南昌工程学院

《单片机课程设计》

题 目 基于温度的直流电机转速控制器

课 程 名 称 单片机原理

系 院 机械与电气工程学院

专 业 自动化

班 级 18自动化01

学 生 姓 名 骆任贤

学 号 2018101649

指 导 教 师 于帧

设计起止时间：2021年 6 月 15 日 至 2021年 6月25 日

**课程设计（论文）任务书**

|  |
| --- |
| **I、课程设计(论文)题目：**基于温度的直流电机转速控制器 |
| **II、课程设计(论文)使用的原始资料(数据)及设计技术要求：** |
| 1、查找有关温度压力流量等物理量检测的有关资料； |
| 2、设计单片机测控系统：含必要的键盘、显示、信号调理等电路； |
| 3、物理量上下限可调，超限给出指示； |
| 4、实时显示温度和直流电机转速； |
| **III、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；  4月16日至5月30日：软件设计总体规划、设计与调试；  6月1日至6月10日： 软、硬件联合调试与实验；  6月11日至6月14日：撰写论文，答辩； |
| 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。 |
| **Ⅳ 主 要参考资料：** |
| 1.朱清慧《Proteus教程—电子线路设计、制版与仿真(第3版）》清华大学出版社2016 |
| 2.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 |
| 3.许维蓥《Proteus电子电路设计及仿真》电子工业出版社2014 |
|  |
|  |

目录

[一、设计的目的和意义 1](#_Toc31444)

[二、电路设计原理 1](#_Toc19970)

[2.1 方案 1](#_Toc12918)

[2.2 总体电路 1](#_Toc7068)

[2.3 最小系统 2](#_Toc28033)

[2.4 测温电路 3](#_Toc3164)

[2.5 显示模块 4](#_Toc4118)

[2.6 电机模块 4](#_Toc16753)

[2.7 按键设置模块 7](#_Toc5552)

[2.8 报警模块 7](#_Toc10950)

[三、 软件设计 8](#_Toc8196)

[3.1 定时器设计 8](#_Toc16527)

[3.2 电机驱动 9](#_Toc4947)

[3.3 获取温度 10](#_Toc21079)

[3.4 主程序 11](#_Toc19897)

[四、电路仿真与实物演示 12](#_Toc27736)

[4.1 温度检测与显示 12](#_Toc8437)

[4.2 电机驱动 12](#_Toc7536)

[4.3 温度上下限设置 14](#_Toc20498)

[4.4 报警模块 16](#_Toc21211)

[五、小结 17](#_Toc725)

[六、参考文献 17](#_Toc10674)

附录一 18

附录二 33

**一、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。

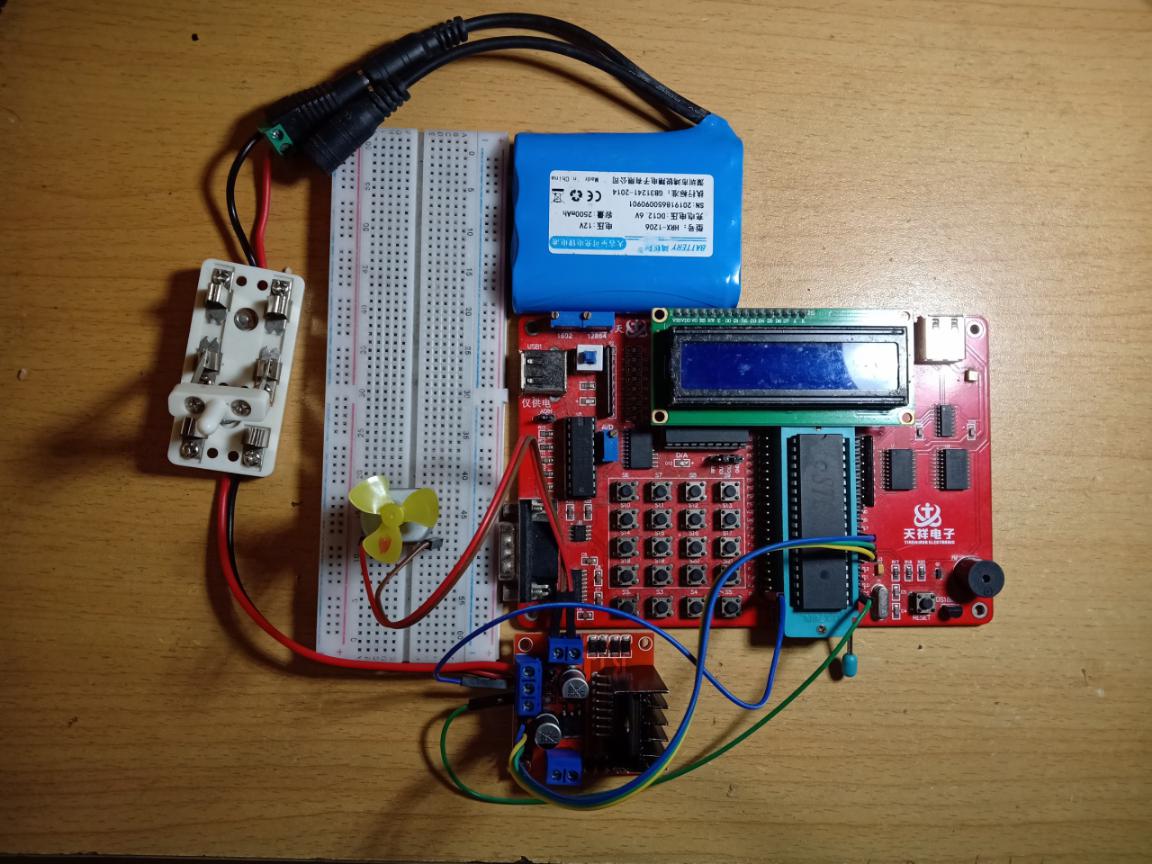
6. 培养严肃、认真的工作作风和科学态度。

**二、电路设计原理**

2.1 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。

2.2 总体电路



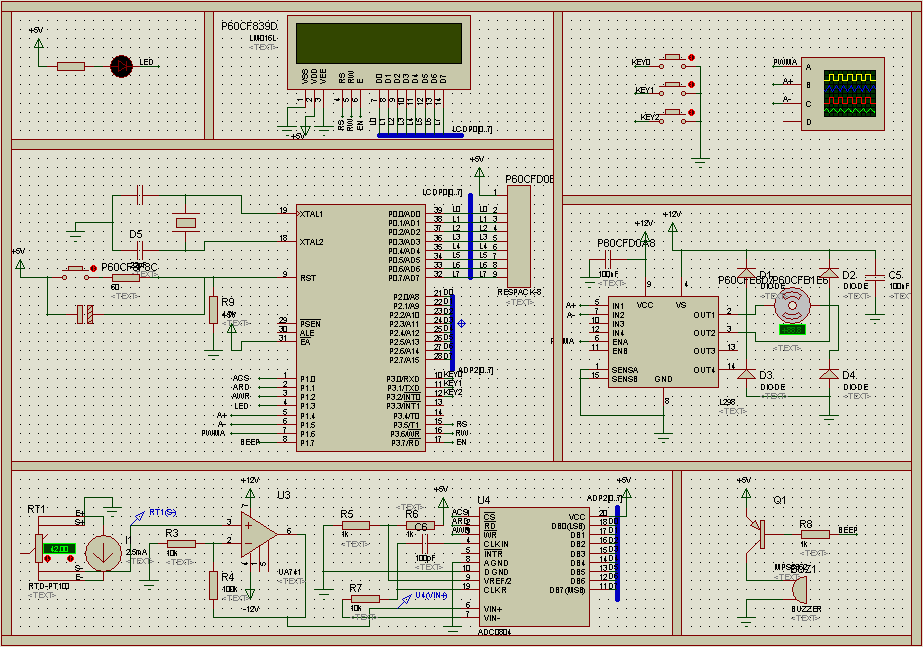


图3.1 总体电路图

2.3 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。

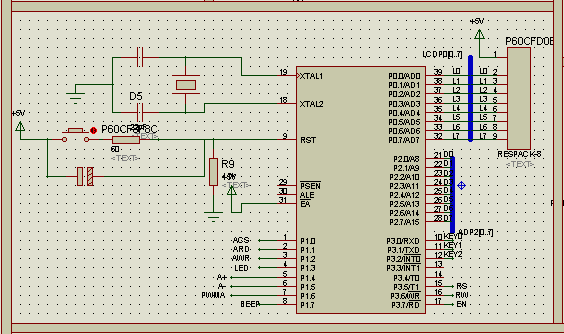


图3.2： 最小系统

2.4 测温电路

测温电路使用了PT100铂热电阻，当PT100在0摄氏度的时候他的阻值为100欧姆，它的阻值会随着温度上升而成近似匀速的增长。在其两端接入恒流源，只要读出其两端电压，就可以计算出其电阻，从而可以通过拟合的温度与电阻的函数关系计算出当前温度。AD的参考电压为5V，为了方便和更加精准的测出电阻两边的电压，在铂电阻和AD之间加入了同向比例放大器，其放大倍数为:



AD采用的是ADC0804，ADC0804是一款8位、单通道、低价格A/D转换器。工作电压：+5V，属于连续渐进式的A/D转换器。



其中:

 :输入ADC的模拟电压值。

:ADC转换后的二进制值。 ADC0804为八位。

 :ADC能够表示的刻度总数。ADC0804为八位ADC,因此=256。

 :ADC参考电压值，仿真中ADC0804的Wref被设置为5V。

因此，对于本次仿真中，转换公式为:



为了尽量滤除干扰，对采集到的电压值进行了中值滤波处理，即连续采样11次，并对其进行排序，取中间值作为结果。

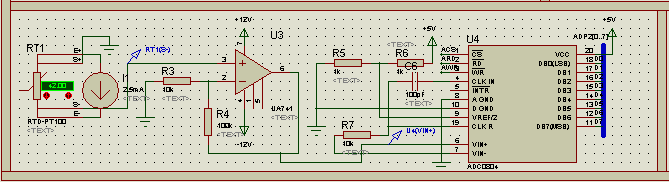


图2.3 测温电路

2.5 显示模块

LCD1602是一种字符型液晶显示模块,不同于七段数码管,它可以通过点阵的形式显示出各种图案或字符,可拓展性较强。其名称中"LCD"即为 Liquid Crystal Display (液晶显示器),"1602"代表显示屏上可同时显示32个字符(16x2)。

1602的操作是通过RS、RW、E进行控制的，我们更多用到它的写操作时序：

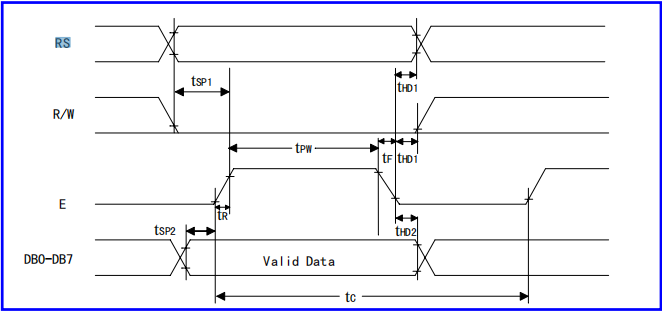


图2.4 1602写操作时序

通过单片机的相应引脚实现上图时序就可以对1602进行写操作，写操作也分为写指令和写数据，当RS=0时写指令，当RS=1时写数据。

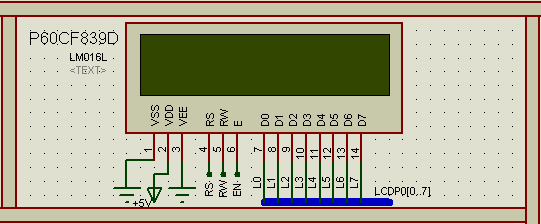


图2.5 1602显示模块

2.6 电机模块

为了完全控制直流电动机，我们必须控制其速度和旋转方向。这可以通过结合这两种技术来实现：

1. PWM –用于控制速度
2. H桥–用于控制旋转方向

直流电动机的速度可以通过改变其输入电压来控制。执行此操作的常用技术是使用PWM（脉冲宽度调制），PWM是通过发送一系列ON-OFF脉冲来调整输入电压平均值的技术。平均电压与称为占空比的脉冲宽度成比例，占空比越高，施加到直流电动机（高速）的平均电压越大，占空比越低，施加到直流电动机（低速）的平均电压越小。

H桥–用于控制旋转方向，直流电动机的旋转方向可以通过改变其输入电压的极性来控制。执行此操作的常用技术是使用H桥。H桥电路包含四个开关，电机在中心，形成H形排列。同时闭合两个特定的开关会反转施加到电动机的电压的极性。这引起电动机旋转方向的改变。下图显示了H桥电路的工作原理。

L298N是一款双通道H桥电机驱动器，能够驱动一对DC电机。下面是它的真值表：

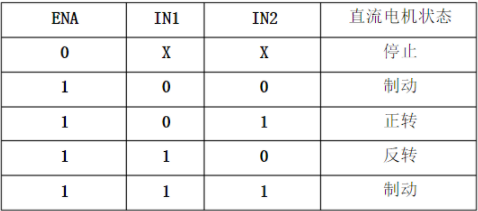


图3.6 L298N驱动模块的控制引脚

当给 ENA PWM的时候就可以实现对电机的调速。

在软件实现上，PWM是通过定时器来实现的，定时器的头文件程序如图3.7：

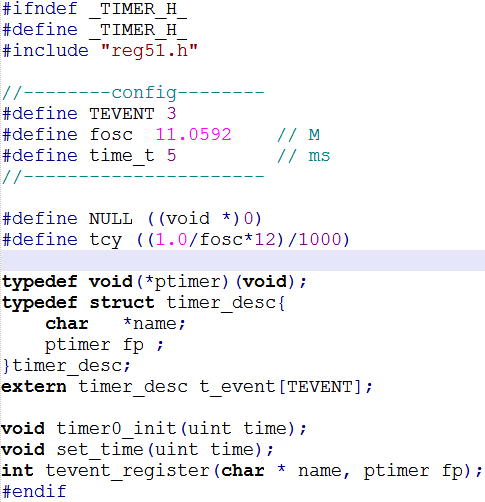
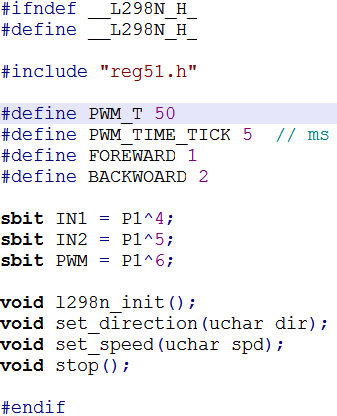
 

图2.7 定时器及电机驱动程序头文件

上图为对应定时器和电机驱动的头文件，具体程序见附录。

定时器：

图3.7中fosc为系统晶振频率，tcy为机器周期，time\_t为定制器单次定时时间，通过这个参数计算出定时器TH和TL的值，TEVENT为能注册的最大处理事件量。ptiemr 为事件处理函数指针，time\_desc 为事件描述结构体，包括事件的名称name事件处理函数指针ptimer。

当需要开启定时器时，首先调用time0\_init(uint time) 函数，time为需要定时的时间，最好设置time为time\_t的倍数，因为在中断中维护了一个中断次数的变量times,每次进入中断都会自增1，当times的值等于time/time\_t时，才会触发调用内部事件处理函数static void timer0\_handle()。此函数会遍历t\_event[TEVENT]中的各事件结构体，若其中的ptimer事件函数指针不为空指针（NULL），则会直接调用ptiemr指定的函数。而tevent\_register(char\* name, ptimer fp);函数则是注册事件结构体，即遍历t\_event[TEVENT]，若结构体中ptimer为空指针（NULL），则说明此结构体未被注册，因此会使用此结构体对申请注册的事件进行注册。

电机驱动：

PWM\_T为周期的计数最大值，同样也是调速的级数，PWM\_TIME\_TICK为每次计数的时间，因此PWM的周期为 PWM\_T \* PWM\_TIME\_TICK。

首次调用time0\_init(PWM\_TIME\_TICK);函数开启定时器，定时时间为PWM\_TIME\_TICK，然后调用l298n\_init();会向定时器注册一个名为PWM的事件，指定处理函数为void pwm();此函数会对设定的占空比pwm\_h进行比较，当前的计数值小于pwm\_h时，PWM引脚输出高，反之，输出低，当大于最大计数值PWM\_T是时，当前计数值置0，从而实现的占空比可调的PWM。

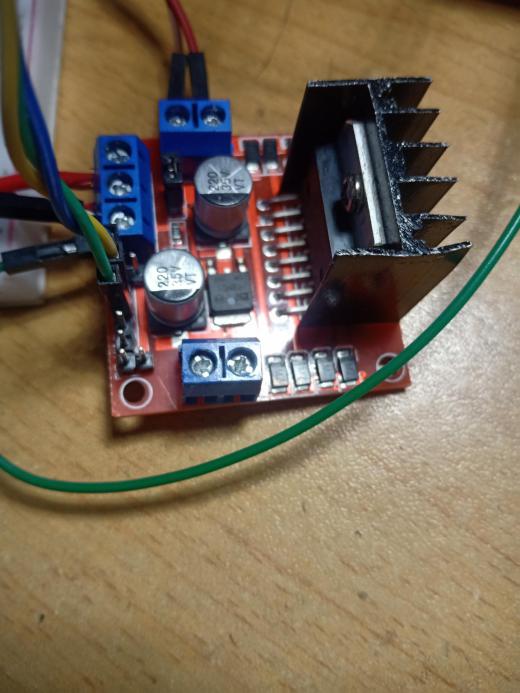
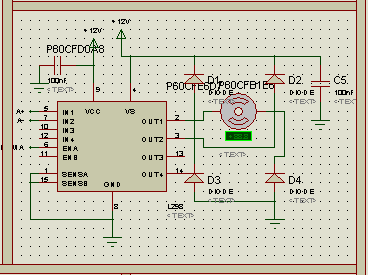


图2.8 电机驱动

2.7 按键设置模块

这里使用了三个按键，第一个按键为模式切换按键，在程序中，定义了一个枚举类型，设置了三个模式分别为NORMAL、HIGH\_THRESHOLD、LOW\_THRESHOLD，即正常运行模式、设置温度上限模式、设置下限模式。每按一次此按键，系统将按上述顺序对系统工作模式进行切换。第二个和第三个按键只在系统处于HIGH\_THRESHOLD、LOW\_THRESHOLD这两个模式时有效，分别为增加和减少上下限温度阈值。

当切换到NORMAL模式时，LCD1602上光标会消失，当处于HIGH\_THRESHOLD模式时，光标会出现在对应H下面并闪烁提示设置上限阈值，当处于LOW\_THRESHOLD模式时，光标会出现在对应L下面并闪烁提示设置下限阈值。

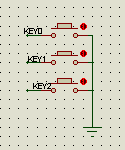
 

图2.9 按键模块

2.8 报警模块

报警模块使用的是LED和蜂鸣器，当电机超过了温度上限或者低于温度下限时，LED会闪烁，同时蜂鸣器发出报警声。

LED的闪烁同样借助了定时器，向定时器注册了名为LED的事件，在注册的处理函数中，定义了一个static计数变量去控制闪烁的频次。

蜂鸣器通过PNP晶体三极管进行控制，当BEEP=0时，三极管导通，蜂鸣器发出声音，当BEEP=1是，三极管截止，蜂鸣器停止工作。

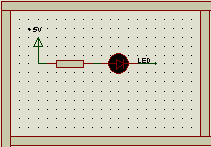
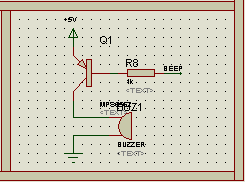
 

图2.10 报警模块

**三、 软件设计**

3.1 定时器设计

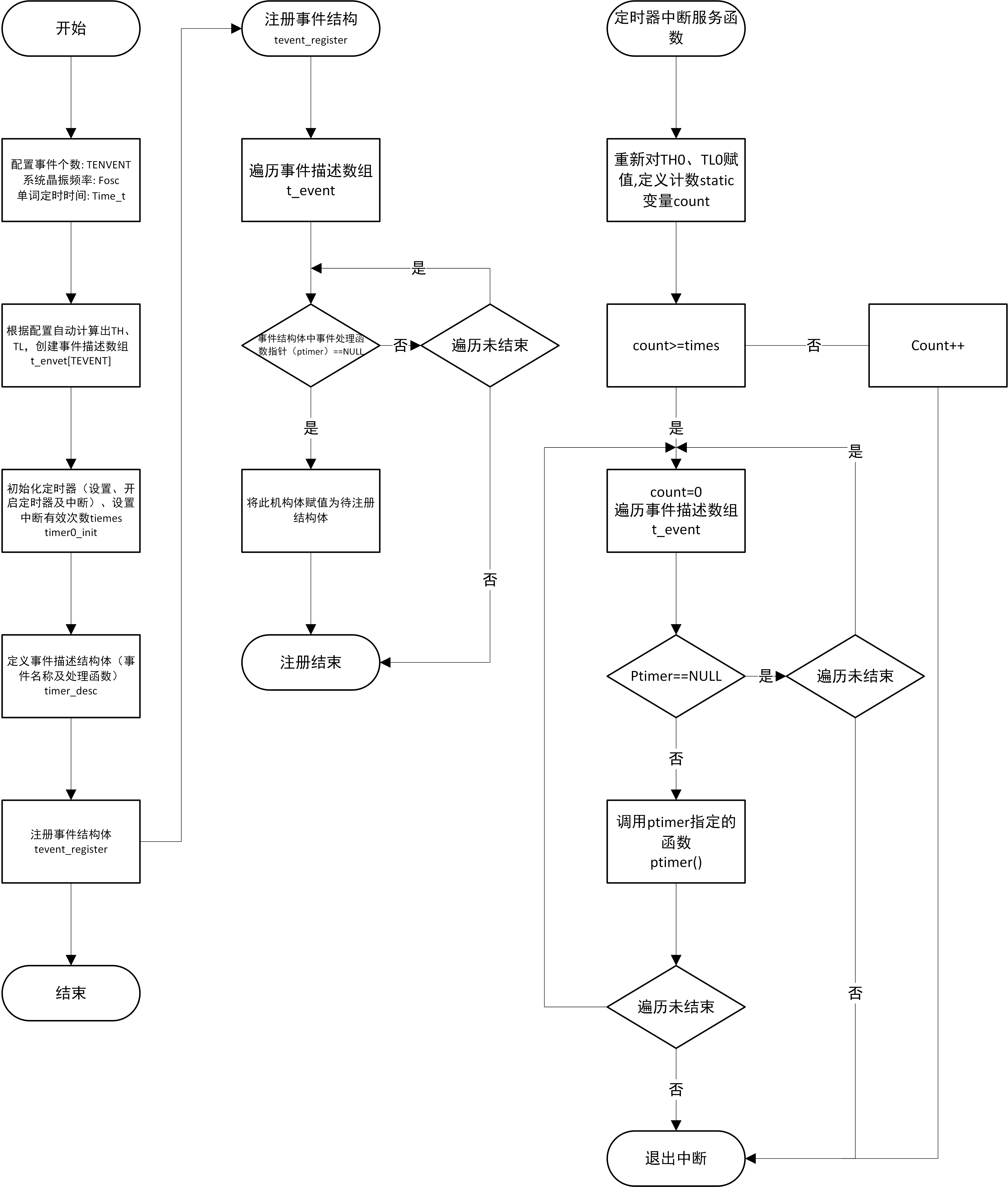


图3.1 定时器流程图

3.2 电机驱动

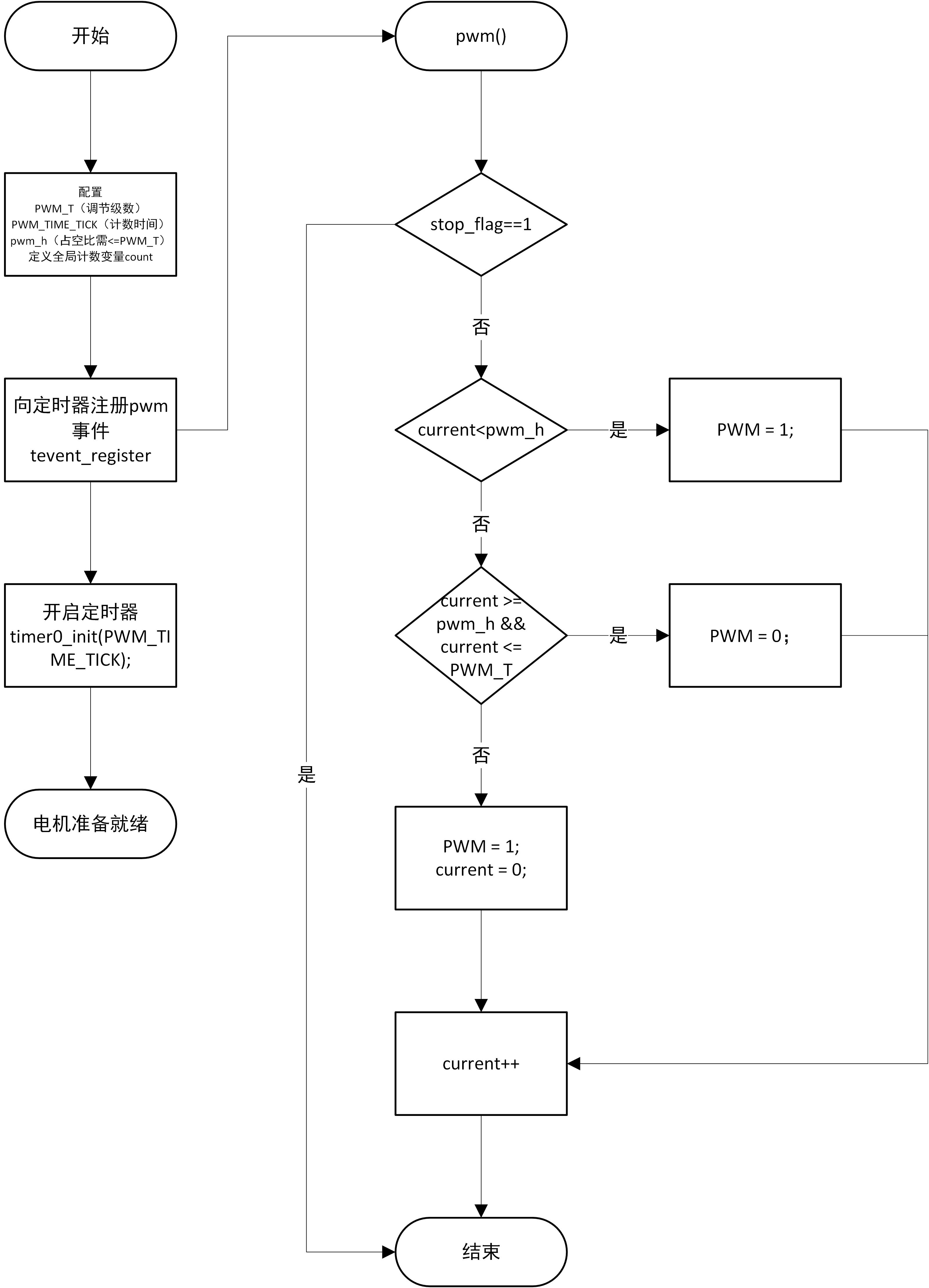


图3.2 电机驱动流程图

3.3 获取温度



图3.3 温度获取流程图

3.4 主程序

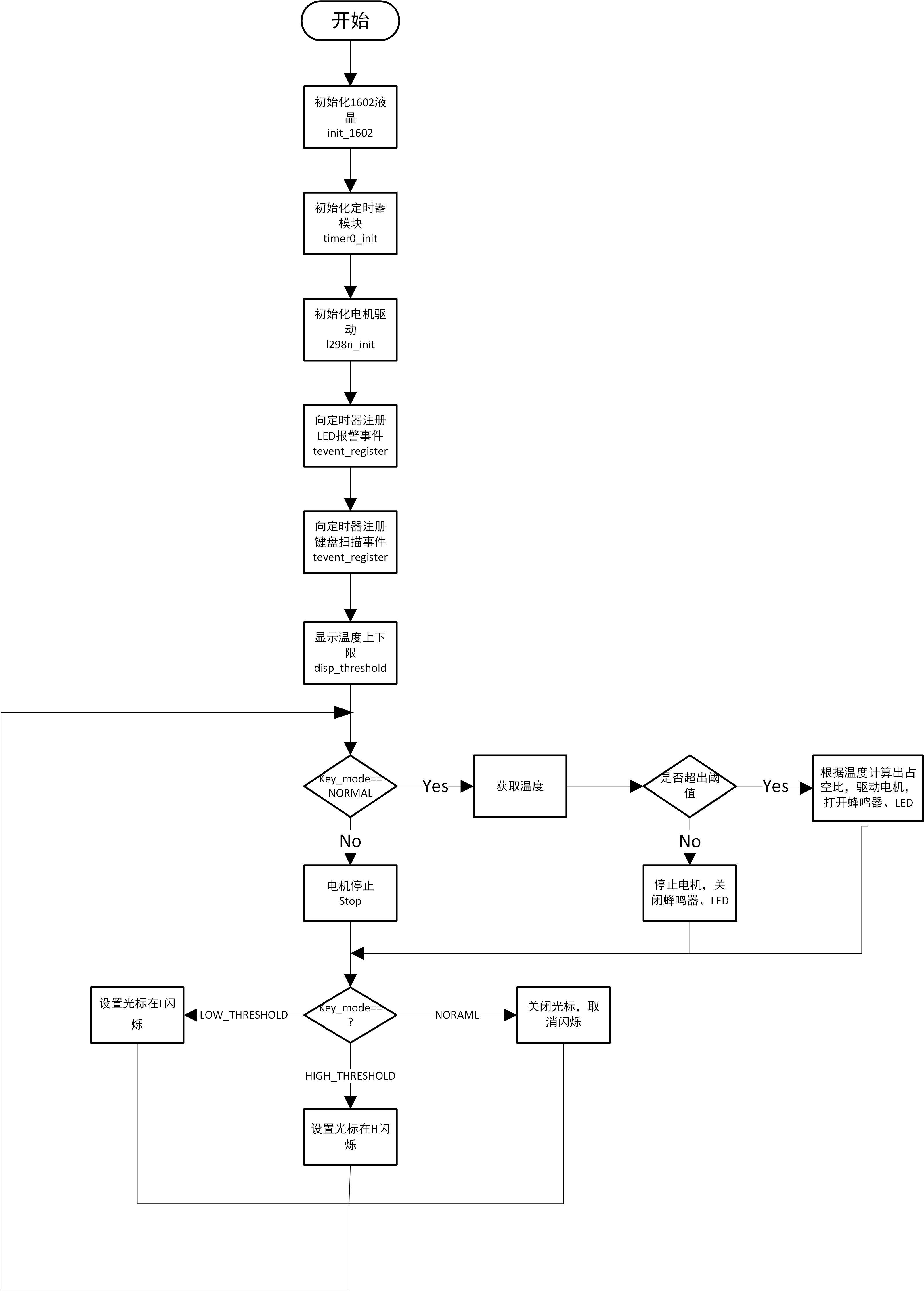


图3.4 主程序流程图

**四、电路仿真与实物演示**

4.1 温度检测与显示

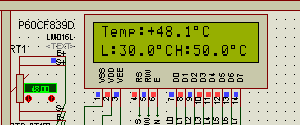


图4.1 温度测量

从图4.1左边仿真中可以看到，设置的当前温度为48℃，检测出的温度为48.1℃，而右边为实物测量的是室温，为25.6℃。

4.2 电机驱动

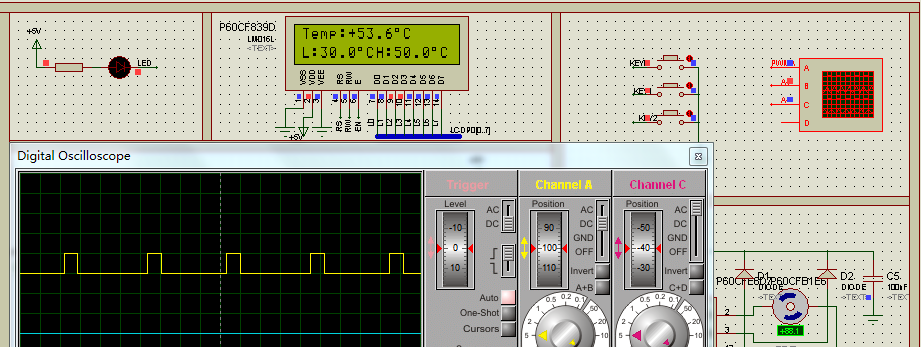


图4.2 电机驱动（1）

从图4.2中可以看到此时超过温度上限值约3℃，此时ENA输出PWM波，同时电机正转，此时的转速为+88。

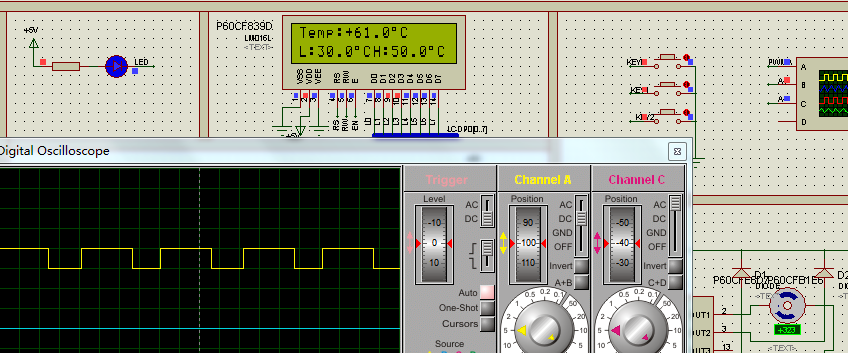


图4.3 电机驱动（2）

从图4.3中可以看到此时温度超过温度上限值11℃，ENA输出PWM波的占空比变大，同时电机速度也提升为+323。

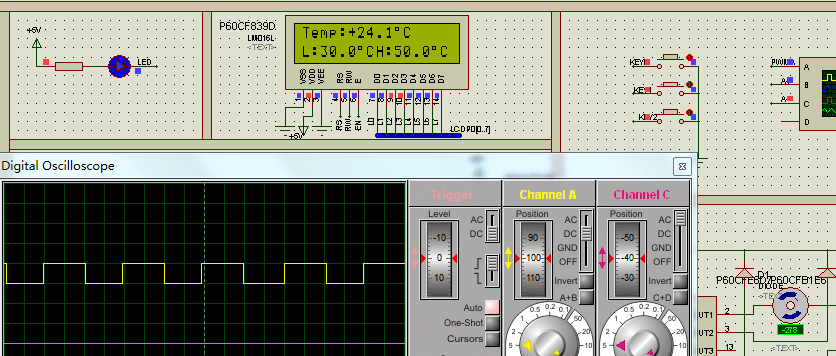


图4.4 电机驱动（3）

从图4.4中可以看到此时温度低于温度下限值6℃，此时电机处于反转状态，速度为-278。

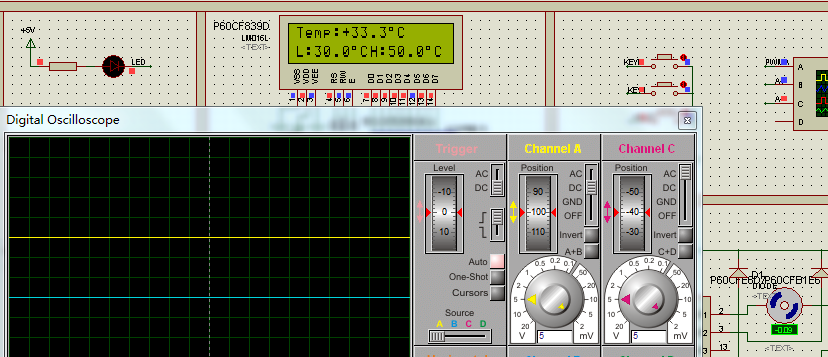


图4.5 电机驱动（4）

从图4.5中可以看到此时的温度在上下限温度之间，ENA输出为0，电机速度也为0。

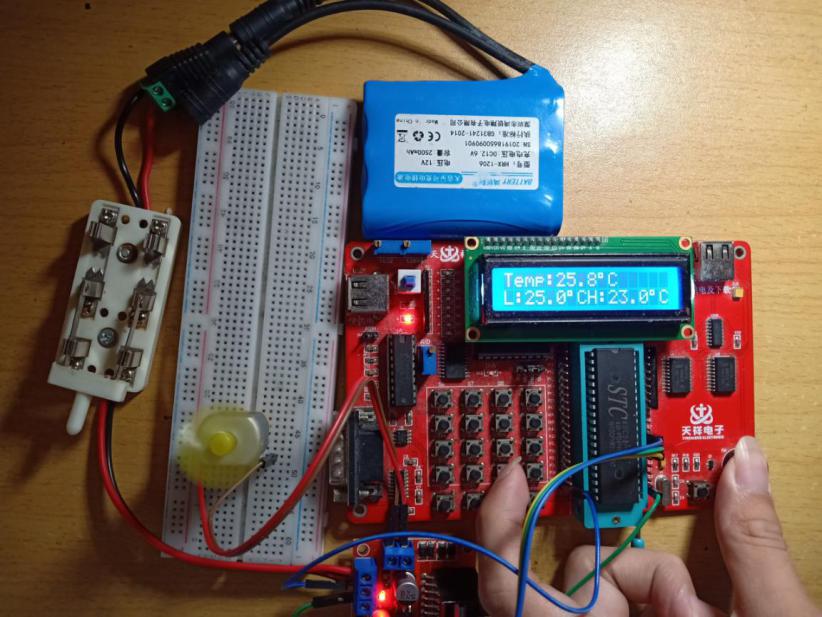


图4.6 电机驱动（实物）

4.3 温度上下限设置

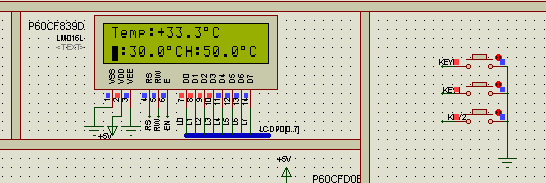


图4.7 按键模式切换

如图4.7，当按一次第一个按键时，光标会出现在L的位置，同时闪烁，提示可以设置上限值。

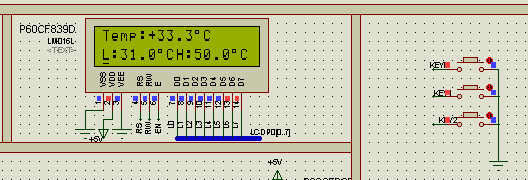


图4.8 按键温度上下限设置（1）

当按下第二个按键后，下限值加一变为31℃。

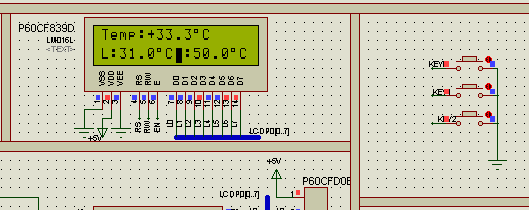


图4.9 按键温度上下限设置（2）

当再次按下第一个按键时，关标出现在H处，同时闪烁，提示可以设置上限值。

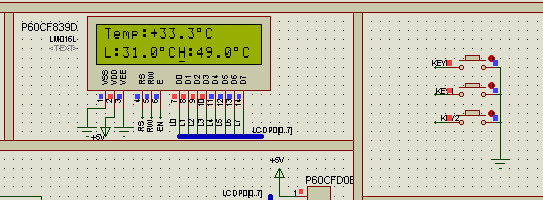


图4.10 按键温度上下限设置（3）

当按下第三个按键时，上限值减一，变为49℃。

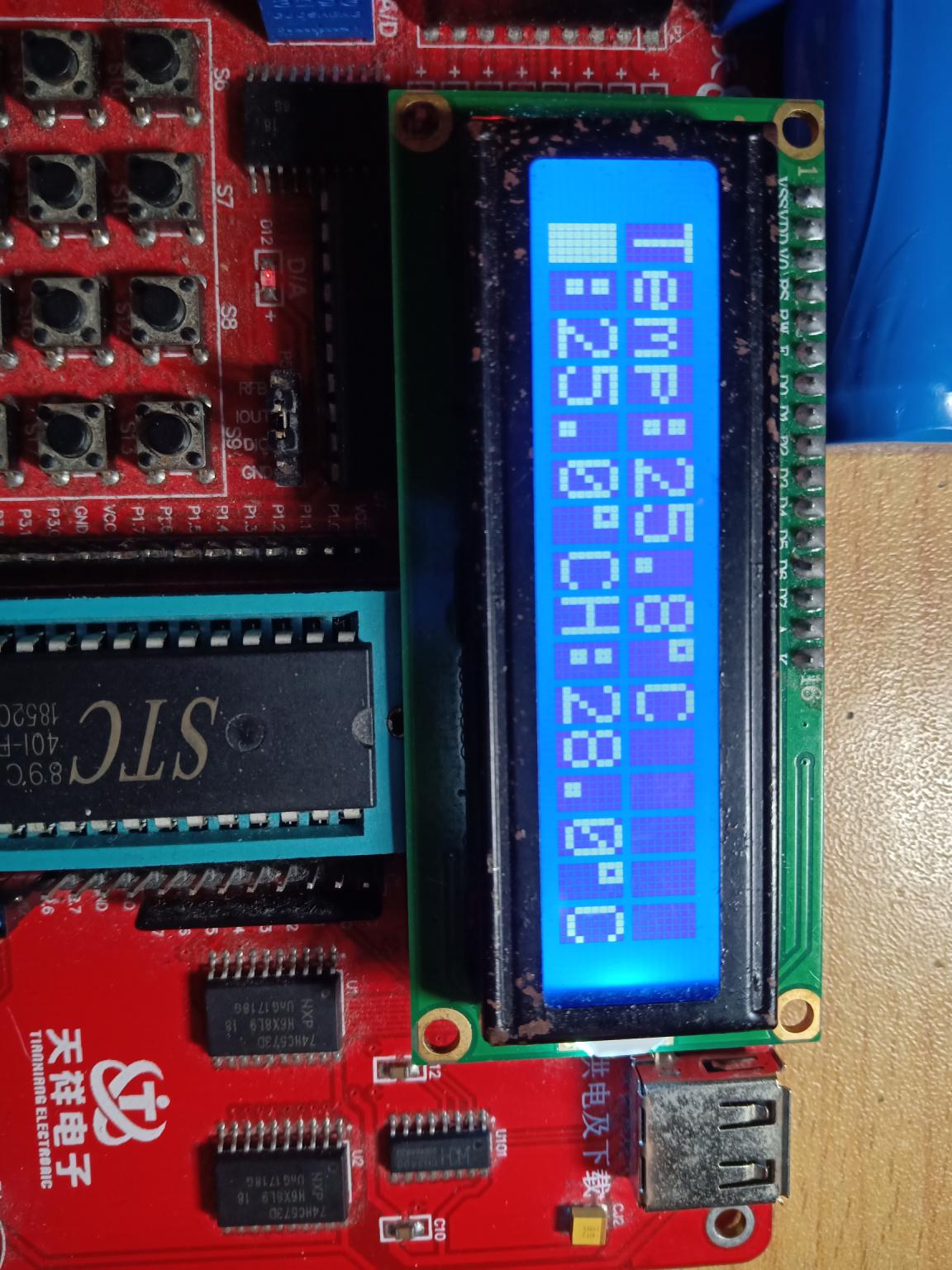
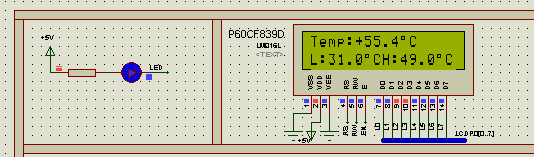
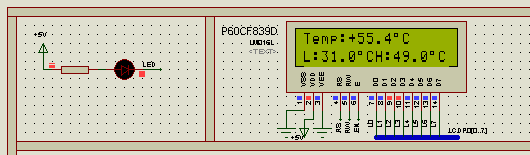


图4.11 按键温度上下限设置（实物）

4.4 报警模块





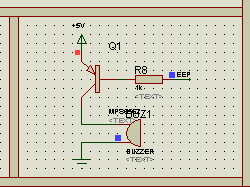


图4.12 报警演示

从图4.12中可以看到，此时温度超过上限值，LED灯开始闪烁，同时BEEP端输出低电平，三极管导通，蜂鸣器发出声音，达到报警效果。

**五、小结**

通过这次课程设计，我觉得自己无论是动手能力还是对课堂中所学的知识都有了不小的提高。

对于Protues仿真软件，大多数人都是第一次接触，从安装、熟悉到比较熟练的使用，大家在使用中都能及时发现问题，查阅资料，解决问题，互帮互助，互相学习，共同提高。

对于在设计当中遇到的问题，大家也会积极讨论解决，通过对大家的问题进行学习，也能从中获益不少，有些是自己也会遇到的问题，有些虽然自己没有遇到，但也能从中学习到别人的设计思想，丰富自己的知识。

学习不能永远停留在理论上，实践能够巩固我们的知识，甚至更加深入的去理解课本中的知识，在本次课程设计中，自己通过仿真去验证了书本上的知识，在实现过程中，自己也发现了许多细节上的处理是在课本学习中所容易忽略的，有些参数的细微调整会输出完全不同的结果，这些也只有通过实践才能够有更加深刻的体会。

**六、参考文献**

1.朱清慧《Proteus教程—电子线路设计、制版与仿真(第3版）》清华大学出版社2016

2.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

3.许维蓥《Proteus电子电路设计及仿真》电子工业出版社2014

### 附录一

-------------------------------------------------------------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 fosc 11.0592 // M

#define time\_t 5 // ms

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

#define NULL ((void \*)0)

#define tcy ((1.0/fosc\*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/tcy) / 256;

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

}

### **附录二**

