
69. self.tableWidget\_2.setItem(0, 1, item)
70. item = QtWidgets.QTableWidgetItem()
71. item.setTextAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
72. self.tableWidget\_2.setItem(0, 2, item)
73. item = QtWidgets.QTableWidgetItem()
74. item.setTextAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
75. self.tableWidget\_2.setItem(0, 3, item)
76. item = QtWidgets.QTableWidgetItem()
77. item.setTextAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
78. self.tableWidget\_2.setItem(0, 4, item)
79. item = QtWidgets.QTableWidgetItem()
80. item.setTextAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
81. self.tableWidget\_2.setItem(1, 0, item)
82. item = QtWidgets.QTableWidgetItem()
83. item.setTextAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
84. self.tableWidget\_2.setItem(1, 1, item)
85. item = QtWidgets.QTableWidgetItem()
86. item.setTextAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
87. self.tableWidget\_2.setItem(1, 2, item)
88. item = QtWidgets.QTableWidgetItem()
89. item.setTextAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
90. self.tableWidget\_2.setItem(1, 3, item)
91. item = QtWidgets.QTableWidgetItem()
92. item.setTextAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
93. self.tableWidget\_2.setItem(1, 4, item)
94. item = QtWidgets.QTableWidgetItem()
95. item.setTextAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
96. self.tableWidget\_2.setItem(2, 0, item)
97. item = QtWidgets.QTableWidgetItem()
98. item.setTextAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
99. self.tableWidget\_2.setItem(2, 1, item)
100. item = QtWidgets.QTableWidgetItem()
101. item.setTextAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
102. self.tableWidget\_2.setItem(2, 2, item)
103. item = QtWidgets.QTableWidgetItem()
104. item.setTextAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
105. self.tableWidget\_2.setItem(2, 3, item)
106. item = QtWidgets.QTableWidgetItem()
107. item.setTextAlignment(QtCore.Qt.AlignmentFlag.AlignCenter)
108. self.tableWidget\_2.setItem(2, 4, item)
109. self.groupBox = QtWidgets.QGroupBox(parent=self.centralwidget)
110. self.groupBox.setGeometry(QtCore.QRect(0, 10, 161, 301))
111. self.groupBox.setObjectName("groupBox")
112. self.pushButton = QtWidgets.QPushButton(parent=self.groupBox)
113. self.pushButton.setGeometry(QtCore.QRect(10, 30, 141, 51))
114. font = QtGui.QFont()
115. font.setPointSize(12)
116. self.pushButton.setFont(font)
117. self.pushButton.setObjectName("pushButton")
118. self.pushButton\_2 = QtWidgets.QPushButton(parent=self.groupBox)
119. self.pushButton\_2.setGeometry(QtCore.QRect(10, 100, 141, 51))
120. font = QtGui.QFont()
121. font.setPointSize(12)
122. self.pushButton\_2.setFont(font)
123. self.pushButton\_2.setObjectName("pushButton\_2")
124. self.pushButton\_3 = QtWidgets.QPushButton(parent=self.groupBox)
125. self.pushButton\_3.setGeometry(QtCore.QRect(10, 170, 141, 51))
126. font = QtGui.QFont()
127. font.setPointSize(12)
128. self.pushButton\_3.setFont(font)
129. self.pushButton\_3.setObjectName("pushButton\_3")
130. self.pushButton\_4 = QtWidgets.QPushButton(parent=self.groupBox)
131. self.pushButton\_4.setGeometry(QtCore.QRect(10, 240, 141, 51))
132. font = QtGui.QFont()
133. font.setPointSize(12)
134. self.pushButton\_4.setFont(font)
135. self.pushButton\_4.setObjectName("pushButton\_4")
136. self.groupBox\_2 = QtWidgets.QGroupBox(parent=self.centralwidget)
137. self.groupBox\_2.setGeometry(QtCore.QRect(10, 320, 251, 181))
138. font = QtGui.QFont()
139. font.setPointSize(12)
140. self.groupBox\_2.setFont(font)
141. self.groupBox\_2.setObjectName("groupBox\_2")
142. self.label = QtWidgets.QLabel(parent=self.groupBox\_2)
143. self.label.setGeometry(QtCore.QRect(10, 20, 91, 31))
144. font = QtGui.QFont()
145. font.setPointSize(12)
146. self.label.setFont(font)
147. self.label.setObjectName("label")
148. self.label\_4 = QtWidgets.QLabel(parent=self.groupBox\_2)
149. self.label\_4.setGeometry(QtCore.QRect(120, 20, 41, 31))
150. font = QtGui.QFont()
151. font.setPointSize(12)
152. self.label\_4.setFont(font)
153. self.label\_4.setObjectName("label\_4")
154. self.label\_2 = QtWidgets.QLabel(parent=self.groupBox\_2)
155. self.label\_2.setGeometry(QtCore.QRect(10, 50, 91, 31))
156. font = QtGui.QFont()
157. font.setPointSize(12)
158. self.label\_2.setFont(font)
159. self.label\_2.setObjectName("label\_2")
160. self.label\_5 = QtWidgets.QLabel(parent=self.groupBox\_2)
161. self.label\_5.setGeometry(QtCore.QRect(120, 50, 41, 31))
162. font = QtGui.QFont()
163. font.setPointSize(12)
164. self.label\_5.setFont(font)
165. self.label\_5.setObjectName("label\_5")
166. self.label\_3 = QtWidgets.QLabel(parent=self.groupBox\_2)
167. self.label\_3.setGeometry(QtCore.QRect(10, 110, 111, 31))
168. font = QtGui.QFont()
169. font.setPointSize(12)
170. self.label\_3.setFont(font)
171. self.label\_3.setObjectName("label\_3")
172. self.label\_7 = QtWidgets.QLabel(parent=self.groupBox\_2)
173. self.label\_7.setGeometry(QtCore.QRect(130, 110, 71, 31))
174. font = QtGui.QFont()
175. font.setPointSize(12)
176. self.label\_7.setFont(font)
177. self.label\_7.setObjectName("label\_7")
178. self.label\_6 = QtWidgets.QLabel(parent=self.groupBox\_2)
179. self.label\_6.setGeometry(QtCore.QRect(10, 80, 91, 31))
180. font = QtGui.QFont()
181. font.setPointSize(12)
182. self.label\_6.setFont(font)
183. self.label\_6.setObjectName("label\_6")
184. self.label\_8 = QtWidgets.QLabel(parent=self.groupBox\_2)
185. self.label\_8.setGeometry(QtCore.QRect(120, 80, 41, 31))
186. font = QtGui.QFont()
187. font.setPointSize(12)
188. self.label\_8.setFont(font)
189. self.label\_8.setObjectName("label\_8")
190. self.label\_9 = QtWidgets.QLabel(parent=self.groupBox\_2)
191. self.label\_9.setGeometry(QtCore.QRect(10, 140, 151, 31))
192. font = QtGui.QFont()
193. font.setPointSize(12)
194. self.label\_9.setFont(font)
195. self.label\_9.setText("")
196. self.label\_9.setObjectName("label\_9")
197. self.label\_10 = QtWidgets.QLabel(parent=self.groupBox\_2)
198. self.label\_10.setGeometry(QtCore.QRect(170, 140, 41, 31))
199. font = QtGui.QFont()
200. font.setPointSize(12)
201. self.label\_10.setFont(font)
202. self.label\_10.setObjectName("label\_10")
203. self.label\_11 = QtWidgets.QLabel(parent=self.centralwidget)
204. self.label\_11.setGeometry(QtCore.QRect(270, 470, 71, 31))
205. font = QtGui.QFont()
206. font.setPointSize(12)
207. self.label\_11.setFont(font)
208. self.label\_11.setObjectName("label\_11")
209. self.label\_12 = QtWidgets.QLabel(parent=self.centralwidget)
210. self.label\_12.setGeometry(QtCore.QRect(590, 470, 91, 31))
211. font = QtGui.QFont()
212. font.setPointSize(12)
213. self.label\_12.setFont(font)
214. self.label\_12.setObjectName("label\_12")
215. self.label\_13 = QtWidgets.QLabel(parent=self.centralwidget)
216. self.label\_13.setGeometry(QtCore.QRect(340, 475, 221, 21))
217. font = QtGui.QFont()
218. font.setPointSize(12)
219. self.label\_13.setFont(font)
220. self.label\_13.setText("")
221. self.label\_13.setObjectName("label\_13")
222. self.label\_14 = QtWidgets.QLabel(parent=self.centralwidget)
223. self.label\_14.setGeometry(QtCore.QRect(690, 475, 101, 21))
224. font = QtGui.QFont()
225. font.setPointSize(12)
226. self.label\_14.setFont(font)
227. self.label\_14.setText("")
228. self.label\_14.setObjectName("label\_14")
229. self.groupBox\_2.raise\_()
230. self.tableWidget.raise\_()
231. self.tableWidget\_2.raise\_()
232. self.groupBox.raise\_()
233. self.label\_11.raise\_()
234. self.label\_12.raise\_()
235. self.label\_13.raise\_()
236. self.label\_14.raise\_()
237. MainWindow.setCentralWidget(self.centralwidget)
238. self.menubar = QtWidgets.QMenuBar(parent=MainWindow)
239. self.menubar.setGeometry(QtCore.QRect(0, 0, 956, 26))
240. self.menubar.setObjectName("menubar")
241. MainWindow.setMenuBar(self.menubar)
242. self.statusbar = QtWidgets.QStatusBar(parent=MainWindow)
243. self.statusbar.setObjectName("statusbar")
244. MainWindow.setStatusBar(self.statusbar)
245. self.retranslateUi(MainWindow)
246. QtCore.QMetaObject.connectSlotsByName(MainWindow)

249. def retranslateUi(self, MainWindow):
250. \_translate = QtCore.QCoreApplication.translate
251. MainWindow.setWindowTitle(\_translate("MainWindow", "MainWindow"))
252. item = self.tableWidget.horizontalHeaderItem(0)
253. if item is not None:
254. item.setText(\_translate("MainWindow", "帧长"))
255. item = self.tableWidget.horizontalHeaderItem(1)
256. if item is not None:
257. item.setText(\_translate("MainWindow", "帧号"))
258. item = self.tableWidget.horizontalHeaderItem(2)
259. if item is not None:
260. item.setText(\_translate("MainWindow", "设备地址"))
261. item = self.tableWidget.horizontalHeaderItem(3)
262. if item is not None:
263. item.setText(\_translate("MainWindow", "功能号"))
264. item = self.tableWidget.horizontalHeaderItem(4)
265. if item is not None:
266. item.setText(\_translate("MainWindow", "数据"))
267. item = self.tableWidget.horizontalHeaderItem(5)
268. if item is not None:
269. item.setText(\_translate("MainWindow", "校验码"))
270. item = self.tableWidget\_2.verticalHeaderItem(0)
271. if item is not None:
272. item.setText(\_translate("MainWindow", "状态"))
273. item = self.tableWidget\_2.verticalHeaderItem(2)
274. if item is not None:
275. item.setText(\_translate("MainWindow", "状态"))
276. item = self.tableWidget\_2.horizontalHeaderItem(0)
277. if item is not None:
278. item.setText(\_translate("MainWindow", "抽屉1"))
279. item = self.tableWidget\_2.horizontalHeaderItem(1)
280. if item is not None:
281. item.setText(\_translate("MainWindow", "抽屉2"))
282. item = self.tableWidget\_2.horizontalHeaderItem(2)
283. if item is not None:
284. item.setText(\_translate("MainWindow", "抽屉3"))
285. item = self.tableWidget\_2.horizontalHeaderItem(3)
286. if item is not None:
287. item.setText(\_translate("MainWindow", "抽屉4"))
288. item = self.tableWidget\_2.horizontalHeaderItem(4)
289. if item is not None:
290. item.setText(\_translate("MainWindow", "抽屉5"))
291. \_\_sortingEnabled = self.tableWidget\_2.isSortingEnabled()
292. self.tableWidget\_2.setSortingEnabled(False)
293. item = self.tableWidget\_2.item(0, 0)
294. if item is not None:
295. item.setText(\_translate("MainWindow", "关闭"))
296. item = self.tableWidget\_2.item(0, 1)
297. if item is not None:
298. item.setText(\_translate("MainWindow", "关闭"))
299. item = self.tableWidget\_2.item(0, 2)
300. if item is not None:
301. item.setText(\_translate("MainWindow", "关闭"))
302. item = self.tableWidget\_2.item(0, 3)
303. if item is not None:
304. item.setText(\_translate("MainWindow", "关闭"))
305. item = self.tableWidget\_2.item(0, 4)
306. if item is not None:
307. item.setText(\_translate("MainWindow", "关闭"))
308. item = self.tableWidget\_2.item(1, 0)
309. if item is not None:
310. item.setText(\_translate("MainWindow", "抽屉6"))
311. item = self.tableWidget\_2.item(1, 1)
312. if item is not None:
313. item.setText(\_translate("MainWindow", "抽屉7"))
314. item = self.tableWidget\_2.item(1, 2)
315. if item is not None:
316. item.setText(\_translate("MainWindow", "抽屉8"))
317. item = self.tableWidget\_2.item(1, 3)
318. if item is not None:
319. item.setText(\_translate("MainWindow", "抽屉9"))
320. item = self.tableWidget\_2.item(1, 4)
321. if item is not None:
322. item.setText(\_translate("MainWindow", "抽屉10"))
323. item = self.tableWidget\_2.item(2, 0)
324. if item is not None:
325. item.setText(\_translate("MainWindow", "关闭"))
326. item = self.tableWidget\_2.item(2, 1)
327. if item is not None:
328. item.setText(\_translate("MainWindow", "关闭"))
329. item = self.tableWidget\_2.item(2, 2)
330. if item is not None:
331. item.setText(\_translate("MainWindow", "关闭"))
332. item = self.tableWidget\_2.item(2, 3)
333. if item is not None:
334. item.setText(\_translate("MainWindow", "关闭"))
335. item = self.tableWidget\_2.item(2, 4)
336. if item is not None:
337. item.setText(\_translate("MainWindow", "关闭"))
338. self.tableWidget\_2.setSortingEnabled(\_\_sortingEnabled)
339. # self.groupBox.setTitle(\_translate("MainWindow", "控制区"))
340. self.pushButton.setText(\_translate("MainWindow", "开关柜"))
341. self.pushButton\_2.setText(\_translate("MainWindow", "压缩机控制"))
342. self.pushButton\_3.setText(\_translate("MainWindow", "温度控制"))
343. self.pushButton\_4.setText(\_translate("MainWindow", "绘制温度曲线"))
344. # self.groupBox\_2.setTitle(\_translate("MainWindow", "系统参数"))
345. self.label.setText(\_translate("MainWindow", "当前温度:"))
346. self.label\_4.setText(\_translate("MainWindow", "30"))
347. self.label\_2.setText(\_translate("MainWindow", "设定温度:"))
348. self.label\_5.setText(\_translate("MainWindow", "30"))
349. self.label\_3.setText(\_translate("MainWindow", "压缩机状态："))
350. self.label\_7.setText(\_translate("MainWindow", "关闭"))
351. self.label\_6.setText(\_translate("MainWindow", "温度偏差:"))
352. self.label\_8.setText(\_translate("MainWindow", "2"))
353. self.label\_9.setText(\_translate("MainWindow", "压缩机启动延时："))
354. self.label\_10.setText(\_translate("MainWindow", "1"))
355. self.label\_11.setText(\_translate("MainWindow", "设备ID:"))
356. self.label\_12.setText(\_translate("MainWindow", "设备地址:"))

## style.qss

1. */\* ===== 全局样式 ===== \*/*
2. QWidget {
3. font-family: 'Segoe UI', 'Microsoft YaHei', sans-serif;
4. font-size: 10pt;
5. background-color: #f5f7fa;
6. color: #333333;
7. }
8. */\* ===== 主窗口 ===== \*/*
9. QMainWindow {
10. background-color: #f0f2f5;
11. }
12. */\* ===== 分组框 ===== \*/*
13. QGroupBox {
14. border: 1px solid #d1d9e0;
15. border-radius: 6px;
16. margin-top: 1.5ex;
17. padding-top: 12px;
18. font-weight: bold;
19. }
20. QGroupBox::title {
21. subcontrol-origin: margin;
22. subcontrol-position: top center;
23. padding: 0 8px;
24. background-color: transparent;
25. color: #3a6ea5;
26. }
27. */\* ===== 按钮 ===== \*/*
28. QPushButton {
29. background-color: #3a6ea5;
30. color: white;
31. border: none;
32. border-radius: 4px;
33. padding: 8px 16px;
34. font-weight: 500;
35. min-width: 80px;
36. }
37. QPushButton:hover {
38. background-color: #2d5590;
39. }
40. QPushButton:pressed {
41. background-color: #1e3c6c;
42. }
43. */\* ===== 表格 ===== \*/*
44. QTableWidget {
45. gridline-color: #d1d9e0;
46. background-color: white;
47. border: 1px solid #d1d9e0;
48. border-radius: 4px;
49. selection-background-color: #d6e4f7;
50. selection-color: #333333;
51. }
52. QHeaderView::section {
53. background-color: #e8ecef;
54. color: #3a6ea5;
55. padding: 6px;
56. border: none;
57. font-weight: bold;
58. }
59. QTableCornerButton::section {
60. background-color: #e8ecef;
61. border: none;
62. }
63. */\* ===== 标签 ===== \*/*
64. QLabel {
65. color: #444444;
66. padding: 2px 0;
67. }
68. */\* ===== 输入框 ===== \*/*
69. QLineEdit {
70. background-color: white;
71. border: 1px solid #d1d9e0;
72. border-radius: 4px;
73. padding: 6px 8px;
74. selection-background-color: #d6e4f7;
75. }
76. QLineEdit:focus {
77. border: 1px solid #3a6ea5;
78. }
79. */\* ===== 复选框 & 单选按钮 ===== \*/*
80. QCheckBox, QRadioButton {
81. spacing: 8px;
82. color: #444444;
83. }
84. QCheckBox::indicator, QRadioButton::indicator {
85. width: 16px;
86. height: 16px;
87. border: 1px solid #d1d9e0;
88. border-radius: 3px;
89. }
90. QCheckBox::indicator {
91. background-color: white;
92. }
93. QRadioButton::indicator {
94. border-radius: 8px;
95. }
96. QCheckBox::indicator:checked, QRadioButton::indicator:checked {
97. background-color: #3a6ea5;
98. border: 1px solid #3a6ea5;
99. }
100. */\* ===== 对话框 ===== \*/*
101. QDialog {
102. background-color: #f5f7fa;
103. border: 1px solid #d1d9e0;
104. border-radius: 6px;
105. }
106. */\* ===== 状态栏 ===== \*/*
107. QStatusBar {
108. background-color: #e8ecef;
109. color: #666666;
110. border-top: 1px solid #d1d9e0;
111. }