

班 级 1401052
学 号 14010520033

西安电子科技大学

本科毕业设计论文



题 目 基于单片机的实时温度监控系统

学 院 通信工程学院

专 业 信息工程

学生姓名 张 涛

导师姓名 刘飞航

摘 要

温度是一种最基本也是十分重要的环境参数,对于我们来说,不仅仅是一个量的反映,更能直接影响和作用到我们的生活中,人民的生活与环境的温度息息相关,随着工业和农业的发展,温度测量在生产和生活中有着越来越大的作用。

随着近年智能家居,物联网等技术的发展,单片机智能控制系统发展的越来越完善,市场也越来越宽广,运用单片机和一些传感器搭建智能控制系统,能让我们的生活越来越自动化,智能化。

本文讲述的温度实时监控系统可以对环境温度进行实时监测,并且可以根据环境需求设置报警温度,当温度超过警戒值时会发出警报,可以手动远程开启温度调节模块自由调节温度,或者通过自动控制升温或是降温模块使温度稳定在一个较小范围。本系统采用 STC90C52 单片机作为控制芯片,DS18B20 作为温度测量器件,LCD1602 作为显示器,通过蓝牙串口传输数据以及指令,配合其他元器件构成了电路结构简单、功能实用的温度监控系统。

通过蓝牙串口,将数据和各模块状态存入 MySQL 数据库,然后通过 Web 技术,查看或是更改数据库的数据,串口可以发出指令操控单片机,进而实现远程智能控制功能。

关键字： 单片机, 温度监控系统, 温度传感器, 远程操控

ABSTRACT

Temperature is a basic environmental and important parameter. For us, it is not just a reflection of quantity. It can also have a direct impact on our lives. People's lives are closely related to the temperature of the environment. With the development of industry and agriculture. Temperature measurement plays an increasingly important role in production and life.

With the development of smart home and internet of things technology in recent years, the development of the MCU intelligent control system has become more and more complete, and the market has become more and more broad. Using an SCM and some sensors to build an intelligent control system can make our lives more and more automated,smart.

The temperature real-time monitoring system described in this article can monitor the ambient temperature in real time and set the alarm temperature according to the environment requirements. When the temperature exceeds the alarm value, an alarm will be issued. You can manually turn on the fan to adjust the temperature remotely, or control the fan or heating automatically to let the device stabilize the temperature in a smaller range. The system uses STC90C52 microcontroller as the control chip, DS18B20 as a temperature measurement device, LCD1602 as a monitor, through the bluetooth serial port to transfer data and instructions, with other components to form a simple circuit structure and practical temperature monitoring system. Through the Bluetooth serial port, the data and the status of each module is stored in the MySQL database, and then through the Web technology, the data of the database is viewed or changed. The serial port also issues commands to control the microcontroller, thereby achieving remote intelligent control.

Keywords: **single chip microcomputer, temperature monitoring system,**
 temperature sensor, remote temperature control

目 录

摘 要 2

ABSTRACT 3

第一章 绪论 2

1.1 实时温度监控系统概述	2
1.2 研究背景与意义	2
1.3 国内外发展现状	3
1.3.1 国内发展近况:	3
1.3.2 国外发展近况	3
1.4 本章小结	4

第二章 温度监控系统总体设计 5

2.1 引言	5
2.2 系统框图设计	5
2.3 系统工作流程	7
2.4 系统数据传输	7
2.5 本章小结	8

第三章 硬件系统设计 9

3.1 系统电路原理图	9
3.2 单片机最小系统	9
3.2 温度采集电路	11
3.2 LCD 显示电路	13
3.3 升温、降温和声光报警模块	14
3.3 蓝牙串口模块电路图	16

第四章 软件系统设计 17

4.1 引言	17
--------------	----

4.2 单片机系统程序设计	18
4.2.1 单片机系统整体设计.....	18
4.2.2 DS18B20 读写程序	18
4.2.3 LCD1602 读写程序	21
4.2.4 蓝牙串口程序.....	22
4.2.5 蓝牙数据帧格式.....	23
4.3 串口服务程序设计	25
4.4 Web 程序设计	27
4.5 数据库设计	28
 第五章 总结与展望 30	
5.1 总结	30
5.1.1 测试结果.....	30
5.1.2 总结与分析.....	32
5.2 展望未来	32
 致谢 33	
 参考文献 34	
 附录 35	

第一章 绪论

1.1 实时温度监控系统概述

对于我们的生产和生活来说，温度是一个十分重要的量，生活在一个合适的温度环境里会让人觉得舒适，温度的掌控能促进工业和农业的发展，提升产品的质量以及产出效率，所以温度的监控对我们的生产和生活都有着非常重大的意义，温度实时监控系统主要分为两部分，首先是监测，其次是控制。监测主要是对环境温度进行实时测量，了解当前温度状况，控制是根据需求对温度进行适当调整，让其在一个合理范围内相对稳定下来。如，对于温室大棚，不同时期需要不同的温度，往往需要根据当前作物生长状况对温度进行适当调整，这种情况一般选用手动控制比较好；而对于恒温孵化器这类恒温系统，手动控制其温度在一个恒定范围的话一方面是操作性比较差，另一方面是不易控制，所以这种情况一般采用自动控制。本次设计采用 STC89C52 单片机作为主控芯片，对 DS18B20 温度传感器检测到的环境温度进行数据分析和处理、数值显示并采用串口获取数据并存储于数据库，进而通过网页操作控制参数，最终实现单片机对温度的实时智能控制。

1.2 研究背景与意义

温度在生活中有着无与伦比的重要作用，温度与我们的生产和生活息息相关。工业革命至今，温度的把控极大地促进了工业的发展，在冶金，钢铁等诸多行业，超过 80% 的工业都需要考虑温度因素。温度是一种最基本的环境参数，对于我们来说，温度不仅仅是一个物理量的反映，更能直接影响和作用到我们的生产和生活中。随着工业和农业的发展，温度测量在生产和生活中有着越来越大的作用，因此研究温度的测量方法和装置具有重要的意义。而当今，随着社会经济的高速发展，温度测量技术越来越重要，温度测量技术不仅能用于工业生产领域，还能用于农业、畜牧业、医疗、卫生以及安全等诸多其他领域。以农业来说，温室大棚技术现在已经很成熟了，温度的控制是温室大棚的一个重要技术，合理控

制室温模拟季节的变化能让农作物反季,更好更快的生长,同时还能增加产量提升收益。在畜牧业中,给家禽家畜提供舒适的环境对于养殖也是至关重要的,其中温度就是其中的一个重要因素,在家禽养殖中,人工孵化对温度的控制也是至关重要的。

1.3 国内外发展现状

1.3.1 国内发展近况:

我国在温度计量技术领域的发展相对国外发达国家起步较晚,但是经过多年来的不断发展,还是取得了很多优异的成绩。我国工程技术人员在吸收发达国家温度测控技术的基础上,才掌握了温度室内微机控制技术,该技术仅限于对温度的单项环境因子的控制。我国温度测控设施计算机应用,在总体上正从消化吸收、简单应用阶段向实用化、综合性应用阶段过渡和发展。在技术上,以单片机控制的单参数单回路系统居多,尚无真正意义上的多参数综合控制系统,与发达国家相比,存在较大差距。我国温度测量控制现状还远远没有达到工厂化的程度,生产实际中仍然有许多问题困扰着我们,存在着装备配套能力差,产业化程度低,环境控制水平落后,软硬件资源不能共享和可靠性差等缺点。

1.3.2 国外发展近况

国外对温度控制技术研究较早,始于 20 世纪 70 年代。先是采用模拟式的组合仪表,采集现场信息并进行指示、记录和控制。80 年代末出现了分布式控制系统。目前正开发和研制计算机数据采集控制系统的多因子综合控制系统。990 年代中期,智能温控仪问世,它是微电子技术、计算机技术和自动测试技术的结晶。目前,国际上已开发出多种智能温控器系列产品。智能温控器内部都包含温度传感器、A/D 转换器、信号处理器和接口电路。有的产品还有多路选择器、中央控制器(CPU)、随机存储器(RAM)和只读存储器(ROM)。现在世界各国的温度测控技术发展很快,一些国家在实现自动化的基础上正向着完全自动化、无人化的方向发展。

1.4 本章小结

温度与我们生产和生活息息相关，如果我们能合理的把控温度，那么我们的生产和生活也会有质的改变，所以说，温度监控是至关重要的。在这个智能化以及自动化的时代，能随时轻松便捷的观测或是控制温度是有必要的，本次设计的核心正是在于此，你可以通过手机远程操控温度监控设备，非常智能化。

第二章 温度监控系统总体设计

2.1 引言

本系统的控制核心器件选用的是 STC 公司生产的 51 系列单片机，使用 C 语言进行控制软件编写，自由度大，通过算法实现数字逻辑控制简单方便。能与温度传感器通信还能操控 LCD 显示器以及升温模块和降温模块等其他器件，还可以与 PC 机通，上传数据。

温度测量模块使用 DS18B20 传感器，相比于传统测温手段，该传感器输出温度的数字量，使用单片机能很好地接受和处理，省去了复杂的外围模数转换电路，节省了空间，缩小了系统整体体积。其输出的数字量温度经单片机转换处理后转换为模拟量，通过 LCD 显示，简单方便。

为了实现远程以及智能控制，加入了蓝牙串口传输数据，蓝牙串口能与 PC 进行通讯，使用无线技术传输数据，相比有线串口，没有冗长的连接线，并且还能与手机蓝牙通讯达到无线控制，当然我们后面采用了 Web 技术，不光可以手机控制，还能实现远程控制，所以手机蓝牙控制只是在设计阶段用于研究与测试。

2.2 系统框图设计

系统框图设计如图 2.1 所示，系统整体可分成三个模块：温度监测模块、温度控制模块和 PC 部分的上位机，其中监测模块包括 DS18B20 温度传感器，LCD1602 字符型液晶显示器和一个嗡鸣器加 LED 灯组成的声光报警器；温度控制模块由一个升温模块和一个降温模块构成，当选择自动控制模式时，升温模块和降温模块由单片机系统构成的下位机直接控制，当工作在手动模式时，升温模块和降温模块由上位机发出指令控制；上位机通过蓝牙串口与单片机连接，由于我的电脑不支持有蓝牙，所以使用了汇承 HC-06-USB 蓝牙虚拟串口模块与单片机蓝牙串口进行通信

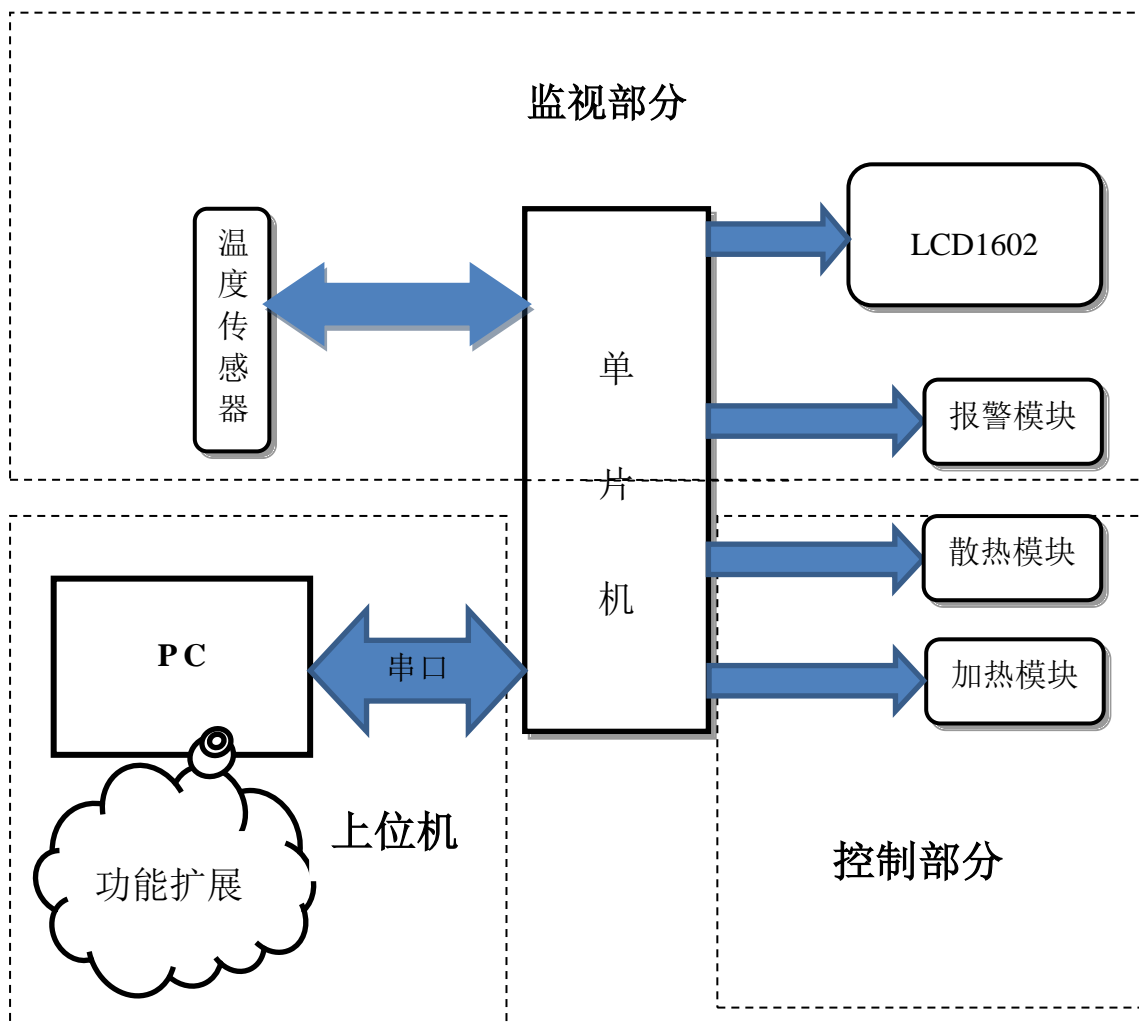


图 2.1 系统原理框图

单片机系统输入为 DS18B20 温度传感器采集到的温度，系统输出为 LCD 显示模块，升温模块，降温模块以及报警模块，其中，当工作在自动调节模式时，升温模块和降温模块由单片机直接自动操控；当工作在手动调节模式时，为了达到远程无线观测和控制，升温模块和降温模块由蓝牙串口发出指令控制：在 Web 上发出操控指令，指令传到数据库，串口模块读取数据库指令进而操控单片机控制升温模块和降温模块。其中串口模块至关重要，它联系着单片机系统和 Web 系统，从硬件层次来说是两者沟通的桥梁，从软件层次来说是两个系统通信的接口。

2.3 系统工作流程

温度采集模块 DS18B20 与单片机之间采用单总线工作方式,单片机发出获取温度指令后 DS18B20 传回温度数字量,经单片机转换后为模拟量通过 LCD1602 显示,同时单片机将当前温度以及系统状态定时(温度,报警温度,升温模块以及降温模块工作状态)通过蓝牙串口发给 PC 端,PC 端将收到的数据后存于数据库,而且 PC 端定时将当前时间通过蓝牙串口发给单片机,单片机收到后通过 LCD1602 显示。通过 Web 技术,获取数据库信息,进行渲染后可得到当前温控系统工作状态,修改数据库信息,当串口通信模块检测到数据库信息发生更改时下达指令给单片机对各模块进行相应控制,从而达到远程智能无线控制的效果。

2.4 系统数据传输

本系统用到的数据传输模式有如下几种:

- (1) DS18B20-单片机之间的单总线模式:单总线是一种串行通信技术的扩展技术。它仅仅使用一根信号线就能完成时钟以及数据的传输,其特色之处在于使用一根线就能完成数据的收发,优点较多,如节省 I/O 线,简单的资源结构,降低成本和便于维修和扩展等。
- (2) LCD1602 和单片机之间的并行通信模式:并行通信是指两个设备或模块之间一组数据的各数据位在多条线上同时传输,并行通信传输速率高,但传输距离较小,适合近距离大量数据交换时使用。
- (3) 单片机与 PC 之间的蓝牙串口通信模式:蓝牙串口是基于 SPP 协议(Serial Port Profile),能在蓝牙设备之间创建串口进行数据传输的一种设备。蓝牙串口的目的是针对如何在两个不同设备(通信的两端)上的应用之间保证一条完整的通信路径。
- (4) 网页浏览器与服务器之间的 HTTP 通信协议:HTTP 协议(HyperText Transfer Protocol,超文本传输协议)是基于 TCP 协议的用于从 Web 服务器传输超文本到本地浏览器的传输协议。它可以使浏览器更加高效,使网络传输减少。

2.5 本章小结

本系统采用了多种通信技术将各个功能模块完美契合，形成一个统一整体，进而协同监测和控制温度，使用有线传输和无线传输两种数据传输模式，通过网络技术，使系统更加智能化和自动化，这是一个物联网的时代，只要有网络，我们随时随地都能观测和控制温度，解放了我们的双腿，让一切变得更简单。

第三章 硬件系统设计

3.1 系统电路原理图

整个系统以 51 单片机为中心进行展开,许多器件都是直接使用现成的模块,使整个系统更加模块化,数字化,同时也省去了复杂的外围电路,这个硬件系统可以分为四个部分:单片机最小系统(控制中心),温度采集电路(系统输入),LCD 字符型液晶显示器电路、升温模块、降温模块以及声光报警模块电路(系统输出),蓝牙串口模块电路(单片机与 Web 沟通的桥梁)。

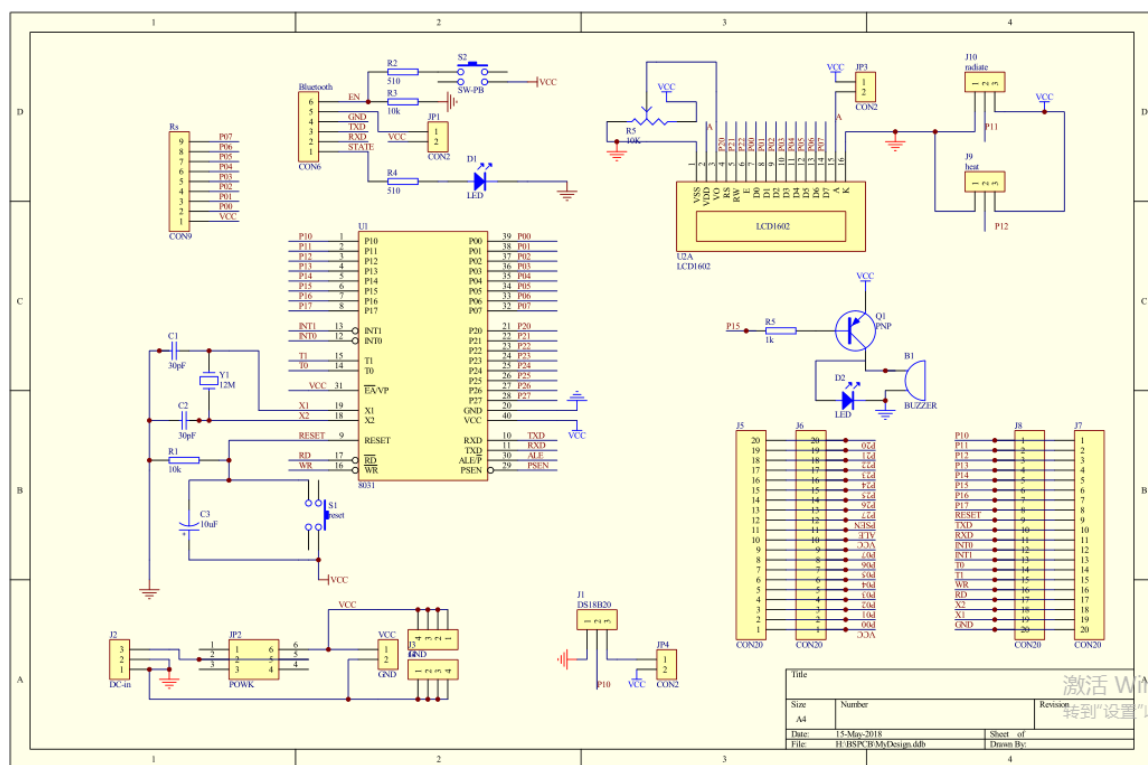


图 3.1 系统电路原理

3.2 单片机最小系统

如图 3.1 所示,一个单片机最基本的系统包括单片机,时钟电路和复位电路。

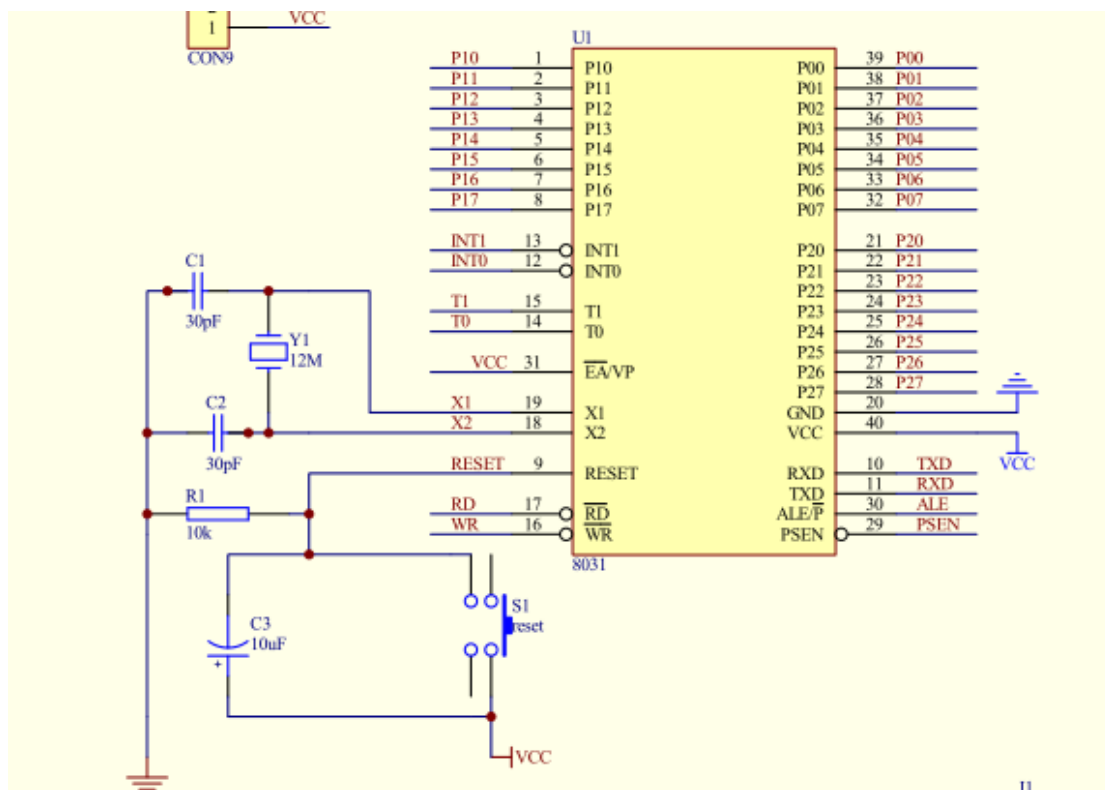


图 3.2 单片机最小系统电路图

单片机：本次设计使用的单片机型号为 STC90C516RD+，新型 8051 系列微控制器的功耗低，强抗干扰能力强，指令代码与 8051 完全兼容，但速度提高了 8-12 倍。内部集成了 MAX810 专用复位电路。封装形式为 DIP40 封装和 5V 电源。其中当程序从内部 ROM 开始执行时 EA 接 VCC，当程序从外部 ROM 开始执行时，EA 接 GND，由于本系统没有使用外部 ROM，所以 EA 接 VCC。

复位电路：复位电路由 R1, C3 和 S1 组成，当系统处于稳态时，C3 隔绝直流，此时 S1 处于断开状态，R1 两端等势，所以 RESET 引脚电位为 0 电位。这款单片机是 RESET 引脚为低电平时是正常工作，高电平时复位，所以此时可以正常工作；上电的瞬间，电容 C3 + 极是 5V 电压，R1 左侧是 0V 电压，由于“电容电压不能突变”，此时电容 C3 要进行充电，此时电容等价于一根导线，全部电压都加在了 R1 上，那么 RESET 端口为高电平。电容充电过程中 C3 两端电压增大，充电电流逐渐减小，R1 两端电压减小，则 RESET 两端电压随之减小，当电容充电完成后，电路中没有电流，R1 两端也就不存在电压差，此时 RESET 变为低电平，也就是 0V。从上面的分析可知，给单片机加电后，RESET 引脚会先持续一小段时间的高电平，最后逐渐降为低电平，这个变化的过程为上电复位过程。其中电容充

电这段时间 一般要求不少于两个机器周期的时间；按键复位：按键复位即通过按下 S1 使系统复位，也称手动复位，按键复位有两个过程，开始系统处于稳态，由上面分析可知此时 RESET 引脚的输入电压值是 0V，手动按下 S1 后电容 C3 被短路，电容会很快被放电，RESET 电压值由 0V 逐渐变化为 5V，变为高电平复位状态。当断开 S3 类似于上电复位，C3 重新充电，随后 RESET 电压逐渐从 5V 变为 0V。

时钟电路：时钟电路即为晶振电路，由于本系统需要使用串口进行通信，蓝牙串口默认波特率为 9600，为了得到 9600 波特率使用 11.0592MHz 无源晶振，在时钟电路中，晶振本身并不能产生时钟，只是起到选频的作用，51 单片机内有一个高增益的反相放大器，该放大器输入端与输出端分别为单片机的引脚 X1 与 X2。它与石英晶共同构成自激振电路，旁路电容 C1、C2 与外接石英晶体接在具有反馈功能的放大器中，构成了并联反馈振荡电路。

3.2 温度采集电路

温度采集模块使用 DS18B20，采用单总线方式工作，单总线与单片机 P10 口进行通信，其中 JP4 为跳线帽，用于调试或故障检测时使用，电路如图 3.3 所示。

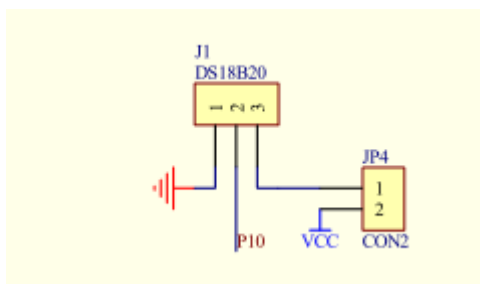


图 3.3 温度采集电路

DS18B20 是一款比较常见的数字化温度传感器，抗干扰能力强，体积小，硬件开销低，精度高，且接线方便，封装后可应用于多种场合，如管道式，不锈钢封装式，螺纹式，磁铁吸附式，型号多种多样，有 LTM8874，LTM8877 等等，改变其外封装后可应用于不同场合。封装后的 DS18B20 可用于温室大棚测温，锅炉测温测温，养殖场测温，孵化间测温，实验室测温等各种常见温度场合。

DS18B20 采用单总线模式传输数据与指令，顾名思义，只用一根数据线来完成数据的读和写，系统中的一切数据交换以及控制指令都由这根线完成，以下是单总线协议工作的基本流程：

（1） 初始化

基于单总线的所有传输过程都是以初始化开始的，初始化过程由主机发出的复位脉冲和从机响应的应答脉冲组成。应答脉冲使主机知道总线上有从机设备且准备就绪。

（2） ROM 操作指令（写）

当主机检测到应答脉冲后，就可以发出 ROM 命令。这些命令与各个从机设备的唯一 64 位 ROM 序列码相关，当单总线上连接多个从机设备时，允许主机指定操作某个从机设备。这些命令还使得主机可以检测到总线上有多少个从机设备及其设备类型，或者有没有设备处于报警状态。从机设备可能支持五种 ROM 命令（实际情况与单总线器件型号有关），每种命令长度为 8 位，主机在发出功能命令之前，必须送出合适的 ROM 命令。

（3） 存储器操作指令（写）

存储器操作命令即为功能命令，通过 ROM 操作命令使得总线主机与总线上某些或某一从机设备确定了通信关系之后，主机发出的命令可以驱动从机设备进行相应的动作。

（4） 数据传输（读/写）

从机设备会把主机要求的信息以串行传输的方式送到单总线上。

所有的单总线器件要求采用严格的通信协议来确保数据的完整性。其协议中规定的信号类别有：复位脉冲，应答脉冲、写 0、写 1、读 0 和读 1。所有这些信号，除了应答脉冲之外，均由总线主机产生。并且发送所有的命令和数据都是字节的低位在前。

每次访问单总线器件，都必须严格遵守这个命令序列。如果出现序列混乱，则单总线器件不会响应主机。

3.2 LCD 显示电路

LCD1602 显示电路如图 3.4 所示，其中 R5 为 10k Ω 变阻器，用于调节 LCD 对比度，JP3 为跳线帽，用于故障排查和测试。

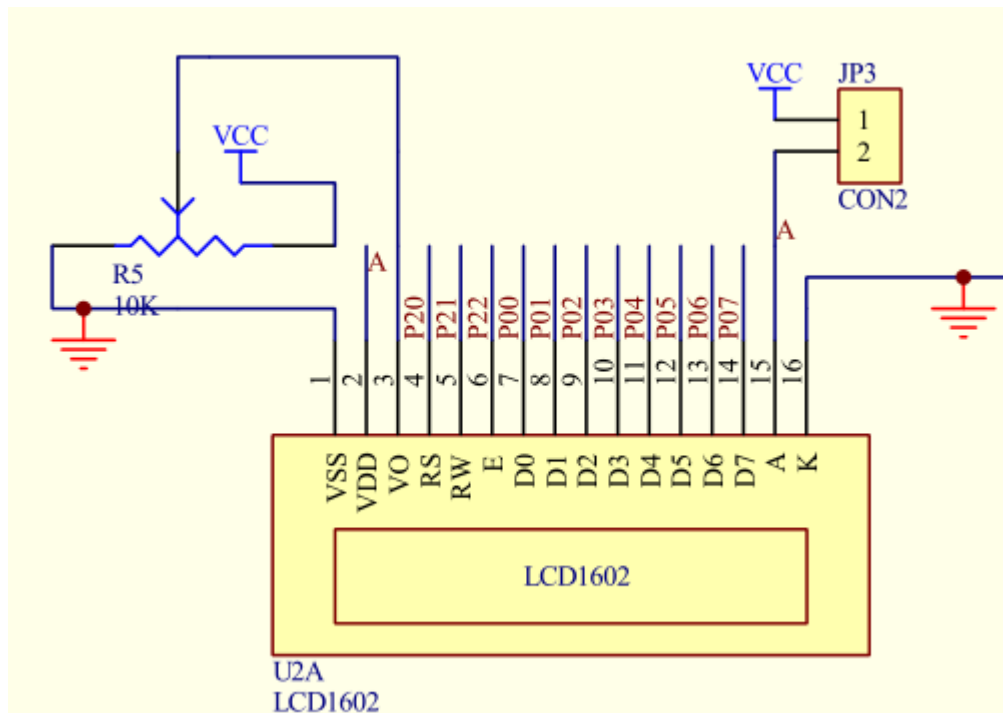


图 3.4 LCD 电路原理图

表 3.1 LCD1602 接口信号说明

编号	符号	引脚说明	编号	符号	引脚说明
1	VSS	电源地	9	D2	DataI/O
2	VDD	电源正极	10	D3	DataI/O
3	VL	液晶显示器偏压信号	11	D4	DataI/O
4	RS	数据命令选择端(H/L)	12	D5	DataI/O
5	R/W	读写选择短(H/L)	13	D6	DataI/O
6	E	使能信号	14	D7	DataI/O
7	D0	DataI/O	15	BLA	背光源正极
8	D1	DataI/O	16	BLK	背光源负极

LCD1602 液晶显示器也叫 1602 字符型液晶显示器，共 19 个引脚，其引脚说明如表 3.1 所示。它是一种专门用来显示字母、数字、符号的点阵型液晶模块。它是由若干个 5x7 或者 5x11 的点阵字符位组成，每个点阵字符位都可以用显示一个字符，每位之间有一个点距的间隔，每行之间也有间隔，起到了字符间距和行间距的作用，正因为如此，所以它不能很好的显示图片。

3.3 升温、降温 and 声光报警模块

升温模块，降温模块以及声光报警模块三个模块电路结构十分相似，电路图如图 3.5，图 3.6 以及图 3.7 所示，只不过降温模块使用了相对集成的带驱动的现成模块，而声光报警器以及升温模块焊接了一个 PNP 三极管做放大器来驱动，三个模块工作方式一样，都是通过单片机 IO 给一个低电平触发，启动相应模块工作。

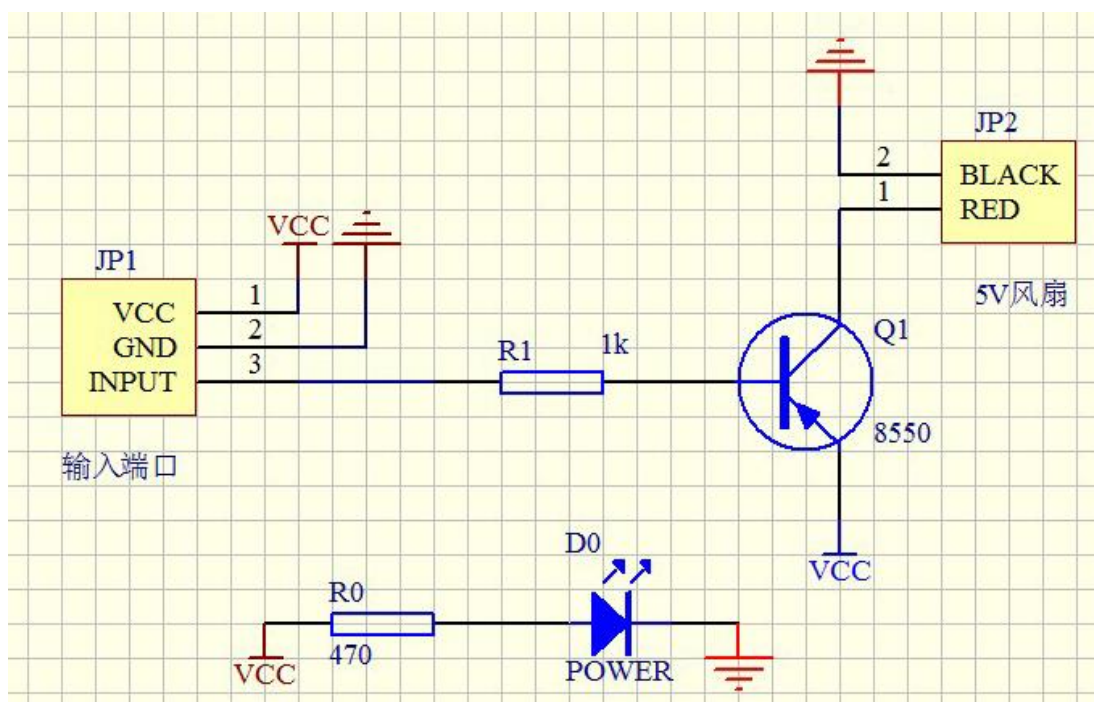


图 3.5 降温模块电路图

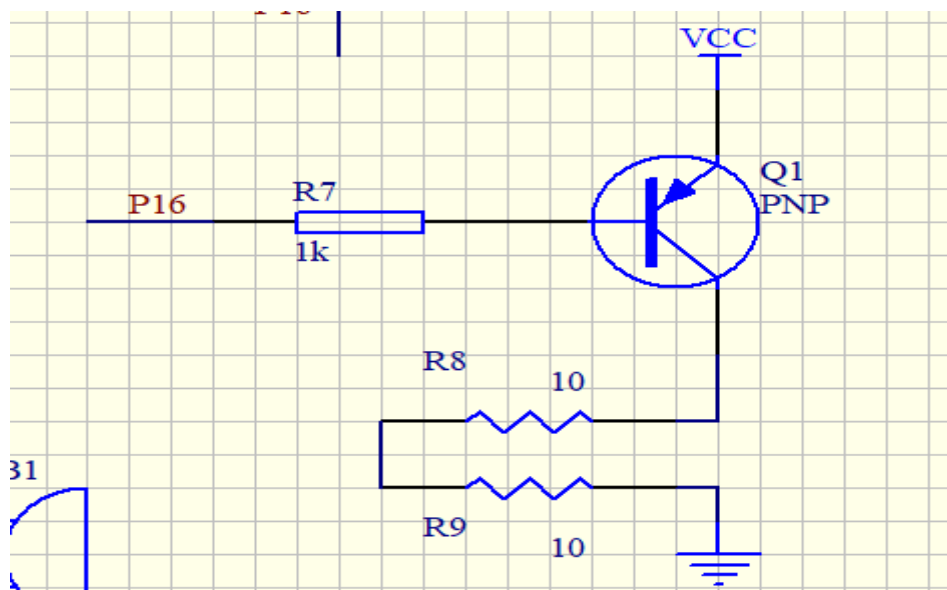


图 3.6 声光报警模块电路图

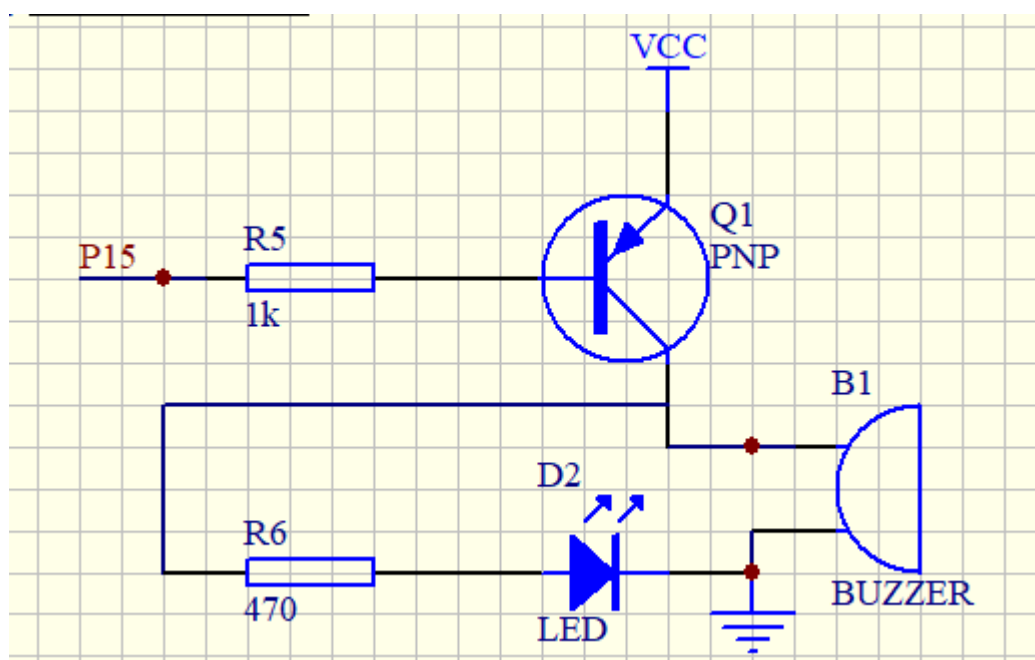


图 3.7 声光报警模块电路图

这是一个比较常见的 PNP 三极管的共射放大电路，当单片机给升温，降温以及声光报警模块输入低电平时，三极管将单片机输出的小电流信号放大，驱动功率需求较大的设备。共射放大电压和电流放大倍数较大，因而功率放大倍数较大，其输入和输出电阻适中，比较符合我们的要求。

3.3 蓝牙串口模块电路图

如图 3.8 所示，蓝牙串口选用 HC-02，S2 为复位开关，按下开关时会断开当前连接，重新搜索蓝牙设备，JP1 为跳线帽，用于调试以及故障排查，D1 为蓝牙状态指示灯，当蓝牙处于连接状态时，D1 长亮，当蓝牙处于未连接状态时 D1 灯熄灭。

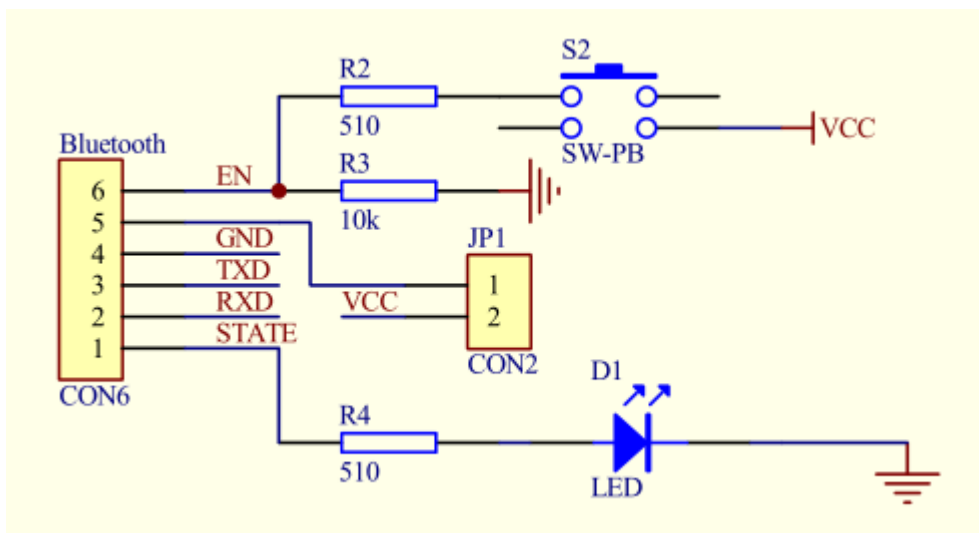


图 3.8 蓝牙串口模块电路图

HC-02 蓝牙串口模块是基于蓝牙 2.0 版本研发，高稳定性，超低功耗，工业级的蓝牙数传模块。用户无需关心复杂的无线通信配置以及传输算法，只需要通过 TTL 串口连接到设备。

HC-02 从机模块上电，可跟手机配对后连接进行数传。另外，可以和 HC-05 或 HC-06 主机进行连接，即可取代一条传统的串口线，省去布线工作，使用非常灵活，如图 3.9 所示。

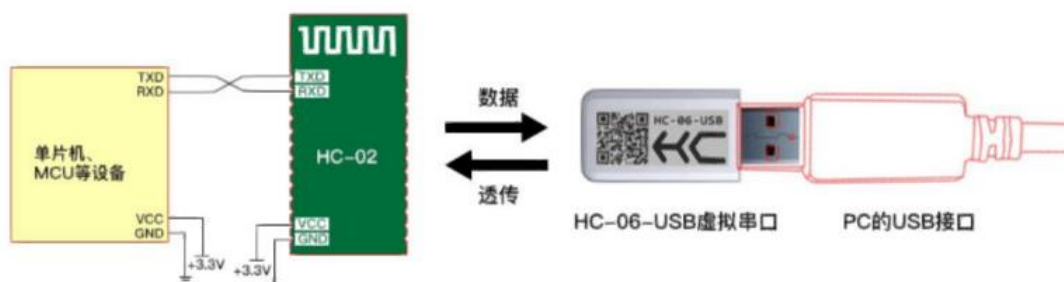


图 3.9 蓝牙串口与 PC 连接

第四章 软件系统设计

4.1 引言

软件系统整体架构如图 4.1 所示，整个系统一共三大模块，一共使用了五种编程语言，单片机程序使用 C 语言，PC 串口服务以及数据库读写程序使用当前比较火的编程语言 python，Web 后台服务程序使用 Java，Web 前端使用 JavaScript 和 HTML。其中单片机程序主要功能是与 DS18B20 通信，获取当前温度，并通过 LCD1602 显示，同时负责与 PC 通信，将温度与当前单片机系统各模块工作状态发送给 PC，并接收来自 PC 发送的时间，以及控制指令。串口服务程序主要负责与单片机通信，接受单片机发来的温度信息以及温控系统各模块工作状态，同时将接收到的温度信息以及单片机各模块工作状态上传到数据库，或者从服务器获取单片机各模块工作状态并与本地数据对比，从而发送指令给单片机对各模块进行控制。Web 程序主要负责获取数据库内存储的温度以及系统各模块的工作状态并在网页显示，当收到用户对温度系统状态修改指令时，修改数据库存储的状态值进而达到远程智能控制。

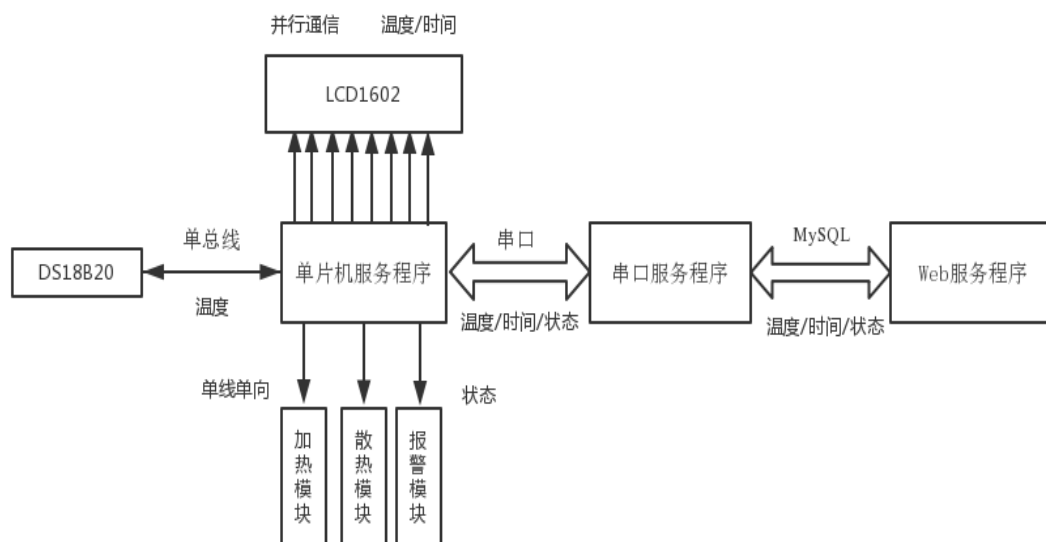


图 4.1 软件系统结构

4.2 单片机系统程序设计

4.2.1 单片机系统整体设计

单片机软件系统如图 4.2 所示，主程序主要负责温度获取，数据显示，数据发送，数据帧解析，调度各功能模块，串口中断服务程序主要负责数据接收以及将接收到数据进行拆帧，并丢弃无效帧。

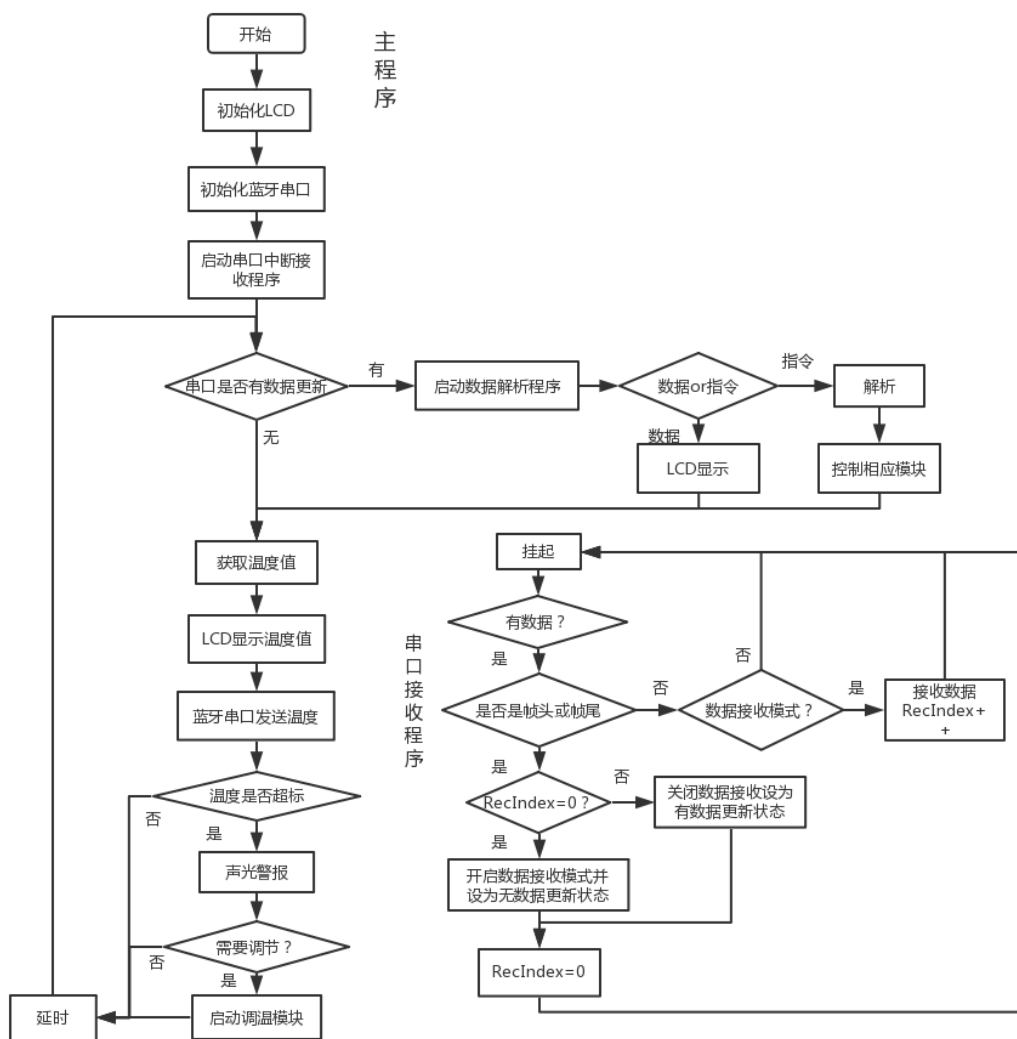


图 4.2 单片机系统软件流程图

4.2.2 DS18B20 读写程序^[14]

前面我们提到 DS18B20 采用单总线协议通信，主机发出各种操作命令都是

向 DS18B20 写 0 和写 1 组成的命令字节, 接收数据时也是从 DS18B20 读取 0 或 1 的过程。因此首先要搞清主机是如何进行写 0、写 1、读 0 和读 1 的。

单总线传输数据, 读和写都在这根线上完成, 所以单总线器件对时序要求非常严格, 从上面我们可以看出单总线工作模式共有三种工作时序

(1) 初始化时序: 初始化时序如图 4.3 所示, 主机首先发出一个 $480\mu\text{s}$ — $960\mu\text{s}$ 的低电平脉冲, 然后释放总线变为高电平, 并在随后的 $480\mu\text{s}$ 时间内对总线进行检测, 如果有低电平出现说明总线上有器件已做出应答。若无低电平出现一直都是高电平说明总线上无器件应答。

做为从器件的 DS18B20 在一上电后就一直在检测总线上是否有 $480\mu\text{s}$ — $960\mu\text{s}$ 的低电平出现, 如果有, 在总线转为高电平后等待 $15\mu\text{s}$ — $60\mu\text{s}$ 后将总线电平拉低 $60\mu\text{s}$ — $240\mu\text{s}$ 做出响应存在脉冲, 告诉主机本器件已做好准备。若没有检测到就一直在检测等待。

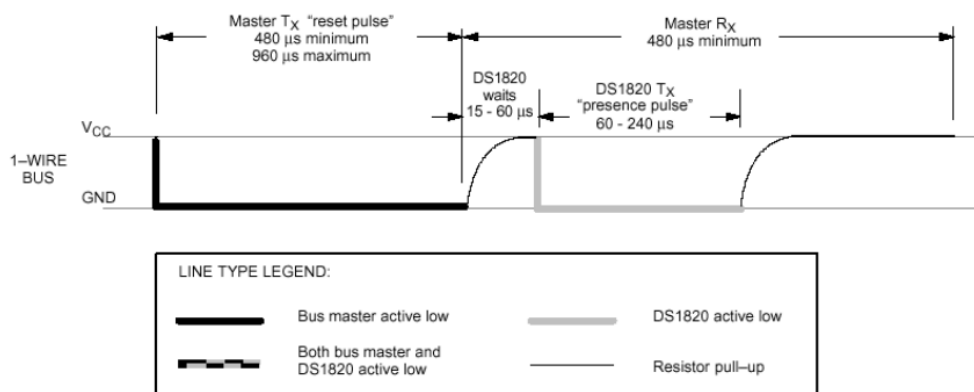


图 4.3 单总线初始化时序图

(2) 写时序: 写时序如图 4.4 所示, 写周期最少为 $60\mu\text{s}$, 最长不超过 $120\mu\text{s}$ 。写周期一开始做为主机先把总线拉低 $1\mu\text{s}$ 表示写周期开始。随后若主机想写 0, 则将总线置为低电平, 若主机想写 1, 则将总线置为高电平, 持续时间最少 $60\mu\text{s}$ 直至写周期结束, 然后释放总线为高电平至少 $1\mu\text{s}$ 给总线恢复。而 DS18B20 则在检测到总线被拉底后等待 $15\mu\text{s}$ 然后从 $15\mu\text{s}$ 到 $45\mu\text{s}$ 开始对总线采样, 在采样期内总线为高电平则为 1, 若采样期内总线为低电平则为 0。

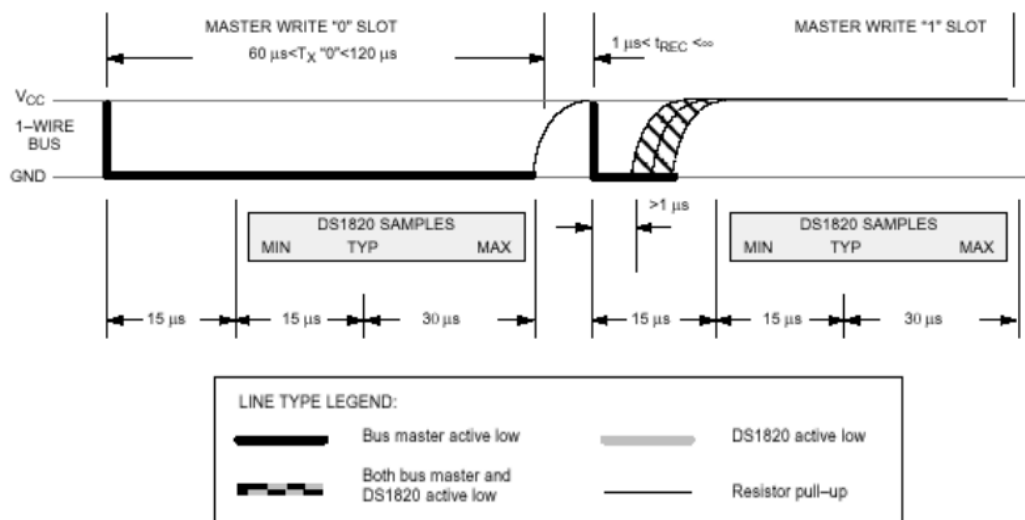


图 4.4 单总线写操作时序图

(3) 读时序：读操作时序如图 4.5 所示，读周期是从主机把单总线拉低 $1\mu s$ 之后就得释放单总线为高电平，以让 DS18B20 把数据传输到单总线上。作为从机 DS18B20 在检测到总线被拉低 $1\mu s$ 后，便开始送出数据，若是要送出 0 就把总线拉为低电平直到读周期结束。若要送出 1 则释放总线为高电平。主机在一开始拉低总线 $1\mu s$ 后释放总线，然后在包括前面的拉低总线电平 $1\mu s$ 在内的 $15\mu s$ 时间内完成对总线进行采样检测，采样期内总线为低电平则确认为 0。采样期内总线为高电平则确认为 1。完成一个读时序过程，至少需要 $60\mu s$ 才能完成

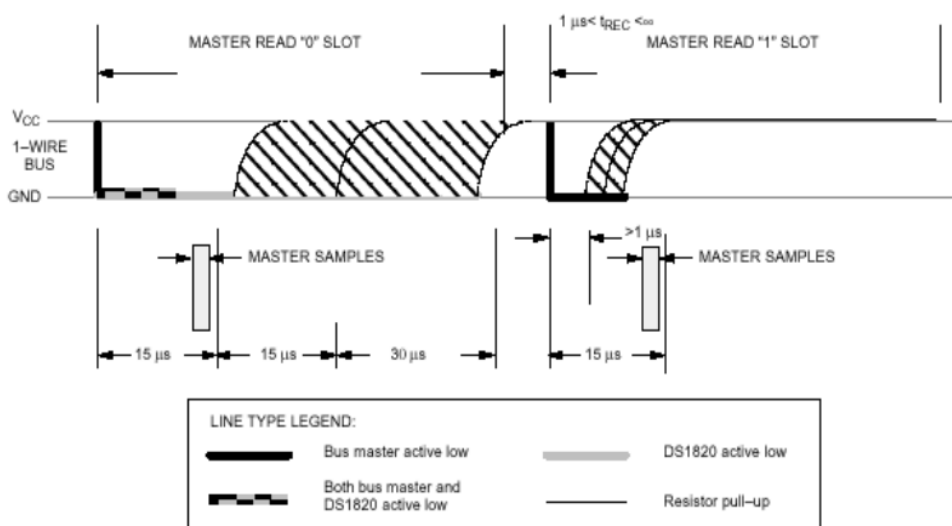


图 4.5 单总线读操作时序图

简单的读取温度值的步骤如下：

- (1) 跳过 ROM 操作 (0xcc)
- (2) 发送温度转换命令 (0x44)
- (3) 跳过 ROM 操作 (0xcc)
- (4) 发送读取温度命令 (0xbe)
- (5) 读取温度值

DS18B20 存储的温度值是以补码的形式存储的，所以读出来的温度值是实际温度值的补码，要把的转换为原码。

正温度的话，原码就是补码本身，所以在 12 位分辨率下，温度的计算公式是：

$$\text{温度值} = \text{读取值} * 0.0625$$

负温度的话，原码是补码减 1 再取反，所以在 12 位分辨率下，计算公式为：

$$\text{温度值} = -(\text{读取值} - 1 \text{ 再取反}) * 0.0625$$

4.2.3 LCD1602 读写程序^[15]

LCD1602 采用并行接口进行数据收发，LCD 共四种操作时序。

- (1) 读状态：输入：RS=L, RW=H, E=H;
输出：D0~D7=状态字。
- (2) 写指令：输入：RS=L, RW=L, E=高脉冲，D0~D7=指令码；
输出：无。
- (3) 读数据：输入：RS=H, RW=H, E=H;
输出：D0~D7=数据。
- (4) 写数据：输入：RS=H, RW=L, E=高脉冲，D0~D7=数据；
输出：无。

但本次设计只用到其中的两种时序：写数据和写指令，时序图如图 4.6 所示。LCD 写数据与写指令操作十分类似，不同的是写数据时 RS 引脚接高电平，写指令时 RS 接低电平。其操作步骤为：

- (1) 初始化
- (2) 写指令 (RS=L) 设置显示坐标
- (3) 写数据 (RS=H)

以写指令为例，先将 RS，RW 置为低电平，输入数据 Data，短暂延时后将使能端 E 置为高电平，再延时一段时间后将使能端 E 置为低电平，写数据过程类似，这里不再做说明。

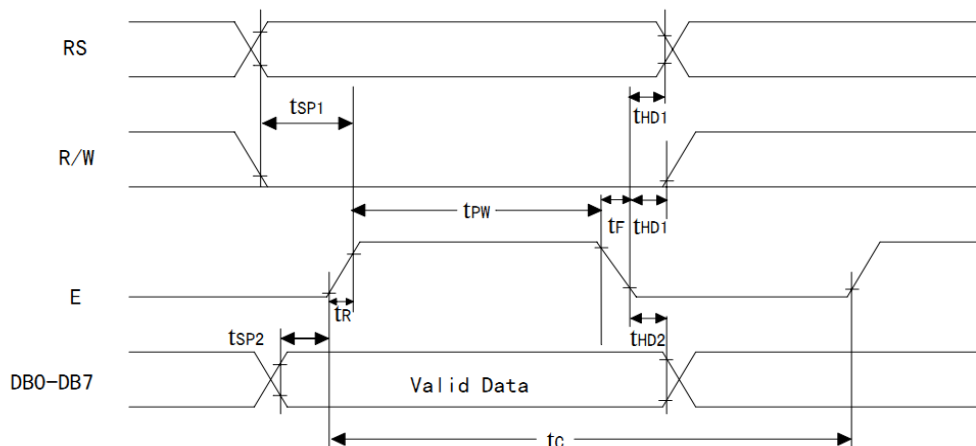


图 4.6 LCD1602 写操作时序图

4.2.4 蓝牙串口程序

蓝牙串口操作与普通串口操作一样，单片机串口通信使用定时器 T1，所以使用串口的时候定时器 T1 就不能用于其他功能操作。

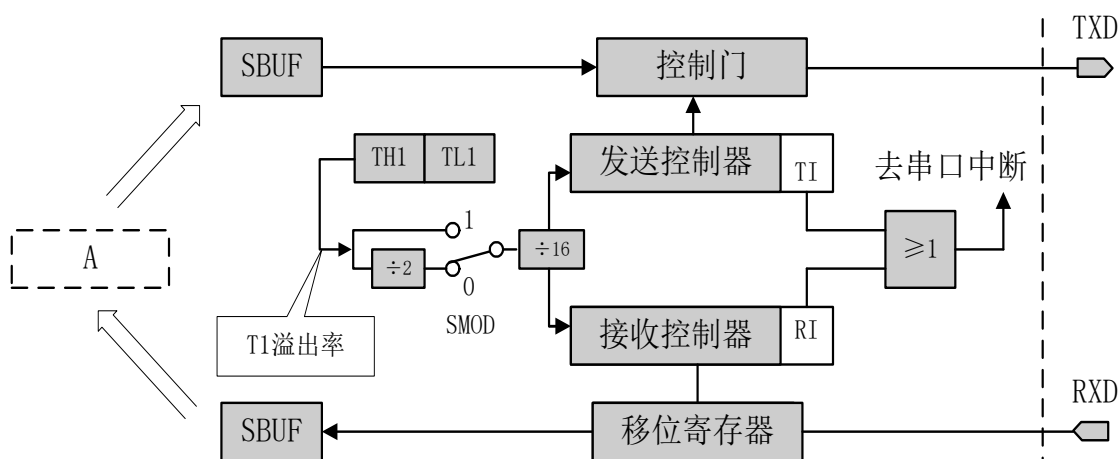


图 4.7 单片机串口内部结构

如图 4.7 所示，串口内部有两个物理上独立的接收、发送缓冲器 SBUF，它们占用同一地址 99H；接收器是双缓冲结构；发送缓冲器，因为发送时 CPU 是主动的，不会产生重叠错误。

串口使用异步通信方式，不要求收发双方时钟的严格一致，实现容易，设备开销较小，但每个字符要附加 2~3 位用于起止位，各帧之间还有间隔，因此传输效率不高。

串口操作步骤：

(1) 先设置波特率：

设置定时器 T1 为工作方式 2（设置 TMOD 寄存器）；

给计数器赋初值（工作方式 2 会自动重装）。

(2) 设置串口工作方式：

设置 SCON（如果允许）。

(3) 如果使用中断方式，那么打开相应的中断和总中断。

(4) 打开定时器 T1，开始产生波特率：

设置 TRx。

4.2.5 蓝牙数据帧格式

由于蓝牙串口与 PC 之间数据交换的过程中，不光有数据，还有指令，串口以字节为单位传输数据，所以一个数据或一条指令要分次一个字节一个字节的逐一发送，并且串口收发是相互独立的，设计过程中发现存在接收一组数据或指令的过程中在发另一组数据或指令，这个过程中比较容易出现一组数据不能正常接收或是发送，最终出现数据乱码或是不是我们想要的结果。为了较好的区分数据和指令以及减少错误信息数量，本系统对数据进行了封装，以帧为单位发送数据，组帧方式如表 4.1 所示，每一帧以 7FH 作为一帧的开头以及结尾，帧的第二位用于区分当前帧是数据还是指令，第三位开始直到 7FH 为具体的数据或是指令。而接受端接受过程中从 0x7F 后开始，首先判断是数据还是指令，数据的话将数据存入接受数据存储区，指令的话对指令进行判断后作出相应的控制操作。

表 4.1 单片机接收数据帧格式

Index	0	1	x
数据	0x7F	0（数据）/1（指令）	（数据、指令）	0x7F

单片机对接收到的数据处理函数如下所示，首先对接收到的数据帧第一位进行判断，如果是数据就讲数据显示，当然这里的数据主要是时间信息，在调试过程中也可以发送一些调试信息用于测试系统，而在 PC 端对数据处理方式也是采用类似方法，不同的是由于单片机没有对 PC 端的控制操作，所以对功能选择位做了稍微的修改，当为 0 时表示当前温度，为 1 时表示当前系统工作状态。

```
void DataResolve() {
    uchar i;
    // static uchar tmp_time[17];
    i=1;
    if(BluRecData[0]=='0') { //数据
        while(BluRecData[i] != '\0')
        {
            now_time[i-1] = BluRecData[i];
            i++;
        }
        LcdDisplayStr(0,0,now_time);
    }
    else { //指令
        switch(BluRecData[1])
        {
            case '0': mode = 0; //自动模式
                       break;
            case '1': mode = 1; //手动模式
                       break;
            case '2': warming = off; //关闭加热
                       break;
            case '3': warming = on; //开启加热
                       break;
            case '4': cooling = off; //关闭散热
                       break;
            case '5': cooling = on; //开启散热
                       break;
            case '6': up_val= BluRecData[2]; //设置温度上限
                       break;
            case '7': down_val = BluRecData[2]; //设置温度下限
                       break;
            case '8': stable = BluRecData[2]; //设置稳定值
                       break;
            default: break;
        }
    }
}
```

4.3 串口服务程序设计

由于我们 PC 与蓝牙串口传输数据使用了 USB 虚拟串口，所以在 PC 端接受蓝牙串口数据时同一般串口一样，调试过程中可以使用串口调试助手，但由于我们后续要讲数据存入 MySQL 数据库，而在网上也没有找到现成的软件可以将串口接收数据存入数据库，所以我使用了当前比流行的编程语言 python 编写了一个串口服务程序，对接收到的数据帧进行解析，将接收到的单片机系统发来的温度以及系统状态信息保存于数据库，并且通过多线程，通过定时器每过一定时间，将系统时间发送给单片机，单片机接收到后将其显示，然后一个定时存储数据线程，定时将当前温度以及时间存入数据库，用于 Web 程序绘制温度变化曲线，串口服务程序入口函数如下所示，每个线程都是一直循环工作。之所以使用定时器加多线程是为了减少前面提到的情况，收发同时进行容易出现错帧的情况。

串口服务程序流程图如图 4.8 所示

```
if __name__ == '__main__':
    #创建串口连接
    global NowTemp;
    updttemp.flag=0;
    plist = list(serial.tools.list_ports.comports());
    if len(plist) <= 0:
        print("the serial port can't find") ;
    else:
        plist_0 = list(plist[0]) ;
        serialName = plist_0[0] ;
        serialFd = serial.Serial(serialName, 9600, timeout=60) ;
        print("check which[%s] port was really used>" % serialFd.name);
    thread.start_new_thread(save_temp, (15,));#启动定时保存数据线程
    NOWSTATUS=NowStatus(serialFd);#创建一个系统工作状态对象
    timer = threading.Timer(1, send_time);#启动定时向串口发送时间线程
    timer.start();
    data_resolve(serialFd,NOWSTATUS);#接收并解析串口数据（主线程）
```

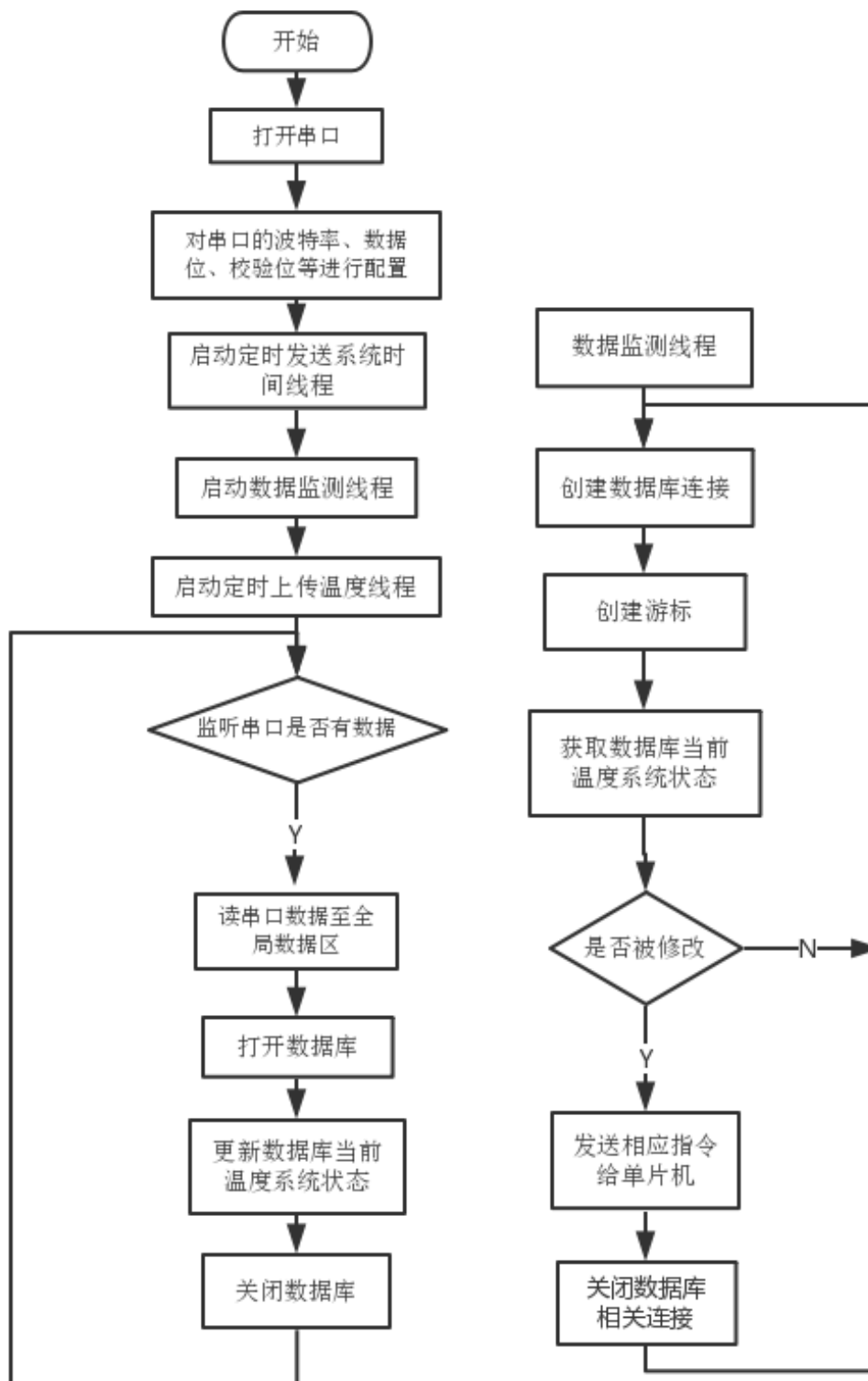


图 4.8 串口服务程序流程图

串口服务程序使用 python 开源社区提供的 pyserial 模块进行串口数据操作，PyMySQL 模块进行 MySQL 数据库操作。串口服务程序是整个系统的重要环节，

单片机系统与 Web 应用沟通的桥梁，使用数据库将单片机系统与 Web 系统完美连接，达到了远程智能控制的效果。

4.4 Web 程序设计

Web 程序使用 Java 作为后台语言,JavaScript 配合 HTML 作为前端页面设计,配合 JSP、DIV、CSS、Ajax, JDBCC 等技术完成 Web 应用程序,通过网页浏览器,访问互联网,我们可以远程观测以及控制温度系统,并且可以直观地观看温度变化曲线,其程序流程图如图 4.9 所示。

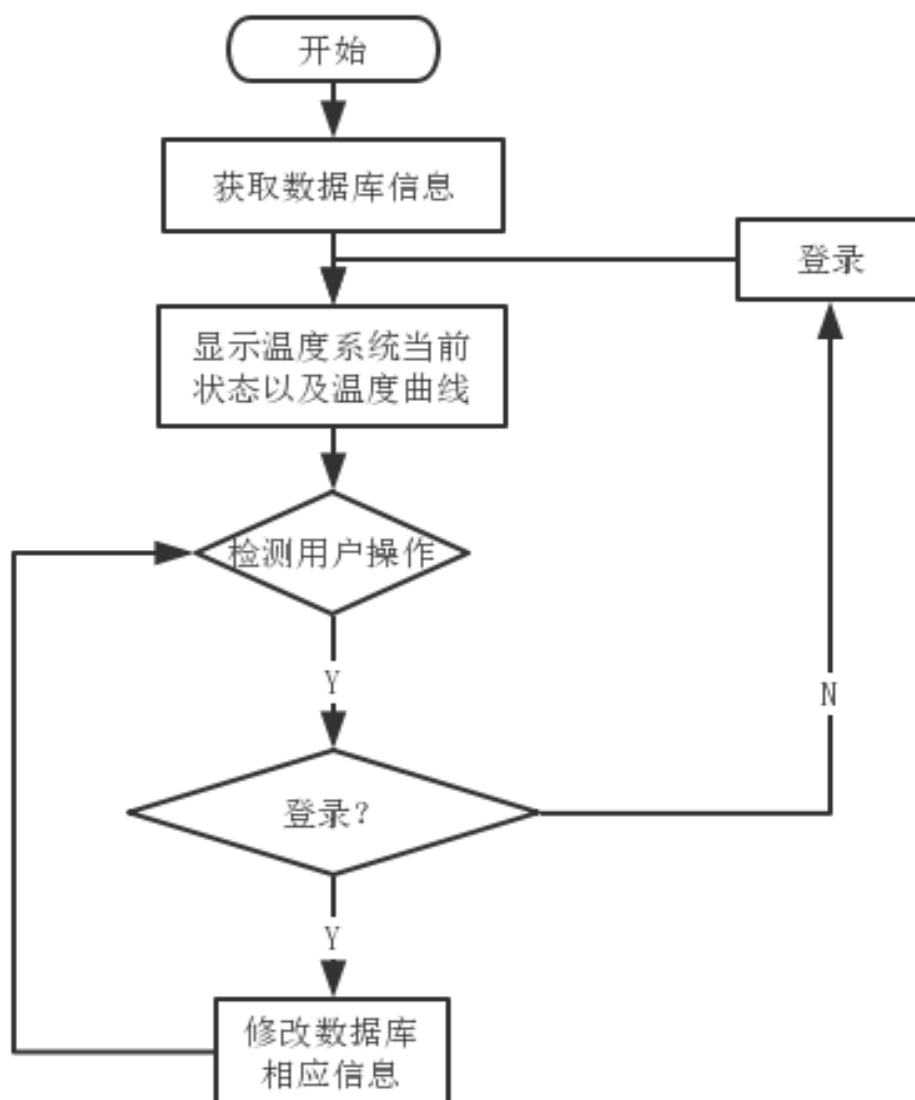


图 4.9 Web 服务程序流程图

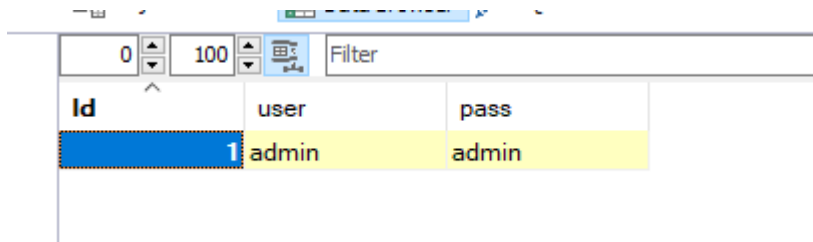
通过 Web 技术，我们可以在网页上看到系统当前温度信息，还能直观的用温

度曲线显示温度变化情况。为了避免系统信息被随意修改，设置了登录机制，只有当登录之后才能更改系统各模块状态信息，未登录即游客模式只能看到系统工作状态

4.5 数据库设计

数据库内共存储 3 种数据结构：管理员身份验证数据，温度系统当前状态，温度记录。

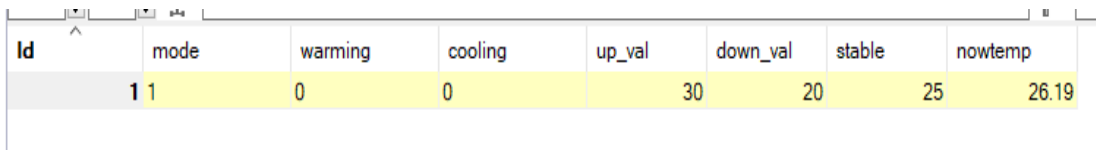
- (1) 管理员身份验证数据：管理员身份验证数据如图 4.10 所示，该数据即为登录时使用的用户名及密码，用于授权操控温度系统。



Id	user	pass
1	admin	admin

图 4.10 管理员身份验证数据

- (2) 温度系统当前状态信息：温度系统当前状态信息数据如图 4.11 所示，包括当前温度，报警温度，各模块工作状态，系统工作模式（手动或自动），在前端页面修改这些信息后将作用于数据库进而通过串口控制单片机。



Id	mode	warming	cooling	up_val	down_val	stable	nowtemp
1	1	0	0	30	20	25	26.19

图 4.11 温度系统当前状态信息数据

- (3) 温度记录：温度记录如图 4.12 所示，从整点开始，每隔 15 分钟串口会存储一个温度值，这些值用于显示温度曲线，让温度变化更直观的展现。

Id	date	temp
363	2018-05-22 00:15:00	25.25
364	2018-05-22 00:30:00	24.75
365	2018-05-22 00:45:00	24.69
366	2018-05-22 01:00:00	24.63
367	2018-05-22 01:15:00	24.69
368	2018-05-22 01:30:00	24.69
369	2018-05-22 01:45:00	24.63
370	2018-05-22 02:00:00	24.56
371	2018-05-22 02:15:00	24.63
372	2018-05-22 02:30:00	24.56
373	2018-05-22 02:45:00	24.56
374	2018-05-22 03:00:00	24.5
375	2018-05-22 03:15:00	24.56
376	2018-05-22 03:30:00	24.5
377	2018-05-22 03:45:00	24.44
378	2018-05-22 04:00:00	24.44
379	2018-05-22 04:15:00	24.44
380	2018-05-22 04:30:00	24.38
381	2018-05-22 04:45:00	24.38
382	2018-05-22 05:00:00	24.38
383	2018-05-22 05:15:00	24.38

图 4.12 温度记录数据

第五章 总结与展望

5.1 总结

5.1.1 测试结果

经测试以及调试后，整个系统运行正常。

单片机能正常显示当前时间、温度以及工作状态，并且能正常接收蓝牙虚拟串口发出的控制信息，并作出相应的控制动作，如图 5.1 所示。

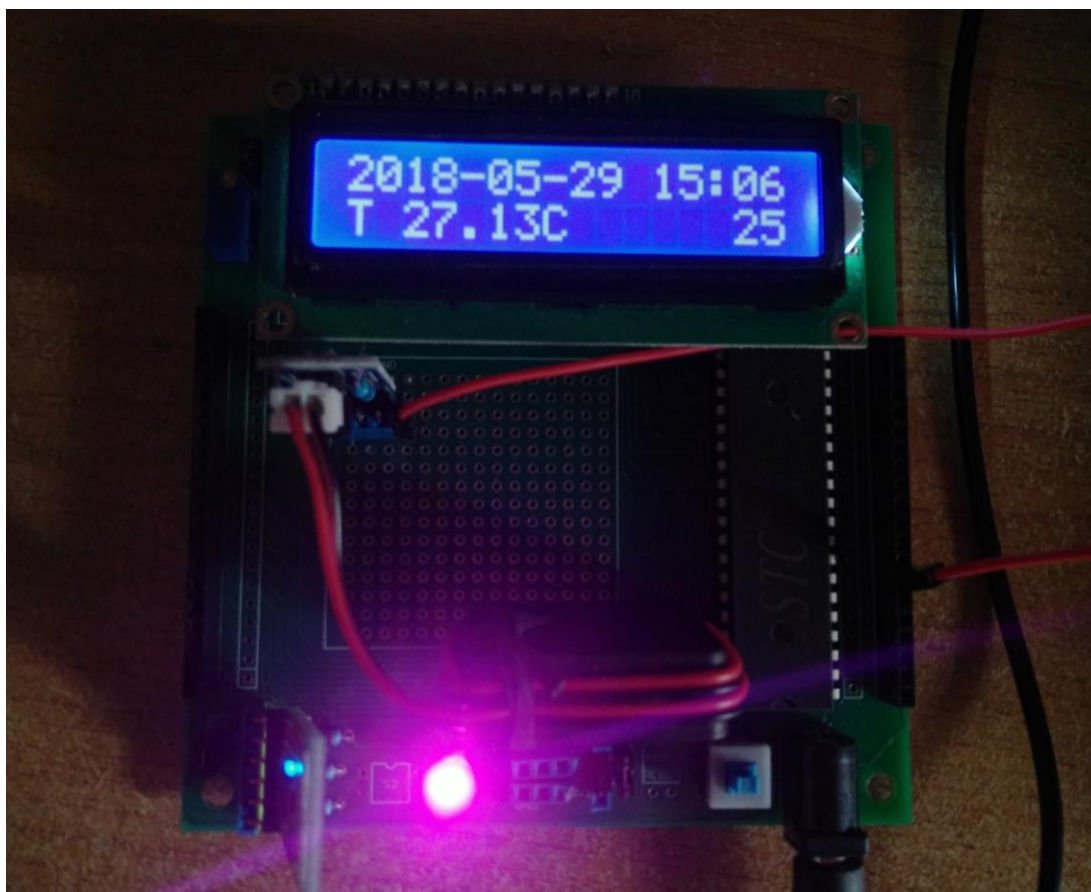
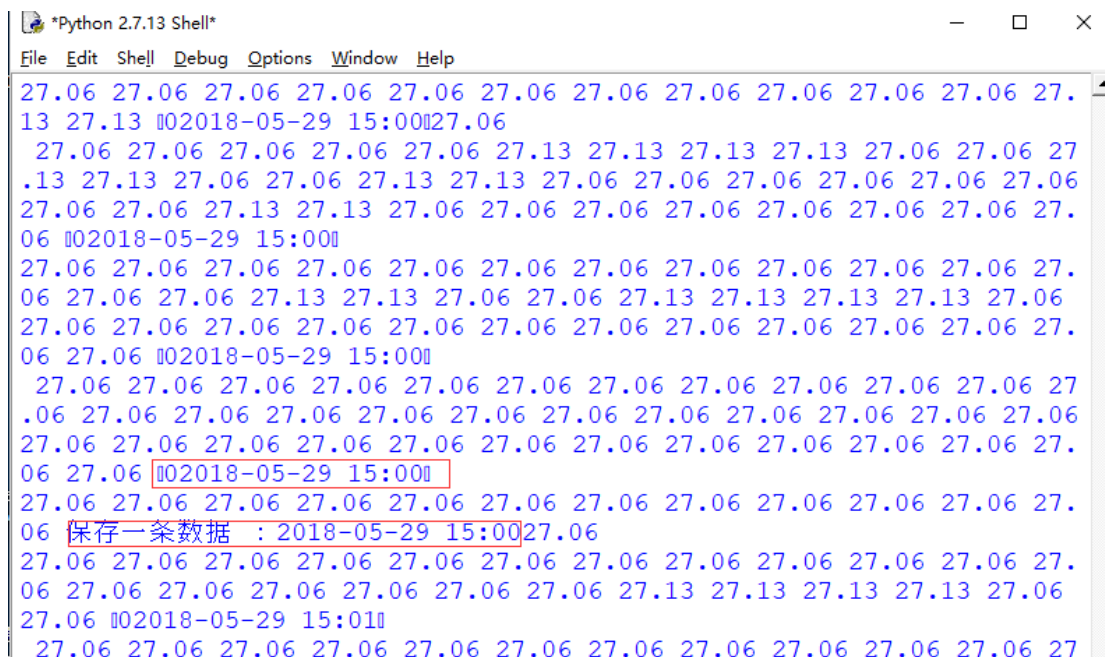


图 5.1 单片机系统运行效果

PC 端串口服务程序能正常接收温度信息以及温度系统状态，能正常发出控制指令给单片机，并且数据库连接正常，能正常读写数据库温度以及系统状态信息，如图 5.2 所示；



```
*Python 2.7.13 Shell*
File Edit Shell Debug Options Window Help
27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.
13 27.13 2018-05-29 15:0027.06
27.06 27.06 27.06 27.06 27.06 27.13 27.13 27.13 27.13 27.06 27.06 27
.13 27.13 27.06 27.06 27.13 27.13 27.06 27.06 27.06 27.06 27.06 27.06
27.06 27.06 27.13 27.13 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.
06 2018-05-29 15:00
27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.
06 27.06 27.13 27.13 27.06 27.06 27.13 27.13 27.13 27.13 27.06
27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.
06 2018-05-29 15:00
27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27
.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.
06 2018-05-29 15:00
27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.
06 保存一条数据 : 2018-05-29 15:0027.06
27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.
06 27.06 27.06 27.06 27.06 27.06 27.06 27.13 27.13 27.13 27.06
27.06 2018-05-29 15:01
27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27.06 27
```

图 5.2 PC 串口服务程序运行效果

最终 Web 程序也能正常工作，网页能正常浏览并控制温度系统，运行结果如图 5.3，5.4 所示



图 5.2 Web 自动调节模式运行效果

5.1.2 总结与分析

经过不懈的努力以及长期的制作，最后毕业设计圆满完成。通过本次设计，我学习到了很多东西，首先学会了电路原理图以及 PCB 的绘制，然后学会了使用单片机驱动 LCD1602，对 LCD1602 的工作方式有了更深入的理解与认识，学会了使用 DS18B20 温度传感器，对蓝牙通信有了更深入的理解，通过 PC 串口程序的编写，我对计算机系统有了更深入的认识，计算机之间数据的交换是那么的复杂又是那么的简单。本次设计顺利的完成是自己努力的结果，从最开始一堆的元器件到最后一整个的系统，从开始一堆的代码 BUG 到最后能顺利运行的系统，每一次的成功，内心都充满了喜悦。

5.2 展望未来

在完成基本要求温度测量并显示的前提下，对整个系统进行了功能扩展，首先是增加了蓝牙串口模块，使单片机能与 PC 进行无线通信，使得单片机系统能显示当前时间，在此基础上又增加了 MySQL 数据库，通过 Web 技术，可以使用网页监测和控制温度。但是由于时间仓促以及准备不充分，系统还有许多美中不足的地方，首先是电路板的美观性，由于前期制作过程中积极性不高，为了赶进度草草的设计出了电路板，在布局以及功能性上考虑不周全，导致了后期新增模块使得排线较多影响美观；其次，增加蓝牙串口起初是为了让系统能通过手机蓝牙也能进行温度操控，但由于时间限制，手机 APP 的制作并没有顺利完成，好的是现在手机能通过浏览访问网页来操控温度。我在后续的时间里争取完善系统，如增加手机蓝牙访问，增加自动变温调节等功能。

致谢

经过漫长的编写以及修改，最终顺利完成毕业设计，在论文完稿之际，谨对在本论文的撰写过程中，给予我帮忙的导师、同学和亲爱的家人，表示深深的感谢！首先要感谢我的导师刘飞航老师。在承担繁重的教学和工作任务的状况下，他主动关心我的毕设进展，督促着我顺利完成毕设。从论文的选题、开题报告的撰写、资料的查找，到结构的完善，都给予悉心指导，使我顺利成文。

另外，要感谢我的女朋友程涛，是她给了我灵感，在完成题目基本要求的前提下对系统功能进行了扩展，后期论文撰写是无聊而枯燥的，是她鼓励和监督了我，给了我动力让我坚持了下来，最终顺利完成了文章的撰写。

其次要感谢我的室友李凌冰，李杰，姚文，徐博凡，江之源，郭伟和王明明，是他们陪伴着我，在遇到不懂问题的时候是他们给与了我帮助和支持。

最后要感谢我的家人，是家人的鼓励和支持，使我能够全心投入学习和工作之中，顺利完成学业。最后衷心感谢在百忙之中评阅论文和参加答辩的各位专家、教授！

参考文献

- [1] C 程序设计（第三版）. 西安：西安电子科技大学出版社, 2012
- [2] 谭浩强, C 程序设计. 北京：清华大学出版社, 1999
- [3] 郭天祥, 51 单片机 C 语言教程. 北京：电子工业出版社, 2014. 11.
- [4] 张毅刚, 单片机原理及应用. 北京：高等教育出版社, 2008
- [5] Andrew S. Tanenbaum , David J. Wetherall. Computer Networks. 严伟, 潘爱民译. 北京：清华大学出版社, 2012
- [6] 彭伟, 单片机 C 语言程序设计实训 100 例—基于 8051+Proteus 仿真. 北京：电子工业出版社, 2010. 6
- [7] Wesley J. Chun , Python 核心编程（第二版）, 宋吉广译, 人民邮电出版社, 2018. 7
- [8] Martin C. Brown , Python 技术参考大全, 康博译, 北京：清华大学出版社, 2002
- [9] <https://www.cnblogs.com/wt11/p/6141225.html>
- [10] <https://www.cnblogs.com/lanchang/p/6921196.html>
- [11] <https://blog.csdn.net/u012734441/article/details/42047715>
- [12] https://blog.csdn.net/Zach_z/article/details/72784369
- [13] <http://www.docin.com/p-276425776.html>
- [14] <http://www.51hei.com/mcu/2210.html>
- [15] http://www.eeworld.com.cn/mcu/article_2018011537384.html
- [16] <https://wenku.baidu.com/view/0d714227b4daa58da0114a5f.html>

附录 B PCB 设计

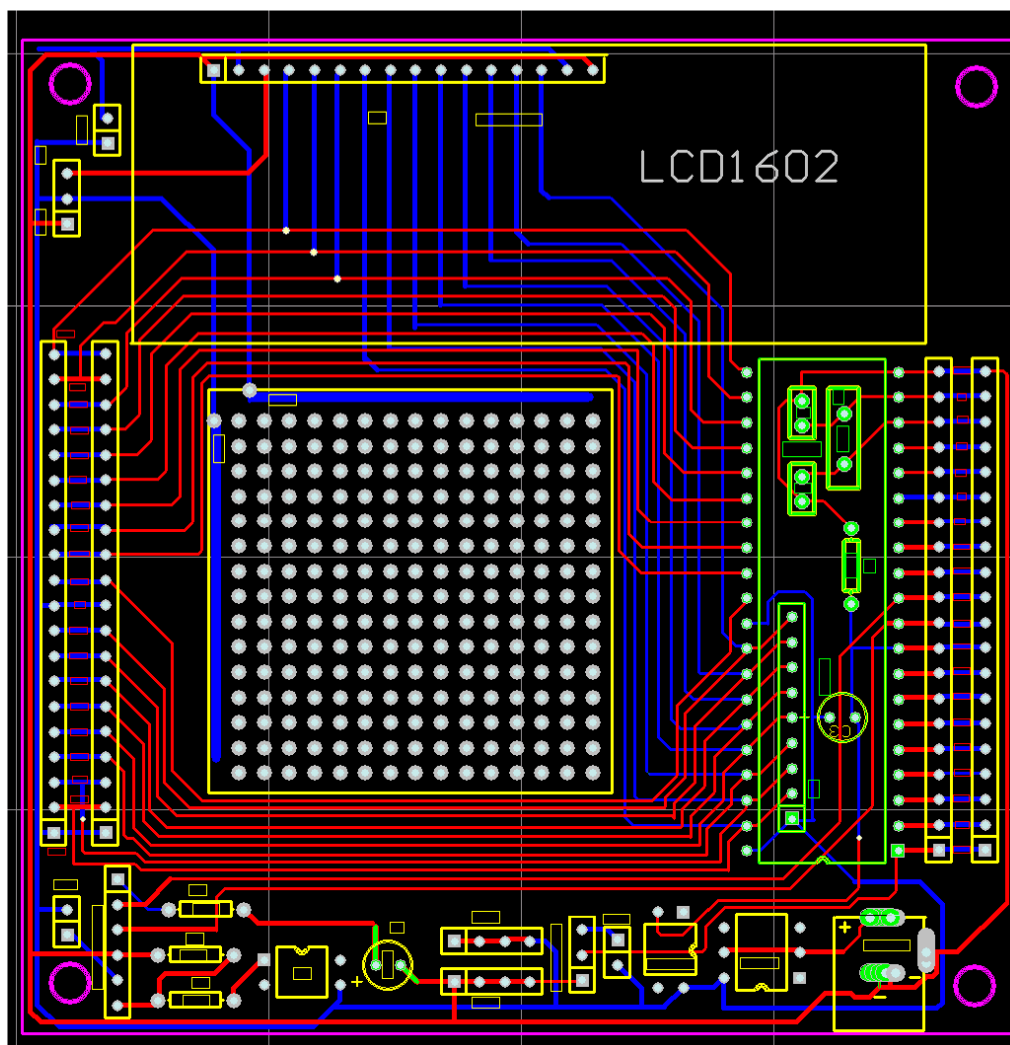


图 B1 PCB 设计图

附录 C 单片机程序清单

主程序:

```
//main.c

#include<reg51.h>
#include"lcd.h"
#include<string.h>
#include"temperature.h"
#include"bluetooth.h"

#define off 1
#define on 0

extern uchar flag;//是否有数据更新
extern uchar BluRecData[];//蓝牙接收到的数据帧
sbit RESET=P3^2;

uchar now_time[]={ "loading..." };
uchar *now_temp="Not ready"; //字符模式当前温度
uchar mode=1; //mode 为 0 自动模式，mode 为 1 手动模式
uchar up_val = 30;
uchar down_val = 20;
uchar stable = 27;

float num_temp; //数值模式当前温度
int TempDecThreshold = 45;//报警温度
void DataResolve(); //对蓝牙接受到的数据进行解析

main()
{ //初始化
    int i=0;
    RESET=0;
    LcdInit();
    BluInit();
```

```
warming=off;
cooling=off;
num_temp=DS18B20NumTemp();
now_temp=DS18B20toString(num_temp);
LcdDisplayStr(0,0,now_time);

while(1)
{

    if(flag==1)//如果有数据更新
    {
        DataResolve(); //处理数据
    }

    num_temp=DS18B20NumTemp();//获取当前温度值（float）
    now_temp=DS18B20toString(num_temp); //获取当前温度值（string）

    //显示时间
    LcdDisplayStr(0,1,"T ");
    LcdDisplayStr(2,1,now_temp);
    LcdDisplayChar(7,1,'C');

    //通过蓝牙发送时间
    BluSendByte(0x7F);
    BluSendByte('0');
    BluSendStr(now_temp);
    BluSendByte(0x7F);

    //显示稳定值或警戒值
    LcdDisplayChar(14,1,stable/10+'0');
```

```
LcdDisplayChar(15,1,stable%10+'0');

//now_temp = TempFormat(now_temp);
flag=0;//每趟循环结束，设置为无数据更新状态
Lcd_Delay1ms(500);
}
}

void DataResolve()
{
    uchar i;
//    static uchar tmp_time[17];
    i=1;
    if(BluRecData[0]=='0') //数据
    {
        while(BluRecData[i] != '\0')
        {
            now_time[i-1] = BluRecData[i];
            i++;
        }
        LcdDisplayStr(0,0,now_time);
    }
    else //指令
    {
        switch(BluRecData[1])
        {
            case '0': mode = 0; //自动模式
                        break;
            case '1': mode = 1; //手动模式
                        break;
```

```
        case '2': warming = off;      //关闭加热
                break;
        case '3': warming = on;        //开启加热
                break;
        case '4': cooling = off;        //关闭散热
                break;
        case '5': cooling = on;         //开启散热
                break;
        case '6': up_val= BluRecData[2]; //设置温度上限
                break;
        case '7': down_val = BluRecData[2]; //设置温度下限
                break;
        case '8': stable = BluRecData[2]; //设置稳定值
                break;
        default: break;
    }
}
}
```

DS18B20 服务程序:

```
// temperature.c
```

```
#include"temperature.h"
```

```
#include"stdio.h"
```

```
/******
```

```
*****
```

```
* 函数名          : Delay1ms
```

```
* 函数功能        : 延时函数
```

```
* 输入            : 延时时长, 单位 ms
```

```
* 输出            : 无
```

```
*****
```

```
*****/
```

```
void Delay1ms(uint c)
```

```
{
```

```
    uchar a,b;
```

```
    for (; c>0; c--)
```

```
    {
```

```
        for (b=199;b>0;b--)
```

```
        {
```

```
            for(a=1;a>0;a--);
```

```
        }
```

```
    }
```

```
}
```

```
/******
```

```
*****
```

```
* 函数名          : DS18B20Init
```

```
* 函数功能        : 初始化
```

* 输入 : 无

* 输出 : 初始化成功返回 1, 失败返回 0

*****/

uchar DS18B20Init()

```
{
    uint i;
    DSPORT=0;          //将总线拉低 480us~960us
    i=70;
    while(i--); //延时 642us
    DSPORT=1;          //然后拉高总线, 如果 DS18B20 做出反应会将在
15us~60us 后总线拉低
    i=0;
    while(DSPORT) //等待 DS18B20 拉低总线
    {
        i++;
        if(i>5000) //等待>5MS
            return 0; //初始化失败
    }
    return 1; //初始化成功
}
```

/*****

* 函数名 : DS18B20WriteByte

* 函数功能 : 向 18B20 写入一个字节

* 输入 : dat

* 输出 : 无

*****/

void DS18B20WriteByte(uchar dat)

```
{
    uint i,j;
    for(j=0;j<8;j++)
    {
        DSPORT=0;          //每写入一位数据之前先把总线拉低 1us
        i++;
        DSPORT=dat&0x01; //然后写入一个数据，从最低位开始
        i=6;
        while(i--); //延时 68us，持续时间最少 60us
        DSPORT=1; //然后释放总线，至少 1us 给总线恢复时间才能接着写入第
        二个数值
        dat>>=1;
    }
}
```

/*****

* 函数名 : DS18B20ReadByte

* 函数功能 : 读取一个字节

* 输入 : 无

* 输出 : 1 个字节数据

*****/

uchar DS18B20ReadByte()

```
{
    uchar byte,bi;
```

```

uint i,j;
for(j=8;j>0;j--)
{
    DSPORT=0;//先将总线拉低 1us
    i++;
    DSPORT=1;//然后释放总线
    i++;
    i++;//延时 6us 等待数据稳定
    bi=DSPORT;    //读取数据，从最低位开始读取
    /*将 byte 左移一位，然后与上右移 7 位后的 bi，注意移动之后移掉那位
补 0。*/
    byte=(byte>>1)|(bi<<7);
    i=4;    //读取完之后等待 48us 再接着读取下一个数
    while(i--);
}
return byte;
}

/*****
*****
* 函数名          : DS18B20TransfTemp
* 函数功能        : 让 18b20 开始转换温度
* 输入            : 无
* 输出            : 无
*****
*****/

void DS18B20TransfTemp()
{
    DS18B20Init();
    Delay1ms(1);

```

```

    DS18B20WriteByte(0xcc);    //跳过 ROM 操作命令
    DS18B20WriteByte(0x44);    //温度转换命令
    Delay1ms(100); //等待转换成功

}

/*****

*****

* 函数名          : DS18B20ReadTempCom
* 函数功能        : 发送读取温度命令
* 输入            : 无
* 输出            : 无

*****/

void DS18B20ReadTempCom()
{

    DS18B20Init();
    Delay1ms(1);
    DS18B20WriteByte(0xcc);    //跳过 ROM 操作命令
    DS18B20WriteByte(0xbe);    //发送读取温度命令
}

/*****

*****

* 函数名          : DS18B20ReadTemp
* 函数功能        : 读取温度
* 输入            : 无
* 输出            : 温度值数字量（实际温度的补码）

*****/

*****/

```

```

int DS18B20ReadTemp()
{
    int temp=0;
    uchar tmh,tml;
    DS18B20TransfTemp();    //先写入转换命令
    DS18B20ReadTempCom();    //然后等待转换完后发送读取温度命令
    tml=DS18B20ReadByte();    //读取温度值共 16 位，先读低字节
    tmh=DS18B20ReadByte();    //再读高字节
    temp=tmh;
    temp<<=8;
    temp|=tml;
    return temp;
}

/*****
*****
* 函数名      : DS18B20toString
* 函数功能    : 读取温度值并转化为浮点数（模拟量，保留两位小数）
* 输入        : 无
* 输出        : float 温度 （温度值模拟量，保留两位小数）
*****
*****/

float DS18B20NumTemp()
{
    float Atemp;
    int Dtemp=DS18B20ReadTemp();
    if(Dtemp< 0)    //当温度值为负数
    {

```

```

        //因为读取的温度是实际温度的补码，所以减 1，再取反求出原码
        Dtemp=Dtemp-1;
        Dtemp=~Dtemp;
        Atemp=Dtemp*0.0625;
    }
    else
    {
        //如果温度是正的那么，那么正数的原码就是补码它本身
        Atemp=Dtemp*0.0625;
    }
    return    Atemp;
}

/*****
*****
* 函数名          : DS18B20toString
* 函数功能        : 读取温度值并转化为字符串（模拟量，保留两位小数）
* 输入            : float 温度值
* 输出            : 字符串指针 （温度值模拟量，保留两位小数）
*****
*****/

uchar* DS18B20toString(float numtemp)
{
    int i=0,j=0;
    uchar temp_str[7]={'\0'};
    uchar test[7];
    int itemp= numtemp*100+0.5;
    if(itemp< 0)          //当温度值为负数
        temp_str[i]='-';
    if(itemp>=10000)

```

```

        temp_str[i++] = itemp / 10000+'0';    //百位
    if(itemp>=1000)
        temp_str[i++] = itemp % 10000 / 1000+'0'; //十位
    temp_str[i++] = itemp % 1000 / 100+'0';
    temp_str[i++] = '.';
    temp_str[i++] = itemp % 100 / 10+'0';
    temp_str[i++] = itemp % 10+'0';
    temp_str[i] = '\0';
    do
        //严格 C 语法不需要这部分，但 keil 好像存在
        一些问题
    {
        test[j]=temp_str[j];
        j++;
    }while(temp_str[j]!='\0');
    test[j]='\0';
    return test;
}

```

LCD1602 服务程序:

//lcd.c

#include"lcd.h"

```

/*****
*****
* 函 数 名      : Lcd1602_Delay1ms
* 函数功能      : 延时函数，延时 1ms
* 输    入      : c
* 输    出      : 无
* 说    名      : 该函数是在 12MHZ 晶振下，12 分频单片机的延时。
*****

```

*****/

void Lcd_Delay1ms(uint c) //误差 0us

```
{
    uchar a,b;
    for (; c>0; c--)
    {
        for (b=199;b>0;b--)
        {
            for(a=1;a>0;a--);
        }
    }
}
```

/******

* 函 数 名 : LcdWriteCom
 * 函数功能 : 向 LCD 写入一个字节命令
 * 输 入 : com
 * 输 出 : 无

*****/

void LcdWriteCom(uchar com) //写入命令

```
{
    LCD_E = 0;        //使能
    LCD_RS = 0;       //选择发送命令
    LCD_RW = 0;       //选择写入
```

```

LCD_DATAPINS = com;    //放入命令
Lcd_Delay1ms(1);      //等待数据稳定

LCD_E = 1;            //写入时序
Lcd_Delay1ms(5);      //保持时间
LCD_E = 0;
}

/*****
*****

* 函 数 名      : LcdWriteData
* 函数功能      : 向 LCD 写入一个字节的数据
* 输    入      : dat
* 输    出      : 无
*****
*****/

void LcdWriteData(uchar dat)    //写入数据
{
    LCD_E = 0; //使能清零
    LCD_RS = 1; //选择输入数据
    LCD_RW = 0; //选择写入

    LCD_DATAPINS = dat; //写入数据
    Lcd_Delay1ms(1);

    LCD_E = 1; //写入时序
    Lcd_Delay1ms(5); //保持时间
    LCD_E = 0;
}

```



```

/*****
*****

* 函 数 名          : LcdDisplayChar
* 函数功能          : 在 x,y 处显示单个字符
* 输    入          : X （列）  Y（行） dat（单个字符）
* 输    出          : 无

*****
*****/

```

```

void LcdDisplayChar(uchar X,Y ,uchar dat)
{
    if (Y) X += 0x40; //当要显示第二行时地址码+0x40;
    X += 0x80; // 算出指令码
    LcdWriteCom(X);
    LcdWriteData(dat);
}

```

```

/*****
*****

* 函 数 名          : LcdDisplayStr
* 函数功能          : 从 x,y 处开始显示字符串
* 输    入          : X （列）  Y（行） str 字符串指针
* 输    出          : 无

*****
*****/

```

```

void LcdDisplayStr(uchar X,Y ,uchar *str)
{

```

```

uchar index=0;
while (str[index]!='\0') //若到达字符串尾则退出
{
    LcdDisplayChar(X, Y, str[index++]); //显示单个字符并且字符标志位后移一位
    X++;
}
}

```

```

/*****

```

```

*****

```

```

* 函数名      : LcdInit()

```

```

* 函数功能    : 初始化 LCD 屏

```

```

* 输 入      : 无

```

```

* 输 出      : 无

```

```

*****

```

```

*****/

```

```

void LcdInit()                //LCD 初始化子程序

```

```

{
    LcdWriteCom(0x38); //开显示
    LcdWriteCom(0x0c); //开显示不显示光标
    LcdWriteCom(0x06); //写一个指针加 1
    LcdWriteCom(0x01); //清屏
    LcdWriteCom(0x80); //设置数据指针起点
}

```

蓝牙串口服务程序:

```

//bluetooth.c

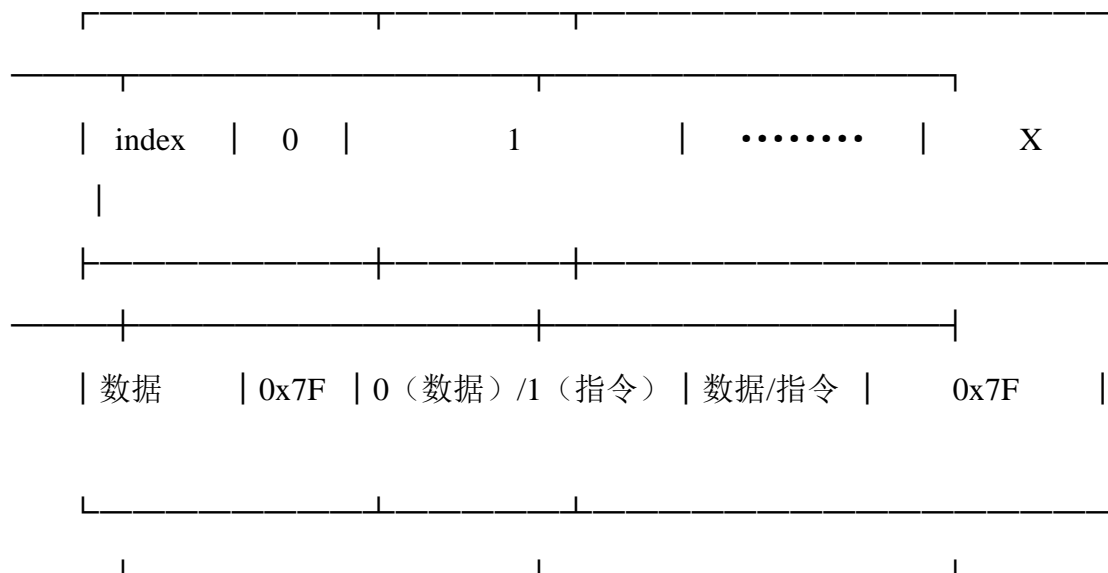
```

```

/*****

```

单片机接受数据帧格式



```

*****/

```

```

#include <bluetooth.h>

```

```

#define MAXSIZE 17 //定义数据帧最大接受/发送长度

```

```

uchar BluRecData[MAXSIZE];

```

```

uchar RecIndex=0;

```

```

uchar flag=0;//是否有数据更新

```

```

uchar isRec=0;//是否接收数据标志

```

```

/*****

```

```

*****

```

```

* 函数名          : BluInit

```

```

* 函数功能        : 初始化蓝牙模块 波特率 9600

```

```

* 输入            : 无

```

```

* 输出            : 无

```

```

*****

```

```

*****/

```

```
void BluInit(void)
```

```
{
    TMOD = 0x20;
    SCON = 0x50;
    TH1 = 0xFD;
    TL1 = TH1;
    PCON = 0x00;
    EA = 1;
    ES = 1;
    TR1 = 1;
}
```

```
/******
```

```
*****
```

```
* 函数名      : Delay1ms
* 函数功能    : 延时函数
* 输入        : 延时时长, 单位 ms
* 输出        : 无
```

```
*****
```

```
*****/
```

```
void Delay_1ms(uint c)
```

```
{
    uchar a,b;
    for (; c>0; c--)
    {
        for (b=199;b>0;b--)
        {
            for(a=1;a>0;a--);
        }
    }
}
```

```

    }
}

/*****
*****

* 函数名      : BluSendByte
* 函数功能    : 蓝牙发送单个字节
* 输入        : 无
* 输出        : 无
*****
*****/

void BluSendByte(uchar c)
{
    ES=0;                //关闭串口中断
    TI=0;                //清发送完毕中断请求标志位
    SBUF=c;              //发送
    while(TI==0);        //等待发送完毕
    TI=0;                //清发送完毕中断请求标志位
    ES=1;                //允许串口中断
    TH0=0;
    TL0=0;
}

/*****
*****

* 函数名      : BluSendStr
* 函数功能    : 蓝牙发送字符串
* 输入        : 字符指针 s
* 输出        : 无
*****
*****/

```

```
*****/
```

```
void BluSendStr(uchar *s)
```

```
{
    uchar i=0;
    uchar *str;
    str=s;
    while(str[i]!='\0')
    {
        BluSendByte(str[i]);
        Delay_1ms(5);
        i++;
    }
}
```

```
/******
```

```
*****
```

* 函数名 : BluReceive

* 函数功能 : 串口中断函数 负责接受蓝牙数据

* 输入 : 无

* 输出 : 无

```
*****
```

```
*****/
```

```
void BluReceive(void) interrupt 4
```

```
{
    if(RI)
    {
        RI = 0;
        if(SBUF==0x7F)
        {
```

```
//RecIndex=0 则是帧头，否则为帧尾
if(RecIndex!=0)//接受完一组数据后刷新 flag 并设置为不接收数据模式（数据不存入 BluRecData）
{
    flag=1;
    isRec=0;
    BluRecData[RecIndex]='\0';
}
else
{
    isRec=1;//设置为接收数据模式（数据存入 BluRecData）
    flag=0;//
}
RecIndex=0;//接受完一组数据和刚开始接受数据都将 RecIndex 置 0
}
else
{
    if(isRec)//若为数据接收模式则将数据存入 BluRecData
    {
        BluRecData[RecIndex]=SBUF;
        RecIndex++;
    }
}
}
else
    TI = 0;
}
```