第七届“飞思卡尔”杯全国大学生
智能汽车竞赛

技 术 报 告

学　　校：　　北京邮电大学

队伍名称：　　金色量子队

参赛队员：　　陈小川 张子文 赵佳瑜

带队教师：　　高荔

关于技术报告和研究论文使用授权的说明

本人完全了解第七届“飞思卡尔”杯全国大学生智能汽车竞赛关保留、使用技术报告和研究论文的规定，即：参赛作品著作权归参赛者本人，比赛组委会和飞思卡尔半导体公司可以在相关主页上收录并公开参赛作品的设计方案、技术报告以及参赛模型车的视频、图像资料，并将相关内容编纂收录在组委会出版论文集中。

参赛队员签名：_____

带队教师签名：_____

日　　期：_____

第一章 引言

现在半导体在汽车中的应用原来越普及，汽车的电子化已成为行业发展的必然趋势。它包括了汽车电子控制装置，即通过电子装置控制汽车发动机、底盘、车身、制动防抱死及动力转向系统等，到车载汽车电子装置，即汽车信息娱乐系统、导航系统、汽车音响及车载通信系统等等，几乎涵盖了汽车的所有系统。汽车电子的迅猛发展必将满足人们逐步增长的对于安全、节能、环保以及智能化和信息化的需求。

作为全球最大的汽车电子半导体供应商，飞思卡尔公司一直致力于为汽车电子系统提供全范围应用的单片机、模拟器件和传感器等器件产品和解决方案。飞思卡尔公司在汽车电子的半导体器件市场拥有领先的地位并不断赢得客户的认可和信任。其中在 8 位、16 位及 32 位汽车微控制器的市场占有率居于全球第一。飞思卡尔公司生产的 S12 是一个非常成功的芯片系列，在全球以及中国范围内被广泛应用于各种汽车电子应用中。例如引擎管理、安全气囊、车身电子、汽车网络和资讯娱乐等。

1.1 概述

“飞思卡尔”杯全国大学生智能汽车邀请赛今年进行到第七届，多年经验的积累使得比赛形式丰富，比赛规则比较完善，为广大同学提供了一个良好的学习提高的平台。第七届大赛中，为了拓展赛车的形式和赛道检测形式，规则较上一届有所变化。摄像头和光电组别的赛道检测从中线改变为了检测边线，而电磁组更是有了较大的改变，要求赛车直立行走。

为响应教育部关于加强大学生的创新意识、合作精神和创新能力的培养的号召，我们学校组队并积极参加了这一届“飞思卡尔”杯全国大学生智能汽车邀请赛电磁组的比赛。从 2011 年 11 月开始着手进行准备，历时近 10 个月。鉴于本届电磁组与光电、摄像头的赛车有着本质的区别，即电磁组需要直立行驶，这就需要结合陀螺仪和加速度计模块实现闭环控制保持车身姿态；此外在检测方法上，我们在传感器设计上仍然采用了较为传统的电感线圈来检测赛道磁场，利用所获取的信号进行处理，实现对赛车转向进行控制。

同时，我们利用了前几届比赛积累下的经验，继续加强在电源管理、噪声抑制、驱动优化、整车布局等方面的研究工作，使智能车能够满足高速运行下的动力性和稳定性需求，获得了良好的综合性能和赛场表现。

本技术报告将针对我们的传感器信号处理设计安装、机械结构、电路设计、K60 控制软件主要理论、控制算法等方面进行阐述，并列出了模型车的主要技术参数。

1.2 整车设计思路

1.2.1 控制系统

智能车的整体结构如图 1 所示：以左右两电感作为电磁传感器获取赛道某点电磁特性，信号输入到 K60 控制核心，进行进一步处理用以控制赛车的转向；通过陀螺仪和加速度计模块融合可以获取车身的角度用以对车身姿态进行闭环控制来实现智能车的直立；通过光电编码器转速传感器检测车速，并采用 K60 的输入捕捉功能进行脉冲计数计算车速来实现速度闭环控制达到稳定的速度；电机转速控制采用 PID 控制，通过 PWM 控制驱动电路调整电机的功率。

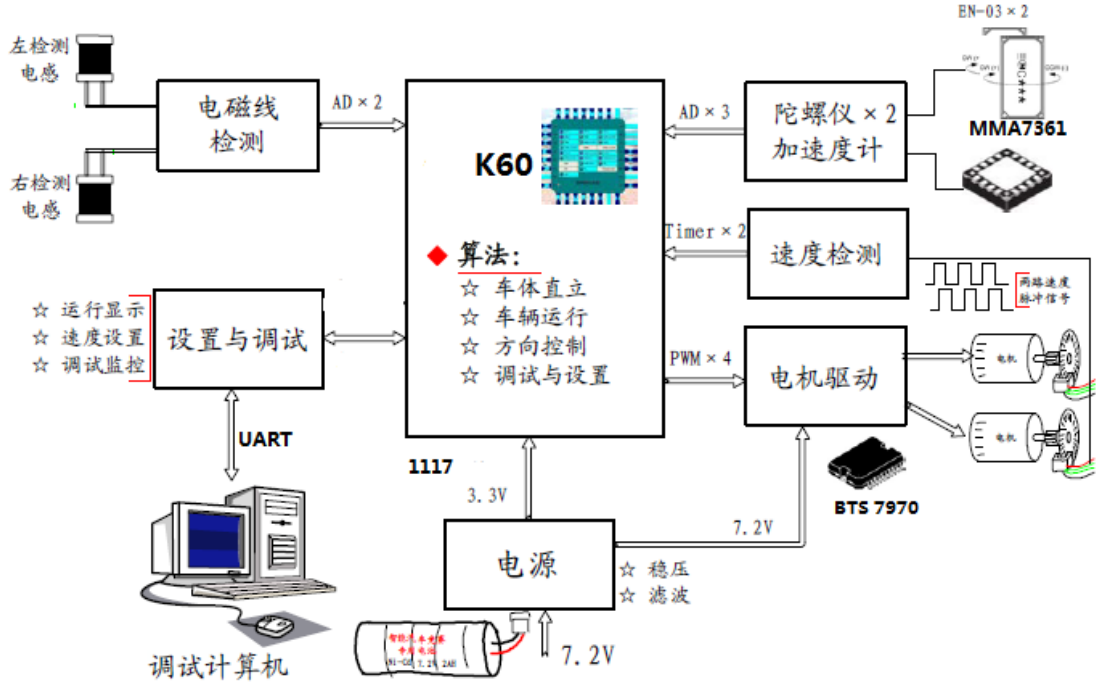

图 1 智能车整体结构图

1.2.2 整车布局

鉴于今年电磁赛车和赛道规则的特点，在整车布局上布局的思路有所改变。我们采用了低重心紧贴型车身的设计，使整车中质量最大的部分电池紧贴车底以最大限度降低车体的整体重心。由于车为直立姿态行进，为了使得车在转动时相对灵活，需要采用碳杆向前支出电磁传感器，以尽量减小传感器支撑结构的转动惯量。整体布局如下图所示。

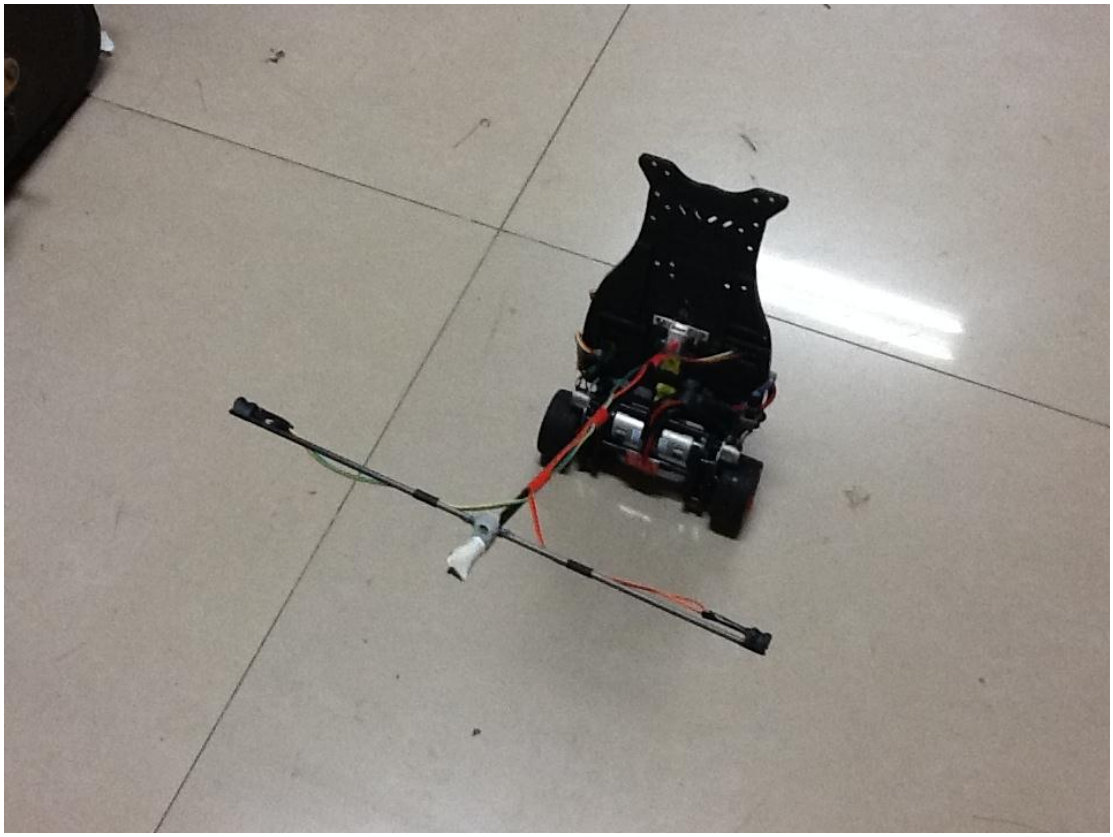

图 2 车身整体布局图

1.2.3 电磁组主要特点

1．车身直立姿态

今年的电磁组别的规则有了较大的变动，即要求车身保持直立（即车体仅两个动力轮着地）的情况下行进。为了实现车身姿态的精确控制，这就需要采用较以前更多类型的传感器来获取多种车体参数。利用这些参数实现对于车身姿态的闭环控制，这样方能够达到对车身姿态的稳定控制。由于电磁车需要直立行进，故直立是第一步，做好了直立姿态的控制是迈向成功的重要一步。故我们花费了大量的时间和精力研究改进神的整体结构、车身的姿态控制算法和参数来实现这点。

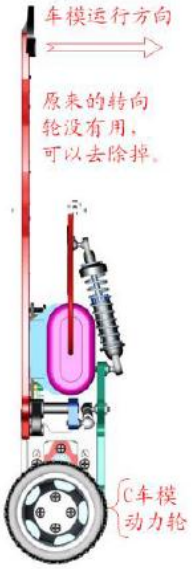

图 3 电磁组车模运行状态

2．车速控制的稳定性

直立行进的电磁车不比往届的四轮模式，很能在保持车体姿态稳定性的同时做出短时大幅度的速度调整，这就需要对于车速控制的较高稳定性。在弯道有明显的加减速这是这届大多数电磁车的一个通病。这可能由于几个方面的因素导致。首先要确定测速码盘的准确和在平地上的控速稳定，若无问题则大多情况是陀螺仪的安放不水平于地面需要微调。在不断的摸索中我们可以较好地保证车模的速度控制稳定性。

3. 传感器信号为模拟值

电磁组需要检测的信号为大小 100mA，频率为 20KHz 的方波信号，赛道由导线铺成，导线周围分布着交变的电磁场，由于赛道的各种形状，使得磁场发生叠加，不同的赛道形状形成不同的特征磁场。赛道信息相对于传统黑白线具有信号可以提供模拟信息的优势，我们利用电磁赛道这种优势，完善小车控制算法，达到了较好的控制效果。

第二章 硬件设计

2.1 传感器方案

2.1.1 磁场传感器

使用 10mH 电感和 6.8nF 电容并联谐振，来感应 20KHz 的磁场信号，经放大电路放大后，得到正弦波，再用 AD 采样，得到正弦波的峰值，以判断传感器离导线的距离，从而定位导线。由于官方给出的三极管放大电路不易调节放大倍数，检波电路信号变化速度较慢，我们决定使用运放放大直接由 AD 采样。使用 max4489 双运放，可以满足探测要求。

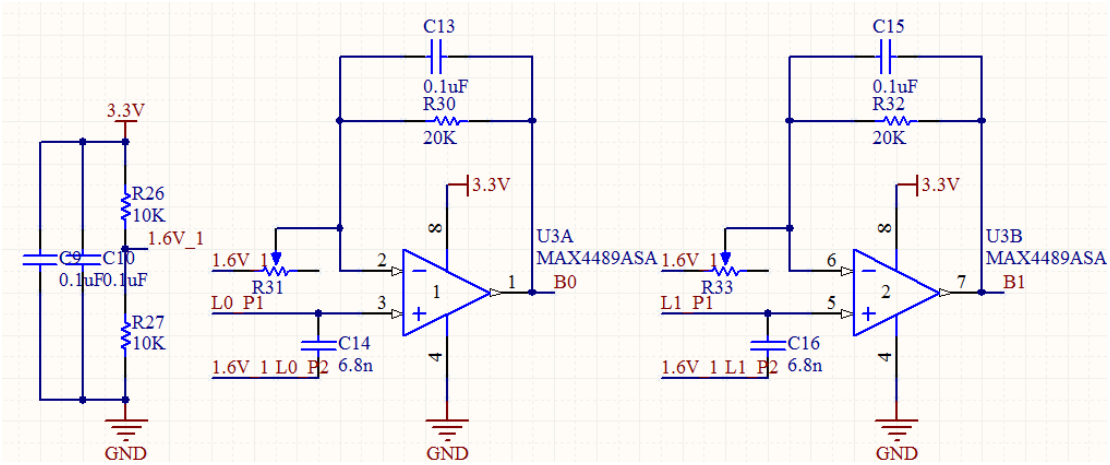

图 2-1 磁场传感器放大

2.1.2 磁场传感器布局

电磁传感器测出的信号为当前所在位置的某个方向的磁场信息，所以传感器的布局至关重要。通过实验发现，当传感器相距较大，视角宽，得到的赛道信息量大。由于今年电磁车改为直立行走，速度明显降低，因此我们采用最普通的两个传感器水平放置的方法，一样可以满足要求。

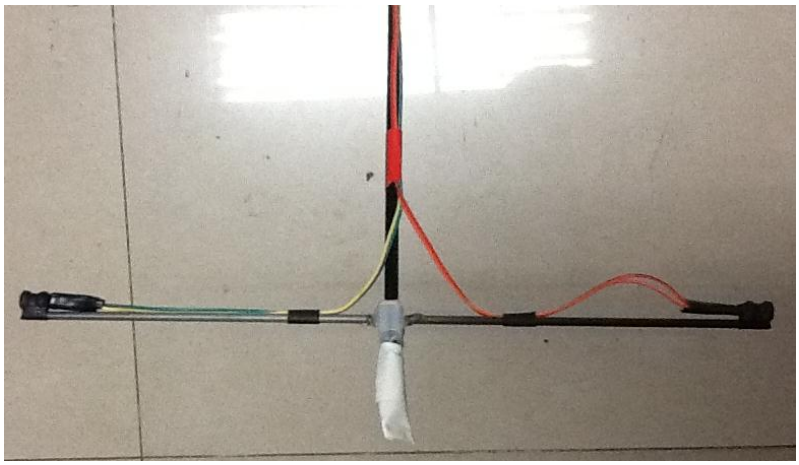

图 2-2 磁场传感器布局

2.1.3 加速度计传感器

加速度计传感器可以采集车体的姿态，为让车体直立起来，我们需要实时得到车体姿态，不断的调整姿态，使其保持直立。加速度计采集其于每个轴上的重力分量大小，因此可以数学力分解可以得知当前车模姿态。但是加速度计在车模运动起来后会存在巨大噪声，因此不能单独使用。我们选用的是龙丘 MMA7361 加速度计模块。

2.1.4 陀螺仪传感器

我们使用的是官方提供的陀螺仪方案，陀螺仪反应在转轴方向的角速度大小，但是出来的信号比较小，因此需要经过一个运放将其放大，然后使用 AD 读取，然后处理。

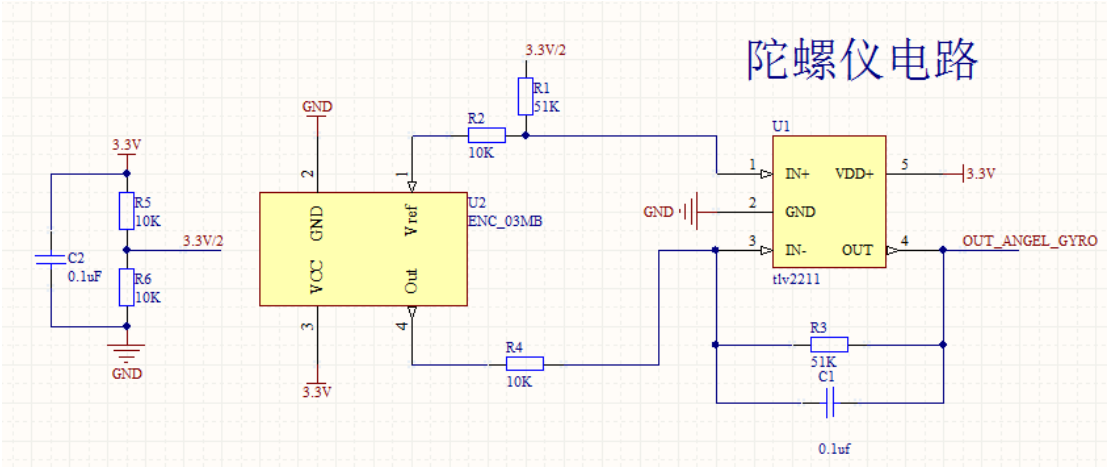

图 2-3 陀螺仪电路

2.2 主板电路设计

2.1.1 电源模块

5V 提供给编码器和 BTS7970 使用，3.3V 提供给单片机以及运放使用。

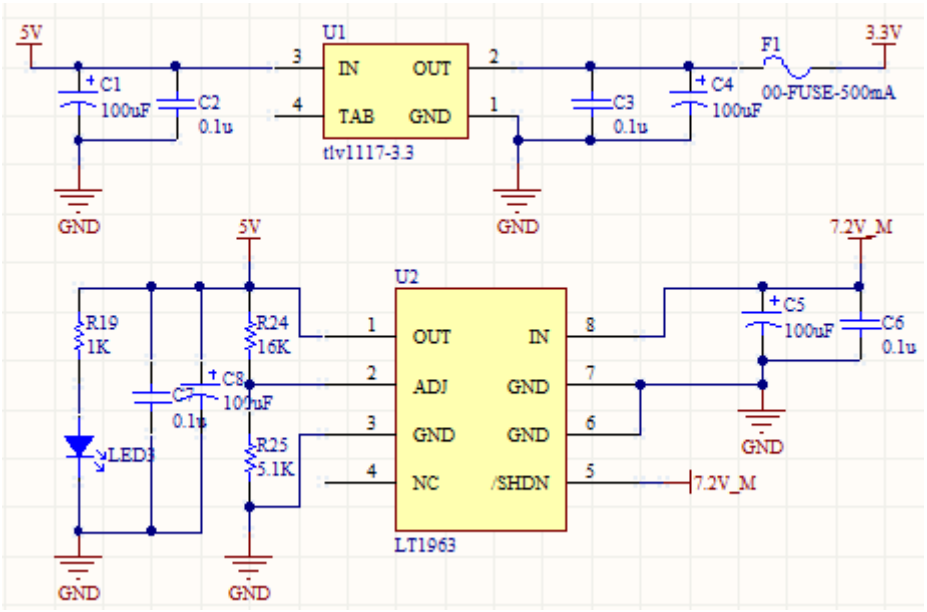

图 2-4 电源模块设计

2.2.2 驱动模块

电机驱动使用了 4 块 BTS7070，电路图如下。

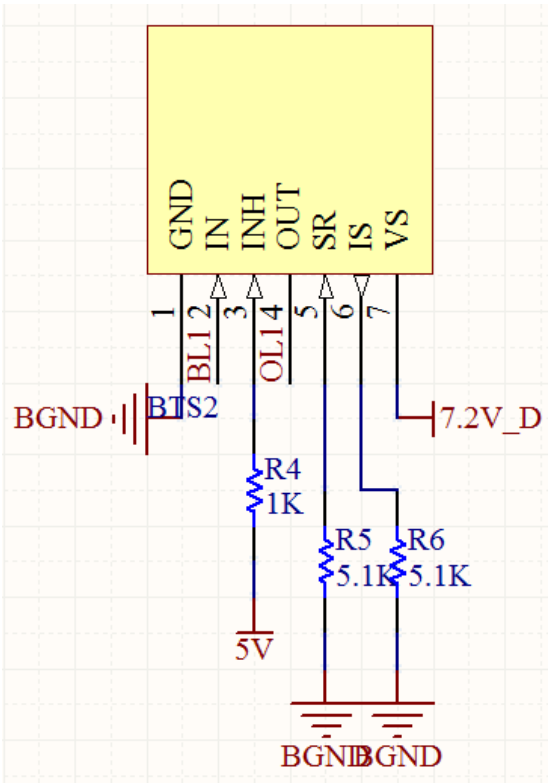

图 2-5　BTS7970 电路

2.2.3 各部分接口

包括最小系统接口、串口、编码器、加速陀螺仪模块。

2.3 最小系统板

我们使用的是龙丘的 K60 最小系统板。

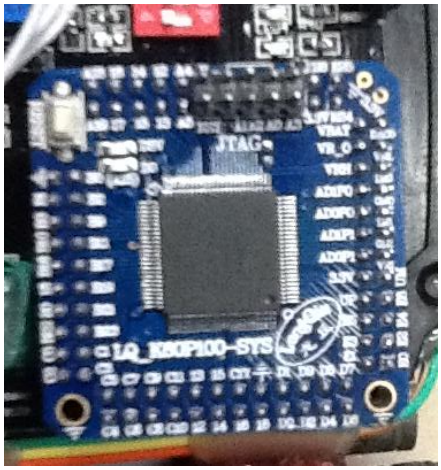

图 2-6 K60 最小系统板

第三章 机械设计

车模的机械部分的设计对车模的行驶性能影响很大，同等控制能力的情况下，机械的性能会直接影响比赛的成绩，所以这部分需要精心地调教。虽然比赛规则限制了对车模的改装，但是仍有许多可以注意的地方。

3.1 电磁传感器安装

电磁传感器测出的信号为当前所在位置的某个方向的磁场信息，所以传感器的布局至关重要。通过实验发现，当传感器相距较大，视角宽，得到的赛道信息量大。所以，采用

尽量架宽主要传感器的方式，以获得丰富的赛道信息，并且提前预知赛道形状。

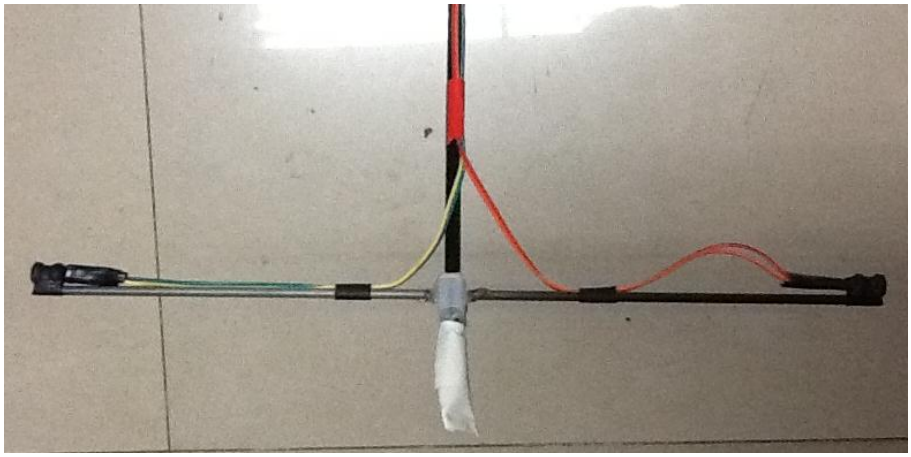

图 3-1 电磁传感器安装

3.2 加速度计传感器安装

角度传感器安装位置直接影响小车直立的效果，传感器尽量安装在车体重心的位置，及防止了放在高处由于车模震动带来的测值不准，又避免了安装位置过低带来的反应不够灵敏。

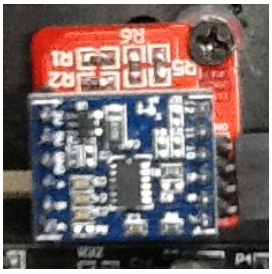

图 3-2 加速度计安装

3.3 陀螺仪传感器安装

陀螺仪传感器的位置应和加速度计的位置尽量的在一起，这样测出来的角度才是最准确的，同时陀螺仪的安装位置必须水平，否则会导致左转和右转出现加减速状况。

图 3-3 陀螺仪安装

3.4 测速模块

之前我们使用的是官方推荐的光电码盘，但是由于安装不当导致左右的差别特别明显，因此我们更换为编码器，编码器安装并没有太大的讲究，只要编码器的齿轮和电机齿轮之间不要有太大阻力就好。

图 3-4 编码器安装

3.5 电池摆放位置

电池占整车的大部分重量，因此电池决定了车的重心在哪儿，我们参加全国赛的时候选用了将车电池下移的策略，可以降低一些重心，行驶更为平稳。

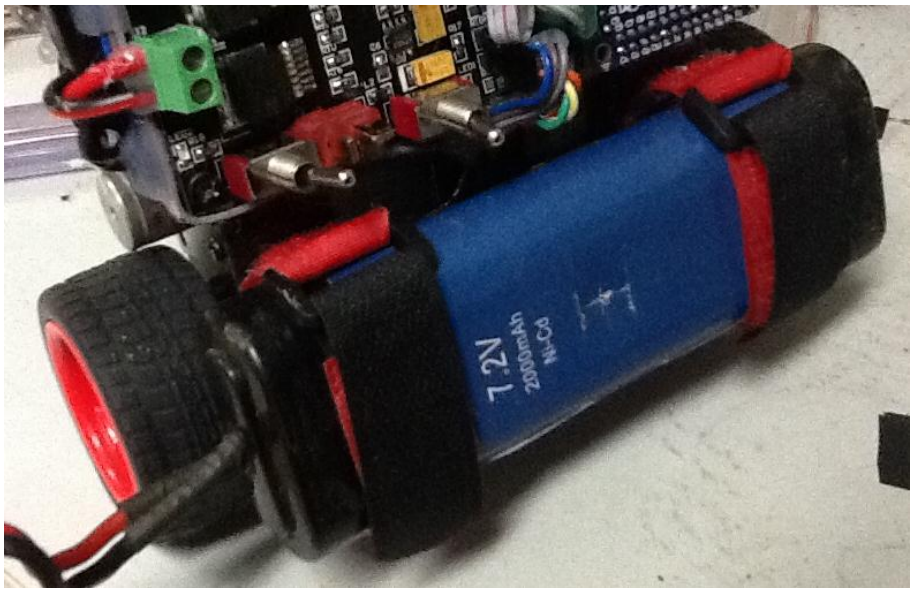

图 3-5 电池安装

第四章 软件设计

4.1 程序整体设计

4.1.1 程序框图

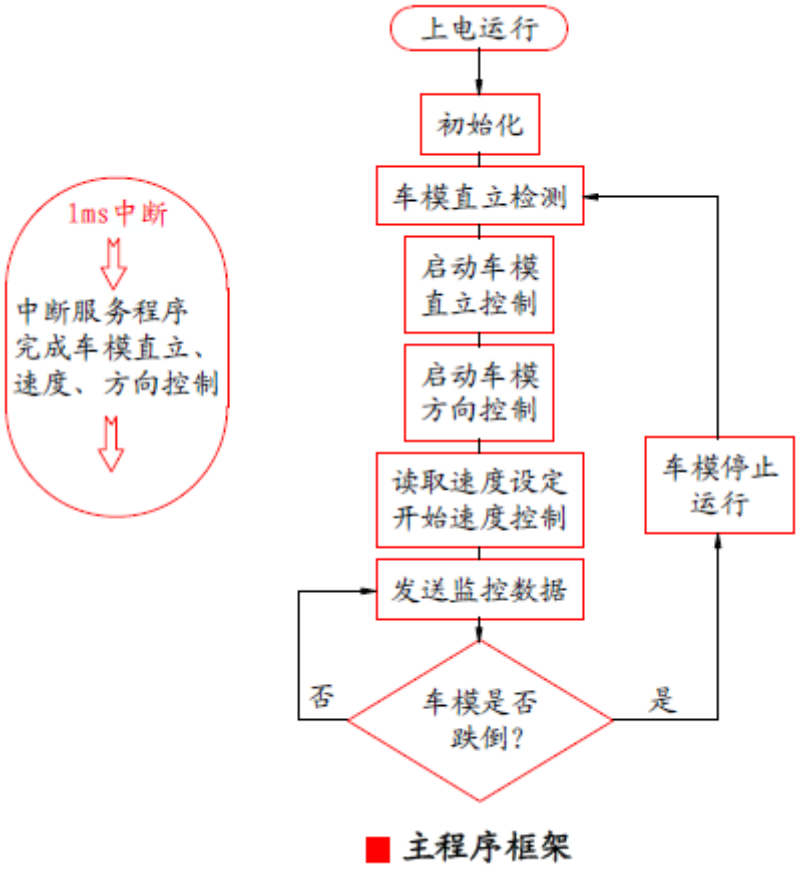

图 4 程序框图

4.1.2 1ms 中断调用

程序调用了 PIT 产生的 1ms 中断，来完成传感器数据采集、直立速度方向控制、电机输出等任务。注意避免的 1ms 中断调用中的任务用时超过 1ms，同时需要避免中断之间、中断与主程之间的互相干涉，提高了程序执行的性能。

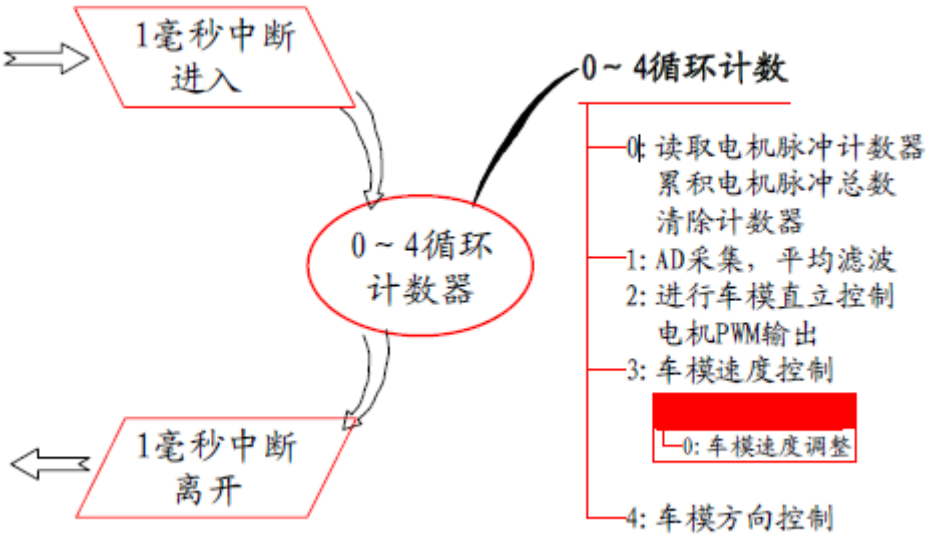

图 5 1ms 中断流程图

4.2 直立控制

车身姿态闭环控制需要对车体的倾角做反馈控制，在采用陀螺仪与加速度相配合的方式加上软件上的互补滤波来获取融合后的车体倾角。这一部分是直立控制的关键。姿态的采用了 PD 控制，即需要车体倾角和车的纵向角速度。直立的 P 和 D 参数有着不同的作用，P 参数可以加大车身在偏离给定角度后的恢复力，而 D 参数则是对这种恢复产生一定的阻尼作用。P 与 D 参数一同作用可以使车模保持一段时间的稳定直立。直立的 P 与 D 参数应在抖动不明显的前提下尽可能取大，这将有利于减弱速度和方向控制对于车模姿态的影响。

4.3 速度控制

车模的速度获取采用了编码器而非光电码盘，原因在于光电码盘的安装很难使两侧完全对称。速度的控制采用了 PI 控制，即使用车模当前速度和到现在为止的累加速度进行控制。P 参数可以加大车模对于速度的调节力度，而 I 参量的一些滞后可以使车模的速度稳定性有所提高。由于车模需要保持直立前进故不同于以往不可能使车模进行短时大幅的速度调整，否则会极大地影响车模的姿态。故在必须进行速度调整的时候需要逐渐地改变给定的速度，使得速度控制的输出值能够较为平滑的过渡。

4.4 方向控制

方向控制采用了左右两组电感获取的相对值来进行 PD 控制。P 参数可以使得车模在转向时力量较大，D 参数可以使车模在转向时更加连续平滑。

4.5 调试方法

有效的调试手段，对于及时发现并解决问题至关重要。而调试工具最重要的职能是获取智能车在行驶时的各项数据。我们在调试中主要使用了无线模块。

通过无线模块，可以将智能车行驶过程中的数据实时传送到电脑，是一种实用高效的调试手段。我们使用的无线模块是蓝牙模块只需一块，多辆车同时调试时不会互相影响。通过蓝牙模块以及一套相关的调试系统，我们可以在静态和动态的情况下对车模的控制参数进行调整，极大地简便了调试的过程。

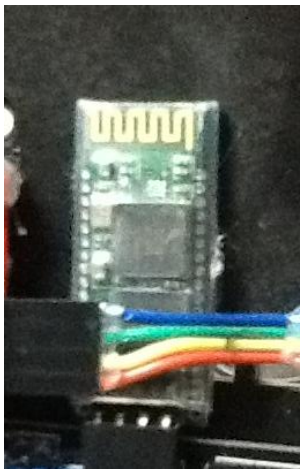

图 6 蓝牙模块

第五章 赛车主要参数

车模总质量	860g
模型车长/宽/高	420/160/300 mm
电容总容量	470uf
加速度计个数	1
陀螺仪个数	1
电磁传感器个数	4

第六章 结论

飞思卡尔智能车比赛已举行了七届了，回顾这几年的技术发展，从第一年以光电车为主，到第二年摄像头车成绩遥遥领先，到第三年激光管车的出现，再到第五年电磁车的出现，每一个新的检测方法的出现都使车速提升到一个新的高度。今年，组委会将电磁改为直立行走，这可谓是一个不小的创新，需要首先使小车直立起来，可以说，直立越稳定的车提速空间越大，因此不仅需要做到很好的路径优化，更重要的是车模的稳定性，因此调整车模的机械结构使其更加稳定花费了我们大量的时间。相信在不久的将来，车模的形态还会出现很大的改变，只有创新才会有进步。

参考文献

【1】 第七届全国大学生“飞思卡尔”杯智能汽车竞赛 《电磁组直立行车参考设计方案》00202012.3

附录

程序源代码

```
/*
*********************************************************************************
*                          北京邮电大学 K60
*********************************************************************************
*/



#include "common.h"
#include "include.h"
#include "isr.h"
#include "global.h"
#include "control.h"

u8 PITcounter = 0;
s16 g_nSpeedControlCount=0;
s16 g_nDirectionControlCount =0;
/*********************************************************************************
*   函数名称：PIT0_IRQHandler
*   功能说明：PIT0  定时中断服务函数
```

```
*   参数说明：无
*   函数返回：无
*   修改时间：2012-2-18       已测试
*   备       注：建议大家在写程序时尽量用宏定义，这样写出的代码可移值性强
*******************************************************************************/
void PIT0_IRQHandler(void)
{
  PIT_Flag_Clear(PIT0);                         //清中断标志位

  DisableInterrupts;
  //SPI0CR1_SPIE =1;

  //SCI0CR2_RIE = 1;
  TimerCount = (TimerCount+1)%10000;
  g_nAcce_speed_count++ ;
  g_nspeedAcount++;
  g_nUart_transmit_count++;
  g_nSpeedControlPeriod ++;
  SpeedControlOutput();
  // PORTA_PA7 = 1 ;
  g_nDirectionControlPeriod ++;
  DirectionControlOutput();
  //  PORTA_PA7 = 0;

  gpio_init    (PORTE,25,GPO,LOW);
  gpio_init    (PORTD,0,GPO,LOW);
  switch(PITcounter)
  {
  case 0:            //705us
    //GPIO_SET_1(PORTE,25) ;
    Acce_Result();                      //读取加速度计值
    //GPIO_SET_1(PORTE,25) ;
   // DIR_GET_Perpendicular();       //电感电压计算 12us    改后<20us
   // DIR_CAL_Perpendicular();
    //GPIO_SET_0(PORTE,25) ;
  break;
  case 1:         //280us
    //GPIO_SET_1(PORTE,25) ;
    GetPulseNumber();                     //脉冲计数
    Gyro_Result();                      //读取陀螺仪值
```

```
    Acce_Result_Cal();
    Get_Angle();                          //计算角度值单位角度
    AngleControl();                       //计算角度控制值
    DIR_GYRO();                           //方向陀螺仪测量
    //GPIO_SET_0(PORTE,25) ;
break;
case 2:         //780us
  //GPIO_SET_1(PORTE,25) ;
  // speed_P_I_control();

  g_nSpeedControlCount ++;
  if(g_nSpeedControlCount >= SPEED_CONTROL_COUNT)
  {
     // speed_P_I_control();
     if(g_nStartSpeedUpFlag ==1)
     {
        g_fStartSpeedUpCount +=1; //起步使用增量计数
     }
     StartSpeedUp();
        //变速控制
       SpeedUpControl();
       SpeedDownControl();
       SpeedControl();
       g_nSpeedControlCount = 0;
       g_nSpeedControlPeriod = 0;


  }


  //GPIO_SET_0(PORTE,25) ;
break;
case 3:         //810us
    //GPIO_SET_1(PORTE,25) ;
      DIR_GET_SP();
      DIR_CAL_SP();
    //GPIO_SET_1(PORTE,25) ;
  // DIR_GET_BZ(); //采样电感 615us
  // DIR_CAL_BZ();
    motorCtrl();                          //电机输出
```

```
      //GPIO_SET_0(PORTE,25) ;
    break;
    case 4:         //<20us
      //GPIO_SET_1(PORTE,25) ;
      g_nDirectionControlCount ++;

      VOLTAGE_LEFT =    SP_LEFT; //+ BZ_LEFT;//SP_LEFT ;
      VOLTAGE_RIGHT =    SP_RIGHT; //+ BZ_RIGHT;//BZ_RIGHT;
      DirectionVoltageSigma();
      if(g_nDirectionControlCount >= DIRECTION_CONTROL_COUNT)
      {
        DirectionControl();
        DirControlCarSpeedChange();
        g_nDirectionControlCount = 0;
        g_nDirectionControlPeriod = 0;
      }
      //GPIO_SET_0(PORTE,25) ;

    break;

    default:
    ASSERT_RST(0,"状态错误");
    }


    GPIO_TURN(PORTD,0);//用于测试 PIT
    PITcounter = (PITcounter + 1) % 5;    //PIT 状态转换
    EnableInterrupts;          //开总中断
}



/*****************************************************************************
*   函数名称：USART1_IRQHandler
*   功能说明：串口 1  中断  接收  服务函数
*   参数说明：无
*   函数返回：无
*   修改时间：2012-2-14       已测试
*   备       注：
*****************************************************************************/
```

```
void UART0_IRQHandler(void)
{
   uint8 ch;
   int temp;
   DisableInterrupts;                    //关总中断

   //接收一个字节数据并回发
   ch=uartRecvChar (UART0);                          //接收到一个数据
   uart0Printf("Input = %c\n",ch);

   switch(ch)
   {

   case 'u':
      g_fSpeedControlI -= 0.0001;
      break;
   case 'i':
      g_fSpeedControlI += 0.0001;
      break;
   case 'a':
     g_fSpeedControlP -= 0.001;
      break;
   case 's':
      g_fSpeedControlP += 0.001;
      break;
   case 'f':
      g_fAngleControlD -= 0.0001;
      break;
   case 'd':
      g_fAngleControlD += 0.0001;
      break;
   case 'o':
     g_fAngleControlP -= 0.001;
      break;
   case 'p':
      g_fAngleControlP += 0.001;
   case 'q':
      g_fSpeedControlOutOld = g_fSpeedControlOutNew = g_fSpeedControlOut = 0;
      g_fSpeedControlIntegral=0;
      break;
```

```
case 'j':
   CAR_ANGLE_SET+=0.1;
   break;
 case 'k':
   CAR_ANGLE_SET-=0.1;
   break;
case 'g':
   g_nSpeedControlFlag^=1;
   break;
case 'w':
CAR_SPEED_SET_J+=1;
break;
 case 'e':
CAR_SPEED_SET_J-=1;
break;

case 'r':
DIR_CONTROL_P+=0.01;
break;
case 't':
DIR_CONTROL_P-=0.01;
break;
case 'c':
DIR_CONTROL_D+=0.01;
break;
case 'v':
DIR_CONTROL_D-=0.01;
break;
case 'h':
g_nDirectionControlFlag^=1;
break ;
case 'x':
            g_nStopFlag ^= 1;
          break;
 case 'n':
              g_nCarSpeedUpValue += 1;
          break;
 case 'm':
            g_nCarSpeedUpValue -= 1;
          break;
```

```
    case ' ':
            SPEED_UP_DOWN_FLAG_J ^= 1;
        break;

     case '.':
            g_nCarSpeedDownDivide += 1;
        break;
     case ',':
            g_nCarSpeedDownDivide -= 1;
        break;

}

temp = (int)(g_fSpeedControlP*100000);
uart0Printf("\n SpeedP = %d",temp);

temp = (int)(g_fSpeedControlI*100000);
uart0Printf(" SpeedI = %d",temp);


temp = (int)(g_fAngleControlP*10000);
uart0Printf(" AngleP = %d",temp);

temp = (int)(g_fAngleControlD*100000);
uart0Printf(" AngleD = %d",temp);

temp = (int)(CAR_ANGLE_SET*10);
uart0Printf(" Angle = %d",temp);

uart0Printf(" Speed = %d",(int)CAR_SPEED_SET_J);

temp = (int)(DIR_CONTROL_P*10000);
uart0Printf(" DirP = %d",temp);

temp = (int)(DIR_CONTROL_D*100000);
uart0Printf(" DirD = %d",temp);

  temp = (int)(g_nCarSpeedUpValue);
uart0Printf(" UpValue = %d",temp);
```

```
    temp = (int)(SPEED_UP_DOWN_FLAG_J);
    uart0Printf(" UP_D_FLAG = %d",temp);

    temp = (int)(g_nCarSpeedDownDivide);
    uart0Printf(" DownD = %d",temp);


/*    switch(mode)
    {
        case 1:
            uart0Printf(" Mode = normal");
            break;
        case 2:
        uart0Printf(" Mode = SP");
        break;
        case 3:
        uart0Printf(" Mode = BZ");
        break;
        case 4:
        uart0Printf(" Mode = PER");
        break;
    }
*/
    EnableInterrupts;              //开总中断
}

/*********************************************************************************
*
                                        北京邮电大学  K60
*********************************************************************************
**/


#include "common.h"
#include "include.h"
#include "control.h"


    void main()
{
```

```
  DisableInterrupts;
  uartPrintf("北京邮电大学电磁组例程\n");

  gpio_init    (PORTA,15,GPO,HIGH);              //核心板测试 LED，初始化  PTA15 ：z 输出
高电平 ,即  初始化  LED0，灭
  gpio_init    (PORTA,4,GPO,LOW);                 //主板测试 LED       初始化  PTA4 ：   输出
高电平 ,即  初始化  LED0，灭



  //初始化拨码开关
  DIPswitch_ctrl();

  //电机初始化
  motor_init();

  //初始化 845x
  //MMA845x_Init();--

  //AD 部分初始化    包括 ENC  和电感
  ADdevice_init();

  gyro_init();
  dir_gyro_init();

  //初始化 PIT      1000Hz

  pit_init(PIT0,1000);

  //使能 PIT    测试时关闭    程序正常运行时使能
  //除实验三外都需要关闭使能

  PIT_IRQ_EN(PIT0);

  gpio_init(PORTD,0,GPO,0);     //用于测试 PIT
  //初始化外部测速
  FTM_QUAD_DIR_init(FTM1);
  FTM_QUAD_DIR_init(FTM2);

  //打开串口中断
```

```
    UART_IRQ_EN(UART0);
    EnableInterrupts;

    uartPrintf("anglegyro: %d \n",(int)gyro_stand_voltage);
    uartPrintf("dirgyro: %d \n",(int)gyro_dir_voltage);

    while(1)
    {
//以下 “1”表示打开当前测试程序，“0 表示关闭”
//实验 0
#if 0                //测试主板        测试结果：  成功
                     //核心板和主板上各有一个 LED 闪烁，频率 1Hz
                     GPIO_TURN(PORTA,15);
                     GPIO_TURN(PORTA,4);
                     delayms(500);
//实验一
#elif 0              //测试拨码开关        测试结果：  成功
                     //操作把 PA14      PA15      PA16      PA17 接在拨码开关电路上（主板
上是设计好的就是这几个引脚）
                     uartPrintf("切换拔码开关观看效果\n");
                     DIPswitch_ctrl();
                     delayms(1000);
//实验二
#elif 0            //测试电机            测试结果：成功
                     //操作   将 PC1       PC2       PC3       PC4 接在示波器上，看波型
                     int i = 0;
                     //占空比每一秒加 5%
                     //范围 0%-100%
                     //小于 0 是 PC2 PC4 是有波型 PC1 PC3 为 0 电平
                     //大于 0 是 PC1 PC3 是有波型 PC2 PC4 为 0 电平
                     for(i = -10000;i<10500;i+=500)
                     {
                       uartPrintf("当前占空比 %d %%\n",abs(i)*100/FTM_PRECISON);
#if MOTOR_DOUBLE
                       PWMSetMotor2(i,i);
#else
                       PWMSetMotor(i);
#endif
                       delayms(2000);
                     }
```

```
//实验三
#elif 0             //测试 PIT          测试结果：  成功
                    // 操作 :从初始化部分打开中断 !!! 开启 PD0 初始化
                            //设置 PD0 输出可以看到有一个 500Hz 的方波  说明 PIT1000Hz
正常


//实验四
#elif 0           //测试外部测速     测试结果:  成功
                  //从初始化部分   打开中断!!!!
                  //用  PWM  模拟编码器
                  //将  PC1 分别  与 PA12 短接进行测试
                  //将  PC3 分别  与 PB18 短接进行测试
                  int i = 0;
                  for(i = 1;i<60;i+=4)
                  {
                    FTM_PWM_freq(MOTOR_FTMN,1000*i) ;
                    FTM_PWM_Duty(MOTOR_FTMN,MOTOR_FTMCH1, 5000);
                    FTM_PWM_Duty(MOTOR_FTMN,MOTOR_FTMCH3, 5000);
                    delayms(2000);
                    uartPrintf("当前 PWM 波频率%7dHz ",i*1000);
                    uartPrintf("10ms 读取脉冲(速度):%4d   ",trueSpeedL);
                    uartPrintf("   %4d 个\n",trueSpeedR);
                  }

#elif 0           //测试 IIC      测试结果   成功
                  u16 x8,x10,x12;
                  u16 y8,y10,y12;
                  u16 z8,z10,z12;
                  uartPrintf("\n----------X----------Y----------Z----------\n");
                  while(1)
                  {
                      x8   =   Get845xX_8bit(10);   x10   =   Get845xX_10bit(10);x12   =
Get845xX_12bit(10);
                      uartPrintf("%5d%6d%7d   ",x8,x10,x12);

                      y8   =   Get845xY_8bit(10);   y10   =   Get845xY_10bit(10);y12   =
Get845xY_12bit(10);
                      uartPrintf("%5d%6d%7d   ",y8,y10,y12);
```

```
                z8 = Get845xZ_8bit(10); z10 = Get845xZ_10bit(10);z12 = Get845xZ_12bit(10);
                uartPrintf("%5d%6d%7d\n",z8,z10,z12);

                delayms(1000);      //延时 1s
            }
#elif 0         //测试模拟示波器     测试结果  :成功
            //FTM_PWM_freq(MOTOR_FTMN,100000) ;
            //FTM_PWM_Duty(MOTOR_FTMN,MOTOR_FTMCH0, 5000);
            //FTM_PWM_Duty(MOTOR_FTMN,MOTOR_FTMCH2, 5000);

            while(1)
            {
                OutData[0] = (int)(SP_LEFT);//(int)SP_LEFT;//(int)(BZ_LEFT-BZ_RIGHT);//(int)((g_fAngleControlOut)*1000);//g_fCarSpeed;//(int)g_fAcceData;//GET_ADVAL(ENC_AD,ENC_X_CH);//ad_ave(ACCER_AD,ACCER_Z_CH,ADBITS,20); //PE2
                OutData[1] = (int)(SP_RIGHT);//(int)SP_RIGHT;//(int)(SP_LEFT-SP_RIGHT);//(int)((fptemp)*1000);//g_nLeftMotorPulseSigma;//(int)CarVacceVal;//GET_ADVAL(ENC_AD,ENC_X_CH);      //PE0
                OutData[2] =(int)(0);//(int)(BZ_LEFT);//(int)(g_fSpeedControlIntegral*1000);//g_nRightMotorPulseSigma;//(int)CarGyroVal;//GET_ADVAL(ENC_AD,ENC_Y_CH);      //PE1
                OutData[3] =(int)(0);//(int)(BZ_RIGHT);//(int)(g_fCarSpeed*100);//(int)(fptemp*1000);//(int)CarAngle;

                OutPut_Data();
            }
#elif 0         //测试 ENC03    测试结果 成功
            /*
            u16 ADresult = 0;
            float temp = 0;
            uartPrintf("\n-------X----------Y----------\n");

            while(1)
            {
                ADresult = GET_ADVAL(ENC_AD,ENC_X_CH);
                temp = (float)ADresult*3.3/1024;
                uartPrintf("%10d %3.3f",ADresult,temp);
```

```
        ADresult = GET_ADVAL(ENC_AD,ENC_Y_CH);
        temp = (float)ADresult*3.3/1024;
        uartPrintf("%10d %3.3f\n",ADresult,temp);

        time_delay_ms(1000);        //延时 1s
    }
    */
     while(1)
    {
         OutData[0] =(int)(CarVacceVal);
         OutData[1] =(int)(CarGyroVal);
         OutData[2] = (int)(CarAngle);
         OutData[3] = (int)(0);

         OutPut_Data();
    }



#elif 0             //测试电感模块 0     测试结果 成功
    u16 ADresult = 0;
    float temp = 0;
    uartPrintf("\n-----L00-------L01-------L02-------L03-------L04---\n");
    uartPrintf("\n-----PB0-------PB2-------PB3-------PE24------PE25---\n");
    while(1)
    {
        ADresult = GET_ADVAL(L_AD0,L_00);
        temp = (float)ADresult*3.3/1024;
        uartPrintf("%10d %3.3f",ADresult,temp);

        ADresult = GET_ADVAL(L_AD0,L_01);
        temp = (float)ADresult*3.3/1024;
        uartPrintf("%10d %3.3f",ADresult,temp);

        ADresult = GET_ADVAL(L_AD0,L_02);
        temp = (float)ADresult*3.3/1024;
        uartPrintf("%10d %3.3f",ADresult,temp);

        ADresult = GET_ADVAL(L_AD0,L_03);
```

```
            temp = (float)ADresult*3.3/1024;
            uartPrintf("%10d %3.3f",ADresult,temp);

            ADresult = GET_ADVAL(L_AD0,L_04);
            temp = (float)ADresult*3.3/1024;
            uartPrintf("%10d %3.3f\n",ADresult,temp);

            time_delay_ms(1000);        //延时 1s
        }
#elif 0         //测试电感模块 1    测试结果 成功
        u16 ADresult = 0;
        float temp = 0;
        uartPrintf("\n-----L10-------L11-------L12-------L13-------L14-----\n");
        uartPrintf("\n-----PE2-------PE3-------PB1-------PE10------PE11----\n");
        while(1)
        {
            ADresult = GET_ADVAL(L_AD1,L_10);
            temp = (float)ADresult*3.3/1024;
            uartPrintf("%10d %3.3f",ADresult,temp);

            ADresult = GET_ADVAL(L_AD1,L_11);
            temp = (float)ADresult*3.3/1024;
            uartPrintf("%10d %3.3f",ADresult,temp);

            ADresult = GET_ADVAL(L_AD1,L_12);
            temp = (float)ADresult*3.3/1024;
            uartPrintf("%10d %3.3f",ADresult,temp);

            ADresult = GET_ADVAL(L_AD1,L_13);
            temp = (float)ADresult*3.3/1024;
            uartPrintf("%10d %3.3f",ADresult,temp);

            ADresult = GET_ADVAL(L_AD1,L_14);
            temp = (float)ADresult*3.3/1024;
            uartPrintf("%10d %3.3f\f",ADresult,temp);

            time_delay_ms(1000);        //延时 1s
        }

#elif 0         //中断法读串口数据   测试结果
```

```
                    //直接输入看效果

#elif 1
                    Stand_Test();
                     if(g_nUart_transmit_count>100)
                     {
                        //uartPrintf("control mode: %c\n",g_cDir_method);
                      uartPrintf("S %d",(int)(g_fCarSpeed ));
                      uartPrintf(" A %d",(int)(CarAngle));
                //         uartPrintf(" MODE %d",(int)(DirModeNow ));
                   //      uartPrintf(" C %d",(int)(CarGyroVal*100));
                      uartPrintf(" C_SET %d\n",(int)CAR_SPEED_SET);

                      // uartPrintf("SP_CHA: %d    ",(SP_LEFT-SP_RIGHT));
                      //uartPrintf("sigma: %d\n",(int)(Dir_Change_Sigma_New*1000));
                        g_nUart_transmit_count = 0 ;
                     }

            /*          if(g_nAcce_speed_count > 5000)
                     {
                         while(1)
                     {
                            OutData[0]
=(int)(g_fDirectionControlOutNew*DIR_CONTROL_P*100);
                            OutData[1]
=(int)(-1*g_fDirectionControlOutOld*DIR_CONTROL_D*100);
                            OutData[2] = (int)(g_fDirControlMotorOutNew*100);
                            OutData[3] = (int)(0);

                            OutPut_Data();
                     }

                     }

               */




#endif
```

```
    }
}
```