1、实验要求

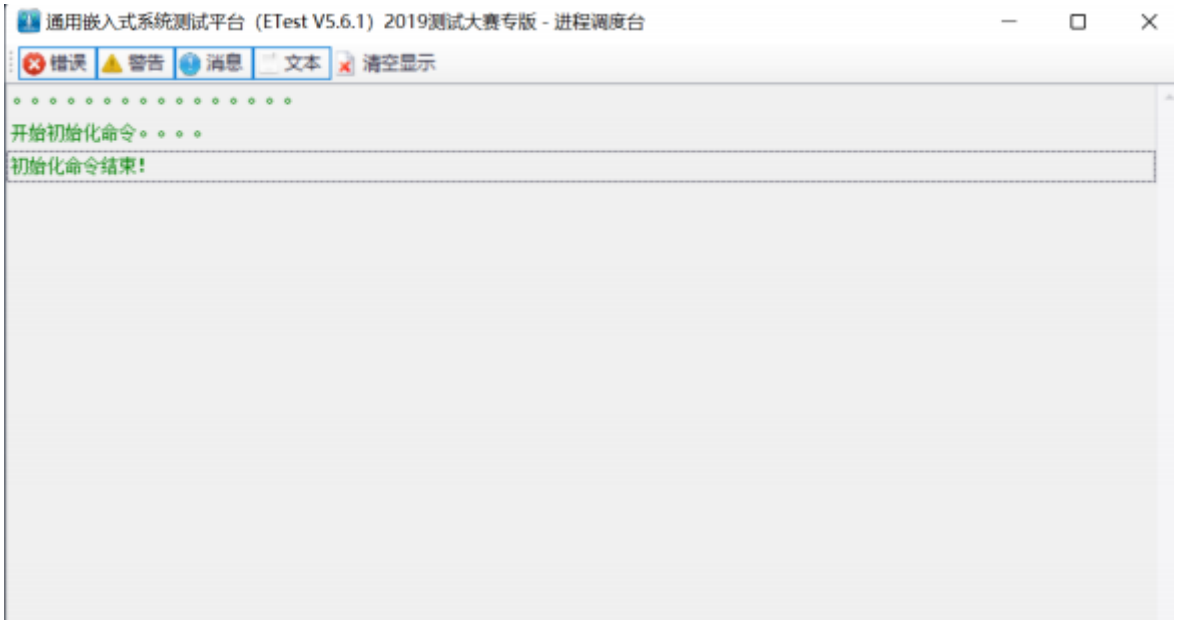
熟悉嵌入式测试的工具的使用

学习测试脚本代码的编写

2、实验过程

2.1 安装相关软件

如图所示



2.2 环境搭建

（1）根据需求文档中的系统模型图，使用vspd将对应串口相连：

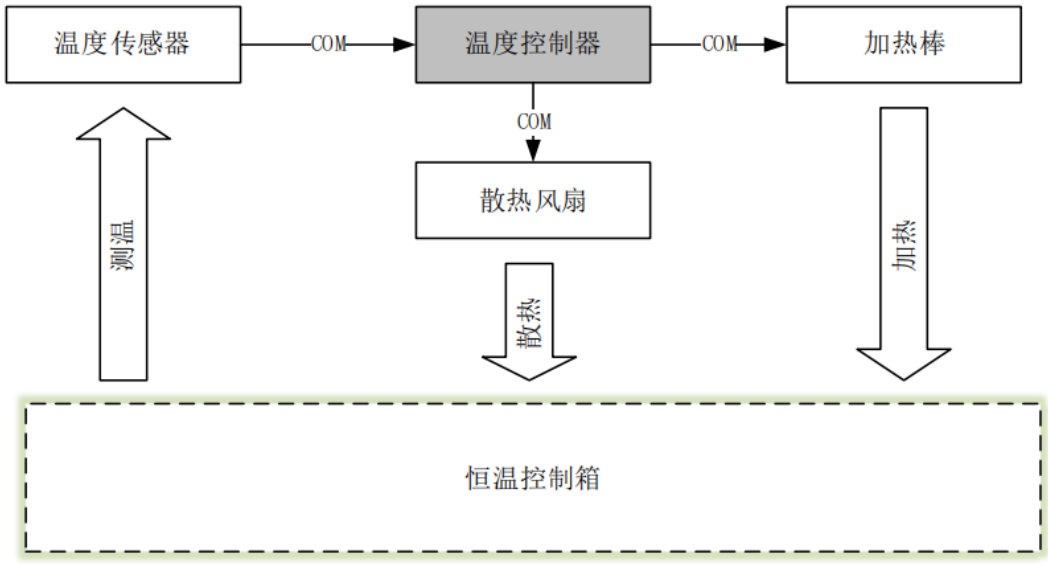
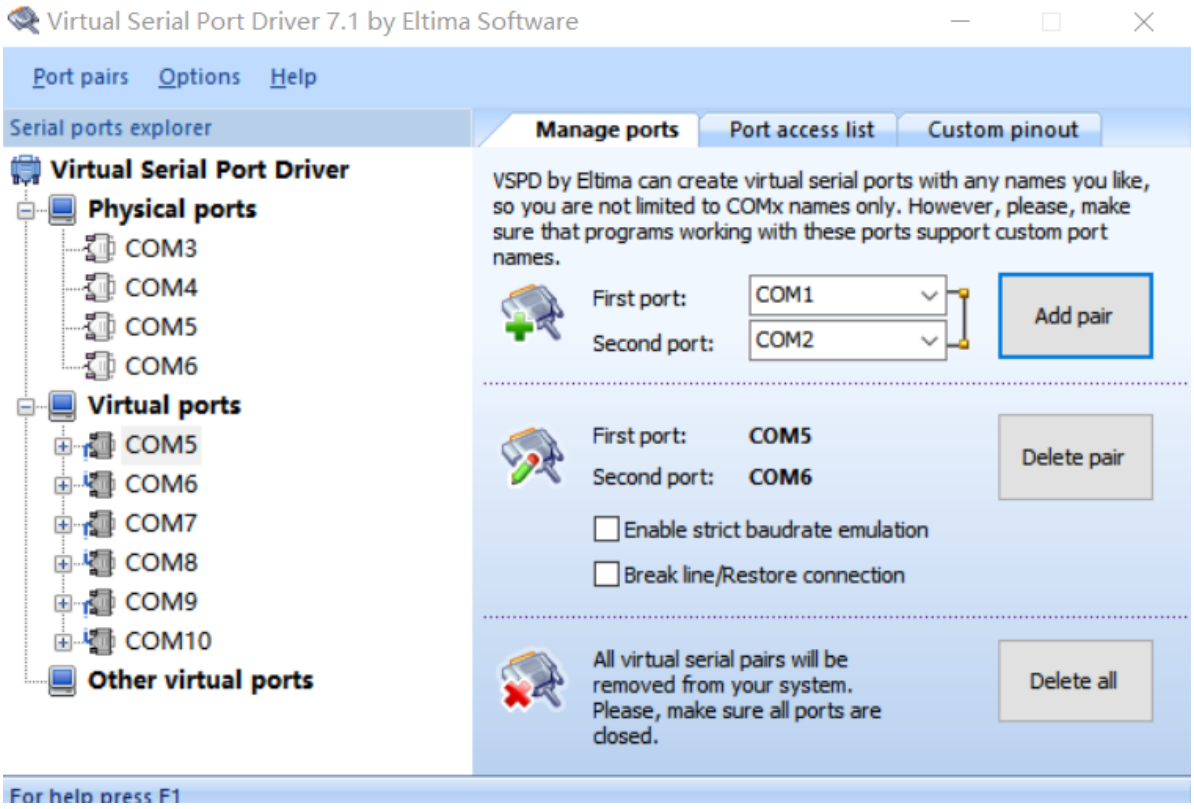


图 1 温度控制系统



（2）打开被测件，并设置环境

2.3 测试功能需求

2.3.1 启动按钮（不做测试要求）

（略）

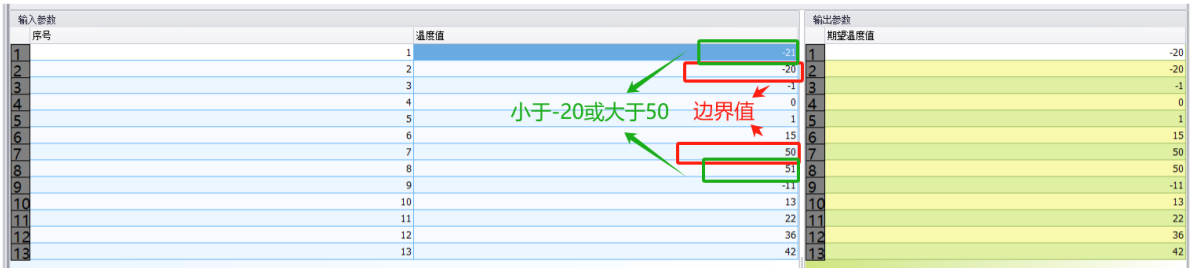
2.3.2 温度采集处理

（1）要求：当前温度的范围是：20-50度。超出范围时，要做截断处理，截断为边界值

（2）根据要求写测试代码，如下所示：

```
# coding:utf-8

import Manu
# arg输入参数
# exp预期输出
def Test(arg,exp):
    CH_232_温度传感器.Clear()
    # 为字段赋值
    Protocol_温度传感器.温度值.Value = arg[1]
    # 协议写
    bool=Protocol_温度传感器.Write()
    print '第%d次期望显示温度值%d'%(arg[0], exp[0])
    # 教程建议采用对话框
    API.Common.Timer.Normal.Sleep(1000)

## Standard_Test:标准测试的方法入口，使用【测试数据】表循环调用Test方法
Standard_Test(Test)
```

（3）在测试数据中，添加测试用例，根据理论课讲的边界值和等价类方法，测试用例设计如下：



（4）运行结果和分析

收发记录

	序号	用户设定温度	采集温度	输出电压	风扇状态
▶	13	20	42	6.7	开
	12	20	36	8.4	关
	11	20	22	11.200000000...	关
	10	20	13	10.350000000...	关
	9	20	-11	19.35	关
	8	20	50	7.1	开
	7	20	50	6.6	关
	6	20	15	13.45	关
	5	20	1	14.95	关
	4	20	0	13.1	关
	3	20	-1	9.35	关
	2	20	-20	10	关
	1	20	-21	10.25	关

第二组数据输入的温度是-21℃，输出的是-21℃，不符合超出范围截断处理的要求。

2.3.3 加热棒

（1）要求：

温度控制器根据温度传感器采集到的当前温度（T）和预先设定的控制温度（T_d），向加热棒发送不同的控制指令，控制加热棒的输出电压（V）。具体情况如下：

（1） $V(k)-V(k-1) = D_P * [e(k)-e(k-1)] + D_I*e(k) + D_d*[e(k)-2e(k-1)+e(k-2)]$

其中：$e(k)= T_d-T(k)$，其中 T_d 为设定温度，$T(k)$ 为当前温度； D_P、D_I、D_d 是控制系数（常量）。D_P=0.05、D_I=0.1、D_d=0.1。

如果计算得到的电压值为负值，截断为 0；电压单位为伏。

（2）根据要求写测试代码，如下所示：

```
import math
import Manu

def VK(t, td, e1, e2):
    # t: 当前温度
    # td: 设定温度
    Dp = 0.1
```

```
    Di = 0.2
    Dd = 0.2
    e = td - t
    result = Dp * (e - e1) + Di * e + Dd * (e - 2 * e1 + e2)
    return result

# arg 输入数据
# exp 预期结果
def Test(arg, exp):
    global e1  # 上一次的温差
    global e2  # 上上次的温差
    global V1  # 输出电压值

    # 通道清理，防止上一次运行的结果在通道中阻塞，影响本次运行结果
    seek_result = CH_232_加热棒.Clear()
    seek_result = CH_232_温度传感器.Clear()

    # 给温度采集器赋值
    Protocol_温度传感器.温度值.Value = arg[2]
    bool = Protocol_温度传感器.Write()
    API.Common.Timer.Normal.Sleep(1000)

    # 第一次
    if arg[0] == 1:
        e1 = arg[1] - arg[2]
    # 第二次
    if arg[0] == 2:
        e2 = arg[1] - arg[2]
        Protocol_加热棒.BlockRead()
        V1 = Protocol_加热棒.加热棒输出电压.Value
    # 三次及以上
    if arg[0] > 2:
        V = VK(arg[2], arg[1], e1, e2) + V1
        print(V1)
        show = []
        str_ = "设定温度为：%d，室温为：%f" % (arg[1], arg[2])
        show.append(str_)
        str_ = "预期电压为：%f" % (V)
        show.append(str_)
        show.append("界面电压显示是否正确？")
        passed = Manu.Check(show)
        e2 = e1
        e1 = arg[1] - arg[2]
        # 协议读，读取通道中传出的数据
        Protocol_加热棒.BlockRead()
        # 获取字段，当前电压，作为下一次的v(k-1)
        V1 = Protocol_加热棒.加热棒输出电压.Value

## Standard_Test:标准测试的方法入口，使用【测试数据】表循环调用Test方法
Standard_Test(Test)
```

（3）在测试数据中，添加测试用例，测试用例设计如下：

输入参数			
	序号	用户设定	采集温度
1	1	20	10
2	2	20	20
3	3	20	20
4	4	20	20
5	5	20	10
6	6	20	20
7	7	20	20
8	8	20	20
9	9	20	10

输出参数	
	输出温度
1	10
2	20
3	20
4	20
5	10
6	20
7	20
8	20
9	10

（4）运行结果和分析

收发记录

	序号	用户设定温度	采集温度	输出电压	风扇状态
▶	31	20	10	5	关
	30	20	20	2.5	关
	29	20	20	2.5	关
	28	20	20	1.5	关
	27	20	10	4	关
	26	20	20	1.5	关
	25	20	20	1.5	关
	24	20	20	0.5	关
	23	20	10	3	关

由测试结果可知，加热棒符合需求说明中的要求。

2.3.4 散热风扇

（1）要求

- 当连续检测到 3 次当前温度大于设定温度+3 时，散热风扇开始转动；
- 当连续检测到 3 次当前温度小于等于设定温度时，散热风扇停止转动；
- 以上两个条件都不满足时，散热风扇的状态保持不变。

（2）根据要求写测试代码，如下所示：

```
import Manu
def Test(arg,exp):
    # 原理：采集三次温度
    # 清理通道
    seekresult = CH_232_温度传感器.Clear()
    seekresult = CH_232_散热风扇.Clear()
    print '测试用例%d'%arg[0]
    # 赋值室温
    Protocol_温度传感器.温度值.Value = arg[1]
    bool = bool=Protocol_温度传感器.Write()
    API.Common.Timer.Normal.Sleep(500)

    Protocol_温度传感器.温度值.Value = arg[2]
    bool = Protocol_温度传感器.Write()
    API.Common.Timer.Normal.Sleep(500)
    CH_232_散热风扇.Clear()

    Protocol_温度传感器.温度值.Value = arg[3]
    bool = Protocol_温度传感器.Write()
    API.Common.Timer.Normal.Sleep(500)
```

```
    show = []

    str = '界面电压显示是否正确？'
    show.append(str)
    passed = Manu.Check(show)

    if(passed):
        print '界面显示与预期一致，界面判断通过'
    else:
        print '界面显示与预期不一致，界面判断不通过'

## Standard_Test:标准测试的方法入口，使用【测试数据】表循环调用Test方法
Standard_Test(Test)
```

（3）在测试数据中，添加测试用例，测试用例设计如下：

	序号	室温1	室温2	室温3		状态
1	1	24	24	25	1	开
2	2	21	21	21	2	不变（开）
3	3	19	19	19	3	关
4	4	26	21	20	4	不变（关）
5	5	10	10	10	5	关

（4）运行结果和分析

	序号	用户设定温度	采集温度	输出电压	风扇状态
	12	20	20	1	关
	11	20	21	1.35	关
	10	20	26	0	关
	9	20	19	0.4	关
	8	20	19	0.3000000000...	关
	7	20	19	0.4	关
	6	20	21	0	开
	5	20	21	0.0999999999...	开
	4	20	21	0.6	开
	3	20	25	0	开
	2	20	24	0	关
	1	20	24	0	关

序号3的测试用例实际运行结果是风扇直接开始转动，不符合连续检测到3 次当前温度大于设定温度+3时，散热风扇开始转动的要求。

序号7的测试用例实际运行结果是风扇直接停止了转动，不符合连续检测到 3 次当前温度小于等于设定温度时，散热风扇停止转动的要求。

2.4 测试接口需求

2.4.1 温度传感器输入接口

（1）要求：

字节号	长度	字段	内容
0-1	2	包头	固定值：0xFF 0xFA
2	1	数据类型 1	固定值：0x01（传感器数据）
3	1	数据类型 2	固定值：0x10（温度）
4	1	数据长度	固定值：0x04（数据长度）
5-8	4	温度值	单精度浮点型，小端字节序
9-10	2	校验和	校验位，xx xx（从第 2 号到 8 号字节按字节进行累加和，得到校验码）小端字节序
11	1	包尾	固定值：0x0F

输入接口处理时，要考虑数据帧格式的容错处理，容错处理的要求如下：

（1） 当接收到的校验和字段发生错误时，应做丢包处理。

（2） 包头、数据类型 1、数据类型 2、数据长度、包尾应该按照要求填写，否则做出丢包处理。

(2) 根据要求，写测试代码：

```
import Manu
def Test(arg,exp):
    datas = arg[0].split(',')
    data = []
    for a in datas:
        data.append(int(a,16))
    # 清理通道
    seekresult = seekresult=CH_232_温度传感器.Clear()
    bool = bool=CH_232_温度传感器.Write(data)

    show = []
    str = '预期结果：%s'%arg[1]
    show.append(str)
    Manu.Check(show)

## Standard_Test:标准测试的方法入口，使用【测试数据】表循环调用Test方法
Standard_Test(Test)
```

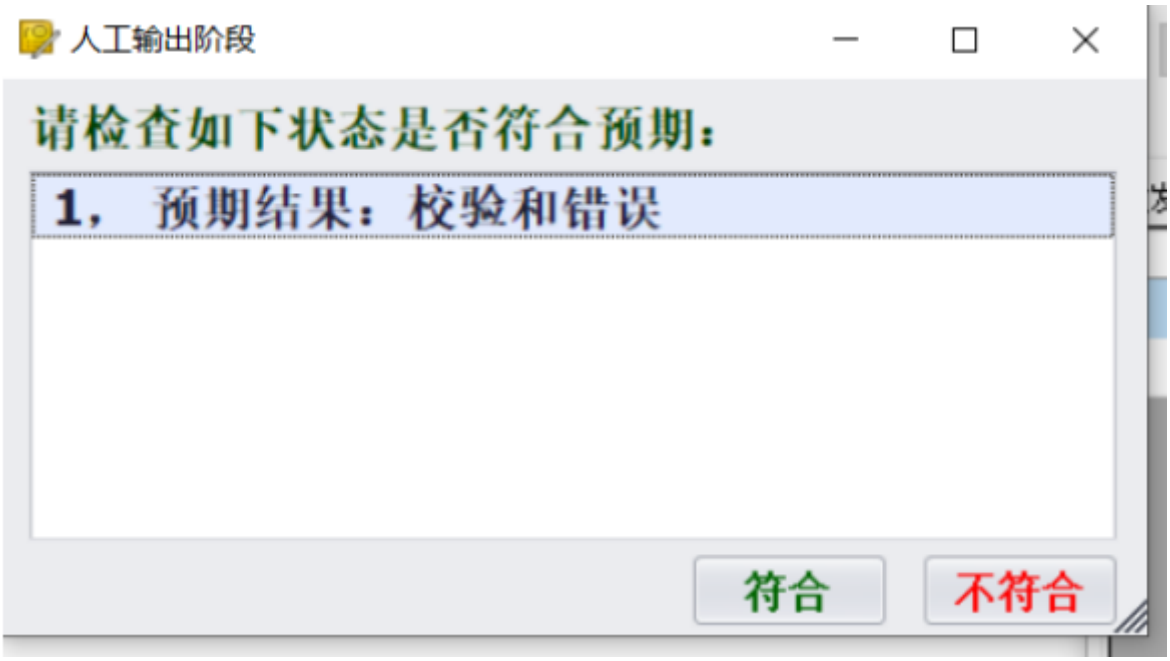
(3) 在测试数据中，添加测试用例，每个错误类型均设置一个测试用例，测试用例设计如下：

	输入参数		输出参数
	数据帧	描述	预期
1	FF,FA,01,10,04,00,00,20,41,76,00,0F	正确帧，10度	10度
2	FF,FA,01,10,04,00,00,20,41,80,00,0F	校验和错误	抛出异常
3	AA,FA,01,10,04,00,00,20,41,76,00,0F	包头错误	抛出异常
4	FF,FA,07,10,04,00,00,20,41,82,00,0F	数据类型1错误	抛出异常
5	FF,FA,01,11,04,00,00,20,41,77,00,0F	数据类型2错误	抛出异常
6	FF,FA,01,10,05,00,00,20,41,77,00,0F	数据长度错误	抛出异常
7	FF,FA,01,10,04,00,00,20,41,76,00,F0	包尾错误	抛出异常

(4) 运行结果和分析

收发记录

	序号	用户设定温度	采集温度	输出电压	风扇状态
▶	2	10	10	0	关
	1	10	10	0	关



校验和错误没有抛出，当输入数据是校验和数据错误的数据帧，传感器收到格式错误的数据帧后没有丢包，仍接受该数据

2.4.2 控制加热棒输出接口

（1）要求：

温度控制器依据功能需求向加热棒发送数据，其数据格式如表 2 所示。

表 2 控制加热棒输出接口数据帧格式

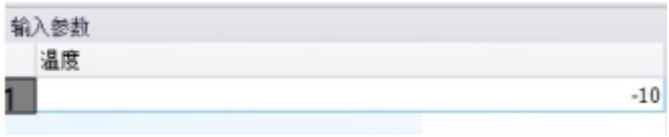
字节号	长度	字段	内容
0-1	2	包头	固定值：0xFF 0xFA
2	1	数据类型 1	固定值：0x02（执行数据）
3	1	数据类型 2	固定值：0x11（工作电机组）
4	1	数据长度	固定值：0x04
5-8	4	加热棒输出电压（单位为伏）	单精度浮点数，小端字节序
9-10	2	校验和	校验位，xx xx（从第 2 号到 8 号字节按字节进行累加和，得到校验码），小端字节序
11	1	包尾	固定值：0x0F

（2）根据要求，写测试代码：

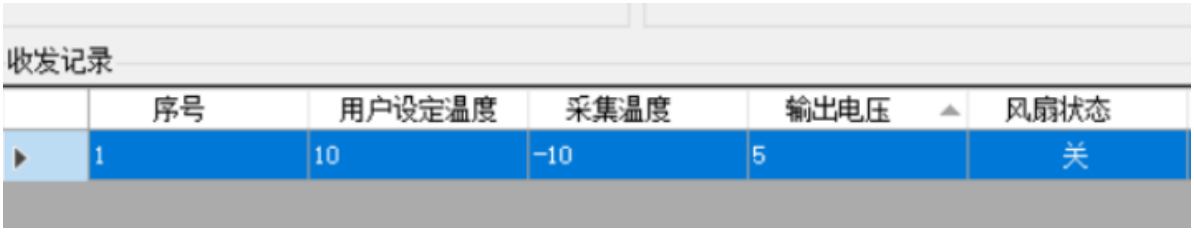
```
def Test(argexp):
```

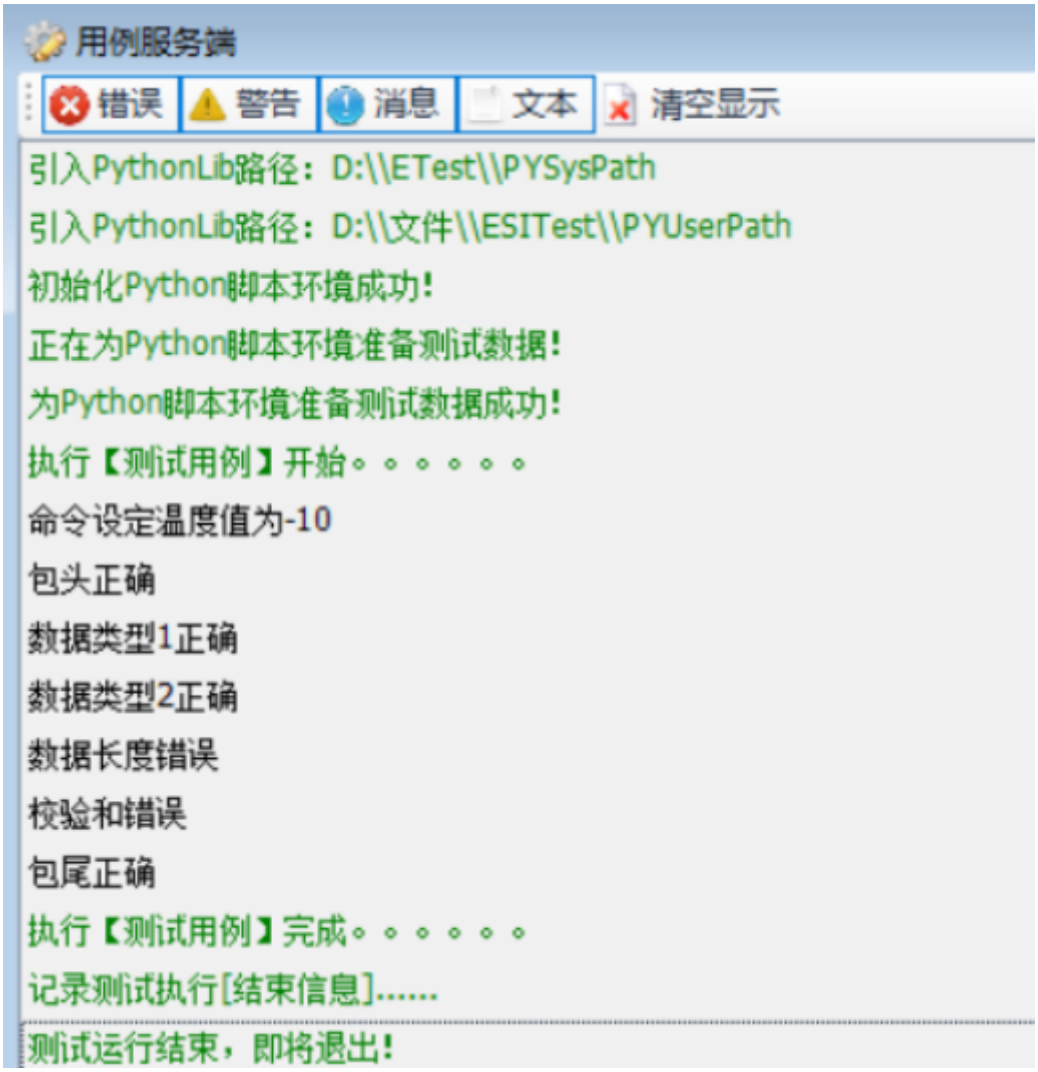
```
    seekresult=CH_232_加热棒Clear()
    seekresult=CH_232_温度传感器.Clear()

    # 设置温度值
    print '命令设定温度值为%d' % (arg[0])
    Protocol_温度传感器.温度值.Value=arg[0]
    bool=Protocol_温度传感器.Write()

    API.Common.Timer.Normal.Sleep(1000)

    # 判断数据包是否正确
    Protocol_加热棒.BlockRead()
    if Protocol_加热棒.包头.Value != 0xFFFA:
        print '包头错误'
    else:
        print '包头正确'
    if Protocol_加热棒.数据类型1.Valuel != 0x02:
        print '数据类型1错误'
    else:
        print '数据类型1正确'
    if Protocol_加热棒.数据类型2.Value!= 0x11:
        print '数据类型2错误'
    else:
        print '数据类型2正确'
    if Protocol_加热棒.数据长度Value != 0x04:
        print '数据长度错误'
    else:
        print '数据长度正确'
    if Protocol_加热棒.数据长度.Value!= 0x04:
        print '数据长度错误'
    else:
        print '数据长度正确'
    if Protocol_加热棒.检验.Checked!= True:
        print '校验和错误'
    else:
        print'校验和正确'
    if Protocol_加热棒.数据长度.Value!= 0x04:
        print '数据长度错误'
    else:
        print '数据长度正确'
    if Protocol_加热棒.检验.Checked!= True:
        print '校验和错误'
    else:
        print '校验和正确'
    if Protocol_加热棒.包尾.Value!= 0x0F:
        print '包尾错误'
    else:
        print '包尾正确'

## Standard_Test:标准测试的方法入口，使用【测试数据】表循环调用Test方法
Standard_Test(Test)
```

(3) 在测试数据中，添加测试用例，测试用例设计如下：



（4）运行结果和分析





数据长度和校验和实际显示错误，实测任意条件下，发送到发热棒的数据帧，其数据长度字段的值均有误

2.4.2 控制散热风扇输出接口

（1）要求：

温度控制器依据功能需求向散热风扇发送数据，其数据格式如表 3 所示。

表 3 控制散热风扇输出接口数据帧格式

字节号	长度	字段	内容
0-1	2	包头	固定值：0xFF 0xFA
2	1	数据类型 1	固定值：0x02（执行数据）
3	1	数据类型 2	固定值：0x22（风扇）
4	1	数据长度	固定值：0x01
5	1	操作指令	0：关闭风扇 1：打开风扇 无符号整形
6-7	2	校验和	校验位，xx xx（从第 2 号到 5 号字节按字节进行累加和，得到校验码），小端字节序
8	1	包尾	固定值：0x0F

（2）根据要求，写测试代码：

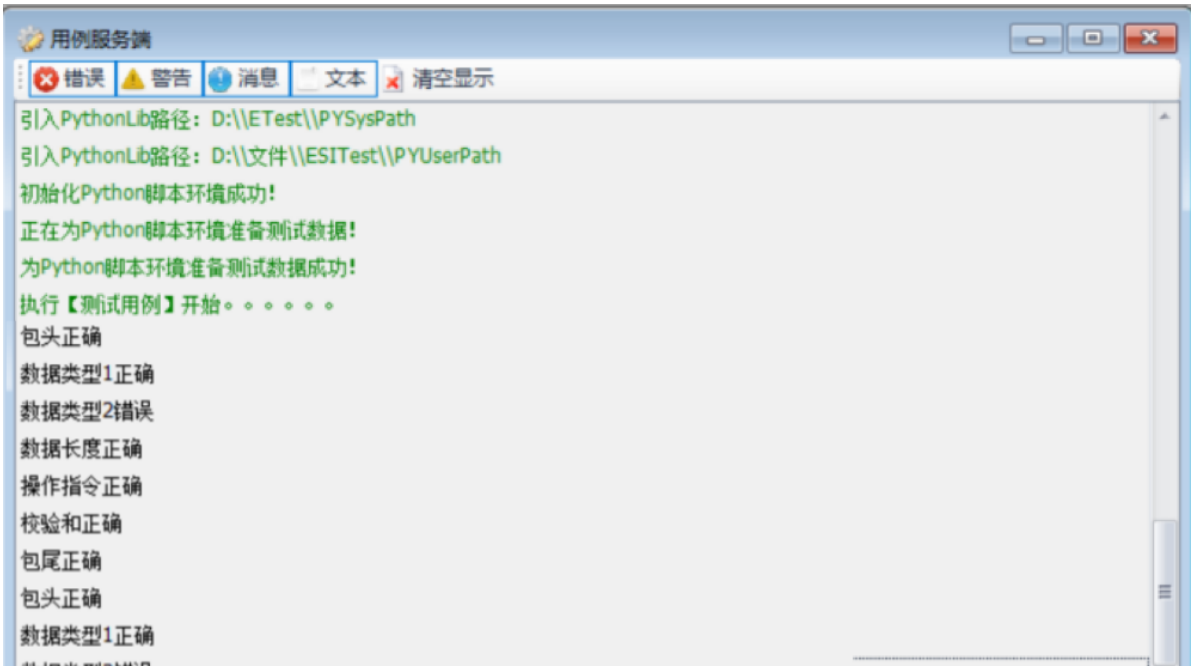
```
def Test(argexp):
    seekresult=CH_232_加热棒Clear()
    seekresult=CH_232_温度传感器.Clear()
    seekresult=CH_232_散热风扇.Clear()

    # 设置初始状态，使风扇不转动
    print '命令设定温度值为%d' % (arg[0])
    Protocol_温度传感器.温度值.Value=arg[0]
    bool=Protocol_温度传感器.Write()
    API.Common.Timer.Normal.Sleep(1000)

    Protocol_温度传感器.温度值.Value=arg[1]
    bool=Protocol_温度传感器.Write()
    API.Common.Timer.Normal.Sleep(1000)

    Protocol_温度传感器.温度值.Value=arg[2]
    bool=Protocol_温度传感器.Write()
    API.Common.Timer.Normal.Sleep(1000)

    # 风扇不转动时，判断数据包是否正确
    Protocol_散热风扇.BlockRead()
    if Protocol_散热风扇.包头.Value != 0xFFFA:
        print '包头错误'
    else:
        print '包头正确'
    if Protocol_散热风扇.数据类型1.Valuel != 0x02:
        print '数据类型1错误'
    else:
        print '数据类型1正确'
    if Protocol_散热风扇.数据类型2.Value!= 0x22:
        print '数据类型2错误'
    else:
```

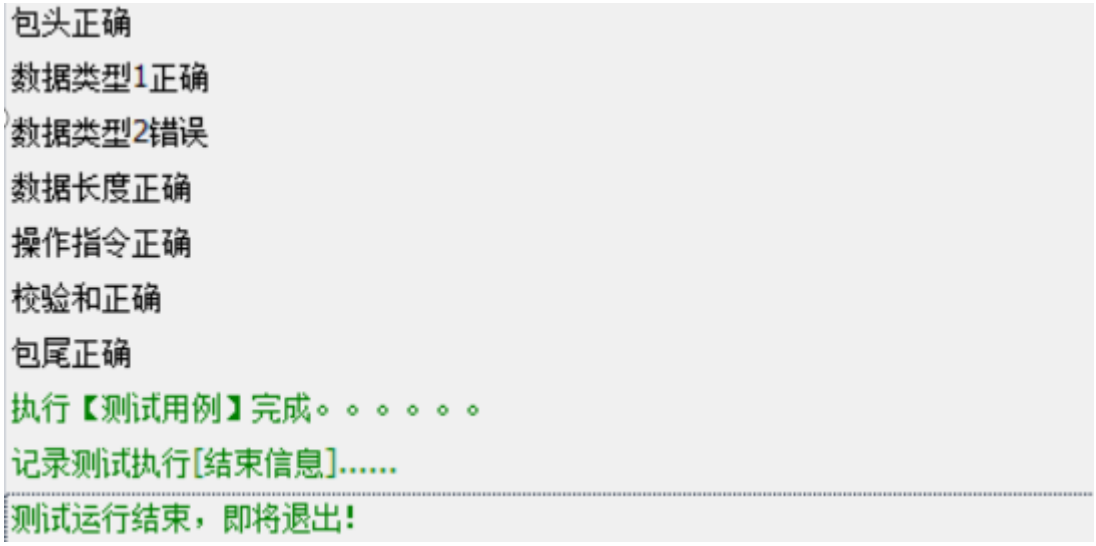
```
        print '数据类型2正确'
    if Protocol_散热风扇.数据长度Value != 0x01:
        print '数据长度错误'
    else:
        print '数据长度正确'
    if Protocol_散热风扇.操作指令.Value!= 0:
        print '操作指令错误'
    else:
        print '操作指令正确'
    if Protocol_散热风扇.检验.Checked!= True:
        print '校验和错误'
    else:
        print '校验和正确'
    if Protocol_散热风扇.包尾.Value!= 0x0F:
        print '包尾错误'
    else:
        print '包尾正确'

    # 设置初始状态，使风扇转动
    seekresult=CH_232_散热风扇.Clear()
    for i in [15, 15, 15]:
        Protocol_温度传感器.温度值.Value = i
        bool=Protocol_温度传感器.Write()
        API.Common.Timer.Normal.Sleep(100)

    # 风扇转动时，判断数据包是否正确
    Protocol_散热风扇.BlockRead()
    if Protocol_散热风扇.包头.Value != 0xFFFA:
        print '包头错误'
    else:
        print '包头正确'
    if Protocol_散热风扇.数据类型1.Valuel != 0x02:
        print '数据类型1错误'
    else:
        print '数据类型1正确'
    if Protocol_散热风扇.数据类型2.Value!= 0x22:
        print '数据类型2错误'
    else:
        print '数据类型2正确'
    if Protocol_散热风扇.数据长度Value != 0x01:
        print '数据长度错误'
    else:
        print '数据长度正确'
    if Protocol_散热风扇.操作指令.Value!= 0:
        print '操作指令错误'
    else:
        print '操作指令正确'
    if Protocol_散热风扇.检验.Checked!= True:
        print '校验和错误'
    else:
        print '校验和正确'
    if Protocol_散热风扇.包尾.Value!= 0x0F:
        print '包尾错误'
    else:
        print '包尾正确'
```

```
## Standard_Test:标准测试的方法入口，使用【测试数据】表循环调用Test方法
Standard_Test(Test)
```

（3）在测试数据中，添加测试用例，测试用例设计如下：

（4）运行结果和分析





三次都返回了数据类型2错误，实测任意条件下，发送到散热风扇的数据帧，其数据类型2字段的值均有误，说明操作指令存在错误

2.5 测试性能需求

（1）要求：

系统的温控稳定时间指标为：不大于 1 分钟。

要求测出下列三种情况下的温控稳定时间，并且判定是否满足温控稳定时间指标。

- 当设定温度为 10，室外温度为 0 时；
- 当设定温度为 20，室外温度为 0 时；
- 当设定温度为 0，室外温度为 10 时。

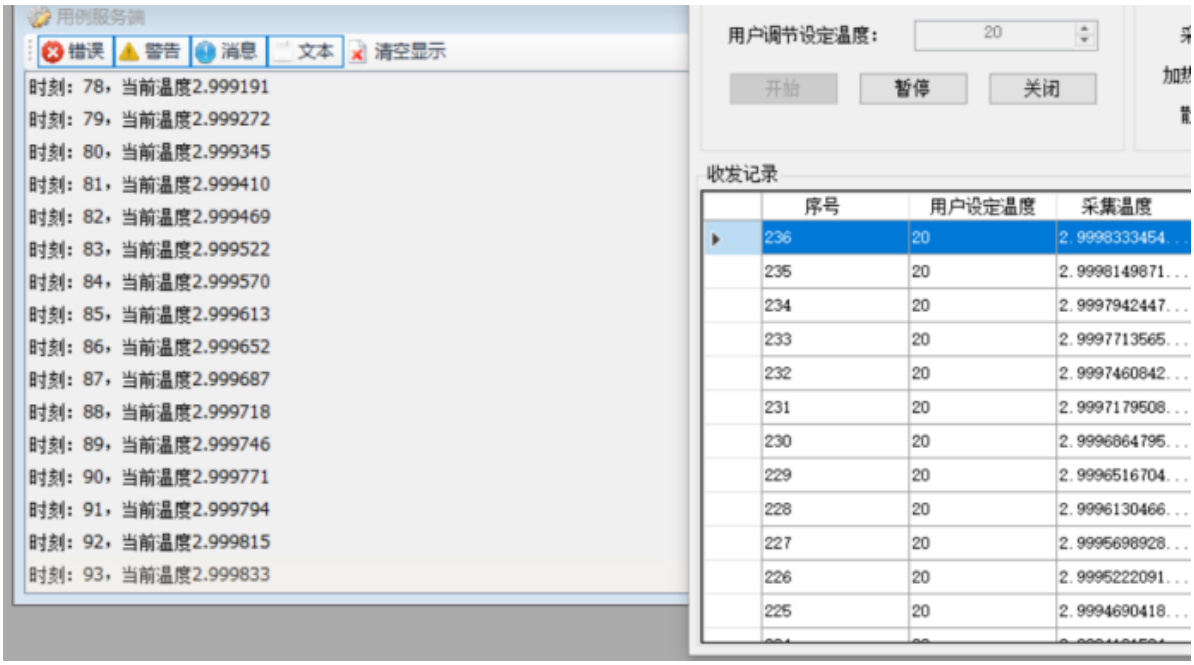
（2）根据要求编写代码

```
# coding:utf-8

# 设定温度为10℃，初始恒温箱外部温度值为3℃
import time


def Test():
    # 需依次修改Tc T0的值为测试数据中对应的值，并在温度控制器调节用户设定温度
    Tc = 10  # 设定温度
    T0 = 0  # 初始外部温度
    T = T0
    n = 0
    Fan = 0
    time0 = time.time()
    print '开始性能测试，设定温度为%s℃，初始外部温度为%s℃' % (Tc, T0)
    while True:
        clear = CH_232_温度传感器.Clear()
        clear = CH_232_加热棒.Clear()
        clear = CH_232_散热风扇.Clear()
        Protocol_温度传感器.温度值.Value = T
        bool = Protocol_温度传感器.Write()
        Protocol_加热棒.BlockRead()
        Protocol_散热风扇.BlockRead()
        V = Protocol_加热棒.加热棒输出电压.Value
        if V < 0:
            V = 0

        Qb = V * V * 0.2
        if Protocol_散热风扇.操作指令.Value == 1:
            Fan = 2
        elif Protocol_散热风扇.操作指令.Value == 0:
            Fan = 0

        Qfan = 0.1 * (T - T0) + Fan
        T = T + (Qb - Qfan)
        if abs(T - Tc) < 0.5:
            n = n + 1
        else:
            n = 0

        if n >= 10:
            time1 = time.time()
            print '稳定时间：%d' % (time1 - time0)
            if time1 - time0 <= 60:
                print '满足控温稳定时间指标要求'
            break

        time2 = time.time()
        if time2 - time0 > 60:
```

```
            print '不满足控温稳定时间指标要求'
            break
        API.Common.Timer.Normal.Sleep(1000)

Standard_Test(Test)
```

（3）在测试数据中，添加测试用例，测试用例设计如下：

（4）输出结果



控温稳定时间分别约为36秒、29秒、大于60秒。两次满足指标要求，一次不满足。

3、实验分析

需求编号	测试用例描述	输入	预期输出	实际输出	问题描述	备注
2.2	设定温度传感器采集的温度值为不同值，测试数据是否正确显示，超边界值有无截断处理	-21	-20	-21	没有发生截断	
2.4	用户设定温度为20，输入特定室温值，连续检测室温，测试散热风扇状态变化是否符合预期	24; 24; 25	关; 关; 开	关; 开; 开	本该在连续3次检测到室温大于设定温度3度以上时才开风扇，但实际上第二次检测到后就开启了风扇	
2.4	用户设定温度为20，输入特定室温值，连续检测室温，测试散热风扇状态变化是否符合预期	19; 19; 19	开; 开; 关	关; 关; 关	本该在连续3次检测到室温低于设定温度时才关掉风扇，但实际上第一次检测到低温后就关闭了风扇	

需求编号	测试用例描述	输入	预期输出	实际输出	问题描述	备注
3.1	输入自定义的数据帧，测试温度传感器的容错处理情况，是否正确处理数据帧	FF FA 01 10 04 00 00 20 41 80 00 0F	丢包（校验和错误）	没有丢包	发送的是校验和数据错误的20℃数据帧，传感器收到格式错误的数据帧后没有丢包，仍接受该数据	
3.2	接受温度控制器向加热棒发送的数据帧，测试格式是否正确	温度控制器在10℃下发送的数据帧	接受到正确的数据帧	数据帧的数据长度字段值错误	数据长度字段值应为0x04，实际为0x02	实测任意条件下，发送到发热棒的数据帧，其数据长度字段的值均有误（0x02）
3.2	接受温度控制器向加热棒发送的数据帧，测试格式是否正确	温度控制器在10℃下发送的数据帧	接受到正确的数据帧	数据帧的校验和值错误	校验和的值应为0x37 0x01，实际为0x0000	实测任意条件下，发送到发热棒的数据帧，其校验和字段的值均有误（0x0）
3.3	接受温度控制器向散热风扇发送的数据帧,测试格式是否正确	温度控制器在多种温度条件下发送的数据帧（5，5，5；21，22，22；25，25，25）	接受到正确的数据帧	数据帧的数据类型2字段值错误	数据类型2字段值应为0x22，实际为0x11	实测任意条件下，发送到散热风扇的数据帧，其数据类型2字段的值均有误（0x11）
4.1	设定温度为10℃，初始室外温度为20℃	10；20	无	控温稳定时间约为36秒	能够满足控温稳定时间指标要求，时间约为36秒	

需求编号	测试用例描述	输入	预期输出	实际输出	问题描述	备注
4.1	设定温度为20℃，初始室外温度为0℃	20；0	无	控温稳定时间约为29秒	能够满足控温稳定时间指标要求，时间约为29秒	
4.1	设定温度为0℃，初始室外温度为10℃	0；10	无	控温稳定时间大于60秒	不能够满足控温稳定时间指标要求	

4、实验总结

这是第一次做嵌入式测试的实验，刚开始做确实很懵，不知道如何下手，之后仔细看实验指导书和视频教程，完成了本次实验。

通过本次实验，我学到了如何设计全面而有针对性的测试用例，以确保系统在各种情况下都能正常运行。测试用例的设计需要考虑到系统的各个模块和组件，覆盖不同的输入情况，以及可能的边界情况。这有助于发现潜在的问题，提高系统的鲁棒性。