**2022“数智中原”河南省大学生电子设计大赛**

**（I 题）**

**【本科组】**



**2022年11月9日**

自动绕障行走小车（I题）

摘 要

本作品采用STM32F103ZET6作为主控，以三轮小车套件作为框架，采用HI229九轴陀螺仪和HR-SC04超声波测距传感器、编码器电机等模块。工作时主要以超声波测距模块得到的数据为依据，进行及时地转向，绕过障碍，其中以陀螺仪和编码器反馈的小车位置姿态数据，不断调整，保证小车路线行驶准确，完成自动绕障行走任务。

关键词：STM32F103ZET6；超声波测距模块；陀螺仪；编码器电机；

Abstract

This work uses STM32F103ZET6 as the main control, three-wheeled trolley kit as the frame, and HIM229 nine-axis gyroscope, HR-SC04 ultrasonic ranging sensor, encoder motor and other modules. When working, it is mainly based on the data obtained from the ultrasonic ranging module to timely turn and bypass obstacles. The position and attitude data of the car fed back by the gyroscope and encoder are constantly adjusted to ensure the accuracy of the car route and complete the task of automatic walking around obstacles.

Key words: STM32F103ZET6; Ultrasonic ranging module; Gyroscope; Encoder motor；

目录

[自动绕障行走小车（I题） 2](#_Toc122016952)

[一、题目分析与方案论证 3](#_Toc122016953)

[1.1题目分析 3](#_Toc122016954)

[1.1.1题目任务 3](#_Toc122016955)

[1.1.2基本要求 3](#_Toc122016956)

[1.2主控选择 3](#_Toc122016957)

[1.3功能模块选择 4](#_Toc122016958)

[1.3.1姿态模块选择 4](#_Toc122016959)

[1.3.2驱动模块选择 5](#_Toc122016960)

[1.3.3障碍识别模块 5](#_Toc122016961)

[二、控制系统 5](#_Toc122016962)

[2.1控制思路 5](#_Toc122016963)

[2.2程序流程图 6](#_Toc122016964)

[2.3 PID控制算法 6](#_Toc122016965)

[三、硬件设计 8](#_Toc122016966)

[3.1电源 8](#_Toc122016967)

[3.2稳压模块 8](#_Toc122016968)

[四、理论分析及设计 8](#_Toc122016969)

[4.1 小车运行方案设计 8](#_Toc122016970)

[五、测试的方案与测试结果 9](#_Toc122016971)

[5.1、测试方案 9](#_Toc122016972)

[5.1.1检查电路 9](#_Toc122016973)

[5.1.2通电观察 9](#_Toc122016974)

[5.1.3观察波形与数据 9](#_Toc122016975)

[5.1.4组装测试 9](#_Toc122016976)

[5.1.5 PID参数整定 9](#_Toc122016977)

[5.2、测试结果 10](#_Toc122016978)

[5.3分析与结论 11](#_Toc122016979)

[附录一：主要源代码 11](#_Toc122016980)

# 一、题目分析与方案论证

## 1.1题目分析

### 1.1.1题目任务

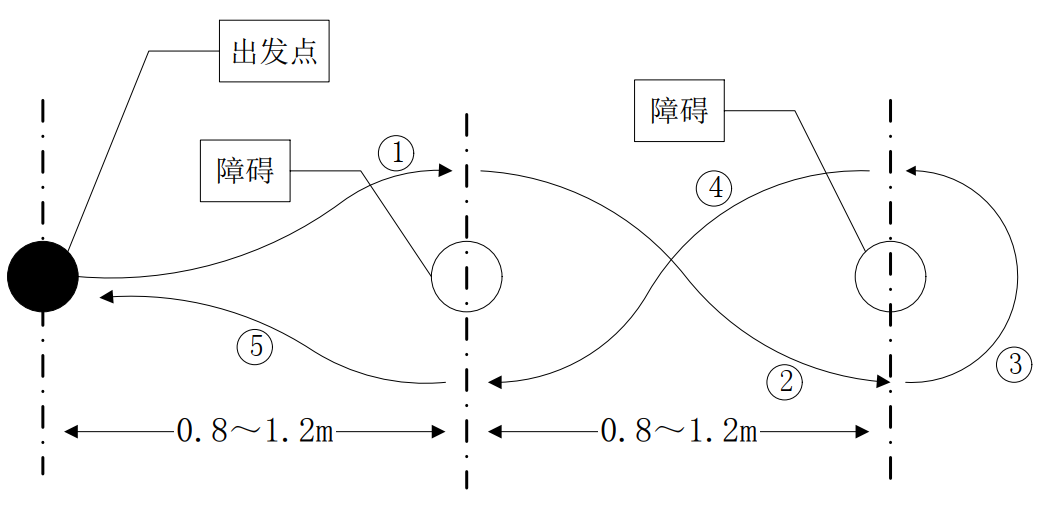
制作一个能自动绕越障碍物的电动小车。

图1 基本要求绕障示意图

### 1.1.2基本要求

如图一所示，地面安放 2 个直径约 15cm、高度 20cm 的圆柱形红色障碍物，两障碍物间距离在 0.8~1.2m 间可以调整。小车从处于二障碍物连线左侧 0.8~1.2m的黑色实心圆（直径 15cm）出发，沿图示箭头轨迹运动，最后回到出发点。行走过程中，小车不得碰撞障碍物；小车中心标志距离障碍物的距离不得超过 40cm；返回出发点后小车中心标志应尽量与黑色实心圆中心重合；小车完成整个行程不得超过 30 秒。

## 1.2主控选择

方案一：采用Arduino系列单片机。Arduino单片机模块库丰富，使用调用库的方式开发会较为简单，高效，使用方便，但此单片机字长有限，数据处理能力很弱，处理速度较慢，资源不够丰富，难以满足本设计要求。

方案二：采用STM32F1系列单片机。STM32系列单片机外设丰富，功能强大，32bits的处理能力和72Mhz的主频能够完成大多数任务，资料丰富，标准库、HAL库开放方式成熟，调用起来简单高效。

总结：最终我们团队采用STM32F103ZET6单片机作为主控，主要使用了其自带硬件定时器、串口、GPIO等外设，能够很好地控制外围电路，电机等设备，能够很好地同多个传感器通讯，获取数据，很好地满足了题目设计要求。

## 1.3功能模块选择

### 1.3.1姿态模块选择

方案一：采用MPU6050模块。采用 InvenSense 公司的 MPU6050 作为主芯片， 能同时检测三轴加速度、三轴陀螺仪(三轴角速度)的运动数据以及温度数据。利用 MPU6050 芯片内部的 DMP 模块（Digital Motion Processor 数字运动处理器）， 可对传感器数据进行滤波、融合处理，直接通过 IIC 接口向主控器输出姿态解算后的数据，降低主控器的运算量。但是用于我们需要采集小车运动时的运动方向、运动加速度，MPU6050的六轴传感器无法直接获取Z轴方向的数据，即航向角，需要依靠数学解算，且存在积累误差，我们到手的MPU6050在累计误差上达到每小时漂移60度，故放弃采用。

方案二：采用HI229模块。Hi229 是一款低成本、高性能、小体积、低延时的惯性测量单元(IMU)， 得益于最新的微机电感测技术，产品整合三轴加速度计、三轴陀螺仪、三轴地磁计、微处理器。 依据良好的感测数据可即时输出四元数与欧拉角。其搭载了九轴传感器，能够实时准确的输出航向角和运动加速度，并搭载了微处理器。减少计算负担，累计误差每小时1度，满足我们完成任务要求。

故选择方案二。

### 1.3.2驱动模块选择

方案一：使用 l298n 电机驱动模块，由 12v 稳压电池直接供电，再由微控制板给予 PWM控制信号来控制电机转速和正反转。

方案二：使用 TB6612FNG 电机驱动模块，由 12v 稳压电池直接供电，再由微控制板给予 PWM控制信号来控制电机转速和正反转。

经分析与要求，l298n 相对于 TB6612FNG 模块，占用了更大的体积，效率较低，，对于集成度较高的小车来说，显然选用 TB6612FNG 驱动模块能为小车腾出更多的位置安装其他模块。最终选择方案二。

### 1.3.3障碍识别模块

方案一：红外测距避障模块。红外测距都是采用三角测距的原理。 红外发射器按照一定角度发射红外光束，遇到物体之后，光会反向回来，检测到反射光之后，通过结构上的几何三角关系，就可以计算出物体距离D。而本题目选用的障碍物为圆柱形红色障碍物，且受环境光线等环境因素的干扰，所以不采用红外测距模块。

方案二：超声波模块。超声波测距的原理是利用超声波在空气中的传播速度为已知，测量声波在发射后遇到障碍物反射回来的时间，根据发射和接收的时间差计算出发射点到障碍物的实际距离。超声波模块价格相比双目摄像头较低，相比红外测距模块较为稳定，干扰因素较少。

最终选择方案二。

# 二、控制系统

## 2.1控制思路

小车采用STM32F103ZET6作为主控，利用芯片内部定时器输出PWM波控制TB6612模块输出高低电压控制控制电机转速，并使用定时器的编码器输入模式捕获电机上的编码器的反馈电压，来得到电机转速，进而闭环控制电机转速。

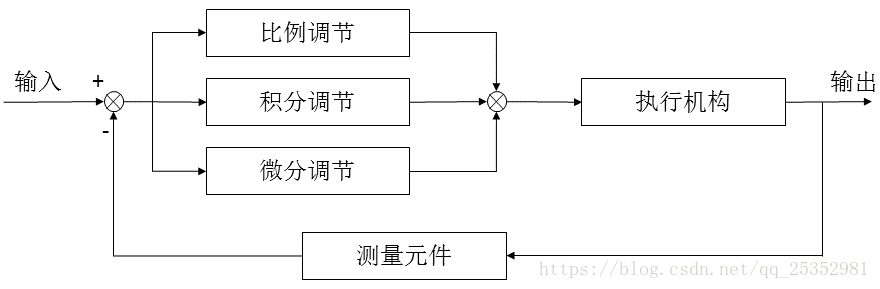
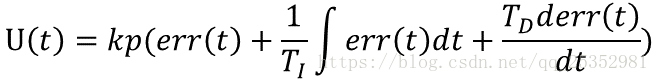
并且开启串口同HI229九轴陀螺仪通讯，实时获取小车姿态，精准控制小车转向，实现直线行走、精准转向，不会偏离行进路线。

在前进过程中，不断采集超声波模块数据，获取前方障碍情况，到达一定距离，进行避障。

## 2.2程序流程图

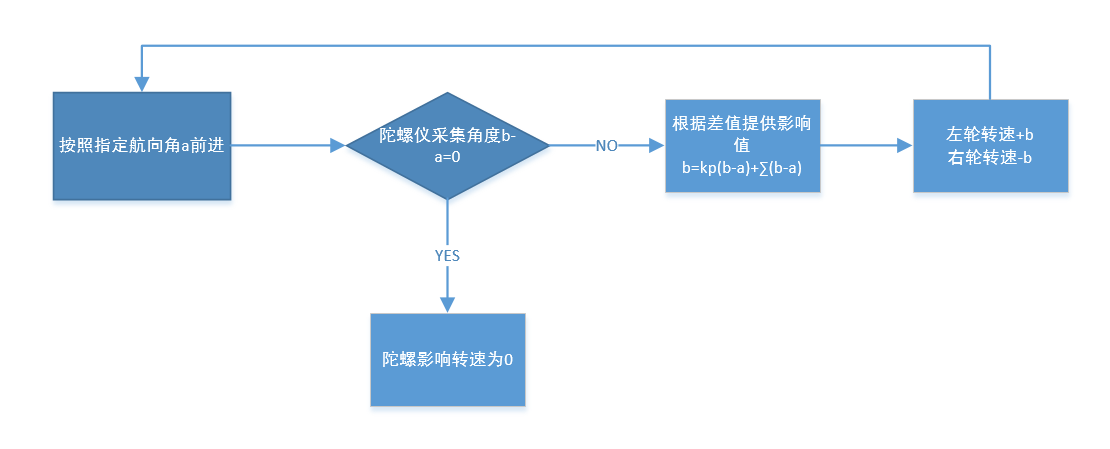
图二 总体程序流程图

## 2.3 PID控制算法

PID控制算法是结合比例、积分和微分三种环节于一体的控制算法，它是连续系统中技术最为成熟、应用最为广泛的一种控制算法，该控制算法出现于20世纪30至40年代，适用于对被控对象模型了解不清楚的场合。实际运行的经验和理论的分析都表明，运用这种控制规律对许多工业过程进行控制时，都能得到比较满意的效果。PID控制的实质就是根据输入的偏差值，按照比例、积分、微分的函数关系进行运算，运算结果用以控制输出。

在作品中，我们多次使用到PID控制算法，主要通过采集电机转速来控制电机稳定目标转速运行。

另外。我们采集小车航向角，利用pid控制算法，产生影响参数，作用在两个电机上，以产生差速来调整小车姿态。

我们还采集计算小车行走的路程，利用pid控制算法，控制小车前进或后退，来控制小车位置。

图三 直线行进程序流程图

# 三、硬件设计

## 3.1电源

采用3s12v直流稳压电池组作为电源进行供电。

## 3.2稳压模块

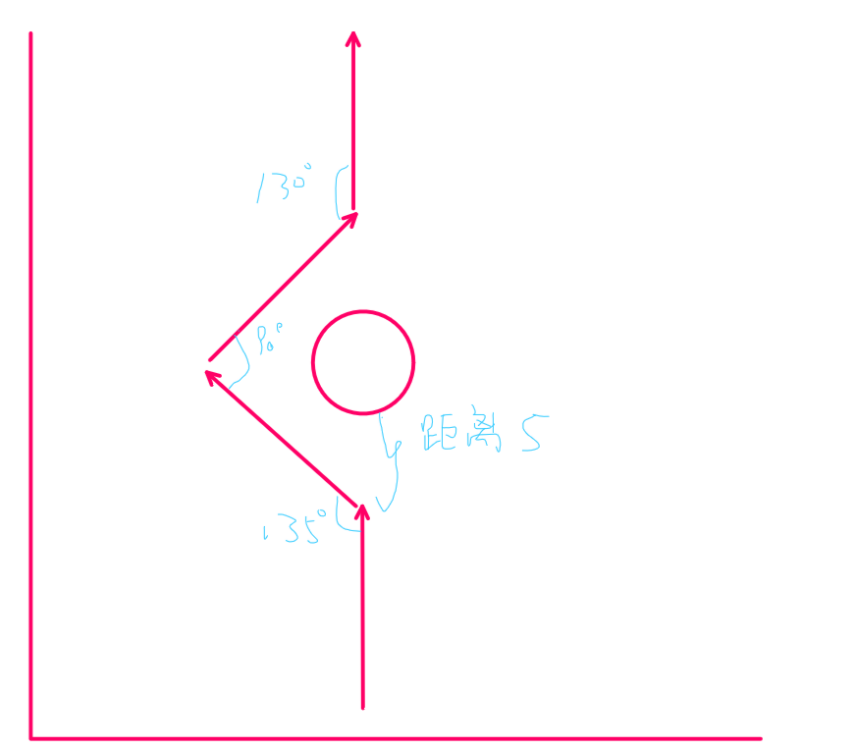
采用LM25965稳压芯片，将12V的直流电源降压至 5v 作为主控模块供电来源。

# 四、理论分析及设计

## 4.1 小车运行方案设计

根据题目要求，车身中心点距离障碍物距离不得大于40cm，以及障碍物位置不定，我们采用定方向行进，遇障碍转向的方式避障。

如图四。小车检测目标障碍物距离S<40cm时，小车向左转45度，直线前进（S+7.5）\*1.41cm，再右转90度，前进（S+7.5）\*1.41cm，完成后，左转45度，回归直线，继续前进，检查下一个目标。

最终绕障碍物三返回时也是如此方案，仅多旋转半周。

图四 小车运行方式图

# 五、测试的方案与测试结果

## 5.1、测试方案

### 5.1.1检查电路

任何电路，通电调试之前都必须再仔细的检查核对电路，，看是否有连线错误的的部分。最重要的是正负极有没有接反，短接，然后是元器件的正负极，还有元器件的引脚有没有焊实。

### 5.1.2通电观察

检查电路无误后，首先要做的不是用仪器观测波形和数据，二是先观察整个电路是否有冒烟、异常气味、发光放电、元器件发烫等现象。如果有，先断电， 继续检查电路，找出故障。再重新通电。

### 5.1.3观察波形与数据

通电观察无异常后，用电压表和示波器测试输入信号和输出信号是否和期望， 代码编写的结果一致。

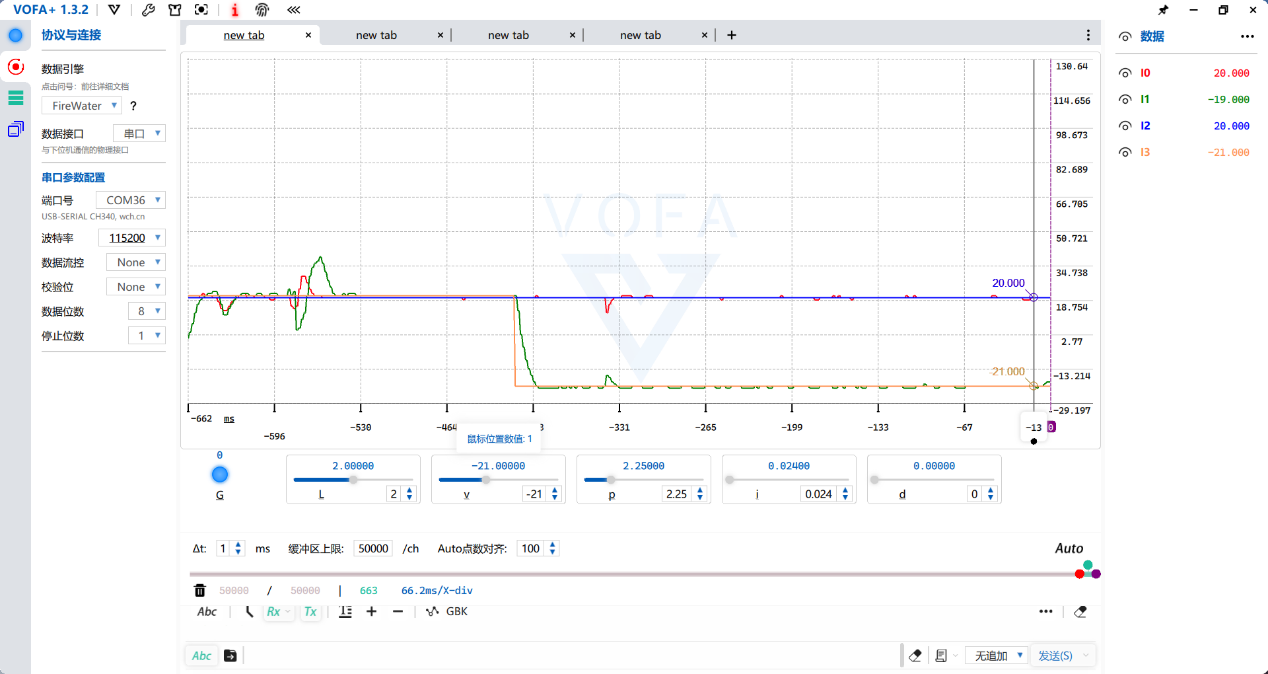
### 5.1.4组装测试

待前几步骤都无误后，就可以组装整个硬件系统测试了。如果测试结果不符合设计要求，应检查问题所在，一般是对某些元件参数加以调整和改变，当然局部返工也是有可能的，这就需要我们有足够的耐心去处理出现的意料之外的问题。

### 5.1.5 PID参数整定

使用pid控制算法，对不同的控制体系需要不同的三个参数，分别是kp,ki,kd.

我们通过蓝牙串口上传采集的数据，并进行参数的调整。

如图五

图五 电机PID控制参数整定

我们还进行了转向PID参数整定，位置PID参数整定，均达到目标效果，运行良好。

## 5.2、测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| 次数 | 完成时间/s | 碰撞障碍物次数 | 中心标志距离障碍物的距离超过 40cm次数 |
| 测试1 | 25 | 3 | 2 |
| 测试2 | 32 | 1 | 2 |
| 测试3 | 34 | 0 | 0 |
| 测试4 | 33 | 0 | 0 |
| 测试5 | 35 | 2 | 1 |
| 测试6 | 35 | 1 | 0 |
| 测试7 | 36 | 0 | 0 |
| 测试8 | 34 | 0 | 0 |
| 测试9 | 35 | 0 | 0 |

## 5.3分析与结论

通过不断增大控制电机PWM参数，减少完成的总时间、调试修改超声波模块距离判别参数，优化减少碰撞障碍物次数和中心标志距离障碍物的距离，不断尝试优化误差补偿，减小失误。通过测试，本作品可以顺利完成题目要求

# 附录一：主要源代码

#define car\_stop\_part 1

#define car\_go\_part 2

#define car\_angle\_part 3

#define car\_circle\_part 4

#define car\_distance\_part 5

/\*\*\*\* \*\*\*全局变量声明调用\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

extern int goal1, goal2,speed1,speed2;

extern raw\_t raw;

extern float distance1,distance2;

extern float kp3,ki3,kd3,kp1,ki1,kd1,kp2,ki2,kd2;

int distance\_sign=0;

float distance;

int goal\_out;

int sign;

void date\_uploading(void);

void set\_distance(float a,float b);

float dert(float a,float b);

uint8\_t angle\_set(float goal\_angle);

int main()

{

/\*\*\*\*\*\*\*\*变量定义\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

float yaw=0;

int distance\_sign=1;

int distance\_back;

/\*\*\*\*\*\*\*板子外设初始化\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

USART\_Config();

LED\_Config();

printf("你是猪");

Encoder\_Init\_TIM2();

Encoder\_Init\_TIM4();

Motor\_Config();

sg\_biu(100);

Motor\_GPIO\_Config();

uart2\_init();

hc\_sr04\_Config();

/\*\*\*\*\*\*\*开始了\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

PEout(5)=1;PEout(6)=1;

BASIC\_TIM\_Init();

distance=get\_hc\_sr();

for(;distance>500;)

{

distance=get\_hc\_sr();

delay\_ms(10);

}

distance\_back=distance\*10;

printf("%d\n",distance\_back);

sign=1;

while(1)

{

switch(sign)

{

case 1:

if(distance<40)

{

yaw=45;sign=2;

distance1=distance2=0;

distance\_sign=0;

}

goal1=goal2=60;

angle\_set(yaw);

break;

case 2:

if((distance1+distance2)/2>=450)

{

yaw=-45;sign=3;

distance1=distance2=0;

}

goal1=goal2=30;

angle\_set(yaw);

break;

case 3:

if((distance1+distance2)/2>=450)

{

yaw=0;sign=4;

distance\_sign=1;

distance=distance1=distance2=0;

}

goal1=goal2=30;

angle\_set(yaw);

break;

//越过第一个障碍

case 4:

if(distance<25&&distance!=0)

{

yaw=-45;sign=5;

distance1=distance2=0;

distance\_sign=0;

}

goal1=goal2=10;

angle\_set(yaw);

break;

case 5:

if((distance1+distance2)/2>=400)

{

yaw=45;sign=6;

distance1=distance2=0;

}

goal1=goal2=30;

angle\_set(yaw);

break;

case 6:

if((distance1+distance2)/2>=450)

{

yaw=0;sign=7;

distance\_sign=1;

distance=distance1=distance2=0;

}

goal1=goal2=30;

angle\_set(yaw);

break;

//越过第二个障碍

case 7:

if(distance<25&&distance!=0)

{

yaw=45;sign=8;

distance1=distance2=0;

distance\_sign=0;

}

goal1=goal2=10;

angle\_set(yaw);

break;

case 8:

if((distance1+distance2)/2>=400)

{

yaw=-45;sign=9;

distance1=distance2=0;

}

goal1=goal2=30;

angle\_set(yaw);

break;

case 9:

if((distance1+distance2)/2>=400)

{

yaw=-135;sign=10;

distance1=distance2=0;

}

goal1=goal2=30;

angle\_set(yaw);

break;

case 10:

if((distance1+distance2)/2>=400)

{

yaw=135;sign=11;

distance1=distance2=0;

}

goal1=goal2=30;

angle\_set(yaw);

break;

case 11:

if((distance1+distance2)/2>=580)

{

yaw=180;sign=12;

distance\_sign=1;

distance=distance1=distance2=0;

}

else

goal1=goal2=30;

angle\_set(yaw);

break;

//旋转越过第三个障碍，返回

case 12:

if(distance<25&&distance!=0)

{

yaw=135;sign=13;

distance1=distance2=0;

distance\_sign=0;

}

goal1=goal2=10;

angle\_set(yaw);

break;

case 13:

if((distance1+distance2)/2>=400)

{

yaw=-135;sign=14;

distance1=distance2=0;

}

goal1=goal2=30;

angle\_set(yaw);

break;

case 14:

if((distance1+distance2)/2>=450)

{

yaw=180;sign=15;

distance\_sign=1;

distance=distance1=distance2=0;

}

goal1=goal2=30;

angle\_set(yaw);

break;

//返途第一个障碍

case 15:

if(distance<25&&distance!=0)

{

yaw=-135;sign=16;

distance1=distance2=0;

distance\_sign=0;

}

goal1=goal2=10;

angle\_set(yaw);

break;

case 16:

if((distance1+distance2)/2>=400)

{

yaw=135;sign=17;

distance1=distance2=0;

}

goal1=goal2=30;

angle\_set(yaw);

break;

case 17:

if((distance1+distance2)/2>=450)

{

yaw=180;sign=18;

distance\_sign=0;

distance=distance1=distance2=0;

}

goal1=goal2=30;

angle\_set(yaw);

break;

//返途第二个障碍

case 18:

if((distance1+distance2)/2>=distance\_back-120)

{

yaw=180;

distance\_sign=0;

goal1=goal2=0;

}

else

goal1=goal2=30;

angle\_set(yaw);

break;

//返回黑色起点。

}

if(distance\_sign)

{

if((distance1+distance2)/2>10)

{

distance=get\_hc\_sr();

distance1=distance2=0;

}

}

Delay\_ms(1);

}

}