浙江工业大学

本科毕业设计（论文、创作）

题目：多源多点温控系统的设计与实现

作者姓名 马佳磊

指导教师 赵云波老师

专业班级 自动化 1402 班

学　　院 信息工程学院

提交日期 2018 年 6 月 1 日

Dissertation Submitted to Zhejiang University of Technology
for the Degree of Bachelor

Design and Implementation of Multi-source and Multi-point Temperature Control System

Student: Ma Jialei

Advisor: Professor Zhao Yunbo

College of Information Engineering
Zhejiang University of Technology

June 2018

浙江工业大学

本科生毕业设计(论文、创作)诚信承诺书

本人慎重承诺和声明：

1. 本人在毕业设计（论文、创作）撰写过程中，严格遵守学校有关规定，恪守学术规范，所呈交的毕业设计（论文、创作）是在指导教师指导下独立完成的；

2. 毕业设计（论文、创作）中无抄袭、剽窃或不正当引用他人学术观点、思想和学术成果，无虚构、篡改试验结果、统计资料、伪造数据和运算程序等情况；

3. 若有违反学术纪律的行为，本人愿意承担一切责任，并接受学校按有关规定给予的处理。

学生（签名）：马佳磊

2018 年 6 月 1 日

浙 江 工 业 大 学

本科生毕业设计（论文、创作）任务书

专 业 自动化 班 级 自动化 1402 学生姓名/学号 马佳磊/201403080419

一、设计（论文、创作）题目：

多源多点温控系统的设计与实现

二、主要任务与目标：

1. 提出一套适合多源多点温控系统的硬件设计方案；

2. 初步实现适应该系统的硬件平台；

3. 在该平台上实现某一算法。

三、主要内容与基本要求：

1. 阅读相关文献，了解该领域基本研究现状；
2. 研究针对特定系统的硬件实现方案；
3. 搭建并测试相关实现方案；
4. 撰写毕业论文。

四、计划进度：

2018 开学前 收集相关资料文献，学习相关知识，完成外文翻译、文献综述；熟悉课题，做好开题准备。 第 1-3 周 完成开题报告，参加开题交流 第 4-8 周 完成平台搭建和算法验证等工作，接受中期检查 第 9-14 周 在平台基础上进行算法设计和实现等工作。撰写毕业论文初稿 第 15 周 论文修改，毕业答辩，提交相关文档资料

五、主要参考文献：

1. 《自然通风动态温度调控的智能气动开窗系统》， 司慧萍，浙江大学，博士论文，2015 2. 《智能温度控制系统》，刘美霞，南京航空航天大学，硕士论文，2003 3. 《工业无线温度控制系统设计》，马俊卿，东华大学，硕士论文，2017

任务书下发日期 2017 年 12 月 20 日

设计（论文、创作）工作自 2017 年 12 月 20 日 至 2018 年 6 月 4 日

设计（论文、创作）指导教师 赵云波

系主任（专业负责人） 杨马英

主管院长 张有兵

多源多点温控系统的设计与实现

摘　要

随着人们生活水平的提高和温度控制算法的进步，大家对于舒适的生活环境的需求越来越强烈。由于每个人对于环境温度的感知不同，所以每个人感觉到的最舒适的温度也会不同，如何让每个人都感觉到舒适，同时又能兼顾到节能，多源多点温控系统的概念应运而生。

本文的研究需要完成的内容是，多源多点温度控制系统的硬件平台的搭建。通过传感器感应到周围温度，Arduino 程序将数据发送到串口缓冲区中，然后上位机从串口缓冲区中读取数据，并显示在界面上；将设定温度发送到串口缓冲区中，硬件平台读取到数据后，并给上位机发送 receive，表示已经收到设定温度。由于条件和个人能力的限制，没有实现硬件平台对空调的控制，所以这里还对温差和空调功率的关系进行了调研。

本文研究的主要工作内容如下：

1.综述课题研究背景与意义，以及温度控制理论的发展过程中涉及到的几种控制算法，介绍了我国暖通空调自动控制系统的发展。

2.介绍了硬件平台的搭建过程，包括原理图的设计、PCB 板的绘制和元器件的选择等，简述了 Arduino 温度采集程序的原理。

3.对温差（即设定温度与实际温度的差值）与空调功率的关系进行调研，分为定频空调部分和变频空调部分。

4.介绍温度控制界面的设计，包括接收测温数据和发送设定温度数据。

关键词：多源多点、温度控制、Arduino、空调、程序界面

DESIGN AND IMPLEMENTATION OF MULTI-SOURCE AND MULTI-POINT TEMPERATURE CONTROL SYSTEM

ABSTRACT

With the improvement of people's living standards and the advancement of temperature control algorithms, the demand for comfortable living environments has become stronger and stronger. Because everyone has a different perception of the ambient temperature, the most comfortable temperature that each person feels will be different. How to make everyone feel comfortable and at the same time take into account the energy saving, the concept of multi-source and multi-point temperature control system produced.

The research content of this paper is that the hardware platform of multi-source and multi-point temperature control system is built. The sensor senses the ambient temperature. The Arduino program sends the data to the serial buffer. The host computer then reads the data from the serial buffer and displays it on the interface. The set temperature is sent to the serial buffer and the hardware platform reads it. After receiving the data, it sends a “receive” to the host computer to indicate that the set temperature has been received. Due to the limitations of the conditions and individual capabilities, the control of the air conditioner on the hardware platform was not implemented, so the relationship between the temperature difference and the air conditioning power was investigated here.

The main work of this paper is as follows:

1. Overview The background and significance of the research, as well as several control algorithms involved in the development of temperature control theory, introduced the development of our country's HVAC automatic control system.

2. Introduced the construction process of the hardware platform, including the design of the schematic diagram, the drawing of the PCB board and the selection of the components. The principle of the Arduino temperature acquisition program was briefly described.

3. The relationship between the temperature difference (ie, the difference between the set temperature and the actual temperature) and the air conditioning power is investigated and divided into fixed frequency air conditioning and inverter air conditioning.

4. Introduce the design of temperature control interface, including receiving temperature measurement data and sending set temperature data.

Keywords: multi-source and multi-point, temperature control, Arduino,air conditioning, program interface

目　　录

摘要 ..... **I**
ABSTRACT ..... **II**
第 1 章 绪论 **1**
1. 1 课题研究背景及意义 1
1.2 温度控制理论的发展 1
1.2.1 PID 控制 1
1.2.2 神经网络控制 3
1.2.3 模糊控制 4
1.2.4 遗传算法 5
1.2.5 广义预测控制 6
1.3 我国暖通空调自动控制系统的发展 7
1.3.1 设备集中化启停控制系统 7
1.3.2 模拟仪表控制系统 7
1.3.3 集散式监控系统 7
1.3.4 直接数字控制系统 8
1.4 空调运行与温度调节关系的调研 8
1.4.1 压缩机的功率对温度调节的影响 8
1.4.2 空调风速对温度调节的影响 9
1.4.3 空调风向对温度调节的影响 9
1.5 本文研究内容 11
1.6 本章小结 11
第 2 章 硬件平台的搭建 **13**
2.1 硬件平台搭建的目的 13
2.2 原理图的设计 13
2.2.1 Arduino 的简单介绍 13
2.2.2 原理图设计的过程 14
2.3 PCB 板的绘制 15
2.4 元器件的选取 16
2.4.1 Arduino 开发板的选取 17
2.4.2 温度传感器的选取 18

2.5 Arduino 温度采集程序 18
2.6 本章小结 19
第 3 章 程序界面设计 21
3.1 程序设计的目的 21
3.2 程序设计的过程 21
3.3 程序运行流程图 22
3.4 程序测试 24
3.4.1 程序与串口助手的测试 24
3.4.2 程序与程序的测试 25
3.5 程序运行的结果 25
3.6 本章小结 26
第 4 章 总结与展望 27
4.1 毕设工作总结 27
4.2 未来展望 27
参考文献 29
附录一 31
附录二 35
致谢 50

第 1 章 绪 论

1.1 课题研究背景及意义

温度是构成我们整个物质世界的重要因素之一，高温能融化金属，低温能冰冻物质，而适宜的温度又能让我们感到非常舒服，因而温度对于我们的意义十分重大，与我们的生活、生产密切相关。在一定的空间内，有计划地通过调节一些冷却或者加热设备的输出，使空间内的温度达到人们所需要的温度，这就是所谓的温度控制。温度控制在农业大棚养殖、工业冶金、产品存放和满足人的舒适度等方面都有着广泛的应用。本文单单就满足人的舒适度方面，对多源多点的温度控制方法进行阐述。

在一个密闭空间中，其中只有较少的人且集中在一处，则只需要满足这一处位置的温度要求，其他位置的温度可以不用去管。可以将一个温度传感器置于该处，将该位置的温度数据发送给上位机，通过调节空调风机的运行方式，使得该位置的温度达到设定值。这就是“源”和“点”分离的温度控制需求。

在一个密闭的大型空间内比如商场或者办公室，在其中各个位置分布着温度传感器，可能不同的区域有不同的温度需求，它们将感应到的温度都传给上位机进行处理，通过调节多台空调风机的运行方式，来实现满足整个大型空间内有不同温度分布的要求。这就是多源多点的温度需求。

将温度调节到合适的位置能够提高我们的舒适度，而合理的调节方法又能够减少能量的损耗，因此运用好的温度控制算法和设计出好的温控系统具有极强的现实意义。

1.2 温度控制理论的发展

近几年温度控制理论发展迅速，PID 控制、神经网络控制、模糊控制和广义预测控制等理论被广泛应用于各行各业的温度控制中。

1.2.1 PID 控制

在控制过程中，为了控制系统的输出，从而对由反馈值与设定值比较所得偏

差的比例（P）参数、积分（I）参数以及微分（D）参数进行调节，叫做 PID 控制。PID 控制中的三个参数的调节对于系统性能的影响各不相同。

在模拟控制系统中，PID 控制规律的表达式为：

$$u(t) = K_p[e(t) + \frac{1}{T_i}\int_0^t e(t) + T_D\frac{de(t)}{dt}] \quad (1\text{-}1)$$

工作原理如下图所示：

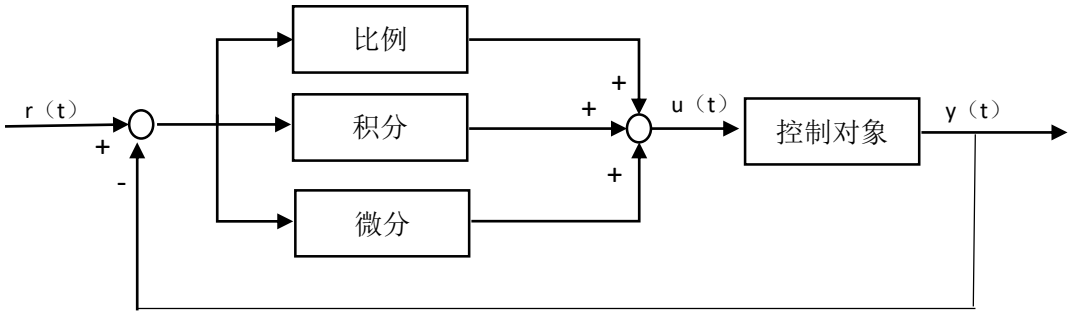

图 1-1　PID 工作原理图

（1）调节比例系数 Kp 的值会影响系统的功能。

为了实现提高系统的调节速度，同时降低稳态误差的目标，发以通过增加比例系数 Kp 实现。当比例系数 Kp 增大时，可以更好地降低甚至消除静态误差，但如果将比例系数 Kp 的值增加得过大时，反而会增大系统的超调量，甚至会使系统无法达到一个稳定的值，从而出现震荡现象。当出现系统的调节速度太过缓慢的情况，这时候应该是比例系数 Kp 的值偏小的原因，因此可以通过增大比例系数 Kp 的值来调节。

（2）调节积分时间 Ti 的值也会影响系统的功能。

为了使系统的稳态误差被去除掉，以及使得系统的控制精度能够获得提升，可以对系统增加积分作用。当积分时间 Ti 太小时，这样会使积分作用对系统的调节作用过于强烈，从而会增大系统的超调量，甚至会使系统无法达到一个稳定的值，进而出现震荡现象。

（3）调节微分时间 Td 的值同样会影响系统的功能。

微分作用可以使得系统的动态特性得到改善，以及使得系统的调节速度得到

提高，对由反馈值与设定值比较所得到的偏差量的变动快速做出反应，按偏差倾向进行调节控制，降低了系统的调节过程的时间和动态偏差量。但当微分时间 Td 太大时，又会使调节作用过于强烈，从而会使系统出现超调，甚至会使系统无法达到一个稳定的值，进而出现震荡现象。

因此，完美的控制效果就是，开始时比例为主，积分为次，微分适时制动，随着距离设定温度越来越近，比例分量为 0，积分分量正好等于设定温度下的热流失速度，从而温度保持不变[1]。PID 控制可以分为线性的和非线性的，在工业控制的大领域里，非线性的 PID 控制是一种得到广泛业界认可，并且历史及其悠久，效果显著的控制方式[2]，非线性的 PID 的主要特点便是结构简单、易于操作调整并且具有一定的鲁棒性[3]。

人们把专家系统、模糊控制、神经网络等理论，整合到 PID 控制器中。这样，既保持了 PID 控制器的结构简单、适用性强和整定方便等优点，又通过智能技术在线调整 PID 控制器的参数，以适应被控对象特性的变化[4]。

1.2.2 神经网络控制

由于一些环境系统具有强非线性、多变量、强耦合以及抗干扰能力较差的特性，且所在的环境存在大量不确定因素，难以建立其精确地数学模型，因此传统的 PID 控制方式已难以满足高品质温度控制系统需求[5]。而结合神经网络控制能够有效地弥补传统 PID 控制的不足。

由大量人工神经元作为处理单元，构成普遍互联的网络，叫做神经网络，这是现在乃至以后都十分重要的人工智能技术之一。神经网络控制是一种比较新型的控制理论方法，它基于神经网络，可以较好地处理非线性、不确定系统的相关控制问题，同时还不用去获得被控对象的精确的数字模型。工作原理如下图所示:

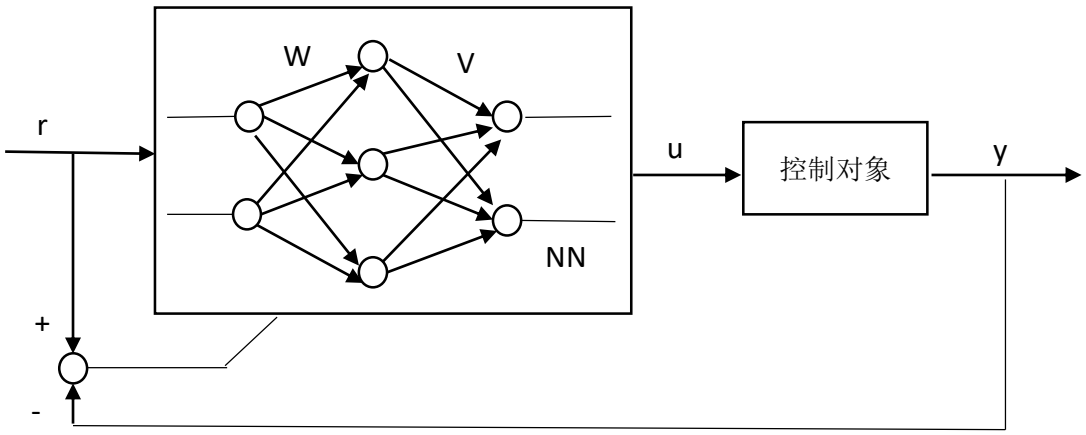

图 1-2　神经网络控制原理图

相比与那些传统控制理论的方法，神经网络控制方法具备很多更为优越的特点性质:

（1）非线性。在非线性系统中应用神经网络控制方法，反映出很好的拟合性，通过理论可以证明出多层神经网络能够拟合任意一种函数[6];

（2）并行的结构和处理的特性。神经网络具有容错能力大和数据处理能力强的特点，这是因为神经网络的相应结构和具有的实现功能都是并行的;

（3）学习成长性和自适应调节性。神经网络可以对所提供的信息完成学习和记忆;

（4）多变量处理的特性。神经网络能够非常好地应用于多变量系统，在多输入信号同时具有多输出的情况下，表现出十分好的效果。

1.2.3 模糊控制

所谓的模糊控制，只是在控制方法上应用了模糊数学知识，其基本原理仍和经典控制、现代控制理论一样没有改变[5]。模糊逻辑控制是一种模拟人思维的控制方法[7]，它主要适用于非线性且系统干扰较大的系统，并且它的控制不依赖于精确地数学模型[8]。事物与事物之间联系信息存在着一定的模糊性，但是这样的信息相比于准确的信息往往拥有的信息量更大、内容更丰富。对于很多较为复杂的系统，越是强调准确性，越是难以控制，这时候，模糊控制反而能够实现很好

地控制。工作原理如下图所示：

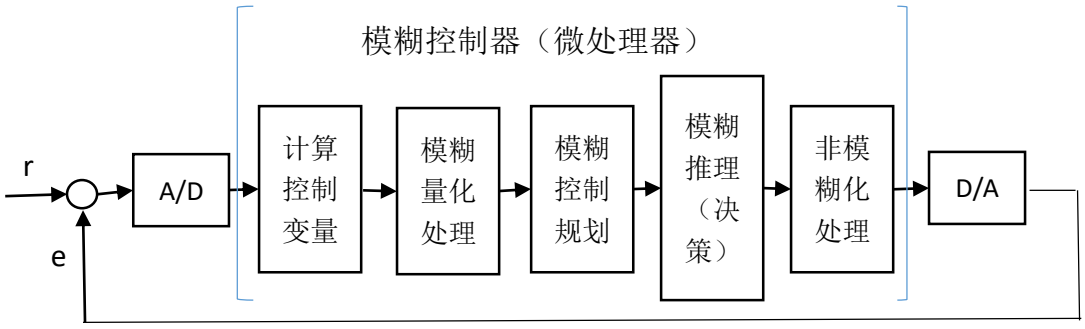

图 1-3　模糊控制工作原理图

模糊控制具有以下优点:

a. 响应速度快；

b. 模糊控制方法是不会随着参数的改变而变化，对具有时变与滞后特性的对象能够有很好的控制成效，同时也具备一定的抗干扰能力；

c.以规则为基础，模糊控制在运用于很复杂的控制系统时，可以不用建立精确数字模型，但是依旧具有很好的控制效果，同时该算法的控制理论和方法很容易被人理解和应用；

d.在诸多控制规则之间，模糊控制可以很好地切换，控制性能好；

e.模糊控制算法能够仿真人工操作控制流程，具有智能化的特点[9]。

然而，模糊控制算法自身还有着一些缺陷:如果提供的信息太过简单，模糊控制反倒会降低系统的控制精度；自适应能力较差；模糊控制无法对控制对象进行设定；模糊控制的理论还不够完善。

1.2.4 遗传算法

遗传算法是有 Holland 教授于 1975 年提出的，并在他的著作《Adaptation in Natural and Artificial System》[10]中对该算法进行了叙述。遗传算法，正如其名称一样，自然遗传过程中可能会发生的各种现象进行仿真，在每次迭代过程中按照要求择优选取进行筛选，通过选用遗传算子完成各种组合，进行繁衍得到下一代，如此循环，直到达到某个收敛的目标为止。该算法最显著的特点就是对相关对象

直接完成计算，不需要进行一些推导;覆盖的范围比较大，可以对多个解完成评算;通过采用一定的寻优方法，可以自主地调节寻优的方向，这样更容易获得准确的最优解。目前，在遗传算法的研究中，对于各种计算方法的优化研究在逐渐减少，同时在工业生产中的应用显著增加，这反映了该算法的理论研究已经开始日渐成熟，转而将重点开始偏向实际应用中。遗传算法应用于温度控制系统可以获得更好地控制效果，而这也正是现代自动化工业发展的重要方向之一[11]。

基于遗传算法的自适应 PID 控制的原理框图如下图所示：

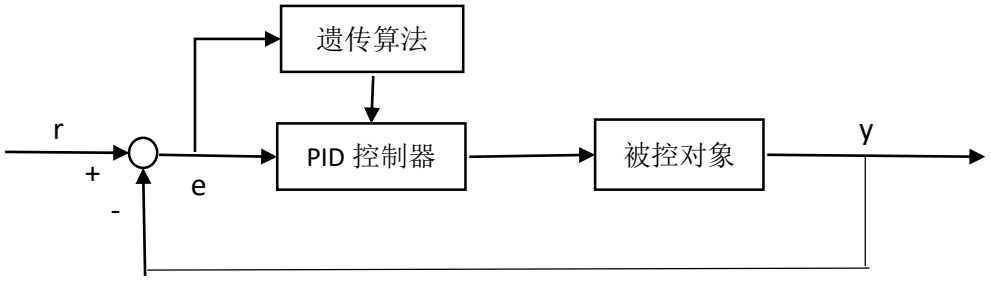

图 1-4　基于遗传算法的自适应 PID 控制原理框图

1.2.5 广义预测控制

广义预测控制（GPC）[12]是自适应调节控制研究开发的预测控制算法。各种最小方差控制器在已知对象估计不精确的情况下，控制的准确性会变得很差。由于当时的算法对于模型的准确度要求特别高，这就使得这些算法难以应用于较为复杂的工业控制过程中。人们迫切想要可以得到一种鲁棒性强同时数字模型要求较低的自适应控制算法。在这个背景下，1987 年，Clarke 等人研究出了对于当时乃至现在的工业控制都非常重要的算法，那就是广义预测控制算法。GPC 基于特定的参数模型，引入了不相等的预测水平和控制水平，系统设计灵活方便，具有预测模型、滚动优化和在线反馈校正等特征，表现出优良的控制性能和鲁棒性，被广泛地应用于工业过程控制，取得了明显的经济效益[13]。目前 GPC 算法的理论分析还相当不足,稳定性和鲁棒性的剖析，大多依托于计算机的模拟仿真和实践应用中的调节控制,广义预测控制系统的分析十分复杂，这是因为 GPC 算法引入了柔化作用和多步预测以及其自身的特殊性。

1.3.我国暖通空调自动控制系统的发展

暖通空调系统，即 HVAC，是现代房屋建筑中不可缺少的的室内环境调节系统[7]。暖通空调的功能主要是室内空气调节、室内温度控制以及通风这三种。在我国，暖通空调的自动调节控制系统可以用四个时期相应的典型系统来概括其发展。

1.3.1 设备集中化启停控制系统

这时候还是人工手动控制，只是将设备的控制开关（比如开关、继电器、接触器等）集中到一个地方，这样便于管理和集中控制，而管理人员根据建筑物环境的变化，进行合理调节开关，以满足建筑物的环境要求。

1.3.2 模拟仪表控制系统

该系统是一个十分经典的负反馈控制系统，通过温度传感器测量出环境温度的值，而后将该值与之前设定的给定值相比，如果低于给定值，则通过控制器调节执行器来提高环境温度，反之则降低环境温度，最终达到将温度调节到设定温度的目标。该系统中最关键的就是控制器。常见的模拟控制器有：双位控制器（On/Off 控制）、比例积分控制(PI 控制)和比例积分微分控制器（PID 控制）[14]。前期研究出来的模拟控制器的相关参数，包括比例参数、积分时间、微分时间等，是确定的同时更是不能够改变的，不能依据不同用户的特定要求来改变。随着技术的发展，模拟控制器的特性参数能够根据需要进行调整。

1.3.3 集散式监控系统

在该系统中，参数的调节控制仍然使用的是模拟控制仪表，模拟信号通过模数转换得到数字信号，将数字信号集中传输到计算机上，即数字信息传输技术，从而代替了传统模拟仪表控制系统的参数传递，实现对各种设备和参数的统一集中监管测量并进行显示输出。该系统钻研到后期，可以使用计算机对一些控制参数实现再次的设定。

1.3.4 直接数字控制系统

在直接数字控制系统中，经过模拟量输入通道（AI）和开关量输入通道（DI），计算机获得实时数据，而后依据某种特定的规律进行计算操作，最终输出控制信号，同时因为是由模拟量输出通道(AO)和开关量输出通道（DO）控制直接调节生产过程，因此该系统是一个闭环控制系统。直接数字控制系统的控制器是直接数字控制器，替代了传统的模拟控制仪表。当控制内容及规模相同时，总成本大为降低，其性能和灵活性又是传统控制器望沉莫及的[15]。DDC 控制器的控制原理示意图如下图所示：

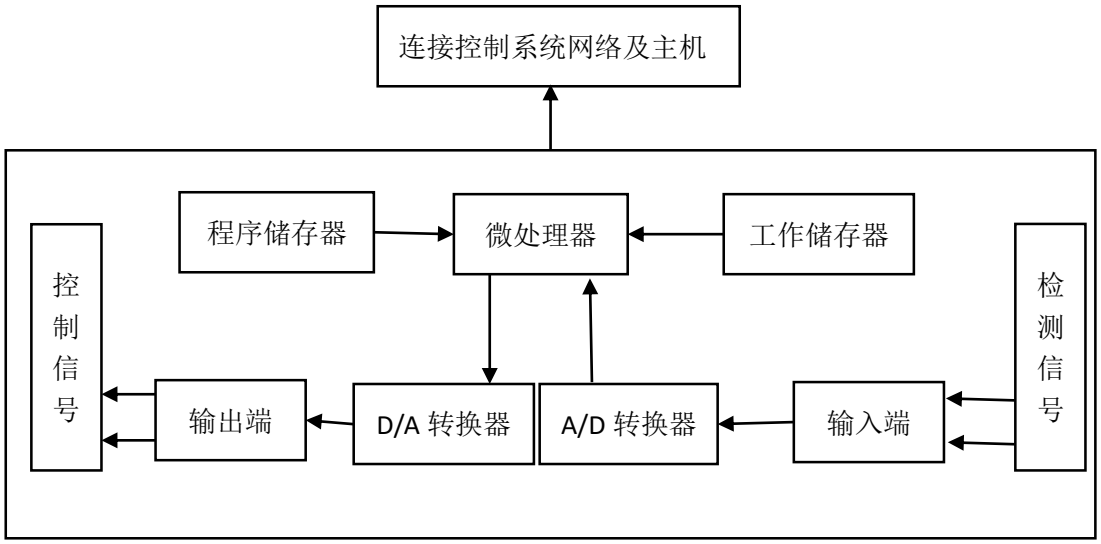

图 1-5　DDC 控制器控制原理示意图

空气调节器，简称空调，通过人工方法，改变一个密闭空间内的温度、湿度、风速和洁净度。空调中一般包括冷源设备、热源设备、冷热介质输配系统，末端装置和一些辅助设备等。本项研究中，主要涉及到空调的温度调节。

1.4 空调运行与温度调节关系的调研

1.4.1 压缩机的功率对温度调节的影响

（1）定频空调，又称为定速空调，因为空调的供电频率不能改变，则压缩机的转速也基本不变，只能通过控制压缩机的启、停来调节室内的温度。所以空

调的功率与设定温度没有多大关系。在制冷的情况下，当设定温度低于实际温度超出 1℃时，则压缩机处于满功率运行；当设定温度高于实际温度超出 1℃时，则压缩机停止工作。而在加热的情况下，则与制冷情况的调节方法相反。

（2）变频空调，就是在常规空调的结构上增加了一个变频器，通过改变空调的供电频率，来改变压缩机的转速，压缩机的转速直接影响到空调的使用效率。变频空调的基本结构以及调节温度的原理和常规空调基本相同。变频空调的温度控制精度高，变频空调是通过控制压缩机的转动速度来实现对于温度的控制，精确度可以达到一度左右，对于温度的控制十分精确。变频空调可以保持室温恒定，变频空调通过变频压缩机控制转速来自动地调节室温，使室温保持恒定。[16]

变频空调的控制算法除了 PID 控制之外，还有矩阵电子控制、系统-相关指令控制和模糊控制算法等。

在 PID 控制中，存在两种作用，即正作用和负作用。负作用是当偏差信号为正时，增加频率输出，当偏差信号为负时，则频率输出降低。正作用则刚好相反。在空调的温度调节中，用到的是负作用调节，根据温度偏差进行调节，温度偏差较大时，空调的功率较大，温度偏差较小时，空调的功率较小。

1.4.2 空调风速对温度调节的影响

空调的风量大，房间内的空气流动就越剧烈，这样在空调加热的情况下，热空气就遍布房间的各个角落，使得房间里各个地方都是热的。但是这样有个缺点就是增加了耗电量，一方面是空调的风机的功率较大，另一方面是空调的加热时间增加，其实只要人呆的地方热就行了，其他地方的温度可以不用管。空调的风速太小时，热空气会堆积在空调附近，这样会使得人呆的地方还没有达到设定温度，而空调附近已经达到了，空调就停止工作了。同样的，在空调制冷的时候也是一样的情况。所以将空调的风速调在中风比较合适。

1.4.3 空调风向对温度调节的影响

从物理特性上来分析，较热的空气要比冷空气轻的，由于空气对流方式，会导致室内温度的分布是上热下冷的。[17]在使用空调制冷时，应将空调的风向调整至上方，由于空气对流，使得室内的空气温度分布均匀，达到快速降温的目的，

如果将风向朝下，空调的冷气就直接在下方了，很难影响到上方的热空气，制冷效果会大大下降。反之，在用空调加热升温的时候，应将空调的风向调整至下方。

空调的风向的角度不同，对于室内温度的影响也是不同的。下面几个表格是空调在加热情况下，不同角度的风向时的温度。

风向为 45°时：

表 1-1　风向为 45°时的温度测量表[18]

测量点	最高温度℃	最低温度℃	平均温度℃
高	24.24	23.94	24.09
中	24.28	23.74	24.01
低	24.04	23.68	23.86

风向为 60°时：

表 1-2　风向为 60°时的温度测量表[18]

测量点	最高温度℃	最低温度℃	平均温度℃
高	25.11	24.46	24.79
中	25.14	24.42	24.78
低	25.10	24.18	24.64

风向为 90°时：

表 1-3　风向为 90°时的温度测量表[18]

测量点	最高温度℃	最低温度℃	平均温度℃
高	27.80	26.84	27.32
中	25.52	24.75	25.14
低	22.35	22.17	22.26

风向为 120°时：

表 1-4　风向为 120°时的温度测量表[18]

测量点	最高温度℃	最低温度℃	平均温度℃
高	29.39	27.19	28.29
中	26.08	24.92	25.50
低	23.36	22.90	23.13

如上述实验表格可知，空调在加热情况下，风向为 90°和 120°，高处的最高温度均超出设定温度较多，同时高处的平均温度要比低处的平均温度相差 5℃。风向在 60°时，高处的平均温度与低处的平均温度的温差相差较小，且高中低三处的温度都与设定温度相近。所以风向在 60°时，空调加热的效率最高。

1.5 本文研究内容

本论文依据每个人对舒适温度的要求不同，提出了多源多点的温度控制方法。通过对原理图的设计和元器件的选取，完成了硬件平台的搭建，调研和界面的设计更加完善了温度控制系统。主要章节内容如下：

第一章是总体上综述课题研究背景与意义，介绍了温度控制理论的发展，其中着重挑选了 PID 控制、神经网络控制、模糊控制、遗传算法和广义预测控制这五个温度控制理论进行详细介绍。还介绍了我国暖通空调自动控制系统的发展。

第二章是阐述了硬件平台的搭建过程，包括原理图的设计、PCB 板的绘制和元器件的选取。在 Arduino 软件上，完成温度采集程序的编写。

第三章对温差与空调的功率之间的关系进行调研，分定频空调和变频空调，分别调研。

第四章介绍在对温度控制界面设计时的想法，和界面工作运行的原理以及界面实现的功能。

第五章是对本次毕业设计所做的工作进行总结，以及对未来温度控制的发展的展望。

1.6 本章小结

本章内容在查阅了温度控制的相关文献的基础上，阐述了课题研究的背景和意义，介绍了温度控制理论的发展，包括 PID 控制、神经网络控制、模糊控制、遗传算法和广义预测控制。同时还介绍了暖通空调自动控制系统的发展，包括设备集中启停控制系统、模拟仪表控制系统、集散式监控系统和直接数字控制系统。做了相关调研，研究空调运行与温度控制的关系，即压缩机功率、风速和风向对温度调节的影响。压缩机功率的影响，定频空调与变频空调两者是不一样的。定

频空调，温差对压缩机功率没有影响；而变频空调，温差对空调功率的影响主要体现在算法对空调功率的控制，不同的算法调节作用是不一样的。风速的影响，风速太大，增加耗电量；风速太小，房间内的实际温度没有达到设定值。风向的影响，在加热的情况下，风向在 60°时，空调的加热效率最高。还介绍了本文的主要研究内容。

第 2 章 硬件平台的搭建

2.1 硬件平台搭建的目的

本文研究是要搭建出多个硬件平台，该硬件平台能够采集到周围的环境温度，在平台的 LED 屏上显示出来，并将温度数据发送到串口，上位机读取串口缓冲区中的数据，从而获得房间内各个点的温度数据。上位机通过算法来控制空调的功率，从而达到有效调节温度的目的，最终使得房间内的各个点的温度都达到对应的设定温度。由于实验室的中央空调无法通过红外线控制，同时控制开关是触屏的，无法拆卸，所以控制空调的模块就没有设计，本次研究的硬件平台最终设计了温度采集模块，将采集到的温度数据发送到串口缓冲区中，并在 LED 屏上显示出来。

2.2.原理图的设计

2.2.1 Arduino 的简单介绍

Arduino 是一款基于 AVR 单片机系统，软硬件系统都具有高度模块化的开源电子产品开发平台[19]。Arduino 建构于简易输出/输入（simple I/O）接口板，其内部封装了各种常用的开发库和常用传感器测量的库函数供开发者调用，开发者只需着眼于程序的逻辑结构而无需了解底层函数设计，大大简化了传感器测量系统的设计[20]。

Arduino 被设计出来的历史比较有趣。由于一些学生抱怨找不到便宜好用的微控制器，则 Massimo Banzi 和 David Cuartielles 设计电路板，David Mellis 为开发板设计编程语言，最后仅用了几天时间就设计出了 Arduino。他们将设计图放到了网上，并保持设计的开放源代码理念。在本项研究中，使用 Arduino 作为处理器的原因，是因为 Arduino 具有如下特点：

（1） Arduino 电路图的设计，被放到了网上，供大家下载，还向大家开放了源代码，实现了知识共享；

（2） 可简单地与感测器等电子组件连接，如红外线、热敏电阻等；

（3） 支持多样的交互程序，如 Aobe Flash、Max/MSP、VVVV、C 等；

（4） 可依据 Arduino 官方网站，获取硬件的设计档，加以调整电路板及组件，以匹配自己实际设计的需求；

（5） 使用低价格的微处理控制器；

（6） USB 接口，不需外接电源。

（7） Arduino IDE 能够跨平台，在 windows、Linux 和 Macintosh OSX 这三个常用的操作系统上运行，这相比于其他大多数控制器只能在 windows 环境下进行开发，具有更为广泛的应用。

（8） Arduino 的开发方式简单，用户更容易上手，让开发者更注重创意和实现，大大减少学习的时间和开发的周期。

在使用 Arduino 时与其他微处理器的比较：

（1） Arduino 的处理速度相比较而言比较慢，且实时性不高。

（2） 使用 Arduino 时需要更注意时间片格、优先级、抢占机制和线程通讯等，所以在多任务、多线程编程时，就会显得比较麻烦。

（3） 因为 Arduino 片上没有太多的集成，所以很多东西都要外扩。

（4） PWM 输出和模拟输入口比较多，但是 PWM 的频率最大是 500Hz，对于有些电机的控制是不符合要求的。

2.2.2 原理图设计的过程

在设计的时候要考虑这个电路原理图要实现的功能，通过功能需求来指导要在原理图上添加的元件。在本次研究中，需要实现的功能如下：

（1）有一个微处理器，运行程序；

（2）能够实现无线串口通信，将采集到的温度数据传给上位机；

（3）电路需要一个 USB 接口通电源供电；

（4）通过温度传感器，采集空气中的温度；

（5）有一个 LED 屏幕，将采集到的温度数据显示在屏幕上；

（6）设计按键，通过按键来输入设定温度的值。

综上所述，需要在原理图上添加 Arduino、温度传感器、USB 接口、无线串口收发模块和按键等元件。

创建一个原理图，在元件库中找到需要添加的元件放置在原理图中，了解各

个元件要实现相应的功能需要连接的线及其接法，然后对元件进行连线，对元件关联对应的封装模型，然后 compile,检查出原理图中的错误，并改正。

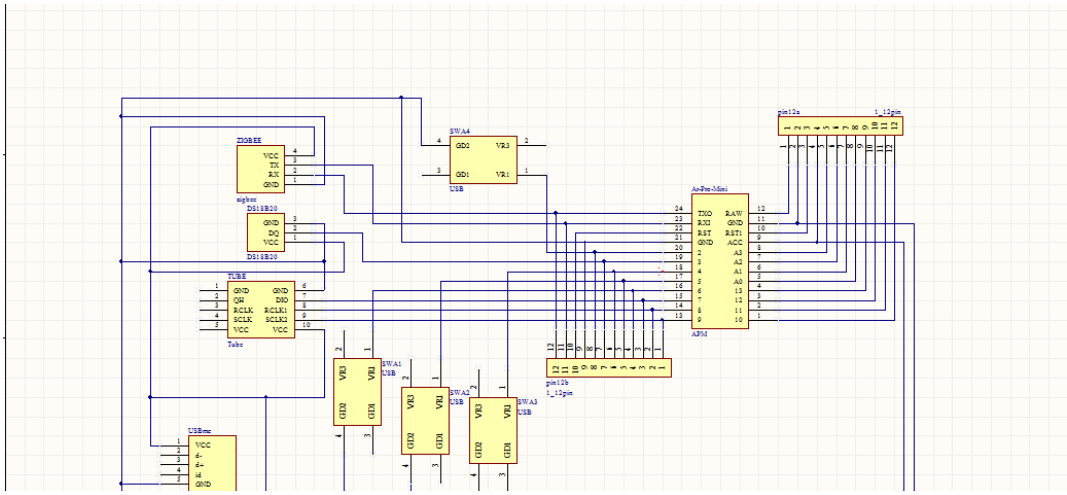

图 2-1　电路原理图

2.3.PCB 板的绘制

PCB 板的绘制过程如下所示：

（1）利用原理图设计工具设计出原理图，并生成对应的网络表。当电路板比较简单同时已经有网络表的情况下，可以不用画原理图，直接进入 PCB 板的绘制界面，然后直接去用零件封装，人工生成网络表。手工更改网络表，将原理图上和 PCB 封装库中引脚名称不一样的，都改成 PCB 封装库的名称。

（2）根据需求，可以自己画出需要的 PCB 封装器件，并放入自己建立的 PCB 专用库设计文件中。

（3） 设置 PCB 设计环境，规划电路板，确定电路板的边框。

（4） 打开 PCB 库文件，调入生成的网络表和修改零件封装。

（5） 零件布局，可选择进行自动布局，也可选择手动布局。

（6） 元件经过调整之后，将全部元件锁定。

（7）设置布线规则，包括安全间距、走线层面和方向、过孔形状、走线线宽和敷铜连接形状的设置等。

（8）自动布线，然后对布线进行手工调整。

（9）切换到单层显示模式下，并将每个布线层单独打印出来，方便之后改线时参考。最后取消单层显示模式，并存盘。

（10）对所有过孔和焊盘补泪滴，然后放置覆铜区，最后再做一次 DRC。

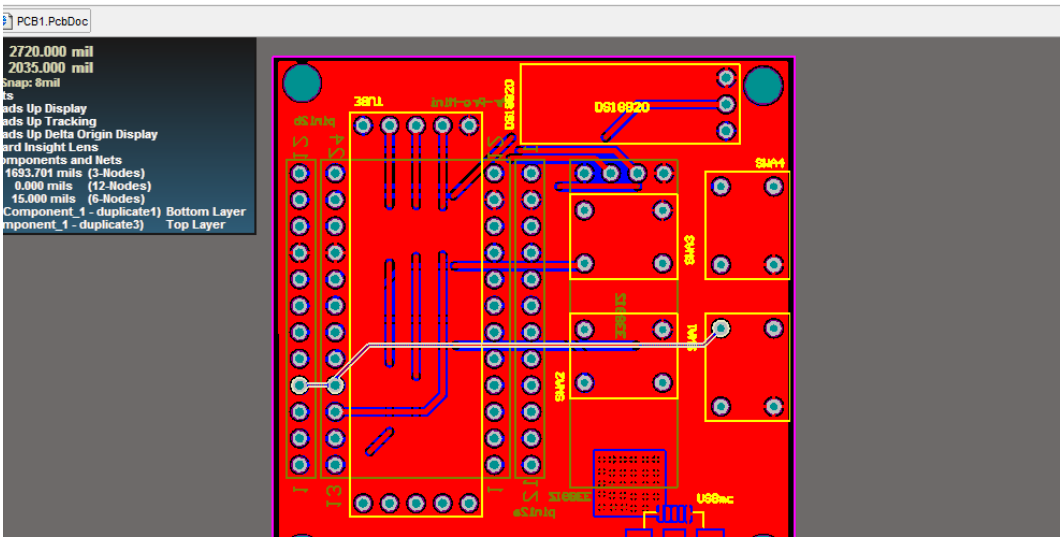

图 2-2　PCB 板示意图

经过上述一系列的操作过程，完成了如上图所示的 PCB 板的绘制。将绘制好的 PCB 板图纸拿到工厂，进行加工。

2.4.元器件的选取

元器件选取得好坏，对于一个研究能否顺利进行具有非常重要的意义。为了保证系统和设备的稳定性和可靠性，除了要根据实际情况之外，一般要考虑的因素如下：

（1）通过根据产品应用的环境和要实现的功能，来选取相应种类、型号、质量等级和封装形式的元器件；

（2） 优先选取质量稳定、可靠性高的元器件；

（3） 优先采用符合国家标准且技术成熟的元器件；

（4） 对于非标类的元器件，使用时要经过严格的验证；

（5） 优先选取价格合理、有良好技术服务的企业生产的元器件；

（6）考虑在产品使用的整个寿命中，元器件能否保持正常工作，而不会超

出元器件的寿命；

（7）尽可能地将元器件由一家或者较少的几家厂商供应，尽量做到定点供应；

（8）在元器件质量达到要求，同时性价比也相当的情况下，可优先选取国产的元器件；

（9）降低生产成本，在质量达到要求的情况下，优先选取价格较低的元器件。

在本研究中，通过比较元器件的几种不同型号的性能特点、成本和大小等因素，来挑选出适合要使用的元器件。

2.4.1 Arduino 开发板的选取

每种 Arduino 都具有各自的用途，根据自己所需要的用途来选择对应的 Arduino。下面对 5 种不同型号的 Arduino 的性能进行比较，从而挑选出符合本实验使用要求的，同时性价比也较高的 Arduino。

表 2-1　几种 Arduino 开发板的比较

	UNO R3	Nano	Mini	Leonardo	MEGA2560 R3
MCU	ATmega328	ATmega168/328	ATmega168/328	ATmega32u4	ATmega2560
工作电压/IO 电压	5v	5v	5v	5v	5v
数字 IO	14	14	14	20	54
PWM	6	6	6	7	15
模拟输入 IO	6	8	8	12	16
时钟频率	16MHz	16MHz	16MHz	16MHz	16MHz
Flash	32KB	16KB/32KB	16KB/32KB	32KB	256KB
SRAM	2KB	1KB/2KB	1KB/2KB	2.5KB	8KB
EEPROM	1KB	512bytes/1KB	512bytes/1KB	1KB	4KB
USB 芯片	ATmega16u2	FTDI FT232RL			ATmega16u2
其他特点	目前使用人数最多的型	功能和 Duemilanove 一	最小的 Arduino 控制器，但下载	可以模拟鼠标键盘等	配置最高的 8 位 Arduino

	号,适合初学者使用	致，但更为小巧	程序得搭配外部的下载器	USB 设备	控制器
价格/元（数据来源于淘宝）	27	13	12	28	228

如上表所示，Arduino pro mini 最为小巧，与同类产品相比，性能还可以，同时价格是最便宜的，所以本次研究就用 Arduino pro mini 作为硬件平台的微型处理器。

2.4.2 温度传感器的选取

温度传感器是将感受到的温度转化为电子信号传输的电子元件。温度传感器根据材料和元器件特性分为热电阻和热电偶两类。热电阻是半导体材料，它对温度的变化很灵敏，但是线性度极差。而热电偶具有宽温度范围并适应各种大气环境，但是不适合高精度的测量和应用。下面选取了三种不同型号的温度传感器，通过比较它们的性能、精度和价格，来选取适合本次研究所用到的传感器。

表 2-2　几种温度传感器的比较

	DS1820	DS18B20	DS1822
A/D 转换精度	9 位	9 位~12 位	9 位~12 位
测温范围（℃）	-55~125	-55~125	-55~125
测温精度（℃）	0.5（-10~85℃）	0.5（-10~85℃）	2
是否支持“一线总线”接口	支持	支持	支持
价格/元（数据来源于淘宝）	6	7	18

如上表所示，三种温度传感器进行比较，DS18B20 支持的 A/D 转换精度较宽，测温精度较精确，同时价格也相对较便宜。所以本次研究就用 DS18B20 作为硬件平台的测温装置。

2.5.Arduino 温度采集程序

在 Arduino 上运行的程序可以使用任何能够被编译成 Arduino 机器码的编程

语言编写。而 Atmel 也提供了数个可以开发 Atml 微处理机程序的集成开发环境，AVR Stduio 和更新的 Atml Atudio。

Arduino 计划也提供了 Arduino Software IDE,一套以 Java 编写的跨平台应用软件。Arduino Software IDE 使用与 C 语言和 C++相仿的编程语言，并且提供了包含常见的输入/输出函数的 Wiring 软件库。一个典型的 Arduino C/C++ sketch 程序会包含两个函数，它们会在编译后合成为 main（）函数：

（1）setup():在程序运行开始时会运行一次，用于初始化设置；

（2）loop():知道 Arduino 硬件关闭前会重复运行函数放的代码。

在使用 GUN toolchain 编译和链接后，Arduino Software IDE 提供了一个程序“avrdude”用来转换可执行档成为能够烧写入 Arduino 硬件的固件。

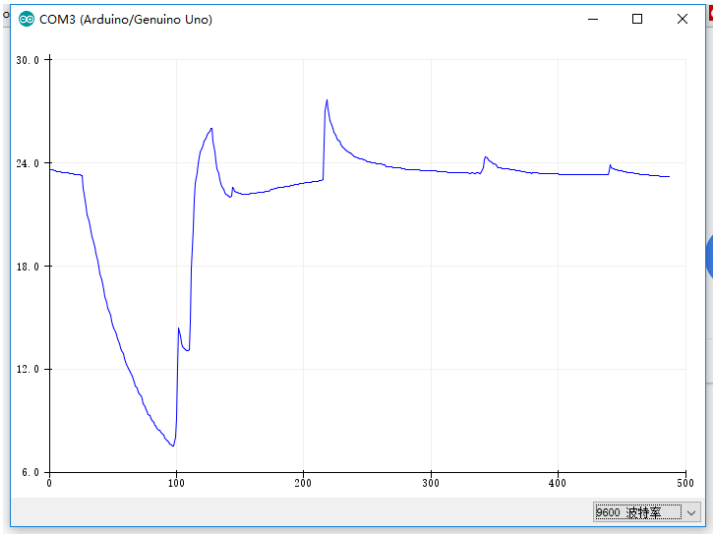

图 2-3　采集温度数据的图像

打开 Arduino IDE 软件的串口绘图器，将从串口缓冲器中获得的数据按照获得的时间顺序，绘制在图上，形成了如上图所示的温度数据的采集图像。由图像上可以看到，最低温度达到了 7℃左右，最高温度达到了 28℃左右。

2.6 本章小结

本章主要是对温度控制平台实现硬件的搭建和 Arduino 程序的设计，对原理

图的设计和 PCB 板的绘制进行了简单的描述，又对几种 Arduino 和温度传感器的性能、大小和价格等进行了比较，挑选出相对较适合本次实验的元器件。最终搭建出的硬件平台，如下图所示：

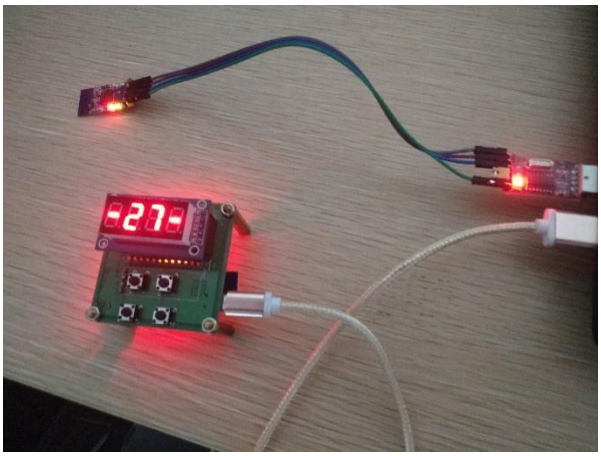

图 2-4　硬件平台实物图

通过 WIRELESS UART 将 Arduino IDE 上编写的代码烧入到 Arduino 上，程序调用函数，获取温度传感器上测量到的温度数据，然后将温度数据传到串口缓冲区中，并在 LED 屏幕上显示出来。

第 3 章　程序界面设计

该程序采用事件驱动方式的编程方法，即当串口接收缓冲区接收到数据时，程序就会自动执行接收数据的函数，对缓冲区中的数据进行相应的处理，该方式提高了程序执行的效率。

3.1 程序设计的目的

在硬件平台上，通过温度传感器采集到房间内空气的温度，将温度数据传给 Arduino，并在 LED 屏上显示出来。当只有一个硬件模块的时候，温度数据看起来还比较直观，但是当硬件模块为多个且分布在房间的各个地方时，再通过 LED 屏来查看该点的温度数据就比较麻烦了。这时候，可以将它们采集到的温度数据都收集到一个界面上，进行统一的管理分析。

需要设计出的程序的功能：

（1） 能够修改端口号、波特率、数据位、校验和停止位；

（2） 既能打开串口，接收到各个硬件平台发送到串口缓冲区中的数据，也能关闭串口，停止接收数据；

（3） 接收到的温度数据能够在界面区域上显示出来，比如“A.温度；B.温度；C.温度……”同时还能够清除该区域内显示的数据信息；

（4） 能够在界面上给硬件平台发送设定温度的数据，硬件平台接收到设定温度的数据之后，通过程序给串口发送”receive”，表示硬件平台已经接收到设定温度的数据。

3.2 程序设计的过程

该程序要实现的功能是，要实现串口通信，从串口接受缓冲区中读取测量到的温度数据，并在界面上显示出来，再将设定温度在界面上输入，传入到串口发送缓冲区中，让 Arduino 读取。实现的过程如下：

（1）在主对话框的实现文件（扩展名为 cpp）中定义全局变量，比如串口的句柄、事件线程句柄、窗口句柄、事件响应函数等。

（2）打开串口。定义使用读写方式和重叠方式，设置判断语句，如果成功打开串口，则获取当前串口参数、设置串口通信参数；如果没能成功打开串口，则弹出“打开串口失败”的框。

创建并立即执行事件线程。设置允许的事件类型，创建事件线程，然后指定线程函数名称，再设置为创建线程后立即执行，最后允许事件函数执行循环体。

事件线程函数的编写。

（3）发送数据。使用 WriteFile（）函数，采用重叠的处理方式。创建 Wol.hEvent 事件句柄，通过 Wol 结果来知晓写操作的完成情况。将发送函数的程序指定在一个按钮的单击事件上。

（4）消息函数。在头文件最前面加入消息的 ID 声明，在主对话框构造函数中加入消息函数声明。在主对话框中，加入自定义消息映射和消息实现。

（5）关闭串口和事件线程。关闭创建串口和事件线程时返回的那些句柄。

3.3 程序运行流程图

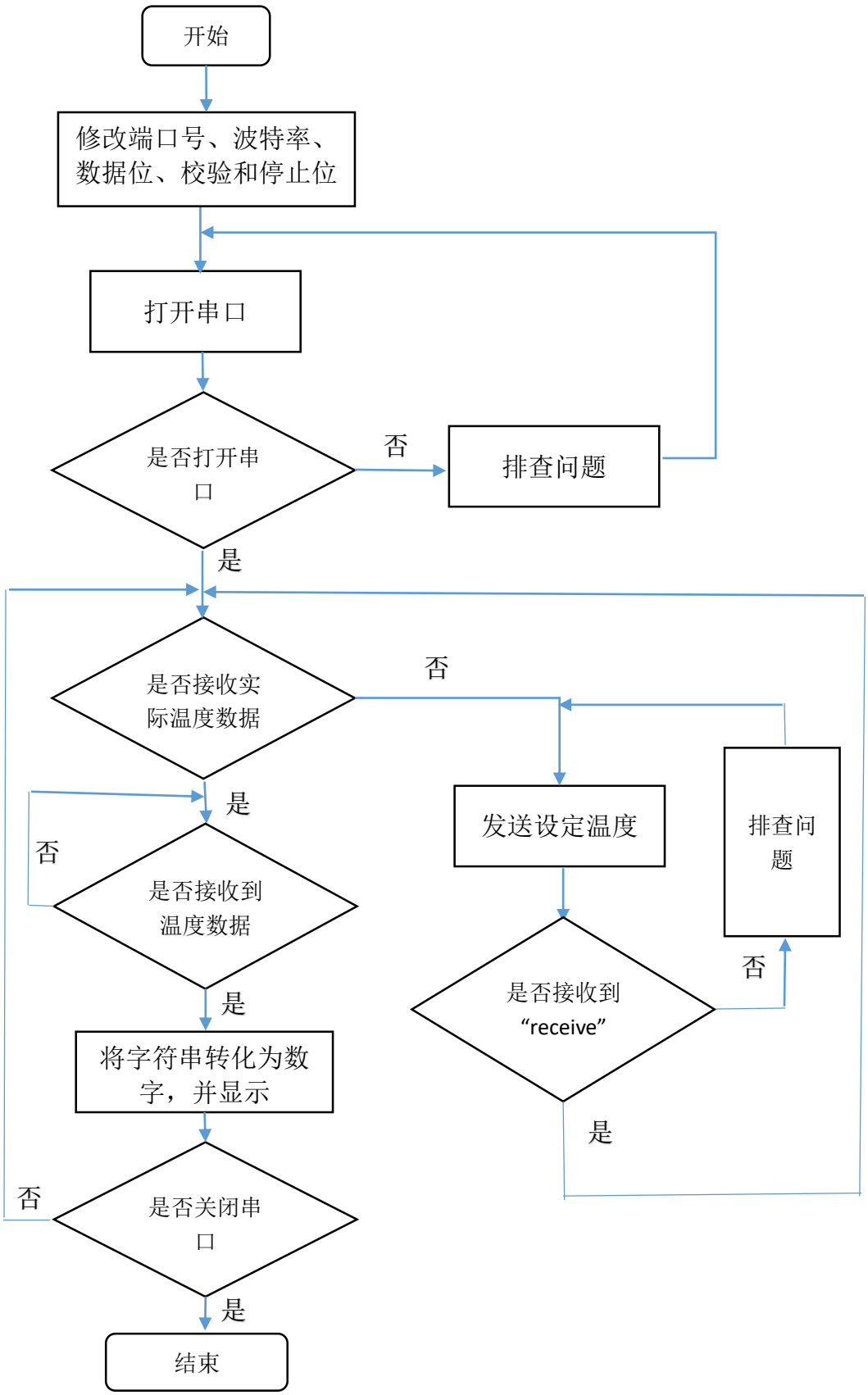
开始
修改端口号、波特率、数据位、校验和停止位
打开串口
是否打开串口
否
排查问题
是
是否接收实际温度数据
否
是
否
是否接收到温度数据
是
将字符串转化为数字，并显示
否
是否关闭串口
是
结束
发送设定温度
是否接收到“receive”
否
排查问题
是

图 3-1　程序运行流程图

首先修改端口号、波特率、数据位、校验和停止位，然后打开串口，判断选择的功能是否为接收实际温度数据，如果是则再判断是否从串口缓冲区接收到数据，如果没有接收到数据则继续接收，如果接收到数据则将温度数据显示在区域中；如果选择的功能不是接收实际温度数据，则为发送设定温度数据功能，发送之后，判断是否接收到“receive”，如果接收到了，则继续选择功能，如果没有接收到，则继续发送设定温度。最后判断是否关闭串口，如果关闭了，则结束，如果没有关闭，则继续选择功能。

3.4 程序测试

3.4.1 程序与串口助手的测试

将程序和串口助手都打开，第一种情况是程序向串口助手发送数据，串口助手接收数据。程序的端口设为“com2”，串口助手的端口设为“com1”，此时两者就能进行串口通信了，程序发送“54321”给串口助手，而串口助手接收到“54321”；第二种情况是串口助手向程序发送数据，程序接收数据。此时，串口助手发送“12345”给程序，而程序接收到“12345”。如图十一所示。

所以，说明该程序与串口助手能够正常通信。

图 3-2　程序与串口助手通信测试图

3.4.2 程序与程序的测试

打开两个程序，一个程序作为发送数据方，另一个程序作为接收数据方，左侧的程序端口设为“com1”，右侧的程序端口设为“com2”，左侧的程序给右侧的程序发送“12345”，右侧的程序接收到“12345”；右侧的的程序给左侧的程序发送“54321”，左侧的程序接收到“54321”。如图十二所示。

所以，两个程序之间能够正常通信。

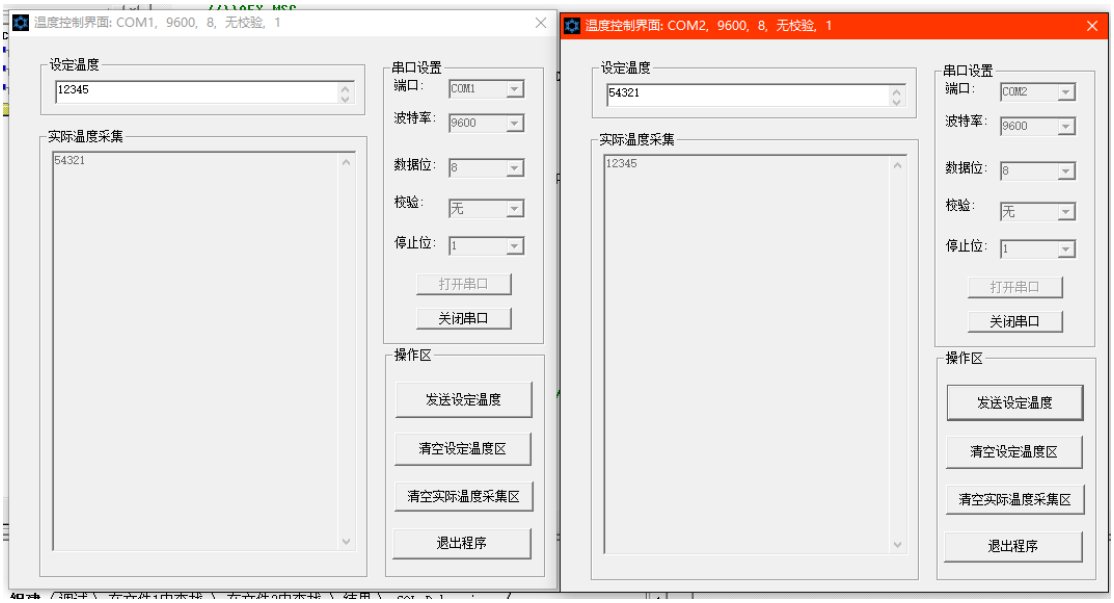

图 3-3　程序与程序通信测试图

3.5 程序运行的结果

图 3-4　温度控制界面图

如上图所示，界面实现的功能是：

1.接收实时温度。Arduino 将温度传感器感应到的温度数据，发送到串口，该程序从串口读取数据，并在实际温度采集区显示出来。

2.发送设定温度。该程序将设定温度发送到串口，Arduino 从串口读取到是数据，则通过程序发送 receive，表示已经接收到设定温度。

3.6 本章小结

本章主要是进行了温度控制界面的设计，简单介绍了设计该程序的目的，详细介绍了串口通信的编程过程，还绘制了程序运行的流程图，更加直观地阐述程序运行的过程。程序编译出来之后，进行测试，观察是否能实现通过串口通信，包括实时温度数据的获取和设定温度数据的发送的功能。测试成功之后，再将该程序应用到本次研究中来。

第 4 章　总结与展望

4.1 毕设工作总结

随着人们生活水平的提高和温度控制算法的进步，大家对于舒适的生活环境的需求越来越强烈。由于每个人对于环境温度的感知不同，所以每个人感觉到的最舒适的温度也会不同，如何让每个人都感觉到舒适，同时又能兼顾到节能，多源多点温控系统的概念应运而生。

本论文依据每个人对舒适温度的要求不同，提出了多源多点的温度控制方法。通过对原理图的设计和元器件的选取，完成了硬件平台的搭建，调研和界面的设计更加完善了温度控制系统。现将本文的毕设工作总结如下：

（1）在查阅了温度控制理论研究文献的基础上，总体上综述课题研究背景与意义，介绍了温度控制理论的发展，其中着重挑选了 PID 控制、神经网络控制、模糊控制、遗传算法和广义预测控制这五个温度控制理论进行详细介绍。还介绍了我国暖通空调自动控制系统的发展。

（2）阐述了硬件平台的搭建过程，包括原理图的设计、PCB 板的绘制和元器件的选取。在 Arduino 软件上，完成温度采集程序的编写。最后将程序烧到硬件上，实现温度采集并上传的功能。

（3）对温差与空调的功率之间的关系进行调研，分定频空调和变频空调，分别调研。

（4）阐述了温度控制界面设计的过程，介绍了界面能够实现的功能。

4.2 未来展望

随着毕业设计课题的深入研究，越来越深刻地意识到本文研究内容的不足。由于本人的时间和所学有限，目前的课题内容研究还处于初级阶段，还有很多的地方可以进行改进：

（1）不能通过硬件平台来直接控制空调的功率，还只是停留在理论阶段，对设定温度与对空调功率的影响进行了调研。

（2）硬件平台还是需要 USB 接电脑获得电源，没有自己独立的电源。

（3）能够同时接收到多个硬件平台上传的温度数据，但是目前只能给一个硬件平台发送设定温度的数据，不能给多个同时发。

我觉得多源多点的温度控制具有极强的现实意义，能够更加精确地控制温度，在日常生活、工业、农业中能得到更为广泛的应用。甚至以后科技发达了，可以通过这种控制方法，直接控制体表衣物的温度，使身体各个部位达到最舒适的温度。

参考文献

[1]胡传志，沈建华，彭晓晶.简单有效的PID温控算法[J].实验室研究与探索，2017，（10）：4-7，40.

[2]李广军，张晶.挠性结构非线性 PID 控制研究[J].测控技术，2007（12）

[3]尹君驰，黄勇.非线性 PID 控制系统设计[J].中小企业管理与科技，2012，（21）：209-210

[4]王蕾，宋文忠.PID 控制[J].自动化仪表，2004，（4）

[5]薄永军.温室温度控制系统神经网络PID控制算法[J].安徽农业科学，2014，（9）：4102-4104.

[6]Simon Hayjin.神经网络原理[M].叶世伟，等译.北京：机械工业出版社，2004：23-25.

[7]Liu X D.The development of fuzzy rough sets with the use of structures and algebras of axiomatic fuzzy sets[J].IEEE Transactions on Knowledge and Data Engineering,2009,21(3):443-462

[8]沙毅，范倩雯，张立立，朱丽春.自校正模糊 PID 控制的 FAST 节点位移控制方法.2018，（4）：487-491

[9]顾俊，张宇.模糊控制的应用现状与发展趋势[J].化工自动化及仪表.2017，（9）：811-812，902.

[10] Holland, J．H．,Adaptation in Natural and Artificial Systems[M]．The University of Michigan Press,1975．

[11]罗乐，笪贤进.基于遗传算法的温度控制系统设计[J].现代电子技术.2012，（18）：16-18.

[12]Clarke D W,Mohtadi C,Tuffs P S.Generalized Predictive Control:Part 1 and Part 2[J].Automatica,1987,23(1):137-160

[13]胡耀华，贾欣乐.广义预测控制综述[J].信息与控制.2000,（3）:248-256

[14]陈延祥.暖通空调自动控制系统的现状综述[J].工程技术研究，2016，（5）：100，111.

[15]潘云钢.我国暖通空调自动控制系统的现状与发展[J].暖通空调，2012，（11）：

235.

[16]刘虎，孙博，刘振兴.中央空调空气处理机组 DDC 控制器的研制[J].制冷与空调（四川），2007，（2）：94-96.

[17]徐娟.浅谈变频技术与变频空调[J].科技创新与应用，2017，（35）：51-52

[18]黄诗淇.简述空调风向对室内温度的影响[J]，黑龙江科技信息，2016，（2）：45-45

[19]吴勇，李林涛，李景等.基于 Arduino 开发环境的光点编码器检测仪的设计[J]，现代电子技术，2014,37（2）：124:126.

[20]纪欣然.基于 Arduino 开发环境的智能寻光小车设计[J],现代电子技术，2012，35（15）：161-163.

附录一

温度采集程序代码：

```
/*DS18B20 采集温度送到数码管显示，1 秒钟采样一次*/
#include <LiquidCrystal.h>
#include <DallasTemperature.h>
#include <OneWire.h>

#define ONE_WIRE_BUS 3       //定义单总线连接的端口 3
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

      unsigned char LED_0F[] =
      {// 0      1      2      3      4      5      6      7      8      9      A      b      C      d
E      F      -

     0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90,0x8C,0xBF,0xC6,0xA1
,0x86,0xFF,0xbf
      };
      unsigned char LED[4];   //用于 LED 的 4 位显示缓存
      int SCLK = 9;
      int RCLK = 8;
      int DIO = 7; //这里定义了那三个脚
      int curTemp = 27;
      unsigned long    curtime;

      void setup ()
      {
        pinMode(SCLK,OUTPUT);
        pinMode(RCLK,OUTPUT);
        pinMode(DIO,OUTPUT); //让三个脚都是输出状态

        Serial.begin(9600);

        sensors.begin();       //启动单总线
        LED[0]=16;
        LED[1]=16;
        LED[2]=16;
        LED[3]=16;
      }

      void loop()
```

```
    {
        curtime = millis();

        LED_OUT(-1);
        LED_OUT(0x08);

        digitalWrite(RCLK,LOW);
        digitalWrite(RCLK,HIGH);

        sensors.requestTemperatures();       //发送温度测量请求命令
        curTemp = sensors.getTempCByIndex(0);       //获取 0 号传感器温度数据
        Serial.println("A"+String(sensors.getTempCByIndex(0)));//将数据发送到
串口

        LED[3]=16; LED[0]=16;
        LED[2] = curTemp/10;
        LED[1] = curTemp%10;

        while(millis() - curtime < 1000)    //采样时间,单位毫秒
        {
              LED4_Display();
         // receive();
        }
    }

     void LED4_Display (void)
     {
        unsigned char *led_table;                  //  查表指针
        unsigned char i;
        //显示第 1 位
        led_table = LED_0F + LED[0];
        i = *led_table;
        LED_OUT(i);
        LED_OUT(0x01);
          digitalWrite(RCLK,LOW);
          digitalWrite(RCLK,HIGH);
        //显示第 2 位
        led_table = LED_0F + LED[1];
        i = *led_table;
        LED_OUT(i);
        LED_OUT(0x02);
          digitalWrite(RCLK,LOW);
          digitalWrite(RCLK,HIGH);
        //显示第 3 位
```

```
    led_table = LED_0F + LED[2];
    i = *led_table;
    LED_OUT(i);
    LED_OUT(0x04);
     digitalWrite(RCLK,LOW);
     digitalWrite(RCLK,HIGH);
    //显示第 4 位
    led_table = LED_0F + LED[3];
    i = *led_table;
    LED_OUT(i);
    LED_OUT(0x08);
     digitalWrite(RCLK,LOW);
     digitalWrite(RCLK,HIGH);
}

void LED_OUT(unsigned char X)
{
    unsigned char i;
    for(i=8;i>=1;i--)
    {
        if (X&0x80)
                {
                   digitalWrite(DIO,HIGH);
                 }
                else
                {
                   digitalWrite(DIO,LOW);
                }
        X<<=1;
                digitalWrite(SCLK,LOW);
                digitalWrite(SCLK,HIGH);
    }
}

/* void receive()
 {
   String comdata="";
   if (Serial.available()>0)
            {
             comdata+=char(Serial.read());
            }
    if(comdata.length()>0)
            {
              Serial.println("receive");//将数据发送到串口
```

```
                }
    }*/
```

附录二

```
/*********************************************************************
*********
 * Funtion:    OnOpencom
 *  Description:    串口参数的设置，包括端口号，波特率，数据位，奇偶校验，
停止位等。
                    打开串口，创建线程函数，开始执行线程。
 * Input: void
 * Output: void;
 * Call by: CCkDlg

*********************************************************************
*******/
#include "math.h"
void CCkDlg::OnOpencom()
{
    // TODO: Add your control notification handler code here

    /************************打开串口************************/
    long buadrate;
    int data;

    UpdateData(true);

    /*打开按钮变暗*/
    m_Closecom.EnableWindow(true);
    m_Opencom.EnableWindow(false);
```

```
/*设置端口号，通过获取组合框的索引*/
switch (m_com.GetCurSel())
{
    case 0 : com = "COM0";                    //端口号为 COM0
             break;
    case 1 : com = "COM1";
             break;
    case 2 : com = "COM2";
             break;
    case 3 : com = "COM3";
             break;
    case 4 : com = "COM4";
             break;
    case 5 : com = "COM5";
             break;
    case 6 : com = "COM6";
             break;
    case 7 : com = "COM7";
             break;
    case 8 : com = "COM8";
             break;
    case 9 : com = "COM9";
             break;
    case 10 : com = "COM10";
             break;
    default : break;
}

/*设置波特率，通过获取组合框的索引*/
switch (m_baud.GetCurSel())
```

```
{
    case 0 : buadrate = 9600;              //波特率为 9600
             break;
    case 1 : buadrate = 14400;
             break;
    case 2 : buadrate = 19200;
             break;
    case 3 : buadrate = 38400;
             break;
    case 4 : buadrate = 56000;
             break;
    case 5 : buadrate = 57600;
             break;
    case 6 : buadrate = 115200;
             break;
    case 7 : buadrate = 128000;
             break;
    case 8 : buadrate = 256000;
             break;
    default : break;
}

/*设置数据位，通过获取组合框的索引*/
switch (m_data.GetCurSel())
{
    case 0 : data = 5;        //5 位数据位
             break;
    case 1 : data = 6;
             break;
    case 2 : data = 7;
```

```
                    break;
        case 3 : data = 8;
                    break;
        default:break;
    }

    /*打开串口*/
    hcom = CreateFile(com, GENERIC_READ | GENERIC_WRITE,
                        0,
                        NULL,
                        OPEN_EXISTING,
                        FILE_ATTRIBUTE_NORMAL                    |
FILE_FLAG_OVERLAPPED,
                        NULL);

    /*判断串口是否打开成功*/
    if (hcom != INVALID_HANDLE_VALUE)
    {
        SetupComm(hcom, 1024, 512);   //设置发送与接收的缓冲区大小
        DCB myDCB;                          //DCB 结构
        GetCommState(hcom, &myDCB);   //获取串口状态
        myDCB.BaudRate = buadrate;     //设置波特率
        myDCB.fBinary = true;

        /*设置奇偶校验，通过获取组合框的索引*/
        CString parity;              //窗口标题使用
        switch (m_parity.GetCurSel())
        {
            case 0 : myDCB.fParity = false;   //设置有无奇偶校验
                        myDCB.Parity = NOPARITY;
```

```
                parity = "无校验";
                 break;

    case 1 : myDCB.fParity = true;   //设置奇校验
                 myDCB.Parity = ODDPARITY;
                parity = "奇校验";
                 break;

    case 2 : myDCB.fParity = true;   //设置偶校验
                 myDCB.Parity = EVENPARITY;
                parity = "偶校验";
                 break;

    default:break;
}

myDCB.ByteSize = data;       //设置数据位

/*设置停止位，通过获取组合框的索引*/
CString stopbits;              //窗口标题使用
switch (m_stop.GetCurSel())
{
    case 0 : myDCB.StopBits = ONESTOPBIT;
                stopbits = "1";
                 break;

    case 1 : myDCB.StopBits = TWOSTOPBITS;
                 stopbits = "2";
                 break;
```

```
        default:break;
    }

    SetCommState(hcom, &myDCB);   //设置串口参数

    m_com.EnableWindow(false);
    m_baud.EnableWindow(false);
    m_data.EnableWindow(false);
    m_parity.EnableWindow(false);
    m_stop.EnableWindow(false);                    //各组合框失能

    m_rx.EnableWindow(false);     //接收框变灰

    // 设置窗口标题
    CString windowsTitle, temp;
    windowsTitle = "温度控制界面: ";
    windowsTitle += com;                            //窗口显示端口号
    windowsTitle += ",    ";
    temp.Format("%d", buadrate);
    windowsTitle += temp;                           //窗口显示波特率
    windowsTitle += ",    ";
     temp.Format("%d", data);
    windowsTitle += temp;                           //窗口显示数据位数
    windowsTitle += ",    ";
    windowsTitle += parity;                         //窗口显示奇偶校验
    windowsTitle += ",    ";
    windowsTitle += stopbits;                       //窗口显示停止位
    this->SetWindowText(windowsTitle);

}
```

```
else
{
    AfxMessageBox("sorry,  打开串口失败!   -_-");

    /*关闭按钮变暗*/
    m_Closecom.EnableWindow(false);
    m_Opencom.EnableWindow(true);
}

hWnd = GetSafeHwnd();     //得到当前窗口的句柄

/*************************** 创 建 并 执 行 事 件 进 程 ************************/
DWORD dwParam;
if (!SetCommMask(hcom, EV_RXCHAR | EV_TXEMPTY))     //设置允许的事件类型
{
    //AfxMessageBox("创建事件掩码失败!   -_-... ");

    /*关闭按钮变暗*/
    m_Closecom.EnableWindow(false);
    m_Opencom.EnableWindow(true);
}

//创建事件进程
hThreadEvent = CreateThread(NULL,
                            0,

(LPTHREAD_START_ROUTINE)ThreadProcEvent,     //指定线程函数名称
                            &dwParam,
```

```
                                    0,
//创建之后立即执行
                                    &dwThreadID);

    if (hThreadEvent == INVALID_HANDLE_VALUE)
    {
        AfxMessageBox("创建事件进程失败!   -_-... ");

        /*关闭按钮变暗*/
        m_Closecom.EnableWindow(false);
        m_Opencom.EnableWindow(true);
    }

    fEventRun = true;                   //允许事件进程执行循环体
}

/*************************************************************************
*********
 * Funtion:    OnClosecom
 * Description:    对各组合框使能，便于下次的选择
                        对句柄等系统资源的撤销，释放资源
 * Input: void
 * Output: void;
 * Call by: CCkDlg

*************************************************************************
*******/
void CCkDlg::OnClosecom()
{
    // TODO: Add your control notification handler code here
```

```
    /*关闭按钮变暗*/
    m_Closecom.EnableWindow(false);
    m_Opencom.EnableWindow(true);

    m_com.EnableWindow(true);
    m_baud.EnableWindow(true);
    m_data.EnableWindow(true);
    m_parity.EnableWindow(true);
    m_stop.EnableWindow(true);                    //使能个组合框，便于下次的选择

    m_rx.EnableWindow(true);

    fEventRun = false;                                    //停止事件进程循环操作
    WaitForSingleObject(hThreadEvent, INFINITE);    //等待事件进程函数退出
    CloseHandle(hThreadEvent);                            //关闭事情线程句柄
    CloseHandle(hcom);                                    //关闭串口句柄

}

/*
 * 事件线程函数
 */
DWORD ThreadProcEvent(LPVOID pParam)   //事件响应函数
{
    DWORD dwEvtMask, dwRes;
    Eol.hEvent = CreateEvent(NULL,     //设置 Eol.hEvent 成员为无信号状态
                             true,
                              false,
                            NULL);
```

```
while(fEventRun)
{
    WaitCommEvent(hcom,                    //监视串口事件
                  &dwEvtMask,              //存放事件掩码组合值
                  &Eol);

     dwRes = WaitForSingleObject(Eol.hEvent, 100);     //等待事件对象句柄

     switch (dwRes)
    {
        case WAIT_OBJECT_0:                //成功得到事件监视结果
            switch(dwEvtMask)
            {
                case EV_RXCHAR:            //接收到数据
                if (!fStopMsg)
                {
                    fStopMsg = true;     //向主线程发送消息，接收到数据
                    ::PostMessage(hWnd,                    //目的窗口句柄
                                  WM_MYMSG,                //消息名
称
                                  0,                       //参数
                                  (LPARAM)EV_RXCHAR);
                }
                break;

                case EV_TXEMPTY:        //发送缓冲区空

                    //AfxMessageBox("发送缓冲区空! -_-...");
                break;
```

```
                }
                break;
            }
        }

        return true;
}

void CCkDlg::OnButton3()
{
        // TODO: Add your control notification handler code here
        UpdateData(true);

        int i, iLen;
        char charInput[512];
        BYTE arrSendData[512];

        /*字符串转为 BYTE 类型*/
        strcpy(charInput, m_txData);
        iLen = m_txData.GetLength();

        for (i = 0; i < iLen; i++)
        {
            arrSendData[i] = (BYTE)charInput[i];
        }

        Wol.hEvent = CreateEvent(NULL,          //创建 Wol.hEvent 事件句柄，并设置为无信号状态
                                 true,
                                 false,
```

```
                                    NULL);
    WriteFile(hcom,                         //写数据
              &arrSendData,
              iLen,
              NULL,
             &Wol);
}

long CCkDlg::OnReceiveEvent(WPARAM wParam, LPARAM lParam)        //自定义消息函数来读取数据
{
    BYTE myByte[1024];
    DWORD dwRes;
    DWORD dwRead;
    DWORD dwErrors;
    COMSTAT Rcs;

    fStopMsg=true;

    ClearCommError(hcom,
                   &dwErrors,        //存放出错信息的掩码组合
                     &Rcs);

    if (ReadFile(hcom,
                 &myByte,
                 Rcs.cbInQue,
                 NULL,
                 &Rol))
    {
```

```
        //数据成功得到，处理数据
        CString s;

        /*转换为字符串，输出到编辑框*/
        for (int i = 0; i < Rcs.cbInQue; i++)
        {
            s += myByte[i];
        }

        //s += "\12\15\12\15";
        m_rx.SetSel(10000, 10000);
        m_rx.ReplaceSel(s);

        UpdateData(false);
    }
    else
    {
        Rol.hEvent = CreateEvent(NULL,
                                 true,
                                 false,
                                 NULL);

        dwRes = WaitForSingleObject(Rol.hEvent, 5000);

        switch (dwRes)
        {
            case WAIT_OBJECT_0:
                if (!GetOverlappedResult(hcom,
                                         &Rol,
                                         &dwRead,
```

```
                                                    TRUE))
                    {
                    }
                      else
                    {

                    }
                    break;

                case WAIT_TIMEOUT:
                break;

                default:
                break;
            }
        }

        fStopMsg = false;                //允许事件函数发送消息
        return 0;
}

void CCkDlg::OnButton5()
{
        // TODO: Add your control notification handler code here
        m_rx.SetWindowText("");      //清空接收框
}

void CCkDlg::OnButton4()
{
        // TODO: Add your control notification handler code here
```

```
    m_tx.SetWindowText("");     //清空输出框
}
```

致　谢

外面的阳光太过明媚，空气中充斥着一股燥热的气息，往年的这个时候，感觉特别开心，因为马上就要迎来一个长长的假期了，到时候就可以旅游、打游戏、一觉睡到大中午，怎么开心，怎么来。不过今年暑假不太一样，我就要离开校园，正式踏入社会了，在工作岗位上奋斗自己的青春。在大学期间，有欢笑，也有泪水，有过春风得意时，也有过沮丧低迷时，在这里正好可以借此机会向所有帮助过我的老师和同学表示深深的感谢。

首先，我要感谢我的母校浙江工业大学，这里有浓郁的学习氛围，师资力量雄厚，还有非常多的学习资源可以获取。在这里，我收获了知识，结识了非常多的好朋友。一待就是四年，爸妈有时候还开玩笑说“学校反倒像是你的家，而家里却像是个客栈，住一晚就走”。

其次，我要感谢我的导师赵云波教授。赵老师上课教学严谨，为人风趣幽默，经常能够提出一些非常有趣的想法供同学们思考讨论。我大三的时候，上过赵老师的数学建模课，印象最深的是赵老师和同学们一起讨论 AGV 小车的路径规划问题的场景，带动了同学们的学习积极性。在整个毕设的完成过程中，赵老师给了我很多建设性的意见，指导我选择论文的研究方向。非常感谢赵老师的帮助。

接着，我要感谢在我毕设阶段帮助过我的所有学长学姐们。在我迷茫于毕设如何进行下去时，他们给了我很多的宝贵意见。在对于 Arduino IDE 这款软件的使用和一些硬件的选取时，他们给我讲解了这款软件的使用方法，介绍了很多元器件的原理、能实现的功能和他们之间的区别等。

然后，我要感谢我的几位真挚的朋友们，大学四年我们一起学习，一起打闹，一起旅游，有问题一起讨论解决。不是家人，胜似家人。在我找工作的时候，给我提供了莫大的帮助。还有我要感谢一位小学妹，在我找工作最低迷的时候，鼓励着我不要放弃，给我提供了很多宝贵的意见，也有很多珍贵的回忆，然人世之事，非人世所能尽，甚悔矣。

最后我要感谢我的父母，是你们给了我学习的机会，让我在大学阶段，能够不用担心经济问题，专心完成学业，也在精神上给予我鼓励，让我更加自信地面对生活。

今当远离，临表涕零，不知所言，望诸君珍重。山高水长，江湖再见！