通信与信息工程学院

微控制器原理及应用课程设计报告

专 业 班 级：

设 计 题 目：　　智能温度控制系统

学　　号	姓　　名	成　　绩
19407010224		
19407010222		
17407111432		

指 导 教 师：

设 计 时 间：

通信与信息工程学院

二〇二二年

目录

智能温度控制系统

一、设计内容

1.1 课题设计的主要工作

本课题设计的系统用于对环境进行相对智能的温度控制，操作者可以通过按键和编程设置两种方式来随时的设定环境的理想温度,通过编程或键盘来确定水温上限和下限的范围值，并且可以说明温度的控制范围和控制的精度。具体要求如下:

1）采用 DS18B20 温度传感器，其测量温度范围在 -55℃～125℃；

2）可自定义设置最高和最低警告温度；

3）系统可以设定温度的控制范围在 -10℃～40℃；

4）设定温度和检测的温度可以实时的显示；

5）温度的控制精度为 ±0.1℃ 。

本系统设计了一套比较适合大众化的环境温度控制装置，用温度传感器 DS18B20 来检测环境温度，将测量得到的环境温度模拟量转换为数字量，再传给单片机；以应用广泛、操作容易、价格便宜的 Atmel89C51 单片机作为控制中心，采用温度比较算法来实现控制，并同时使用 PID 算法来保证精度。首先采用独立按键来设定环境温度的上下限，然后通过温度传感器来检测环境温度，当环境温度在合适的温度区间内，不采取任何行动；当环境温度低于最低设定温度时，报警电路中的绿灯亮，提醒使用者当前环境温度低于设定温度的最小值；当环境温度高于最高设定温度时，报警电路中的红灯亮，提醒使用者当前环境温度高于设定温度的最大值，报警装置是由红色和绿色两个 LED 灯组成，容易被操作人员发现，从而采取相应的人工干预；这套装置还有数码管实时显示的功能，以便操作人员知道当前环境内的实时温度。

二、总体方案设计

2.1 功能分析

本系统应用于对环境的温度控制，利用单片机技术使监控当前环境的温度。智能温度控制系统可以分为最小系统单元、键盘电路、传感器模块、显示模块、报警模块和控制电路。

传感器模块的设计主要对象是 DS18B20 传感器，设计合理的连接电路，采集温度信号，然后送到单片机中。DS18B20 是 DALLAS 公司生产的一线式数字温度传感器，具有 3 引脚 TO-92 小体积封装形式;温度测量范围为—55℃～+125℃,可编程为 9 位～12 位 A/D 转换精度，测温分辨率可达 0.0625℃，被测温度用符号扩展

的 16 位数字量方式串行输出;其工作电源既可在远端引入，也可采用寄生电源方式产生;多个 DS18B20 可以并联到 3 根或 2 根线上，CPU 只需一根端口线就能与诸多 DS18B20 通信，占用微处理器的端口较少，可节省大量的引线和逻辑电路。以上特点使 DS18B20 非常适用于远距离。

按键输入模块使用一个 4×4 矩阵键盘，用来对环境温度的上下限进行相应的设定。温度比较控制技术目前是环境温度控制的通用控制方法,是一种应用广泛、成熟的控制方法。

显示模块是用 4 位共阴极的 LED 数码管，采用动态显示的方式来显示检测到当前的环境温度和已经设定的温度阈值。

报警模块是采用灯光报警装置来实现的，当温度超过或低于水温允许的范围时，报警装置启动。

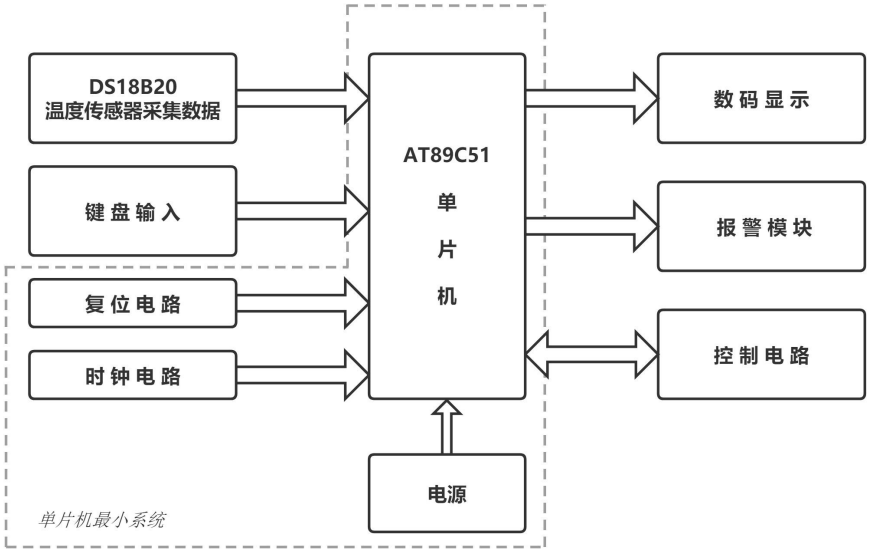

图 2.1 系统组成框图

2.2 各模块功能说明

2.2.1 最小系统单元

该系统使用 Atmel89C51 单片机作为主控。AT89C51 是一种带 4K 字节闪烁可编程可擦除只读存储器（FPEROM）、低电压、高性能 CMOS 的 8 位微处理器。该器件采用 ATMEL 高密度非易失存储器制造技术制造，与工业标准的 MCS-51 指令集和输出管脚相兼容。由于将多功能 8 位 CPU 和闪烁存储器组合在单个芯片中，ATMEL 的 AT89C51 是一种高效微控制器，为很多嵌入式控制系统提供了一种灵活性高且价廉的方案。

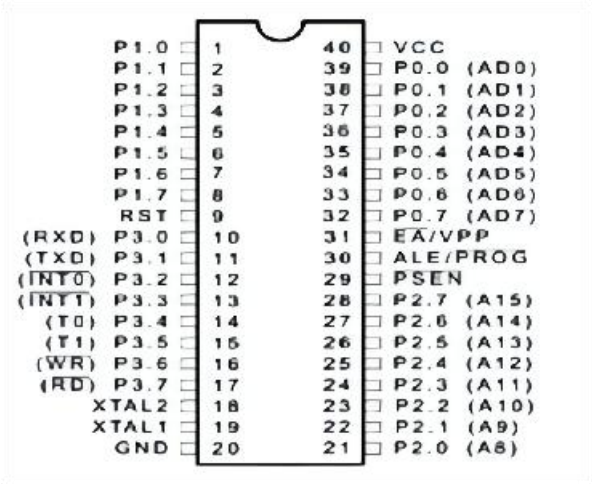

图 2.2.1 Atmel89C51 单片机管脚图

如上图所示，Atmel89C51 单片机和以往的 51 单片机一样有 40 根引脚，以双列直插方式封装。P0 口为数据总线(DB)，P0 与 P2 口为组成地址总线(AB)，P3 口为控制总线(CB)，并且 P3 口有复用功能，P1 与 P3 口组成用户 I/O 口。

时钟电路采用 12MHZ 的晶体振荡器产生振荡信号。

复位电路通过连接 RXD(P3.0)来实现复位功能。

2.2.2 传感器模块

该系统使用 DS18B20 数字温度传感器来实现对温度的采集。DS18B20 数字温度传感器接线方便，封装后可应用于多种场合，如管道式， 螺纹式，磁铁吸附式，不锈钢封装式。主要根据应用场合的不同而改变其外观。

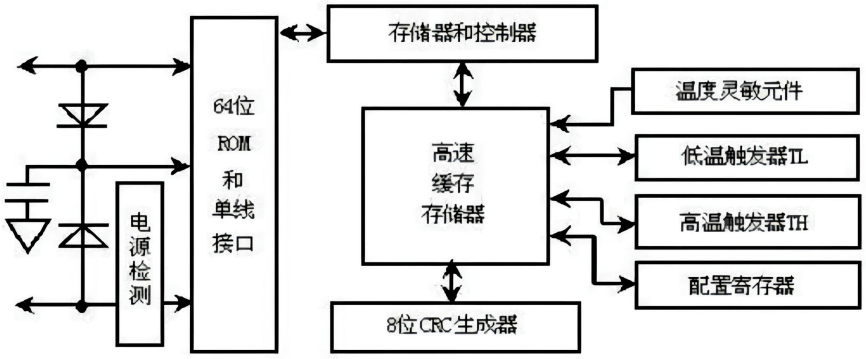

图 2.2.2 DS18B20 数字温度传感器内部结构图

DS18B20 数字温度传感器具有如下特点：

1) 独特的单线接口方式，DS18B20 在与微处理器连接时仅需要一条线即可实现微处理器与 DS18B20 的双向通讯；

2) DS18B20 在使用中不需要任何外围元件，全部传感元件及转换电路集成在形如一只三极管的集成电路内；

3) 可编程的分辨率为 9~12 位，对应的可分辨温度分别为 0.5°C、0.25°C、0.125°C 和 0.0625C，可实现高精度测温，在上电状态默认的精度 12 位;

4) 在 9 位分辨率时最多在 93.75ms 内把温度转换为数字，12 位分辨率时最多在 750ms 内把温度值转换为数字，速度更快；

5) 温度范围为-55°C～+125C。

表 2.2.1　使用到的 DS18B20 部分指令表

指令	约定代码	功能
跳过 ROM	CCH	忽略 64 位 ROM 地址，直接向 DS18B20 发温度变换命令，适用于单片机工作

RAM 指令表

指令	约定代码	功能
温度转换	44H	启动 DS18B20 进行温度转换，12 位转换时长为 750ms，结果存放于内部 9 字节的 RAM 中
读暂存器	BEH	读内部 RAM 中 9 字节的内容

上表为部分被使用到的 DS18B20 指令。在使用中，由于上电状态下默认的精度 12 位，12 位转化后得到的 12 位数据，存储在 DS18B20 的两个 8 位的 RAM 中，高字节的前 5 位是符号位，如果测得的温度大于 0，这 5 位为‘0’，只要将测到的数值乘以 0.0625 即可得到实际温度；如果温度小于 0，这 5 位为‘1’，测到的数值需要先减 1 再取反再乘以 0.0625 即可得到实际温度。

2.2.3 按键输入模块

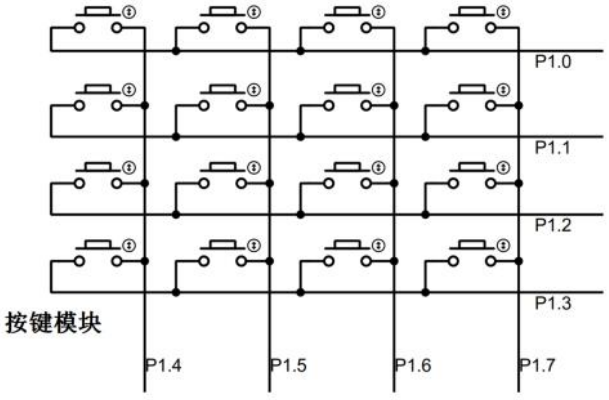

图 2.2.3 按键输入模块图

上图为 4×4 矩阵键盘的模块图，用单片机的 P1 口接 4×4 矩阵键盘，按下指定功能的按键后可以进行相应功能的操作。该矩阵键盘采用行扫描列输入的方式工作，矩阵键盘从上到下每一行分别接单片机的 P1.0、P1.1、P1.2、P1.3 引脚，

矩阵键盘从左到右每一列分别接单片机的 P1.4、P1.5、P1.6、P1.7 引脚。初始时，P1 口的高 7 位全部给高电平，第一位(P1.0)给低电平，进行第一次行扫描，此轮扫描完毕后若没有列为低电平(P1 口高 4 位全为高)则进行下一行行扫描，依此类推扫描 4 次；若在一次行扫描中有一列是低电平，则可判断按键按下且同时可判断出是哪个按键被按下。如：第一行行扫描时，最左边一列为低电平，则此时有按键被按下且按键是①。该键盘有 7 个功能键，如上图标注，分别的功能如下：

1）①按键按下可显示当前环境温度

2）②按键按下可显示系统设定的最大温度值

3）③按键按下可增加系统设定的最大温度值

4）④按键按下可减小系统设定的最大温度值

5）⑤按键按下可显示系统设定的最小温度值

6）⑥按键按下可增减系统设定的最小温度值

7）⑦按键按下可减小系统设定的最小温度值

注：按键按下的完整过程是按下并弹起。

2.2.4 显示模块

改系统使用一个型号为 7SEG-MPX4-CC 的 4 位共阴极二极管和一个型号为 7SEG-MPX1-CC 的 1 位共阴极二极管。其中 4 位二极管用来显示温度，1 位二极管用来显示温度模式。温度模式有：1 为当前温度模式；2 位设置最高报警温度；3 位设置最低报警温度。

图 2.2.4 7SEG-MPX4-CC 二极管模块展示图

上图所示是型号为 7SEG-MPX4-CC 的 4 位共阴极二极管。该模块采用动态显示的原理，减少了模块对端口的浪费使用。左下角 8 个端口通过接收二进制数据来判断某根二极管要亮；右下角 4 个端口来接收二进制数据来选择某一位，然后将 8 个端口输入的数值给到这一位。

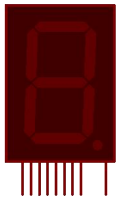

图 2.2.5 7SEG-MPX1-CC 二极管模块展示图

上图所示是型号为 7SEG-MPX1-CC 的 1 位共阴极二极管。左下角 8 个端口通过接收二进制数据来判断某根二极管要亮；右下角 1 个端口来接收二进制数据来判断是否被选择工作。

2.2.5 报警模块

报警模块由两个不同颜色的二极管指示灯组成。基于软件设置的最大警告温度和最小警告温度，当温度过高时，红灯亮起；当温度过低时，绿灯亮起；当温度适中，亮灯都不亮。

2.2.6 控制电路

控制电路由一个型号为 IRF5305 的场效应管和一个红色二极管指示灯组成，当温度过低，场效应管导通，二极管亮起，代表开始加热升温。

三、硬件系统设计

下面将给出各模块的硬件连接图以及系统的总电路图。

3.1 按键模块

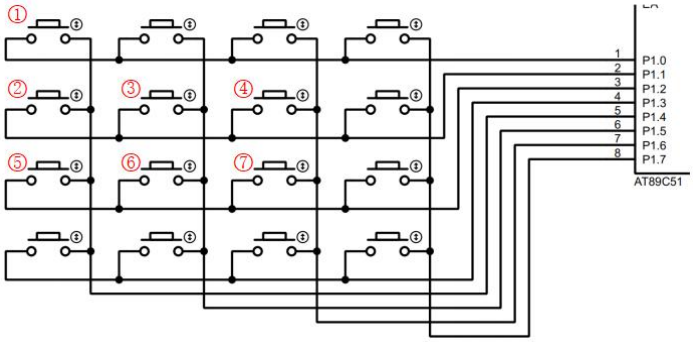

图 3.1.1 按键输入模块图

上图所示为按键输入模块展示图。

3.2 单片机最小系统

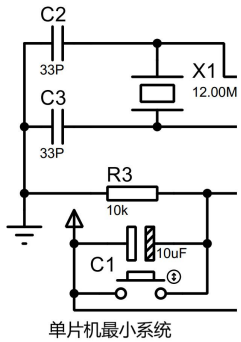

图 3.2.1 单片机最小系统模块图

上图所示为单片机最小系统模块展示图。

3.3 显示模块

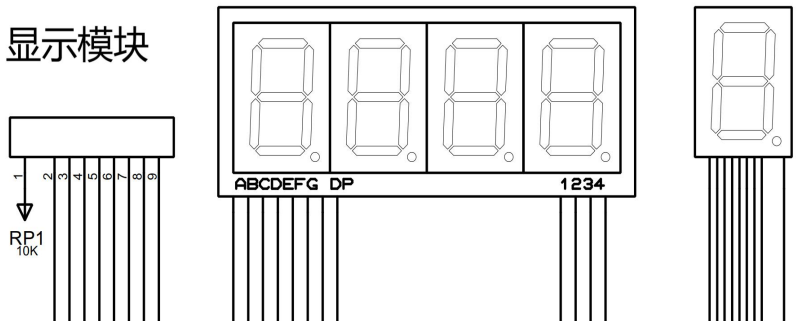

图 3.3.1 显示模块图

上图所示为显示模块展示图。

3.4 报警模块

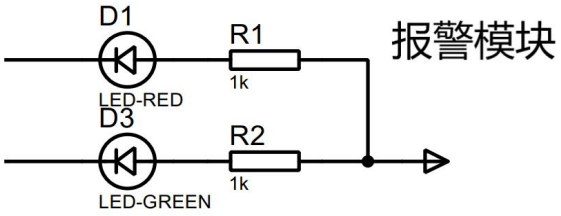

图 3.4.1 报警模块图

上图所示为报警模块展示图。

3.5 传感器模块

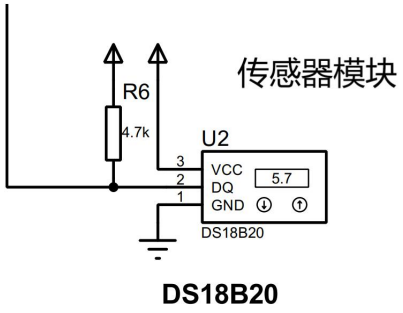

图 3.5.1 传感器模块图

上图所示为传感器模块展示图。

3.6 控制模块

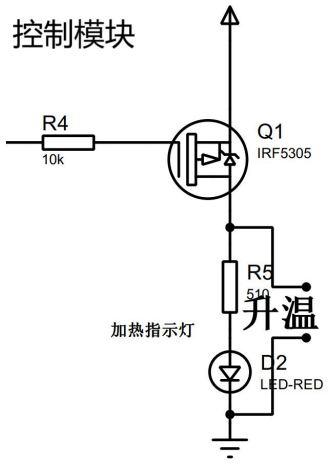

图 3.6.1 控制模块图

上图所示为控制模块展示图。

3.7 系统整体设计图

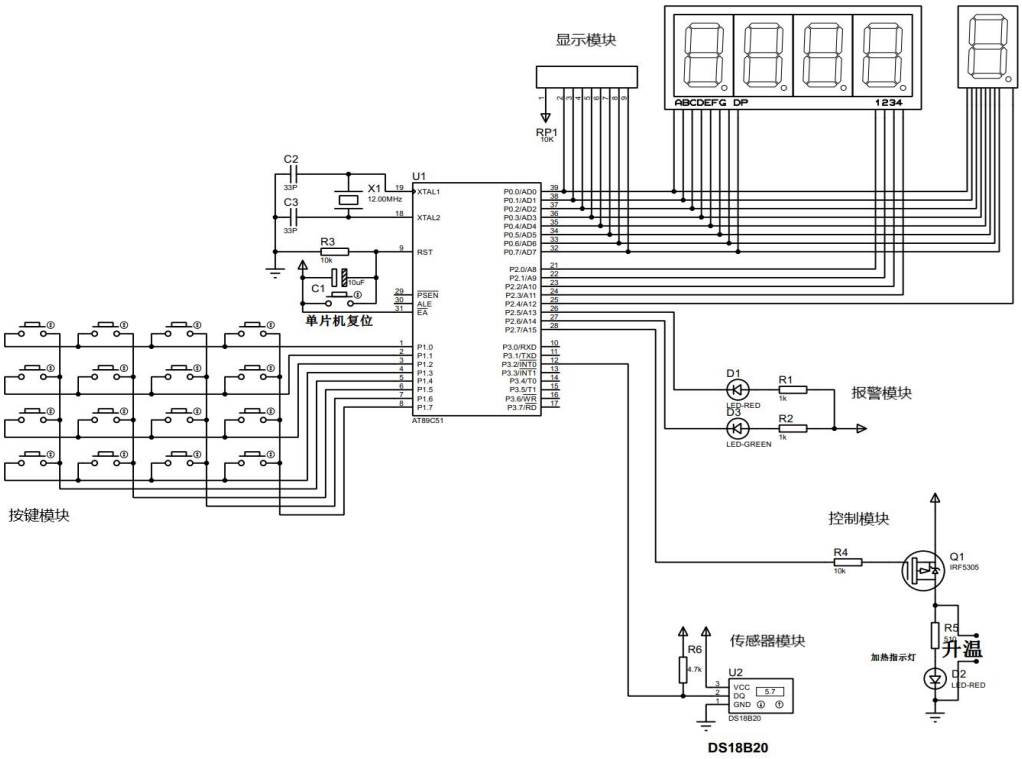

图 3.7.1 系统整体图

上图所示为系统整体连接展示图。

四、软件系统设计

4.1 程序总体流程图

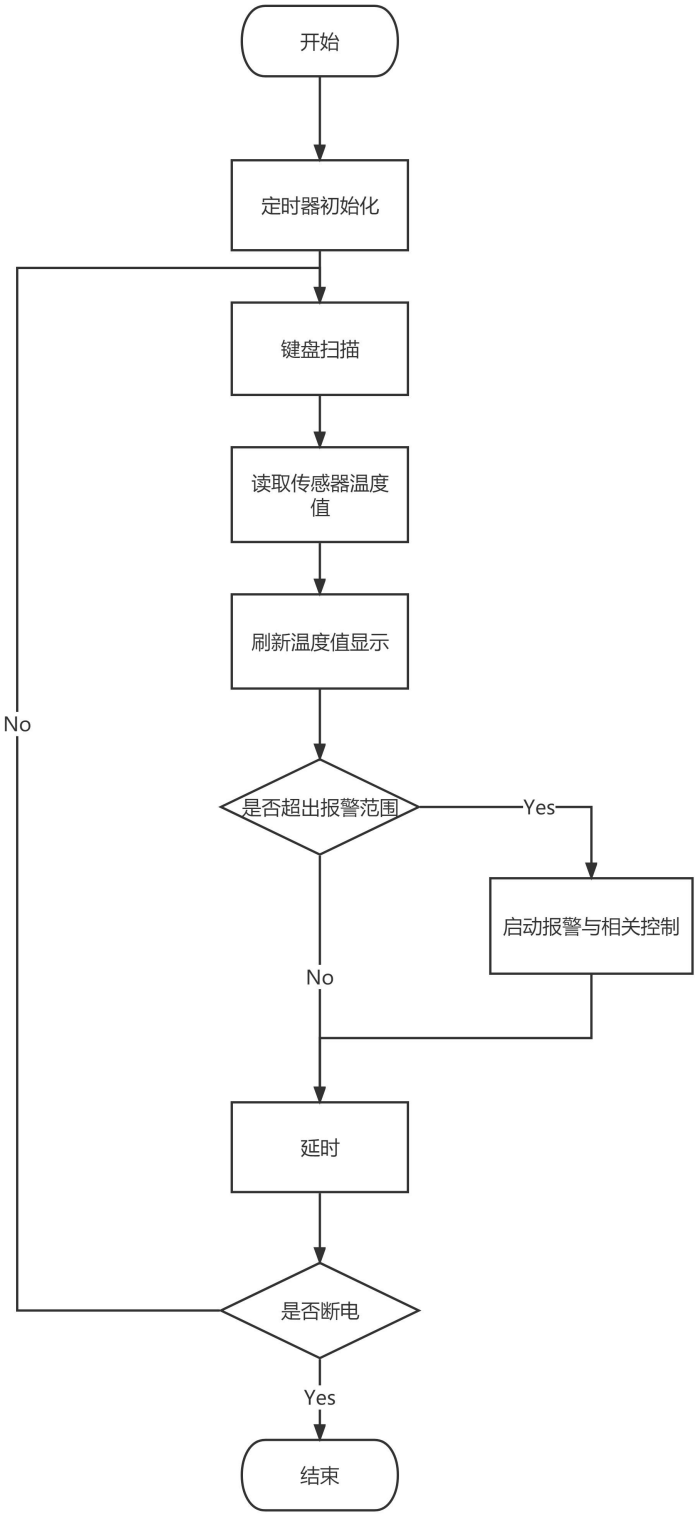

图 4.1.1 系统主程序流程图

上图所示为系统主程序流程。

4.2 主要模块程序流程

4.2.1 按键模块

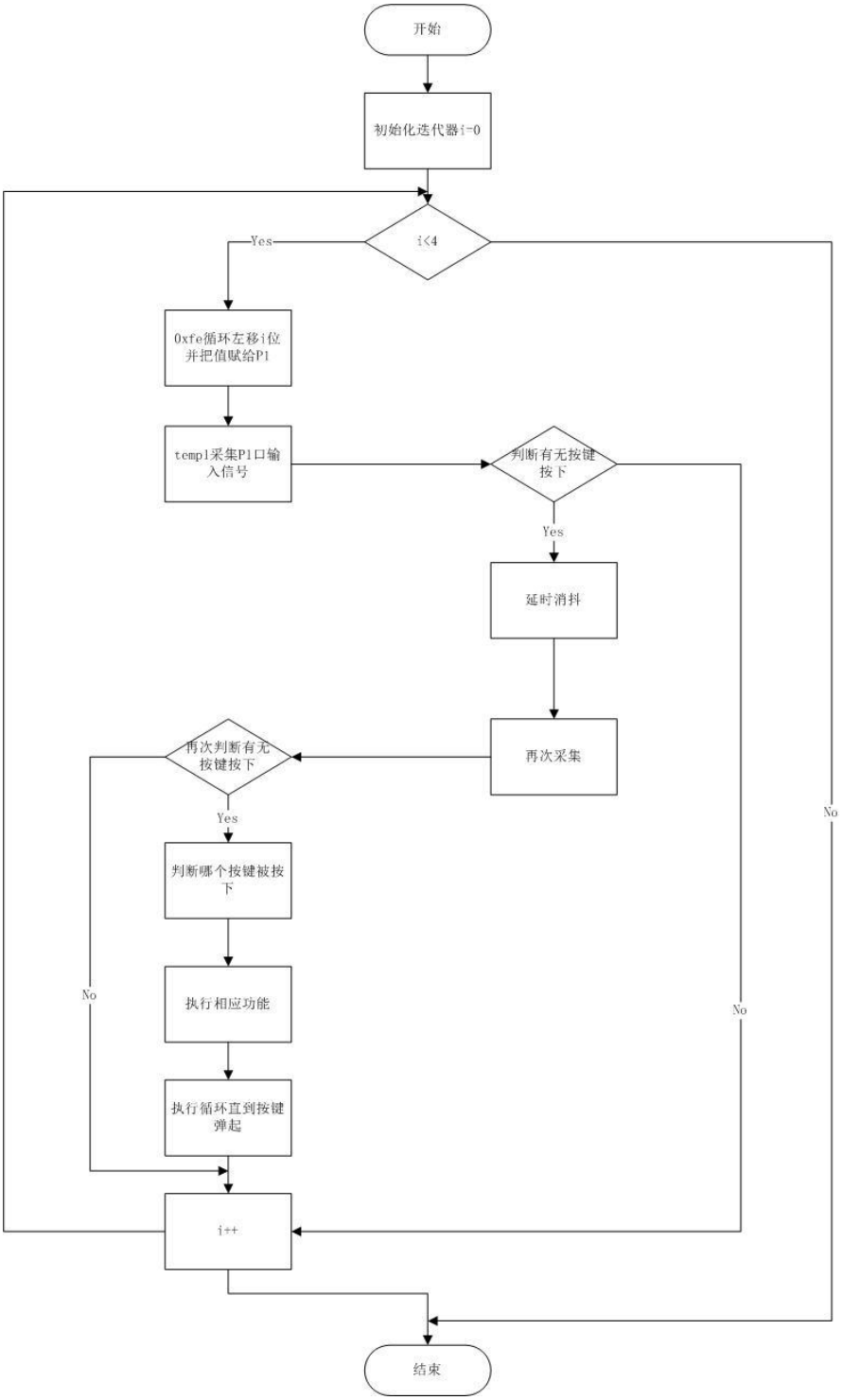

图 4.2.1 按键输入工作流程图

上图所示为按键输入工作流程。

4.2.2 报警模块与控制电路

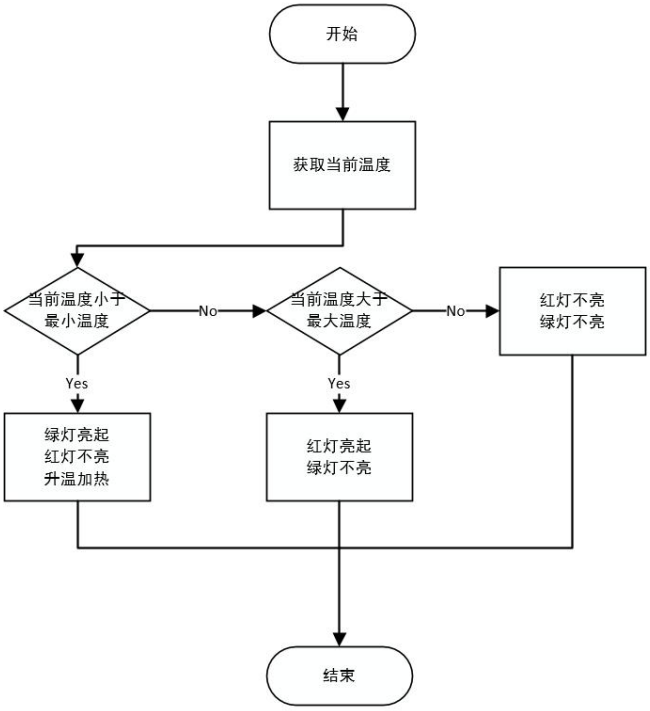

图 4.2.2 报警模块与控制电路工作流程图

上图所示为报警模块与控制电路工作流程。

4.2.3 显示模块

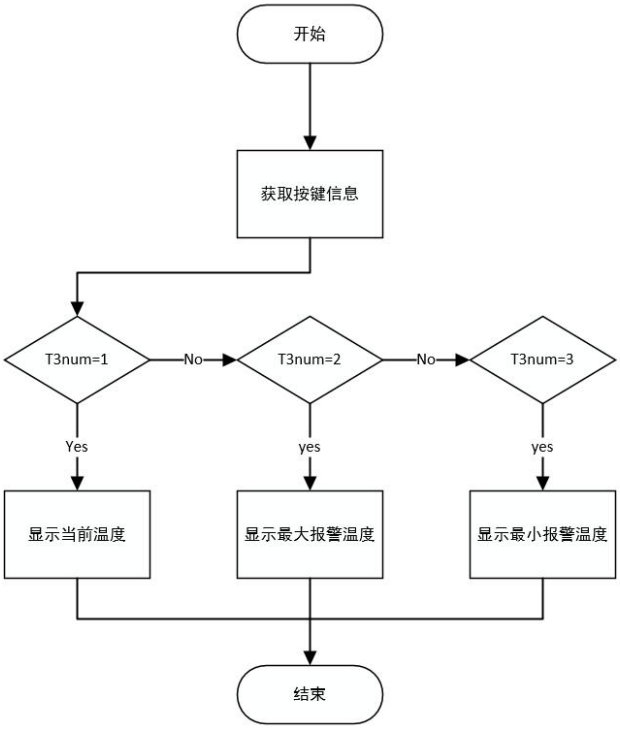

图 4.2.3 显示模块工作流程图

上图所示为显示模块工作流程。

4.2.4 传感器模块

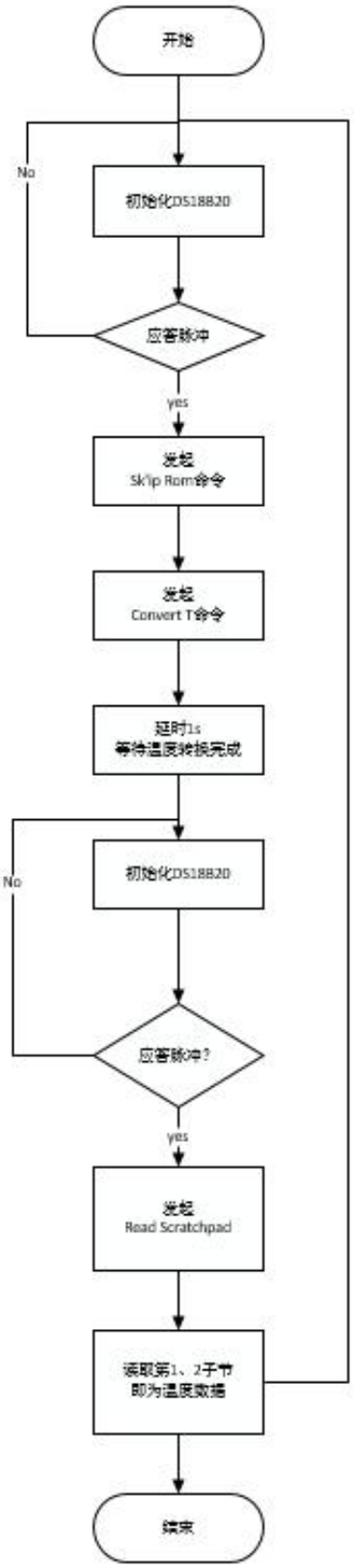

图 4.2.4 传感器模块工作流程图

上图所示为传感器模块工作流程。

五、源代码

5.1 定义与声明全局变量

```
struct PID {
    unsigned int SetPoint;        //设定目标  Desired Value
    unsigned int Proportion;      //比例常数  Proportional Const
    unsigned int Integral;        //积分常数  Integral Const
    unsigned int Derivative;      //微分常数  Derivative Const
    unsigned int LastError;       // Error[-1]
    unsigned int PrevError;       // Error[-2]
    unsigned int SumError;        // Sums of Errors
};
struct PID spid;          // PID Control Structure
unsigned int rout;        // PID Response (Output)  响应输出
unsigned int rin;         // PID Feedback (Input)//反馈输入
unsigned char high_time,low_time,count=0;          //占空比调节参数
#define uchar unsigned char
#define uint unsigned int
#define CLEAR_BIT(x, bit) (x &= ~(1 << bit))       //CLR
#define SET_BIT(x, bit)(x |= (1 << bit))           //SETB
uchar set[2]={0};
uchar n,num;
int set_temper_max=40, set_temper_min=10, temper, temp;  //温度变量定义
int temper, temp;
uchar temp1;                                       //按键标志
unsigned int s;
float f_temp;                                      //转换后的温度
uint tvalue;
uchar tflag;                                       //温度正负标志
uchar code LEDzf[]=                                //LED 显示正负
{
    0x00,0x40
```

```
};
uchar code LEDData[]=                                    //LED 显示数字 0~9
{
    0x3f,0x06,0x5b,0x4f,0x66,0x6d,0x7d,0x07,0x7f,0x6f
};
uchar t3num;                                             //三种温度的标识
```

5.2 定义特殊功能寄存器的位变量

```
/*****************************************/
/**       定义特殊功能寄存器的位变量       **/
/*****************************************/
sbit output=P2^7;                    //控制反馈输出
sbit ds=P3^2;                        //ds18b20 与单片机连接口
sbit DQ=P3^2;                        //ds18b20 与单片机连接口
sbit ledred=P2^6;                    //红灯与单片机连接
sbit ledgreen=P2^5;                  //绿灯与单片机连接
```

5.3 延时函数与初始化函数

```
/*********************/
/**       延时函数       **/
/*********************/
void delay(i)
{
    uint j;
    for(i;i>0;i--)
        for(j=111;j>0;j--);              //延时(111×i)ms
}
/***********************************/
/**        按照时序操作的初始化        **/
/***********************************/
void init() {
     t3num = 1;
}
```

5.4 DS1820 程序相关程序

```
/************************/
/**        延时 2 微秒      **/
/************************/
void delay_18B20(unsigned int i)
{
    while(i--);
}

/************************/
/**       ds1820 复位       **/
/************************/
void ds1820rst(void)
{
    unsigned char x=0;
    DQ = 1;                  //DQ 复位
    delay_18B20(4);          //延时
    DQ = 0;                  //DQ 拉低
    TR0 = 0;
    delay_18B20(100);        //精确延时大于
    TR0 = 1;
    DQ = 1;                  //拉高
    delay_18B20(40);
}

/*********************/
/**       读数据       **/
/*********************/
uchar ds1820rd(void)
{
    unsigned char i=0;
```

```
    unsigned char dat = 0;
   TR0=0;
    for (i=8;i>0;i--)
    {
        DQ = 0;                    //给脉冲信号
        dat>>=1;
        DQ = 1;                    //给脉冲信号
        if(DQ)
        dat|=0x80;
        delay_18B20(10);
    }
   return(dat);
}

/*********************/
/**       写数据       **/
/*********************/
void ds1820wr(uchar wdata)
{
    unsigned char i=0;
   TR0=0;
   for (i=8; i>0; i--)
   {
        DQ = 0;
        DQ = wdata&0x01;
        delay_18B20(10);
        DQ = 1;
        wdata>>=1;
   }
}
```

```
/**********************/
/**        获取温度        **/
/**********************/
uint get_temper()
{
    uchar a,b;
    ds1820rst();
    ds1820wr(0xcc);              //跳过读序列号
    ds1820wr(0x44);              //启动温度转换
    ds1820rst();
    ds1820wr(0xcc);              //跳过读序列号
    ds1820wr(0xbe);              //读取温度

    a=ds1820rd();
    b=ds1820rd();
    tvalue=b;
    tvalue<<=8;
    tvalue=tvalue|a;             //将低 8 位数据给到 tvalue
    TR0=1;
    if(tvalue<0x0fff)
        tflag=0;
    else
    {
        tvalue=~tvalue+1;
        tflag=1;
    }
    tvalue=tvalue*(0.625);       //温度值扩大 10 倍，精确到 1 位小数
    temp=tvalue;
    return temp;
}
```

5.5 显示温度相关程序

```
/**********************/
/**        显示温度        **/
/**********************/
void show_3temp(int t)                  //通用的显示温度函数
{
    uchar tflaguse;                     //定义要使用的温度正负标志位
    int d0,d1,d2,d3;
    int dt;
    if(t3num == 1)
    {
        tflaguse = tflag;
        dt = t;
    }
    else
    {
        if(t >= 0)
        {
            tflaguse = 0;
            dt = t;
        }
        else
        {
            tflaguse = 1;
            dt = -t;
        }
    }

    if(tflaguse==0)
    {
        d0=dt/1000;
```

```
        d1=dt%1000/100;              //百位(实际十位)
        d2=dt%100/10;                //十位(实际个位)
        d3=dt%10;                    //个位(实际小数点后一位)

        CLEAR_BIT(P2, 3);            //P2.3 清零(低电平有效)
        //P2 = 0xf7;
        P0 = LEDData[d3];            //将值给到 P2.3 对应位显示
        delay(3);
        SET_BIT(P2, 3);              //P2.3 置位

        CLEAR_BIT(P2, 2);
        //P2 = 0xfb;
        P0 = LEDData[d2];
        SET_BIT(P0, 7);
        delay(3);
        SET_BIT(P2, 2);

        CLEAR_BIT(P2, 1);
        //P2 = 0xfd;
        P0 = LEDData[d1];
        delay(3);
        SET_BIT(P2, 1);

        CLEAR_BIT(P2, 0);
        //P2 = 0xfe;
        P0 = LEDzf[tflaguse];
        delay(4);
        SET_BIT(P2, 0);
    }
    else
    {
```

```
            d0=dt/1000;
            d1=dt%1000/100;
            d2=dt%100/10;
            d3=dt%10;

            CLEAR_BIT(P2, 3);
            //P2 = 0xf7;
            P0 = LEDData[d3];
            delay(3);
            SET_BIT(P2, 3);

            CLEAR_BIT(P2, 2);
            //P2 = 0xfb;
            P0 = LEDData[d2];
            SET_BIT(P0, 7);
            delay(3);
            SET_BIT(P2, 2);

            CLEAR_BIT(P2, 1);
            //P2 = 0xfd;
            P0 = LEDData[d1];
            delay(3);
            SET_BIT(P2, 1);

            CLEAR_BIT(P2, 0);
            //P2 = 0xfe;
            P0 = LEDzf[tflaguse];
            delay(4);
            SET_BIT(P2, 0);
        }
    }
```

```
void dis_temp(int t)                    //选择三种温度模式
{
    if(t3num == 1)
    {
        CLEAR_BIT(P2, 4);
        P0 = 0x06;                      //温度模式位显示 1
        delay(5);
        SET_BIT(P2, 4);
        show_3temp(t);
    }
    else if(t3num == 2)
    {
        CLEAR_BIT(P2, 4);
        P0 = 0x5b;                      //温度模式位显示 2
        delay(5);
        SET_BIT(P2, 4);
        show_3temp(set_temper_max*10);
    }
    else if(t3num == 3)
    {
        CLEAR_BIT(P2, 4);
        P0 = 0x4f;                      //温度模式位显示 3
        delay(5);
        SET_BIT(P2, 4);
        show_3temp(set_temper_min*10);
    }
}
```

5.6 4×4 键盘相关程序

```
/************************/
/**          键盘          **/
/************************/
```

```
    void keyscan()                  //键盘扫描
    {
        unsigned char i, num = 16;  //如果让 num 初始化为 0-15 之间的数，则没有按下键盘也会显示 num 的初始值
        for(i = 0; i < 4; i++)      //行扫描
        {
            P1 = _crol_(0xfe,i);    //高 7 位都给高，第一位给低(列全高，第一行给低)(循环左移实现)
            temp1 = P1;             //temp 采集 P1 信号(P1 口高 4 位是输口，低 4 位是输出口)
            temp1 = temp1 & 0xf0;   //屏蔽低 4 位输出信号，采集高 4 位输入信号
            if(temp1 != 0xf0)       //若高四位都为 1，则没有被按下；反之有按键被按下
            {
                delay(20);          //延时 20 毫秒左右，之后重新采集(消抖)
                temp1 = P1;
                temp1 = temp1 & 0xf0;
                if(temp1 != 0xf0)   //再次判断是否有按键被按下
                {
                    temp1 = P1;     //再次采集(避免重新判断扫描的是第几行按键)，准备判断按键号
                    switch(temp1)   //判断哪个按键被按下
                    {
                        case 0xee:
                            t3num = 1;
                            break;
                        case 0xde:
                            break;
                        case 0xbe:
                            break;
```

```
case 0x7e:
    break;

case 0xed:
    t3num = 2;
    break;
case 0xdd:
    if(t3num == 2)
        set_temper_max++;
    break;
case 0xbd:
    if(t3num == 2)
        set_temper_max--;
    break;
case 0x7d:
    break;

case 0xeb:
    t3num = 3;
    break;
case 0xdb:
    if(t3num == 3)
        set_temper_min++;
    break;
case 0xbb:
    if(t3num == 3)
        set_temper_min--;
    break;
case 0x7b:
    break;
```

```
                    case 0xe7:
                        break;
                    case 0xd7:
                        break;
                    case 0xb7:
                        break;
                    case 0x77:
                        break;

                    default:
                        break;
                }
                while((temp1 & 0xf0) != 0xf0)       //判断按键是否弹起
                {
                    temp1 = P1;
                    temp1 = temp1 & 0xf0;
                }
            }
        }
    }
}
```

5.7 判断高低温相关程序

```
/************************/
/**        PID 初始化        **/
/************************/
void PIDInit (struct PID *pp)
{
    memset (pp,0,sizeof(struct PID));                    //用参数 0 初始化 pp
}
```

```
/************************/
/**          PID 计算          **/
/************************/
unsigned int PIDCalc(struct PID *pp, unsigned int NextPoint )
{
    unsigned int dError,Error;
    Error = pp->SetPoint - NextPoint;                    //偏差
    pp->SumError += Error;                               //积分
    dError = pp->LastError - pp->PrevError;              //当前微分
    pp->PrevError = pp->LastError;
    pp->LastError = Error;
    return (pp->Proportion * Error                       //比例
    + pp->Integral * pp->SumError                        //积分项
    + pp->Derivative * dError);                          //微分项
}

/*********************************************************/
/**                    温度比较处理子程序                    **/
/*********************************************************/

void compare_temper(void)
{

    unsigned char i;
    temper = get_temper()/10;
    if(set_temper_min>temper)                            //设置温度大于当前温度
    {
        ledred=0;
        ledgreen=1;
        if(set_temper_min-temper>1)                      //温度相差 1 度以上
        {
```

```
            high_time=100;
            low_time=0;
        }
        else                                    //设置温度不大于当前温度
        {
            for(i=0;i<10;i++)
            {
                get_temper();
                rin = s;                                // Read Input
                rout = PIDCalc ( &spid,rin );       //Perform PID Interation
            }
            if (high_time<=100)       high_time=(unsigned char)(rout/10000);
            else high_time=100;
            low_time= (100-high_time);
            delay_18B20(498);
        }
    }
    else if(set_temper_max<=temper)                 //设置温度不大于当前温度
    {
        ledred=1;
        ledgreen=0;
        if(temper-set_temper_max>0)                 //温度相差 0 度以上
        {
            high_time=0;
            low_time=100;
        }
        else
        {
            for(i=0;i<10;i++)
            {
                get_temper();
```

```
                rin = s; // Read Input
                rout = PIDCalc ( &spid,rin );      // Perform PID Interation
            }
            if (high_time<100) high_time=(unsigned char)(rout/10000);
            else       high_time=0;
            low_time= (100-high_time);
        }
    }
    else
    {
        ledred=1;
        ledgreen=1;
    }
}
```

5.8 主程序和 T0 中断服务子程序

```
/*********************************************************/
/**          用于控制电平的翻转 ,40us*100=4ms 周期          **/
/*********************************************************/
void serve_T0() interrupt 1 using 1
{
    if(++count<=(high_time))     output=0;
    else if(count<=100)
    {
        output=1;
    }
    else count=0;
    TH0=0x2f;
    TL0=0x40;
}
```

```
/*************************************/
/**                主函数                **/
/*************************************/
void main(void)
{
    //unsigned char i;
    init();//LED 初始化
    TMOD=0x01;
    TH0=0x2f;
    TL0=0x40;
    EA=1;
    ET0=1;
    TR0=1;
    high_time=50;
    low_time=50;
    PIDInit ( &spid );                    // Initialize Structure
    // Set PID Coefficients
    spid.Proportion = 10;                 //   P   10
    spid.Integral = 8;                    //   I    8
    spid.Derivative = 6;                  //   D   6
    spid.SetPoint = 100;                  //   Set PID Setpoint
    while(1)
    {
        delay(1);
        keyscan();                        //按键扫描
        dis_temp(get_temper());           //显示温度值
        compare_temper();                 //比较温度
    }
}
```

六、测试结果及分析

6.1 温度模式

分别通过按键①、②、⑤来选择显示当前环境温度、最大报警温度、最小报警温度。

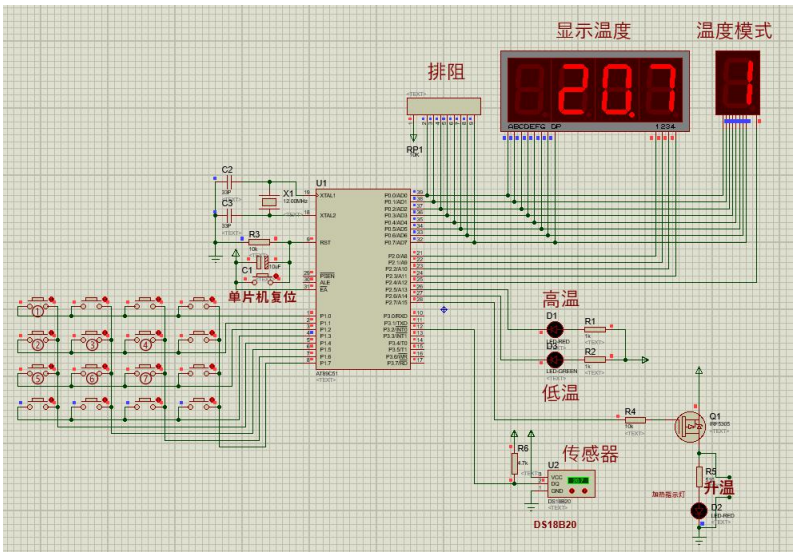

图 6.1.1 通过按键选择显示当前环境温度效果图

如图 6.1.1 所示，点击按键①，显示当前环境温度。

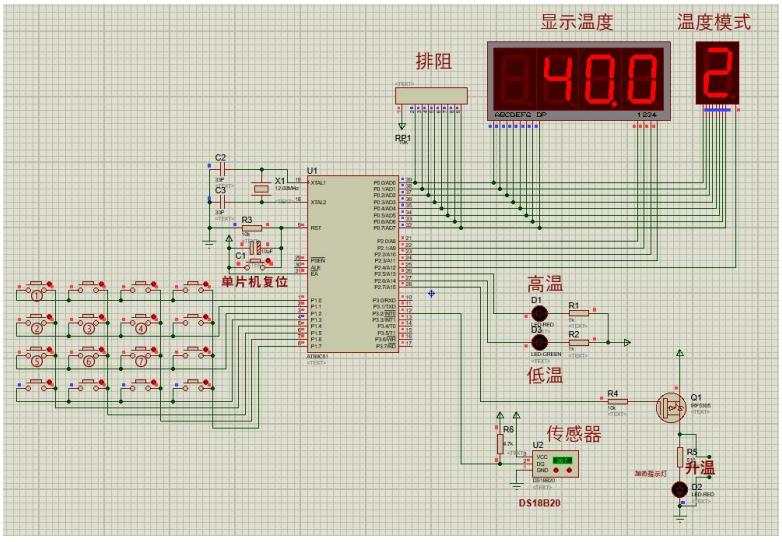

图 6.1.2 通过按键选择显示最大报警温度效果图

如图 6.1.2 所示，点击按键②，显示最大报警温度。

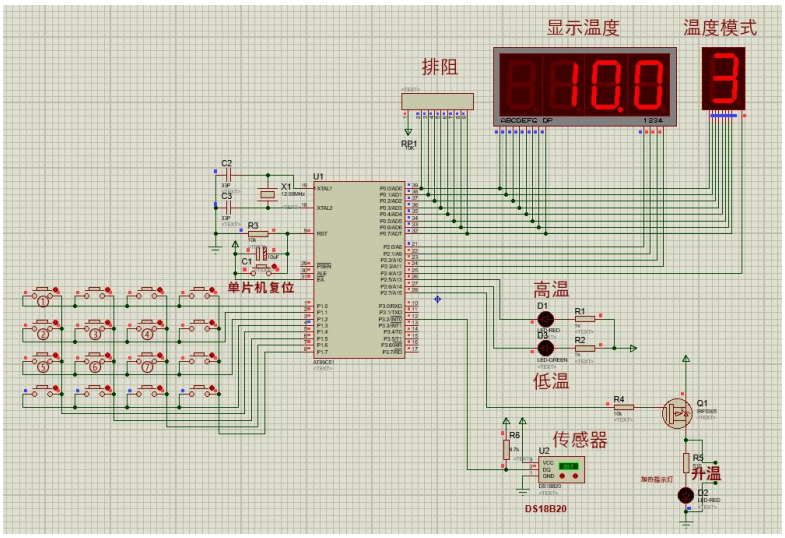

图 6.1.3 通过按键选择显示最小报警温度效果图

如图 6.1.3 所示，点击按键⑤，显示最小报警温度。

6.2 报警温度设置

分别通过按键③、④、⑥、⑦来选择设置最大报警温度、最小报警温度。

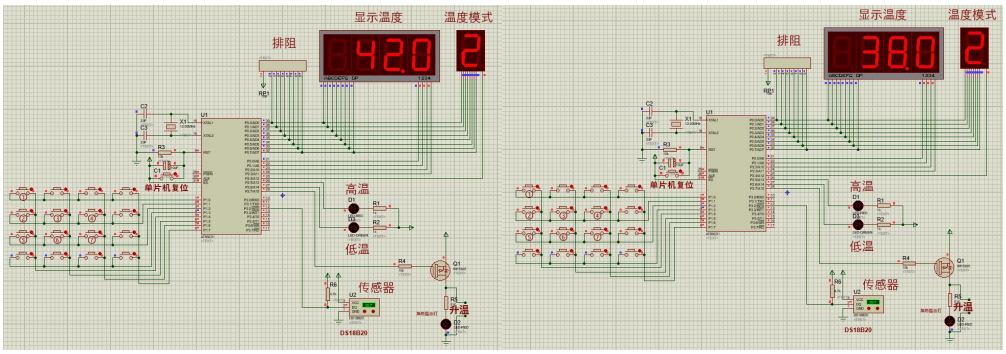

图 6.2.1 通过按键选择增加(左)和减少(右)最大报警温度效果图

如图 6.2.1 所示，点击按键③，增加最大报警温度；点击按键④，减小最大报警温度。

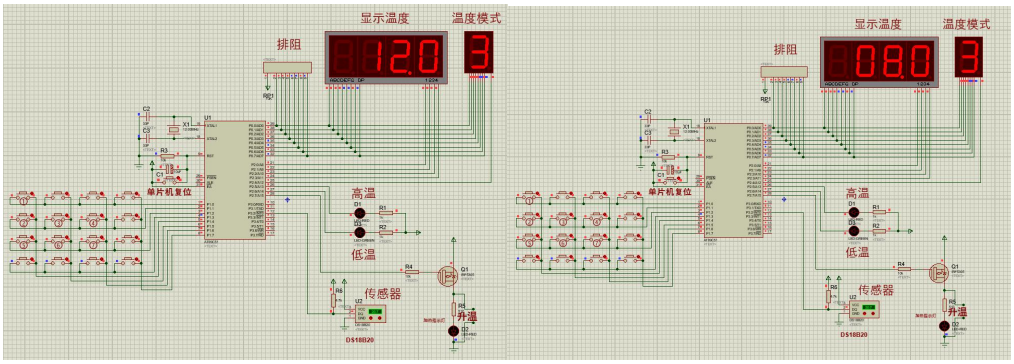

图 6.2.2 通过按键选择增加(左)和减少(右)最小报警温度效果图

如图 6.2.2 所示，点击按键⑥，增加最小报警温度；点击按键⑦，减小最小报警温度。

6.3 报警模块与控制电路

令温度高于 6.2 中设置好的最高报警温度(38℃)和低于 6.2 最低报警温度(8℃)，来观察高温和低温的指示灯。

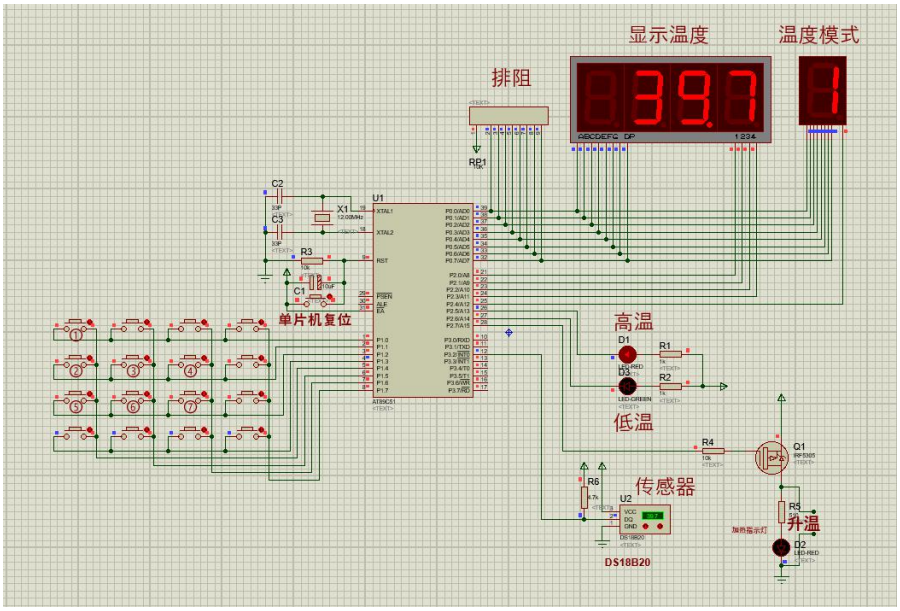

图 6.3.1 高温红灯亮效果图

如图 6.3.1 所示，让传感器测度温度升高，超过最高报警温度(38℃)，红色指示灯变亮。

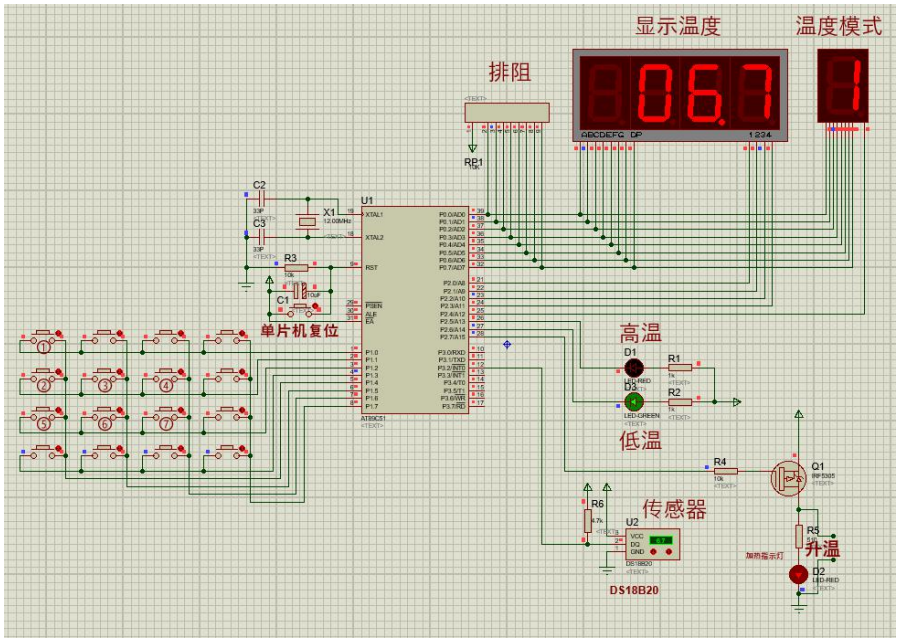

图 6.3.2 低温绿灯亮且升温控制启动效果图

如图 6.3.2 所示，让传感器测度温度降低，低于最低报警温度(8℃)，绿色色指示灯变亮，并且升温指示灯变亮。

通过上述对实验结果的测试，可见该系统在完成对各个模块的整合后，可以完成对 3 种模式温度的显示和设置，以及报警模块与控制电路的自动化工作。

七、设计总结

7.1 小组分工

表 7.1.1 小组分工情况表

姓名	任务

7.2 总结与收获

参考文献

[1] 吴友宇.模拟电子技术基础.北京.高教出版社，2001.

[2] 康华光.电子技术基础(模拟部分) (第四版).北京:高等教育出版社，1999.

[3] 李万臣.模拟电子技术基础与课程设计.哈尔滨:哈尔滨工程大学出版社，2001.3

[4] 胡宴如.模拟电子技术.北京: 高等教育出版社，2000.

[5] 沈尚贤.电子技术导论(下册).北京:高等教育出版社，1986.

[6] 李建兵周长林. Multisim 与 Protel 的应用.北京国防工业出版，2009.

通信与信息工程学院课程设计评价表

<table>
<tr><td></td><td>毕业要求</td><td colspan="4">课程目标</td></tr>
<tr><td rowspan="2">平时表现</td><td>问题分析（20）</td><td colspan="4">通过对单片机最小系统的应用，理解单片机系统的组成及以单片机为主的系统设计方法。提出系统设计方案、组成及各模块设计，培养解决电子信息工程领域复杂问题能力，达到对问题分析 3-2 指标点的支撑。</td></tr>
<tr><td>研究（10）</td><td colspan="4">通过对 IO 接口扩展，实现简单最小系统的设计，掌握单片机接口扩展的方法。通过硬件接口设计并编写程序解决电子信息工程实践中的问题，达到对研究中 4-2 指标点的支撑要求。</td></tr>
<tr><td rowspan="2">设计报告</td><td>问题分析（12）</td><td colspan="4">通过对单片机最小系统的应用，理解单片机系统的组成及以单片机为主的系统设计方法。提出系统设计方案、组成及各模块设计，培养解决电子信息工程领域复杂问题能力，达到对问题分析 3-2 指标点的支撑。</td></tr>
<tr><td>研究（18）</td><td colspan="4">通过对 IO 接口扩展，实现简单最小系统的设计，掌握单片机接口扩展的方法。通过硬件接口设计并编写程序解决电子信息工程实践中的问题，达到对研究中 4-2 指标点的支撑要求。</td></tr>
<tr><td rowspan="2">验收答辩</td><td>问题分析（35）</td><td colspan="4">通过对单片机最小系统的应用，理解单片机系统的组成及以单片机为主的系统设计方法。提出系统设计方案、组成及各模块设计，培养解决电子信息工程领域复杂问题能力，达到对问题分析 3-2 指标点的支撑。</td></tr>
<tr><td>研究（15）</td><td colspan="4">通过对 IO 接口扩展，实现简单最小系统的设计，掌握单片机接口扩展的方法。通过硬件接口设计并编写程序解决电子信息工程实践中的问题，达到对研究中 4-2 指标点的支撑要求。</td></tr>
<tr><td rowspan="4">学生得分</td><td>姓名</td><td>平时（30）</td><td>报告（20）</td><td>验收（50）</td><td>总分</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td></tr>
</table>