**一、设计任务**

**1. 课程设计题目:**

基于温度的直流电机转速控制器

1. **课程设计(论文)使用的原始资料(数据)及设计技术要求:**
2. 查阅相关文献资料，设计基于温度的直流电机转速控制器软硬件；
3. 硬件系统包括：单片机最小系统、显示模块、测温模块、直流电机驱动模块、按键设置模块、报警模块。
4. 当电机超过设定值上限时电机正转，低于设定值下限时电机反转，同时LED闪烁。
5. 实时显示温度。
6. 用按键设置温度的上下限。
7. **课程设计（论文）工作内容及完成时间:**

6月15日至6月16日:查找资料，方案制定；

6月17日至6月19日:传感器、驱动芯片选型，阅读Datasheet；

6月20日至6月21日:驱动电路设计；

6月22日至6月24日:程序设计与仿真；

6月25日至6月25日:撰写课程设计报告；

1. **主要参考资料:**
2. 朱清慧 《Protues教程--电子线路设计、制版与仿真（第3版）》 清华大学出版社2016
3. 高吉祥 《全国大学生电子设计大赛培训教程》电子工业出版社2007
4. 许维 《Protues 电子线路设计及仿真》电子工业出版社2014

**二、设计的目的和意义**

1. 进一步巩固和熟悉单片机系统的硬件设计和软件编程。

2. 学会使用仿真软件对单片机进行仿真。

3. 通过对单片机系统的设计过程、设计要求、设计方法培养学生运用综合运用已学知识解决实际工程技术问题的能力、查阅图书资料和各种工具书的能力。

4. 复习所学的专业知识，使课堂学习的专业知识运用于实践。

5. 掌握一般单片机电路的设计方法，提高设计能力和实践动手能力，为以后从事电子电路设计、研发电子产品打下练好基础。

6. 培养严肃、认真的工作作风和科学态度。

**三、电路设计原理**

3.1.方案

使用STC89C51单片机作为控制器；显示模块使用LCD1602；仿真中温度检测使用PT100，通过放大电路以及AD转换获取到电压，经转换公式获取到温度，实物中使用的是DS18b20;电机驱动模块使用L298n芯片；报警模块使用发光二极管以及蜂鸣器；同时通过三个按键来设置温度的上下限。当检测到的温度超过上限值或者低于下限值时，通过控制器给出PWM信号驱动电机正反转，温度偏离设定值越大，PWM占空比越高，转速越快，同时LED灯闪烁，蜂鸣器报警。

3.2 总体电路

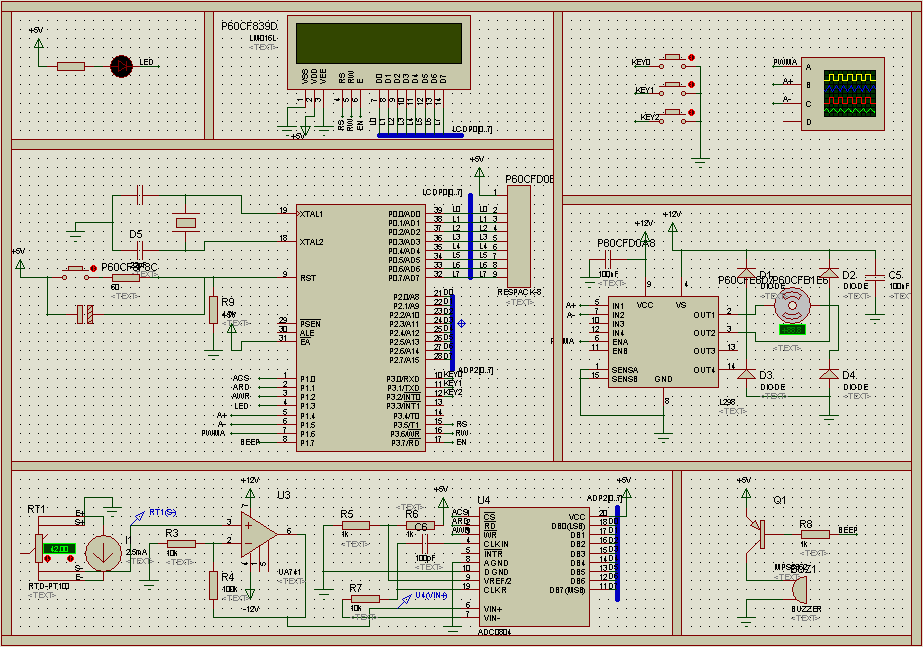


图3.1 总体电路仿真图

3.3 最小系统

最小系统由STC89C51、按键复位电路、晶振电路、上拉排阻组成。

按键复位电路:首先RST保持两个机器周期以上的高电平时自动复位。  
 1、上电复位：上电瞬间，电容充电电流最大，电容相当于短路，RST端为高电平，自动复位；电容两端的电压达到电源电压时，电容充电电流为零，电容相当于开路，RST端为低电平，程序正常运行。  
 2、手动复位：首先经过上电复位，当按下按键时，RST直接与VCC相连，为高电平形成复位，同时电解电容被短路放电；按键松开时，VCC对电容充电，充电电流在电阻上，RST依然为高电平，仍然是复位，充电完成后，电容相当于开路，RST为低电平，正常工作。

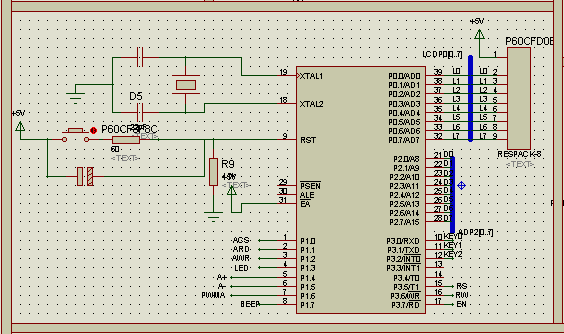


图3.2： 最小系统

3.3 测温电路

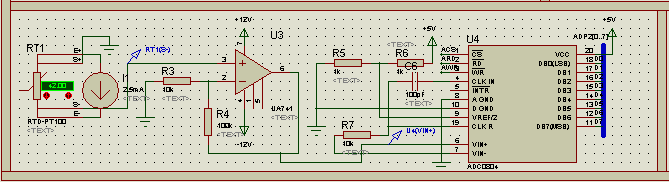


图3.3 测温电路

Main.c

#include "reg51.h"

#include "1602.h"

#include "dev\_l298n.h"

#include "adc0804.h"

#include "pt100.h"

#include "timer.h"

sbit BEEP = P1^7;

sbit LED = P1^3;

sbit key1 = P3^0;

sbit key2 = P3^1;

sbit key3 = P3^2;

bit led\_flag = 0;

uint h\_threshold = 50, l\_threshold = 30;

uchar threshold\_flag = 0;

enum KEY\_MODE

{

NORMAL,

HIGH\_THRESHOLD,

LOW\_THRESHOLD

} key\_mode = NORMAL;

void LED\_F()

{

static uchar count = 0;

if(led\_flag && ( count++ >= 100))

{

count = 0;

LED = LED == 1 ? 0 : 1;

}

}

void disp\_threshold()

{

write\_com(0x0c);

write\_string(0x80 + 0x40, "L:");

display\_1602(0, l\_threshold\*10);

write\_dat(0xDF);

write\_dat(0x43);

write\_string(0, "H:");

display\_1602(0, h\_threshold\*10);

write\_dat(0xDF);

write\_dat(0x43);

}

void display\_temp(float temp)

{

write\_string(0x80, "Temp:");

get\_sign() == 0 ? write\_dat('+') : write\_dat('-');

display\_1602(0, temp\*10);

write\_dat(0xDF);

write\_dat(0x43);

}

void keydscan()

{

key1 = 1;

key2 = 1;

key3 = 1;

if(key1==0) // 按键模式切换

{

delay2(5);

if(key1==0)

{

switch(key\_mode)

{

case NORMAL:

key\_mode = LOW\_THRESHOLD;

break;

case LOW\_THRESHOLD:

key\_mode = HIGH\_THRESHOLD;

break;

case HIGH\_THRESHOLD:

key\_mode = NORMAL;

break;

}

while(!key1);

}

}

if(key2==0) // 加

{

delay2(5);

if(key2==0)

{

switch(key\_mode)

{

case HIGH\_THRESHOLD:

h\_threshold++;

break;

case LOW\_THRESHOLD:

l\_threshold++;

}

disp\_threshold();

while(!key2);

}

}

if(key3==0) // 减

{

delay2(5);

if(key3==0)

{

switch(key\_mode)

{

case HIGH\_THRESHOLD:

h\_threshold--;

break;

case LOW\_THRESHOLD:

l\_threshold--;

}

disp\_threshold();

while(!key3);

}

}

}

int main()

{

float temp = 0, speed = 0;

uchar i;

uint ad\_data[11] = {0};

init\_1602();

timer0\_init(PWM\_TIME\_TICK);

l298n\_init();

tevent\_register("LED", LED\_F);

tevent\_register("KEY", keydscan);

disp\_threshold();

while(1)

{

//keydscan();

if(key\_mode == NORMAL)

{

for(i=0; i<11; i++)

{

ad\_data[i] = ad\_read();

}

temp = temperature(ad\_data, 11) - 1;

if(temp <= l\_threshold || temp >= h\_threshold)

{

led\_flag = 1;

BEEP = 0;

if(temp <= l\_threshold)

{

set\_direction(BACKWOARD);

speed = (l\_threshold - temp) / l\_threshold;

}

else

{

set\_direction(FOREWARD);

speed = (temp - h\_threshold) / h\_threshold;

}

speed > 1 ? set\_speed(PWM\_T) : set\_speed(PWM\_T \* speed);

}

else

{

led\_flag = 0;

LED = 1;

BEEP = 1;

set\_speed(0);

}

display\_temp(temp);

}

else

{

stop();

}

switch(key\_mode)

{

case NORMAL:

write\_com(0x0c);

break;

case HIGH\_THRESHOLD:

write\_com(0x80 + 0x40 + 8);

write\_com(0x0f);

break;

case LOW\_THRESHOLD:

write\_com(0x80 + 0x40);

write\_com(0x0f);

}

}

return 0;

}

---------------------------------------------------------------------------------------------------------------------------------

Time.h

#ifndef \_TIMER\_H\_

#define \_TIMER\_H\_

#include "reg51.h"

//--------config--------

#define TEVENT 3

#define fos 11.0592 // M

#define time\_t 5 // ms

//----------------------

#define NULL ((void \*)0)

#define ft ((1.0/fos\*12)/1000)

typedef void(\*ptimer)(void);

typedef struct timer\_desc{

char \*name;

ptimer fp ;

}timer\_desc;

extern timer\_desc t\_event[TEVENT];

void timer0\_init(uint time);

void set\_time(uint time);

int tevent\_register(char \* name, ptimer fp);

#endif

Time.c

#include"timer.h"

timer\_desc t\_event[TEVENT];

uchar T\_H = (uint)(65536 - time\_t/ft) / 256;

uchar T\_L = (uint)(65536 - time\_t/ft) % 256;

uint times = 1; // T = times \* time\_t ms

uchar time\_count = 1;

static void timer0\_handle()

{

int i;

for(i = 0; i < TEVENT; i++)

{

if(t\_event[i].fp)

{

t\_event[i].fp();

}

}

}

void timer0\_init(uint time)

{

times = time / time\_t;

EA = 1;

ET0 = 1;

TMOD = 0x01;

TH0 = T\_H; // 50ms

TL0 = T\_L;

TR0 = 1;

}

void set\_time(uint time)

{

times = time / time\_t;

time\_count = 0;

}

int tevent\_register(char \* name, ptimer fp)

{

int i;

for(i = 0; i < TEVENT; i++)

{

if(!t\_event[i].fp)

{

t\_event[i].fp = fp;

t\_event[i].name = name;

return 0;

}

}

return -1;

}

void time() interrupt 1

{

TH0 = T\_H;

TL0 = T\_L;

if(time\_count >= times)

{

time\_count = 1;

timer0\_handle();

}

else

{

time\_count++;

}

}

--------------------------------------------------------------------------------------------------------------------------------

1602.h

#ifndef \_\_1602\_\_H

#define \_\_1602\_\_H

sbit lcden=P3^7;

sbit lcdrs=P3^5;

sbit lcdrw=P3^6;

void delay2(uint z);

void write\_dat(uchar dat);

void write\_com(uchar com);

void init\_1602();

uchar dig\_convert(uint d, uint\* ret);

void display\_1602(uchar pos, uint v);

void write\_string(uchar pos, const char\* str);

#endif

--------------------------------------------------------------------------------------------------------------------------------

1602.c

#include "reg51.h"

#include "1602.h"

uchar dig\_convert(uint d, uint\* ret)

{

if(d <= 999)

{

ret[0] = d/100;

ret[1] = d/10%10;

ret[2] = d%10;

return 0;

}

else

{

ret[0] = d/1000;

ret[1] = d/100%10;

ret[2] = d/10%10;

ret[3] = d%10;

return 1;

}

}

void display\_1602(uchar pos, uint v)

{

uint da[4];

uchar flag;

if(pos != 0)

write\_com(pos);

flag = dig\_convert(v, da);

write\_dat(0x30 + da[0]);

write\_dat(0x30 + da[1]);

if(flag)

{

write\_dat(0x30 + da[2]);

write\_dat('.');

write\_dat(0x30 + da[3]);

}

else

{

write\_dat('.');

write\_dat(0x30 + da[2]);

}

}

void write\_string(uchar pos, const char\* str)

{

if(pos != 0)

write\_com(pos);

while(\*str)

{

write\_dat(\*str);

str++;

}

}

void delay2(uint z)

{

uchar x,y;

for(x=z;x>0;x--)

for(y=110;y>0;y--);

}

void write\_dat(uchar dat)

{

lcden=0;

lcdrs=1;

P0=dat;

delay2(5);

lcden=1;

delay2(5);

lcden=0;

lcdrs=0;

}

void write\_com(uchar com)

{

lcden=0;

lcdrs=0;

P0=com;

delay2(5);

lcden=1;

delay2(5);

lcden=0;

}

void init\_1602()

{

lcdrw = 0;

write\_com(0x38);

write\_com(0x0c);

write\_com(0x06);

write\_com(0x01);

write\_com(0x80+0x00);

}

--------------------------------------------------------------------------------------------------------------------------------

Pt100.h

#ifndef \_\_PT100\_H\_

#define \_\_PT100\_H\_

#include "reg51.h"

#define R3 10.0

#define R4 100.0

#define I 2.5 //恒流源电流 ma

#define K ((R3 + R4) / R3)

// u\_out mv

float calc\_pt100res(uint u\_out);

uint median(uint\* dat, uchar num\_d);

float temperature(uint\* u\_out, uchar num);

bit get\_sign();

#endif

--------------------------------------------------------------------------------------------------------------------------------

Pt100.c

#include "pt100.h"

bit ng;

float calc\_pt100res(uint u\_out)

{

return u\_out / K / I;

}

uint median(uint\* dat, uchar num\_d) //需要排序的数组

{

uchar i,j;

uint temp;

for(i=0;i<num\_d;i++) //采用冒泡法对采样温度进行排序

{

for(j=0;j<i;j++)

{

if(\*(dat+i)>\*(dat+j))

{

temp=\*(dat+i); //交换两值 从大到小排

\*(dat+i)=\*(dat+j);

\*(dat+j)=temp;

}

}

}

return dat[(num\_d-1)/2];

}

float temperature(uint\* u\_out, uchar num)

{

float temp1, T\_out;

uint temp2;

temp1 = median(u\_out, num);

temp1 = calc\_pt100res(temp1); //计算Pt100阻值

temp2 = (uint)temp1; //取Pt100阻值高位

if(temp2<100){ng=1;}else{ng=0;}//负温度

if(temp2<96) T\_out=777; //小于-10℃

else if(temp2<100) T\_out=255.1239-2.5511\*temp1; //若阻值在小于0℃到负20℃之间

else if(temp2<112) T\_out=2.5703\*temp1-257.0501; //若阻值在0℃~30℃之间

else if(temp2<123) T\_out=2.5935\*temp1-259.6416; //若阻值在30℃~60℃之间

else if(temp2<135) T\_out=2.6171\*temp1-262.5538; //若阻值在60℃~90℃之间

else if(temp2<139) T\_out=2.6412\*temp1-265.7953; //若阻值在90℃~120℃(146.06)之间 100℃(138.5)

else if(temp2>=139) T\_out=2.6412\*temp1-265.7953;; //若阻值在大于100℃之间 \*/

return(T\_out);

}

bit get\_sign()

{

return ng;

}

--------------------------------------------------------------------------------------------------------------------------------

Adc0804.h

#ifndef \_\_ADC0804\_H\_

#define \_\_ADC0804\_H\_

#include "reg51.h"

#include<intrins.h>

#define dataIO P2

sbit adrd = P1^1;

sbit adwr = P1^2;

sbit adcs = P1^0;

uint ad\_read();

#endif

--------------------------------------------------------------------------------------------------------------------------------

Adc0804.c

#include "adc0804.h"

uint ad\_read()

{

uint v;

adcs = 0;

adwr = 1;

\_nop\_();

adwr = 0;

\_nop\_();

adwr = 1;

dataIO=0xff;

adrd = 1;

\_nop\_();

adrd = 0;

\_nop\_();

v = dataIO;

adrd = 1;

v = (v/255.0)\*5000;

return v;

}

--------------------------------------------------------------------------------------------------------------------------------

Dev\_l298n.h

#ifndef \_\_L298N\_H\_

#define \_\_L298N\_H\_

#include "reg51.h"

#define PWM\_T 50 // real\_time = PWM\_T \* PWM\_TIME\_TICK

#define PWM\_TIME\_TICK 5 // ms

#define FOREWARD 1

#define BACKWOARD 2

sbit IN1 = P1^4;

sbit IN2 = P1^5;

sbit PWM = P1^6;

void l298n\_init();

void set\_direction(uchar dir);

void set\_speed(uchar spd);

void stop();

#endif

--------------------------------------------------------------------------------------------------------------------------------

Dev\_l298n.c

#include "dev\_l298n.h"

#include "timer.h"

uchar pwm\_h = 1;

uchar current = 0;

bit stop\_flag = 0;

// dir: 1 正 2 反

void set\_direction(uchar dir)

{

switch(dir)

{

case FOREWARD:

{

IN1 = 1;

IN2 = 0;

break;

}

case BACKWOARD:

{

IN1 = 0;

IN2 = 1;

break;

}

default:

break;

}

}

void pwm()

{

if(!stop\_flag)

{

if(current < pwm\_h)

{

PWM = 1;

}

else if(current >= pwm\_h && current <= PWM\_T)

{

PWM = 0;

}

else

{

PWM = 1;

current = 0;

}

current++;

}

}

void l298n\_init()

{

PWM = 0;

set\_direction(1);

tevent\_register("PWM", pwm);

stop();

}

void set\_speed(uchar spd)

{

pwm\_h = spd <= PWM\_T ? spd : PWM\_T;

current = 0;

stop\_flag = 0;

}

void stop()

{

PWM = 0;

stop\_flag = 1;

}