电子科技大学电子科学与工程学院

实 验 报 告

实验名称 现代电子信息系统综合实验

姓名：张前锋

学号：2020020910019

评分：

教师签字

电子科技大学教务处制

电 子 科 技 大 学

实 验 报 告

学生姓名：张前锋　　学号：2020020910019　　指导教师： 王军

实验地点：综合实验大楼 335　　实验时间：2022 年 11 月 22 日

一、实验室名称： 电子技术综合实验室

二、实验项目名称：小车快速精确定位设计与实现

三、实验学时： 40

四、实验目的与任务：

1、熟悉系统设计与实现原理

2、掌握 KEIL C51 的基本使用方法

3、熟悉智能小车实验平台应用

4、独立编程调试，测试和掌握各部分的功能

5、完成系统软件的编写与调试

6、完成设计报告

五、实验器材

1、PC 机一台

2、智能小车实验平台一套

六、实验原理、步骤及内容

（一）试验要求

基本要求：80 分

（1）小车开机运行程序，在 8 位数码管的最右边 3 位显示小车定位距离，初始值为 12.5（单位：cm）并启动超声波测距，将距离值显示

在最左边 4 位(xxx.x cm) ；

(2)利用按键设置定位距离,“+”按键每次增加 0.5cm,上限为 15.0cm；“-”按键每次减少 0.5cm，下限为 10.0cm；当按下该按键时，蜂鸣器响 0.1 秒（按键提示音)。

（3）设定好定位距离的小车放置在障碍物 1 米以外的位置。利用光敏遥控启动小车，同时启动“秒表计时器” 作为小车运行时间计时，并在数码管最右边 3 位显示时间（要求定时中断实现)；尽量保持小车直线前进，要求小车速度至少有两个速度档位，距离障碍物越近，速度越慢。小车第一次进入定位距离范围内，停止计时，要求该时间不大于 3.2 秒，并记录小车运行时间。

（4）小车运行过程中，数码管上始终实时显示运行时间和小车到障碍物的距离；

（5）小车在距离障碍物为定位距离±0.5cm 范围内停止行驶，通过速度调节和前进后退等方式使小车精确定位在目标范围，若小车位于（定位距离-0.5cm）以内 ，则声光报警，即用一个发光二极管指示灯闪烁，点亮 0.1s，熄灭 0.3s；用蜂鸣器响 0.1s，静音 0.3s 报警；若大于等于（定位距离-0.5cm） ，则撤销声光报警。小车不能碰撞前方障碍物!

（6）采用滑动平均值滤波提高测距稳定性，建议每测量三个结果取一次均值。

扩展要求：20 分

（1）秒表停止计时后，闪烁显示最终计时时间（点亮 0.1s+熄灭 0.3s）

（2）为避免测距不稳定导致的小车频繁运动，进入设定距离后，启动判断机制，当小车连续三次检测到实测距离符合前进或后退的要求才运动。

（二）实验内容

1、 系统硬件构成

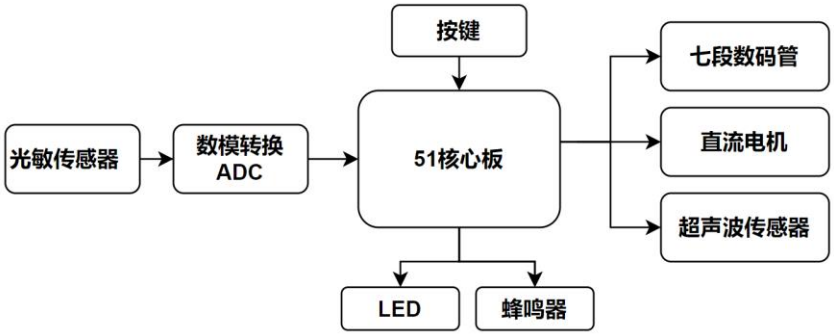
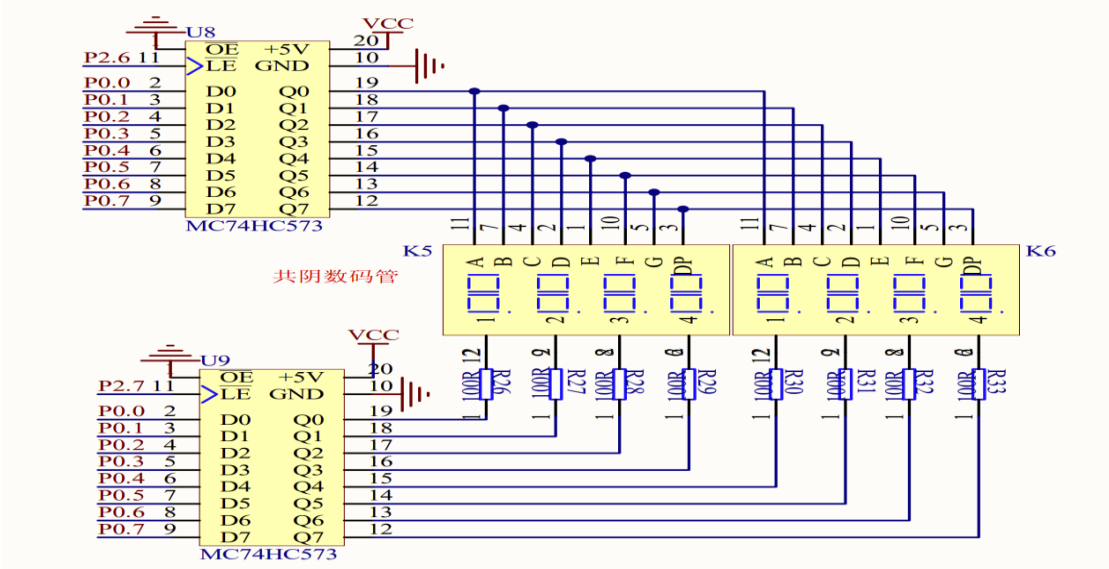


图 1 系统硬件构成框图

系统主要包括，STC89C52 单片机核心板，主要功能为处理软件程序逻辑；按键模块包含两个独立按键，主要用来调整小车初始定位距离；光敏传感器模块和 ADC 模块，主要用来启动小车和秒表计时器，遮光时将光强转换为模拟信号再转换为数字信号，达到设置阈值的时启动相应控制模块；LED 和蜂鸣器模块用来再小车位于小于定位范围时报警；七段数码管用来显示初始定位距离，启动后显示秒表计时器数值超声波定位距离；超声波传感器用来测量小车与障碍物之间的距离，根据回响信号脉宽得到传播时间，进而计算出距离；直流电机模块用于驱动小车运动，根据 pwm 信号的有效值来调整小车速度。

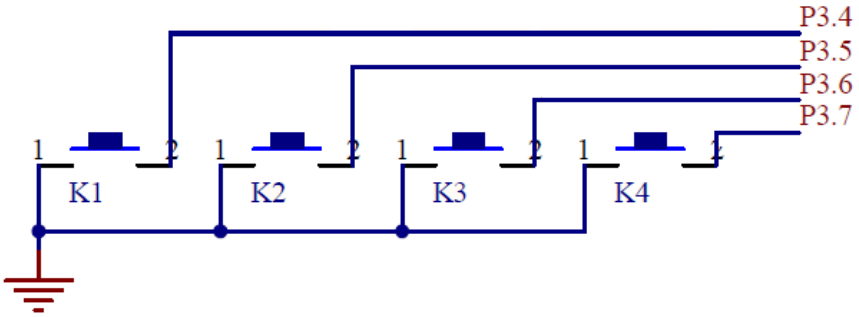
2、 主要模块硬件原理

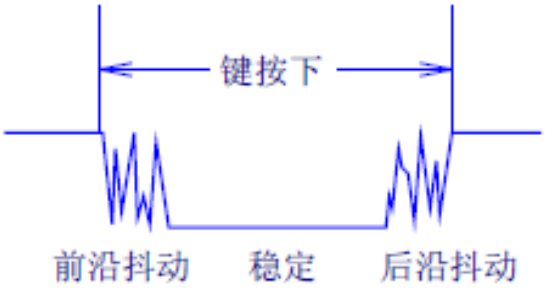
2.1 数码管动态扫描



系统原理图如上，使用锁存器来节省 IO 口资源，只占用 P0 口。使用 code 类型来定义数码管字型数据存储在代码存储器 ROM 中，定义显示缓存区保存每个数码管将要显示的数据；消隐后，先进行段选，将显示缓存区数据送入锁存器，再进行片选，将位数据送入锁存器，就会显示相应数据。当刷新频率达到一定程度时，人眼无法分辨从而实现数码管动态显示， 频率太慢会导致人眼可以分辨，频率过快导致低速器件数码管系统功耗增加，显示效果变暗,1KHz 左右最好，当数码管数量过多时，可以将数码管分组实现动态显示。将数码管程序写在 1ms 定时中断程序中后，只需要改变显示缓存区数字就可以方便实现动态显示。

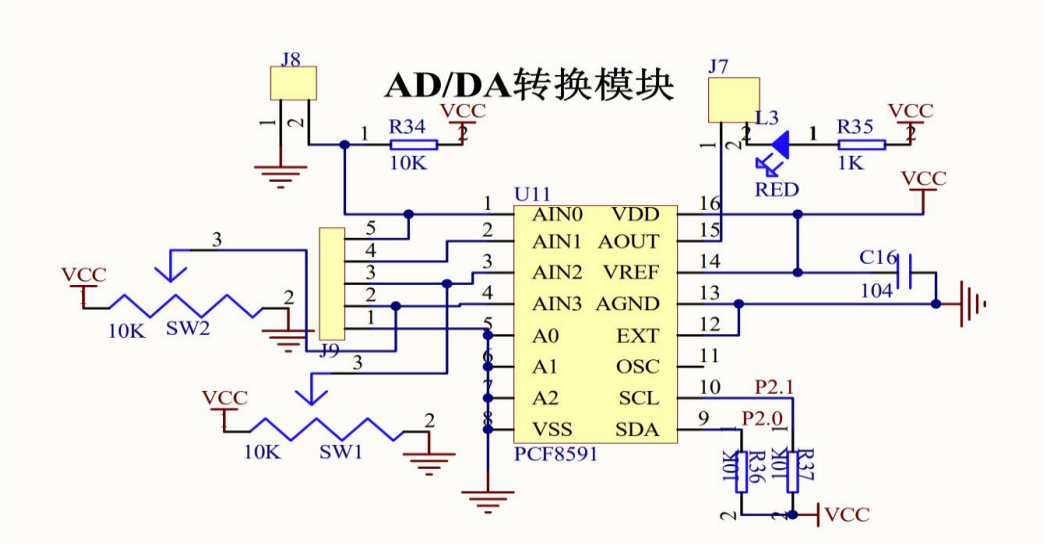
2.2 按键扫描及去抖原理





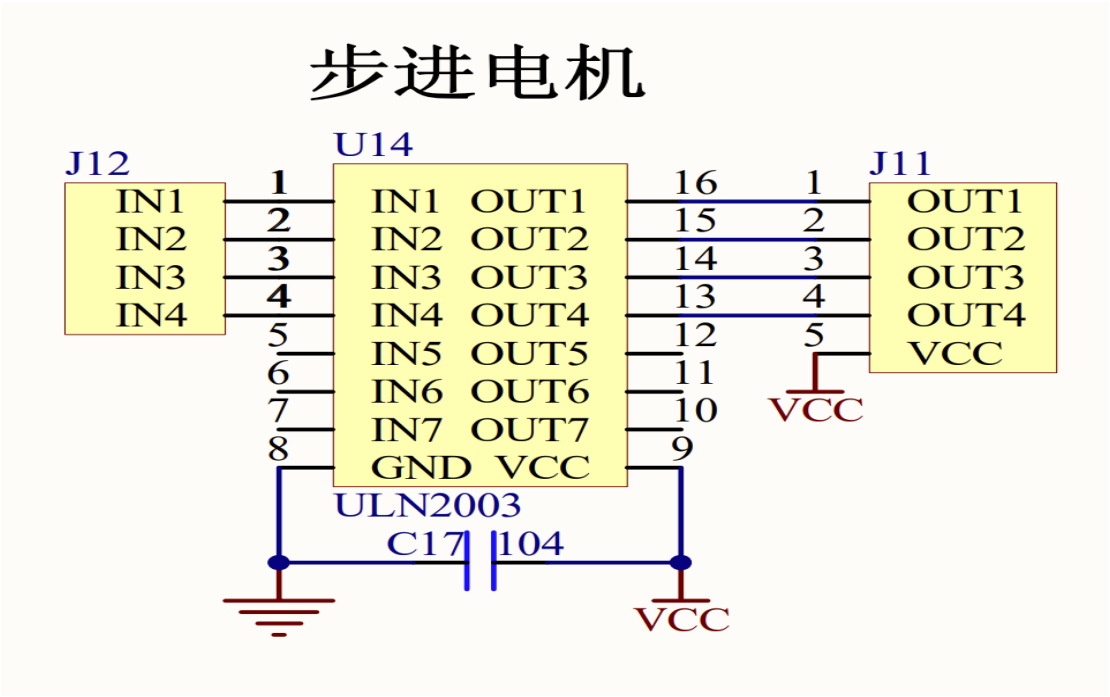
本次实验只涉及四个独立按键的使用，如图，当按键按下时，会出现前沿抖动；按键抬起时会出现后沿抖动。在判断按键按下次数时，一般会定义标志变量，不进行消抖按下一次会导致变量在稳定器件发生很多次变化，无法确定按键是否按下，所以要进行按键消抖。按键识别可以通过外部输入中断方式和轮询方式，51 单片机资源有限，所以本实验使用轮询方式并进行软件消抖。当检测到按键首次按下时延时 20ms 左右进行前沿消抖，再一次判断按键是否按下，如果按下则进行了一次有效的按键操作，之后等待按键释放 while（!key）；之后再延时 20ms 左右进行后沿消抖。除了软件消抖之外，还有使用寄存器的硬件消抖方式。

2.3 模数转换 ADC 采样光敏传感器原理

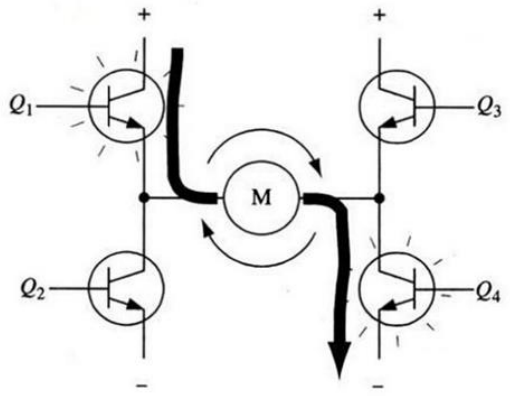


模数转换器（ADC）,A/D 可以将连续的模拟信号转换为时间离散、幅值也离散的数字信号，将模拟电压成正比的转换成对应的数字量，一般包括取样、保持、量化及编码 4 个过程，通过内部的标准参考电压，触发器、编码器和电压比较器等数模混合电路将电压进行量化，最终输出数字量。分辨率是衡量 A/D 转换器对输入信号的分辨能力，n 位则可以区分 2^n 个不同等级的输入模拟电压，精度为 1/2^n。转换时间指从模拟信号到来到得到稳定的数字信号所使用的时间，速度越快性能越好。本次实验使用 PCF8591 数模转换芯片，具有 8 位 D/A、A/D 转换功能，通过 I2C 总线协议与单片机通信。四个模拟输入通道 AIN0-AIN3，一个模拟输出通道 AOUT。光敏传感器电压信号输入 D/A,得到输出数字量再送入 A/D 控制 led，当数字量达到设定的阈值时，启动小车运动。

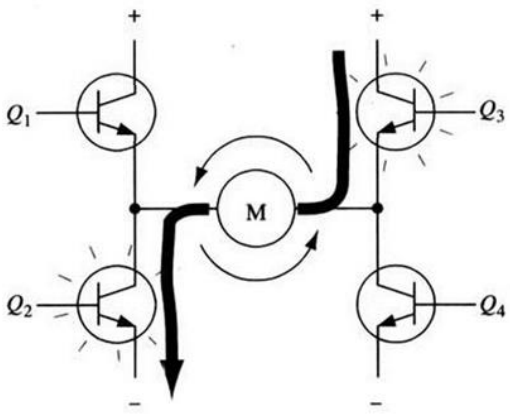
2.4 电机驱动及 PWM 调速的原理



单片机的 IO 无法提供直流电机所需的电流，所以需要使用电机驱动电路，为直流电机提供足够大的驱动电流。本次实验使用的是 L293D 电机驱动电路，是具有高电压大电流的全桥驱动芯片，响应频率高，一片 L293D 可以分别控制两个直流电机，具有使能端，利用 PWM 方便的实现调速控制。

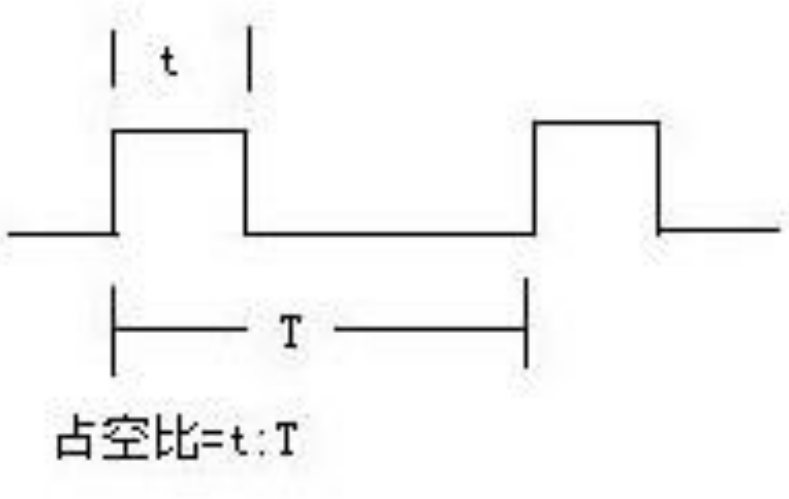


当 Q1 管和 Q4 管导通时，电流按图中箭头所示，电流将从左至右流过电机，从而驱动电机按特定方向转动。



当 Q2 和 Q3 导通时，电流按图中箭头所示，电流将从右至左流过电机，从而驱动电机沿另一方向转动。

直流电机等效为大电感，具有阻碍输入电流和电压突变的能力，因此脉冲输入信号被平均分配到作用时间上，通过改变输入方波的占空比就能改变加在电机两端的平均直流电压大小，从而改变了电机转速。



PWM 即脉冲宽度调制，可以通过软件方式对模拟信号电平进行数值编码，调制方波的占空比，得到不同电压有效值的方波信号，从而控制直流电机的速度。

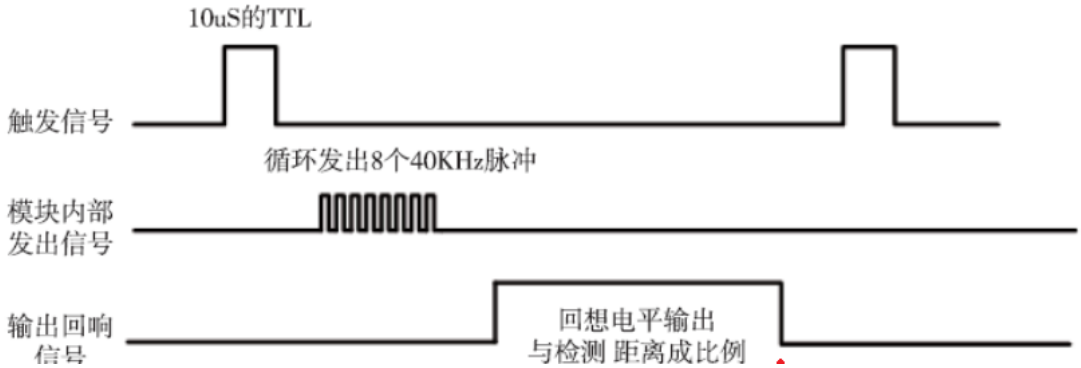
2.5 超声波测距与定位原理

本次实验使用的是超声波模块 HC-SR04,由两个压电晶片和一个共振板组成,压电晶片可以发送和接收超声波,总共有四个引脚 VCC、GND、Trig 和 Echo。超声波模块采用回声探测法，当超声波和障碍物之间的角度可以忽略时，计算公式简化为

$$H=vt/2$$

其中 v 为超声波在实验温度下的传播速度。所以，我们只需要测量出超声波传播所用事件 t 即可。

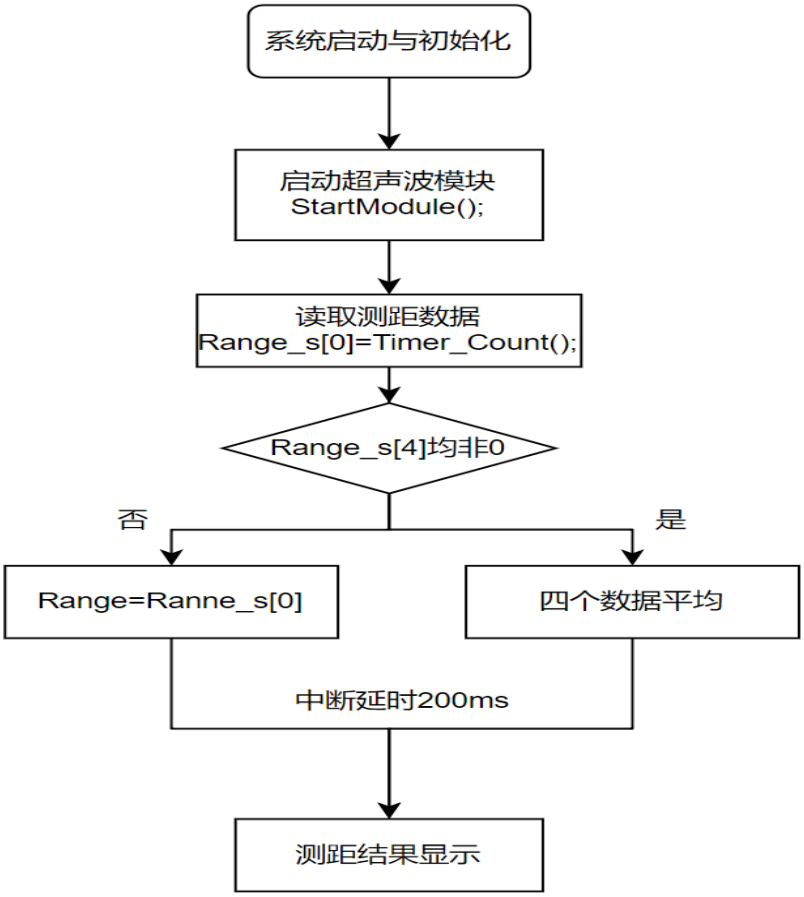
提供一个 10us 以上的触发信号来启动超声波测量，模块内部将发送 8 个 40KHz 电平并检测回波，检测到回波并输出回响信号，回响信号的脉冲宽度与所测的距离成正比。所以只需要测得回响型号的宽度。



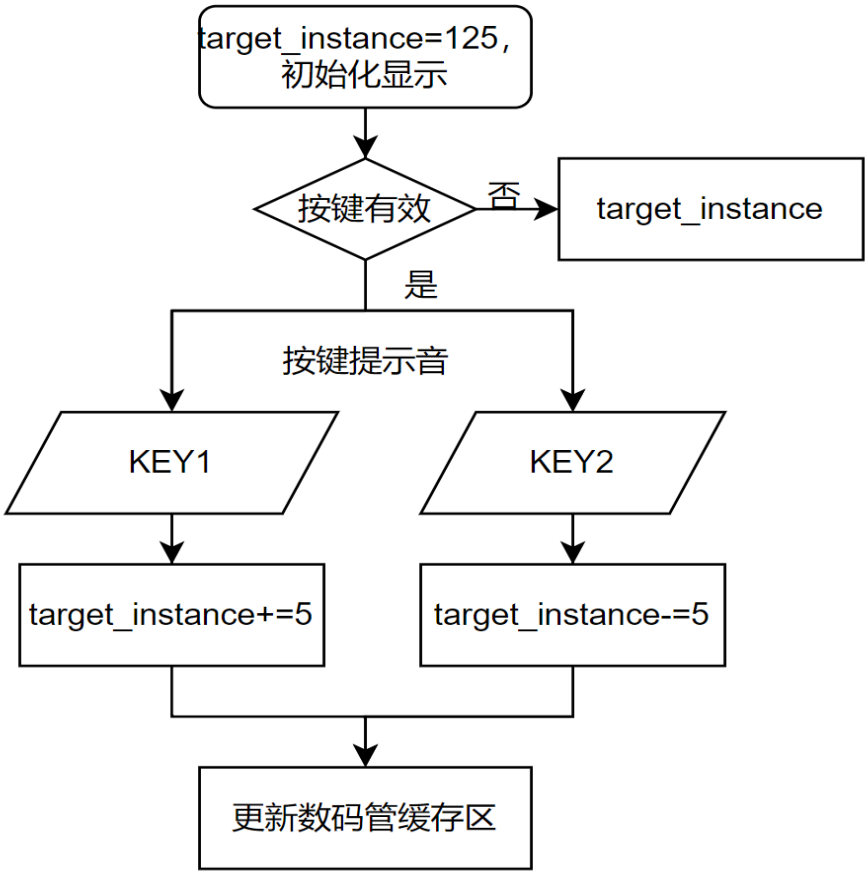
为避免轮询方式影响 cpu 效率和计时的准确性，本次实验采用外部中断方式测量回响信号，当触发信号发出后，while(!Echo)等待输出回响信号，然后启动计时和外部中断，等待回响结束触发外部中断，得到计时时间，利用公式计算出测量距离。

3、 软件设计思路(重点)

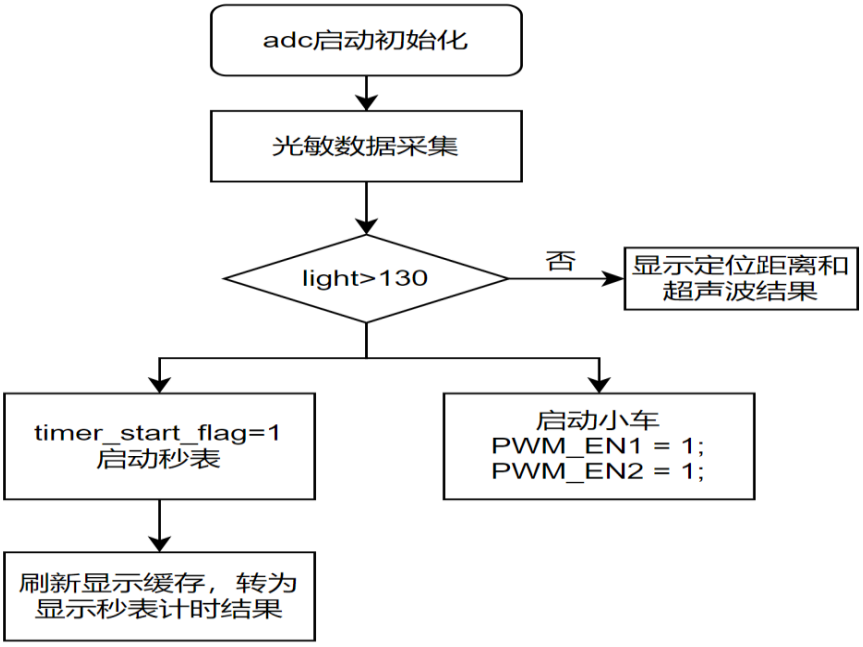
3.1 超声波模块程序框图



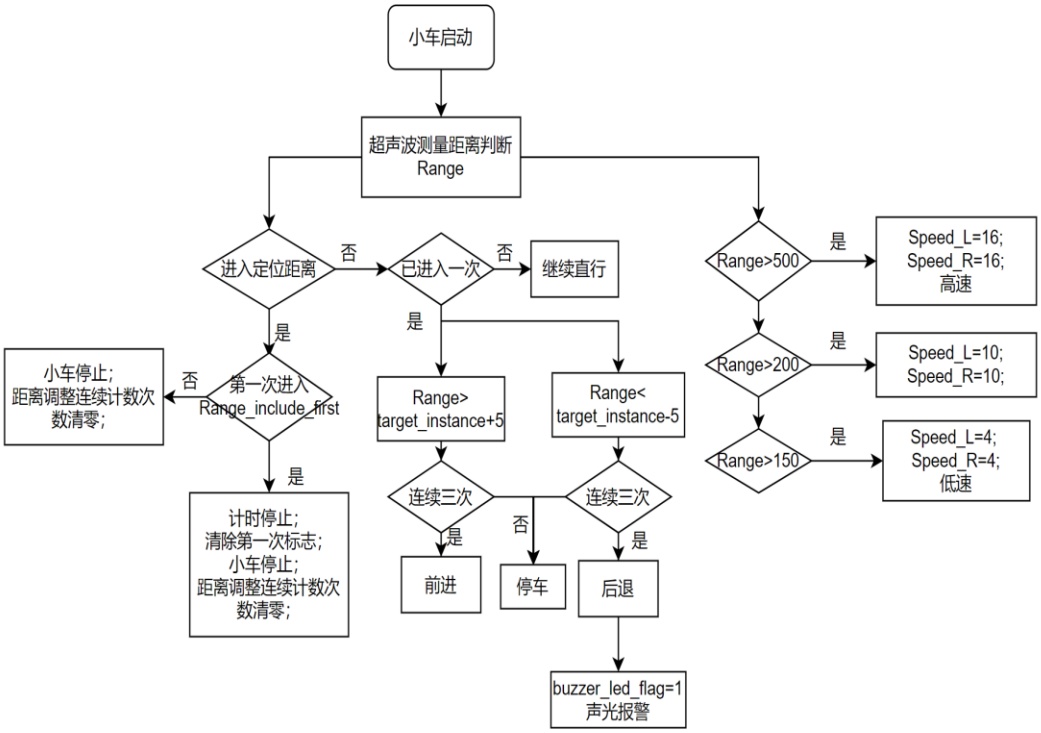
3.2 定位距离设置程序



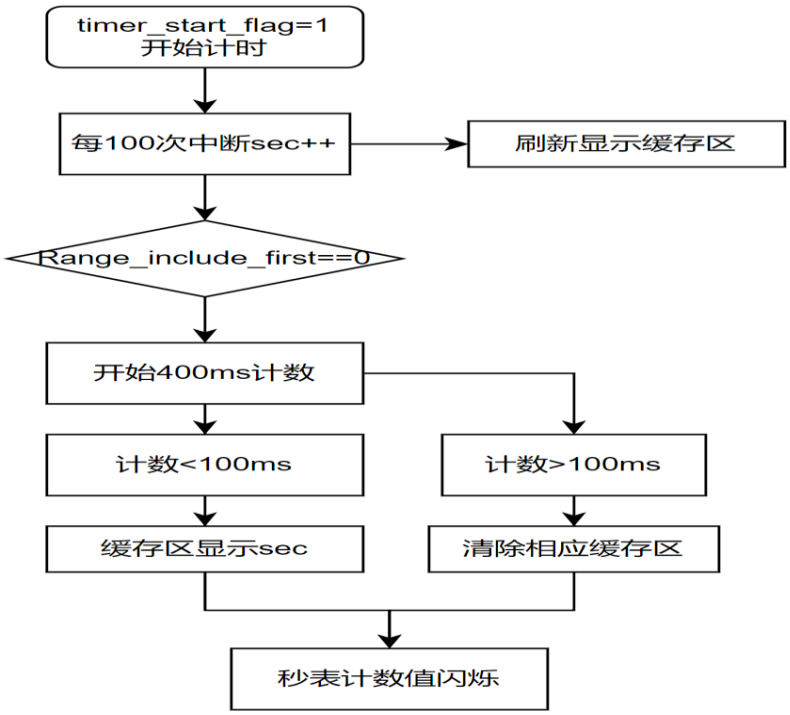
3.3adc 遮光启动程序框图



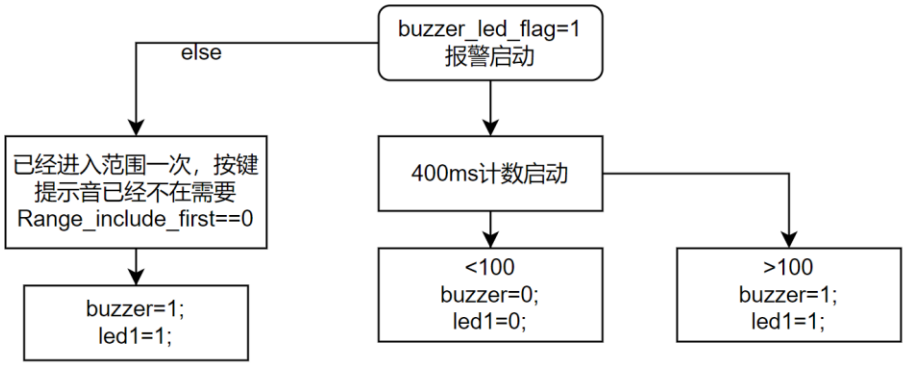
3.4 小车定位模块程序框图



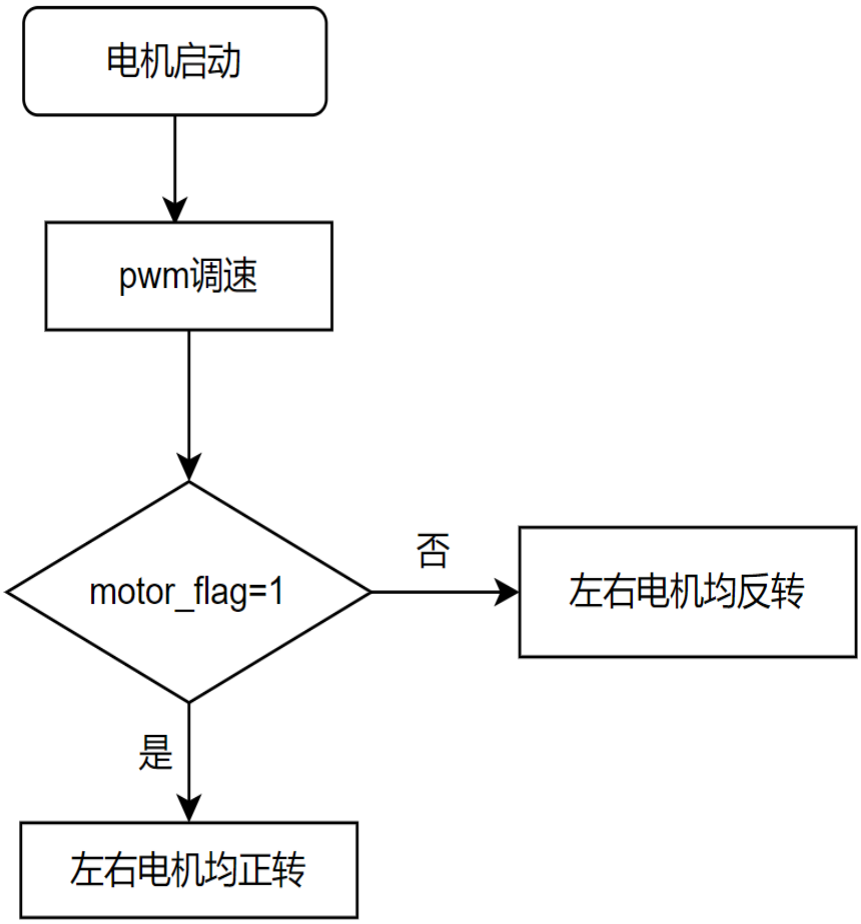
3.5 秒表计数器模块程序框图



3.6 声光报警模块程序框图



3.7pwm 模块程序框图



七、总结及心得体会

通过本次实验，对现代电子系统综合实验中涉及到的 51 单片机和超声波等基本模块进行综合使用，掌握了基本模块的原理和使用方法、简单系统的开发方式，初步掌握了嵌入式软件开发的基本流程以及代码书写规范，为进一步学习其他嵌入式芯片奠定了坚实的基础。

八、对本实验过程及方法、手段的改进建议

（没有可以不写）

九、附录

```
#include <reg52.h>
#include "PCF8591.h"
//定义驱动引脚
sbit Echo = P3^2;   //Echo
sbit Trig = P3^3;    //Trig

#define LED_seg   P0              //8 位数码管的段码和位码驱动通过 P0 端口锁存
#define LED_dig   P0

//定义 I/O 接口
sbit PWM_IN1 = P1^4;       // 高电平 1：左电机后退（反转）
sbit PWM_IN2 = P1^5;       // 高电平 1：左电机前进（正转）

sbit PWM_IN3 = P1^6;       // 高电平 1：右电机前进（正转）
sbit PWM_IN4 = P1^7;       // 高电平 1：右电机后退（反转）

sbit PWM_EN1 = P1^2;       // 高电平 1：使能左电机
sbit PWM_EN2 = P1^3;       // 高电平 1：使能右电机
//定义 PWM 最大级数，也就是调节直流电机的速度等级
#define SPEED_MAX         20
//定义 PWM 级数，分为 0～SPEED_MAX-1 级
unsigned char Speed_L;       //左电机转速调节（调节 PWM 的一个周期 SPEED_MAX*1ms 时间内，左电机正转时间：Speed_L*1ms）
unsigned char Speed_R;       //右电机转速调节（调节 PWM 的一个周期 SPEED_MAX*1ms 时间内，右电机正转时间：Speed_R*1ms）



//定义显示缓冲区（由定时中断程序自动扫描）
unsigned char DispBuf[8];
//=============================================
```

```
//===================自定义变量=====================
bit display_flag=0;   //测量距离显示标志，200ms 显示一次
unsigned int Range_s[4]={0,0,0,0};//定义滑动平均值的数据缓存区
unsigned int target_instance=125; //定义小车定位距离初始值 12.5cm=125mm

//智能小车数码管按键电路管脚定义
sbit KEY1 = P3^4;        //定义按键 1，'+'按键，对应核心板上 K1
sbit KEY2 = P3^5;        //定义按键 2，'-'按键，对应核心板上 K2

sbit led1=P1^0;
sbit buzzer=P2^3; //启动前按键提示，启动后用于声光报警


unsigned char sec = 0;          //秒表计时器，精度 0.1s
bit timer_start_flag = 0;  //定义标志位,秒表启动标志
bit motor_flag=1;//定义电机正反转标志，1 正，0 反转

bit Range_include_first=1;//0 表示第一次进入，且秒表已经暂停

bit buzzer_led_flag=0;//声光报警标志，1 报警

unsigned char detector3_a=0;//连续三次检测前进计数
unsigned char detector3_b=0;//连续三次检测后退计数




//================================================
//智能小车数码管显示电路管脚定义
sbit SS = P2^6;          //数码管段选信号
sbit CS = P2^7;          //数码管位选信号

bit Counter_overflag = 0;          //T0 定时器溢出标志
bit Echo_Over = 0;                      //超声波测距完成标志，无论收到回波或没有收到，
总要置位一次
unsigned int Range = 0;
unsigned long Echo_time = 0;          //T0 定时器合并数值

code unsigned char Tab[] =
    {//定义 0123456789AbCdEF 的数码管字型数据，其他显示字符需自行计算，如‘-’的
字形数据为 0x40
        0x3F,0x06,0x5B,0x4F,0x66,0x6D,0x7D,0x07,
        0x7F,0x6F,0x77,0x7C,0x39,0x5E,0x79,0x71
```

```
    };
//============================//
/*
函数：EX0INTSVC()
功能：外部中断 0 中断服务程序
用途：为避免定时中断干扰测距的准确性，采用外部中断触发后立即停止计时
          外部中断 0 优先级高于定时中断，程序尽量短，避免过多干扰定时中断
*/
void EX0INTSVC() interrupt 0
{
    TR0 = 0;                        //停止计时
    Echo_time = TH0 * 256 + TL0;    //读取定时器 0 计数值
    TH0 = 0;                        //清除定时器 0 计数寄存器，为下次计数做准备
    TL0 = 0;
    Echo_Over = 1;          //表示本次超声波测距完成，可以启动下次的测量
    EX0 = 0;                        //关闭外部中断，否则会马上引起下一次中断
}
//============================//
/*
函数：T0INTSVC()
功能：定时器 T0 的中断服务函数
用途：若超过测距范围，长时间无法收到回波，
          已经启动的 T0 中断会计数溢出，利用定时器溢出标志来判断
*/
void T0INTSVC() interrupt 1
{
    TR0 = 0;                    //停止计时
    Counter_overflag = 1;               //中断溢出标志，未收到回波
    Echo_Over = 1;     //表示本次超声波测距完成，可以启动下次的测量
    EX0 = 0;                    //关闭外部中断
}
/*
函数：T1INTSVC()
功能：定时器 T1 的中断服务函数
*/

void T1INTSVC() interrupt 3 //定时器 1 的中断号为：3
{
    code unsigned char com[] = {0x80,0x40,0x20,0x10,0x08,0x04,0x02,0x01};     // 显 示 位 的
端口控制字节
    static unsigned char n = 0;                                 //n: 扫描显示位计数,0-7

    static unsigned int counter_200ms=0;//200ms 计数用于显示
    static unsigned int counter_1ms = 0;     //1ms 计数用于秒表
```

```
    static unsigned char t = 0;    //pwm 调速计数变量
    static unsigned int counter_400ms=0;//用于秒表闪烁显示
    static unsigned int counter1_1ms = 0;   //1ms 计数用于声光报警
//================数码管定时扫描驱动显示==================
    TR1 = 0;
    TH1 = 0xFC;
    TL1 = 0x66;                             //可以将 FC66 换成 0000，降低扫描速度，观察和
理解动态扫描
    TR1 = 1;
    P0 = 0xFF;          //消隐
    CS = 1;
    CS = 0;
    P0 = DispBuf[n];    //更新扫描显示数据
    SS = 1;
    SS = 0;
    P0 = ~com[n]; //重新显示
    CS = 1;
    CS = 0;
    n++;                            //指向下一位扫描显示
    n &= 0x07;

//==============每 200ms 显示测量距离一次==================
    if(counter_200ms==199){
        counter_200ms=0;
        display_flag=1;
    }else{
        counter_200ms++;
    }
//=========================================================


//==================秒表精度 0.1s====================
    if(timer_start_flag)
    {
        counter_1ms ++;    //1ms 加 1

        if(counter_1ms == 100)
        {
            counter_1ms = 0;            //0.1s 到
            sec++;
            DispBuf[0] = Tab[sec % 10];             //查表取出显示数字对应的段码，存入显
示缓冲器数组
            DispBuf[1] = Tab[sec / 10 % 10] | 0x80;      // "| 0x80"可以使该位带小数点显示
            DispBuf[2] = Tab[sec / 100];
```

```
        }
    }
//================================================

    //======================pwm 调速=====================
    t++;
    if ( t >= SPEED_MAX )
        t = 0;                      //PWM 波的周期为：SPEED_MAX*1ms = 20ms

    if ( t < Speed_L )          //PWM 波高电平时间：(Speed_L)*1ms
    {
        if(motor_flag){
            PWM_IN1 = 0;
            PWM_IN2 = 1;            //左电机的正转
        }else{
            PWM_IN2 = 0;
            PWM_IN1 = 1;            //左电机的反转
        }
    }
    else                        //PWM 波低电平时间：（SPEED_MAX-Speed_L）*1ms
    {
    PWM_IN1 = 0;
        PWM_IN2 = 0;            //左电机的停转
    }

    if ( t < Speed_R )          //PWM 波高电平时间：Speed_R*1ms
    {
        if(motor_flag){
            PWM_IN4 = 0;            //右电机的正转
            PWM_IN3 = 1;
        }else{
            PWM_IN3 = 0;
            PWM_IN4 = 1;        //右电机的反转
        }
    }
    else                        //PWM 波低电平时间：（SPEED_MAX-Speed_R）*1ms
    {
        PWM_IN3 = 0;
        PWM_IN4 = 0;            //右电机的停转
    }
    //================================================

    //========秒表停止后计时闪烁（亮 0.1+暗 0.3s）==============
        if(Range_include_first==0){
```

```
            if(counter_400ms==400){
                counter_400ms=0;
            }else{
                counter_400ms++;
            }

            if(counter_400ms<100){
                DispBuf[0] = Tab[sec % 10];              //查表取出显示数字对应的段码，存入显示缓冲器数组
                DispBuf[1] = Tab[sec / 10 % 10] | 0x80;      //  "| 0x80"可以使该位带小数点显示
                DispBuf[2] = Tab[sec / 100];
            }else{
                DispBuf[0] =0x00;
                DispBuf[1] =0x00;
                DispBuf[2] =0x00;
            }
        }
    //=========================================================

    //=============声光报警模块=======================
            if(buzzer_led_flag){
                counter1_1ms++;//报警计时++
                if(counter1_1ms==400)
                    counter1_1ms=0;
                if(counter1_1ms<=100){
                    buzzer=0;
                    led1=0;
                }else{
                    buzzer=1;
                    led1=1;
                }
            }else if(Range_include_first==0){
                //直接 else 可能会导致按键提示音无效
                //判断进入过一次范围，说明已经启动了不用再用按键
                buzzer=1;
                    led1=1;
            }
    //================================================
}


/*
函数：DispClear()
```

```
功能：清除数码管的所有显示
*/
void DispClear()
{
    unsigned char i;
    for ( i=0; i<8; i++ )
    {
        DispBuf[i] = 0x00; //i 值代表数码管的位数，可以在后面的程序观察是左起还是右起，0x00 可以关闭数码管显示
    }
}

/*
函数：Delay()
功能：延时
说明：
    晶振频率为 11.0592MHz
    延时长度 = 1ms * t
*/
void Delay(unsigned int t)
{
    unsigned int us_ct;
    for (;t > 0;t --)       //执行代码消耗 CPU 时间
        for (us_ct = 113;us_ct > 0;us_ct --);
}


/*
函数：SysInit()
功能：系统初始化
*/
void SysInit()
{

    DispClear();  //初始化显示缓存
    TMOD = 0x11;    //设置定时器 T0 为 16 位定时器，定时器 T1 为 16 位定时器
    EA = 0;                 //关闭总中断，待初始化结束后再打开
//======定时计数器 T0 初始化，用于获取超声波作用时间，若定时溢出，则超出测距范围
    TH0 = 0;
    TL0 = 0;
    ET0 = 1;
//======定时计数器 T1 初始化，用于获取 1ms 定时中断=====
    TH1 = 0xFC;       //设置定时器 1 的初值: 0xFC66，对应定时时间 1ms
    TL1 = 0x66;
```

```
    ET1 = 1;            //使能定时器 T1 中断
    TR1 = 1;            //启动定时器 T1
//======定时计数器 T0 和 T1 初始化完毕=================
    EX0 = 0;            //关闭外部中断
    IT0 = 0;            //外部中断 0 采用电平触发模式，低电平出发
    EA = 1;                 //使能总中断

    PWM_EN1 = 0;
    PWM_EN2 = 0;        //电机无效
    PWM_IN1 = 0;
    PWM_IN2 = 0;        //左电机的停转
    PWM_IN3 = 0;
    PWM_IN4 = 0;        //右电机的停转

    DispClear();

}

/*
函数：StartModule()
功能：启动模块，采用 IO 触发测距，给至少 10us 的高电平信号;
*/
void   StartModule()
{
    unsigned char i;
    Trig = 1;                                   //启动一次模块
    for(i = 0;i < 10;i ++);                       //超声波启动延迟 10us 以上;
    Trig = 0;
}
/*
函数：Range_Display()
功能：超声波距离显示函数
说明：若超出距离则显示“- - - -”
*/
 void Range_Display()
{
    //超出测量范围或者障碍物太近反射角度太小导致无法收到回波，都显示“- - - -”
    if((Range >= 4000) || Counter_overflag == 1)
    {
        Counter_overflag = 0;
        DispBuf[4] = 0x40;
        DispBuf[5] = 0x40;
        DispBuf[6] = 0x40;
        DispBuf[7] = 0x40;
```

```
    }
    //按照 HC-SR04 的指标，大致工作在 2cm—450cm 的范围内，与产品质量和反射面相关
    else        //显示数据单位：厘米
    {
        DispBuf[4] = Tab[Range % 10];
        DispBuf[5] = Tab[Range / 10 % 10] + 0x80;
        DispBuf[6] = Tab[Range / 100 % 10];
        DispBuf[7] = Tab[Range / 1000];
    }
}
/*
函数：Timer_Count()
功能：超声波高电平脉冲宽度计算函数
备注：采用查询模式
说明：超声波模块在等待回波的时候，经常被定时中断打断，导致回波到达时间测量不及时，
干扰了超声波测量精度
改进的办法是分别加入下面两条语句（暂时屏蔽，取消屏蔽可以用来对照）
    TR1 = 0;        //暂停定时器 T1 计数，相当于关闭定时中断 T1
  TR1 = 1;          //重启定时器 T1 计数，相当于打开定时中断 T1
随之带来什么新的问题呢？分析产生的原因。
*/
unsigned int    Timer_Count()
{

    unsigned long Range;
    TR0 = 1;            //开启计数
    EX0 = 1;              //开启外部中断
    while(!Echo_Over);      //等待回波，当 Echo_Over 为 1，表明收到回波或超出测距范围

    Echo_Over = 0;                  //清除 Echo_Over，准备下一次测距
    //程序到这里就已经得到了超声波的响应计数值，结果存在变量 Echo_time 内，Echo_time
* 1.1us 得到响应时间
    //假设环境温度 26.5 摄氏度，根据 Echo_time 的值自行计算测距的长度（单位：毫米）
并替换下面的定值表达式
    //注意：必须用定点运算，精度为毫米！
    Range =Echo_time*11*174/10000;        //单位是毫米

    return (unsigned int)Range;
}

/*
函数：KeyScan()
功能：键盘扫描
返回：扫描到的键值
```

```
*/
unsigned char KeyScan()
{
	unsigned char k = '\0';

	if ( KEY1 == 0 ) k = '+';
	if ( KEY2 == 0 ) k = '-';

	return k;
}



void main()
{
	unsigned char k;			//定义键值变量
	unsigned char i;//缓存区滑动循环变量

	//========adc 启动必须条件========
	unsigned char light;//定义 AD 测量值
	RST_DS = 0; //关时钟 DS1302
	I2C_Init();
  //================

	//电机和定时器初始化
	SysInit();
	//====开机显示初始定位距离==
		DispBuf[0] = Tab[target_instance % 10];
		DispBuf[1] = Tab[target_instance / 10 % 10] + 0x80;
		DispBuf[2] = Tab[target_instance / 100 % 10];
	//=========================
	while(1)
	{
		Echo = 1;				//IO 口读入之前输出 1
		TR1=0;
		ET1=0;
		StartModule();	//启动模块
		while(!Echo);
		ET1=1;
		TR1=1;

		//====================滑动均值====================
		for(i=0;i<3;i=i+1){ //滑动
			Range_s[i+1]=Range_s[i];
```

```
}
Range_s[0] = Timer_Count();       //超声波高电平脉冲宽度计算函数

if(Range_s[0]&&Range_s[1]&&Range_s[2]&&Range_s[3]){//都不为 0 说明进行了三次有效测量
    Range=(Range_s[0]+Range_s[1]+Range_s[2]+Range_s[3]+2)/4;//计算滑动均值
}else{
    Range=Range_s[0];//否则采用最新一次测量结果;
}
//=====================================================



//距离有关处

//========根据距离调节速度 3 档(需上车调试)=========
    if(Range>500){//50cm
        //较远高速行驶，调节两者以保持直线
        Speed_L=16;
        Speed_R=16;
    }else if(Range>200){//20cm
        Speed_L=10;
        Speed_R=10;
    }else if(Range>150){//15cm
        Speed_L=4;   //低速模式最好与下方一致
        Speed_R=4;
    }
//=================================================

//=============进入定位距离====================
if(Range<=target_instance+5&&Range>=target_instance-5){
    if(Range_include_first){//是否第一次进入
        timer_start_flag=0;//计时关闭
        Range_include_first=0;//闪烁显示标志
    }
        Speed_L=0;//小车停止
        Speed_R=0;

        detector3_a=0;//检测计数清零，保证检测三次的连续性
        detector3_b=0;
}
if(Range_include_first==0&&(Range>target_instance+5)){
        detector3_a++; //次数递增
        detector3_b=0;//清零保证连续性
```

```
            if(detector3_a==3){//连续三次正转需求则正转
                motor_flag=1;//电机正转
                Speed_L=4;     //低速行驶
                Speed_R=4;
                detector3_a=0;
            }

        }
        if(Range_include_first==0&&(Range<target_instance-5)){
                detector3_b++;
                detector3_a=0;
            if(detector3_b==3){//连续三次反转需求则反转
                buzzer_led_flag=1;//报警
                motor_flag=0;//电机反转
                Speed_L=4;     //低速行驶
                Speed_R=4;
                detector3_b=0;
            }

        }else{
            buzzer_led_flag=0;
        }
        //=========================================================

        Delay(10);//10ms 延时

    //=========超声波距离显示==========================
        if(display_flag){//
            Range_Display();
            display_flag=0;
        }
        //==============================


        //==============定位距离调节==================
        //Delay(20);//按键后延消抖，前面超声波有延时可以不用
        k = KeyScan();                  //扫描按键
        if ( k != '\0')                 //首次检测到按键按下
        {
            Delay(20);                          //延时 20ms，按键前沿消抖
            k = KeyScan();                  //再次读取按键状态
            if ( k != '\0')                 //确认按键按下后处理按键
            {
                if ( k == '+')
```

```
                {
                    buzzer=0;//按键提示音
                    Delay(100);
                    buzzer=1;

                    if ( target_instance < 150 )
                        target_instance=target_instance+5;
                }
                if ( k == '-')
                {
                    buzzer=0;//按键提示音
                    Delay(100);
                    buzzer=1;

                    if ( target_instance > 100 )
                        target_instance=target_instance-5;

                }

                DispBuf[0] = Tab[target_instance % 10];
                DispBuf[1] = Tab[target_instance / 10 % 10] + 0x80;
                DispBuf[2] = Tab[target_instance / 100 % 10];

                while( KeyScan() != '\0');          //等待松开按键
            }
        }
        //=============================================

        //==============adc 采集光敏数据================
            PCF8591_SendByte(AddWr,0);
            light = PCF8591_RcvByte(AddWr);
            Pcf8591_DaConversion(AddWr,0, light);//将 ad 输入结果作为 da 输出
        //======遮光启动==================
            if(light>130){
            timer_start_flag=1;//计时启动
            PWM_EN1 = 1;                //电机有效
            PWM_EN2 = 1;
        }
        //=========================================


    }

}
```