计算机科学与技术专业 8 班 学号：21160814 教学号：53160814 姓名：王鏖清

实验五 重量测量

一、实验原理

1、使用 Keil 编程环境进行汇编程序的编写，使用程序下载工具将编译过后的程序加载到单片
2、使用点阵式液晶显示屏显示点阵字符。
3、掌握模拟/数字（A/D）转换方式，使用 C51 语言编写实现重量测量的功能。

二、实验内容

编写 C51 程序，使用重量测量实验板测量标准砝码的重量，将结果（以克计）显示到液晶屏上。误差可允许的范围之间。

三、程序流程图

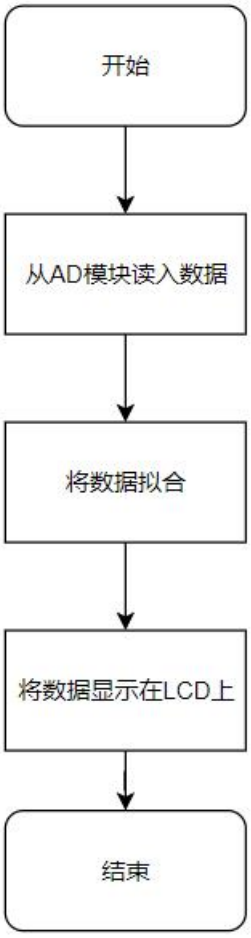

图 1.1 简单流程图

四、实验过程

1、先跟据辅助材料，建立 Keil 工程，并在工程中添加一个汇编语言的文件。下面就是汇编语言的分析编程过程。

2、尝试编写基本的输出命令和数据的子程序，使用软件进行点阵字模的生成。使用字模软件 PCtoLCM2002，生成点阵数据。

3、调零，放上砝码，记录显示屏上的数据。使用不同质量的砝码，重复多次。拟合一个近似曲线函数，进行拟合。

如上图 1.1 所示，实验主要分为两个部分：LCM 的显示和 AD 转换。

图 1.2 AD 转换示意图

LCM 的显示存在一个 BUSY 位，当 BUSY 位为 1 时，表示总线被占用，不可以进行数据的传输。具体流程如图 1.3 所示。

图 1.3 液晶显示屏原理

五、实验代码

```
#include <reg52.h>
#include <intrins.h>
#define uchar unsigned char
#define uint unsigned int
#define LCM_databus P2//LCM 8bit data bus
uchar num;
sbit RS=P3^5;
sbit RW=P3^4;
sbit EN=P3^3;
sbit CS1=P1^7;
sbit CS2=P1^6;
uint loop=0;
uint times=0;
uchar code
fa[]={0x02,0x82,0xE2,0x5E,0x42,0xC2,0x02,0x10,0x10,0x10,0xFF,0x10,0x10,0x18,
0x10,0x00,0x02,0x01,0x7F,0x10,0x10,0x3F,0x01,0x21,0x79,0x27,0x21,0x29,0x31,0
x61,0x01,0x00};
uchar code
ma[]={0x02,0x82,0xE2,0x5E,0x42,0xC2,0x00,0x02,0xFA,0x82,0x82,0x82,0xFE,0x8
0,0x00,0x00,0x01,0x00,0x7F,0x10,0x10,0x3F,0x00,0x04,0x04,0x04,0x44,0x84,0x40,
0x3F,0x00,0x00};
uchar code
zhong[]={0x08,0x08,0x0A,0xEA,0xAA,0xAA,0xAA,0xFF,0xA9,0xA9,0xA9,0xE9,0
x08,0x08,0x08,0x00,0x40,0x40,0x48,0x4B,0x4A,0x4A,0x4A,0x7F,0x4A,0x4A,0x4A,
0x4B,0x48,0x40,0x40,0x00};
uchar code
liang[]={0x40,0x40,0x40,0xDF,0x55,0x55,0x55,0xD5,0x55,0x55,0x55,0xDF,0x40,0x
40,0x40,0x00,0x40,0x40,0x40,0x57,0x55,0x55,0x55,0x7F,0x55,0x55,0x55,0x57,0x5
0,0x40,0x40,0x00};
uchar code
wei[]={0x00,0x10,0x10,0x12,0x14,0x1C,0x10,0xF0,0x9F,0x10,0x10,0x10,0x10,0xF
8,0x10,0x00,0x00,0x00,0x40,0x20,0x10,0x08,0x06,0x01,0x00,0x11,0x26,0x40,0x20,
0x1F,0x00,0x00};
uchar code
mao[]={0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x0
0,0x00,0x00,0x00,0x00,0x36,0x36,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00};
uchar code
L0[]={0x00,0x00,0x00,0xF8,0x04,0x02,0x02,0x02,0x02,0x02,0x04,0xF8,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x1F,0x20,0x40,0x40,0x40,0x40,0x40,0x20,0x1F,0x00,0
x00,0x00,0x00};
uchar code
```

```
L1[]={0x00,0x00,0x00,0x00,0x00,0x08,0x04,0xFE,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x40,0x40,0x7F,0x40,0x40,0x00,0x00,0x00,0
x00,0x00,0x00};
uchar                                                                    code
L2[]={0x00,0x00,0x00,0x18,0x04,0x02,0x02,0x02,0x82,0x82,0x84,0x78,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x78,0x44,0x42,0x41,0x41,0x40,0x40,0x40,0x70,0x00,0
x00,0x00,0x00};
uchar                                                                    code
L3[]={0x00,0x00,0x00,0x0C,0x02,0x02,0x02,0x82,0x82,0x42,0x22,0x1C,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x30,0x40,0x40,0x40,0x40,0x40,0x41,0x22,0x1C,0x00,0
x00,0x00,0x00};
uchar                                                                    code
L4[]={0x00,0x00,0x00,0x00,0x00,0x80,0x60,0x1C,0x02,0xFE,0x00,0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x0C,0x0A,0x09,0x08,0x48,0x48,0x7F,0x48,0x48,0x08,0x00,
0x00,0x00,0x00,0x00};
uchar                                                                    code
L5[]={0x00,0x00,0x00,0x00,0xFE,0x82,0x42,0x42,0x42,0x42,0x42,0x82,0x02,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x31,0x40,0x40,0x40,0x40,0x40,0x40,0x20,0x1F,0x00,0
x00,0x00,0x00,0x00};
uchar                                                                    code
L6[]={0x00,0x00,0x00,0x00,0xF8,0x04,0x82,0x82,0x82,0x82,0x82,0x04,0x18,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1F,0x21,0x40,0x40,0x40,0x40,0x40,0x21,0x1E,0x00,0
x00,0x00,0x00,0x00};
uchar                                                                    code
L7[]={0x00,0x00,0x00,0x00,0x0E,0x02,0x02,0x02,0x02,0x82,0x42,0x32,0x0E,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x70,0x0E,0x01,0x00,0x00,0x00,0x00,0x00,0
x00,0x00,0x00,0x00};
uchar                                                                    code
L8[]={0x00,0x00,0x00,0x00,0x38,0x44,0x82,0x82,0x82,0x82,0x82,0x44,0x38,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x1E,0x21,0x40,0x40,0x40,0x40,0x40,0x21,0x1E,0x00,0
x00,0x00,0x00,0x00};
uchar                                                                    code
L9[]={0x00,0x00,0x00,0x00,0x78,0x84,0x02,0x02,0x02,0x02,0x02,0x84,0xF8,0x00,0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x18,0x20,0x41,0x41,0x41,0x41,0x41,0x20,0x1F,0x00,0
x00,0x00,0x00,0x00};
uchar                                                                    code
ke[]={0x00,0x04,0x04,0xE4,0x24,0x24,0x24,0x3F,0x24,0x24,0x24,0xE4,0x04,0x04,
0x00,0x00,0x00,0x00,0x00,0x80,0x43,0x31,0x0F,0x01,0x01,0x01,0x3F,0x41,0x43,0x40,0
x40,0x70,0x00};
void delay(uint i)
{while(--i);}
void Read_busy()
{
    P2=0x00;
```

```
    RS=0;
    RW=1;
    EN=1;
    while(P2&0x80);
    EN=0;
}
void SelectScreen(uchar screen)
{
  switch(screen)
  {
        case 0: CS1=1;CS2=1;break;
        case 1: CS1=1;CS2=0;break;
        case 2: CS1=0;CS2=1;break;
        default: break;
  }
}
void write_LCM_command(uchar value){
     LCM_databus=0xff;
     Read_busy();
    RS=0;
    RW=0;
    LCM_databus=value;
    EN=1;
    delay(100);
    EN=0;
}
void write_LCM_data(uchar value)
{
     LCM_databus=0xff;
     Read_busy();
    RS=1;
    RW=0;
    LCM_databus=value;
    EN=1;
    delay(100);
    EN=0;
}
void Set_page(uchar page)
{
     page=0xb8|page;
     write_LCM_command(page);
}
void Set_line(uchar startline)
{
```

```
   startline=0xC0|startline;
     write_LCM_command(startline);
}

void Set_column(uchar column)
{
   column=column&0x3f;
     column=0x40|column;
   write_LCM_command(column);
}

void SetOnOff(uchar onoff)
{
   onoff=0x3e|onoff;
   write_LCM_command(onoff);
}

void ClearScreen(uchar screen)
{
   uchar i,j;
   SelectScreen(screen);
   for(i=0;i<8;i++)
   {
      Set_page(i);
       Set_column(0);
     for(j=0;j<64;j++){
       write_LCM_data(0x00);
     }
   }
}
void InitLCM()
{
    Read_busy();
     SelectScreen(0);//
     SetOnOff(1);//
     SelectScreen(0);
    ClearScreen(0);//
     Set_line(0);//
}
void Display(uchar ss,uchar page,uchar column,uchar *p)
{
   uchar i;
   SelectScreen(ss);
   Set_page(page);
```

```
    Set_column(column);
    for(i=0;i<16;i++)
    {
       write_LCM_data(p[i]);//??16???
    }
    Set_page(page+1);
    Set_column(column);
    for(i=0;i<16;i++)
    {
       write_LCM_data(p[i+16]);
      }
}
sfr       ADC_CONTR   =   0xBC;
sfr       ADC_RES     =   0xBD;
sfr       ADC_LOW2    =   0xBE;//ADC_RESL
sfr  P1ASF      =   0x9D;
sfr  AURX1      =   0xA2;
#define ADC_POWL2     0X80
#define ADC_FLAG 0X10
#define ADC_START     0X08
#define ADC_SPEEDLL  0X00
#define ADC_SPEEDL    0X20
#define ADC_SPEEDH    0X40
#define ADC_SPEEDHH 0X60
void InitADC_n(uchar n);
uint GET_ADC(uchar n);
uchar GetADCResult(uchar ch);
void Delay(uint n);
//void ShowResult(uchar ch);
/*void ShowResult(uchar ch)
{
     SendData(ch);
     SendData(GetADCResult(ch));
}*/
void InitADC_n(uchar n)
{
     n =n& 0x07;
     AURX1 |= 0x04;
     P1ASF = 1<<n;
}
uint GET_ADC(uchar n)
{
     uint adc_data;
     n &= 0x07;
```

```
    ADC_RES = 0;
    ADC_LOW2 = 0;
    ADC_CONTR = 0;
    ADC_CONTR|= (ADC_POWL2|ADC_SPEEDLL|n|ADC_START);
    _nop_();_nop_();_nop_();_nop_();_nop_();_nop_();
    while(!((ADC_CONTR & ADC_FLAG) == 0x10))
    adc_data = (ADC_RES & 0x03)    * 256 + ADC_LOW2;
    ADC_CONTR &= 0xef;
    return adc_data;//adc_data
}
void Delay1(uint n)
{
    uint x;
    while(n--)
    {x = 5000;
    while(x--);}
}
uint getKg(uint i)
{
    if(i<6)
        return 0;
    if(i<40)
        return 10;
    if(i<65)
        return 20;
    if(i<120)
        return 30;
    if(i<155)
        return 50;
    if(i<175)
        return 60;
    if(i<210)
        return 70;
    return 100;
}
void main(){
    InitLCM();
    while(1){
        uint A = 0;
        uchar a,b,c;
        InitADC_n(0);
        Display(1,loop,2*16,fa);
        Display(1,loop,3*16,ma);
        Display(2,loop,0*16,zhong);
```

```
 Display(2,loop,1*16,liang);
 A = GET_ADC(0);
 A = getKg(A);
 a = A/100;
 b = A%100/10;
 c = A%10;
switch(a)
      {
           case 0:Display(1,loop+4,2*16,L0);break;
           case 1: Display(1,loop+4,2*16,L1);break;
           case 2: Display(1,loop+4,2*16,L2);break;
           case 3: Display(1,loop+4,2*16,L3);break;
           case 4: Display(1,loop+4,2*16,L4);break;
           case 5: Display(1,loop+4,2*16,L5);break;
           case 6: Display(1,loop+4,2*16,L6);break;
           case 7: Display(1,loop+4,2*16,L7);break;
           case 8: Display(1,loop+4,2*16,L8);break;
           case 9: Display(1,loop+4,2*16,L9);
      }
 switch(b)
 {
           case 0:Display(1,loop+4,3*16,L0);break;
           case 1:Display(1,loop+4,3*16,L1);break;
           case 2:Display(1,loop+4,3*16,L2);break;
           case 3:Display(1,loop+4,3*16,L3);break;
           case 4:Display(1,loop+4,3*16,L4);break;
           case 5:Display(1,loop+4,3*16,L5);break;
           case 6:Display(1,loop+4,3*16,L6);break;
           case 7:Display(1,loop+4,3*16,L7);break;
           case 8:Display(1,loop+4,3*16,L8);break;
           case 9:   Display(1,loop+4,3*16,L9);
 }
 switch(c){
           case 0:Display(2,loop+4,0*16,L0);break;
           case 1:   Display(2,loop+4,0*16,L1);break;
           case 2:   Display(2,loop+4,0*16,L2);break;
           case 3:   Display(2,loop+4,0*16,L3);break;
           case 4:   Display(2,loop+4,0*16,L4);break;
           case 5:   Display(2,loop+4,0*16,L5);break;
           case 6:   Display(2,loop+4,0*16,L6);break;
           case 7:   Display(2,loop+4,0*16,L7);break;
           case 8:   Display(2,loop+4,0*16,L8);break;
           case 9:   Display(2,loop+4,0*16,L9);
 }
```

```
            Display(2,loop+4,1*16,ke);
            Delay1(50);
    }
}
```

六、思考题

1. 调零的原理，软件调零和调零调零的区别。

调零是指在未放置砝码时，液晶屏示数应该是 0。

软件调零是在程序中通过拟合函数使得没有砝码时，显示为 0。

而，硬件调零是通过调整压敏电阻的阻值，进行调零。

2. 模/数和数/模的信号转换原理。

A/D 转换器是用来通过一定的电路将模拟量转变为数字量。模拟量可以是电压、电流等电信号，也可以是压力、温度、湿度、位移、声音等非电信号。

但在 A/D 转换前，输入到 A/D 转换器的输入信号必须经各种传感器把各种物理量转换 成电压信号。

A/D 转换器的工作原理方法： 逐次逼近法：基本原理是从高位到低位逐位试探比较，好像用天平称物体，从重 到轻逐级增减砝码进行试探。逐次逼近法转换过程是：初始化时将逐次逼近寄存 器各位清零；转换开始时，先将逐次逼近寄存器最高，送入 D/A 转换器，经 D/A 转换后生成的模拟量送入比较器，称为 Vo，与送入比较器的待转换的模拟量 Vi 进行比较，逼近寄存器最低位。转换结束后，将逐次逼近寄存器中的数字量送入缓冲寄存器。

DA 转换器的内部电路构成无太大差异，一般按输出是电流还是电压、能否作乘 法运算等进行分类。

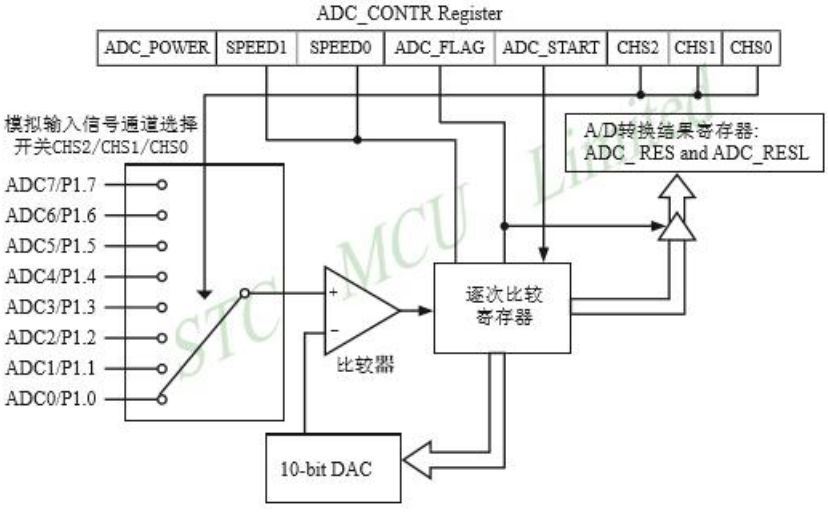

图 1.4 AD 原理图

3. I^2C 总线在信号通讯过程中的应用。

I2C 总线是用于在连接于总线上的器件之间传送信息，提供集成电路（ICs）之间的通信线路，广泛应用于电视，录像机和音频等设备。I2C 总线的意思：“完成集成电路或功能单元之间信息交换的规范或协议”。

实验六 直流电机脉宽调制调速

一、实验原理

1、 了解脉宽调制调速的原理与方法。
2、 使用频率和周期测量的方法。
3、了解闭环控制的原理。

二、实验内容

1、在液晶显示屏上显示出直流电机的当前转速、低目标转速、高目标转速。
2、向 P1.1 输出 0，测量每秒钟电机转动的转数，将其显示在数码管，每秒刷新一次。
3、使用脉宽调制的方法，动态调整直流电机的转速，使得电机转速能够稳定在一个预定值附近。

三、程序流程图

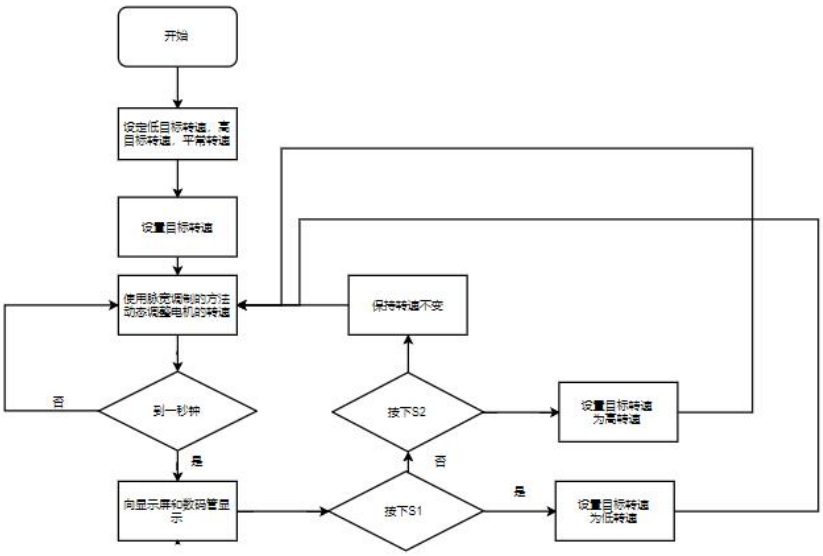

图 2.1 系统流程图

四、实验过程

实验主要包括两个部分：显示部分和控制直流电机的转动部分。

显示部分中的 LCM 显示在实验五中已经进行了阐述。数码管是 8 位边沿触发式，时钟每次由低变高时，数据右移一位，这里在前面的汇编语言实验中也有所涉及。

其中，控制部分有三个中断，其中包括一个外部中断和两个定时中断。分别为直流电机的外部中断，控制 1s 的定时中断和控制 0.1ms 的定时器中断。

当直流电机转动一圈时，光圈被遮挡，便产生一个上升沿的外部中断。电机转速就是一秒钟之内 INT0 的中断个数。

题目要求每秒测量电机的转数以及每秒进行数码管的刷新。其中，使用了 25ms 的定时中断，这个中断进行了 20 次以完成 1s 的定时器中断。

控制转速的时候，使用了脉冲调制的方法。所以，本文使用了一个 0.1ms 的定时中断，在这个中断后动态改变 P0 的输出。

在每一个 1 秒钟中断测量出当前转速之后，将其与目标值相对比，如果不够则增加控制变量，否则减少之，这样就能逐步达到稳定转速的目的。同时将速度显示出来。

为了达到稳定输出的效果，这次实验使用了累加进位法。设置一个累加变量 x，每次加 N，若结果大于 256，则输出 0，并减去 256；否则输出 1。这样就可以使直流电机控制在一预定数值附近。

五、实验代码

```
#include <reg52.h>
#include <intrins.h>
#define uchar unsigned char
#define uint unsigned int
sfr P4=0xC0;
sfr P4SW=0xBB;
sbit sclk=P4^4;
sbit sdata=P4^5;
sbit CS1=P1^7;
sbit CS2=P1^6;
sbit E=P3^3;
sbit RW=P3^4;
sbit RS=P3^5;
sbit RES=P1^5;
sbit BUSY=P2^7;
sbit swh1=P3^6;
sbit swh2=P3^7;
sbit motor=P1^1;
uchar code zima[20][32]=
{
0x00,0x00,0xC0,0xE0,0x30,0x10,0x08,0x08,0x08,0x08,0x08,0x18,0x30,0xE0,0xC0,
0x00,
0x00,0x00,0x07,0x0F,0x18,0x10,0x20,0x20,0x20,0x20,0x20,0x10,0x18,0x0F,0x07,0
x00,
0x00,0x00,0x00,0x10,0x10,0x10,0x10,0xF0,0xF8,0x00,0x00,0x00,0x00,0x00,0x00,0
x00,
0x00,0x00,0x00,0x20,0x20,0x20,0x20,0x3F,0x3F,0x20,0x20,0x20,0x20,0x00,0x00,0
x00,
0x00,0x00,0x60,0x50,0x10,0x08,0x08,0x08,0x08,0x08,0x08,0x98,0xF0,0x70,0x00,0
x00,
0x00,0x00,0x20,0x30,0x28,0x28,0x24,0x24,0x22,0x22,0x21,0x20,0x30,0x18,0x00,0
x00,
```

0x00,0x00,0x30,0x30,0x08,0x08,0x88,0x88,0x88,0x88,0x58,0x70,0x30,0x00,0x00,0
x00,
0x00,0x00,0x18,0x18,0x20,0x20,0x20,0x20,0x20,0x20,0x31,0x11,0x1F,0x0E,0x00,0
x00,
0x00,0x00,0x00,0x00,0x00,0x80,0x40,0x20,0x10,0xF0,0xF8,0xF8,0x00,0x00,0x00,0
x00,
0x00,0x04,0x06,0x05,0x05,0x04,0x24,0x24,0x24,0x3F,0x3F,0x3F,0x24,0x24,0x24,0
x00,
0x00,0x00,0x00,0xC0,0x38,0x88,0x88,0x88,0x88,0x88,0x88,0x88,0x08,0x08,0x00,0
x00,
0x00,0x00,0x18,0x29,0x21,0x20,0x20,0x20,0x20,0x20,0x30,0x11,0x1F,0x0E,0x00,0
x00,
0x00,0x00,0x80,0xE0,0x30,0x10,0x98,0x88,0x88,0x88,0x88,0x88,0x98,0x10,0x00,0
x00,
0x00,0x00,0x07,0x0F,0x19,0x31,0x20,0x20,0x20,0x20,0x20,0x20,0x11,0x1F,0x0E,0
x00,
0x00,0x00,0x30,0x18,0x08,0x08,0x08,0x08,0x08,0x88,0x48,0x28,0x18,0x08,0x00,0
x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x38,0x3E,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0
x00,
0x00,0x00,0x70,0x70,0xD8,0x88,0x88,0x08,0x08,0x08,0x08,0x98,0x70,0x70,0x00,0
x00,
0x00,0x0C,0x1E,0x12,0x21,0x21,0x20,0x21,0x21,0x21,0x23,0x12,0x1E,0x0C,0x00,
0x00,
0x00,0xE0,0xF0,0x10,0x08,0x08,0x08,0x08,0x08,0x08,0x18,0x10,0xF0,0xC0,0x00,
0x00,
0x00,0x00,0x11,0x33,0x22,0x22,0x22,0x22,0x22,0x32,0x11,0x1D,0x0F,0x03,0x00,0
x00,
0x08,0x08,0x0A,0xEA,0xAA,0xAA,0xAA,0xFF,0xA9,0xA9,0xA9,0xE9,0x08,0x08,
0x08,0x00,
0x40,0x40,0x48,0x4B,0x4A,0x4A,0x4A,0x7F,0x4A,0x4A,0x4A,0x4B,0x48,0x40,0x
40,0x00,
0x40,0x40,0x40,0xDF,0x55,0x55,0x55,0xD5,0x55,0x55,0x55,0xDF,0x40,0x40,0x40,
0x00,
0x40,0x40,0x40,0x57,0x55,0x55,0x55,0x7F,0x55,0x55,0x55,0x57,0x50,0x40,0x40,0
x00,
0x00,0x00,0x00,0x00,0x00,0x00,0xC0,0xC0,0xC0,0xC0,0x00,0x00,0x00,0x00,0x00,
0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x30,0x30,0x30,0x30,0x00,0x00,0x00,0x00,0x00,0
x00,
0x00,0x04,0x04,0xE4,0x24,0x24,0x24,0x3F,0x24,0x24,0x24,0xE4,0x04,0x04,0x00,0
x00,
0x00,0x00,0x80,0x43,0x31,0x0F,0x01,0x01,0x01,0x3F,0x41,0x43,0x40,0x40,0x70,0
x00,

```
};
uchar tab[15]=
       {0xC0,0xF9,0xA4,0xB0,0x99,
            0x92,0x82,0x0F8,0x80,0x90};
uchar tspeed=0;
uchar cspeed=0;
uchar xspeed=120;
uchar speedUp = 140;
uchar speedLow =100;
uchar t1_cnt=0;
int N=40;
int M=256;
int X=0;
void send_byte(uchar dat ,uchar cs1,uchar cs2);
void send_all(uint page,uint lie,uint offset);
void init();
void clearscreen();
void init_yejing();
void sendbyte(uchar ch);
void display(uchar n);
void delay1();
void delay2();
void delay(uint x)
{
      while(x--) ;
}
void main()
{
       init();
       init_yejing();
       motor=0;
       while(1)
       {
      clearscreen();

      send_all(1,3,speedLow/100);
            send_all(1,4,(speedLow/10)%10);
            send_all(1,5,speedLow%10);

      send_all(3,3,cspeed/100);
            send_all(3,4,(cspeed/10)%10);
            send_all(3,5,cspeed%10);

          send_all(5,3,speedUp/100);
```

```
            send_all(5,4,(speedUp/10)%10);
            send_all(5,5,speedUp%10);


          delay1();
                display(cspeed);
       delay(50000);
        }
}
void init()
{       P4SW=0x30;

        IT0=1;
        EA=1;
        ET1=1;
        ET0=1;
        EX0=1;
        TMOD=0x11;
        TH1=0x3C;
        TL1=0xB0;
        TH0=0xFF;
        TL0=0x9C;
        TR0=1;
        TR1=1;
}
void ex_int0() interrupt 0
{
        tspeed++;
}
void t1_int() interrupt 3
{
        if(++t1_cnt<20)
        {           TH1=0x3C;
                            TL1=0xB0;
                if(swh1==0)
                {

                  xspeed = speedLow;

                }


                if(swh2==0){
```

```
                    xspeed = speedUp;

                }
                if(swh1==1 &&swh2==1 ){
                    xspeed = 120;
                }

            return;
        }
        t1_cnt=0;
        cspeed=tspeed;
        tspeed=0;
        if(cspeed>xspeed) N--;
        if(cspeed<xspeed) N++;
}
void t0_int() interrupt 1
{
         TH0=0xFF;
        TL0=0x9C;
                X+=N;
         if(X>M)
         {
                motor=0;
                 X-=M;
         }
         else
                 motor=1;
}
void init_yejing()
{
        send_byte(192,1,1);
        send_byte(63,1,1);
}
void send_byte(uchar dat,uchar cs1,uchar cs2)
{
        P2=0xff;
        CS1=cs1; CS2=cs2;
        RS=0; RW=1; E=1;
        while(BUSY) ;

        E=0;
        RS=!(cs1&&cs2),RW=0;
        P2=dat;
        E=1; delay(3); E=0;
```

```
    CS1=CS2=0;
}

void send_all(uint page,uint lie,uint offset)
{
    uint i,j,k=0;
    for(i=0;i<2;++i)
    {
        send_byte(184+i+page,1,1);
        send_byte(64+lie*16-(lie>3)*64,1,1);
        for(j=0;j<16;++j)
            send_byte(zima[offset][k++],lie<4,lie>=4);
    }
}
void clearscreen()
{
   int i,j;
   for(i=0;i<8;++i)
   {
      send_byte(184+i,1,1);
      send_byte(64,1,1);
         for(j=0;j<64;++j)
            {
               send_byte(0x00,0,1);
               send_byte(0x00,1,0);
            }
   }
}
void sendbyte(uchar ch)
{
    uchar shape,c;
    shape=tab[ch];
    for(c=0;c<8;c++)
    {
        sclk=0;
        sdata=shape & 0x80;
        sclk=1;
        shape <<= 1;
    }
}
void display(uchar n)
{
    sendbyte(n%10);
```

```
        sendbyte((n/10)%10);
        sendbyte(n/100);
}
void delay1()
{
        int i,j;
        for(i=0;i<1000;i++)
                for(j=0;j<500;j++);
}
void delay2()
{
        int i,j;
        for(i=0;i<1000;i++)
                for(j=0;j<1000;j++);
}
```

六、实验中遇到的问题与解答

实验中的 1s 定时器是使用的 20 次 25ms 进行实现。在 1s 计时之后，会进行数码管和 LCM 的显示，所以，应该注意在相应显示之后在开始下一个 1s 的计时。

七、思考题

1. 讨论脉宽调速和电压调速的区别、优缺点和应用范围。

PWM 不需要在计算机接口中使用 D/A 转换器，适用于低频大功率控制。

电压调速是改变加大电枢上的电压大小，一般是连续的供电，电机低速连续转动。电压调速工作时不能超过特定电压，优点是机械特性较硬并且电压降低后硬度不变，稳定性好，适用于对稳定性要求较高的环境

2. 说明程序原理中累加进位法的正确性。

累加进位法：设置一个累加变量 x，每次加 N，若结果大于 M，则输出 1，并 减去 M；否则输出 0。这样整体的占空比是 N/M，所以，就可以达到脉冲调速的功能。

3. 计算转速测量的最大可能误差，讨论减少误差的办法。

在使用定时器中断时，我们会对某些变量进行初始化，这些机器指令执行的周期的时间不会算在在中断计时中。

同时，电机转动 1 周触发 1 次中断，本实验是通过对 1s 触发的中断进行计数来间接得到转速。我们可以在电机转动 1 圈时，多设置几次中断。

实验七 温度测量与控制

一、实验原理

1、了解温度传感器结构和温度测量的原理。
2、掌握 PID 控制原理及实现方法。

二、 实验内容

1 编写实现温度控制的功能，使用测量温度，将温度测量的结果（单位为摄氏度）显示到液晶屏上。
3、 实现测量当前教室的温度，显示在 LCM 液晶显示屏上。
4、通过 S1 设定一个高于当前室温的目标温度值。
4、编程实现温度的控制，将当前温度值控制到目标温度值并稳定的显示。

三、程序流程图

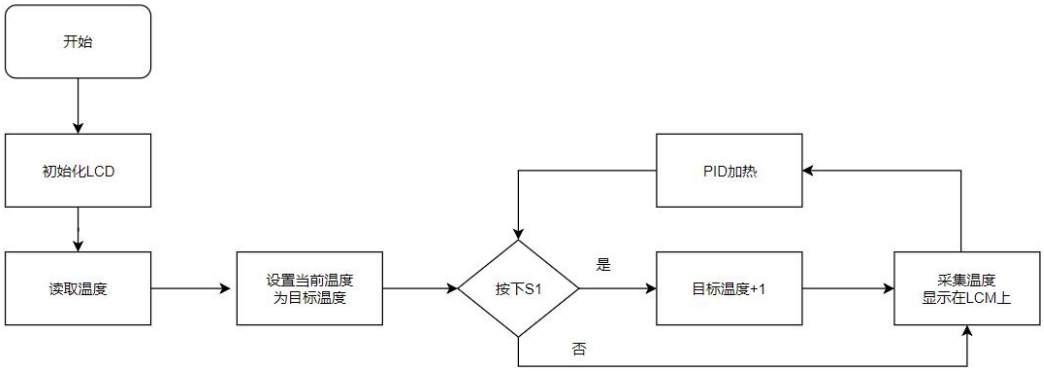

图 3.1 程序流程图

四、实验过程

实验的原理如图 3.2 所示，实验使用 STC89C516RD+单片机实验板。P1.4 与 DS18B20 的 DQ 引脚相连，进行数据和命令的传输。同时，P1.1 连接热电阻。当 P1.1 为高电平时，加热热电阻。

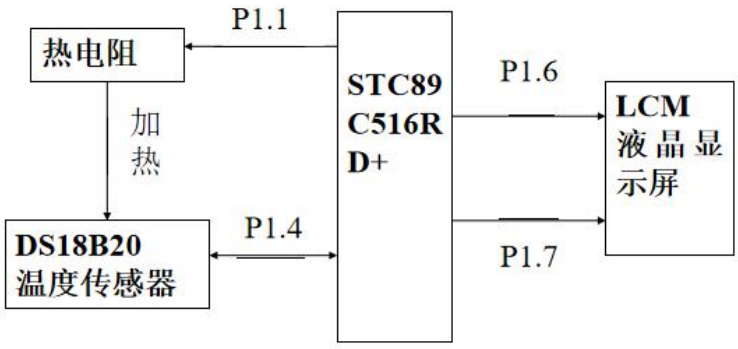

图 3.2 实验原理图

DS18B20 的工作过程如图 3.3 所示。

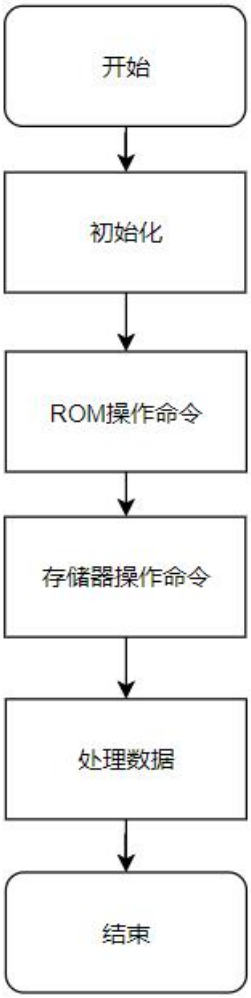

图 3.3 DS18B20 工作流程

其中，初始化过程为：将 DS18B20 的数据总线 P1.4 拉低 500us 以上，然后释放，DS18B20 收到信号后等待，之后发送存在低脉冲，单片机收到该信号表示初始化成功。

接收温度时，实验使用了两个 8 位存贮器 RAM 编号为 0 号和 1 号。

1 号存贮器存放温度值的符号，如果温度为负（℃），则 1 号存贮器 8 位全为 1，否则全为 0。0 号存贮器用于存放温度值的补码 LSB(最低位)的 1 表示 0.5℃。将存贮器中的二进制数求补再转换成十进制数并除以 2，就得到被测温度值。

控制温度的方法是 PID。PID 算法是控制行业最经典、最简单、而又最能体现反馈控制思想的算法。当得到系统的输出后，将输出经过比例，积分，微分 3 种运算方式，叠加到输入中，从而控制系统的行为。PID 算法的执行流程是利用反馈来检测偏差信号，并通过偏差信号来控制被控量。而控制器本身就是比例、积分、微分三个环节的加和。其功能框图如图 3.4：

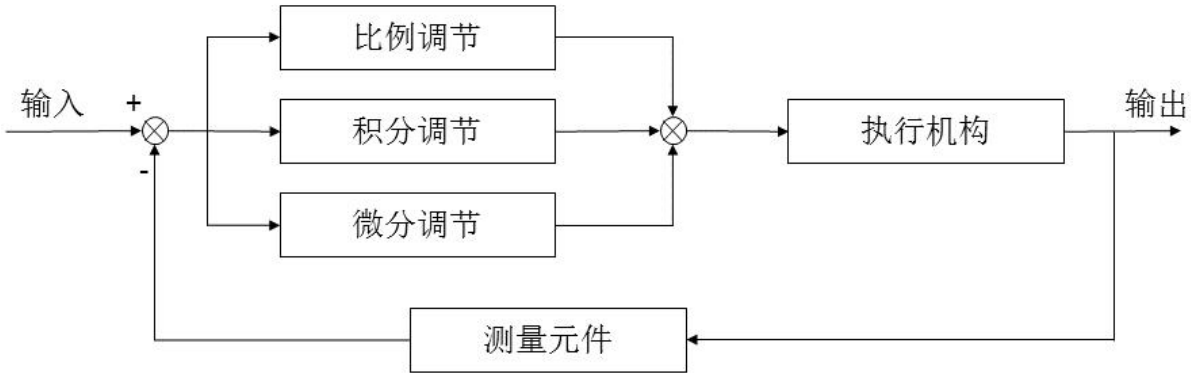

图 3.4 PID 功能图

五、实验代码

```
#include<reg52.h>
#include<intrins.h> //声明本征函数库
#include<math.h>
typedef unsigned char uchar;
typedef unsigned int uint;
sbit s1 = P3^6;
sbit s2 = P3^7;
sbit RS=P3^5;//寄存器选择信号
sbit RW=P3^4;//读写操作选择信号，高电平读，低电平写
sbit EN=P3^3;//使能信号
sbit CS1=P1^7;//左半屏显示信号，低电平有效
sbit CS2=P1^6;//右半屏显示信号，低电平有效
sbit DQ=P1^4;
sbit up=P1^1;
uchar Ek,Ek1,Ek2;
uchar Kp,Ki,Kd;
uint res,Pmax;
uint xx=0; //页面
uint times=0;//延时函数
void delay_us(uchar n)
{
  while (n--)
  {
  _nop_();
  _nop_();
  }
}
unsigned    char code shu[10][32]={
{0x00,0x00,0x00,0xF8,0x04,0x02,0x02,0x02,0x02,0x02,0x04,0xF8,0x00,0x00,0x00,
0x00,
0x00,0x00,0x00,0x1F,0x20,0x40,0x40,0x40,0x40,0x40,0x20,0x1F,0x00,0x00,0x00,0
```

```
x00},
{0x00,0x00,0x00,0x00,0x00,0x08,0x04,0xFE,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,
0x00,0x00,0x00,0x00,0x00,0x40,0x40,0x7F,0x40,0x40,0x00,0x00,0x00,0x00,0x00,0
x00},
{0x00,0x00,0x00,0x18,0x04,0x02,0x02,0x02,0x82,0x82,0x84,0x78,0x00,0x00,0x00,
0x00,
0x00,0x00,0x00,0x78,0x44,0x42,0x41,0x41,0x40,0x40,0x40,0x70,0x00,0x00,0x00,0
x00},
{0x00,0x00,0x00,0x0C,0x02,0x02,0x02,0x82,0x82,0x42,0x22,0x1C,0x00,0x00,0x00,
0x00,
0x00,0x00,0x00,0x30,0x40,0x40,0x40,0x40,0x40,0x41,0x22,0x1C,0x00,0x00,0x00,0
x00},
{0x00,0x00,0x00,0x00,0x80,0x60,0x1C,0x02,0xFE,0x00,0x00,0x00,0x00,0x00,0x00,
0x00,
0x00,0x00,0x0C,0x0A,0x09,0x08,0x48,0x48,0x7F,0x48,0x48,0x08,0x00,0x00,0x00,
0x00},
{0x00,0x00,0x00,0xFE,0x82,0x42,0x42,0x42,0x42,0x42,0x82,0x02,0x00,0x00,0x00,
0x00,
0x00,0x00,0x00,0x31,0x40,0x40,0x40,0x40,0x40,0x40,0x20,0x1F,0x00,0x00,0x00,0
x00},
{0x00,0x00,0x00,0xF8,0x04,0x82,0x82,0x82,0x82,0x82,0x04,0x18,0x00,0x00,0x00,
0x00,
0x00,0x00,0x00,0x1F,0x21,0x40,0x40,0x40,0x40,0x40,0x21,0x1E,0x00,0x00,0x00,0
x00},
{0x00,0x00,0x00,0x0E,0x02,0x02,0x02,0x02,0x82,0x42,0x32,0x0E,0x00,0x00,0x00,
0x00,
0x00,0x00,0x00,0x00,0x00,0x00,0x70,0x0E,0x01,0x00,0x00,0x00,0x00,0x00,0x00,0
x00},
{0x00,0x00,0x00,0x38,0x44,0x82,0x82,0x82,0x82,0x82,0x44,0x38,0x00,0x00,0x00,
0x00,
0x00,0x00,0x00,0x1E,0x21,0x40,0x40,0x40,0x40,0x40,0x21,0x1E,0x00,0x00,0x00,0
x00},
{0x00,0x00,0x00,0x78,0x84,0x02,0x02,0x02,0x02,0x02,0x84,0xF8,0x00,0x00,0x00,
0x00,
0x00,0x00,0x00,0x18,0x20,0x41,0x41,0x41,0x41,0x41,0x20,0x1F,0x00,0x00,0x00,0
x00}
};
unsigned    char code shiji[2][32]={
{0x10,0x0C,0x04,0x84,0x14,0x64,0x05,0x06,0xF4,0x04,0x04,0x04,0x04,0x14,0x0C,
0x00,
0x04,0x84,0x84,0x44,0x47,0x24,0x14,0x0C,0x07,0x0C,0x14,0x24,0x44,0x84,0x04,0
x00},
{0x00,0xFE,0x22,0x5A,0x86,0x00,0x20,0x22,0x22,0x22,0xE2,0x22,0x22,0x22,0x20,
```

```
0x00,
0x00,0xFF,0x04,0x08,0x07,0x10,0x0C,0x03,0x40,0x80,0x7F,0x00,0x01,0x06,0x18,0
x00}
};
unsigned    char code mubiao[2][32]={
{0x00,0x00,0xFE,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0x22,0xFE,0x00,0x00,
0x00,
0x00,0x00,0xFF,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0x42,0xFF,0x00,0x00,0
x00},
{0x10,0x10,0xD0,0xFF,0x90,0x10,0x20,0x22,0x22,0x22,0xE2,0x22,0x22,0x22,0x20,
0x00,
0x04,0x03,0x00,0xFF,0x00,0x13,0x0C,0x03,0x40,0x80,0x7F,0x00,0x01,0x06,0x18,0
x00}
};
unsigned    char code du[]=
{0x00,0x00,0xFC,0x24,0x24,0x24,0xFC,0x25,0x26,0x24,0xFC,0x24,0x24,0x24,0x0
4,0x00,
0x40,0x30,0x8F,0x80,0x84,0x4C,0x55,0x25,0x25,0x25,0x55,0x4C,0x80,0x80,0x80,0
x00};

void delay(uint i)//延时子程序,i 最大 256,超过 256 部分无效
{
     while(--i);
}
void Read_busy()    //等待 BUSY=0
{          //busy p2^7
     P2=0xff;
     RS=0;//RS/RW=0/1,读取状态字指令
     RW=1;
     EN=1;//控制 LCM 开始读取
     while(P2&0x80);//判忙，循环等待 P2.7=0.
     EN=0;//控制 LCM 读取结束
}
void write_command(uchar value)//设置地址或状态
{
    P2=0xff;
    Read_busy();//等待 LCM 空闲
     RS=0;//RS/RW=00,设置 LCM 状态或选择地址指令
     RW=0;
     P2=value;//设置
     EN=1;//控制 LCM 开始读取
     delay(100);
     EN=0;//控制 LCM 读取结束
}
```

```
void write_data(uchar value)//写数据到显示存储器
{
     P2=0xff;
     Read_busy();
      RS=1;// RS/RW=10,写数据指令
      RW=0;
      P2=value;//写数据
      EN=1;
      delay(100);
      EN=0;
}
void Set_column(uchar column)//选择列地址(Y)
{
   column=column&0x3f;//高两位清零 0,保留后六位的地址
   column=0x40|column;//01000000|column,根据后六位选择列地址
   write_command(column);
}

void Set_line(uchar startline)//显示起始行设置
{
   startline=0xC0|startline;// 11000000|startline,  根据 startline 后六位选择起始行
   write_command(startline);
}

void Set_page(uchar page)//选择页面地址(X)
{
   page=0xb8|page;//10111000|page,根据 page 后三位确定选择的页
   write_command(page);
}
void display(uchar ss,uchar page,uchar column,uchar *p)
{//ss 选择屏幕,page 选择页面,column 选择列,P 是要显示的数据数组的指针
      uchar i;
      switch(ss)
   {
     case 0: CS1=1;CS2=1;break; //全屏
      case 1: CS1=1;CS2=0;break; //左半屏
      case 2: CS1=0;CS2=1;break; //右半屏
      default: break;
   }
   page=0xb8|page;//10111000|page,根据 page 后三位确定所选择的页
   write_command(page);

   column=column&0x3f;//高两位清 0,保留后六位的列地址
   column=0x40|column;//01000000|column,根据后六位选择列地址
```

```
    write_command(column);
    for(i=0;i<16;i++)//列地址自动+1
    {
       write_data(p[i]);//写前 16 个长度数据
    }
    page++;
    write_command(page);
//    column--;
    write_command(column);
    for(i=0;i<16;i++)//列地址自动+1
    {
       write_data(p[i+16]);//写后 16 个数据长度
    }
}
void SetOnOff(uchar onoff)//显示开关设置
{
    onoff=0x3e|onoff;//00111110|onoff,根据最后一位设置开关触发器状态，从而控制显示屏的显示状态
    write_command(onoff);
}
void ClearScreen()//清屏
{
    uchar i,j;
    CS2=1;
    CS1=1;
    for(i=0;i<8;i++)
    {
       Set_page(i); //依次选择个页面
        Set_column(0);//选择第 0 列
      for(j=0;j<64;j++)//列地址具有自动加一功能，依次对页面的 64 列写入 0 从而清屏
      {
        write_data(0x00);
      }
    }
}
void InitLCD()//初始化
{
     Read_busy();
      CS1=1;CS2=1;
      SetOnOff(0);
     CS1=1;CS2=1;
      SetOnOff(1);//打开显示开关
      CS1=1;CS2=1;
```

```
    ClearScreen();//清屏
    Set_line(0);//设置显示起始行
}
bit DS_init()
{
     bit flag;
     DQ = 0;
     delay_us(255);     //500us 以上
     DQ = 1;              //释放
     delay_us(40);      //等待 16~60us
     flag = DQ;
     delay_us(150);
     return flag;       //成功返回 0
}
uchar read()     //byte
{
      uchar i;
      uchar val = 0;
      for (i=0; i<8; i++)
      {
           val >>= 1;
           DQ = 0; //拉低总线产生读信号
           delay_us(1);
           DQ = 1; //释放总线准备读信号
           delay_us(1);
           if (DQ) val |= 0x80;
           delay_us(15);
     }
 return val;
}
void write(char val)     //byte
{
 uchar i;

 for (i=0; i<8; i++)
 {
 DQ = 0; //拉低总线产生写信号
 delay_us(8);
 val >>= 1;
 DQ = CY;
 delay_us(35);
 DQ = 1;
 delay_us(10);
 }
```

```
}
void PID()
{
    uchar Px,Pp,Pi,Pd,a,b,c;
    uint count;
    Pp = Kp*(Ek-Ek1);
    Pi = Ki*Ek;
    Pd = Kd*(Ek-2*Ek1+Ek2);
    Px = Pp+Pi+Pd;
    res = Px;
        a=res/100;
        b=res%100/10;
        c=res%10;

        display(1,4,2*16,shu[a]);delay(255);
        display(1,4,3*16,shu[b]);delay(255);
        display(2,4,0*16,shu[c]);delay(255);
    Ek2 = Ek1;
    Ek1 = Ek;
    count = 0;
    if(res>Pmax)
        res =Pmax ;
    while((count++)<=res)
    {
        up = 1;
        delay_us(250);
        delay_us(250);
    }
    while((count++)<=Pmax)
    {
        up = 0;
        delay_us(250);
        delay_us(250);
    }
}
void main()
{
    uchar aim,low,high,b,c;
    uint result;
    InitLCD();
    Set_line(0);
    aim = 40;
    Kp = 4;
    Ki = 5;
```

```
Kd = 2;
Pmax = 5;
Ek1 = 0;
Ek2 = 0;
res = 0;

while(1)
{
    if(s1 == 0)
        aim++;
    if(s2 == 0)
        aim--;
    while(DS_init());
    write(0xcc);   //跳过 ROM 命令
    write(0x44);   //温度转换命令
    delay(600);
    while(DS_init());
    write(0xcc);
   write(0xBE);      //读 DS  温度暂存器命令
    low = read(); //采集温度
    high = read();
    delay(255);
    result = high;
    result <<= 8;
    result |= low;
    result >>= 4 ; //result /= 16;

    Ek = aim - result;

    b=result/10;
    c=result%10;

    display(1,0,0*16,shiji[0]);delay(255);
    display(1,0,1*16,shiji[1]);delay(255);
    display(1,0,3*16,shu[b]);delay(255);
    display(2,0,0*16,shu[c]);delay(255);
    display(2,0,1*16,du);delay(100);

    b=aim/10;
    c=aim%10;

    display(1,2,0*16,mubiao[0]);delay(255);
    display(1,2,1*16,mubiao[1]);delay(255);
    display(1,2,3*16,shu[b]);delay(255);
```

```
            display(2,2,0*16,shu[c]);delay(255);
            display(2,2,1*16,du);delay(100);
            if(aim>=result)
                PID();
            else
                up = 0;
        }
}
```

六、思考题

1. 进行精确的延时的程序有几种方法？各有什么优缺点？

答：（1）定时器延时：通过设置定时器处置可以实现以 us 为单位的精确定时。

优点：定时精确，程序移植性好；复用性好。

缺点：设置定时器本身需要一定的时间，要求延时较短的情况下不满足要求，实现复杂。时间有上限，且浪费计时器

（2）软件定时：使用 while 循环，每执行一次循环需要 3-5us。

优点：实现简单

缺点：不精确，不同机器的机器时钟不同，软件中断的时间就有所不同。