硅湖职业技术学院毕业论文（设计）

题目 基于单片机的智能温控系统设计

年级 2013 级

专业 机电一体化技术

姓名 陈韦凯

学号 130000570

指导老师 潘世丽

2016 年 4 月 20 日

基于单片机的智能温控系统设计

陈韦凯

[摘要]：在日常生活中，温度是至关重要的物理量，在各个领域中温度的控制都有着重要的的意义。本篇论文主要介绍基于单片机的智能温控系统设计，其中着重介绍了智能温度检测的设计思路，主要的功能和原理，通过温度传感器的检测在LED数码管上实时地显示出来。论文分为系统总体分析，系统硬件设计，系统软件设计三个部分，总体分析是基础，系统硬件设计是重要部分，软件设计通过C语言实现。同时做出了实物，达到了控制要求。

[关键词]：单片机　温度传感器　数码管　C语言

一、引言

温度的采集测量在日常生活和工业生产都中具有非常重要的作用。温度作为外部实时环境条件，在工业实时控制，农业实时测量，科研方面乃至人们的平常生活都有着千丝万缕的联系。因此，与温度的采集、测量、显示、报警的研发在如今就显得十分重要。由单片机与温度传感器组成的检测温度的系统可广泛的应用于很多领域。随着现代人物质生活的不断提升,对外部环境检测的要求也逐渐的提高，通过单片机来控制温度毫无疑问是人类不断研究探索的目标，也给我们的生活带来了全新的体验。数字温度计就是一个颇具意义的应用，它充分利用单片机技术，本着为现代人提供更科学、更实效、更快捷的电子仪器为目的，最终朝着数字化，智能化、现代化方向发展。

单片机自上世纪70年代问世以来，硬件、软件方面的性能一直在不断的扩充和完善，其灵活多变的技术还能满足很多不同工作环境的需求。单片机以其集成度高、体积小、功耗低、速度快、功能强、价格便宜等优点，广泛应用于工业控制、数据采集和处理、仪器仪表、高级计算器、医疗器械、智能电器等领域。

该论文介绍的数字测温系统较传统的测温仪具有测量方便，读数精确、显示直观等优点，采用数字显示的温度输出方式，主要用于医疗实验室、现场过程控制等对温度要求比较高的场所。该设计所采用的 CPU 控制器是单片机AT89S51，数字温度传感器是DS18B20，输出显示采用共阳极四位LED八段数码管，通过串行口传输数据,实现温度的实时监控显示,从而达到预期的要

求。本文主要介绍基于 AT89S51 单片机控制的集成温度传感器 DS18B20 的工作特性、使用原理和编程方法，以及如何用单片机 AT89S51 对 DS18B20 的编程实现温度测量,有很强的学习价值和研究意义。

二、 设计方案

2.1 方案一

由于本设计针对测温电路，最常用的方法是使用电阻传感器——热敏电阻，充分利用其热敏效应，将温度的变化转换为电阻的变化，然后采用电桥等测量电路再将电阻的变化转换为电压或者电流，最后进行 A/D 转换，将温度的数值转换为数字量，再利用 AT89S51 单片机对数字量进行处理，最终控制数码管，进行温度的显示。这种设计方法需要用电桥电路、滤波电路、A/D 转换电路等，因此中间环节的感温电路设计较为复杂。

2.2 方案二

基于上述原因，所以最终方案利用数字温度传感器 DS18B20 来检测温度。根据调研，单片机的测温电路一般都是使用数字温度传感器，这与之前的热敏电阻相比相对简单。通过该传感器，可以直接将被测的温度模拟量转换为对应的数字量。通过两种方案对比可以看出，采用方案二，电路设计成本较低，硬件设计相对简单，软件编程较为成熟，故采用了方案二。

智能温控系统总体设计方框图如图 1 所示：

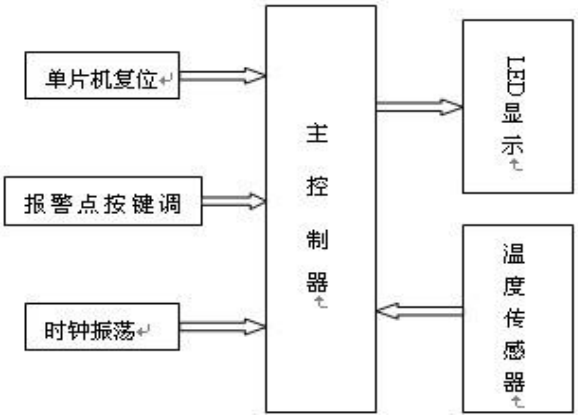

图 1 总体设计方框图

主控制器采用单片机 AT89S51，温度测量采用 DS18B20 数字传感器，温度显示用 4 位 LED 共阳极数码管（通过串行口传输数据）。

三、芯片介绍

3.1 主控制器

单片机芯片 AT89S51 具有较低电压供电、体积较小等特点，一般只需要 2 个端口（共 4 个端口）就可以完成一个简单系统的设计要求。它是一个功耗低，高性能 CMOS 8 位单片机。

89s51 是 89c51 的升级版本，89SXX 可以向下兼容 89CXX 等 51 系列芯片。其区别如下：

1、89S51 在工艺上进行了改进，89S51 采用 0.35 新工艺，成本降低,而且将功能提升,增加了竞争力。

2、新增加很多功能，性能有了较大提升。

3、89s51 有 ISP 在线编程功能，这个功能的优势在于改写单片机存储器内的程序不需要把芯片从工作环境中剥离。速度更快、稳定性更好，烧写电压也仅仅需要 4～5V 即可。

4、最高工作频率为 33MHz，89C51 的极限工作频率是 24M。

5、89s51 具有双工 UART 串行通道。

6、89s51 内部集成看门狗计时器，不再需要像 89C51 那样外接看门狗计时器单元电路。

7、89s51 带有双数据指示器。

8、89s51 带有电源关闭标识。

9、89s51 带有全新的加密算法，这使得对于 89S51 的破解变为不可能，程序的保密性大大加强，这样就可以有效的保护知识产权不被侵犯。

10、电源范围：89S5*电源范围宽达 4～5.5V，而 89C5*系列在低于 4.8V 和高于 5.3V 的时候则无法正常工作。

11、烧写寿命更长：89S5*标称的 1000 次，实际最少是 1000 次～10000 次，这样更有利初学者反复烧写，减低学习成本。

3.2 四位共阳数码管

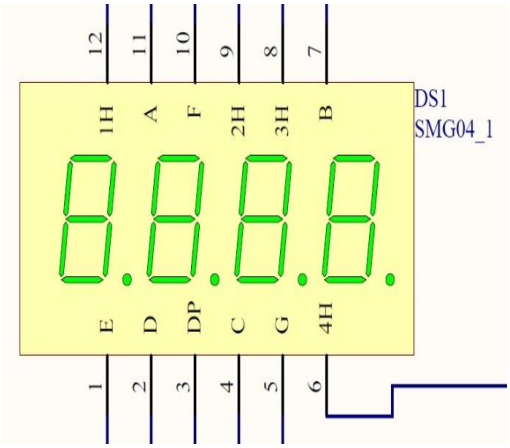

图 2 四位共阳数码管

四位数码管的引脚图如图 2 所示，其中 1、2、3、4H 就是 4 个 LED 的公共端，如果是共阳的话你想要第四个“日”亮，就给 4H 接电源的正极，A.B.C.D.E.F.G 就是这个“日”的各个笔划，叫做段选，1、2、3、4H 分别是位选。当位选打开时，送入相应的段码，则相应的数码管打开，关掉位选，打开另一个位选，送入相应的段码，则数码管打开，在本设计中为了到达视觉的角度上的同时显示，每次打开关掉相应的位选时，时间间隔低于 20ms。

3.3 温度传感器 DS18B20

美国 Dallas 半导体公司 1984 年成立，目前已发布了 360 多种专有的基本产品。主要应用于宽带通讯、无机、蜂窝基站、Internet 通讯、网络、服务器、数据存储器以及工业设备。

DS18B20 温度传感器是该公司发明的一种改进型智能温度传感器。DS18B20 是一种单总线数字温度传感器。测试温度范围-55℃-125℃，温度数据位可配置为 9、10、11、12 位，对应的刻度值分别为 0.5℃、0.25℃、0.125℃、0.0625℃，对应的最长转换时间分别为 93.75ms、187.5ms、375ms、750ms。出厂默认配置为 12 位数据，刻度值为 0.0625℃，最长转换时间为 750ms。从以上数据可以看出，DS18B20 数据位越低、转换时间越短、反应越快、精度越低。DS18B20 的性能特点如下表 1：

表 1 DS18B20 的性能特点

1、无须外部电路器件；
2、可通过数据线供电，电压范围为 3.0~5.5V；
3、用户可定义报警设置；
4、若要实现多点组网功能，可将多个 DS18B20 并联在唯一的三线上；
5、单线接口进行通信时仅需要一个端口引脚；
6、温度以 9 或 1 2 位数字；
7、电源极性若接反，温度计不会因发热而烧毁，但不能正常工作，
8、零待机功耗；

DS18B20 采用 3 脚 PR－35 封装或 8 脚 SOIC 封装，其内部结构框图如图 3 所示。DS18B20 温度传感器的主要由一个高速暂存ＲＡＭ、一个非易失性的可电擦除的 EERAM 组成。高速暂存 RAM 是一个 8 字节的存储器，结构见图 4：包含测得的温度信息的为头 2 个字节，ＴＨ和ＴＬ的拷贝为第 3 和第 4 字节，是易失的，每次上电复位时被刷新。第 5 个字节，为配置寄存器，它的内容确定了温度值的数字转换分辨率，。

DS18B20 工作时寄存器中的分辨率转换为相应精度的温度数值。DS18B20 字节定义见图 4， DS18B20 是工作模式还是在测试模式由ＴＭ来确定，R1 和 R0 的组合决定了温度转换的精度,可以设置分辨率，DS18B20 出厂时该位被设置为 0 。

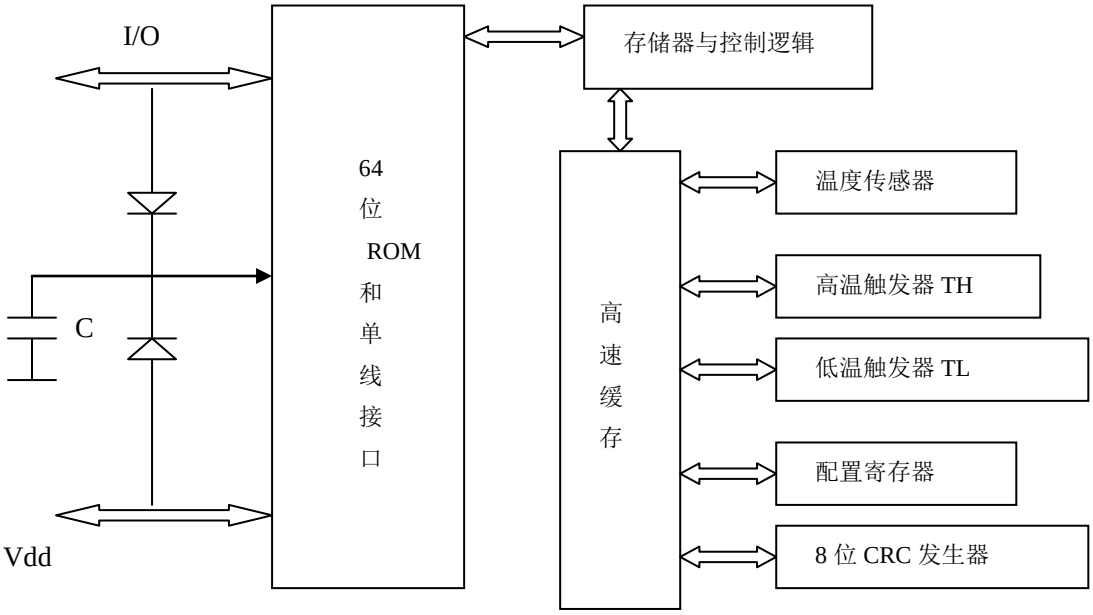

图 3 DS18B20 内部结构

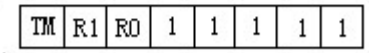

图 4 DS18B20 字节定义

通过表 2 发现，DS18B20 温度转换时间随着分辨率的提高而延长。由图 4 可知高速暂存可读存储器的第 6 、7 、8 字节处于保留状态，逻辑上全为 1 。第 9 字节通过读取前 8 字节的 CRC 码，负责检验数据，目的保证了数据传输的正确性。

表 2 DS18B20 温度转换表

R1	R0	分辨率/位	温度最大转向时间/ms
0	0	9	93.75
0	1	10	187.5
1	0	11	375
1	1	12	750

当 DS18B20 接收到温度转换命令后开始工作。转换结束后，检测的温度值以 16 位带符号扩展的二进制补码形式存储在高速暂存ＲＡＭ的第 1 、2 字节。单片机接收到信号后，通过单线接口读出 16 位数据，读数据时先是低八位，然后高八位，数据格式采用 0.0625℃／LSB 的形式。

当符号位 S＝ 0 时，代表测得的温度值为正值，可以直接将二进制位转换为十进制；当符号位 S＝ 1 时，代表测得的温度值为负值，要先将补码变成原码，再进行十进制计算。部分温度值所对应的十六进制数据见表 3。

表 3 温度对应值表

温度/℃	二进制表示	十六进制表示
+125	0000 0111　1101 0000	07D0H
+85	0000 0101　0101 0000	0550H
+25.0625	0000 0001　1001 0000	0191H
+10.125	0000 0000　1010 0001	00A2H
+0.5	0000 0000　0000 0010	0008H
0	0000 0000　0000 1000	0000H
-0.5	1111 1111　1111 0000	FFF8H
-10.125	1111 1111　0101 1110	FF5EH
-25.0625	1111 1110　0110 1111	FE6FH
-55	1111 1100　1001 0000	FC90H

当 DS18B20 将温度转换完成之后，然后把检测到的温度值与 RAM 中的 TH、TL 字节中的内容作比较。若 T ＞TH 或 T＜TL，则主机发出的报警搜索命令，同时置位器件内的报警标志位。

若用多只 DS18B20 同时挂在总线上，进行温度测量并报警搜索，即可实现快速实时的显示。循环冗余检验码（CRC）存储在 64 位 ROM 的最高有效字节中。CPU 的 ROM 的前 56 位来计算 CRC 值，并和存入 DS18B20 的 CRC 值作比较，以判断主机收到的 ROM 数据是否正确。

DS18B20 检测温度的原理如下：温度对低温度系数晶振的 fosc 的影响微

乎其微，因此减法计数器 1 接收其产生的固定频率的脉冲信号；高温度系数晶振随温度变化也发生变化，在温度的变化下其振荡频率也产生了明显改变，减法计数器 2 将其产生的信号作为脉冲输入。

DS18B20 中的计数门负责对低温度系数振荡器所产生的时钟脉冲进行计数达到温度测量的目的，而计数门的开启时间则由高温度系数振荡器来控制。在进行测量前，－55℃所对应的一个基数值必须分别置入减法计数器 1 和温度寄存器中。

低温度系数的晶振所产生的脉冲信号由减法计数器 1 进行减法计数，当一个脉冲信号过来，减法计数器 1 就减 1，直到其预置值减到 0 ，这时温度寄存器的值累计加 1 ，同时将预置值重新装入减法计数器 1 ，然后又开始对脉冲信号进行计数。如此循环，一直到减法计数器 2 计数为 0 时，这时温度寄存器值停止累加，那么存在温度寄存器中的当前数值就是所测温度值。

同时，读写时序很重要。DS18B20 具有单线通信功能，分时完成各种任务。系统按协议对 DS18B20 进行各种操作。操作协议如下流程图 5：

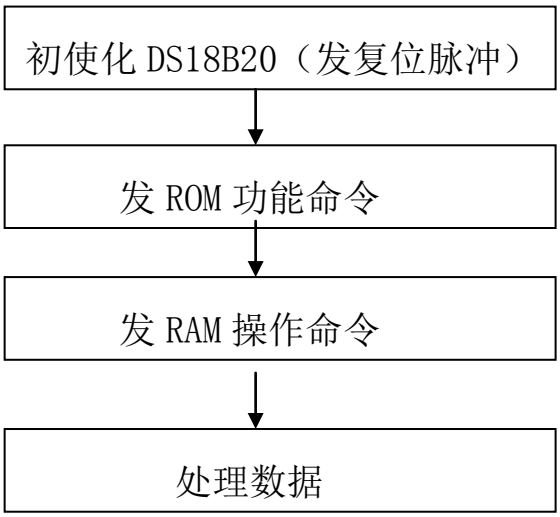

图 5 操作协议流程图

3.4 三极管 8550

采用三极管 8550 来驱动 4 位数码管与报警器，不仅操作简单，而且成本相对低。三极管在电路中起电流放大器的作用，每个三极管有三个极，分别叫做集电极 C，基极 B，发射极 E。根据电流方向和电压正负的不同将三极管分为 NPN 和 PNP 两种类型。在本设计中采用的是 PNP 三极管的共发射极放大电路。三极管驱动引脚如图 6。

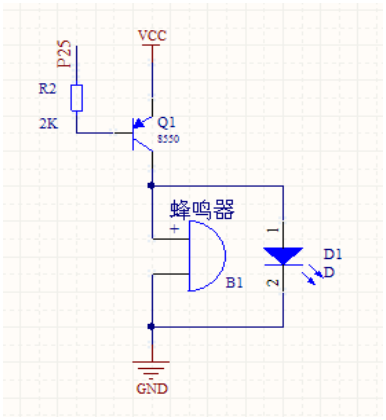

图 6 蜂鸣器、发光三极管驱动引脚图

三极管的放大作用：当基极电压 Ub 有一个微小的变化时，基极电流 Ib 也会随之有一定的变化，受基极电流 Ib 的控制，集电极电流 Ic 会有一个很大的变化，基极电流 Ib 越大，集电极电流 Ic 也越大，反之，基极电流越小，集电极电流也越小，即基极电流控制集电极电流的变化。

由于数码管是由多段发光二极管组成，当多数数码段都点亮后需要很大的电流才可以，为了目测更清楚，那么就需要借助三极管的电流放大原理来驱动发光二极管工作，否则由于电流分流太大而使得数码管总亮度降低，变暗。在这个设计中三极管起着位驱动的作用，驱动电路的接线图如图 7。

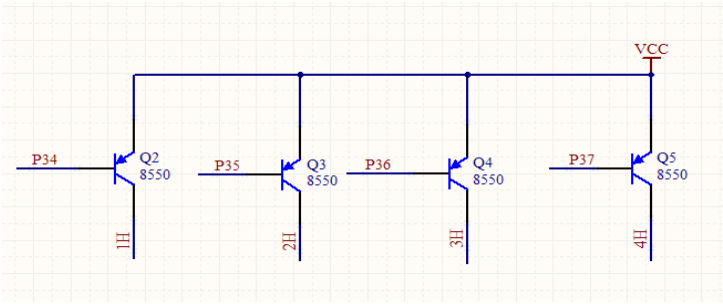

图 7 驱动电路

四、硬件电路

系统整体的硬件电路框图见下图 8：

4.1 DS18B20 与单片机的接口电路

见图 9。

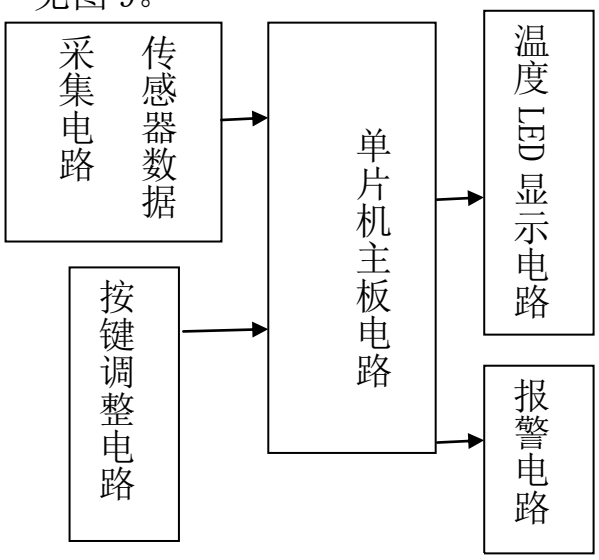

图 8 系统硬件电路框图

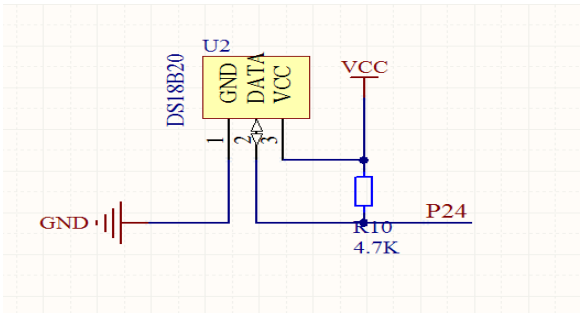

图 9 温度传感器电路引脚图

4.2 主板电路

图 10 中三个独立式按键的设置用来调整温度计的上限和下限报警，当被测温度不在设置的范围内时，单片机控制蜂鸣器发出鸣叫声音同时发光二极管闪烁，LED 数码管实时显示当前温度值。图 10 中的按健复位电路具有上电复位和手动复位功能，使用灵活，在程序进入死循环时，不必重起单片机电源，可以手动实现复位。

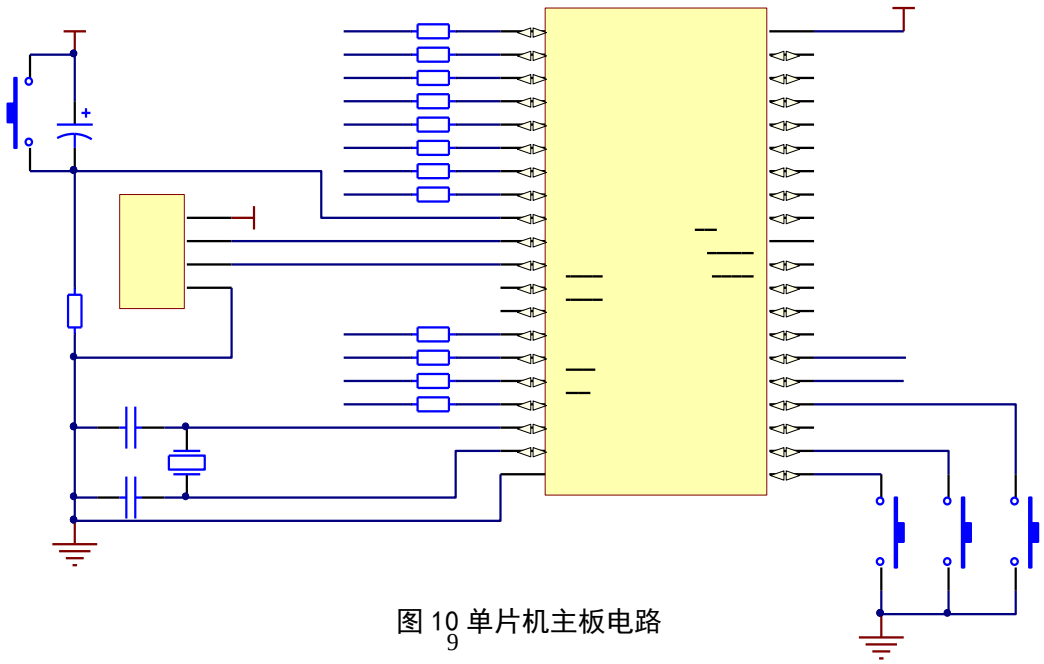

图 10 单片机主板电路

4.3 显示电路

显示电路如图 11 所示，采用四位数码管动态显示，1234 作为位选码，轮流被选中，ABCDEFG 作为段码显示相应的数字。当第一个位选打开时，送入相应的段码，则相应的数码管显示，然后关掉该位选，打开另一个位选，输送相应的段码，则下一个数码管显示，以此循环。每次打开关掉相应的位选时，时间间隔低于 20ms，由于人眼的视觉暂留效应，看到的是全部数码管同时显示。

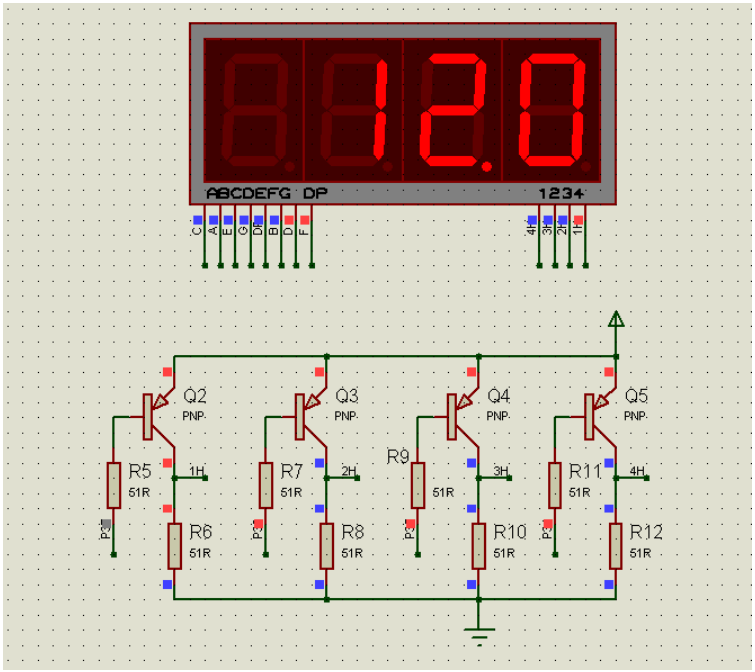

图 11 温度显示电路

4.4 实物图

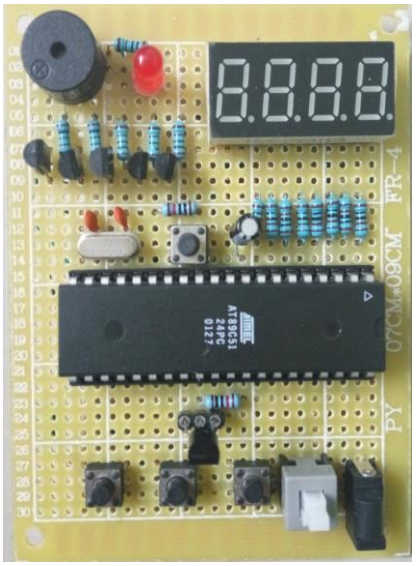

图 12 实物图

五、系统软件控制

5.1 程序结构分析

主程序调用了 3 个子程序，分别是温度信号处理程序、数码管显示程序、按键设定温度报警程序。温度信号处理程序功能：对温度传感器转换来的数据进行判断、处理。数码管显示程序功能：控制 LED 数码管的温度实时显示。按键设定温度报警程序功能：设定低温和高温报警界限值，可精确到 0.1 度。

5.2 系统程序流程图

主程序的主要任务是处理 DS18B20 的当前温度值，负责温度的实时采集、处理、显示。一秒的时间内测量一次被测温度，读出该测量值并与按键设定的上下限报警温度比较，图 13 为其程序流程图。

通过调用读温度的子程序，把存入 RAM 中温度值的整数部分与小数部分通过求余求整的算法，分开存放在两个不同的单元中，然后通过调用显示子程序把当前温度值显示出来。

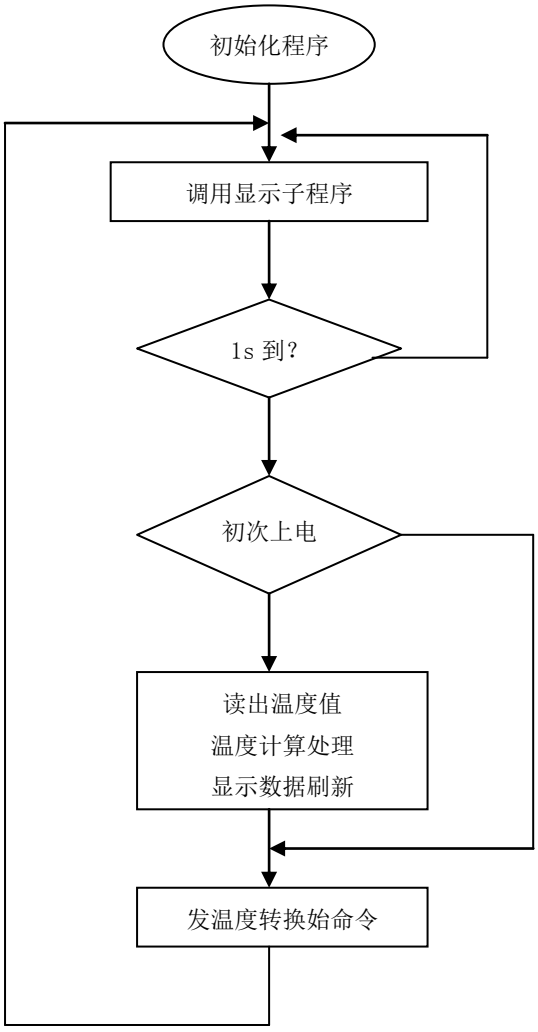

图 13 DS18B20 温度流程图

5.3 DS18B20 初始化程序流程图

在 DS18B20 工作之前需要进行初始化，图 14 为其流程图：

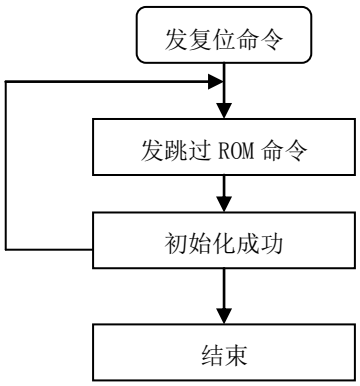

图 14 初始化程序流程图

```
/**********************18b20 初始化函数*****************************/
void init_18b20()
{
  bit q;
  dq = 1;               //把总线拿高
  delay_uint(1);        //15us
  dq = 0;               //给复位脉冲
  delay_uint(80);       //750us
  dq = 1;               //把总线拿高 等待
  delay_uint(10);       //110us
  q = dq;               //读取 18b20 初始化信号
  delay_uint(20);       //200us
  dq = 1;               //把总线拿高 释放总线
}
```

5.4 读温度子程序流程图

读温度子程序的主要任务是：从数字温度传感器 DS18B20 中读出温度数据，移入温度暂存器保存。图 15 为 其流程图：

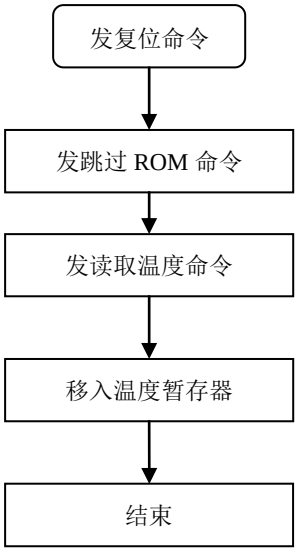

图 15 温度子程序流程图

```
/*************读取 18b20 内的数据***************/
uchar read_18b20()
{
  uchar i,value;
  for(i=0;i<8;i++)
  {
      dq = 0;             //把总线拿低读时间隙开始
      value >>= 1;    //读数据是低位开始
      dq = 1;             //释放总线
      if(dq == 1)         //开始读写数据
         value |= 0x80;
      delay_uint(5);      //60us     读一个时间隙最少要保持 60us 的时间
  }
  return value;       //返回数据
}
```

六、系统设计说明

该系统通过温度传感器 DS18B20 采集数据之后，将采集到的数据读取出来，将读取的数据和设定的温度上限和温度下限相比较，当超过上限或低于下限时启用报警功能，实时地将温度显示在数码管上。由于每个段位的切换时间缩很短从而达到人眼看上去 4 位同时显示。在附录中有程序的详细介绍。

七、总结

本系统基于 DS18B20 检测温度的原理，以 AT89S51 为控制核心，利用 1 个数字式集成温度传感器 DS18B20 ，通过单根总线和单片机进行连接，整个系统由温度上限报警电路、温度下限报警电路、和看门狗电路以及时钟电路组成，从而构成温度采集、处理及监控电路。温度采集测量系统具有很强的实用性、可靠性，结构简单、外围电路少，价格低廉，能够迅速地采集输送回多路温度信号。

考虑到系统在硬件上可以得到更加广阔的影响力及扩展性，通过硬件上的改造及软件上的完善，可以增加许多功能：比如将多个 DS18B20 传感器挂在总线上，可以实现更多测量点的温度巡回测量监控。这样的话使用的场合就更加广泛了,比如：作为一个监测仪器单独来用，应用于蔬菜大棚研究温度的高低对农作物生长的影响；工业环境现场不同工位的实时温度；亦可成为智能检测系统的一个分支系统，配合其它设备（上位机）一同工作。

八、心得体会

经过这次单片机毕业设计，终于完成了我的数字温度计的设计，虽然没有完全达到设计要求，但从心底里说，还是高兴的，毕竟这次设计把实物都做了出来，高兴之余不得不深思呀！

在做毕业设计的过程中，我发现虽然之前做过类似的电子小制作，也编写过几次程序，但之前所用算法少之又少。在这次毕业设计过程中，我通过自己上网查阅资料、设计电路原理图，购买元器件，亲手焊接调试，进入单片机论坛请教前辈编程设计的问题，在一定意义上完成了一个小的实用性项目。

在调试的时候，刚开始的时候板子还是不能正常的运行起来，后来通过请教老师和前辈了解到有可能是焊接上还存在的一定的问题，通过自己的慢慢的检查调整，最终板子成功的运行了起来。通过本次的设计，同时也真正的意识到，看似不起眼的一个元器件在电路中也起着重大的作用。在以后的工作中，我要继续锻炼自己的程序编写能力，多多学习算法这才是编程的核心所在。

【参考文献】

[1]李朝青.单片机原理及接口技术（简明修订版）[M].北京航空航天大学出版社:1998,11-15

[2]李广弟.单片机基础[M].北京航空航天大学出版社:1994,23-25

[3]陈京培.徐永梅.基于AT89S52单片机的液晶显示[J].现代电子技术,2008,31(22):22-25.

[4]高云红.数字温度传感器在多点温度测量系统中的应用[J].沈阳航空工业学院学报,2006,23(2):61-63

附录 A（程序清单）

程序采用单片机C语言编写，程序如下：

```
#include <reg51.h>
#include "eepom51.h"
#define uchar unsigned char
#define uint  unsigned int

//数码管段选定义      0     1     2     3     4     5  6     7     8     9
uchar code smg_du[]={0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8,0x80,0x90,
                0x88,0x83,0xc6,0xa1,0x86,0x8e,0xff};   //断码
//数码管位选定义
uchar code smg_we[]={0xef,0xdf,0xbf,0x7f};
uchar dis_smg[8] = {0xc0,0xf9,0xa4,0xb0,0x99,0x92,0x82,0xf8};
uchar smg_i = 3;     //显示数码管的个位数
sbit dq   = P2^4;  //18b20 IO 口的定义
sbit beep = P2^5;   //蜂鸣器 IO 口定义
uchar a_a;

uint temperature ;  //
bit flag_300ms ;
uchar key_can;       //按键值的变量
uchar menu_1;         //菜单设计的变量
uint t_high = 300,t_low = 100;
bit flag_lj_en;      //按键连加使能
bit flag_lj_3_en;    //按键连 3 次连加后使能 加的数就越大了
uchar key_time,flag_value;       //用做连加的中间变量
bit key_500ms  ;
uchar flag_clock;
uchar zd_break_en,zd_break_value;    //自动退出设置界面


/************************1ms 延时函数*****************************/
void delay_1ms(uint q)
{
  uint i,j;
  for(i=0;i<q;i++)
      for(j=0;j<120;j++);
}

/************************小延时函数*****************************/
void delay_uint(uint q)
{
  while(q--);
}
```

```
/**********************数码显示函数*****************************/
void display()
{
  uchar i;
  for(i=0;i<smg_i;i++)
  {
     P3 = smg_we[i];           //位选
     P1 = dis_smg[i];      //段选
     delay_1ms(1);
     P3 = 0xff;         //位选
     P1 = 0xff;         //消隐

  }
}

/******************把数据保存到单片机内部 eepom 中******************/
void write_eepom()
{
  SectorErase(0x2000);
  byte_write(0x2000, t_high % 256);
  byte_write(0x2001, t_high / 256);
  byte_write(0x2002, t_low % 256);
  byte_write(0x2003, t_low / 256);
  byte_write(0x2055, a_a);
}

/******************把数据从单片机内部 eepom 中读出来****************/
void read_eepom()
{
  t_high  = byte_read(0x2001);
  t_high <<= 8;
  t_high  |= byte_read(0x2000);
  t_low   = byte_read(0x2003);
  t_low <<= 8;
  t_low   |= byte_read(0x2002);
  a_a      = byte_read(0x2055);
}

/**********************18b20 初始化函数*****************************/
void init_18b20()
{
  bit q;
  dq = 1;              //把总线拿高
  delay_uint(1);        //15us
  dq = 0;              //给复位脉冲
```

```
  delay_uint(80);       //750us
  dq = 1;               //把总线拿高 等待
  delay_uint(10);       //110us
  q = dq;               //读取 18b20 初始化信号
  delay_uint(20);       //200us
  dq = 1;               //把总线拿高 释放总线
}

/*************写 18b20 内的数据***************/
void write_18b20(uchar dat)
{
  uchar i;
  for(i=0;i<8;i++)
  {                  //写数据是低位开始
     dq = 0;            //把总线拿低写时间隙开始
     dq = dat & 0x01; //向 18b20 总线写数据了
     delay_uint(5);     // 60us
     dq = 1;            //释放总线
     dat >>= 1;
  }
}

/*************读取 18b20 内的数据***************/
uchar read_18b20()
{
  uchar i,value;
  for(i=0;i<8;i++)
  {
     dq = 0;            //把总线拿低读时间隙开始
     value >>= 1;   //读数据是低位开始
     dq = 1;            //释放总线
     if(dq == 1)        //开始读写数据
        value |= 0x80;
     delay_uint(5);     //60us    读一个时间隙最少要保持 60us 的时间
  }
  return value;      //返回数据
}

/*************读取温度的值 读出来的是小数***************/
uint read_temp()
{
  uint value;
  uchar low;           //在读取温度的时候如果中断的太频繁了，就应该把中断给关
了，否则会影响到 18b20 的时序
  init_18b20();        //初始化 18b20
  write_18b20(0xcc);      //跳过 64 位 ROM
```

```
  write_18b20(0x44);       //启动一次温度转换命令
  delay_uint(50);          //500us

  init_18b20();         //初始化 18b20

  write_18b20(0xcc);       //跳过 64 位 ROM
  write_18b20(0xbe);       //发出读取暂存器命令

  EA = 0;
  low = read_18b20();      //读温度低字节
  value = read_18b20();  //读温度高字节
  EA = 1;
  value <<= 8;          //把温度的高位左移 8 位
  value |= low;         //把读出的温度低位放到 value 的低八位中
  value *= 0.625;           //转换到温度值 小数
  return value;         //返回读出的温度 带小数
}

/*************定时器 0 初始化程序***************/
void time_init()
{
  EA   = 1;        //开总中断
  TMOD = 0X01;     //定时器 0、定时器 1 工作方式 1
  ET0  = 1;        //开定时器 0 中断
  TR0  = 1;        //允许定时器 0 定时
}

/****************独立按键处理函数************************/
void key()
{
  static uchar key_new = 0,key_old = 0,key_value = 0;
  if(key_new == 0)
  {                        //按键松开的时候做松手检测
     if((P2 & 0x0f) == 0x0f)
        key_value ++;
     else
        key_value = 0;
     if(key_value >= 10)
     {
        write_eepom();
        key_value = 0;
        key_new = 1;
        flag_lj_en = 0;       //关闭连加使能
        flag_lj_3_en = 0; //关闭 3 秒后使能
        flag_value = 0;       //清零
     }
```

```
  }
  else
  {
     if((P2 & 0x0f) != 0x0f)
        key_value ++;  //按键按下的时候
     else
        key_value = 0;
     if(key_value >= 7)
     {
        key_value = 0;
        key_new = 0;
        flag_lj_en = 1;    //连加使能
        zd_break_en = 1; //自动退出设置界使能
        zd_break_value = 0; //自动退出设置界变量清零
     }
  }
  key_can = 20;
  if(key_500ms == 1)
  {
     key_500ms = 0;
     zd_break_en = 1; //自动退出设置界使能
     zd_break_value = 0; //自动退出设置界变量清零
     key_new = 0;
     key_old = 1;
  }
  if((key_new == 0) && (key_old == 1))
  {
     switch(P2 & 0x0f)
     {
        case 0x0e: key_can = 4; break;     //得到k1键值
        case 0x0d: key_can = 3; break;     //得到k2键值
        case 0x0b: key_can = 2; break;     //得到k3键值
        case 0x07: key_can = 1; break;     //得到k4键值
     }
  }
  key_old = key_new;
}

/****************按键处理数码管显示函数***************/
void key_with()
{
  if(key_can == 4)
  {
     menu_1 ++;
     if(menu_1 >= 3)
     {
```

```
            menu_1 = 0;
        }
        if(menu_1 == 0)
        {
            dis_smg[0] = smg_du[temperature % 10]; //取温度的小数显示
            dis_smg[1] = smg_du[temperature / 10 % 10] & 0x7f; //取温度的个位显
示
            dis_smg[2] = smg_du[temperature / 100 % 10] ;    //取温度的十位显示
            smg_i = 3;
        }
        if(menu_1 == 1)
        {
            dis_smg[0] = smg_du[t_high % 10];              //取小数显示
            dis_smg[1] = smg_du[t_high / 10 % 10] & 0x7f;  //取个位显示
            dis_smg[2] = smg_du[t_high / 100 % 10] ;     //取 low 十位显示
            dis_smg[3] = 0x89;
            smg_i = 4;
        }
        if(menu_1 == 2)
        {
            dis_smg[0] = smg_du[t_low % 10];               //取 low 小数显示
            dis_smg[1] = smg_du[t_low / 10 % 10] & 0x7f;  //取个位显示
            dis_smg[2] = smg_du[t_low / 100 % 10] ;      //取十位显示
            dis_smg[3] = 0xc7;
            smg_i = 4;
        }
    }
    if(menu_1 == 1)          //设置高温报警
    {
        if(key_can == 3)
        {
            if(flag_lj_3_en == 0)
                t_high ++ ;       //按键按下未松开自动加三次
            else
                t_high += 10; //按键按下未松开自动加三次之后每次自动加 10
            if(t_high > 990)
                t_high = 990;
            dis_smg[0] = smg_du[t_high % 10];              //取小数显示
            dis_smg[1] = smg_du[t_high / 10 % 10] & 0x7f;  //取个位显示
            dis_smg[2] = smg_du[t_high / 100 % 10] ;     //取十位显示
            dis_smg[3] = 0x89;    //H
        }
        if(key_can == 1)
        {
            if(flag_lj_3_en == 0)
                t_high -- ;       //按键按下未松开自动加三次
```

```
            else
                t_high -= 10; //按键按下未松开自动减三次之后每次自动减 10
            if(t_high <= t_low)
                t_high = t_low + 1;
            dis_smg[0] = smg_du[t_high % 10];              //取小数显示
            dis_smg[1] = smg_du[t_high / 10 % 10] & 0x7f;  //取个位显示
            dis_smg[2] = smg_du[t_high / 100 % 10] ;      //取十位显示
            dis_smg[3] = 0x89;    //H
        }
//        write_eepom();
    }
    if(menu_1 == 2)          //设置低温报警
    {
        if(key_can == 3)
        {
            if(flag_lj_3_en == 0)
                t_low ++ ;
            else
                t_low += 10;
            if(t_low >= t_high)
                t_low = t_high - 1;
            dis_smg[0] = smg_du[t_low % 10];              //取小数显示
            dis_smg[1] = smg_du[t_low / 10 % 10] & 0x7f;  //取个位显示
            dis_smg[2] = smg_du[t_low / 100 % 10] ;      //取十位显示
            dis_smg[3] = 0xc7;    //L
        }
        if(key_can == 1)
        {
            if(flag_lj_3_en == 0)
                t_low -- ;
            else
                t_low -= 10;
            if(t_low <= 10)
                t_low = 10;
            dis_smg[0] = smg_du[t_low % 10];              //取小数显示
            dis_smg[1] = smg_du[t_low / 10 % 10] & 0x7f;  //取个位显示
            dis_smg[2] = smg_du[t_low / 100 % 10] ;      //取十位显示
            dis_smg[3] = 0xc7;     //L
        }
//        write_eepom();
    }
}

/***************报警函数***************/
void clock_h_l()
{
```

```
  if((temperature <= t_low) || (temperature >= t_high))
  {
      flag_clock = 1;
  }
  else
  {
      flag_clock = 0;
      beep = 1;
  }
}
void main()
{
  temperature = read_temp();      //先读出温度的值
  time_init();                    //初始化定时器
  read_eepom();
  if(a_a == 0xff)                 //新的单片机初始单片机内问 EEPOM
  {
      t_high = 300;
      t_low  = 100;
      a_a = 1;
      write_eepom();
  }
  delay_1ms(650);
  temperature = read_temp();              //先读出温度的值
  dis_smg[0] = smg_du[temperature % 10]; //取温度的小数显示
  dis_smg[1] = smg_du[temperature / 10 % 10] & 0x7f; //取温度的个位显示
  dis_smg[2] = smg_du[temperature / 100 % 10] ;    //取温度的十位显示
  while(1)
  {
      display();           //显示函数
      key();               //按键程序
      if(key_can < 10)
      {
          key_with();          //设置报警温度
      }
      temperature = read_temp();  //先读出温度的值
      if(flag_300ms == 1)          //300ms 处理一次温度程序
      {
          clock_h_l();            //报警函数
          if(flag_clock == 1)
              beep = ~beep;
          flag_300ms = 0;
          if(menu_1 == 0)
          {
              smg_i = 3;
              dis_smg[0] = smg_du[temperature % 10]; //取温度的小数显示
```

```
                dis_smg[1] = smg_du[temperature / 10 % 10] & 0x7f; //取温度的个
位显示
                dis_smg[2] = smg_du[temperature / 100 % 10] ;//取温度的十位显示
            }
            if(zd_break_en == 1) //自动退出设置界面程序
            {
                zd_break_value ++;  //每 300ms 加一次
                if(zd_break_value > 50)   //15 秒后自动退出设置界面
                {
                    menu_1 = 0;
                    zd_break_en = 0;
                    zd_break_value = 0;
                }
            }
        }
    }
}
/*************定时器 0 中断服务程序***************/
void time0_int() interrupt 1
{
    static uchar value;
    TH0 = 0x3c;
    TL0 = 0xb0;     // 50ms
    value ++;
    if(value % 6 == 0)
    {
        flag_300ms = 1;      //300ms
        value = 0;
    }
    if(flag_lj_en == 1)      //按下按键使能
    {
        key_time ++;
        if(key_time >= 10) //500ms
        {
            key_time = 0;
            key_500ms = 1; //500ms
            flag_value ++;
            if(flag_value > 3)
            {
                flag_value = 10;
                flag_lj_3_en = 1; //3 次后 1.5 秒连加大些
            }
        }
    }
}
```