电子设计大赛技术报告

**队名：鸽子为什么这么大**

**队长： 聂 浩**

**成员： 王璞瑞**

**刘一萱**

目录

[1. 需求分析 3](#_Toc407747148)

[2. 电路设计 3](#_Toc407747149)

[1) 基于l298N的电机控制 3](#_Toc407747150)

[2) 电路设计图 4](#_Toc407747151)

[3) PCB图 4](#_Toc407747152)

[4) 实物图 5](#_Toc407747153)

[5) 传感器 5](#_Toc407747154)

[6) 单片机引脚分配 5](#_Toc407747155)

[3. 结构设计 6](#_Toc407747156)

[4. 程序实现 6](#_Toc407747157)

[1) 中断及优先级 6](#_Toc407747158)

[2) 传感器（定时器） 7](#_Toc407747159)

[3) 串口通信 8](#_Toc407747160)

[4) 基本运动 11](#_Toc407747161)

[5) 底层逻辑 13](#_Toc407747162)

[6) 策略实现 14](#_Toc407747163)

[7) 宏定义 16](#_Toc407747164)

[5. 人员分工 19](#_Toc407747165)

[6. 比赛成绩 19](#_Toc407747166)

## 需求分析

本次电子设计大赛的赛题和坑息息相关，题目要求在给定平台上，有四个状态不明的坑，在一定周期内，仅在升起来的坑内有一个得分点，在该点停5s得分，但时间-

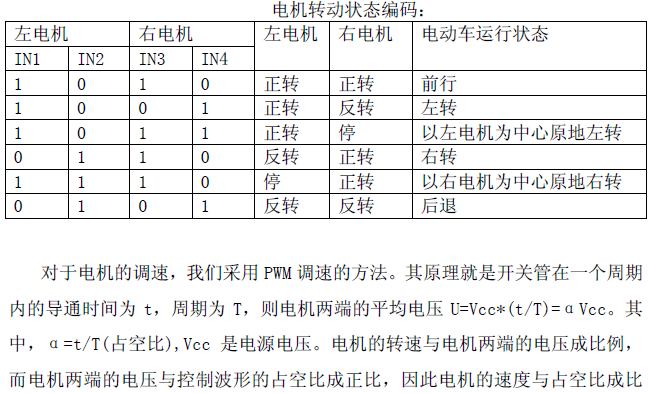
综合以上分析，需要做到的有：

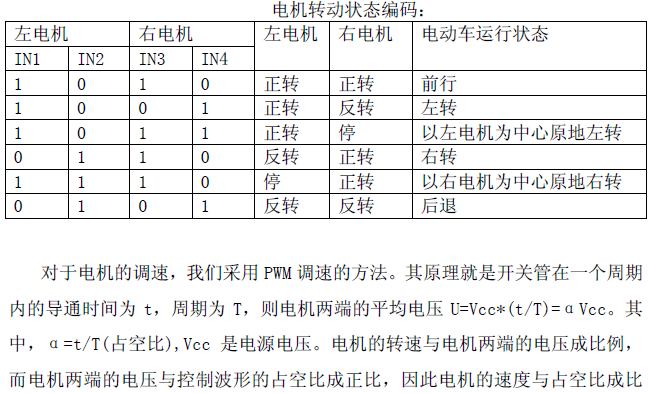
1. 利用电池，通过合适的降压稳压电路给单片机和电机驱动电路供电。
2. 设计合理的电机驱动电路，从而使用单片机来控制电机的运转。
3. 调试好通讯模块和单片机之间的通讯(UART)，以低丢包率获得平台信息。
4. 对电机的参数进行测量，使能精确的控制车的运动（直行、转向）。
5. 使用传感器，判定是否为坑。
6. 通过合理的算法，能够优雅高效遍历4个坑，并判断是否为得分点，然后判断是否进入。
7. 通过适当的优化，使得车能从合适的方向探坑并优雅的进出。
8. 因为存在30s的时间片，需要判断时间来决定是停下还是继续运动，时间片结束后所有坑的状态都改变需要注意。
9. 存在对抗。最高速下，车可以在5s内到达场上任一点。故最好考虑如何干扰对方的分和如何保住得分点。

## 电路设计

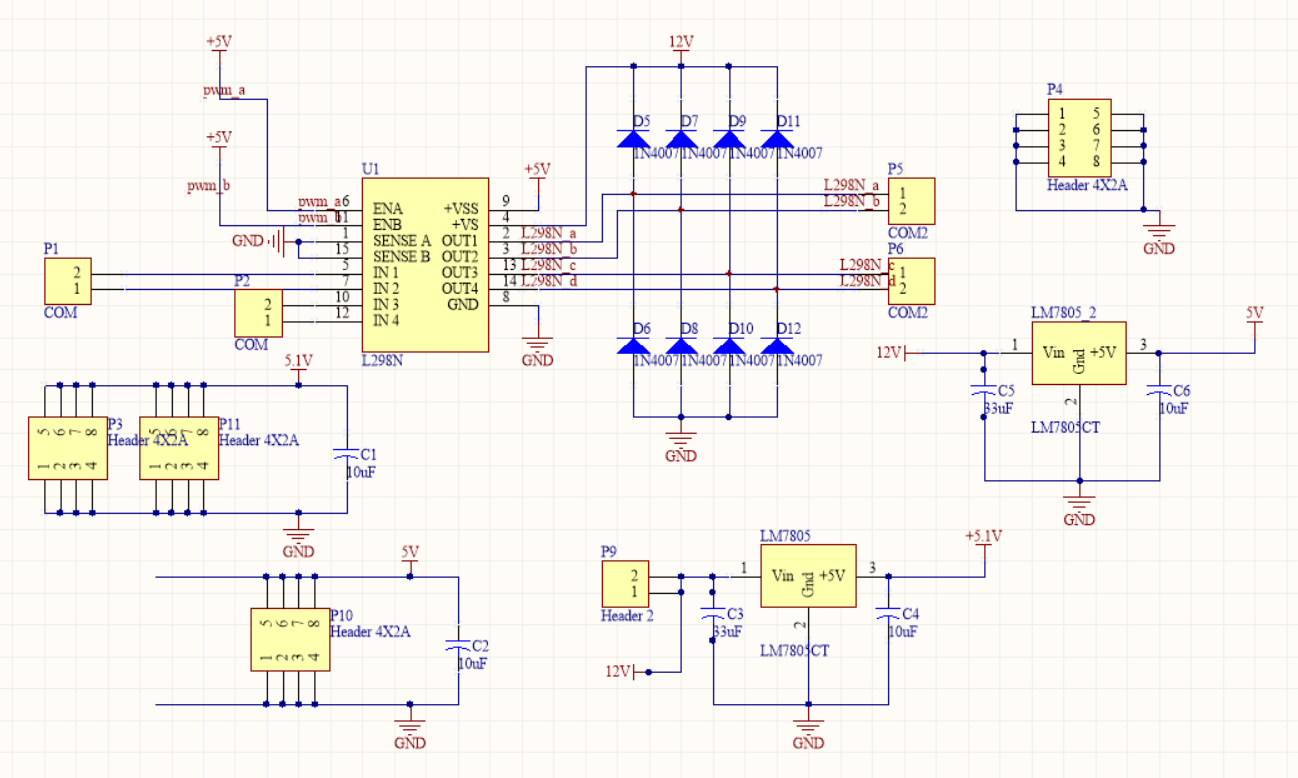
本次设计只需要电源板和驱动板，这里把他们设计在了一起。利用了组委会提供的L298N和Lm7805，使用12V锂电池组供电。

### 基于l298N的电机控制

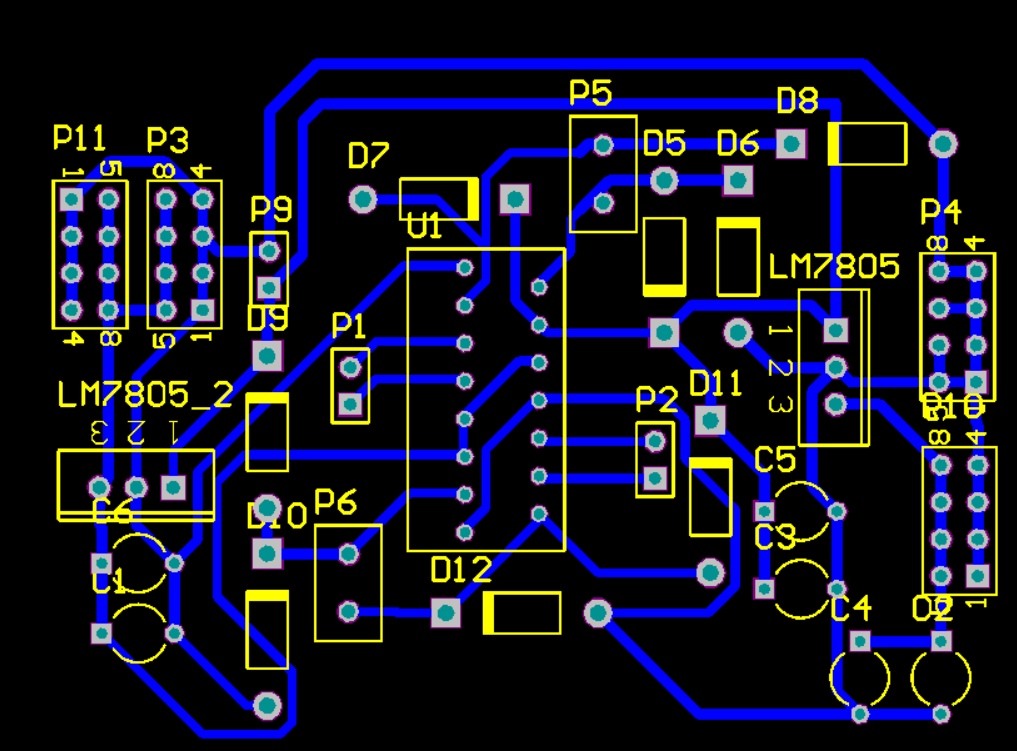




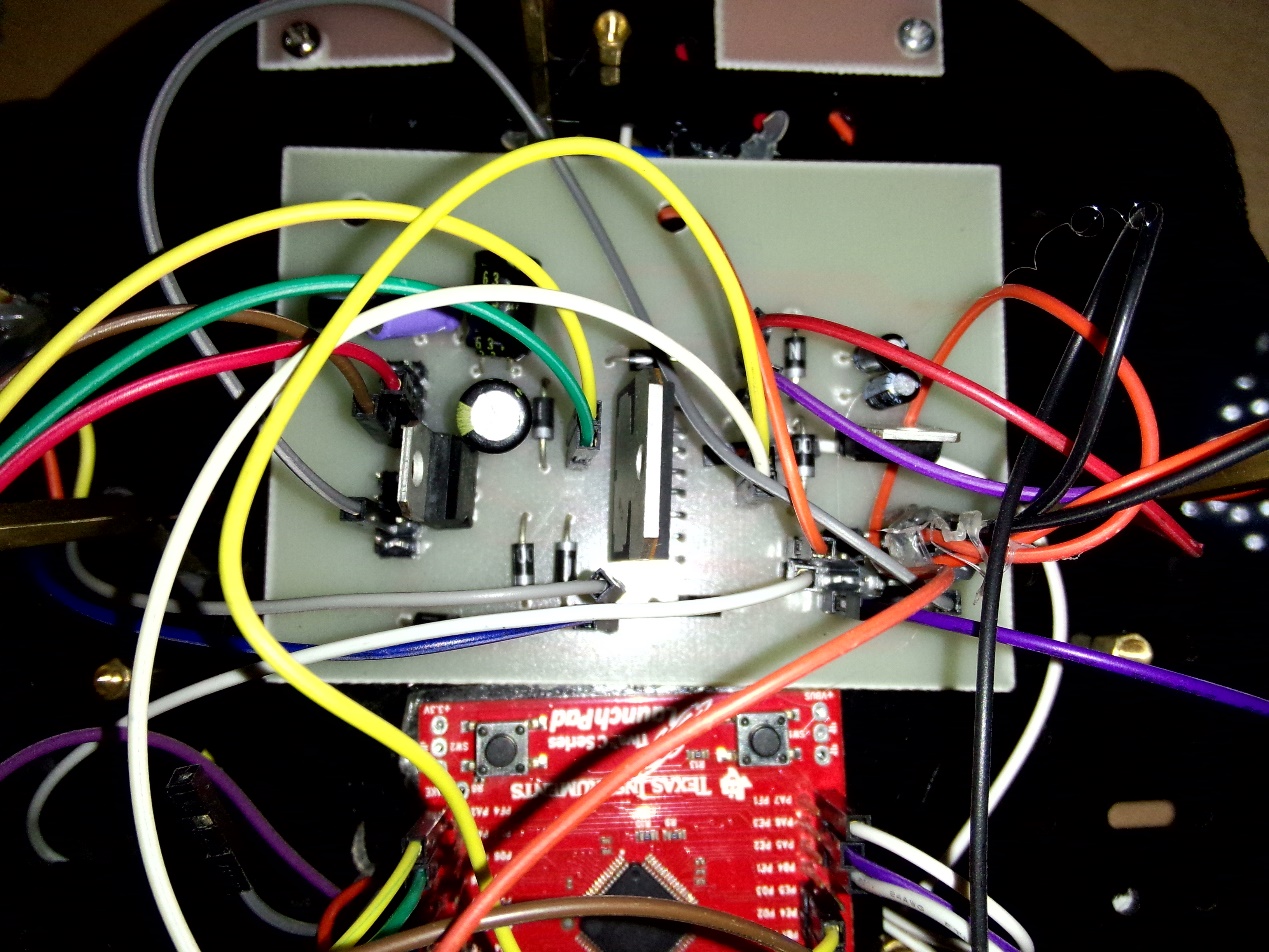
### 电路设计图



### PCB图



### 实物图



### 传感器

使用组委会提供的HC-SR04其工作方式：

1)采用IO口TRIG触发测距，给至少10us的高电平信号;

2)模块自动发送8个40khz的方波，自动检测是否有信号返回；

3)有信号返回，通过IO口ECHO输出一个高电平，高电平持续的时间就是超声波从发射到返回的时间。 试距离=(高电平时间\*声速(340M/S))/2;

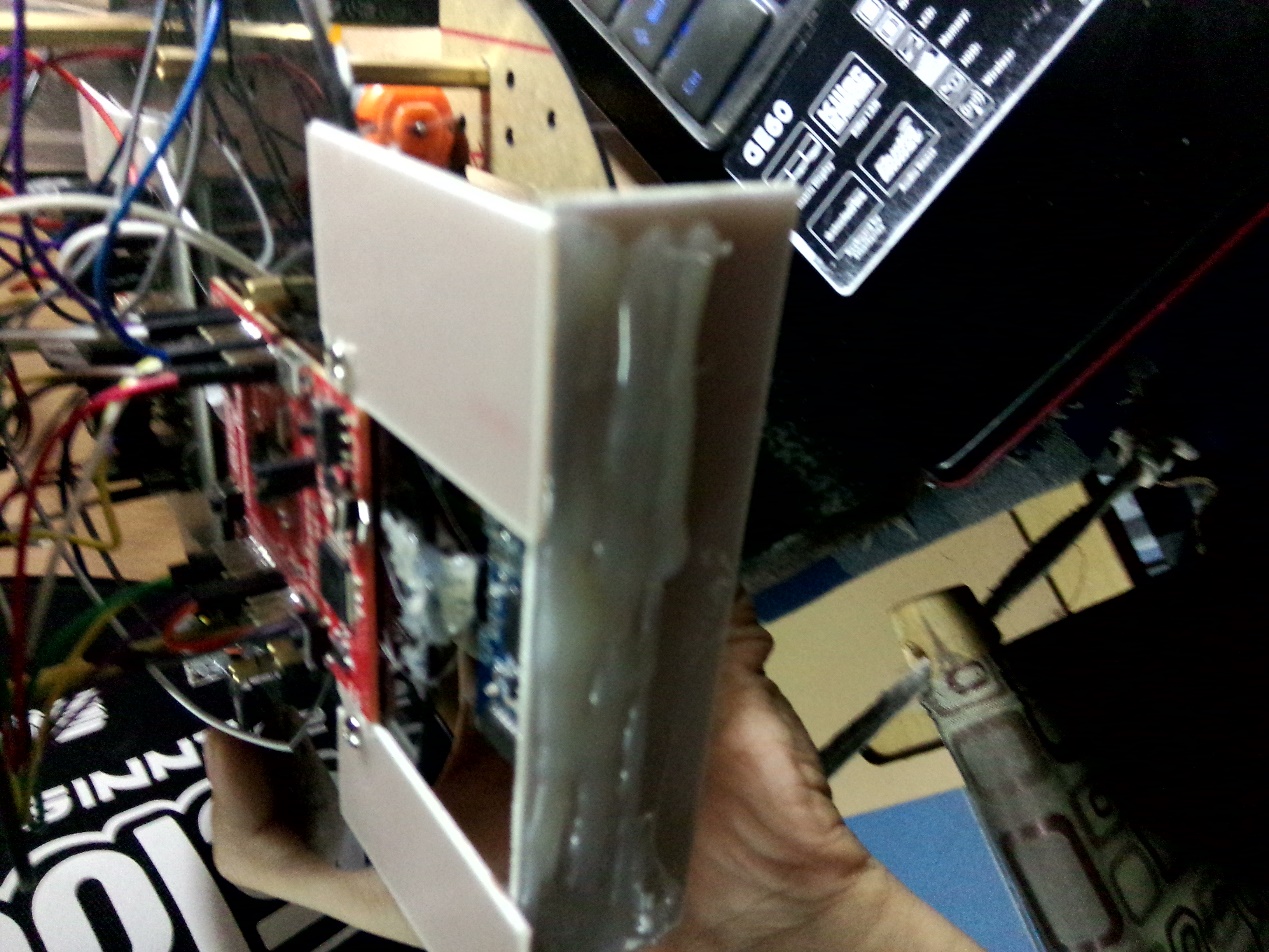
本次使用高电平触发中断的方式得到传感器返回高电平的时长，以此判断是地面还是坑。

### 单片机引脚分配

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A2 | A3 | D2 | D3 | B3 |
| 左轮控制低位 | **左轮控制高位** | **右轮控制低位** | **右轮控制高位** | **触发信号2** |
| B4 | **B5** | **B7** | **B0** |  |
| 触发信号1 | **传感器1串口** | **传感器2串口** | **通讯模块串口** |  |

## 结构设计

利用选手套件组装乘车辆，为了加固，增加了几组铜柱，使用热熔胶固定传感器和单片机以及电源驱动板。因为部分选手使用了电锯，故在传感器外部增加了PCB板制作的保护壳。



## 程序实现

本次参赛的程序代码托管在github上，可以在<https://github.com/niehao100/THEDC2014_why_is_dove_so_large>找到

### 中断及优先级

|  |  |  |  |
| --- | --- | --- | --- |
| 级别 | 函数名 | 作用 | 定义位置 |
| 0 | UARTIntHandler | 通信模块的串口 | commu.c |
| 1 | Timer0IntHandler | 定时器中断，每一段时间  给传感器一个触发信号 | timer.c |
| 2 | GPIOA6INT | 传感器中断,高电平触发，记录高电平时长在全局变量t中 | sensors.c |
| 3 | UART0IntHandler | usb模块通信串口，用于调试 | commu.c |

### 传感器（定时器）

通过或运算读取两个传感器的数据，通过标志位(sensor\_flag timer.c 里更新)区分到底是哪个传感器的信号。

具体代码程序：

void GPIOA6INT(void){

GPIOIntClear(INT\_GPIOA, GPIO\_PIN\_6|GPIO\_PIN\_5|GPIO\_PIN\_7);

IntDisable(INT\_GPIOA);

for(t\_s=0;t\_s<10000&&(GPIOPinRead(GPIO\_PORTA\_BASE,GPIO\_PIN\_5)||GPIOPinRead(GPIO\_PORTA\_BASE,GPIO\_PIN\_7));t\_s++);

if(((senser\_flag%2)==0)&&t\_s>450) {for(i=0;i<150;i++) move\_stright(-4);if((abs(abs(center\_local[0]-128)-(128-\_Px))<40)&&(abs(abs(center\_local[1]-128)-(128-\_Py))<40)) S\_flag--;}

if(((senser\_flag%2)==1)&&t\_s>310) {for(i=0;i<150;i++) move\_stright(4);if((abs(abs(center\_local[0]-128)-(128-\_Px))<40)&&(abs(abs(center\_local[1]-128)-(128-\_Py))<40)) S\_flag--;}/\*

if((senser\_flag%3)==2) {for(i=0;i<50;i++) turn\_left\_back(); S\_flag=1;}\*/

if(S\_flag<-1000) S\_flag=-10;

IntEnable(INT\_GPIOA);

}

void Timer0IntHandler(void)

{

ROM\_TimerIntClear(TIMER0\_BASE,TIMER\_TIMA\_TIMEOUT);

HWREGBITW(&g\_ui32Flags, 0) ^= 1;

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1, g\_ui32Flags << 1);

GPIOPinWrite(GPIO\_PORTF\_BASE,GPIO\_PIN\_1,0x2);

switch (senser\_flag%2){

case 0:

GPIOPinWrite(GPIO\_PORTB\_BASE,GPIO\_PIN\_3,0x8);

SysCtlDelay(SysCtlClockGet()/5000);//0.01s

GPIOPinWrite(GPIO\_PORTB\_BASE,GPIO\_PIN\_3,0);

break;

case 1:

GPIOPinWrite(GPIO\_PORTC\_BASE,GPIO\_PIN\_4,0x10);

SysCtlDelay(SysCtlClockGet()/5000);//0.01s

GPIOPinWrite(GPIO\_PORTC\_BASE,GPIO\_PIN\_4,0);

break;

GPIOPinWrite(GPIO\_PORTF\_BASE,GPIO\_PIN\_1,0);

if((senser\_flag++)>9998) senser\_flag=0;

}

### 串口通信

和通讯模块的通信：

把通过串口得到的数据写入循环队列，利用校验位，当最后一个读入是0x0A，前一位0x0D，之前26位也是0x0A则认为将它和之前的26位是一段正确的数据并将其写入Position数组。

后面是对逻辑的处理，通过各个数据来改变标志位来影响逻辑的执行。

enemyflag是敌人的编号。

void UARTIntHandler(void)

{

uint32\_t ui32Status;

int j=0,k=0;

ui32Status = ROM\_UARTIntStatus(UART1\_BASE, true);

ROM\_UARTIntClear(UART1\_BASE, ui32Status);

while(ROM\_UARTCharsAvail(UART1\_BASE))

{

queue.i++;if(queue.i==50) queue.i=0;

queue.q[queue.i]=UARTCharGet(UART1\_BASE);

}

if(queue.q[queue.i]==0x0A){

IntMasterDisable();

if(queue.i>26) {

if(queue.q[queue.i-1]==0x0D&&queue.q[queue.i-26]==0x0A&&queue.q[queue.i-27]==0x0D){

for(k=0,j=queue.i-25;k<26;k++){

Position[k]=queue.q[j];

j++;

}

head\_local[0]=Position[1+5\*Position[0]];

head\_local[1]=Position[2+5\*Position[0]];

tail\_local[0]=Position[3+5\*Position[0]];

tail\_local[1]=Position[4+5\*Position[0]];

center\_local[0]=(int)((head\_local[0]+tail\_local[0])/2);center\_local[1]=(int)((head\_local[1]+tail\_local[1])/2);

UART\_flag++; if(UART\_flag==10000) UART\_flag=0;

}

IntMasterEnable();

}

else {

if((queue.q[(queue.i==0)?49:(queue.i-1)]==0x0D)&&(queue.q[queue.i+24]==0x0A)){

IntMasterDisable();

for(k=0,j=(((queue.i-25)>=0)?(queue.i-25):(queue.i+25));k<26;k++){

Position[k]=queue.q[j];

j++;if(j==50) j=0;

}

head\_local[0]=Position[1+5\*Position[0]];

head\_local[1]=Position[2+5\*Position[0]];

tail\_local[0]=Position[3+5\*Position[0]];

tail\_local[1]=Position[4+5\*Position[0]];

center\_local[0]=(int)((head\_local[0]+tail\_local[0])/2);center\_local[1]=(int)((head\_local[1]+tail\_local[1])/2);

UART\_flag++; if(UART\_flag==10000) UART\_flag=0;

IntMasterEnable();

}

//回朔

}

// UARTprintf("%c",flag);

//for(j=0;j<26;j++)

}

score=Position[5+5\*enemy\_flag]+Position[5+5\*Position[0]];

if(Position[22]==time) {for(j=0;j<4;j++) check[j]=1;time=Position[22]-30;score\_change=score;}

//if(Position[22]<5) exit(0);

switch(Position[21]){

case 0x08: if((abs(center\_local[0]-\_Px)<13)&&(abs(center\_local[1]-\_Py)<13)) {for(j=0;j<4;j++) check[j]=1;}check[0]=0;break;

case 0x04: if((abs(center\_local[0]-255+\_Px)<13)&&(abs(center\_local[1]-\_Py)<13)) {for(j=0;j<4;j++) check[j]=1;}check[1]=0;break;

case 0x02: if((abs(center\_local[0]-\_Px)<13)&&(abs(center\_local[1]-255+\_Py)<13)) {for(j=0;j<4;j++) check[j]=1;}check[3]=0;break;

case 0x01: if((abs(center\_local[0]-255+\_Px)<13)&&(abs(center\_local[1]-255+\_Py)<13)) {for(j=0;j<4;j++) check[j]=1;}check[2]=0;break;

}

if((abs(abs(Position[5\*enemy\_flag+1]-128)-(128-\_Px))<10)&&(abs(abs(Position[5\*enemy\_flag+2]-128)-(128-\_Py))<10)){

check[(int)(Place(Position[5\*enemy\_flag+1],Position[5\*enemy\_flag+2])/10-1)]=0;

}

IntMasterEnable();

}

### 基本运动

利用PWM控制轮子的转速，主要有以下函数。

|  |  |
| --- | --- |
| void left\_move(int v) | 用来控制左轮的运动。速度正负表示前后 |
| void right\_move(int v) | 用来控制右轮的运动 |
| 通过对这两个函数内部v与占空比关系的调整来消除两轮之间的不对称性。 | |
| void move\_stright(int v) | 用来让小车直走的函数，此时小车左右轮的速度相等 |
|  |  |
| void turn\_left() | 实现小车的左转，此时右轮向前，左轮向后就可以实现左转 |
| void turn\_right() | 实现小车的右转，此时右轮向后，左轮向左就可以实现右转 |
| void turn\_left\_back() | 左后转 ，如果前方有坑则不能向前转则需用到此函数，左轮后退。 |
| void turn\_right\_back() | 右后转，与左后转的原理相同 |
| int head(unsigned char x,unsigned char y) | 朝着目的地的坐标值（x，y）,利用了PI控制（利用平台信息），代码下附 |
| void move\_to(unsigned char x,unsigned char y) | 朝着坐标值（x，y）的大致方向前进，当与之距离20左右时返回 |
| void move\_to\_exact(unsigned char x,unsigned char y) | 朝着坐标值（x，y）的准确方向前进，当与之距离5左右时停止。 |

int head(unsigned char x,unsigned char y){

double a=0,angel=0,sum=0,d=0;

int i=0,t=0,v=0;

head\_begin:

SysCtlDelay(SysCtlClockGet()/3000);

if(t==UART\_flag) goto head\_begin;

t=UART\_flag;

if(Position[25]!=0x0A) goto head\_begin;

if(Position[23]!=1) return 0;

if((abs(x-center\_local[0])<5)&&(abs(y-center\_local[1])<5)) {

return 1;

}

IntMasterDisable();

a=atan2((double)(head\_local[1]-tail\_local[1]),(double)(head\_local[0]-tail\_local[0]));

angel=atan2((double)(y-center\_local[1]),(double)(x-center\_local[0]));

angel=(angel>=0)?angel:(angel+2\*3.1415926);

a=(a>=0)?a:(a+2\*3.1415926);

if((angel-a)>=3.1415926){angel=-2\*3.1415926+(angel-a);}

else {if((a-angel)>=3.1415926)

angel=2\*3.1415926-(-angel+a);

else angel-=a;

}

sum+=angel;

if(sum>200000) sum=0;

IntMasterEnable();

//PID控制

v=(int)(4.0\*angel+0.30\*sum);//+8\*(angel-d));

d=angel;

if(angel>=0.0872){

for(i=0;i<abs(v)+4;i++) turn\_right();

goto head\_begin;

}

if(angel<=-0.0872){

for(i=0;i<abs(v)+4;i++) turn\_left();

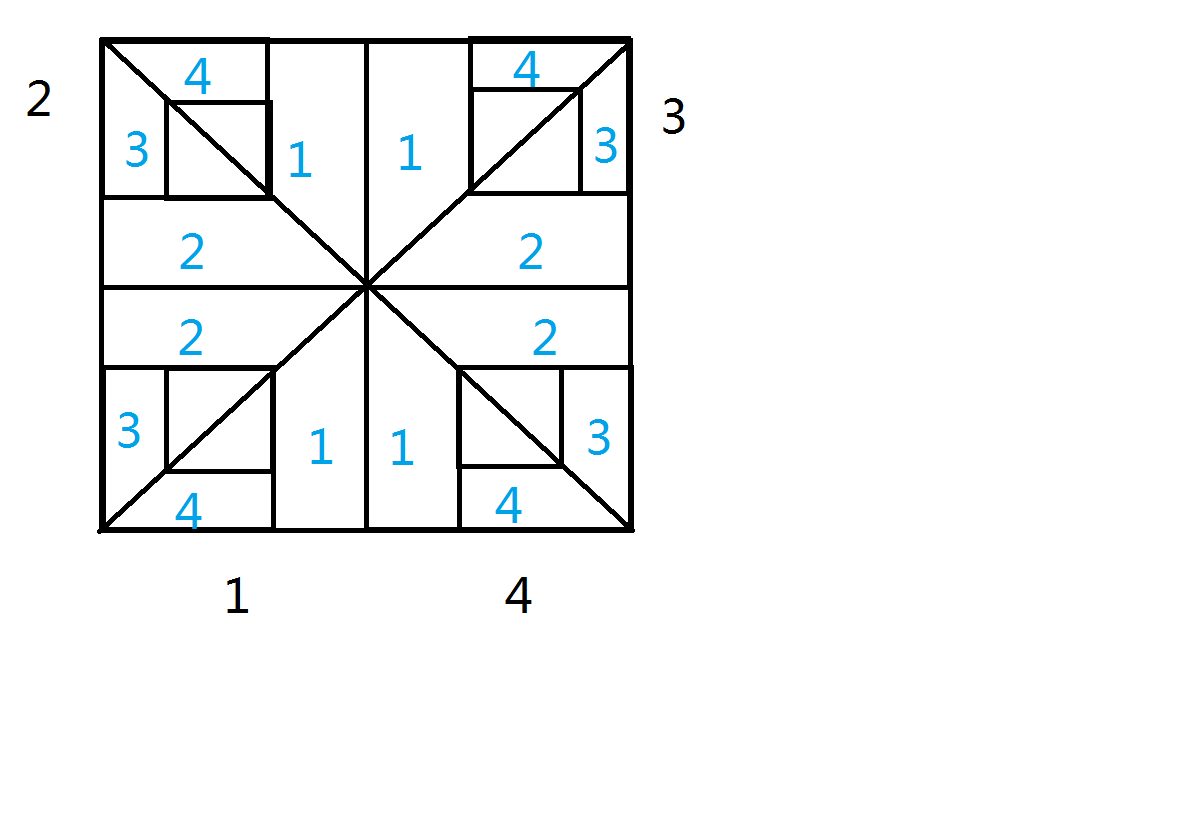
goto head\_begin;

}

}

### 底层逻辑

由于小车所用传感器很少的这一局限性我们必须将小车移动到4个坑直角边才能正确探测并且避免被卡住。所以把平台划分为16个区域，目的地有四个坑，共有64种情况。分为64种情况进行讨论。Place函数用来确定自己所在哪个区域并返回区域所在的值。有（1,1）一直到（4,4）16种情况，由高层逻辑判断应该前往那个坑去得分，



|  |  |
| --- | --- |
| void beat\_enemy | 转向对方并最高速前进 |
| void get\_score | 转向坑并走进去，坐标发生变化则试图回复到得分点，出现不能得分情况时立即退出坑。 |
| void target1(int judge) | 前往一号坑试图得分 |
| void target2(int judge) | 前往二号坑试图得分 |
| void target3(int judge) | 前往四号坑试图得分 |
| void target4(int judge) | 前往三号坑试图得分 |
| int Place(unsigned char x,unsigned char y) | 确定自己的位置 |

### 策略实现

Main函数中有一个长度为4的指针数组，4个元素分别是target1，target2，target3，target4的指针。

通过check数组判断得分坑来调用4个得分函数

每30秒初将check都置为1，尽量选择最短的边让小车去探测坑，并且在距一个30秒结束只有7秒时将在坑外的小车静止不动，在坑内的小车就赶紧往出跑（通过标志位，如果处于该情况，迅速中断除了离开坑的一切函数，这些函数中都写了遇到该标志退出的条件）。

**int** main(**void**)

{

**void** (\*target[4])(**int** judge)={&target1,&target2,&target3,&target4};

**int** i=0,x=0,y=0,k=0;

**unsigned** **char** place=0;

Init();

begin:

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 0x08);

**for**(;;){

SysCtlDelay(1600);

**if**(Position[23]==1) **break**;

}

**if**(Position[0]==0) enemy\_flag=1;

**else** enemy\_flag=0;

time=Position[22]-30;

**for**(;;){

**if**((Position[22]<(time+5))

&&((**abs**(**abs**(head\_local[0]-128)-(128-\_Px))>35

||(**abs**(**abs**(head\_local[1]-128)-(128-\_Py))>35)))){**continue**;}

**if**(((check[0]==0)&&(check[1]==0)&&(check[2]==0)&&(check[3]==0)))

{**for**(i=0;i<4;i++) check[i]=1;}

place=Place(Position[1+5\*Position[0]],Position[2+5\*Position[0]]);

p=(**int**)(place/10)-1;

S\_flag=1;

**if**(check[p]) {target[p](place);check[p]=0;}

**else**{

**if**(check[(p==3)?0:(p+1)]){ target[(p==3)?0:(p+1)](place);check[(p==3)?0:(p+1)]=0;**continue**;}

**if**(check[(p==0)?3:(p-1)]){ target[(p==0)?3:(p-1)](place);check[(p==0)?3:(p-1)]=0;**continue**;}

**if**(check[(p>1)?(p-2):(p+2)]){ target[(p>1)?(p-2):(p+2)](place); check[(p>1)?(p-2):(p+2)]=0;**continue**;}

}

**goto** begin;

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1|GPIO\_PIN\_2|GPIO\_PIN\_3, 0x08);

**return** 0;

### 宏定义

为了避免函数参数的混乱，本程序中使用了大量的全局变量，故在thedc.h里进行了设置。

#ifndef THEDC\_H\_

#define THEDC\_H\_

#include <stdint.h>

#include <stdbool.h>

#include <stdlib.h>

#include "inc/hw\_ints.h"

#include "inc/hw\_memmap.h"

#include "inc/hw\_types.h"

#include "driverlib/debug.h"

#include "driverlib/fpu.h"

#include "driverlib/gpio.h"

#include "driverlib/interrupt.h"

#include "driverlib/pin\_map.h"

#include "driverlib/rom.h"

#include "driverlib/sysctl.h"

#include "driverlib/timer.h"

#include "driverlib/uart.h"

#include "C:/ti/TivaWare\_C\_Series-2.0.1.11577/driverlib/uartstdio.h"

extern unsigned char enemy\_flag,time;

extern char flag;

extern int t;

extern struct Player info;

extern struct Queue queue;

extern unsigned char Position[26];

extern int head\_local[2],tail\_local[2],center\_local[2];

extern int S\_flag;//传感器状态

extern int UART\_flag;

extern char senser\_flag;//判断是那个传感器

extern int check[4]; //有效坑判断

extern unsigned char score,score\_change,enemy\_flag,time,p;

//move

#define ahead 0x4

#define back 0x8

#define stop 0xc

void move\_stright(int);

void left\_move();

void right\_move();

void turn\_left();

void turn\_right();

void turn\_left\_back();

void turn\_right\_back();

int head(unsigned char,unsigned char);

void move\_to(unsigned char,unsigned char);

void move\_to\_exact(unsigned char x,unsigned char y);

//通信函数

void UARTSend(const uint8\_t \*, uint32\_t );

void UARTConfig();

void UARTIntHandler(void);

void UART0IntHandler(void);

void move\_to\_exact(unsigned char x,unsigned char y);

//传感器

void Timer0IntHandler(void);

void TimerConfig();

void GPIOA6INT(void);

//逻辑

void get\_score(int x,int y);

#define \_Px 57

#define \_Py 60

void target1(int judge);

void target2(int judge);

void target3(int judge);

void target4(int judge);

int Place(unsigned char x,unsigned char y);

#endif /\* THEDC\_H\_ \*/

## 人员分工

|  |  |
| --- | --- |
| 队长 | 聂浩 |
| 成员 | 王璞瑞 刘一萱 |
| 总体设计 | 聂浩 |
| PCB设计与制版 | 王璞瑞 聂浩 |
| 焊接 | 王璞瑞 刘一萱 |
| 代码编写 | 聂浩 王璞瑞 |
| 调试 | 聂浩 |
| 报告撰写 | 刘一萱 聂浩 |

## 比赛成绩

于100多个队伍中脱颖而出进入决赛，小组赛一胜两负未出线。