第二阶段进度展示

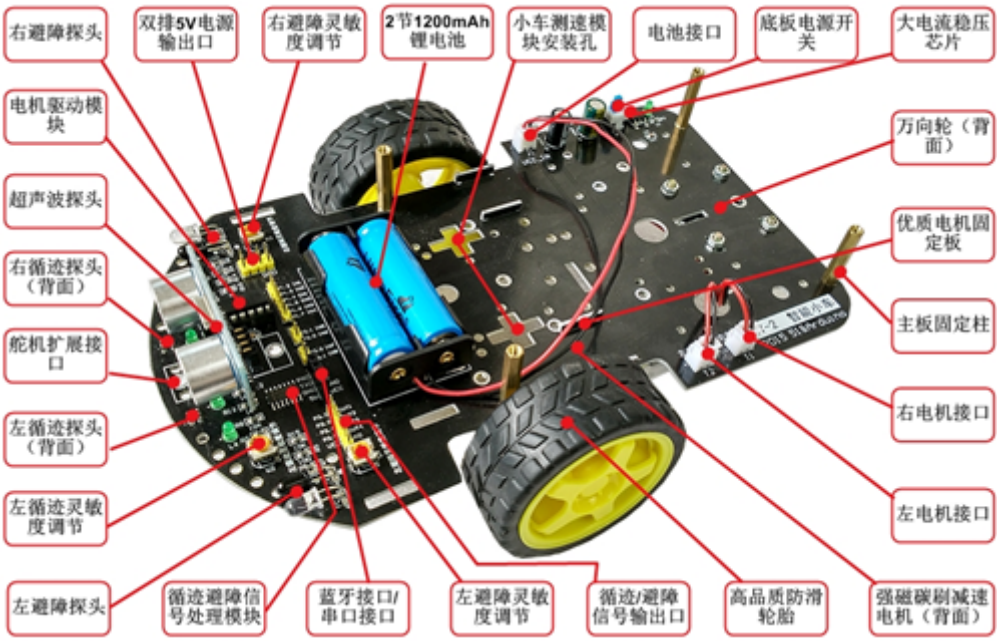
组员：车永祥 周珂

一、小车循迹的实现过程

1.实现原理

（1）底板上的两个红外传感器是向下发射红外光线的，当接收到地面发射回来的红外线时指示灯会亮。

（2）底板上两个红外传感器的位置分布比较关键：他们分布在底板车头位置中间（即图中左右循迹探头）



（3）循迹传感器的主要作用是探测地面情况（是否接收到反射光），当左（或者右）循迹探头正下方为黑色时，接收不到反射光，指示灯不亮

（4） 我们要设计一个宽度合适的黑色轨迹（宽度应尽量接近左右循迹探头的间距）

（5） 接下来就是算法设计了：我们有许多函数接口（笔直前进，左右转，后退等），基本的思路是：当左循迹探头指示灯亮而右循迹指示灯不亮时，说明黑色轨迹（黑色胶布）在小车右侧，此时应当右转；当右循迹探头指示灯亮而左循迹指示灯不亮时，说明黑色轨迹（黑色胶布）在小车左侧，此时应当左转；当两个循迹指示灯都亮或者都不亮时，保持直行，直到出现上面两种情况

2.影响因素分析：

（1） 实验的光线环境：光线太强的环境之下，传感器可能接收到来自环境光线中的红外光线，而使得传感器失去原本的作用，光线太弱，可以理解为周围环境接近黑色轨迹，吸收大量传感器发出的红外光线，而显示亮灯都一直不亮的情况

（2） 黑色轨迹的宽度：黑色轨迹的宽度如果太大，会导致小车每次遇到轨迹边缘时转向，转向后直行到另一边缘再转向，这样小车对于轨迹的追随摇摆幅度过大；如果黑色轨迹宽度太小，那么可能由于小车的惯性，而使得算法不能及时地执行，使得小车轻易脱离轨迹

（3） 小车的速度：小车的速度越慢，算法的执行越准确和即时，速度过快，而红外传感器的信息是即时的，而算法执行需要一定的时间，这样小车容易脱离轨迹

（4） 传感器灵敏度：传感器灵敏度很关键，而且影响的作用方式类似环境光线（环境光线通过影响环境对红外光线的吸收程度发挥作用，灵敏度通过调整传感器对于红外光线的灵敏度来发挥作用，通俗来讲就是调节吸收多少红外光线指示灯会亮）

3. 实验调试：

（1）实验前说明：因为实验中的环境光线适中，且调节不易，而黑色轨迹的宽度近似于两个循迹探头的间距，我们可以将这两个影响因素忽略（可视为这两个因素已经是最优的状态）

（2）正式调试：初次实验发现小车无法循迹，说明其余的两个条件都不友好

a.首先：我调整了两个红外探头的灵敏度：什么样才算是合适的灵敏度呢？放在正常地面上时，两个循 迹指示灯亮起，而正下方有黑色胶布条时，指示灯不亮

b.接着：我修改了各个接口函数的速度这些数值，减慢了直行的速度和转向的速度（整体调慢），提升 了转向的相对速度比例（比如说之前前进速度150，转向速度200，不到前进速度的两倍；现在调整 为：前进速度50，转向速度100，增大为前进速度的两倍)旨在放慢整体速度，增加转向速度

（3）通过调整之后，小车准确的按照黑色轨迹前进，实验成功。

但对比多次实验，依然有着一些问题：

a.通过蜂鸣器指示功能的启动，（松开开关，蜂鸣器响短暂一会，循迹开始）正常情况下，“哔”短暂一 下就开始运行，但有时就会产生蜂鸣器一直响的问题（但设置为按下开关蜂鸣器响，松开开关蜂鸣器 停止响，问题就解决了）

b.小车偶尔也会脱离轨道，分析原因和之前类似，再通过类似的调整问题就可以解决

二、智能小车红外避障

1. 实验原理

利用小车的两个红外避障探头，把红外避障探头探测到的信号作为输入，然后根据返回信号的不同来调节小车的行动。如果前方有障碍物，避障指示灯亮，返回信号标记为low，表示低电平；如果前方没有障碍物，避障指示灯不亮，返回信号记为high，表示高电平。

自然地就有，左边有障碍物，左避障指示灯亮，小车向右转；右边有障碍物，右避障指示灯亮，小车向左转；前面有障碍物，左右避障指示灯都亮，小车先停止，然后后退，试探左右转动绕开障碍物。

2. 红外避障代码解释

（1） 参数设置

```
int Left_motor_go=8;     //左电机前进(IN1)
int Left_motor_back=9;     //左电机后退(IN2)

int Right_motor_go=10;    // 右电机前进(IN3)
int Right_motor_back=11;    // 右电机后退(IN4)

int key=7;//定义按键 数字7 接口
int beep=12;//定义蜂鸣器 数字12 接口

const int SensorRight = 3;    //右循迹红外传感器(P3.2 OUT1)
const int SensorLeft = 4;       //左循迹红外传感器(P3.3 OUT2)

const int SensorRight_2 = 5;    //右红外传感器(P3.4 OUT3)
const int SensorLeft_2 = 6;     //左红外传感器(P3.5 OUT4)
```

```
15
16 int SL;    //左循迹红外传感器状态
17 int SR;    //右循迹红外传感器状态
18 int SL_2;    //左红外传感器状态
19 int SR_2;    //右红外传感器状态
20
```

(2) run()函数

前行即给左右电机都传入high信号，这个run函数在这里设置了一个较低的速度，否则在小车避障时会出现不能及时停止的状况。Right_motor_go，Right_motor_back，Left_motor_go，Left_motor_back信号设置为1,0,0,1

```
1  void run()     // 前进
2  {
3    digitalWrite(Right_motor_go,HIGH);  // 右电机前进
4    digitalWrite(Right_motor_back,LOW);
5    analogWrite(Right_motor_go,130);//PWM比例0~255调速，左右轮差异略增减
6    analogWrite(Right_motor_back,0);
7    digitalWrite(Left_motor_go,LOW);  // 左电机前进
8    digitalWrite(Left_motor_back,HIGH);
9    analogWrite(Left_motor_go,0);//PWM比例0~255调速，左右轮差异略增减
10   analogWrite(Left_motor_back,130);
11   //delay(time * 100);   //执行时间，可以调整
12 }
```

(3) 左转函数

右轮的速度大于左轮才能实现左转，所以左轮的设置有很多种，这里写了了左轮不前进也不后退的左转和左轮向后旋转的左转。

左轮不动，右轮前进时，Right_motor_go，Right_motor_back，Left_motor_go，Left_motor_back信号设置为1,0,0,0

左轮后退，右轮前进时，Right_motor_go，Right_motor_back，Left_motor_go，Left_motor_back信号设置为1,0,1,0

```
1  void left()         //左转(左轮不动，右轮前进)
2  {
3     digitalWrite(Right_motor_go,HIGH); // 右电机前进
4    digitalWrite(Right_motor_back,LOW);
5    analogWrite(Right_motor_go,130);
6    analogWrite(Right_motor_back,0);//PWM比例0~255调速
7    digitalWrite(Left_motor_go,LOW);   //左轮后退
8    digitalWrite(Left_motor_back,LOW);
9    analogWrite(Left_motor_go,0);
10   analogWrite(Left_motor_back,0);//PWM比例0~255调速
11   //delay(time * 100);  //执行时间，可以调整
12 }
13
14 void spin_left(int time)         //左转(左轮后退，右轮前进)
15 {
16    digitalWrite(Right_motor_go,HIGH); // 右电机前进
17   digitalWrite(Right_motor_back,LOW);
18   analogWrite(Right_motor_go,130);
```

```
  analogWrite(Right_motor_back,0);//PWM比例0~255调速
  digitalWrite(Left_motor_go,HIGH);   //左轮后退
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,130);
  analogWrite(Left_motor_back,0);//PWM比例0~255调速
  delay(time * 100);  //执行时间，可以调整
}
```

（4） 右转函数

左轮的速度大于右轮才能实现左转，所以右轮的设置也有很多种，这里写了了右轮不前进也不后退的右转和右轮向后旋转的右转。

右轮不动，左轮前进时，Right_motor_go，Right_motor_back，Left_motor_go，Left_motor_back信号设置为0,0,0,1

右轮后退，左轮前进时，Right_motor_go，Right_motor_back，Left_motor_go，Left_motor_back信号设置为0,1,0,1

```
void right()        //右转(右轮不动，左轮前进)
{
   digitalWrite(Right_motor_go,LOW);   //右电机后退
  digitalWrite(Right_motor_back,LOW);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,0);//PWM比例0~255调速
  digitalWrite(Left_motor_go,LOW);//左电机前进
  digitalWrite(Left_motor_back,HIGH);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,130);//PWM比例0~255调速
  //delay(time * 100);  //执行时间，可以调整
}

void spin_right(int time)        //右转(右轮后退，左轮前进)
{
  digitalWrite(Right_motor_go,LOW);   //右电机后退
  digitalWrite(Right_motor_back,HIGH);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,130);//PWM比例0~255调速
  digitalWrite(Left_motor_go,LOW);//左电机前进
  digitalWrite(Left_motor_back,HIGH);
  analogWrite(Left_motor_go,0);
  analogWrite(Left_motor_back,130);//PWM比例0~255调速
  delay(time * 100);  //执行时间，可以调整
}
```

（5）刹车、 后退函数

刹车即所有的信号都设为low，Right_motor_go，Right_motor_back，Left_motor_go，Left_motor_back信号设置为0,0,0,0

后退即将 后退的信号设为low，Right_motor_go，Right_motor_back，Left_motor_go，Left_motor_back信号设置为0,1,1,0

```
void brake(int time)  //刹车，停车
```

```
{
  digitalWrite(Right_motor_go,LOW);
  digitalWrite(Right_motor_back,LOW);
  digitalWrite(Left_motor_go,LOW);
  digitalWrite(Left_motor_back,LOW);
  delay(time * 100);//执行时间，可以调整
}
void back(int time)          //后退
{
  digitalWrite(Right_motor_go,LOW);  //右轮后退
  digitalWrite(Right_motor_back,HIGH);
  analogWrite(Right_motor_go,0);
  analogWrite(Right_motor_back,150);//PWM比例0~255调速
  digitalWrite(Left_motor_go,HIGH);  //左轮后退
  digitalWrite(Left_motor_back,LOW);
  analogWrite(Left_motor_go,150);
  analogWrite(Left_motor_back,0);//PWM比例0~255调速
  delay(time * 100);     //执行时间，可以调整
}
```

（6）调用基础函数实现避障

```
void loop()
{
  keysacn();     //调用按键扫描函数
  while(1)
  {
    //有信号为LOW  没有信号为HIGH
    SR_2 = digitalRead(SensorRight_2);
    SL_2 = digitalRead(SensorLeft_2);
    if (SL_2 == HIGH&&SR_2==HIGH)
      run();   //前方没有障碍物时，调用前进函数
    else if (SL_2 == HIGH & SR_2 == LOW)// 右边探测到有障碍物，有信号返回，向左转，一定要向左转，否则会陷入死循环
        left();
    else if (SR_2 == HIGH & SL_2 == LOW) //左边探测到有障碍物，有信号返回，向右转

      right();
    else // 都是有障碍物，后退
    {
      back(4.5);//后退
      spin_right(4.5);//有旋转，调整方向
    }
  }
}
```

3. 变形应用——跟随障碍物

跟随障碍物和躲避障碍物恰好是对相同的信号两种完全不同的处理方式，躲避障碍物是返回来低电平则躲避或者停止运动，而躲避障碍物则是遇到低电平继续前行。所以小车的基础行为函数都不用变化，和避障代码完全相同，只在调用时将high和low调换。

```
void loop()
```

```
{
  keysacn();     //调用按键扫描函数
  while(1)
  {
    //有信号为LOW  没有信号为HIGH
    SR_2 = digitalRead(SensorRight_2);
    SL_2 = digitalRead(SensorLeft_2);
    if (SL_2 == LOW&&SR_2==LOW)
      run();   //此时前面有障碍物，调用run函数，继续前行
    else if (SL_2 == HIGH & SR_2 == LOW)// 右边探测到有障碍物，向右转，即实现跟随
      right();
    else if (SR_2 == HIGH & SL_2 == LOW) //左边探测到有障碍物，向左转 ，即实现跟随
      left();
    else // 没有障碍物，停
    brake();
  }
}
```

这让我们思考，除了黑线循迹，小车的前行还可以由障碍物引导，跟随障碍物的应用应该也会有实际的意义。