第九届“飞思卡尔”杯全国大学生
智能汽车竞赛
技术报告

学　　校：东南大学

队伍名称：蓝色妖姬

参赛队员：　智向阳

带队教师：　谈英姿

　　孙　琳

关于技术报告和研究论文使用授权的说明

本人完全了解第九届“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：

目录

第一章 引言

“飞思卡尔”杯全国大学生智能汽车竞赛是教育部为了加强大学生实践、创新能力和团队精神的培养，经研究决定，委托教育部高等学校自动化教学指导分委员会主办的每年一度的全国大学生赛事。该项赛事有着很强的综合性，是涵盖了控制、模式识别、电子、电气、计算机和机械等多学科交叉的科技创意比赛。

为了进一步提高全国大学生智能汽车竞赛创造性和趣味性，激发高校学生参与比赛的兴趣，提高学生的动手能力创新能力和接受挑战的能力，组委会在前年电磁直立组和去年光电直立组的基础上继续规定今年摄像头组车模采用直立行走，这项新的改动给我们带来挑战的同时也给了我们新的、更大的动力

我们积极组队参加第九届全国大学生智能汽车竞赛。经过半年多的准备，我们发现相对于四轮车，机械结构在直立车的性能提升中占有很大的比重。在软件架构差不多的情况下，不同的机械结构会使得车子的运行结果又很大的不同，因此在不断的磕磕碰碰中，我们总结了一些硬件和软件上的经验和教训，将在之后的章节中详细解说。

第二章　机械结构设计

2.1 车模结构

通过华东赛前对车模的调试及华东赛上各种结构车模的表现，我们发现车模结构对车速的提高、转弯控制及过人字的稳定性有着至关重要的影响。主要集中在以下几点：

1. 重心降低可以使得拐弯的极限速度得到提高。当小车高速拐弯时，会出现内侧轮子离地的现象，从而使得车的方向控制失效，甚至出现侧翻的现象。自然，重心越低，对小车过弯也越有利。
2. 本方案采用车模后仰的设计，重心大幅度降低，相较于传统直立方案的转弯极限速度有所提高，而且更容易通过角度变化控制加速度。
3. 车身上半部分的质量对车过障碍的稳定性影响较大。以往的比赛经验中，我们总是尽量减轻车身重量，但从近期的调车过程中我们发现，车身重量并不是越轻越好。过轻的体重，会使得车与地面的摩擦力减小，不太利于转弯及车身稳定。另外，合理的配重还可以使得车身转动惯量较为适中，在直立控制和速度控制稳定性间达到较好平衡。

图 2-1 车模整体图

2.2 摄像头安装

摄像头安装的位置和角度决定了车的前瞻大小及读取到图像的视野。较低的高度可以防止头重脚轻，使得直立更加稳定。但较低的高度同时会使得车在加减速时， 由于车身姿态的改变导致前瞻大幅度变化， 这样不利于转弯算法的稳定。所以我们选择了 35cm，较为适中的高度，并以较轻的空心碳杆固定摄像头，采用软排线尽量减轻车膜上半部分重量，取得较好效果。摄像头镜头使用的是广角镜，120 度视角，可以尽量避免丢线情况发生。

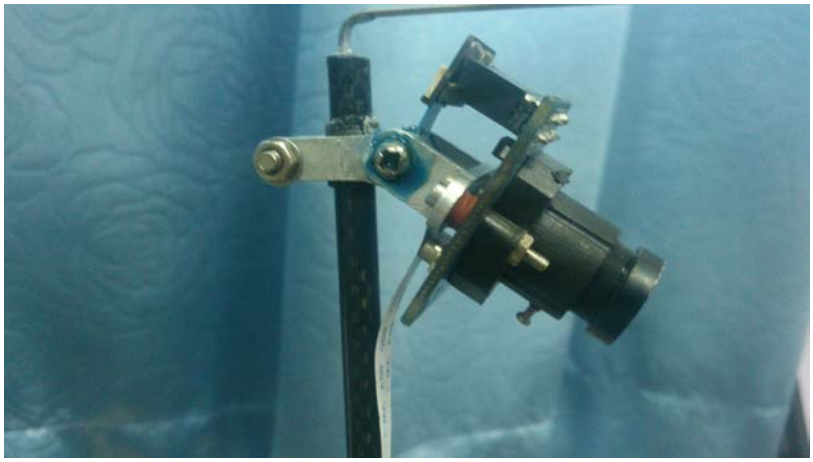


图 2-2 摄像头安装

2.3 轮胎

通过实际的尝试，我们发现，轮胎不仅决定了拐弯的好坏，同时对过人字路也有较大影响。车模自带的带海绵轮胎弹性较小，在经过人字路时转向角速度较大，比较容易引起车身不稳定。为此我们在轮胎内填充了较有弹性的海绵，使得轮胎类似于充了气一样，有较好弹性，从而更适合转弯、避障和过人字。

另外，为了保证车过弯时的摩擦力，我们磨掉部分了轮胎的花纹，使得过弯更加流畅。我们还用 502 胶粘住了轮胎和轮毂，避免轮胎与轮毂之间打滑。

在调车过程中，需要注意经常擦拭赛道和轮胎，避免轮胎打滑。

图 2-3 轮胎

2.4 配重

由于重心降低可以有效提高车的转弯性能，我们在车身前部的电池下方加了铁块作为配重，结合实际测试结果选择了合适的重量。

图 配重

2.5 其他传感器的安装

车模上其他传感器还包括陀螺仪，加速度计等的安装位置见下列图像


图 2-5-1 陀螺仪

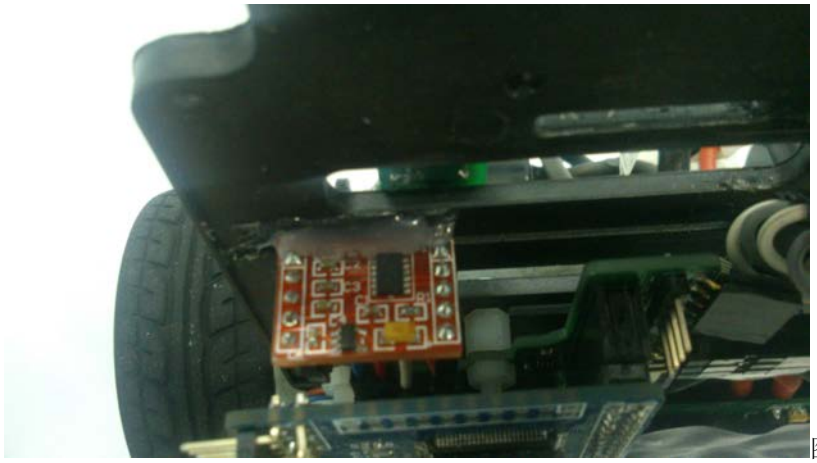
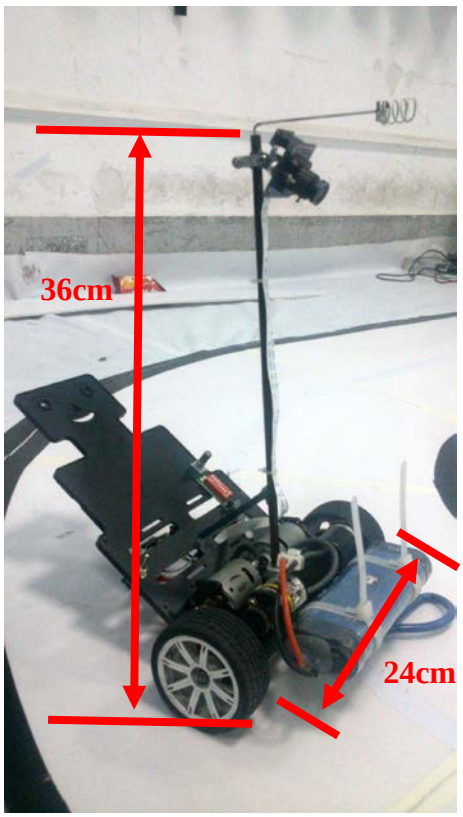


图 2-5-2 加速度计

2.6 车体主要技术参数汇总



改进后的车模静态电流约180mA，静态功耗约 1.3W。

我们总共使用了 5 个传感器，如下所示：

· 1 个加速度计

· 1 个陀螺仪

· 1 个摄像头模块

· 2 个 500 线编码器测速模块

本车没有使用额外的舵机以及电机。

第三章 硬件电路设计

3.1 主要传感器选择

3.1.1摄像头

OV7620是CMOS彩色／黑白图像传感器。它支持连续和隔行两种扫描方式，VGA与QVGA两种图像格式；最高像素为664×492，帧速率为30fp8；数据格式包括YUV、YCrCb、RGB三种，能够满足一般图像采集系统的要求。

OV7620内部可编程功能寄存器的设置有上电模式和SCCB编程模式。本系统采用SCCB编程模式，连续扫描，8位灰度值数据输出。

主要参数：

Array Elements	664 x 492
Pixel Size	7.6 x 7.6 um
Image Area	4.86 x 3.64mm
Electronic Exposure	500 : 1
Scan Mode	progressive interlace
Gamma Correction	128 Curve Settings See specifics
Minimum Illumination	2.5 lux @ f1.4 0.5 lux @ f1.4 (3000K)
S/N Ratio	> 48dB
Power Supply	5VDC, ±5%
Power Requirements	<120mW Active <10uW Standby
Package	48-pin LCC

3.1.2 陀螺仪

为达到角度控制和转向控制的需要，采用意法半导体公司的三轴数字陀螺仪 L3G4200。该 L3G4200D 是一个低功率的三轴角速度传感器，能够提供前所未有的零速度水平和灵敏度。它包括传感元件和一个能够通过数字接口（I2C / SPI）向外部世界提供测得的角速度集成电路接口。

3.1.3加速度计

加速度计模块采用飞思卡尔的MMA7361三轴模拟MEMS加速度计。加速度角度传感器是通过测量由于重力引起的加速度计算出器件相对于某一平面的倾斜角度。角度传感器反应灵敏、输出数据准确且价格适中。

PCB 主要包括最小系统、电源模块、驱动模块，以及其他接口和调试电路

3. 2 最小系统

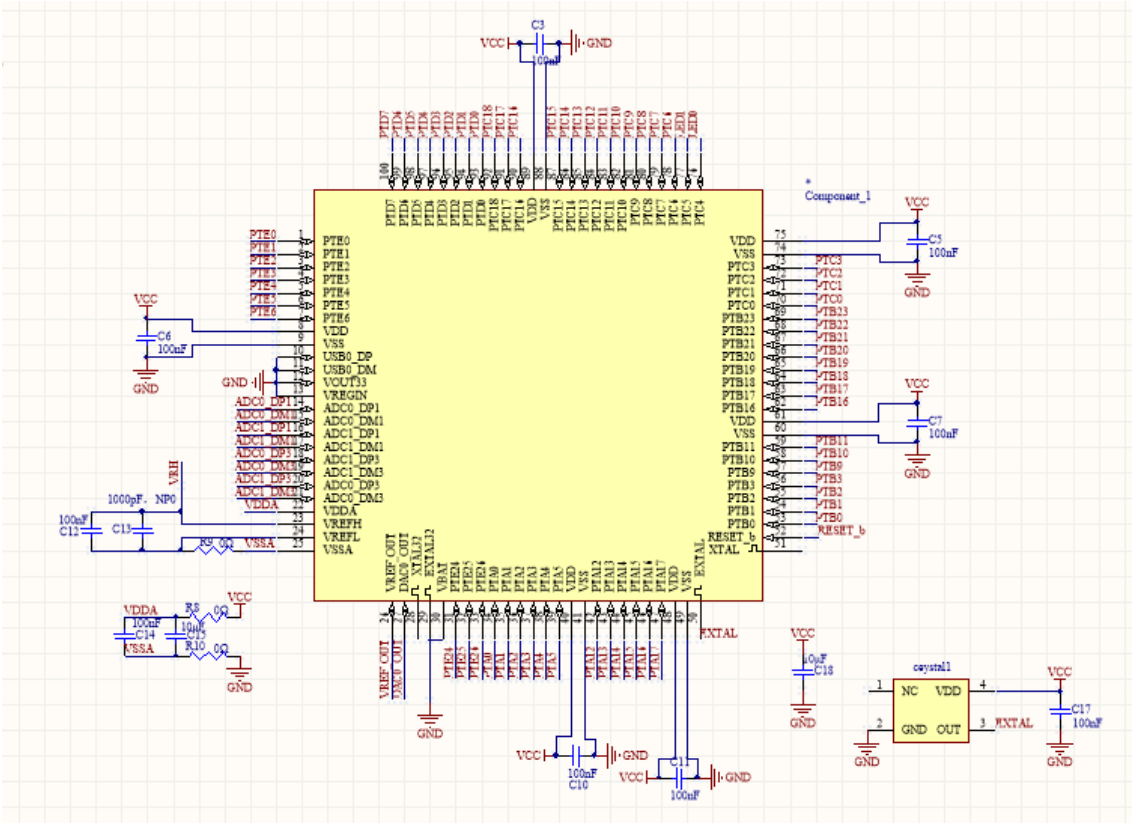


图 3-1 最小系统原理图

MK60N512VLL100 是 K60 系列 MCU。Kinetis 系列微控制器是 Cortex-M4 系列的内核芯片。K60 内存空间可扩展，从 32 KB 闪存/ 8 KB RAM 到 1 MB 闪存 / 128 KB RAM，可选的 16 KB 缓存用于优化总线带宽和闪存执行性能。

3. 3 电源模块

电源管理模块：主要提供系统各模块正常工作所需要的电源。稳定、高效的电源模块是系统正常工作的基础。

电源模块主要包括 5V 与 3.3V 稳压模块，其中电池电压直接对电机进行驱动，5V 主要提供给陀螺仪、摄像头、测速模块，3.3V 负责对核心板、以及调试用的无线模块等进行供电。经过测试，5V/3.3V 系统需要的总电流约为 200-300mA，故我们使用低成本、外围电路简单、占用 PCB 空间较少的 AMS1117 作为 5V 以及 3.3V 电源芯片。

由于这些电路较为简单，就不专门展开讲解。详见下图 3-2

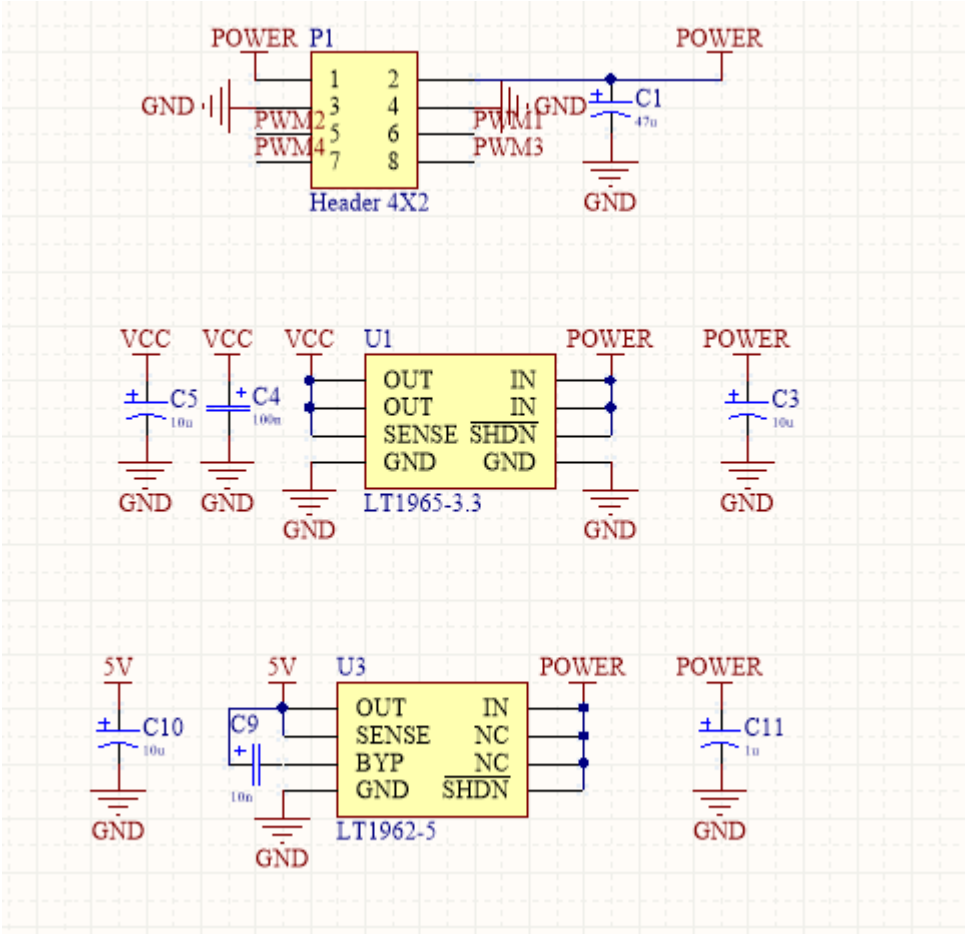


图 3-3 电源模块

3. 4 驱动模块

E 车使用的电机在实际使用时的电流不超过 2A，出于驱动能力和损耗的考虑，我们使用了 IR2104 构成全桥对电机进行驱动。驱动电路图见 3.2，3-3.

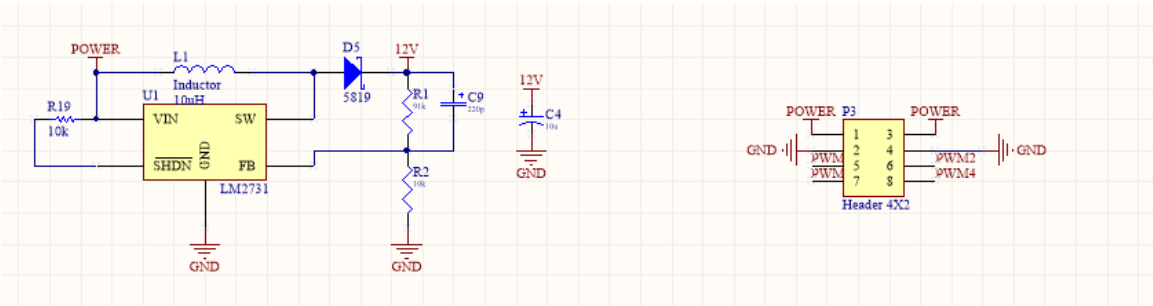


图 3-2 驱动升压电路

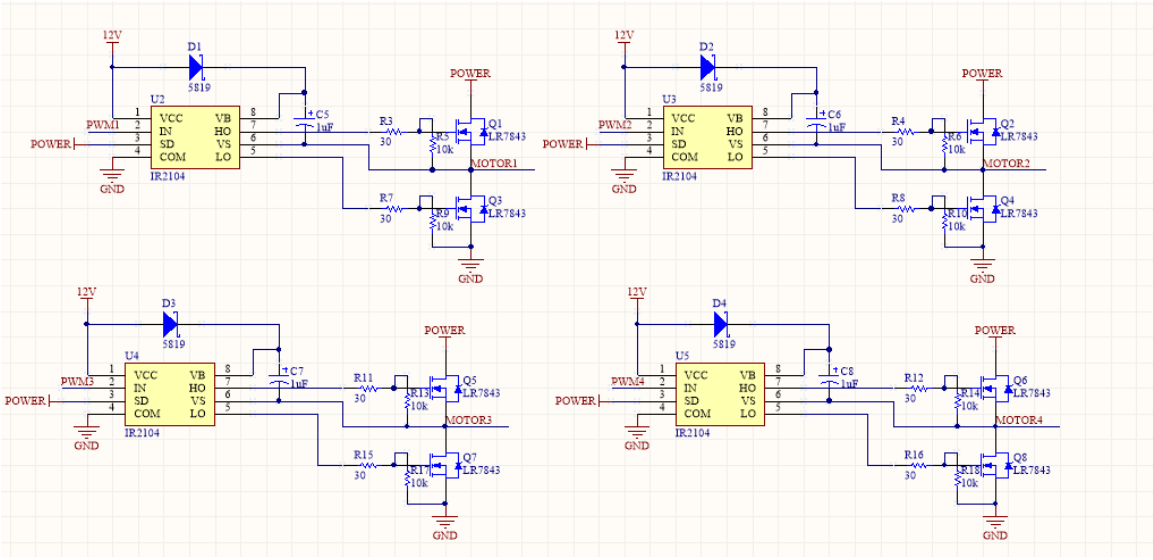


图 3-3 驱动电路

3.5 其他

除了上述模块，本车使用的串口、LED、拨码开关、电量检测等较为简单，在此不再赘述。若需了解可以参见下图电路图纸。

我们的核心板是自行设计的，加速度模块、陀螺仪模块、测速模块以及 摄像头模块均系网上购买而来。故在此不再详述这些模块的电路。

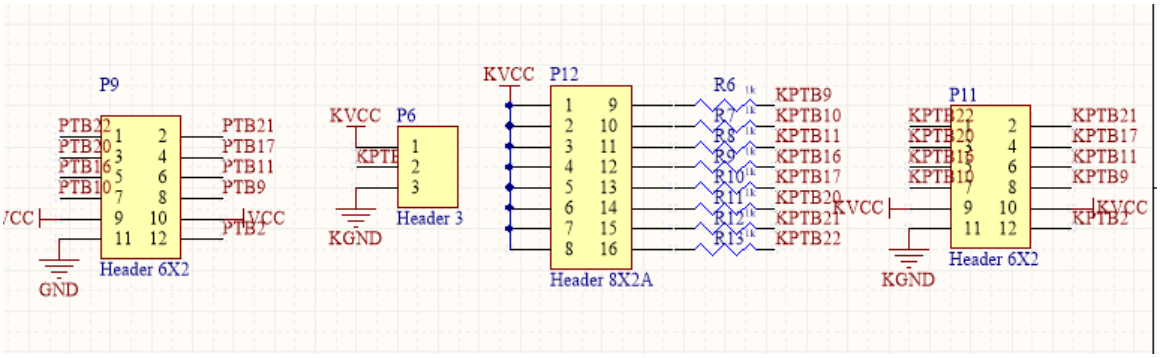


图 3-5 拨码开关接口

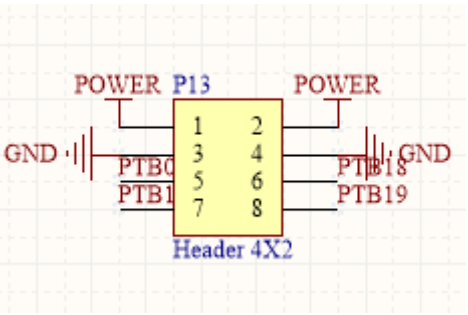


图 3-6 编码器接口

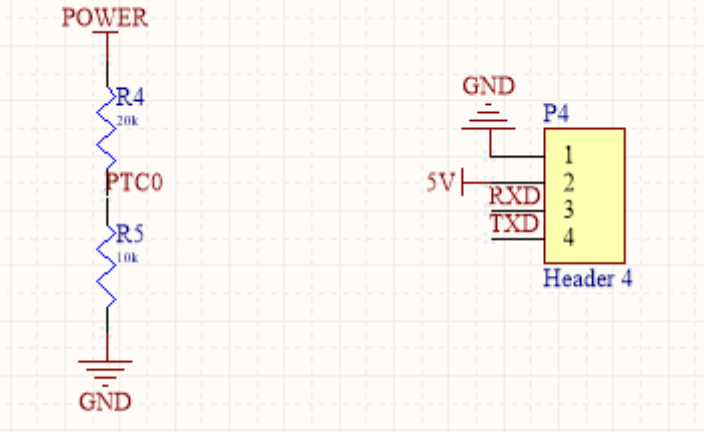


图 3-7 电压测量电路和串口接口

第四章 软件开发

本车使用 PK60DN512VLL10 作为控制芯片，采用摄像头时序，即 16.6ms，作为转向控制周期。直立控制周期为转向周期的 1/4，即 4.16ms，速度控制周期是转向周期的 2 倍，即 33.3ms。通过将车模的直立控制与速度控制和转弯控制线性叠加并控制电机。

4.1 车模直立控制

直立控制采用 PID 算法，整个车模可以看作为一个倒立摆，通过陀螺仪模块，我们可以得到车模当前的角度，通过加速度计模块得到角加速度。由于陀螺仪温漂大， 加速度计噪声大，我们通过卡尔曼滤波的到最合适的当前角度和角加速度。我们设定当前的角度与经过程序滤波后的目标角度之差给电机一个 PWM 使车模保持平衡。D 的作用是使车模更加精确的跟踪到目标角度，而积分的作用是消除角度积累的误差。直立控制算法流程如下：

1.获得卡尔曼滤波后的融合角度 X[0]与角加速度 X[1]。

2.获得车平衡时的中心角度。

3.设定当前的滤波后的目标角度。

4.计算所得直立电机的控制量；

5.将直立电机的控制量加入到 PWM 中。

4.2 车模速度控制

速度控制采用了 PI 算法，我们没有直接将速度控制直接叠加到电机的 PWM 中，而是将速度控制量乘上一个比例系数叠加到程序所设定的目标角度上，这样设置在调节更容易直立控制参数与速度控制参数，两者互不影响。若直接将速度控制与直立控制相叠加则会互相影响。而积分的作用则是消除积分误差量。为了更好的控制直立车，我们采用了分段加速使之加速慢减速快的算法。

速度控制算法的流程如下：

1. 设定一个目标速度。

2. 通过测速模块测得当前的速度 Speed 和上一个周期的速度。

3. 设定分段加速 （ SpeedGrade = SpeedGrade+(SpeedSet-SpeedGrade)*0.6。

4. 计算速度所得的控制量 (Speed-SpeedGrade)*Speed_P+Acceleration*Speed_D + SpeedI)。

5. 将速度所得的控制量加入到目标角度中。

4.3 图像采集及处理

4.3.1 图像采集

通过 sccb 写摄像头寄存器，将 ov7620 的图像输出格式改为 qvga，连续输出模式，然后对行信号及像素信号进行硬件分频，通过 DMA 采集图像，图像大小为 50 行，158 列，样图如图 4-1 所示。

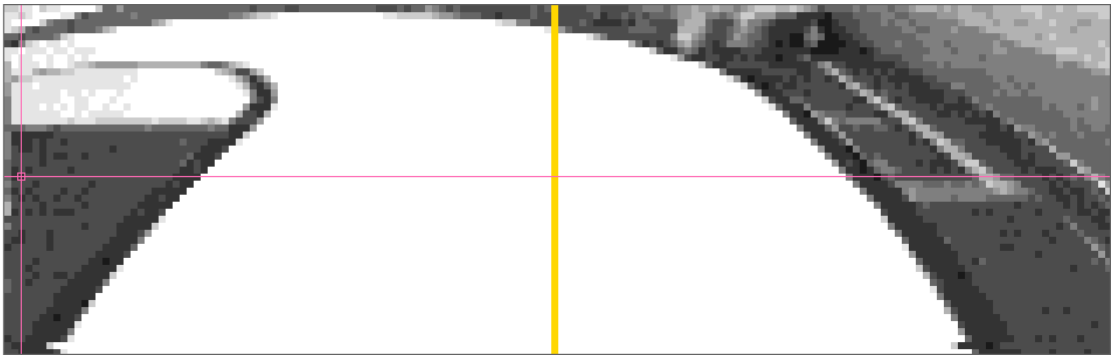


图 4-1

4.3.2 边线提取

边线提取采用窗口搜索算法，该方法经测试可以有效对抗一般反光，其实现方法如下：在赛道中间设置一个窗口，向两边推进。检测窗口内是否有黑白或白黑跳变，如果有，则认为找到了边线并记录。

因为赛道内全部是白色，所以灰度值只会有小范围的波动。而在赛道边缘出现了黑线，灰度值一定会有一个较大的跳变，如图 4-2。这样只要阈值合适，即使有一定的反光，还是可以检测到跳变。而且从赛道中间往两边检测，可以避免误识别其它赛道和赛道外的反光，提高了抗干扰能力。

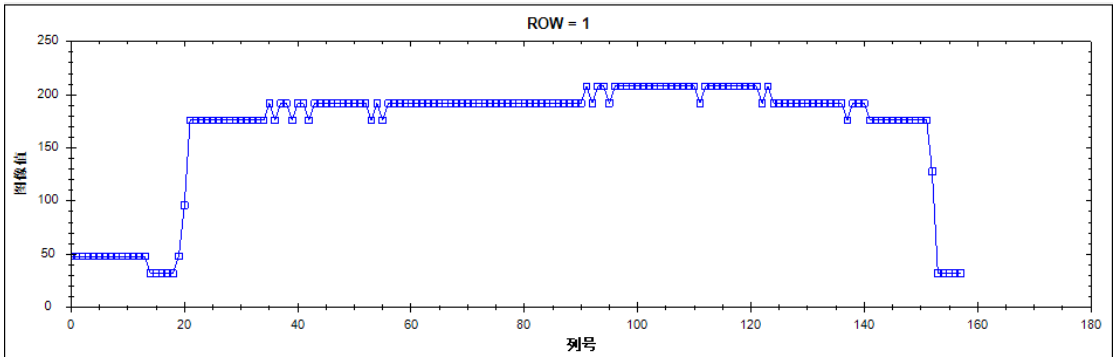


图 4-2

这种方法比较困难的是，确保准确的从赛道中开始检测。我们采取的是根据上一场的赛道来推测。

我们提取的边线如下图 4-3，绿点是找到的左线，红点是右线。

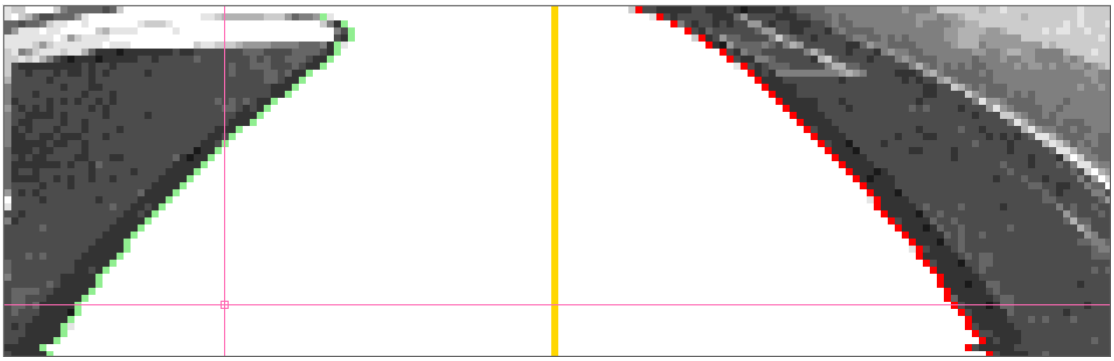


图 4-3

4.3.3 引导线提取

我们的引导线是根据找到的边线来计算得出的：

1)如果两条线都找到了，用左线位置加上右线位置再除以 2，即是引导线如图 4-4 所示。
2)如果只找到了左线，则用左线减去表 1，得到引导线如图 4-5。
3)如果只找到了右线，则用右线加上表 2，得到引导线。

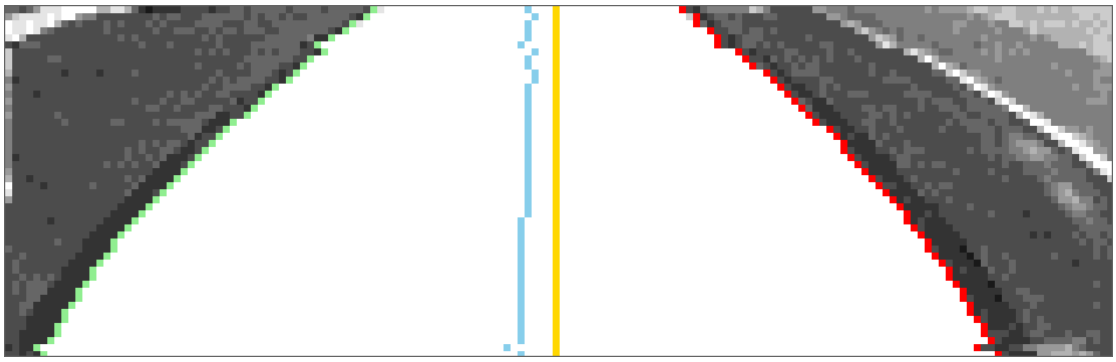


图 4-4

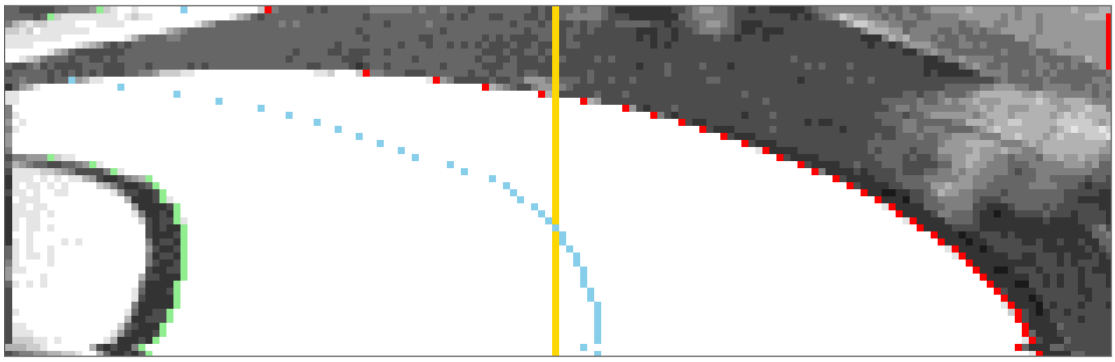


图 4-5

4.3.4 障碍识别与处理

第九届的摄像头平衡组增加了障碍这一元素，障碍由两块砖头叠在一起由黑纸封装，路障内侧与赛道中心线距离为 8 厘米。

我们采集到的障碍图像如图 4-6 所示。

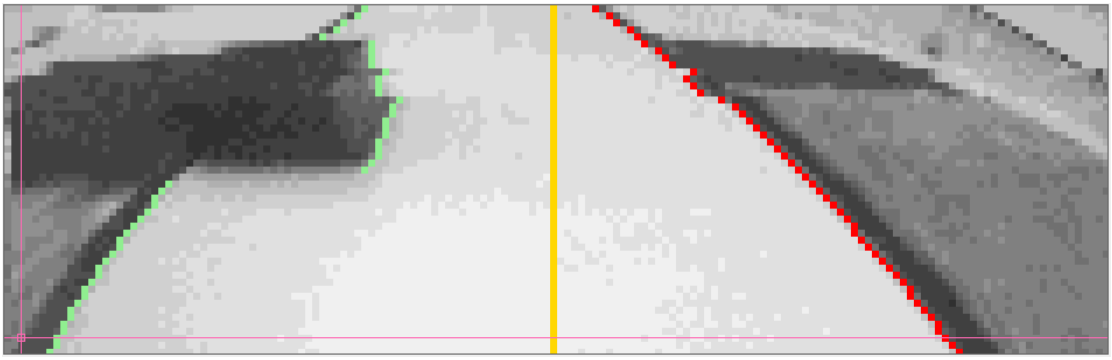


图 4-6

从图像可以看出，在障碍所在位置，边线会寻到障碍内侧，从而存在跳变。根据这一特征，我们可以很明显的找到边界线跳变判断出赛道上有障碍，从而直接改变引导线让车向另一边拐，从而躲过障碍，处理后的引导线如图 4-7 所示。

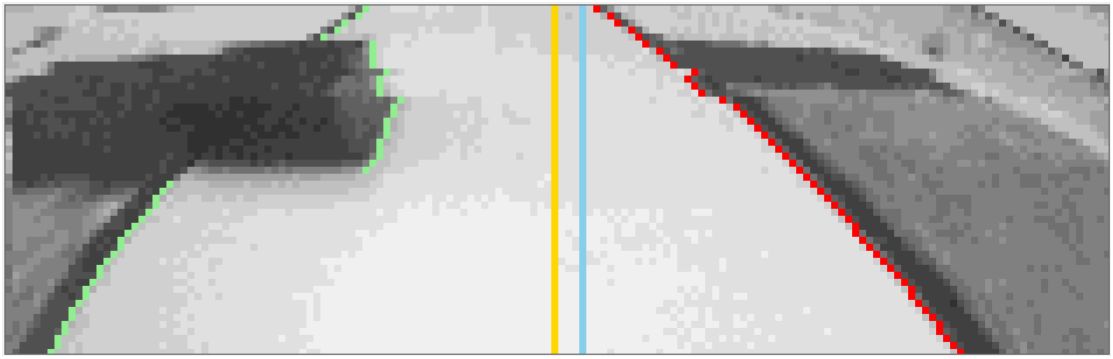
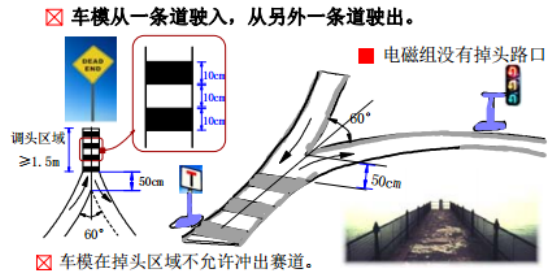


图 4-7

4.3.5 人字路识别与处理

第九届的摄像头平衡组增加了强制调头区这一元素，其情况如图 4-8 所示。



1、掉头区域长度为 1.5 米。其中铺有斑马线，线宽和间距都是 10 厘米。
2、斑马区的第一条黑线距离两条内侧交汇点 50 厘米。
3、交汇的两条道路可以是直线，也可以是曲线。斑马区的中心线处于交会角的角平分线上。

图 4-8

小车看到的人字路图像如图 4-9 所示

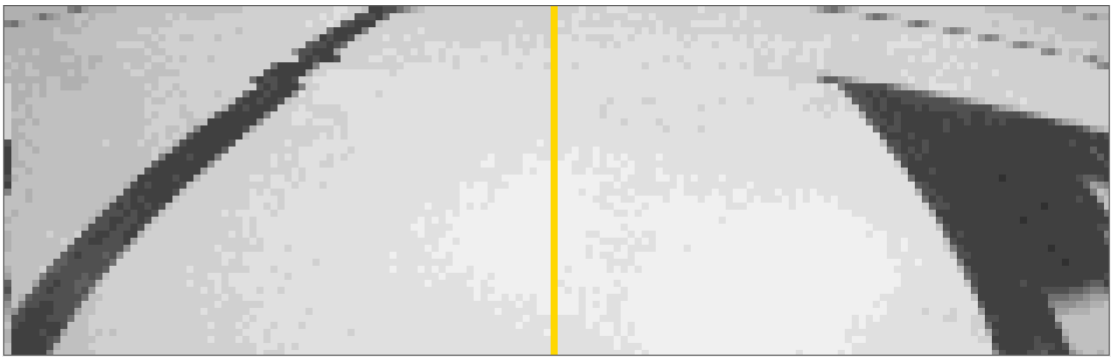


图 4-9

人字我们采用的是通过找人字弯内侧的尖角进行识别，而不是通过斑马线识别，这样就可以提前转弯，顺畅的过人字。

由于小车在过人字路的时候，其最大转弯半径很小，如果当成一般弯道必然转不过去，所以必须特殊处理。我们的做法是虚拟出一条二次曲线作为赛道边线，让车的转弯半径变小，从而转过去。

4.4 车模转向控制

转弯的算法则是通过所得的中心值（center)与图像中心值的差值和此时的中心值与上

一刻的中心值差值所得叠加到两轮的电机的输出上，使之进行转弯。

车模转弯控制流程如下：

1.算得边界（left&&right）和赛道宽度（width）及赛道中心值（center);

2.算得转弯参量;

3.左右轮分别加减转弯控制量。

第五章 开发及调试工具

5.1 开发平台

我们使用飞思卡尔公司基于 eclispe 研发的 CodeWarrior Development Studio 10.4 作为程序开发工具，图 5-1 是该软件的主界面。

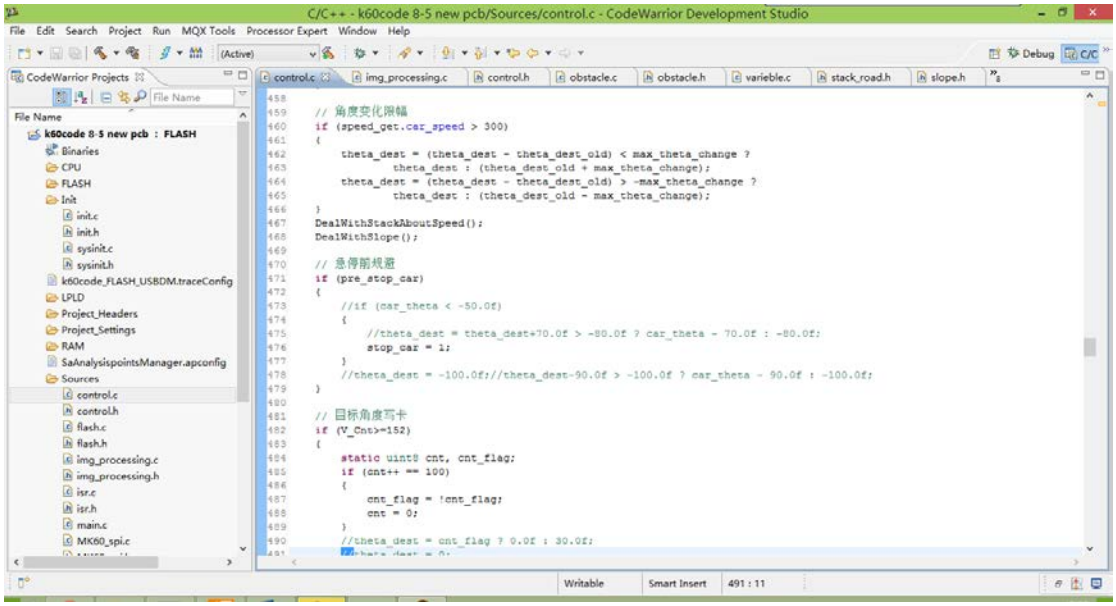


图 5-1

a） SD 卡调试平台

K60 有 SDHC 模块，可以方便的进行 SD 的高速读写，实测写 10kB 数据仅需 1ms 左右，10kB 足够我们存储一幅图像及需要的数据。我们用 C#编写了 SD 卡上位机读取并显示图像以及一些实时数据，这样就可以做到对小车在赛道上行驶的实时监控，方便查出出现的问题，极大的提高了调车效率。

b） 串口调试平台

我们用 C#编写了串口调试工具，通过蓝牙与小车进行通信，该上位机可以接收并同时显示 8 个浮点型变量，还可以发送变量给小车，从而做到实时调试参数，避免了频繁烧写代码的麻烦。串口调试工具的主界面如图 5-2 所示。


串口波形
选择串口号
打开串口
关闭串口
标识数
值
发送数据
紧急停车
启动
通道1
通道2
通道3
通道4
通道5
通道6
通道7
通道8
X轴最大值
600
Y轴最小值
-500
Y轴最大值
500

图 5-2

第六章　总结

由于平衡车第一次使用摄像头作为主传感器，没有前人经验可以参考，而我以前也从未接触过平衡车，一切都是从零开始，其难度可想而知。而人字、障碍、坡道都是对于平衡车来说难度都相当高，在这方面我们花了大量的时间和精力，调出一辆稳定高速的小车一直是我们的最高目标。

调试平台的建立与不断完善，标志着我们的智能车研究进入了一个全新的环节，用好调试工具不仅可以发现程序运行中的问题和算法的错误，更可以在不改变硬件的情况下，使车模的速度趋于最优，真正实现闭环调试。

智能车比赛已经经历九届比赛，几乎每一年都有新的变化。明年或许车模会变，或许赛道会变，但是智能车团队不会变，智能车人追求卓越的信念不会改变，老队员与新队员的传承不会改变，不同学校之间的技术交流更加便利，愿每一个智能车团队在新的一年中都能取得更新的成果，愿智能汽车有朝一日走入人们的生活，为人类的生活增添新的色彩。

参考文献

[1] 竞赛秘书处. 电磁组直立行车参考设计方案. 2012
[2] 田玉平. 自动控制原理[M].北京:科学出版社.2006
[3] 王勤. 计算机控制技术[M].南京:东南大学出版社.2003
[4] 卓晴等. 学做智能车:挑战“飞思卡尔”杯[M].北京:北京航空航天大学出版社.2007
[5] seu04 队. 第八届东南大学 seu04 队技术报告
[6] seu3 队. 第七届东南大学 seu3 队技术报告

附录：

```
#include "common.h"
#include "init.h"
#include "HW_ADC.h"
#include "varieble.h"
#include "MK60_spi.h"
void OV7620(void)
{
    uint8 i=0;

    LPLD_SCCB_Init();
    while(i==0)
        i += LPLD_SCCB_WriteReg(0x14,0x24);              //QVGA(320*120)
    while(i==1)
        i += LPLD_SCCB_WriteReg(0x24, 0x20);         //连续采集模式(320*240)
    while(i==2)
        i += LPLD_SCCB_WriteReg(0x70, 0xc1);         //驱动电流增加一倍
    while(i==3)
        i += LPLD_SCCB_WriteReg(0x06, 0xc0);         //亮度控制

    //while(i==4)
        //i += LPLD_SCCB_WriteReg(0x0c, 0x00);       //白平衡蓝色背景控制
//  while(i==4)
//              i += LPLD_SCCB_WriteReg(0x07, 0xc0);         //对比度控制
    //i += LPLD_SCCB_WriteReg(0x19, 0x04);       //数据从第 4 行开始输出
    //i += LPLD_SCCB_WriteReg(0x1A, 0xf3);       //到 243 行结束

}
void l3g4200d()
{
    uint8 buff[2];
    while(buff[1]!=0xbf)
    {
        buff[0]=0x20; //写 CTRL_REG1 寄存器  0010 0000
        buff[1]=0xbf;       //写入 CTRL_REG1 的值   1011 1111  高 4 位  10 11--ODR 400Hz
cut-off 110
        spi_mosi(SPI0, SPIn_PCS0, buff, NULL, 2);

        buff[0]=0xa0; //读 CTRL_REG1 寄存器  1010 0000
        buff[1]=0x00;       //读 CTRL_REG1 的值   1011  1111  高 4 位  10  11--ODR  400Hz
cut-off 110
        spi_mosi(SPI0, SPIn_PCS0, buff, buff, 2);
    }
```

```
    while(buff[1]!=0x10)
    {
        buff[0]=0x23; //写 CTRL_REG4 寄存器
        buff[1]=0x10; //写入 CTRL_REG4 的值   0001 0000  设置满量程为 500dps
        spi_mosi(SPI0, SPIn_PCS0, buff, NULL, 2);

        buff[0]=0xa3; //读 CTRL_REG4 寄存器
        buff[1]=0x00; //读 CTRL_REG4 的值   0001 0000  设置满量程为 500dps
        spi_mosi(SPI0, SPIn_PCS0, buff, buff, 2);
    }
}
void FTM0(void)                         //电机
{
    PORTD_PCR4 = PORT_PCR_MUX(0x4)| PORT_PCR_DSE_MASK;     //FTM0 的 4 通道
    PORTD_PCR5 = PORT_PCR_MUX(0x4)| PORT_PCR_DSE_MASK;     //FTM0 的 5 通道
    PORTD_PCR6 = PORT_PCR_MUX(0x4)| PORT_PCR_DSE_MASK;     //FTM0 的 6 通道
    PORTD_PCR7 = PORT_PCR_MUX(0x4)| PORT_PCR_DSE_MASK;     //FTM0 的 7 通道

    SIM_SCGC6|=SIM_SCGC6_FTM0_MASK;             //使能 FTM0 时钟

    FTM0_C4SC |= FTM_CnSC_ELSB_MASK;
    FTM0_C4SC &= ~FTM_CnSC_ELSA_MASK;
    FTM0_C4SC |= FTM_CnSC_MSB_MASK;     //边沿对齐 pwm 模式

    FTM0_C5SC |= FTM_CnSC_ELSB_MASK;
    FTM0_C5SC &= ~FTM_CnSC_ELSA_MASK;
    FTM0_C5SC |= FTM_CnSC_MSB_MASK;

    FTM0_C6SC |= FTM_CnSC_ELSB_MASK;
    FTM0_C6SC &= ~FTM_CnSC_ELSA_MASK;
    FTM0_C6SC |= FTM_CnSC_MSB_MASK;     //边沿对齐 pwm 模式

    FTM0_C7SC |= FTM_CnSC_ELSB_MASK;
    FTM0_C7SC &= ~FTM_CnSC_ELSA_MASK;
    FTM0_C7SC |= FTM_CnSC_MSB_MASK;

    FTM0_SC = FTM_SC_PS(3) | FTM_SC_CLKS(1); //采用系统时钟，32 分频

    FTM0_MODE &= ~3;        //BIT0      FTM Enable
    FTM0_OUTMASK &= ~3;     //0 Channel output is not masked. It continues to operate normally.

    FTM0_COMBINE=0;         //Function for Linked Channels (FTMx_COMBINE)
    FTM0_OUTINIT=0;
```

```
    FTM0_EXTTRIG=0;          //FTM External Trigger (FTMx_EXTTRIG)
    FTM0_POL=0;              //Channels Polarity (FTMx_POL)

    FTM0_QDCTRL &=~FTM_QDCTRL_QUADEN_MASK;

    FTM0_INVCTRL=0;        //反转控制
    FTM0_SWOCTRL=0;            // 软 件 输 出 控 制 FTM Software Output Control (FTMx_SWOCTRL)
    FTM0_PWMLOAD=0;        //FTM PWM Load

    FTM0_CNTIN=0;          //Counter Initial Value
    FTM0_MOD=2500;       //Modulo value,The EPWM period is determined by (MOD - CNTIN + 0x0001)
                           //PMW 频 率 =X 系 统 频 率 /2/(2^FTM1_SC_PS)/FTM1_MOD=125000000/2/(2^6)/19531=50HZ
    FTM0_C4V=0;            //设置 the pulse width(duty cycle) is determined by (CnV - CNTIN).
    FTM0_C5V=0;
    FTM0_C6V=0;            //设置 the pulse width(duty cycle) is determined by (CnV - CNTIN).
    FTM0_C7V=0;
    FTM0_CNT=0;             //只有低 16 位可用
}
void FTM1_init(void)         //正交解码测速
{
    SIM_SCGC6 |= SIM_SCGC6_FTM1_MASK;

    PORTB_PCR0 = 0x00000603;//Enable QDM,pull up enabled
    PORTB_PCR1 = 0x00000603;

    FTM1_MODE = 0x04;  //Write protection disabled
    FTM1_MODE |= 0x01;  //FTM enabled
    FTM1_SC = 0x00;        //FTM 0 clock=System clock, divided by 1
    FTM1_CNT = 0;
    FTM1_MOD = 0xFFFF;//Top value is 65535
    FTM1_CNTIN = 0x0000;//Bottom value is 0
    FTM1_FILTER = 0x00; //Filter disabled
    FTM1_QDCTRL = 0x01;      //Enable quadrature mode   双相计数
}


void FTM2_init(void)       //正交解码测速
{
    SIM_SCGC3 |= SIM_SCGC3_FTM2_MASK;
```

```
    PORTB_PCR18 = 0x00000603;//Enable QDM,pull up enabled
    PORTB_PCR19 = 0x00000603;

    FTM2_MODE = 0x04;   //Write protection disabled
    FTM2_MODE |= 0x01;  //FTM enabled
    FTM2_SC = 0x00;         //FTM 0 clock=System clock, divided by 1
    FTM2_CNT = 0;
    FTM2_MOD = 0xFFFF;//Top value is 65535
    FTM2_CNTIN = 0x0000;//Bottom value is 0
    FTM2_FILTER = 0x00; //Filter disabled
    FTM2_QDCTRL = 0x01;       //Enable quadrature mode    双相计数
}

void initPIT0()
{
    //开启定时模块时钟
    SIM_SCGC6 |= SIM_SCGC6_PIT_MASK;
    // 开启  PIT
    PIT_MCR = 0x00;
    //Bus clock is 100M, PIT0 interrupt period is (24+1)/100000000=250ns
    PIT_LDVAL0 = 24;
    PIT_TCTRL0 |= PIT_TCTRL_TIE_MASK;
    PIT_TCTRL0 |= PIT_TCTRL_TEN_MASK;
    PIT_TFLG0 = 1;//Clear flag
    //enable_irq(68);
}
```