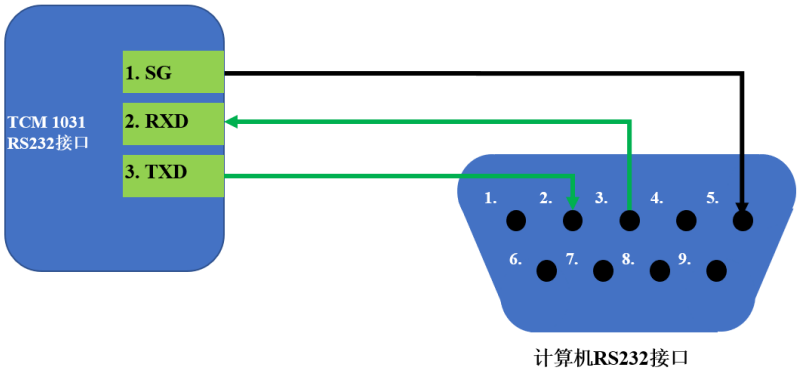
5、温控系统的软件设计

5.1 通讯协议 RS232

温控模块 TCM 1031 的计算机通讯串口 PC RS232，为三针接口。该接口采用串口 RS 232 通讯协议，可以连接到计算机的串口。

连接示意图如图 5-1 所示，右侧的表格为计算机接口的介绍，实际应用中只需要用到针号 2 接收数据信号、针号 3 发送数据信号以及针号 5 地信号，将其分别与温控模块的针脚 3、针脚 2 以及针号 1 连接即可。



针号	功能 说明	缩写
1	载波检测	DCD
2	接收数据	RXD
3	发送数据	TXD
4	数据终端准备	DTR
5	信号地	SG
6	数据准备好	DSR
7	请求发送	RTS
8	清除发送	CTS
9	振铃指示	RT

图 5－1 RS232 引脚接线示意图

Figure 5-1 RS232 Pin Wiring Diagram

由于现在的计算机端口以 USB 接口居多，所以一般使用 RS232 九针接口转 USB 接口转接线进行连接。本文在编写程序时，发送指令的格式参照了业贤科技公司资源文档中的 RS 232 串口控制协议[19]，该协议规定了通讯基本格式。通讯命令一般有设定命令、查询命令以及保存命令。比如设定命令，其语法如下：

$$MODULENAME:PARAMNAME=PARAMVALUE\backslash r$$

整个设定命令由四部分组成：模块参数名、等号、设置值、结束符。含义为：模块：参数=值\r，例如指令：TC1:TCADJUSTTEMP=25\r，含义就是把子模块 TC1 的参数 TCADJUSTTEMP（该参数的含义是温度控制器的调节温度）设置为 25℃。并非每个参数都可以被设置，某些参数只能被查询，某些参数不对外公开。下位机在接收到设定命令后，会进行相关操作，然后返回执行结果信息。返回信息的格式为：CMD:REPLY=ERRORCODE\r，其中 ERRORCODE

代表错误信息码。

关于通讯语法的说明，“MODULENAME”是指子模块名称，该名称的具体定义由下位机确定；“PARAMNAME”指参数名称，同样由下位机确定，用冒号分隔开子模块名称和参数名称。符号“=”、“？”、“!”分别表明是设定命令、查询命令以及保存命令。“\r”则应当注意，其表明 1 个命令结束的回车符，是一个字节的单字符。ASCII 码为 0x0D。“ERRORCODE”是错误信息码，用于表明下位机在执行命令的过程中产生的错误信息，可以帮助用户了解使用的通讯命令的执行情况。

通讯语协议规定，所有的通讯命令都是大小写敏感的。每个通讯命令的结尾都含有符号“\r”来标明该命令完整结束，否则不被识别。此外，所有命令均以 ASCII 码方式发送，一条通讯命令的内部、前面不能有空格，每两条发向下位机的通讯命令的发送时间间隔必须大于 50ms。

5.2 界面设计

本文首先使用 Qt Designer 设计初始界面，设计了四个界面，分别是指令收发界面、单次温度设置界面、连续温度设置界面以及参数设置调整界面。

指令收发界面设计示意图如图 5-2 所示：

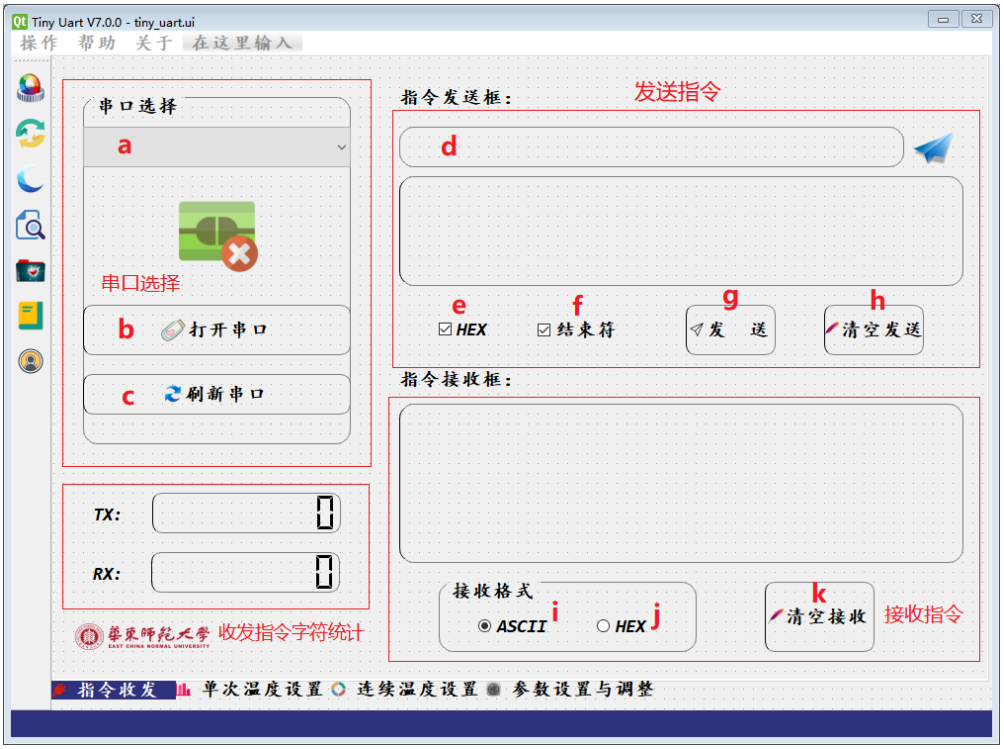


图 5－2 指令收发界面设计示意图

Figure 5-2 Schematic Design of The Command Sending and Receiving Interface

本文在左侧工具栏和底部任务栏分别设计了部分功能，左侧工具栏设计了七个按钮功能，由上至下应当分别实现串口参数配置、串口刷新、白天/夜晚模式切换、指令查询、保存文件路径打开、使用说明书查看以及作者相关介绍。底部任务栏则实现实时更新当前串口连接情况以及实时时间。

指令收发界面中，包含串口的选择、打开、刷新等操作，以及指令发送和接收部分。具体的按键功能表如表 5-1 所示：

表 5－1　按键功能表 1

Table 5-1 Keypad Menu 1

	名称	功能
a.	串口选择框	用于选择已经检测出的串口
b.	打开串口	连接选中的串口
c.	刷新串口	在串口选择框中更新已经检测出的串口
d.	指令发送框	用于给用户输入指令
e.	HEX	用于在指令发送框下显示指令在十六进制下的格式
f.	结束符	用于在用户输入的指令后面添加结束符“\r”
g.	发送	向下位机发送用户输入的指令
h.	清空发送	将用户输入的指令清空
i.	ASCII	将接收到的指令以ASCII码形式显示在指令接收框中
j.	HEX	将接收到的指令以十六进制形式显示在指令接收框中
k.	清空发送	将下位机返回的指令清空

应用默认添加了结束符“\r”，用户不必再单独输入。用户如果选中了十六进制按钮 HEX，则矩形框中会同步显示待发送指令在十六进制形式下的ASCII码值，此时指令最后一个字符会显示 0x0D。

单次温度设置界面可用来设定单次温度和实时温度查询。设定温度在数字框中输入数字并点击设定按钮即可设置，软件默认可保存上次的温度设定值。温度查询是每一秒向下位机查询，并在程序中将返回的指令进行数据提取，显示到下面的蓝色数字框中。同时用户可在此界面进行绘图操作，只需要点击

Draw 按钮即可。如果用户要保存数据，以便后续进行处理，也可以点击 Save 按钮，可将数据保存成文本txt文件，保存路径为“D:\Tiny Uart Data”，保存格式一律为“年-月-日_时-分-秒.txt”，从而避免重名产生覆盖的问题。本文也考虑到温控时间较长的情况，会使得绘出来的数据显得拥挤，所以在主图上方设计了局部放大图，可在里面进行详细查看，实时显示鼠标放置点的横纵坐标数据。右边的差值-时间图，用户可以看到预设温度与实际温度的差值，便于直观显示温控的波动范围及稳定性能。

单次温度设置界面设计示意图如图 5-3所示：

图 5－3 单次温度设置界面设计示意图

Figure 5-3 Single Temperature Setting Interface Design Diagram

其中需要设计的绘图将在工程中的 pyqtgraph.py 文件中生成，并在主文件中导入使用。

该页面的按键功能表如表 5-2 所示：

表 5－2　按键功能表 2

Table 5-2 Keypad Menu 2

	名称	功能
a.	Start	开启温度查询功能
b.	Stop	停止温度查询功能
c.	温度设定	向温控模块发送设定温度值的命令
d.	温度选择	调整或者输入要设定的温度值
e.	Draw	开启画图功能，三个绘图部件同步工作
f.	Save	保存设定与实际温度值的数据在文本文件中
g.	Clear	删除温度数据以及绘图图形

多次温度设置界面可以进行最多四个温度值设定的操作，用户可选择停留时间，其余操作和单次温度设置一致。

界面示意图如图 5-4 所示：

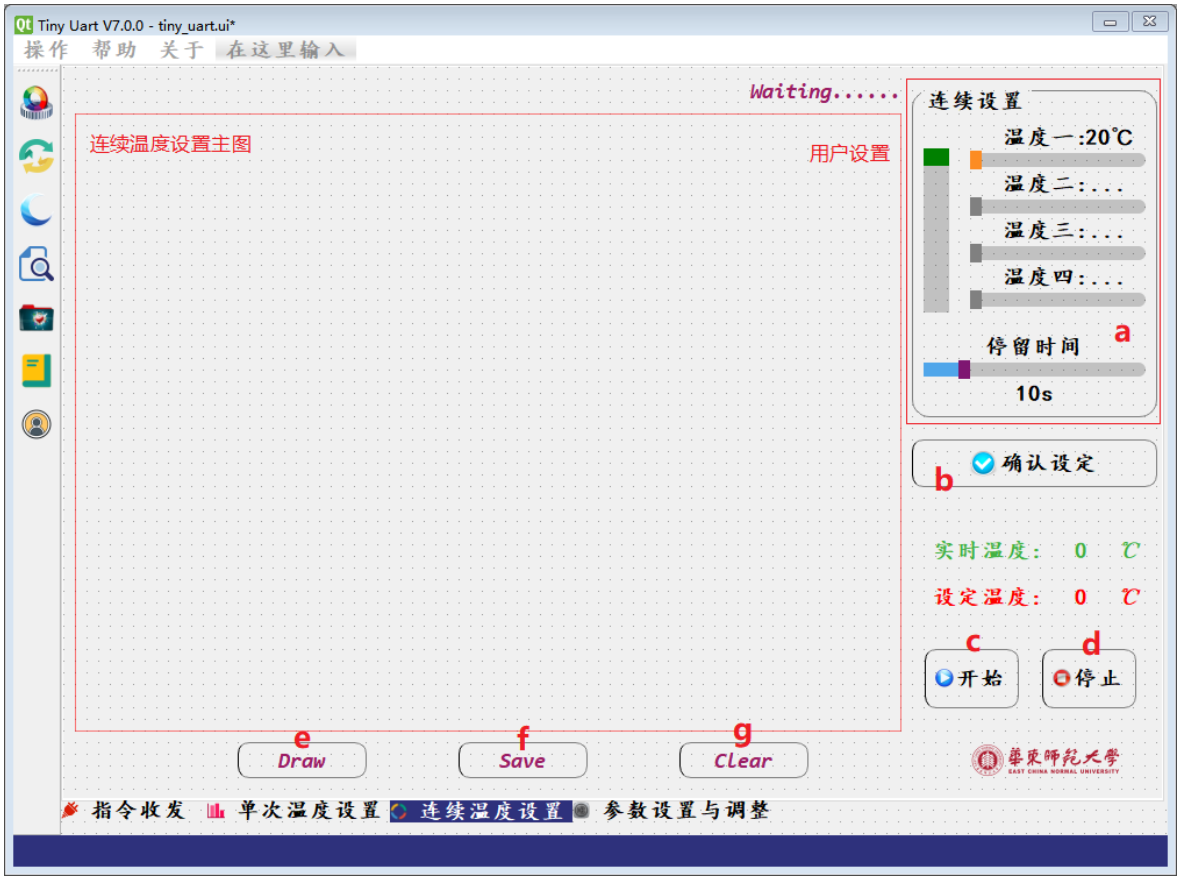


图 5－4　多次温度设置界面设计示意图

Figure 5-4 Multi-temperature Setting Interface Design Diagram

该页面的按键功能表如表 5-3 所示：

表 5－3　按键功能表 3

Table 5-3 Keypad Menu 3

	名称	功能
a.	连续设置	进行多次温度值的设定以及停留时间的设定
b.	确认设定	进行用户设定的温度值、停留时间的设定
c.	启动	开始执行操作，设定以及查询温度
d.	停止	停止所有操作
e.	Draw	开启实时画图功能
f.	Save	保存设定与实际温度值的数据在文本文件中
g.	Clear	删除温度数据以及绘图图形

最后的参数设置与调整界面，一般情况下不需要操作，它用于串口参数的调整，默认情况下软件已经在初始化时调整完毕。

5.3 串口通信实现

串口通信的实现，主要是串口的打开，读取和写入数据三个方面。

串口的打开代码如下：

```
def slot_pushButton_Open(self,parameter):
    if self.state == 0:
         self.ser.port = parameter['comboBox_uart']
        self.ser.baudrate = int(parameter['comboBox_baud'])
        self.ser.bytesize = int(parameter['comboBox_data'])
        self.ser.stopbits = int(parameter['comboBox_stop'])
        self.ser.parity = parameter['comboBox_check']
        try:
            self.ser.open()
            self.state = 1
            self.signal_pushButton_Open_flag.emit(self.state)
        except:
            self.signal_pushButton_Open_flag.emit(0)
         if self.ser.isOpen():
              self.timer.start(100)
    else:
        self.state = 0
```

```
            self.timer.stop()
                try:
                 self.ser.close()
                 self.signal_pushButton_Open_flag.emit(2)
            except:
                 pass
```

该函数首先根据主程序传入的串口参数进行串口配置，进而尝试打开串口，根据不同的情况进行分类：如果串口打开，则继续启动接收函数；反之，则传达打开失败信号给主程序。

串口接收数据函数如下：

```
def receive_data(self):
    length = 0
    if self.ser.isOpen():
        try:
            num = self.ser.inWaiting()
        except:
            self.timer.stop()
            self.ser.close()
            return
        if num > 0:
            data = self.ser.read(num)
            length = len(data)
        n = self.ser.inWaiting()
        length = length + n
        if length > 0 and n == 0:
           try:
               self.signal_readData.emit(data)
           except:
               print('读取不完全')
```

该函数每 100ms 执行一次，保证数据接收的覆盖面。主要用到了 pyserial 库的 read() 函数，为了防止读取不完全，还在接收数据完成之后进行简单的字符长度对比，确保接收字符的完整性。

串口的指令写入函数如下：

```
def slot_sendData(self,send_data):
    if self.state != 1:
       return
    send_data = send_data.encode('utf-8')
    self.ser.write(send_data)
```

该函数主要使用了 pyserial 库的 write() 函数，需要注意，写入指令之前需

要将字符串进行 “utf-8”字节编码，确保写入指令的格式正确。

串口通信线程和主线程的协作关系如图 5-5 所示:

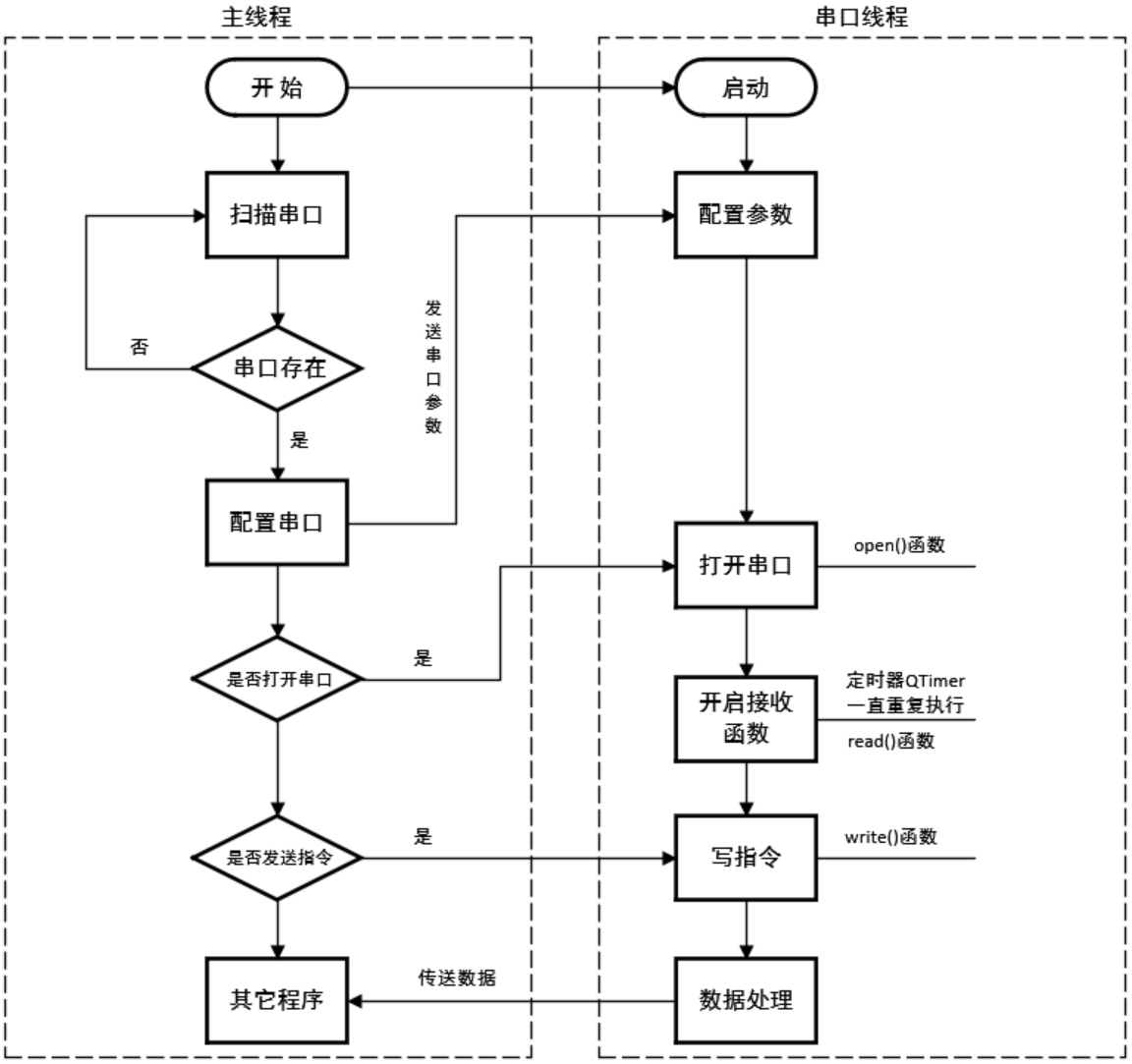


图 5－5 串口线程和主线程协作图

Figure 5-5 Collaboration Diagram between Serial and Main Thread

5.4 数据绘图实现

绘图部件在线程子文件中生成，并进行界面美化和样式设置，主文件只需要装载在预留的框图中即可。这样做以便达到功能分文件管理的效果。两个绘图的子线程工作方式相同，本文主要介绍单次温度设置。

绘图工作主要使用了 pyqtgraph 生成的 plotwidget 控件内嵌的 plot() 功能，绘图函数如下：

```
def pyqtgraph_draw(self,list_y1,list_y2):
        self.subtraction = list(map(lambda x: x[0]-x[1], zip(list_y1, list_y2)))
        self.psignal_pyqtgraph_scroll.emit(self.subtraction)
        try:
            self.p1.plot(list_y1, pen=pg.mkPen({'color': QColor(14, 176, 201), 'width': 1}))
            self.p1.plot(list_y2, pen='r')
            self.plotwidget.plot(list_y1, pen=pg.mkPen({'color':QColor(14, 176, 201), 'width': 2}))
            self.plotwidget.plot(list_y2, pen=pg.mkPen(color='r', width=1))
        except:
            pass
```

根据传入函数已整理的数据列表，可直接进行绘图。因为收集数据的速度影响，绘图更新的频率为每秒更新一次。

绘图线程和主线程的协作关系如图 5-6 所示:

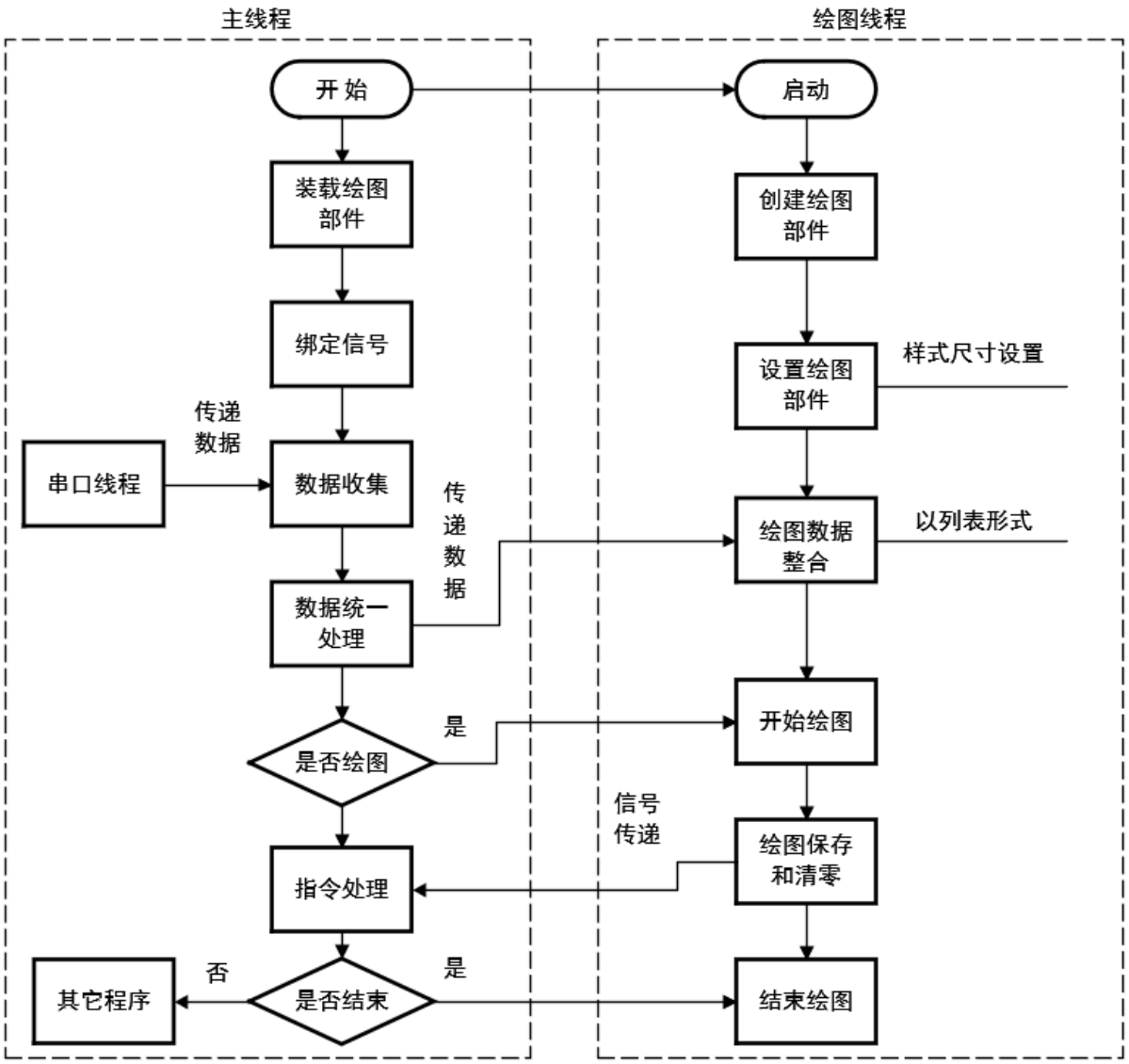


图 5－6 绘图线程和主线程协作图

Figure 5-6 Collaboration Diagram between The Drawing Thread and The Main Thread

5.5 程序文件总结

所有文件放在工程文件夹 Temperature-Uart-7.0 中， src 文件夹用于放置编写的程序，主要是界面文件、配置文件以及功能代码文件，data 文件夹用于存储设定的温度数据，dist 文件夹用于放置生成的可执行文件，doc 用于存放文档文件，ico 用于存放界面美化的图标，如图 5-7 所示：

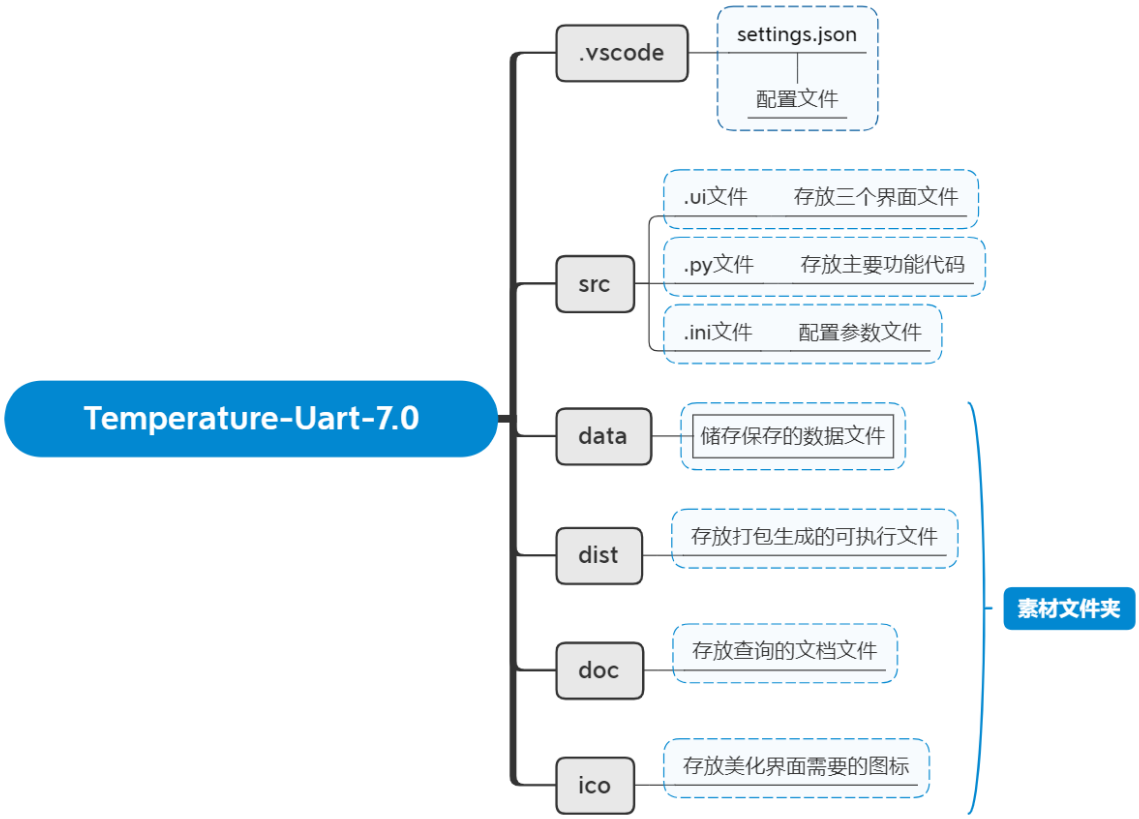


图 5－7 文件结构图

Figure 5-7 Collaboration Diagram between The Drawing Thread and The Main Thread

程序功能的实现主要是 src 文件夹中文件来实现，src 文件夹中有三种类型的文件，主文件是 tiny_uart_main.py ，用于整合其它文件，完成界面功能的绑定。线程文件分为 serial_thread.py，用于创建串口线程进行通讯；pyqtgraph_thread.py 和 pyqtgraph_cycle_thread.py 用于创建绘图线程，分别实现

单次温度设置功能和多次温度设置功能。配置文件 config.ini 则写入了串口参数，用于在程序运行时直接进行参数设置。三个界面文件则是对软件的界面进行设计，其余的 py 文件是对这些界面文件进行调整与设置。

src 文件夹中各个文件的协同关系如图 5-8 所示：

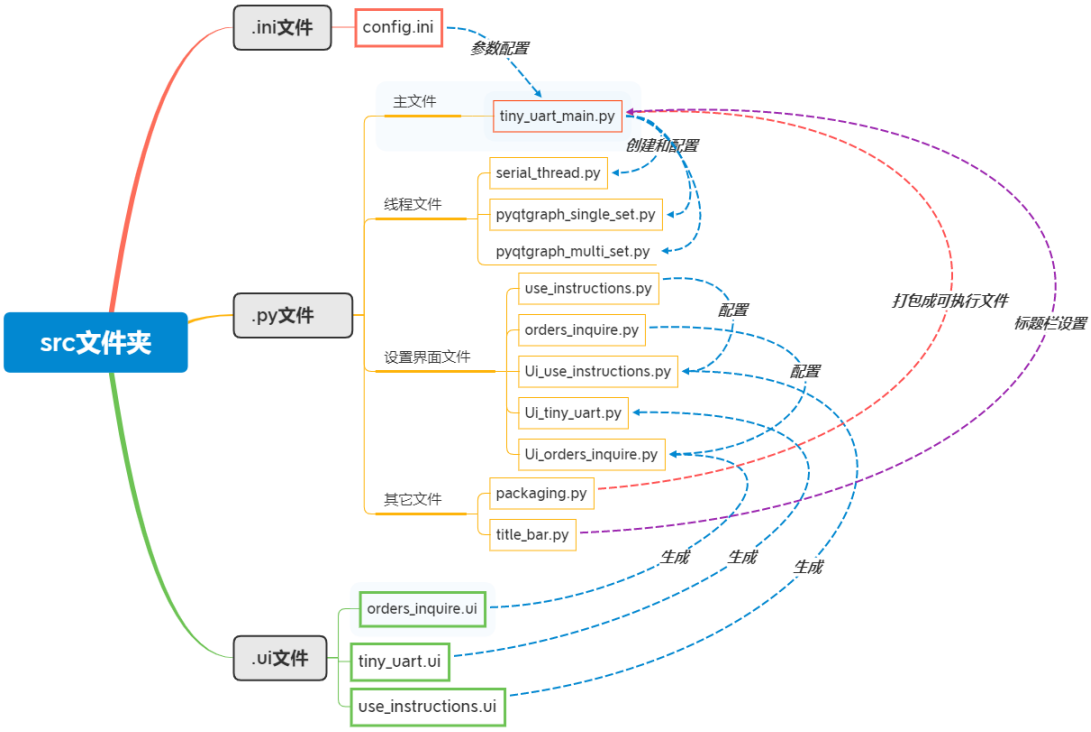


图 5－8 src 文件协同图

Figure 5-8 src File Synergy Map

5.6 程序打包

写完的程序需要提供用户使用，且考虑到用户并没有安装 python 环境，这时就需要将程序打包成可执行文件。通过 pip 指令安装 Pyinstaller 第三方库，再使用指令运行代码即可。

本文打包 tiny_uart_main.py 主程序并且提前给程序指定应用图标为 feather.ico。

程序打包代码如图 5-9 所示：

tiny_uart_main.py | packaging.py | pyqtgraph_cycle_thread.py | pyqtgraph_thread.py

src > packaging.py

```
1  import os
2
3  os.system(command="pyinstaller --clean -Fw .\\src\\tiny_uart_main.py -i .\\ico\\feather.ico")
```

图 5－9　程序打包代码

Figure 5-9　Program Packaging Code

运行该打包代码，得到的结果如图 5-10 所示：

```
115653 INFO: Building PKG (CArchive) tiny_uart_main.pkg completed successfully.
115827 INFO: Bootloader d:\python\install\lib\site-packages\PyInstaller\bootloader\Windows-64bit\runw.exe
115827 INFO: checking EXE
115827 INFO: Building EXE because EXE-00.toc is non existent
115827 INFO: Building EXE from EXE-00.toc
115829 INFO: Copying bootloader EXE to f:\My_Projects\Temperature-Uart-5.0\dist\tiny_uart_main.exe
115837 INFO: Copying icon to EXE
115837 INFO: Copying icons from ['ico\\pikaqiu.ico']
115839 INFO: Writing RT_GROUP_ICON 0 resource with 20 bytes
115839 INFO: Writing RT_ICON 1 resource with 67624 bytes
115842 INFO: Copying 0 resources to EXE
115842 INFO: Emedding manifest in EXE
115842 INFO: Updating manifest in f:\My_Projects\Temperature-Uart-5.0\dist\tiny_uart_main.exe
115843 INFO: Updating resource type 24 name 1 language 0
115846 INFO: Appending PKG archive to EXE
146627 INFO: Building EXE from EXE-00.toc completed successfully.

[Done] exited with code=0 in 147.913 seconds
```

图 5－10　程序打包代码运行结果

Figure 5-10　Results of Packaged Code Runs

可以看到其在最后一行已提示，应用生成成功。程序打包生成的应用就放在 dist 文件夹中，如图 5-11 所示：

图 5－11　生成的应用

Figure 5-11　Generated applications

至此，本文已经完成了对软件的设计，可以在 PC 端下载可执行文件对温控系统进行操作。

6、温控系统的模拟仿真

6.1 仿真设置

本文为了验证软件功能是否实现，选择了 Arduino Uno 板子作为温控装置

进行仿真。因为其端口就是普通的串口，和温控模块的端口一致[20]。

Arduino Uno 实物图如图 6-1 所示：

图 6－1　Arduino Uno

Figure 6-1　Arduino Uno

同时预先向 Arduino Uno 板子烧录了运行代码，其实现了将从 PC 端输出的指令再返回给 PC 端，这样就可以验证指令格式的正确性。

代码如图 6-2 烧录代码所示：

demo_uart

```
char byte_recv = 0;
String data_recv = "";
int flag_recv_str = 0;
int ledpin=13;
int flag = 0;
void setup() {
  // put your setup code here, to run once:
  Serial.begin(57600);
  pinMode(ledpin,OUTPUT);
}

void loop() {
  while(Serial.available()>0){
    byte_recv = Serial.read();
    flag_recv_str = 1;
    data_recv += byte_recv;
    delay(10);
    }
   if(1 == flag_recv_str){
    flag_recv_str = 0;
    //Serial.println("recv_str:");
    Serial.print(data_recv);
    data_recv = "";
    }
}
```

图 6－2　Arduino Uno 烧录代码

Figure 6-2 Arduino Uno Burn-in Code

上述代码 Serial.begin(57600) 将Arduino Uno 的串口波特率设置为 57600，和温控模块一致。

6.2 串口通信仿真

进入程序界面，在串口选择框的下拉列表中选择连接端口 Arduino Uno，进行连接，连接成功后，可以发现连接图标已经更新，并且在任务栏中可以显示出当前连接串口。

在指令输入框中输入正确格式的指令，并向下位机发送将温度设定为25℃的命令：TC1:TCADJUSTTEMP=25，可以看到指令被完全返回。

效果如图 6-3 串口通信效果图所示：

图 6－3　串口通信效果图

Figure 6-3 Serial Communication Effect

需要注意，用户发送的指令在点击结束符选择框后，会默认在命令后自动加上字符“\r”。如果同时选择十六进制框，则用户可以在其下面的指令显示框中清楚看到结束符对应的十六进制数值：0x0D。指令接收框中可以显示出从下位机接收到的指令，用户同样可以选择两种格式来查看，在实际应用中，便于用户查看返回的指令。

此界面主要是给用户连接温控模块进行调试使用，由于软件其它界面内置了自动温度查询、温度设定、实时绘图等功能，用户不需要再在此界面进行指令收

发。考虑到用户可能对于温控模块可能进行其它的指令操作，本文仍旧对此基本功能进行了保留，便于用户进行其它如调节 PID 参数等其它操作。

6.3 单次温度设置仿真

单次温度设置界面主要分为三个区域，分别是区域①：局部放大图和残差图；区域②：单次温度设置主图；区域③：用户设置区域。

用户在连接到串口之后，点击 Satrt 按钮，可以实时查询温度，更新频率为 1s/次，温度设定按钮则是用来将用户选择的温度值作为设定命令进行发送。如果用户没有进行温度选择，则默认设定为 25℃。两个数字显示框会一直更新数据。温度设定按钮的左下方放置了记录设定的温度的数字框，用以记录上次设定的温度值。

单次温度设置仿真图如 6-4 所示：

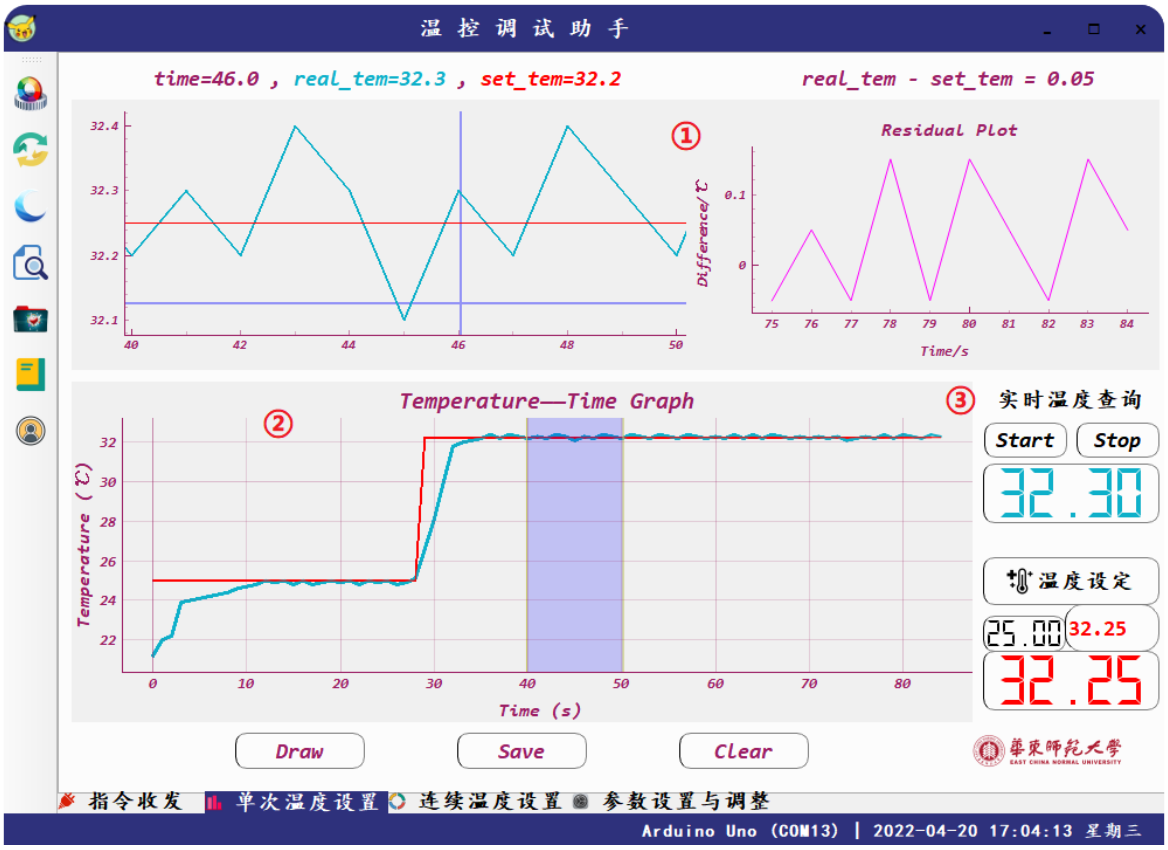


图 6－4　单次温度设置仿真效果图

Figure 6-4 Simulation of A Single Temperature Setting

与此同时，软件会自动记录用户操作之后温控模块传来的温度数值，用户在

点击 Draw 按钮之后，会进行实时绘图，更新频率同样为 1s/次。区域②的主图为从用户开始操作之后的所有数据展示，主图中存在一个可以调节的矩形区域，用户可以自由调节其范围，在不破坏主图完整性的前提下来进行局部区域的查看。当设置温度次数较多，操作时间较长时，数据较多，主图越发不清晰，矩形区域的作用效果则越明显。同样地，用户也可以进行鼠标滚轮进行局部放大，两种查看功能软件都具备。

区域①的右边则是差值图，用以实时显示设定温度与实际温度的差值。由于要兼顾显示的可观性以及实用性，该图采用了实时更新横坐标的模式，保留十秒以内的差值来进行显示。

用户可以保存温度实际和设定数据，其采用列表形式存储在程序中，当用户点击 Save 按钮之后，程序则将数据以列的形式写入文本文件中，第一列为用户设定温度，第二列为实际的温度值。

保存效果以及格式如图 6-5 所示：

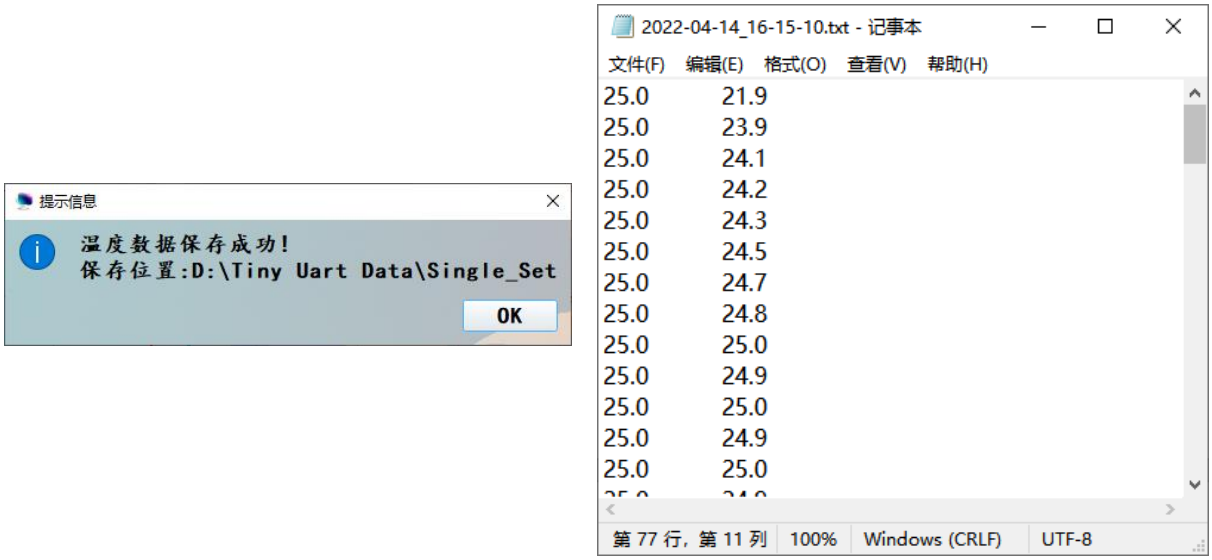


图 6－5　数据保存效果图

Figure 6-5 Data Saving Effect

为了方便查看数据保存文件，用户可以在左侧工具栏点击数据打开路径按钮打开数据保存位置。打开后，可以看到两个文件夹，Single_Set 用于存放单次温度设置的数据，Multi_Sets 用于存放连续温度设置的数据。

点击数据路径按钮效果图如图 6-6 所示：

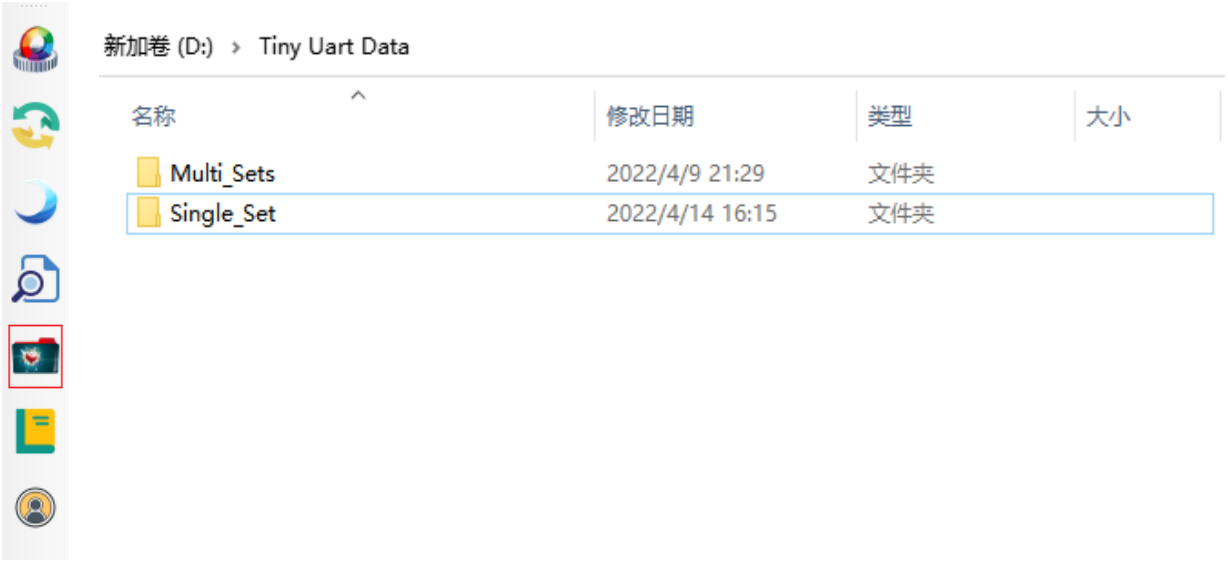


图 6－6　打开数据路径效果图

Figure 6-6 Open Data Path Effect

6.4 多次温度设置仿真

连续温度设置界面，主要分为区域①：主图，区域②：用户设置区域，区域③：用户操作区域。

主图和单次温度设置的主图类似，可以实时更新数据，频率为 1s/次，但是不再设置矩形区域，用户可单独使用鼠标滚轮来放大需要细看的区域，同时鼠标所在处将在主图右上方显示横纵坐标数值，便于用户查看。

用户设置区域，软件最多可以设置四个温度值，用户可以通过拖动左边的纵向滑动条来选择设定温度的数量，默认设定一个。当用户温度设定完成之后，可以通过调整下面的停留时间来调整每个设定温度的停留时间。温度设置范围为 10-200℃，停留时长设置范围为 10-200s。后续可考虑增加设定步长和范围进行温度扫描功能，增强连续多次温度设置的实用性。

本文的仿真选择把设置开关全部打开，即设置四个温度，分别是 68℃，26℃，84℃，48℃，停留时间设置成 65s。当用户设置完毕，需要点击确认设定按钮，将设定信息发送给内部程序待发送指令处，同时软件也会弹出提示框提示设定信息。

温度设定信息提示框如图 6-7 所示：

图 6—7　信息提示框

Figure 6-7 Message Box

确认设定信息之后，点击启动和绘图按钮，即可实现温控设置温度并实时更新数据，连续温度设置的效果图如图 6-8 所示：

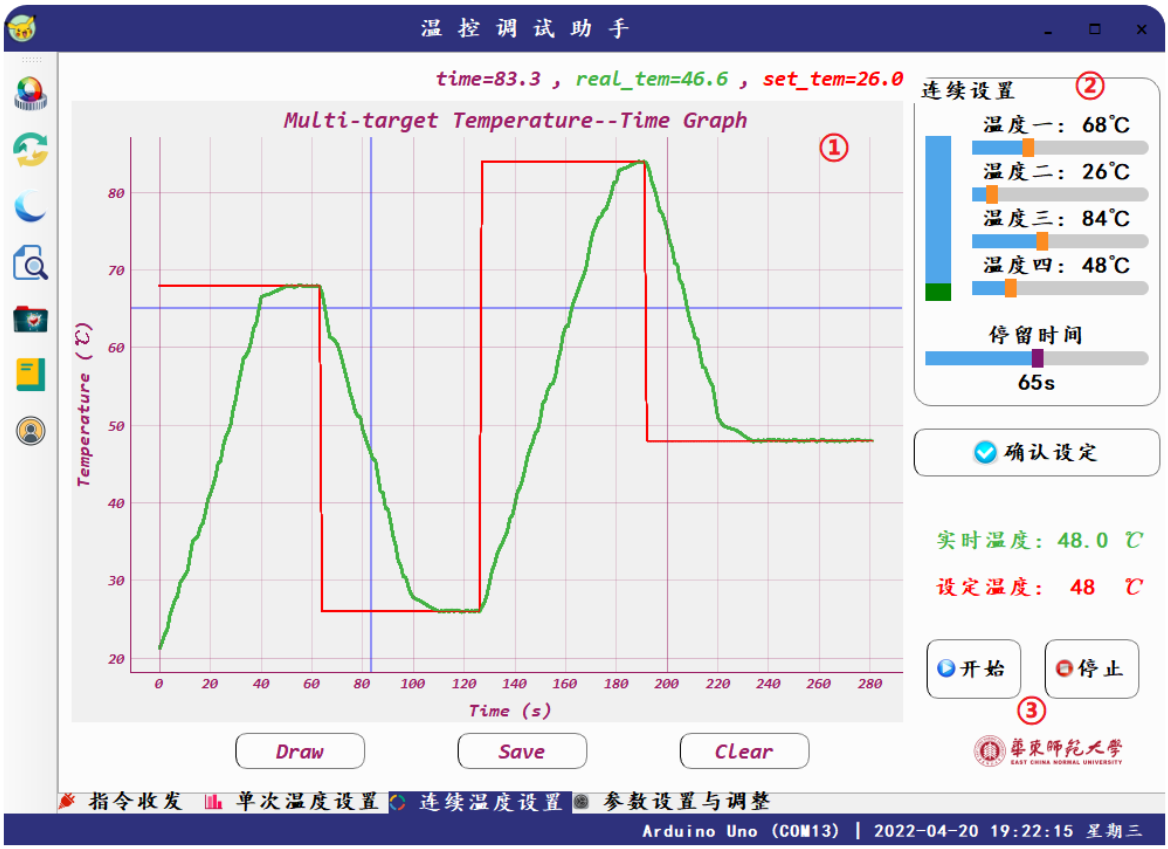


图 6—8　连续温度设置效果图

Figure 6-8 Continuous Temperature Setting Effect

同样，用户也可以进行数据保存，格式和单次温度设置保存的数据格式相同，但数据文件将保存在 Multi_Sets 文件夹中。