一、智能小车超声波测距实验

1、HC-SR04（超声波）模块介绍

（1）模块介绍

VCC引脚：接+5V。

GND引脚：接GND。

Trig引脚：实际是trigger，有触发引发的意思，到时候只需要给这个引脚一个持续10us的高电平，HC-SR04就会自动地发射8个40KHz的方波（即为超声波）。

Echo引脚：echo实际是是回声回波的意思，当HC-SR04成功的向外发射超声波的时刻开始，这个引脚就会变成高电平，高电平会一直持续到HC-SR04接收到回波为止。

注意：由上面的各引脚的功能我们可知道Trig和Echo必须接在arduino的D口（即数字端口）

（2）本产品特点

A. 感应角度：不大于15 度。
B. 探测距离：2cm-400cm
C. 高精度：可达0.3cm。
D. 盲区（2cm）超近。

2、超声波测距原理

（1）理论解释

超声波发射器向某一方向发射超声波，在发射的同时开始计时，超声波在空气中传播，途中碰到障碍物就立即返回来，超声波接收器收到反射波就立即停止计时。声波在空气中的传播速度为340m/s，根据计时器记录的时间t，就可以计算出发射点距障碍物的距离s，即：s=340m/s×t / 2 。简单的来说就是利用了时间差测距法。

（2）具体处理

A. 单位换算问题

在智能小车的实验中，我们得到的时间，距离得到的单位分别是us,cm，所以要注意单位的换算，换算后的公式为：

S = (1/2) * 340 * (time /10^6) * 100 = time / 58

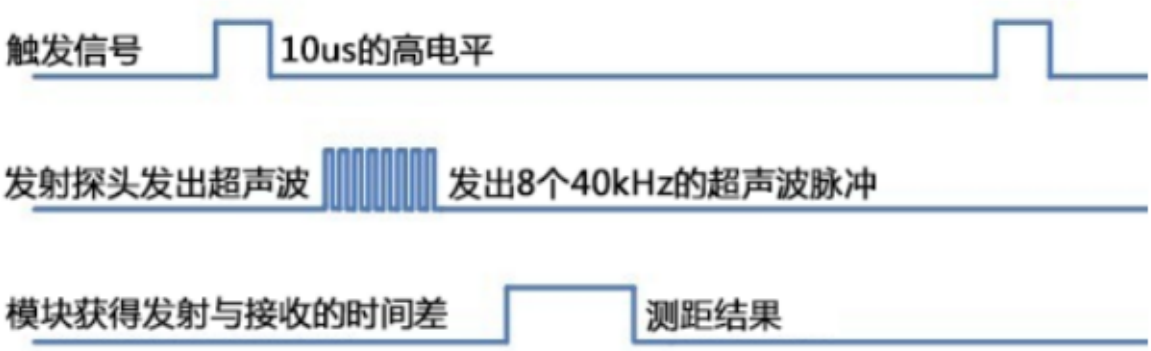
B.对超声波模块返回信号处理问题

在测距函数中调用了函数pulseIn，这个函数的作用是把Echo引脚高电平的持续时间测出，并返回，返回值的单位是微秒us。下面介绍一些这个函数的要点，我们在第一次的文档中已经记录过。

pulseIn函数知识要点：

pulseIn()：用于检测引脚输出的高低电平的脉冲宽度。
pulseIn(pin, value)
pulseIn(pin, value, timeout)
Pin---需要读取脉冲的引脚
Value---需要读取的脉冲类型，HIGH或LOW

Timeout---超时时间，单位微秒，数据类型为无符号长整型。
使用方法及时序图：



- 使用Arduino采用数字引脚给SR04的Trig引脚至少10μs的高电平信号，触发SR04模块测距功能；
- 触发后，模块会自动发送8个40KHz的超声波脉冲，并自动检测是否有信号返回。这步会由模块内部自动完成。
- 如有信号返回，Echo引脚会输出高电平，高电平持续的时间就是超声波从发射到返回的时间。此时，我们能使用pulseIn()函数获取到测距的结果，并计算出距被测物的实际距离。

3、代码解释

Echo，Trig分别为超声波模块的输入输出脚，给超声波模块的输入输出信息定义接口

```
int Echo = A1;  // Echo回声脚(P2.0)
int Trig =A0;  //  Trig 触发脚(P2.1)
```

Distance_test()函数时超声波测距的关键函数，主要利用了前面提到的函数pulseIn。pulseIn函数其实就是一个简单的测量脉冲宽度的函数，默认单位是us。也就是说pulseIn测出来的是超声波从发射到接收所经过的时间。　对除数58的理解是，声音在干燥、摄氏 20度的空气中的传播速度大约为343米/秒，我们作一下单位换算为34,300厘米/秒。所以实际距离就是1厘米，对应58.3微秒。实际上整个测距过程是测的发出声波到收到回波的时间，程序里的时间单位为us。所以换成距离cm，要除以58。当然除以58.3可能更精确。

```
void Distance_test()   // 量出前方距离
{
  digitalWrite(Trig, LOW);   // 给触发脚低电平2μs
  delayMicroseconds(2);
  digitalWrite(Trig, HIGH);  // 给触发脚高电平10μs，这里至少是10μs，触发测距功能
  delayMicroseconds(10);
  digitalWrite(Trig, LOW);    // 持续给触发脚低电
  float Fdistance = pulseIn(Echo, HIGH);  // 读取高电平时间(单位：微秒)
  Fdistance= Fdistance/58;
  Serial.print("Distance:");      //输出距离（单位：厘米）
  Serial.println(Fdistance);         //显示距离
  Distance = Fdistance;
}
```

二、智能小车超声波避障实验

1. 超声波避障实验（无舵机）

（1）超声波避障原理

和红外避障原理一致，只是探测障碍物的元件发生了变化。由超声波模块发射超声波，计算前方物体的距离，障碍物距离设置为2cm-30cm之间，如果距离处于这个之间，那么就需要调用后退函数，然后调用左转函数改变运动状态。

但是超声波模块探测障碍物是有局限的，一方面是距离的局限，一方面是时间的限制。就目前的实验来看，超声避障的效果是不如红外避障的效果的。这是因为超声波模块探测的范围很窄，时常还会探测到一些非前行道路上的障碍物造成干扰。

（2）代码解释

接口设置，与红外避障相比，只是把输出输出脚由原来的红外传感器变为了超声波模块

```
void setup()
{
  Serial.begin(9600);     // 初始化串口
  //初始化电机驱动IO为输出方式
  pinMode(Left_motor_go,OUTPUT); // PIN 8 (PWM)
  pinMode(Left_motor_back,OUTPUT); // PIN 9 (PWM)
  pinMode(Right_motor_go,OUTPUT);// PIN 10 (PWM)
  pinMode(Right_motor_back,OUTPUT);// PIN 11 (PWM)
  // pinMode(SensorRight, INPUT); //定义右循迹红外传感器为输入
  // pinMode(SensorLeft, INPUT); //定义左循迹红外传感器为输入
  //pinMode(SensorRight_2, INPUT); //定义右红外传感器为输入
  //pinMode(SensorLeft_2, INPUT); //定义左红外传感器为输入
  //初始化超声波引脚
  pinMode(Echo, INPUT);    // 定义超声波输入脚
  pinMode(Trig, OUTPUT);   // 定义超声波输出脚
  pinMode(key,INPUT);//定义按键接口为输入接口
  pinMode(beep,OUTPUT);

  lcd.begin(16,2);       //初始化1602液晶工作                                模式
  //定义1602液晶显示范围为2行16列字符
}
```

主函数，每个主函数都调用了按键扫描函数，小车的行动是由按键控制的。

```
void loop()
{
  keysacn();        //调用按键扫描函数
  while(1)
  {
    Distance_test();//测量前方距离
    Distance_display();//液晶屏显示距离
    if(Distance < 30)//数值为碰到障碍物的距离，可以按实际情况设置
      while(Distance < 30)//再次判断是否有障碍物，若有则转动方向后，继续判断
      {
        back(10);//后退
        right(1);//右转
        //brake(1);//停车
        Distance_test();//测量前方距离
        Distance_display();//液晶屏显示距离
      }
```

```
    else
      run();//无障碍物，直行
  }
}
```

2. 超声波避障实验（有舵机）

（1）舵机的作用

加上舵机后解决了超声波探测范围很窄的问题，大大拓宽了超声波的探测范围。

（2）代码解释

对于超声波测量的距离设置了三个变量

```
int Front_Distance = 0;//距前面障碍物的距离
int Left_Distance = 0;//距左边障碍物的距离
int Right_Distance = 0;//距右边障碍物的距离
```

接口设置函数，增加了设定舵机接口为输出接口，pinMode(servopin,OUTPUT);

```
void setup()
{
  Serial.begin(9600);     // 初始化串口
  //初始化电机驱动IO为输出方式
  pinMode(Left_motor_go,OUTPUT); // PIN 8 (PWM)
  pinMode(Left_motor_back,OUTPUT); // PIN 9 (PWM)
  pinMode(Right_motor_go,OUTPUT);// PIN 10 (PWM)
  pinMode(Right_motor_back,OUTPUT);// PIN 11 (PWM)
  pinMode(key,INPUT);//定义按键接口为输入接口
  pinMode(beep,OUTPUT);

  //初始化超声波引脚
  pinMode(Echo, INPUT);    // 定义超声波输入脚
  pinMode(Trig, OUTPUT);   // 定义超声波输出脚
  lcd.begin(16,2);       //初始化1602液晶工作                              模式
  //定义1602液晶显示范围为2行16列字符
  pinMode(servopin,OUTPUT);//设定舵机接口为输出接口
}
```

定义一个脉冲函数，用来模拟方式产生PWM值舵机的范围是0.5MS到2.5MS 1.5MS 占空比是居中周期是20MS

```
void servopulse(int servopin,int myangle)
{
  pulsewidth=(myangle*11)+500;//将角度转化为500-2480 的脉宽值 这里的myangle就是0-180度所以180*11+50=2480  11是为了换成90度的时候基本就是1.5MS
  digitalWrite(servopin,HIGH);//将舵机接口电平置高
   90*11+50=1490uS  就是1.5ms
  delayMicroseconds(pulsewidth);//延时脉宽值的微秒数  这里调用的是微秒延时函数
  digitalWrite(servopin,LOW);//将舵机接口电平置低
 // delay(20-pulsewidth/1000);//延时周期内剩余时间  这里调用的是ms延时函数
  delay(20-(pulsewidth*0.001));//延时周期内剩余时间  这里调用的是ms延时函数
}
```

这是前方，左方，右方的距离探测函数。程序中电脑打印数值部分都被屏蔽了，打印会影响小车遇到障碍物的反应速度，调试时可以打开屏蔽内容Serial.print，打印测到的距离。

```
void front_detection()
{
  //此处循环次数减少，为了增加小车遇到障碍物的反应速度
  for(int i=0;i<=5;i++) //产生PWM个数，等效延时以保证能转到响应角度
  {
    servopulse(servopin,90);//模拟产生PWM
  }
  Front_Distance = Distance_test();
  //Serial.print("Front_Distance:");       //输出距离（单位：厘米）
 // Serial.println(Front_Distance);          //显示距离
 //Distance_display(Front_Distance);
}

void left_detection()
{
  for(int i=0;i<=15;i++) //产生PWM个数，等效延时以保证能转到响应角度
  {
    servopulse(servopin,175);//模拟产生PWM
  }
  Left_Distance = Distance_test();
  //Serial.print("Left_Distance:");       //输出距离（单位：厘米）
  //Serial.println(Left_Distance);          //显示距离
}

void right_detection()
{
  for(int i=0;i<=15;i++) //产生PWM个数，等效延时以保证能转到响应角度
  {
    servopulse(servopin,5);//模拟产生PWM
  }
  Right_Distance = Distance_test();
  //Serial.print("Right_Distance:");       //输出距离（单位：厘米）
  //Serial.println(Right_Distance);          //显示距离
}
```

主函数

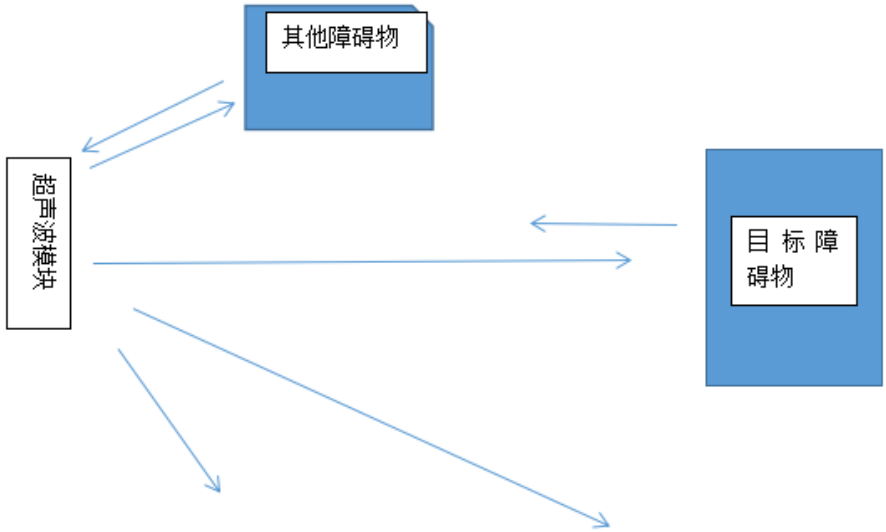
```
void loop()
{
  keysacn();       //调用按键扫描函数
  while(1)
  {
    front_detection();//测量前方距离
    if(Front_Distance < 30)//当遇到障碍物时
    {
      back(2);//后退减速
      brake(2);//停下来做测距
      left_detection();//测量左边距障碍物距离
      Distance_display(Left_Distance);//液晶屏显示距离
      right_detection();//测量右边距障碍物距离
      Distance_display(Right_Distance);//液晶屏显示距离
      if((Left_Distance < 30 ) &&( Right_Distance < 30 ))//当左右两侧均有障碍物靠得比
较近
```

```
          spin_left(0.7);//旋转掉头
        else if(Left_Distance > Right_Distance)//左边比右边空旷
        {
          left(3);//左转
          brake(1);//刹车，稳定方向
        }
        else//右边比左边空旷
        {
          right(3);//右转
          brake(1);//刹车，稳定方向
        }
      }
      else
      {
        run(); //无障碍物，直行
      }
    }
}
```

三、遇到的问题

1. 超声波模块本身的限制

（1）在把超声波对着目标障碍物的时候，模块与目标障碍物之间的其他障碍物【有可能】会影响测距的准确性，因为声波的方向性并不强，并不是只朝着一个方向传播的，当沿某一个方向传播的超声波遇到的障碍物比目标障碍物的距离要近并且回波成功的被超声波模块接收到的时候测出的距离值就是不准的了。



（2）超声波模块探测距离范围是有局限的，为2cm-400cm,超出这个范围则不能被探测到，在实验中明显发现，当距离太近时，超声波探测距离会出bug。

2. 速度改变是一个渐变的过程

超声波模块接收到探测波后，经过计算后发现前方有障碍物，然后执行函数back（）（后退函数），但是这个时候小车仍具有一个向前的速度，由于设置的避障距离太短（20cm-30cm),小车有时会撞在障碍物上。

3. 接线太多，影响到舵机的运作

起初没有考虑到车上会有一些障碍物（线），致使小车做出不正确的判断，然后处理了一下接线，让接线不在超声波模块的探测范围中。

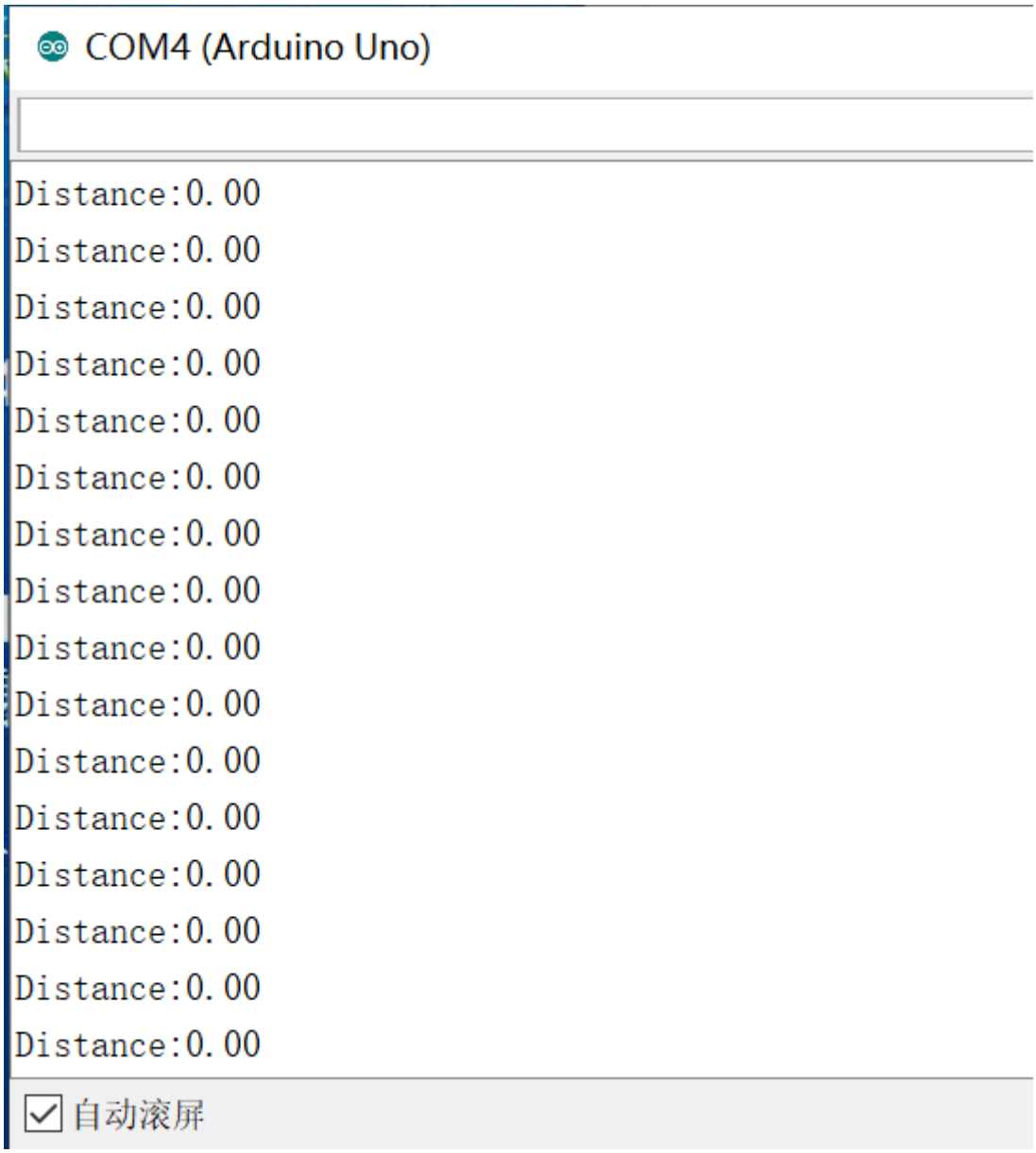
4. 舵机本身性能不是很好

在测试中发现小车总是向右探测，舵机的灵活性不是很好

5. 一些技术问题

主要是一些接线问题，液晶显示屏并不显示

起初串口监视器输出一直为0，然后仔细检查了接线，超声波模块接线有问题。



把按键的一端直接接在了GND上，使按键失去了作用。

接线实在是太多了，很容易弄掉，然后就要每个模块检查是哪里的接线.....