

实验二：嵌入式测试

一、实验目的

熟悉嵌入式测试的工具的使用以及测试脚本代码的编写


二、实验环境

Etest平台

三、实验内容

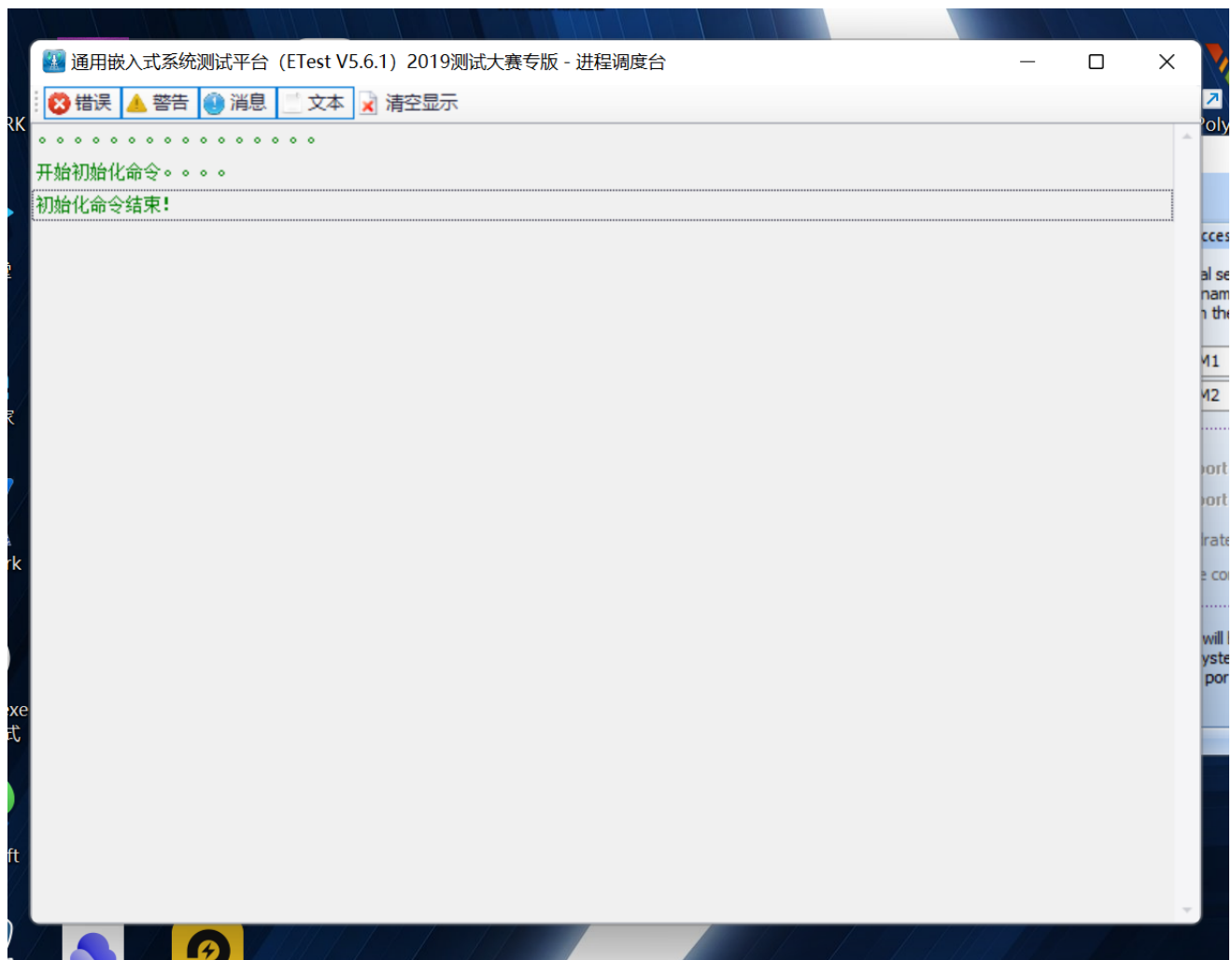
①安装相关软件



 vspdconfig.exe

②环境搭建：

Etest-进程调度服务



打开Etest-测试设计平台

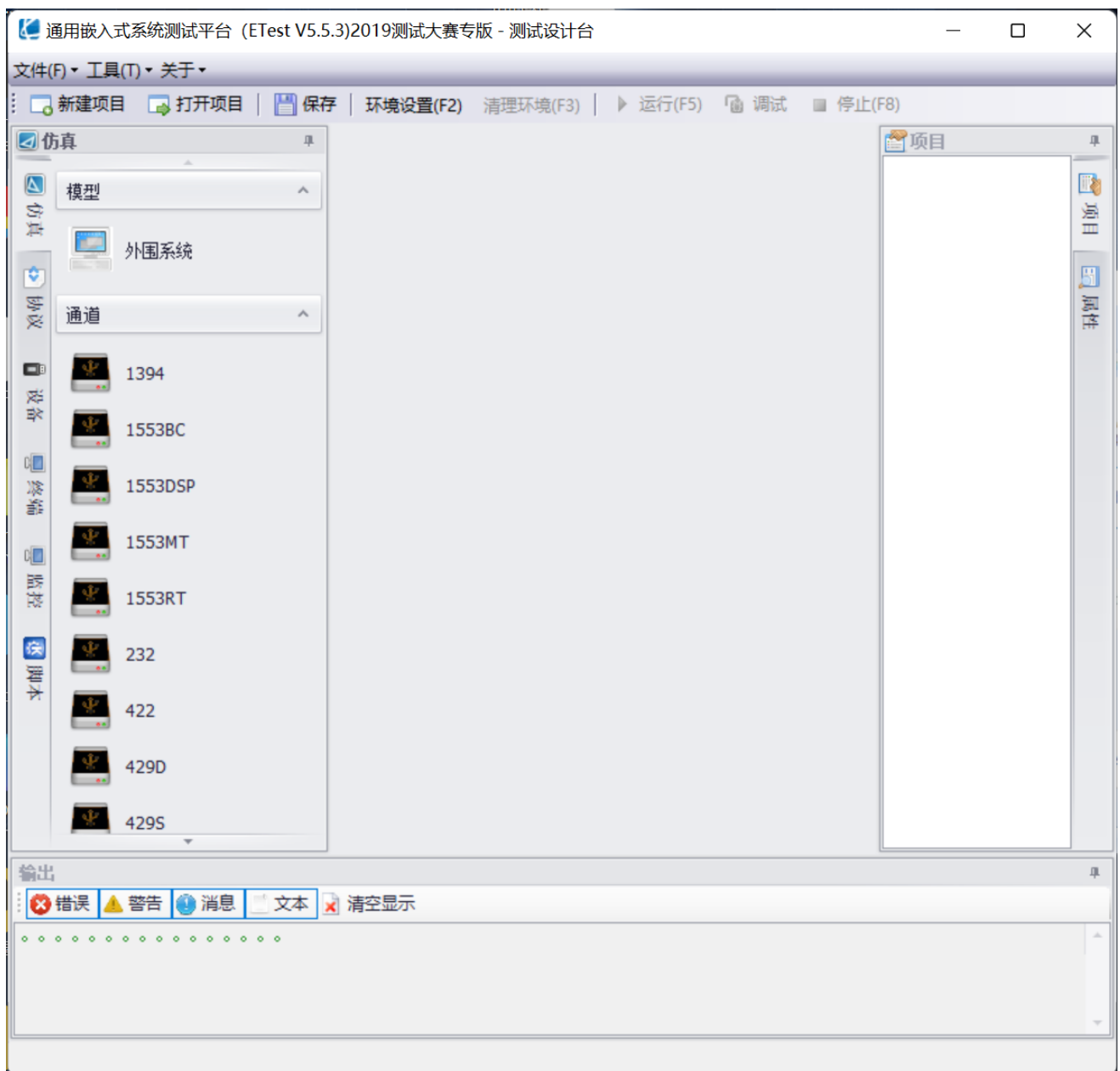


通用嵌入式系统测试平台

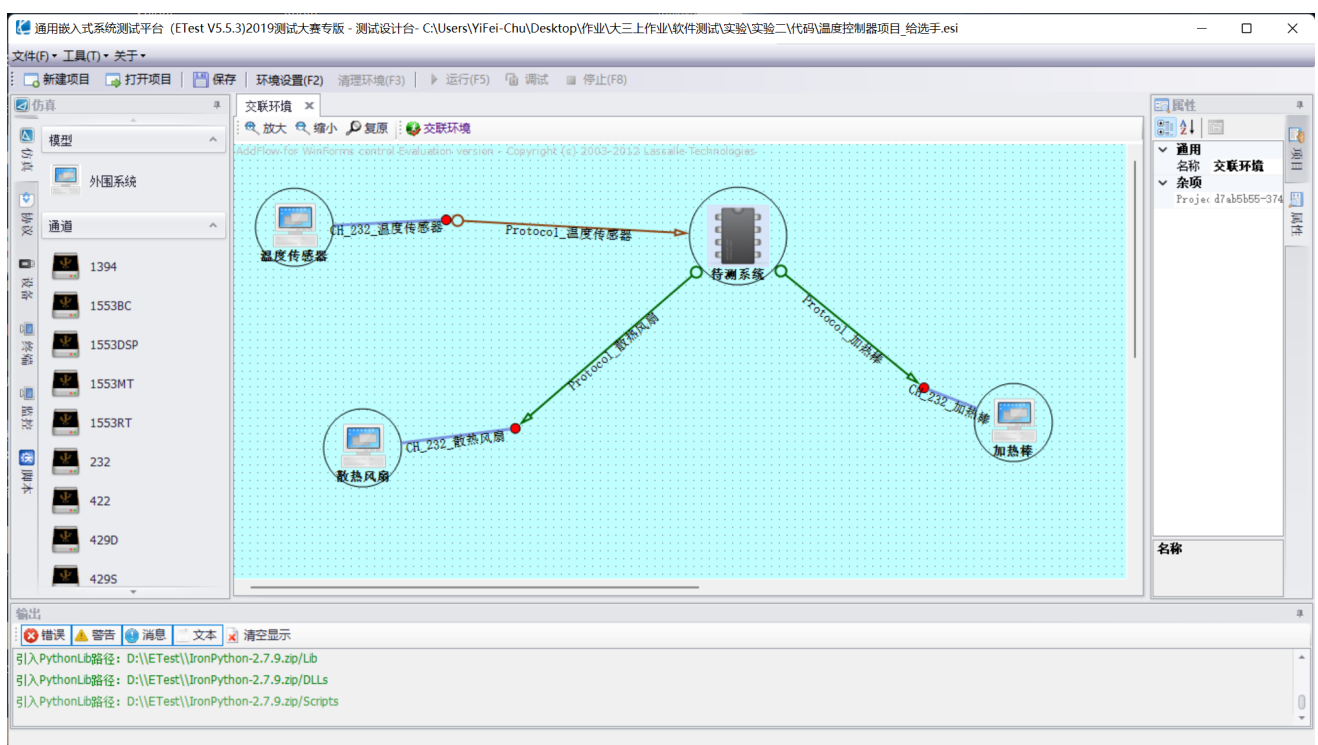
正在加载：加载标准输入/输出模块...



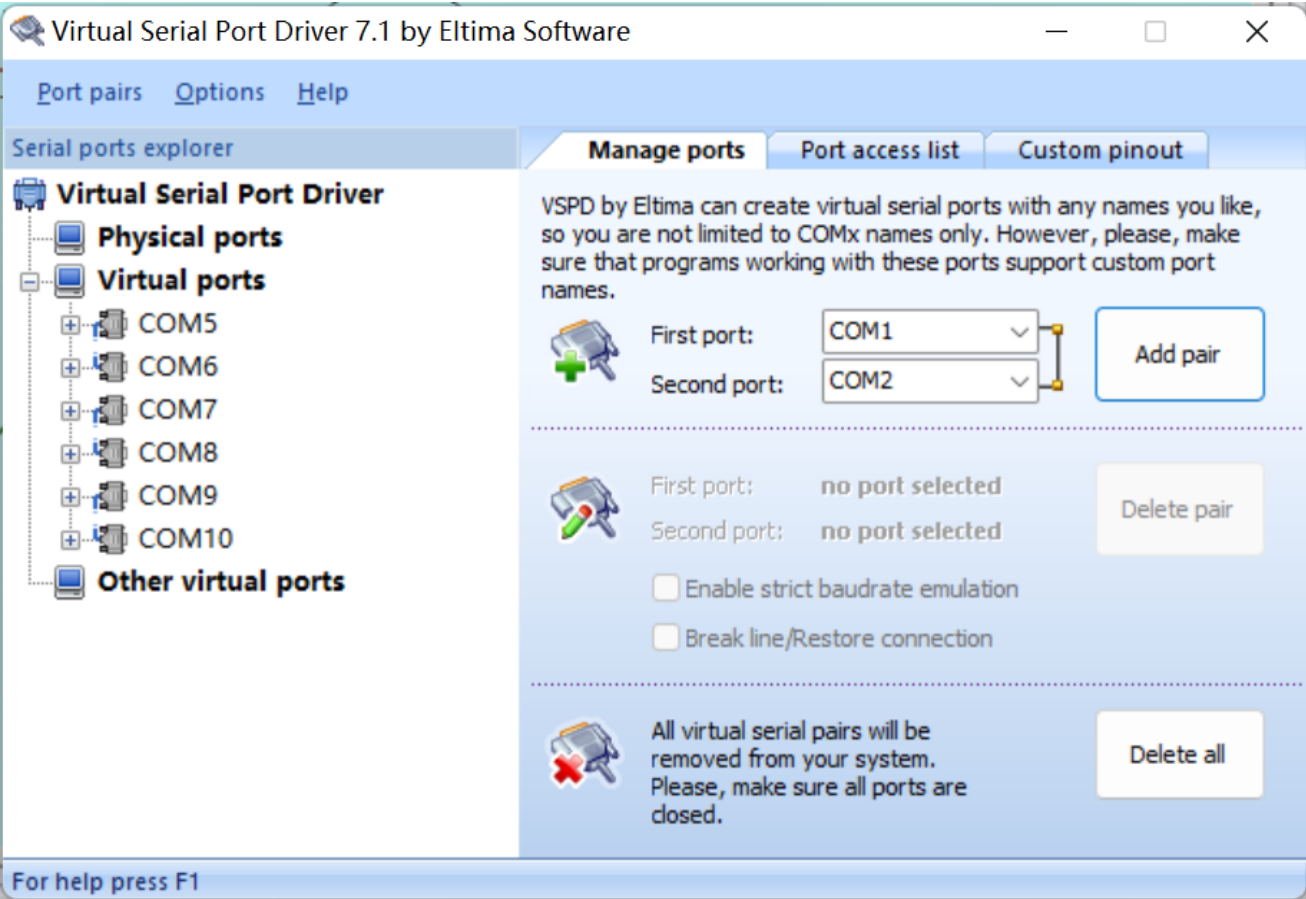
北京旋极信息技术股份有限公司



打开待测件



使用vspd绑定对应串口



打开SensorContorl.exe





Etest环境设置



③根据问题编写脚本进行测试：（具体代码见待测件）

第一项（2.2）——温度采集处理：

温度采集控温功能测试数据集：

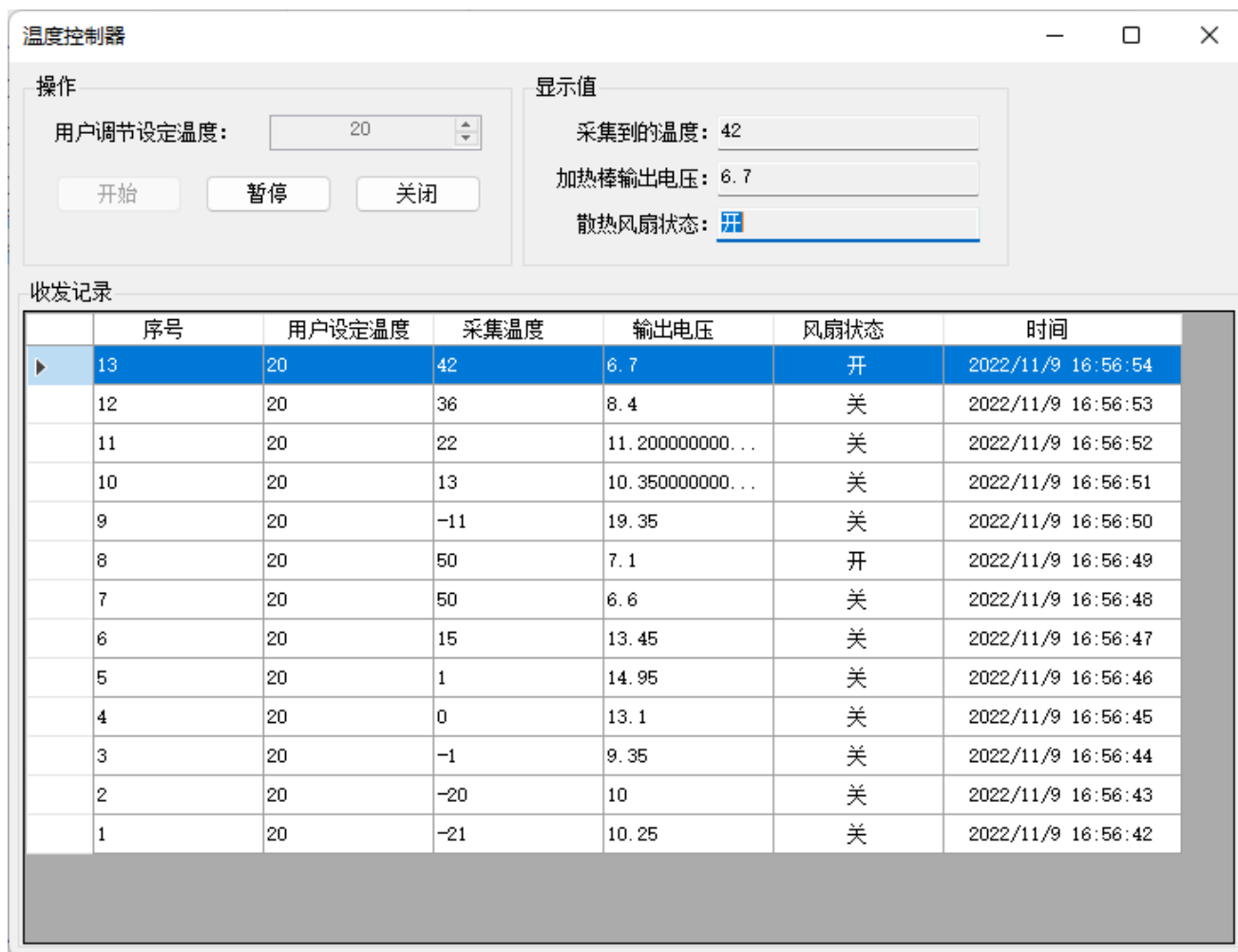
测试用例表			
基本操作 导出用例表 导入用例表 添加行 删除行 参数组合 组合模版 重新生成 清除组合规则			
输入参数			输出参数
序号	温度		预期
1	1	-30	1 -20
2	2	-21	2 -20
3	3	-20	3 -20
4	4	-10	4 -10
5	5	0	5 0
6	6	10	6 10
7	7	20	7 20
8	8	30	8 30
9	9	40	9 40
10	10	50	10 50
11	11	51	11 50
12	12	60	12 50
13	13	100	13 50

代码：

```
# coding:utf-8
def Test(arg,exp):
    CH_232_温度传感器.Clear()
    Protocol_温度传感器.温度值.Value = arg[1]
    bool=Protocol_温度传感器.Write()
    print '第%d次期望显示温度值%d'%(arg[0], exp[0])
    API.Common.Timer.Normal.Sleep(1000)

Standard_Test(Test)
```

测试结果：



具体分析见实验分析与总结

第二项 (2.3) ——加热棒

预设的测试数据集:

输入参数				输出参数	
序号	设定温度	采集温度			
1	1	20	20	1	
2	2	20	20	2	
3	3	20	20	3	
4	4	20	10	4	

代码:

```
import math
import Manu
def VK(t, td, e1, e2):
    # t: 当前温度
    # td: 设定温度
    Dp = 0.05
    Di = 0.1
    Dd = 0.1
    e = td - t;

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



```

def Test(arg,exp):
    global e1 # 上一次的温度， 设定温度-当前温度
    global e2 # 上上次的温度
    global V1 # 当前时刻的输出电压值

    # 通道清理
    seekresult = CH_232_加热棒.Clear()
    seekresult = CH_232_温度传感器.Clear()

    # 给温度采集器赋值， 当前温度
    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)

    # arg[0] 序号
    # arg[1] 设定温度
    # arg[2] 当前温度

    #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]
        Protocol_加热棒.BlockRead()
        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)

```

测试结果:

温度控制器

操作

用户调节设定温度：

20

开始

暂停

关闭

显示值

采集到的温度：10

加热棒输出电压：4

散热风扇状态：

关

收发记录

	序号	用户设定温度	采集温度	输出电压	风扇状态	时间
▶	27	20	10	4	关	2022/10/19 21:22:14
	26	20	20	1.5	关	2022/10/19 21:21:58
	25	20	20	1.5	关	2022/10/19 21:21:57
	24	20	20	0.5	关	2022/10/19 21:21:56

具体分析见实验分析与总结

第三项 (2.4) —— 散热风扇

预设测试数据集：

基本操作 导出用例表 导入用例表 添加行 删除行 参数组合 组合模版 重新生成 清除组合规则

输入参数					输出参数	
序号	温度1	温度2	温度3		散热风扇状态	
1	1	24	24	24	1	转动
2	2	21	21	21	2	不变（开）
3	3	20	20	20	3	关闭
4	4	23	23	23	4	不变（关闭）
5	5	24	24	24	5	转动
6	6	10	10	10	6	关闭

代码：

```
import Manu

def Test(arg,exp):
    # 写入数据前，先清理通道
    seekresult = CH_232_温度传感器.Clear()
    seekresult = CH_232_散热风扇.Clear()

    # 写入第一次温度
    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)

    #写入第二次温度
    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()

    #写入第三次温度
    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)
```

```

#确保控制散热风扇的指令能够正常发出去
Protocol_散热风扇.BlockRead()
show = []
str = '第%d次期望散热风扇输出值: %s 实际输出值: %d' % (arg[0], exp[0], Protocol_散热风扇.操作
指令.Value)
show.append(str)
Manu.Check(show)

Standard_Test(Test)

```

测试结果:

温度控制器

— □ ×

操作

用户调节设定温度: 20

开始 暂停 关闭

显示值

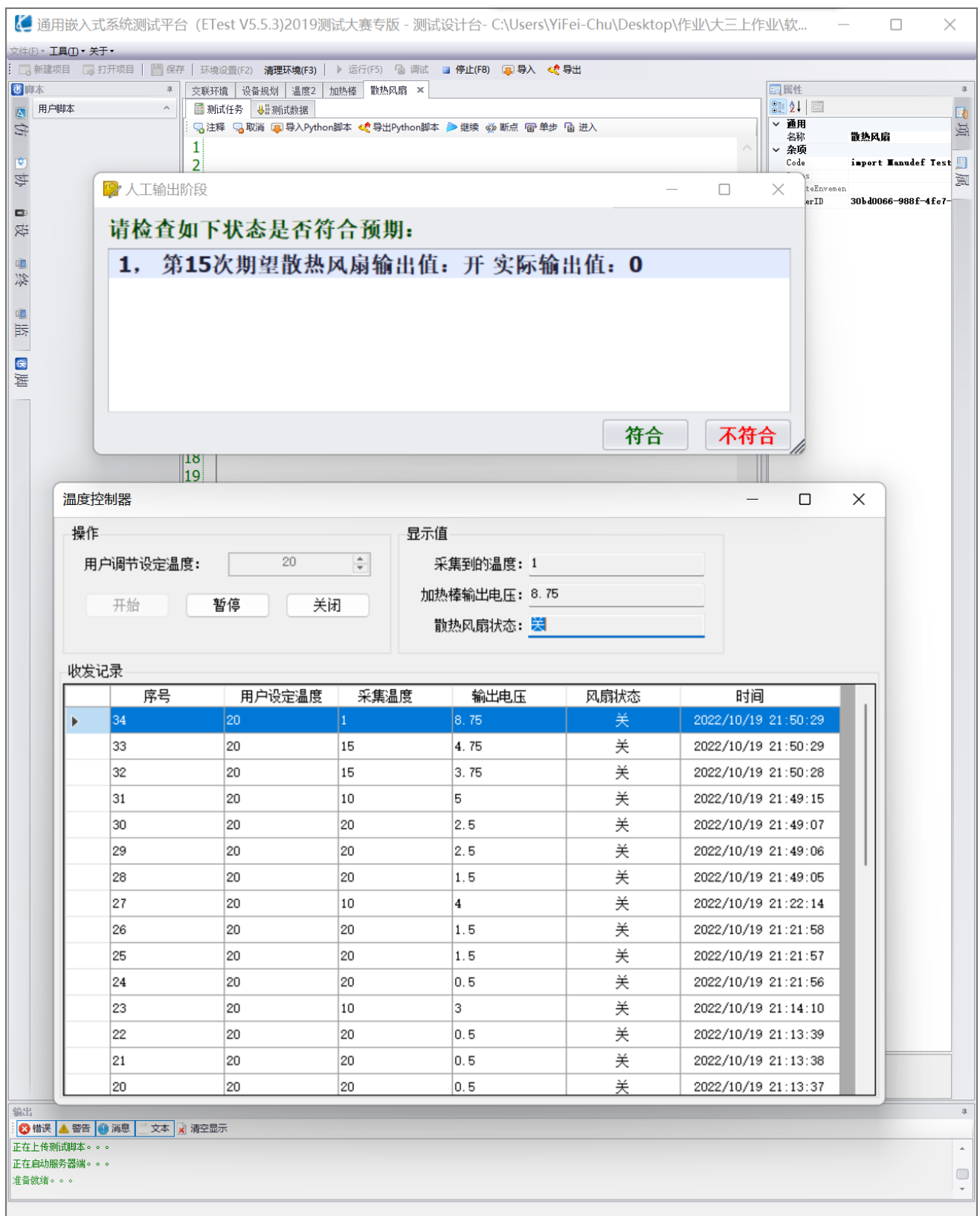
采集到的温度: 10

加热棒输出电压: 3.7

散热风扇状态: 关

收发记录

	序号	用户设定温度	采集温度	输出电压	风扇状态	时间
	18	20	10	3.7	关	2022/10/19 22:52:05
	17	20	10	2.7	关	2022/10/19 22:52:04
	16	20	10	3.1000000000...	关	2022/10/19 22:52:04
	15	20	24	0	开	2022/10/19 22:52:01
	14	20	24	0	开	2022/10/19 22:52:00
	13	20	24	0	关	2022/10/19 22:52:00
	12	20	23	0	关	2022/10/19 22:51:57
	11	20	23	0	关	2022/10/19 22:51:56
	10	20	23	0	关	2022/10/19 22:51:56
	9	20	20	0.0500000000...	关	2022/10/19 22:51:49
	8	20	20	0.0500000000...	关	2022/10/19 22:51:49
▶	7	20	20	0.1500000000...	关	2022/10/19 22:51:48
	6	20	21	0	开	2022/10/19 22:51:46
	5	20	21	0	开	2022/10/19 22:51:45
	4	20	21	0.3500000000...	开	2022/10/19 22:51:45
	3	20	24	0	开	2022/10/19 22:51:39
	2	20	24	0	关	2022/10/19 22:51:39
	1	20	24	0	关	2022/10/19 22:51:38



具体分析见实验分析与总结

第四项 (3.1) ——温度传感器输入接口

预先设置的测试数据集:

测试任务		测试数据	
测试用例表			
基本操作 导出用例表 导入用例表 添加行 删除行 参数组合 组合模版 重新生成 清除组合规则			
输入参数		输出参数	
输入数据包		期望结果	
1	FF FA 01 10 04 00 00 00 40 55 00 0F	1	正常包，显示温度值2
2	FF FA 01 10 04 00 00 40 40 D3 00 0F	2	检验和错误，丢包，temp_value = 3,预期显示温度值2
3	FF FA 01 10 04 00 00 40 40 95 00 0F	3	正常包，显示温度值3
4	FF FF 01 10 04 00 00 00 40 55 00 0F	4	包头错误，丢包，temp_value = 2,预期显示温度值3
5	FF FA 01 10 04 00 00 00 40 95 00 0F	5	正常包，显示温度值2
6	FF FA 01 10 04 00 00 40 40 94 00 0F	6	数据类型1错误，丢包，temp_value = 3,预期显示温度值2
7	FF FA 01 10 04 00 00 40 40 95 00 0F	7	正常包，显示温度值3
8	FF FA 01 00 04 00 00 00 40 45 00 0F	8	数据类型2错误，丢包，temp_value = 2,预期显示温度值3
9	FF FA 01 10 04 00 00 00 40 55 00 0F	9	正常包，显示温度值2
10	FF FA 01 10 00 00 00 40 40 91 00 0F	10	数据长度错误，丢包，temp_value = 3,预期显示温度值2
11	FF FA 01 10 04 00 00 40 40 95 00 0F	11	正常包，显示温度值3
12	FF FA 01 10 04 00 00 00 40 55 00 00	12	包尾错误，丢包，temp_value = 2,预期显示温度值3
13	FF FA 01 10 04 00 00 00 40 55 00 0F	13	正常包，显示温度值2
14	FF FF FA 01 10 04 00 00 40 40 95 00 0F	14	包头冗余，正常处理，temp_value = 3,预期显示温度值3
15	FF FA 01 10 04 00 00 40 40 95 00 0F	15	正常包，显示温度值3
16	FF FA 01 10 04 00 00 00 40 55 00 0F 0F	16	包尾冗余，正常处理，temp_value = 2,预期显示温度值2
17	FF FA 01 10 04 00 00 00 40 55 00 0F	17	正常包，显示温度值2

代码：

```
import Manu
def Test(arg,exp):
    # 先清理温感通道
    seekresult = seekresult=CH_232_温度传感器.Clear()
    # 按空格分割字符串为列表
    str = arg[0].split(' ')
    data = []
    for i in str:
        # 将16进制整型的字符串转化为整形表示
        # 将16进制字符串转化为10进制整型表示
        data.append(int(i, 16))
    # 通道写入数据包
    bool = bool=CH_232_温度传感器.Write(data)
    show = []
    show.append(exp[0])
    Manu.Check(show)

#def Test(arg,exp):
#    datas = arg[0].split(',')
#    I = []
#    for a in datas:
#        I.append(int(a,16))
#    seekresult = seekresult=CH_232_温度传感器.Clear()
#    bool = bool=CH_232_温度传感器.Write(I)
#
#    show = []
#    str = '预期结果: %s'%arg[1]
#    show.append(str)
#    Manu.Check(show)

Standard_Test(Test)
```

温控传感器输入接口测试结果：

温度控制器

操作

用户调节设定温度：

10

开始

暂停

关闭

显示值

采集到的温度：

2

加热棒输出电压：

10.1

散热风扇状态：

关

收发记录

	序号	用户设定温度	采集温度	输出电压	风扇状态	时间
▶	13	10	2	10.1	关	2022/10/20 15:12:51
	12	10	2	9.4	关	2022/10/20 15:12:47
	11	10	3	8.4500000000...	关	2022/10/20 15:12:45
	10	10	3	7.65	关	2022/10/20 15:11:29
	9	10	2	7.2	关	2022/10/20 15:11:26
	8	10	3	6.15	关	2022/10/20 15:11:20
	7	10	2	5.7	关	2022/10/20 15:11:11
	6	10	3	4.75	关	2022/10/20 15:10:56
	5	10	3	3.95	关	2022/10/20 15:10:51
	4	10	2	3.5	关	2022/10/20 15:10:49
	3	10	3	2.55	关	2022/10/20 15:10:41
	2	10	3	1.75	关	2022/10/20 15:10:24
	1	10	2	2	关	2022/10/20 15:10:23

具体分析见实验分析与总结

第五项（3.2）——控制加热棒输出接口

预设置的测试数据：无

代码：

```
#设定温度10℃
def Main():
    print 'Hello world'
    seekresult=CH_232_加热棒.Clear()
    seekresult=CH_232_温度传感器.Clear()
    Protocol_温度传感器.温度值.Value = 20      #不加热
    bool=Protocol_温度传感器.Write()
    Protocol_加热棒.BlockRead()
    if Protocol_加热棒.包头.Value != 0xFFFA:
        print '包头错误，错误的包头为%x'%Protocol_加热棒.包头.Value
    if Protocol_加热棒.数据类型1.Value != 0x02:
        print '执行数据错误，错误的数据类型1为%d'%Protocol_加热棒.数据类型1.Value
    if Protocol_加热棒.数据类型2.Value != 0x11:
        print '工作电机组错误，错误的数据类型2数据为%x'%Protocol_加热棒.数据类型2.Value
    if Protocol_加热棒.数据长度.Value != 4:
        print '数据长度错误，错误的长度值为%d'%Protocol_加热棒.数据长度.Value
    if Protocol_加热棒.检验.Checked != True:
        print '检验值错误，错误的校验值为%x'%Protocol_加热棒.检验.Value
    if Protocol_加热棒.包尾.Value !=0x0F:
        print '包尾错误，错误的包尾为%x'%Protocol_加热棒.包尾.Value
```

```

Protocol_温度传感器.温度值.Value = 5      #加热
bool=Protocol_温度传感器.Write()
Protocol_加热棒.BlockRead()
if Protocol_加热棒.包头.Value != 0xFFFA:
    print '包头错误，错误的包头为%x'%Protocol_加热棒.包头.Value
if Protocol_加热棒.数据类型1.Value != 0x02:
    print '执行数据错误，错误的数据类型1为%d'%Protocol_加热棒.数据类型1.Value
if Protocol_加热棒.数据类型2.Value != 0x11:
    print '工作电机组错误，错误的数据类型2数据为%x'%Protocol_加热棒.数据类型2.Value
if Protocol_加热棒.数据长度.Value != 4:
    print '数据长度错误，错误的长度值为%d'%Protocol_加热棒.数据长度.Value
if Protocol_加热棒.检验.Checked != True:
    print '检验值错误，错误的校验值为%x'%Protocol_加热棒.检验.Value
if Protocol_加热棒.包尾.Value !=0x0F:
    print '包尾错误，错误的包尾为%x'%Protocol_加热棒.包尾.Value

Main()

```

控制加热棒输出接口测试输出信息：



具体分析见实验分析与总结

第六项 (3.3) ——控制散热风扇输出接口

预设的测试数据：无

代码：

```

# 设定温度为10
def Main():
    seekresult=CH_232_温度传感器.Clear()
    seekresult=CH_232_散热风扇.Clear()
    # 写入初始状态，保证风扇不转动
    for i in [8, 8, 8]:
        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)

```

```

# 散热风扇阻塞获取数据包
Protocol_散热风扇.BlockRead()
if Protocol_散热风扇.包头.Value != 0xFFFA:
    print "包头错误, 错误的包头为: %x" % Protocol_温度传感器.包头.Value
if Protocol_散热风扇.数据类型1.Value != 0x02:
    print "数据类型1错误, 错误的数据类型为: %x" % Protocol_散热风扇.数据类型1.Value
if Protocol_散热风扇.数据类型2.Value != 0x22:
    print "数据类型2错误, 错误的数据类型为: %x" % Protocol_散热风扇.数据类型2.Value
if Protocol_散热风扇.数据长度.Value != 1:
    print "数据长度错误, 错误的数据长度为: %d" % Protocol_散热风扇.数据长度.Value
if Protocol_散热风扇.检验.Checked != True:
    print "检验和错误, 错误的检验值为: %x" % Protocol_散热风扇.检验.Value
if Protocol_温度传感器.包尾.Value != 0x0F:
    print "包尾错误, 错误的包尾为: %x" % Protocol_散热风扇.包尾.Value

print "-----"
# 写入初始状态, 保证散热风扇工作, 来判断数据包是否正确
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)
Protocol_散热风扇.BlockRead()
if Protocol_散热风扇.包头.Value != 0xFFFA:
    print "包头错误, 错误的包头为: %x" % Protocol_温度传感器.包头.Value
if Protocol_散热风扇.数据类型1.Value != 0x02:
    print "数据类型1错误, 错误的数据类型为: %x" % Protocol_散热风扇.数据类型1.Value
if Protocol_散热风扇.数据类型2.Value != 0x22:
    print "数据类型2错误, 错误的数据类型为: %x" % Protocol_散热风扇.数据类型2.Value
if Protocol_散热风扇.数据长度.Value != 1:
    print "数据长度错误, 错误的数据长度为: %d" % Protocol_散热风扇.数据长度.Value
if Protocol_散热风扇.检验.Checked != True:
    print "检验和错误, 错误的检验值为: %x" % Protocol_散热风扇.检验.Value
if Protocol_温度传感器.包尾.Value != 0x0F:
    print "包尾错误, 错误的包尾为: %x" % Protocol_散热风扇.包尾.Value

```

Main()

散热风扇输出接口测试输出信息：



散热风扇输出接口测试结果：



具体分析见实验分析与总结

第七项 (4.1) ——温控稳定性时间性能需求

预设设置的测试数据：无

代码：

```
# coding:utf-8
#设定温度为10℃，初始恒温箱外部温度值为3℃
def Main():
    # print 'Hello world'
    import time
    Tc = 10    #设定温度
    T0 = 0     #初始恒温箱外部温度值

    T = T0
    n = 0
    Fs = 0
    Va = 0    #上一次的输出电压
    time0 = time.time()
    while 1:
        seekresult=CH_232_温度传感器.Clear()
        seekresult=CH_232_加热棒.Clear()
        seekresult=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
        # 计算加热棒的加热量
        Qi = V*V *0.2
        if Protocol_散热风扇.操作指令.Value == 1:    #风扇打开
            Fs =2
        elif Protocol_散热风扇.操作指令.Value == 0:    #风扇关闭
            Fs = 0
        # 计算散热风扇的散热量
        Qo = 0.1*(T - T0) + Fs
        #Qo = Fs
        # 温度的变化量
        T = T+(Qi-Qo)    #下次输出温度值
        if abs(T - Tc) < 0.5:
            n=n+1
        else:
            n=0
        # 当超过10次时
        if n >=10:
            time1 = time.time()
            print '稳定时间为%d'%(time1-time0)
            if time1-time0 <=60:
                print '温控稳定时间性能测试合格'
                break
        # 当始终不能达到十次
        time2 = time.time()
        # 超时
        if time2-time0>60:
            print '温控稳定时间性能测试不合格'
            break
```

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

Main()

输出信息:

温度控制器

操作

用户调节设定温度: 10

开始 暂停 关闭

显示值

采集到的温度: 10.2825746536255

加热棒输出电压: 2.25859017372131

散热风扇状态: 关

收发记录

序号	用户设定温度	采集温度	输出电压	风扇状态	时间
20	10	11.243806838...	2.1110527038...	关	2022/10/20 16:47:17
19	10	11.401104927...	2.2167685985...	关	2022/10/20 16:47:16
18	10	11.450404167...	2.3406630039...	关	2022/10/20 16:47:15
17	10	11.362192153...	2.4742996215...	关	2022/10/20 16:47:14
16	10	11.115836143...	2.6057048320...	关	2022/10/20 16:47:13
15	10	10.698162078...	2.7271690368...	关	2022/10/20 16:47:12
14	10	10.170456886...	2.7791643619...	关	2022/10/20 16:47:11
13	10	9.2006902694...	3.0739515781...	关	2022/10/20 16:47:10
12	10	10.523455619...	2.9407309055...	开	2022/10/20 16:47:09
11	10	11.974706649...	2.9548434257...	开	2022/10/20 16:47:08
10	10	13.769252777...	2.8128088951...	开	2022/10/20 16:47:06
9	10	13.066020011...	3.1700437068...	关	2022/10/20 16:47:05
8	10	11.814266204...	3.4879649162...	关	2022/10/20 16:47:04
7	10	10.049827575...	3.7211701393...	关	2022/10/20 16:47:03
6	10	7.9209556579...	3.8216276645...	关	2022/10/20 16:47:02
5	10	5.6823954582...	3.7461980104...	关	2022/10/20 16:47:01
4	10	3.6493029594...	3.4626742601...	关	2022/10/20 16:47:00
3	10	2.08203125	2.9794921875	关	2022/10/20 16:46:59
2	10	1.25	2.1875	关	2022/10/20 16:46:58
1	10	0	2.5	关	2022/10/20 16:46:57

温度控制器

— □ ×

操作

用户调节设定温度：

10

开始

暂停

关闭

显示值

采集到的温度：10.2825746536255

加热棒输出电压：2.25859017372131

散热风扇状态：关

收发记录

序号	用户设定温度	采集温度	输出电压	风扇状态	时间
40	10	10.282574653...	2.2585901737...	关	2022/10/20 16:47:37
39	10	10.264925003...	2.2848876953125	关	2022/10/20 16:47:36
38	10	10.218851089...	2.3107992172...	关	2022/10/20 16:47:35
37	10	10.143930435...	2.3337888717...	关	2022/10/20 16:47:34
36	10	10.042594909...	2.3511646270...	关	2022/10/20 16:47:33
35	10	9.9204187393...	2.3603150844...	关	2022/10/20 16:47:32
34	10	9.7860641479...	2.3589845657...	关	2022/10/20 16:47:31
33	10	9.6508083343...	2.3455663681...	关	2022/10/20 16:47:30
32	10	9.5276784896...	2.3193729877...	关	2022/10/20 16:47:29
31	10	9.4302415847...	2.2808561801...	关	2022/10/20 16:47:28
30	10	9.3712396621...	2.2317318439...	关	2022/10/20 16:47:27
29	10	9.3612518310...	2.1749862670...	关	2022/10/20 16:47:26
28	10	9.4075746536...	2.1147509098...	关	2022/10/20 16:47:25
27	10	9.5134534835...	2.0560480594...	关	2022/10/20 16:47:24
26	10	9.6776685714...	2.0044352054...	关	2022/10/20 16:47:23
25	10	9.8944091796875	1.9655791282...	关	2022/10/20 16:47:22
24	10	10.153290748...	1.9447970867...	关	2022/10/20 16:47:21
23	10	10.439383506...	1.9465939044...	关	2022/10/20 16:47:20
22	10	10.733200073...	1.9742132186...	关	2022/10/20 16:47:19
21	10	11.010734558...	2.0292102813...	关	2022/10/20 16:47:18

时间性能对应的打印信息：

通用嵌入式系统测试平台 (ETest V5.5.3) - IO中心 - [用例服务端]

— □ ×

层叠 水平平铺 垂直平铺 图标 清空所有内容

错误 警告 消息 文本 清空显示

正在初始化Python脚本环境！

引入PythonLib路径：D:\\ETest\\IronPython-2.7.9.zip

引入PythonLib路径：D:\\ETest\\IronPython-2.7.9.zip/Lib

引入PythonLib路径：D:\\ETest\\IronPython-2.7.9.zip/DLLs

引入PythonLib路径：D:\\ETest\\IronPython-2.7.9.zip/Scripts

引入PythonLib路径：D:\\ETest\\PYSysPath

引入PythonLib路径：C:\\Users\\YiFei-Chu\\Documents\\ESITest\\PYUserPath

初始化Python脚本环境成功！

正在为Python脚本环境准备测试数据！

为Python脚本环境准备测试数据成功！

执行【测试用例】开始。。。。。

稳定时间为39

温控稳定时间性能测试合格

执行【测试用例】完成。。。。。

记录测试执行[结束信息].....

测试运行结束，即将退出！

四、实验分析与总结

输入温度为模拟温度传感器模块向待测件发送温度低于下限-20°的温度信息，观察在-20℃时是否做了截断处理		-21	-20	-21	模拟温度传感器模块向待测件发送温度低于下限-20°的温度信息时，待测件没有做下边界截断处理，界面显示更新为-21°，不符合需求2.2中对于温度范围(-20 °)—50 °的描述	
2.2						
2.4	模拟温度传感器模块向待测件连续发送多个连续的温度信息，观察待测件是否能在连续三次读取到输入温度高于当前温度时正确打开散热风扇运行	设定当前温度为20℃，连续三次输入24摄氏度，之后再连续输入三次20摄氏度	风扇状态变换顺序：关关开开开关	风扇状态变换顺序：关开开关关关	模拟温度传感器模块向待测件连续发送六个温度信息，发现待测件只要检测到2次当前温度大于设定温度+3，就开始转动，不符合需求2.4中第一条对于风扇开始转动条件的规定。	风扇的打开与关闭都不符合连续三次读取的要求

<p>输入温度为模拟温度传感器模块向待测件发送温度低于下限-20°的温度信息，观察在-20℃时是否做下边界截断处理</p>		<p>设定当前温度为20℃，连续三次输入24摄氏度，之后再连续输入三次20摄氏度</p>		<p>风扇状态变换顺序：关开开开关</p>		<p>模拟温度传感器模块向待测件发送温度低于下限-20°的温度信息时，待测件没有做下边界截断处理，界面显示更新为-21°，不符合需求2.2中对于温度范围(-20°~50°)的描述</p>	<p>风扇的打开与关闭都不符合连续三次读取的要求</p>
2.4	模拟温度传感器模块向待测件连续发送多个温度信息	2.2	观察待测件是否能在连续三次读取到输入温度低于当前温度时正确关闭散热风扇			<p>模拟温度传感器模块向待测件连续发送多个温度信息，发现待测件只要检测到1次当前温度小于等于设定温度，就停止转动，不符合需求2.4中第二条对于风扇停止转动条件的规定。</p>	
3.1	模拟温度传感器向待测件发送校验和错误的报文，观察是否能做丢包处理一个正常包(温度为2℃)，紧接着一个校验和错误的包(温度为3℃)	正常包：FF FA 01 10 04 00 00 00 40 55 00 0F 校验和错误的包：FF FA 01 10 04 00 00 40 40 D3 00 0F	2	3		<p>校验和错误时本应该进行丢包处理，但是却正常接收了校验和错误的包</p>	
3.1	模拟温度传感器向待测件发送校验和错误的报文，观察是否能做丢包处理一个正常包(温度为2℃)，紧接着一个数据类型1错误的包(温度为3℃)	正常包：FF FA 01 10 04 00 00 00 40 95 00 0F 数据类型1错误的包：FF FA 01 10 04 00 00 40 40 94 00 0F	2	3		<p>数据类型1错误时本应该进行丢包处理，但是却正常接收了数据类型1错的包</p>	
3.1	模拟温度传感器向待测件发送校验和错误的报文，观察是否能做丢包处理一个正常包(温度为2℃)，紧接着一个包头冗余的包(温度为3℃)	正常包：FF FA 01 10 04 00 00 00 40 55 00 0F 包头冗余的包：FF FF FA 01 10 04 00 00 40 40 95 00 0F	2	3		<p>包头冗余的包没有被丢弃</p>	

输入温度为模拟温度传感器模块向待测件发送温度低于下限-20°的温度信息，观察在-20°C时是否做下边界截断处理					模拟温度传感器模块向待测件发送温度低于下限-20°的温度信息时，待测件没有做下边界截断处理，界面显示更新为-21°，不符合需求2.2中对于温度范围(-20°)—50°的描述
3.1	模拟温度传感器向待测件发送校验和错误的报文，观察是否能做丢包处理	正常包：FF FA 01 10 04 00 00 40 40 95 00 0F 包尾冗余的包：FF FA 01 10 04 00 00 00 40 55 00 0F 0F	3	2	
2.2	模拟温度传感器向待测件发送校验和错误的报文，观察是否能做丢包处理	正常包：FF FA 01 10 04 00 00 40 40 95 00 0F 包尾冗余的包：FF FA 01 10 04 00 00 00 40 55 00 0F 0F	-20	-21	
3.2	模拟温度传感器向待测件发送校验和错误的报文，观察是否能做丢包处理。设定当前温度为10°C，设定温度传感器的温度为20°C，此为不加热的情况	当前温度：10 温度传感器温度：20	Protocol_加热棒.数据长度.Value = 4（其他输出正常）	Protocol_加热棒.数据长度.Value = 2（其他输出正常）	不加热时的数据长度输出错误
3.2	模拟温度传感器向待测件发送校验和错误的报文，观察是否能做丢包处理。设定当前温度为10°C，设定温度传感器的温度为5°C，此为加热的情况	当前温度：10 温度传感器温度：5	Protocol_加热棒.数据长度.Value = 4（其他输出正常）	Protocol_加热棒.数据长度.Value = 2（其他输出正常）	加热时的数据长度输出有误
3.2	模拟温度传感器向待测件发送校验和错误的报文，观察是否能做丢包处理。设定当前温度为10°C，设定温度传感器的温度为20°C，此为不加热的情况	当前温度：10 温度传感器温度：20	Protocol_加热棒.检验.Checked = True（其他输出正常）	Protocol_加热棒.检验.Checked = False（其他输出正常）	不加热时的检验和输出有误
3.2	模拟温度传感器向待测件发送校验和错误的报文，观察是否能做丢包处理。设定当前温度为10°C，设定温度传感器的温度为5°C，此为加热的情况	当前温度：10 温度传感器温度：5	Protocol_加热棒.检验.Checked = True（其他输出正常）	Protocol_加热棒.检验.Checked = False（其他输出正常）	加热时的检验和输出有误
3.3	模拟温度传感器向待测件发送温度数据，读取待测件输出的散热风扇控制数据，判断数据类型2字段是否正确。设定当前温度为10°C，初始状态风扇不转，散热风扇阻塞获取数据包	当前温度：10 风扇初始状态：不转	Protocol_散热风扇.数据类型2.Value = 0x22（其他输出正常）	Protocol_散热风扇.数据类型2.Value = 0x11（其他输出正常）	风扇阻塞时的输出的数据类型2有误

模拟温度传感器模块 向待测件发送温度低 于下限-20°的温度信 息时，待测件没有做 下边界截断处理，界 面显示更新为-21°， 不符合需求2.2中对 温度范围(-20°)—50° 的描述					
3.3	输入温度为模拟温度传感器模块向待测件发送温度低于下限-20°的温度信息，观察在-20°C时是否做下边界截断处理	当前温度：10 风扇初始状态：转动	Protocol_散热风扇.数据类型 2.Value = 20	Protocol_散热风扇.数据类型 2.Value = 21	
2.2	数据，判断数据类型2字段是否正确。设定当前温度为10°C，初始状态风扇不转，散热风扇阻塞获取数据包		0x22（其他输出正常）	0x11（其他输出正常）	
4.1	模拟整个温控的过程。计算温控稳定时间，判断温控温度时间是否小于1分钟	设定温度为0，室外温度为10	在1分钟之内到达温控稳定状态	无法达到温控稳定状态，保持在3度	温度控制器在设定温度为0，室外温度为10时，无法达到温控稳定状态，不符合需求4.1中温控稳定时间不大于1分钟的需求。

本次实验中，我们学习了使用ETest进行嵌入式测试，在此之前我对嵌入式测试一直没有什么具体的概念，也没有实际上手操作过。与开发者测试不同，嵌入式测试涉及到的软件更多，要求测试人员对硬件有一定的了解，并且在我的理解和体验中，感觉嵌入式测试主要就是使用黑盒测试、动态测试，同时结合一定的静态测试，而白盒测试难以使用和进行；二者也有很多相同点：都需要根据需求或者要求全面地设计测试用例，当然也都需要自己编写测试代码。