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：　　电子科技大学

队伍名称：　　开拓者

参赛队员：　　陈梦翔

　　杨武

　　张晨

带队教师：　　田雨

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全国大学生“飞思卡尔”杯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：陈梦翔

杨武

张晨

带队教师签名：田雨

日　期：2012 年 8 月 8 号

第一章 引言

全国大学生“飞思卡尔”杯智能汽车竞赛是在规定的模型汽车平台上，参赛选手须使用竞赛秘书处统一指定的竞赛车模套件，采用飞思卡尔半导体公司的 8 位、16 位、32 位微控制器作为核心控制单元，自主构思控制方案进行系统设计，包括传感器信号采集处理、电机驱动、转向舵机控制以及控制算法软件开发等，完成智能车工程制作及调试，于指定日期与地点参加各分（省）赛区的场地比赛，在获得决赛资格后，参加全国决赛区的场地比赛。参赛队伍的名次（成绩）由赛车现场成功完成赛道比赛时间来决定，参加全国总决赛的队伍同时必须提交车膜技术报告。大赛根据车模检测路径方案不同分为电磁、光电与摄像头三个赛题组。车模通过感应由赛道中心电线产生的交变磁场进行路经检测的属于电磁组；车模通过采集赛道图像（一维、二维）或者连续扫描赛道反射点的方式进行进行路经检测的属于摄像头组；车模通过采集赛道上少数孤立点反射亮度进行路经检测的属于光电组。

技术报告以智能小车的设计为主线，包括小车的构架设计、软硬件设计，以及控制算法研究等，共分为六章。其中，第一章为引言部分；第二章主要介绍了小车的总体方案的选取，对单片机资源的分配作了说明。第四章对小车的硬件设计进行了详细的介绍，主要介绍了电路的设计；第五章描述了小车的软件设计和相关算法。第六章中叙述了我们在设计过程中遇到的问题和解决方法。

第二章　方案设计

2.1 整车设计思路

此次摄像头队采用竞赛秘书处统一指定的A型竞赛车模套件，采用飞思卡尔32位微控制器Kinetis系列的MK60DN512ZVLL10作为核心控制单元，自主构思控制方案及系统设计，包括传感器信号采集处理、控制算法及执行、动力电机驱动、转向舵机控制等，完成智能车工程制作及调试.

2.2 路经检测方案设计

透镜成像组用摄像头识别路径，同时也可以结合红外传感器辅助摄像头检测路径，从而有两种方案：

方案一：单独采用摄像头进行路径检测。

摄像头通过面成像，可以对车前方足够的赛道信息进行采集，有利于赛车行驶时对前方赛道进行有效的预测，实现最优路径的选择。摄像头采集图像的这个优点尤其在赛车高速行驶准备入弯时表现得突出，它可以精确判断出不同的弯道从而赛车可采用不同的行驶策略，这对于舵机反应的滞后性有很大的改进。单独采用摄像头检测路径的缺点是摄像头采集路面信息的频率较低，容易受到光线和赛道外物体的干扰，尤其是对起跑线检测时处理算法要进行优化。

方案二：采用摄像头和红外传感器进行路径检测。

红外传感器的电路和软件设计简单，使用红外传感器对路径的检测受光线的干扰较少而且采集的信息可靠，还可以获得很高的检测频率，弥补了摄像头检测频率不高、易受干扰的缺点，在检测路径和起跑线时可以得到比较可靠的信息。但是对黑线的检测精度有限而且红外传感器的作用距离有限，其前瞻距离不可能很远，同时红外传感器的功耗大，电路的重量也不小，严重影响了赛车的启停能力和速度。通过理论论证和现场测试，我们发现只要我们在起跑线

对算法进行优化，完全可以解决起跑线的问题，在其他地方我们只要在算法里进行去干扰处理，可以解决干扰问题，没有必要再用到红外传感器。

最终我们采用方案一。

摄像头方案的选择：

1.采用 CCD 摄像头。CCD 摄像头优点是成像质量好，特别是动态效果比 CMOS 摄像头的效果要好很多。但是 CCD 摄像头的功耗比 CMOS 摄像头要高，工作电流有 100mA 左右。

2.采用 CMOS 数字摄像头。CMOS 摄像头功耗较低，工作电流只有 10mA 左右。并且 CMOS 摄像头出来直接是数字信号，对单片机处理速度有很大的帮助。

综上所述，我们采用 CMOS 数字摄像头。

2.3 速度控制总体方案的设计

在保证赛车稳定性的前提下，提高速度是获胜的关键，也是我们设计的重点。同时赛车的重量和重心的调整也是我们设计时要考虑的问题。对赛车速度的控制主要有两种方案。即：开环控制和闭环控制。

方案一：开环控制

开环控制是指没有反馈的控制。即通过预先设定的方案，没有外部反馈地进行操作。优点在于操作和控制比较简单，只需要提供理论运行的过程然后编程调整即可实现。缺点在于理论和实际始终有一定的误差，实践证明开环控制的精度不高，无法切实有效的提高速度。

方案二：闭环控制

闭环控制是指具有反馈的控制。在系统运行过程中，需要不断检测赛车速度的状态，与预期的状态进行比较，当相差到一定程度时，修正误差，精度很高。但是缺点在于电路的搭建和程序的编制都比开环控制要复杂。实现电机的闭环控制传感器主要是采用旋转编码器，在电机转动一定角度的时候形成脉冲，由外部器件捕获这些脉冲，得出与实际运行的差异。开环速度控制实现

起来比较简单，但速度会随着电池电压的变化而变化，不能实现对速度的精确控制。为了使小车能以不同的速度通过不同的弯道，精确的速度控制是关键。

综上所述，采用速度闭环控制方案。

2.4 转向控制方案

转向模块主要由舵机实现，舵机的响应速度和舵机臂长决定了转向控制的实时性。舵机的响应速度与驱动电压和控制舵机的 PWM 波周期有联系，通过查阅相关资料得知，在电压允许条件下，驱动电压越高舵机的响应速度越快。我们的初步方案是用电池电压直接供电舵机，但调试的过程中，我们发现电池电压值很不稳定，工作一段时间会电压值会下降，这样会导致舵机臂中心值会便宜，这样车的控制会很受影响，所以我们舵机的驱动电压定位官方标定值 6V，响应速度满足我们的要求，我们把控制舵机的 PWM 波频率调整为 60HZ，把舵机臂长加长，整体综合下来，舵机的响应速度有了很大提高，实现了对赛车的快速转向控制。

第三章　智能车机械改造设计

智能小车的机械性能对于其行驶表现具有非常重要的影响，任何控制算法和软件程序都需要通过智能小车的机械结构来执行和实现。为使模型车在比赛中发挥出最佳性能，使其直线行驶高速稳定，入弯转向灵活，结合现代汽车控制理论对智能车的运动特性进行分析，并据此对智能小车的机械结构进行相应的调整和参数优化。

3.1 车体机械建模

智能汽车竞赛摄像头组专用车模

图 3.1

要使赛车跑出好成绩，除了算法的优化，还要调整赛车的机械结构。机械结构的调整是一个需要通盘考虑的问题，赛车的机械性能对小车行驶性能有很大的影响。安装时需要考虑的要点是：

（1）符合组委会规定的赛车的尺寸要求

（2）安装的可靠性

（3）安装的轻便性

（4）方便摄像头准确快速的检测

（5）车体各部分重量的分配。质心问题，保证赛车转弯、加速在安装车模

与车模结构改造过程中，通过不断的调试摸索，经过对比之后，我们对小车的机械结构进行了改进，提高了小车的过弯性能和行驶的稳定性，达到了满意的效果。下面分别进行介绍。

3.2 车模安装

从拿到车模开始，车模安装便是第一项很重要的工作，车模零件比较多并且很繁琐，安装时要有细心和耐心。

首先，认真阅读车模安装说明书，从中了解各零件的编号和作用，安装步骤，车模的基本结构蓝图等。

其次，根据说明书安装。找一个比较宽敞得桌面并在上铺上一块比较大的抹布，把各零件袋和安装工具准备齐全，还可以把比较小且常用的零件放在桌面上，这样既可以提高安装效率，也可以防止安装过程中零件的滑落，丢失。

安装结束后，对小车各部分进行检查和调整，主要是以下几个方面：

（1）驱动电机齿轮传动机构，确保齿轮咬合紧密，顺畅。

（2）前轮转向机构，传动杆与舵机连接可靠，舵机工作时无晃动。

（3）轮胎，胎内海绵垫平铺无褶皱，轮胎轮毂与表面橡胶可靠固定。

（4）前后轮主轴合适固定，减少车轮滚动阻力。

3.3 机械结构调整

在比赛规则允许范围内，通过对小车机械结构进行改造，可以减少软件的复杂性，也可以提高小车行驶稳定性和过弯性能。

3.4 光电编码器

我们采用增量式光电编码器实现对驱动电机转速的检测，通过齿轮传动的方式将测速电机上小齿轮与差速齿轮啮合，可以实时地获得准确的运行速度。最开始我们考虑采用 100 线的小型编码器，该编码器的外径只有 18㎜，安装方便，重量也很小。但是，经过队伍商量以后发现相信随着后期车速的不断地提

高，对测速精度的要求也越来越高，再加上模型车本身震动对编码器的影响，100 线的小编码器已不能满足需求。所以我们又一次考虑和选择安装较大编码器的方式。最终决定选用 500 线 AB 两相光电编码器，当采样时间在 2.5MS 的时候，可以返回 100 多个脉冲，这样使 PID 控制精度足够。

3.3.1 舵机安装方式调整

（1）舵机竖直安装。

（2）加长并加工舵机输出力臂。其原因主要是舵机转动一定角度有时间延迟，时间延迟正比于旋转过的角度。舵机的响应速度直接影响模型车的过弯的最高速度，加长输出力臂，舵机转动很小的角度便可使前轮转动比较大的转角，提高了舵机的响应速度。这时，两前轮的连杆水平，舵机在转动时有最大的输出力矩，转向很轻便。

3.3.2 前轮机械的调整

在车模过弯时，转向舵机的负载会因为车轮转向角度增大而增大。为了尽可能降低转向舵机负载，对前轮的安装角度，即前轮定位进行了调整。前轮定位的作用是保障汽车直线行驶的稳定性，转向轻便和减少轮胎的磨损。前轮是转向轮，它的安装位置由主销内倾、主销后倾、前轮外倾和前轮前束等 4 个项目决定，反映了转向轮、主销和前轴等三者在车架上的位置关系。

主销后倾角是前轮主销与前轮垂直中心线之间的夹角，也就是主销上端向后倾斜的角度。

主销内倾角是前轮主销在赛车水平面内向内倾斜的角度，虽然增大内倾角也可以增大回正的力矩，但增大内倾角会在赛车转向的过程中，增大赛车与路面的滑动，从而加速轮胎的磨损，由于轮胎对地的附着力对防止侧滑有很重要的影响，所以如果轮胎磨损则得不偿失，所以内倾角调整为 1°。

前轮外倾角是前轮的上端向外倾斜的角度，如果前面两个轮子呈现“V”字形则称正倾角，呈现“八”字则称负倾角。由于前轮外倾可以抵消由于车的

重力使车轮向内倾斜的趋势，减少赛车机件的磨损与负重，所以赛车安装了组委会配备的外倾角为1°。

前轮前束是前轮前端向内倾斜的程度，当两轮的前端距离小后端距离大时为内八字，前端距离大后端距离小为外八字。由于前轮外倾使轮子滚动时类似与圆锥滚动，从而导致两侧车轮向外滚开。但由于拉杆的作用使车轮不可能向外滚开，车轮会出现边滚变向内划的现象，从而增加了轮胎的磨损。前轮外八字与前轮外倾搭配，一方面可以抵消前轮外倾的负作用，另一方面由于赛车前进时车轮由于惯性自然的向内倾斜，外八字可以抵消其向内倾斜的趋势。外八字还可以使转向时靠近弯道内侧的轮胎比靠近弯道外侧的轮胎的转向程度更大，则使内轮胎比外轮胎的转弯半径小，有利与转向。

3.3.3 后轮距及后轮差速的调整

差速机构的作用是在车模转弯的时候，降低后轮与地面之间的滑动；并且还可以保证在轮胎抱死的情况下不会损害到电机。当车辆在正常的过弯行进中(假设：无转向不足亦无转向过度)，此时 4 个轮子的转速(轮速)皆不相同，依序为：外侧前轮＞外侧后轮＞内侧前轮＞内侧后轮。差速器的特性是：阻力越大的一侧，驱动齿轮的转速越低；而阻力越小的一侧，驱动齿轮的转速越高以此次使用的后轮差速器为例，在过弯时，因外侧前轮轮胎所遇的阻力较小，轮速便较高；而内侧前轮轮胎所遇的阻力较大，轮速便较低。

由于速度高，赛车在转弯时容易翻倒，为了增加整车的平衡能力，通过改装后轮连接臂上的螺孔位置来调节轮距，从而增加的小车的稳定性。

3.3.4 底盘及车辆重心调整

车辆底盘高度越低，车辆重心越低，后轮抓地力越好，前轮转越敏感。因此在很多赛车比赛中，提高速度有效方法就是降低底盘高度。

另外，我们将电路板做得非常小，刚好能放在舵机和电池架中间的底板上，

这样，不但可以压低模型车的重心，防止高速过弯时翻车，也可以使纵向重心前移，增加转弯性能。同时也可以减轻模型车的质量，让赛车能在最短时间内加到最大速度。原始车模得重心大概在车的中后部，在电池架的中间。我们在调试的过程中发现模型车过弯时，转向往往不足，不够灵敏。我们在装旋转编码器时也把其装的很低，以降低赛车的重心，提高赛车行驶的稳定性。除了对车辆重心纵向的调整之外，车辆重心的前后方向调整，对赛车行驶性能，也有很大的影响。根据车辆运动学理论，车身重心前移，会增加转向，但降低转向的灵敏度(因为大部分重量压在前轮，转向负载增大)，同时降低后轮的抓地力；重心后移，会减少转向，但增大转向灵敏度，后轮抓地力也会增加。因此，调整合适的车身重心，让车模更加适应比赛赛道很关键。为了保证赛车足够的转向和防止在高速入弯时出现甩尾现象，我们让重心大概在赛车的中心，在过弯时，使其前后轮的侧向摩擦力大体相当，从而提高过弯性能和稳定性。

3.4 传感器的安装

3.4.1 摄像头的安装

安装摄像头时，要考虑的因素很多。安装过高时视野比较宽，黑线变得很细甚至在图像的远端根本采集不到黑线。同时受到的干扰和抖动都很强烈，过弯时也容易翻车。安装过低，视野变小，图像变形严重，而且容易反光。在过急弯时“丢线”现象比较严重。摄像头支架安装在模型车的前与后也有不同的效果，安装在前面，过弯时赛车甩的很严重，摄像头抖的厉害，模型车高速行驶时，稳定性能不是很好。安装在中间时，摄像头比较稳定，转弯时几乎不丢线，能正确把握路面信息，提高小车行驶的稳定性和过弯性能。

考虑到以上因素，我们将摄像头安装在模型车的中间稍微靠前一点，安装的高度大概在距底板 20cm 的位置，这样能够满足前瞻性好，图像变形不是很严重，视野足够宽的要求。

3.4.2 旋转编码器的安装

旋转编码器在安装时主要考虑其能否和电机齿轮很好的咬合，不至于速度较快时，出现速度检测不准的现象。我们把旋转编码器齿轮通过一个大齿轮传动和电机齿轮直接咬合，减小误差，同时把其安装在靠近底板位置以降低重心位置，增强后轮的抓地能力。

第四章 智能车的电路设计及实现

赛车共包括六大模块：MK60DN512ZVLL10 主控模块、传感器模块、电源模块、电机驱动模块、速度检测模块和辅助调试模块。

4.1 各模块的作用

MK60DN512ZVLL10 主控模块：智能车系统以 MK60DN512ZVLL10 为控制核心，将采集摄像头、编码器等传感器的信号，根据控制算法做出控制决策，驱动直流电机和舵机完成对智能汽车的控制并实现了单片机硬件的最优化设计和单片机资源的合理化使用。

摄像头模块：可以通过一定的前瞻性，提前感知前方的赛道信息，为智能汽车做出决策提供必要的依据和充足的反应时间。

电源管理模块：为整个系统提供合适而又稳定的电源。
电机驱动模块：驱动直流电机和伺服电机完成智能汽车的加减速控制和转向控制。
速度检测模块：检测反馈智能汽车轮的转速，用于速度的闭环控制。
辅助调试模块：主要用于智能汽车系统的功能调试、赛车状态监控。

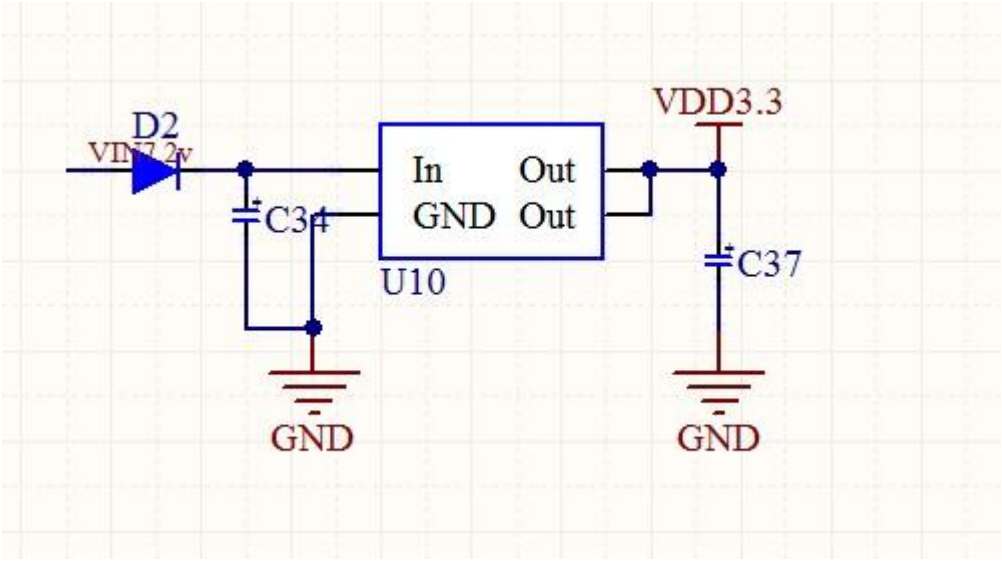
4.2 电源模块

智能车系统采用配发的标准的车模用的 7.2V2000mAhNi-cd 蓄电池进行供电，但各个模块所需要的电压不同，因此需要进行电压调节。电源系统的好坏直接关系到整个系统的稳定性。

4.2.1 3.3V 稳压模块

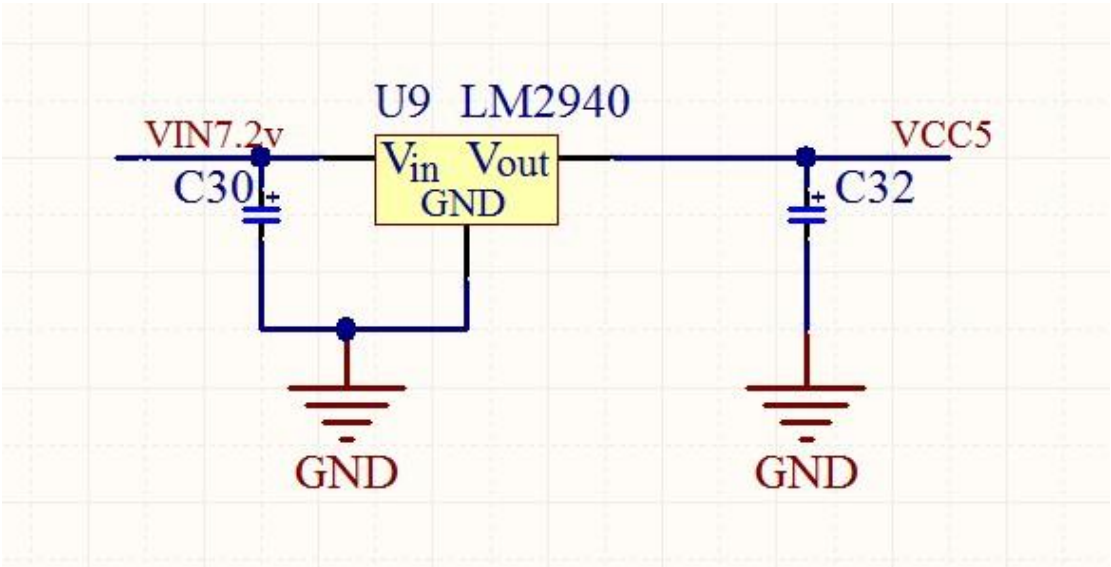
3.3V 主要用于单片机的供电。

电机和舵机的突然启停会产生尖峰脉冲使电池电压骤变，一般会把电源电压拉低 1V 多，会对系统电源造成干扰。所以系统的电源必须有一定的抗干扰能力。鉴于开关电源纹波比较大，而线性稳压电源纹波很小，我们选择了使用线性电源，其中我们选择 LM1117-3.3 线性稳压芯片。



4.2.2 5V 稳压模块

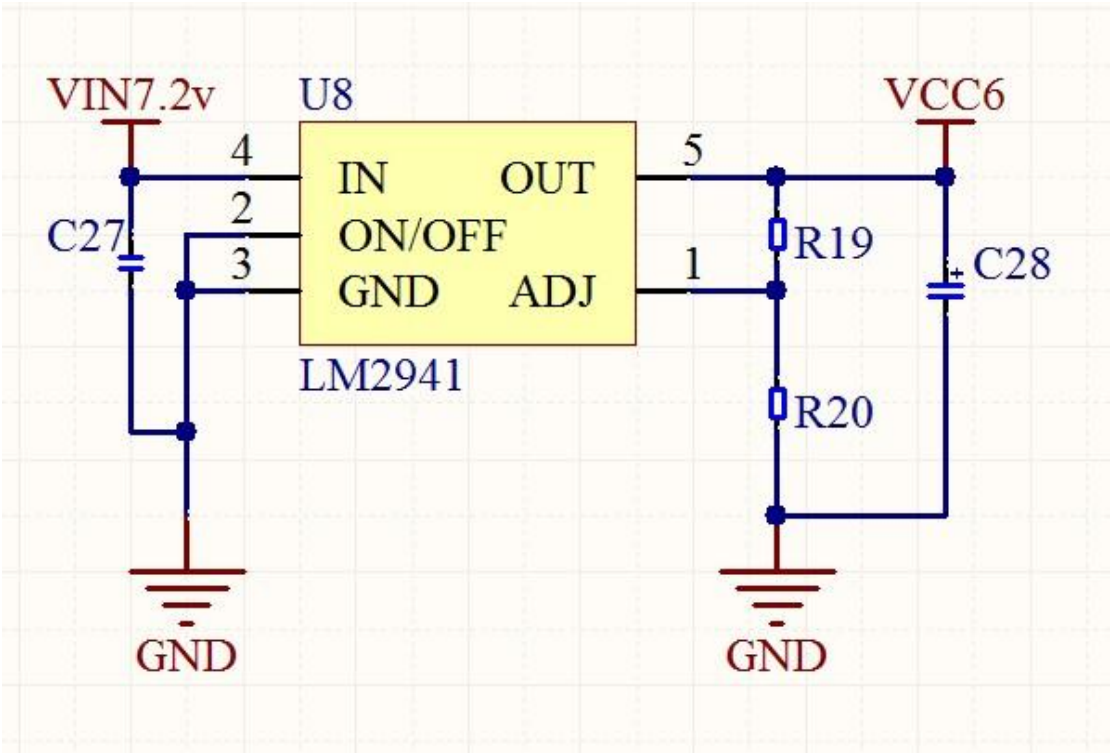
5V 主要用于编码器和摄像头的供电。稳压芯片主要有线性稳压芯片和开关稳压芯片两种，摄像头电源要求比较高，电源纹波必须要小。鉴于开关电源纹波比较大，而线性稳压电源纹波很小，我们选择了使用线性电源，其中我们选择 LM2940 线性稳压芯片。



4.2.3 6V 稳压模块

6V 主要用于舵机的供电。

舵机电源需要稳定电压供电，所以我们选择官方标准的 6V 电压供电，稳压芯片选择 LM2941，这个电源芯片可调，完全满足我们的需要。



4.3 图像采集模块

在电机启动、停止、加速、减速以及舵机的启动、停止时，都会对电源电压产生影响，使电源产生一定的波动。为了解决这一问题，我们专门在 LM2940 电源输入引脚前串一功率电感，可有效降低电源波动对 5V 稳压芯片的干扰。原理图如图 4.1：

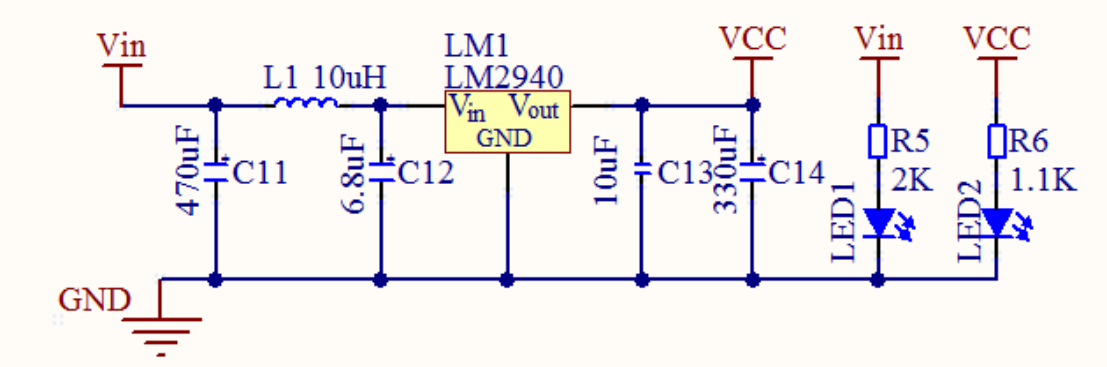


图 4.1

4.3.1 摄像头简介

摄像头分黑白和彩色两种，为达到寻线目的，只需提取画面的灰度信息，而不必提取其彩色信息，所以我们所使用的 CMOS 摄像头输出的信号为黑白视频信号。我们所选用的摄像头芯片为 OV7620，1/3 英寸数字式 CMOS 图像传感器 OV7620,总有效像素单元为 664(水平方向)×492(垂直方向)像素;内置 10 位双通道 A/D 转换器,输出 8 位图像数据;具有自动增益和自动白平衡控制,能进行亮度、对比度、饱和度、γ校正等多种调节功能;其视频时序产生电路可产生行同步、场同步、混合视频同步等多种同步信号和像素时钟等多种时序信号;5V 电源供电,工作时功耗<120mW,待机时功耗<10μW。可应用于数码相机、电脑摄像头、可视电话、第三代网络摄像机、手机、智能型安全系统、汽车倒车雷达、玩具，以及工业、医疗等多种用途。

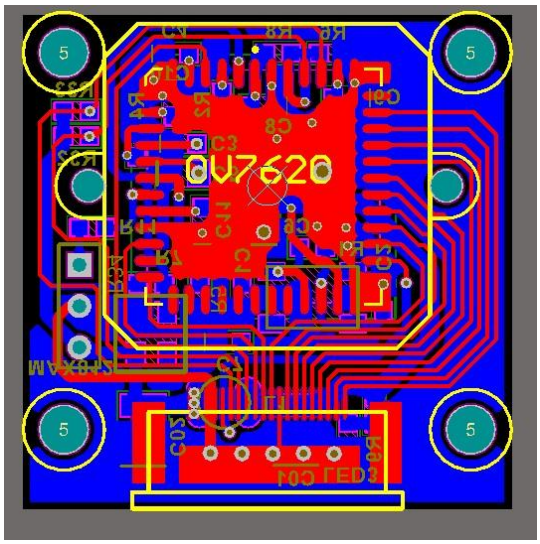
OV7620 是 1/3”CMOS 彩色 / 黑白图像传感器。它支持连续和隔行两种扫描方式，VGA 与 QVGA 两种图像格式；最高像素为 664×492，帧速率为 30fps；数据格式包括 YUV、YCrCb、RGB 三种，能够满足一般图像采集系统的要求。

4.3.2 电路设计

由于我们采用的是数字摄像头，相对于 CCD 来说电路相对简单，而且市场上买的摄像头大多是个人制作，稳定性不好，质量参差不齐，接口不一致，所

以我们自己设计摄像头。

PCB 板图如下：



4.4 速度采集模块

常用的有以下几种方案：

1）使用光电对管配合光码盘

参照鼠标的原理，在后轴上安装一个光码盘，转动时码盘的齿会遮挡红外线，使红外管输出一系列的脉冲。

2)使用霍尔传感器配合稀土磁钢

在齿轮上装上几棵稀土磁钢，将霍尔元件安装在附近，将轮子转动时磁场的变化转化为电脉冲，从而获取轮子的转速。

3)使用旋转编码器。

市售的旋转编码器，每转动一圈可以输出几百个脉冲，通过检测脉冲数就可以计算出车速。

第一种方案安装比较方便，转动一周能输出几十个脉冲。但是在速度较快时，其精度会逐渐降低，对车速不能进行精确的控制。第二种方案安装比较复杂，而且转动一圈只能输出很少的脉冲，调速时不够精确。第三种方案安装

比较方便，输出的是标准的方波，应用时精度非常高，而且几乎不需要辅助电路便可完成速度检测。

所以我们选择第三种方案。

4.5 电机驱动模块

我们选用了驱动芯片 BTS7970B，其为高强度电流的半桥电机驱动芯片。我们利用两片 BTS7970B 构成一个完整的全桥驱动，可以很好实现电机的正转、反转、刹车制动。电路原理图如图 4.3：

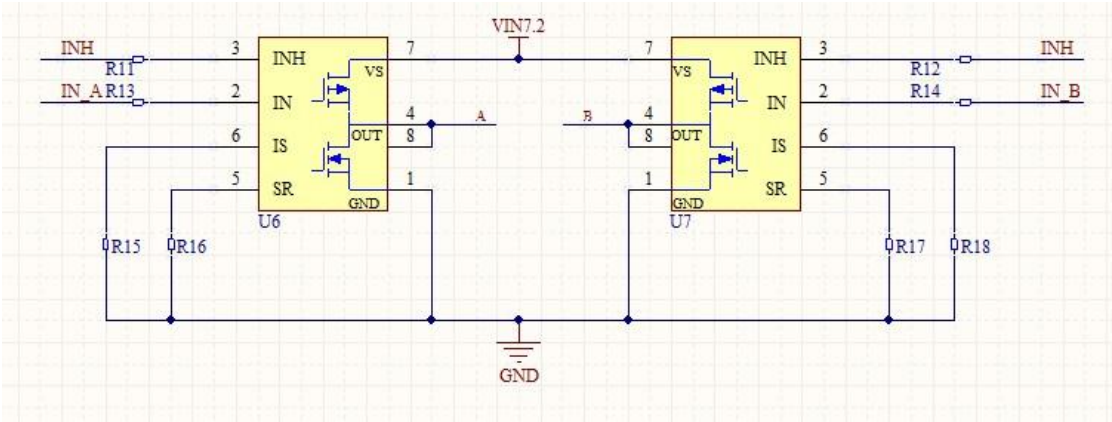


图 4.3

4.6 车模参数汇总

项目	参数
路径检测方法（赛题组）	摄像头组
车模几何尺寸（长、宽、高）（毫米）	280 *160 *230
车模轴距/轮距（毫米）	200*160
车模平均电流（匀速行驶）(毫安)	1300
电路电容总量（微法）	1700
传感器种类及个数	摄像头：1 个 红外管：4 个 陀螺仪：1 个
新增加伺服电机个数	1
赛道信息检测空间精度（毫米）	20
赛道信息检测频率（次/秒）	60

主要集成电路种类/数量	**电源　驱动　摄像头 发射接收管　编码盘　陀螺仪**
车模重量（带有电池）（千克）	1.7

第五章 软件设计及控制策略与算法

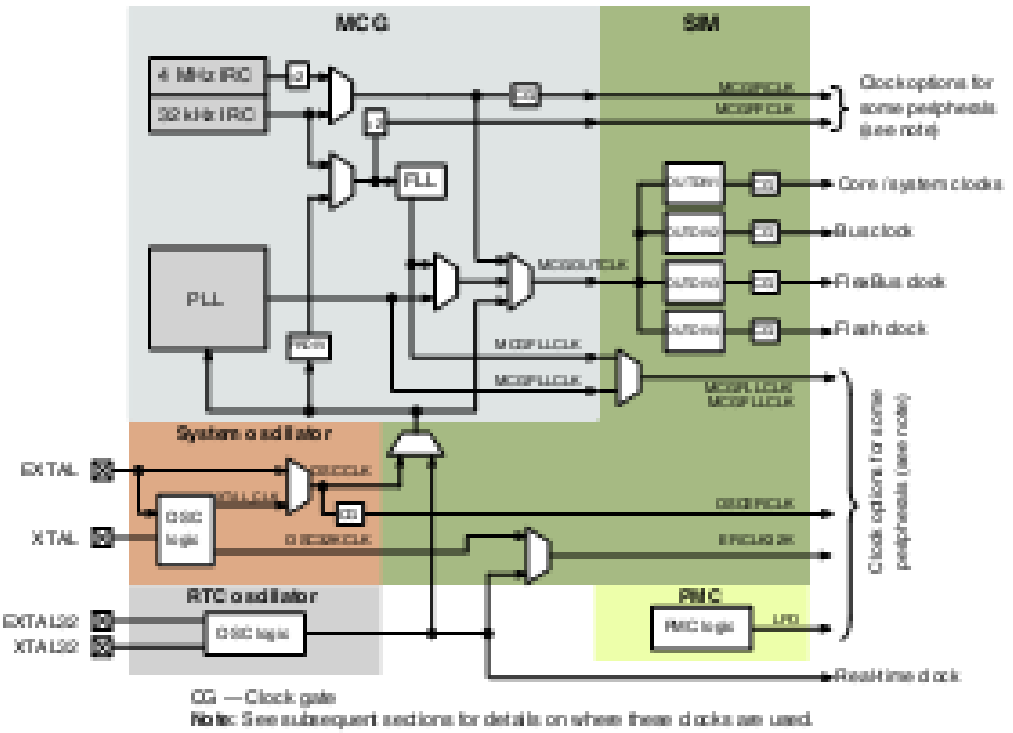
5.1 软件系统设计

按照比赛规则要求，采用飞思卡尔 32 位微控制器 MK60DN512ZVLL10 作为唯一核心控制芯片。

Kinetis 系列微控制器是飞思卡尔公司于 2010 年下半年推出的基于 ARM Cortex-M4 内核的微控制器，是业内首款 Cortex-M4 内核芯片。Kinetis 系列微控制器采用了飞思卡尔 90 纳米薄膜存储器 (TFS) 闪存技术和 Flex 存储器功能(可配置的内嵌 EEPROM)，支持超过 1000 万次的擦写。Kinetis 微控制器系列融合了最新的低功耗革新技术，具有高性能、高精度的混合信号能力，宽广的互连性，人机接口和安全外设。

5.1.1 时钟模块

K60 芯片的时钟系统由振荡器（OSC)、实时振荡器（RTC OSC)、多功能时钟发生器（MCG)、系统集成模块（SIM）和电源管理器（PMC）等模块组成，其中，OSC 和 RTC OSC 模块通过外接的晶体振荡器为系统引入外部参考时钟信号，MCG 模块为系统中的各个模块分配时钟源，SIM 模块为系统中的各个模块选择时钟源，PMC 模块可输出 1KHZ 的参考时钟信号。时钟系统框图如下所示：



综合考虑可靠性，舵机、电机频率要求和 dma 速度要求后，我们决定配置内核时钟为 96M，总线时钟为 48M，FLASH 时钟 24M。

5.1.2 DMA 模块

K60 内含增强型 DMA 模块（eDMA），支持软硬件触发传输，支持可编程的源地址、目的地址、传输字节数等特性。我们选择使用 dma 进行摄像头的数据采集，配置由 PCLK 上升沿触发 dma 读取对应 io 口，相比于采用 fifo 的方案，可以大大提高时效性，同时也没有直接读 io 口的噪点问题。

5.1.3 FTM 模块

FTM 模块是一个多功能定时器模块，主要功能有，PWM 输出、输入捕捉、输出比较、定时中断、脉冲加减计数、脉冲周期脉宽测量。在 K60 中，共有 FTM0，FTM1，FTM2 三个独立的 FTM 模块。其中 FTM0 有 8 个通道，可用于电机或舵机的

PWM 输出，但不具备正交解码功能，也就是对旋转编码器输入的正反向计数功能。而 FTM1 和 FTM2 则具备正交解码功能，但是 FTM1 和 FTM2 各只有两个通道。FTM 模块的时间基准来自一个 16 位的计数器，该计数器的值可读取，即可作为无符号数对待，也可作为有符号数的补码对待。

基于此，我们选择配置 FTM2 的 0 通道作为舵机驱动的输出口，FTM0 的 0 通道和 1 通道作为电机正反转驱动的输出口，FTM1 配置为正交解码模式，作为编码器信号的输入口。

5.1.4 中断系统

Kinetis 中断优先级共分 16 级，所有 GPIO 口均可触发中断，这给硬件布线和软件编程提供了巨大的方便。我们选择使用 PTC12，PTC13 两个引脚分别接入摄像头的行中断信号和场中断信号，用以确定要采集的图像数据时序。

5.1.5 UART 模块

使用 UART 模块连接蓝牙串口用以辅助调试。

5.1.6 SPI 模块

使用 SPI 接口外接 sd 卡用来存储图像。Kinetis 内置 SDHC 模块，但是考虑到使用 SPI 接口访问 sd 卡的资料较多，调试简单，我们依然选择使用 SPI 接口。

5.2 系统资源分配

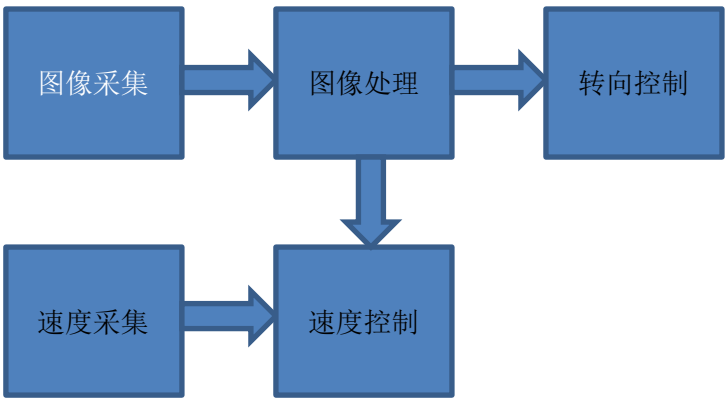
摄像头部分：	场中断：PTC12 行中断：PTC13

	像素时钟：PTD17 数据接口：PTD0-7
舵机部分：	PWM 信号输入：FTM2CH0
电机部分：	正转 PWM：FTM0CH0 反转 PWM：FTM0CH1 BTS7970 使能：PTB3
蓝牙串口部分：	UART1
OLED 屏幕： （采用 GPIO 模拟 SPI）	DC：PTB11 RST：PTB16 SPI_CLK：PTB10 SPI_DATA：PTB9
按键：	KEY_UP: PTB20 KEY_DOWN: PTB21 KEY_LEFT: PTB23 KEY_RIGHT: PTB19 KEY_ENTER: PTB22 KEY_CANCEL: PTB17

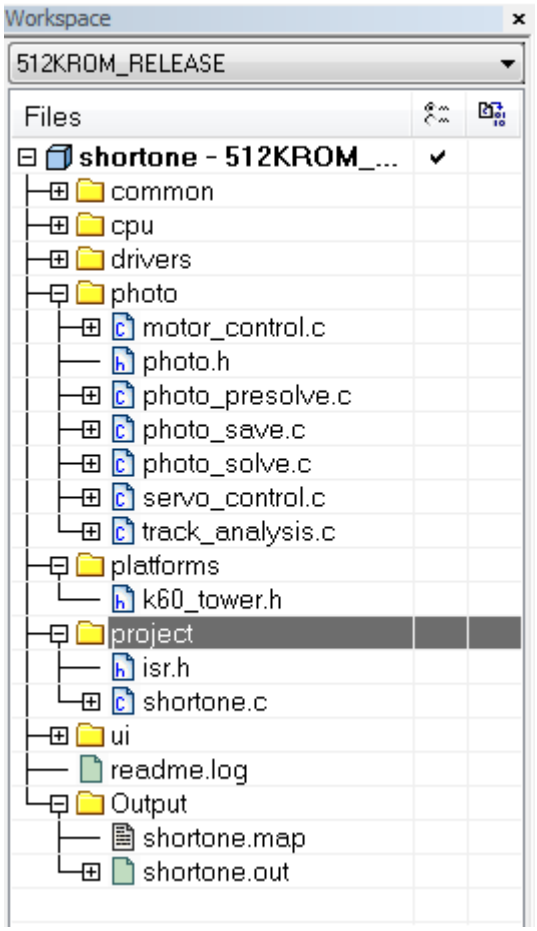
蜂鸣器：	PTE26
MicroSD 卡：	SD_CLK：SPI1_SCK MISO：SPI1_SIN MOSI：SPI1_SOUT CS：SPI1_PCS0

5.3 算法架构

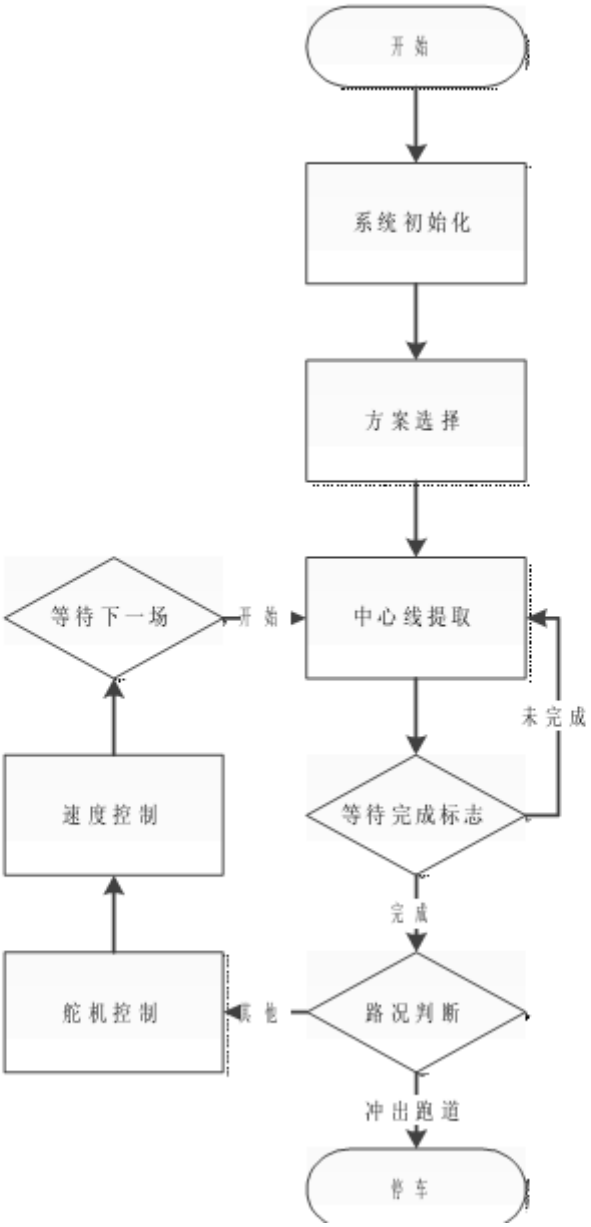
整个软件系统要实现的功能就是根据采集到的跑道数据和速度数据对赛车舵机跟电机作出一系列的控制，使赛车沿着跑道高速行驶。按功能划分整个软件系统可以分为图像采集模块、图像处理模块、转向控制模块、速度采集控制模块、速度控制模块这五大模块。



工程结构组织：


Workspace
512KROM_RELEASE
Files
shortone - 512KROM_...
common
cpu
drivers
photo
motor_control.c
photo.h
photo_presolve.c
photo_save.c
photo_solve.c
servo_control.c
track_analysis.c
platforms
k60_tower.h
project
isr.h
shortone.c
ui
readme.log
Output
shortone.map
shortone.out

软件流程图设计：



5.3.1 图像处理部分

图像信号处理中提取的赛道信息主要包括：

（1）每一行的赛道中心位置

（2）每一行的赛道宽度

（3）赛道的曲率

（4）赛道的变化幅度

（5）赛道任一点的导数值

（6）赛道形式，包括直道、左弯、右弯、大 S 弯、小 S 弯等等

每幅图像，我们只对其中的 52 行进行采样，实际测试中 52 行的采样点已经能够为赛车提供足够的控制信息。

5.3.2 PID 控制器

PID 是一种线性控制器，采用输出量和参考输入的误差及其微分、积分的线性组合来产生控制信号。PID 控制具有控制结构简单，参数个数少而且容易确定，不必求出被控对象的精确性模型就可以整定参数的特点，在控制系统中应用极为广泛。

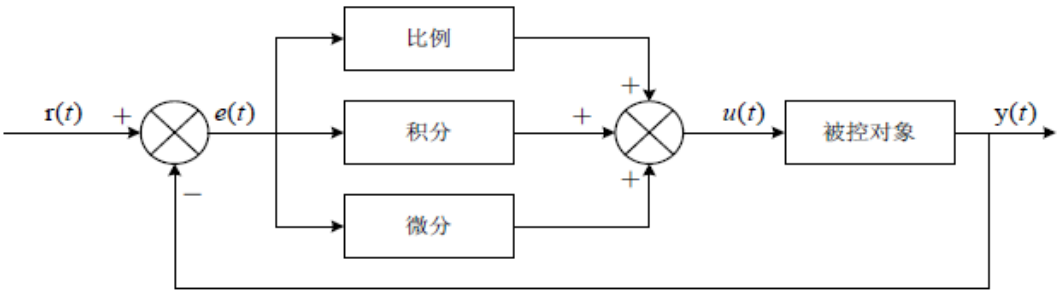
连续 PID 控制规律如下：

$$u(t) = Kp[e(t) + \frac{1}{Ti}\int_0^t e(t)dt + Td\frac{de(t)}{dt}]$$

Kp---控制器的比例系数

Ti---控制器的积分时间，也称积分系数

Td---控制器的微分时间，也称微分系数



模拟 PID 控制系统原理

1.比例部分

比例部分的数学式表示是：Kp*e（t）在模拟 PID 控制系统中，比例环节的作用是对偏差瞬间做出反应。偏差一旦产生，控制器立即产生控制作用，使控制量向偏差减少的方向变化。控制作用的强弱取决于比例系数 Kp，比例系数 Kp 越大，控制作用越强，则过度过程越快，控制过程的静态偏差也就越小；但是 Kp 越大，也月容易产生振荡，破坏系统的稳定性。故而，比例系数 Kp 选择必须恰当，才能过渡时间少，静差小而又稳定

2.积分部分

从积分部分的数学表达式可以知道，只要存在偏差，则它的控制作用就不断的增加；只有在偏差 e（t）=0 时，它的积分才能使一个常数，控制作用才是一个不会增加的常数。可见，积分部分可以消除系统的偏差。

积分环节的调节作用虽然会消除静态偏差，但也会降低系统的响应速度，增加系统的超调量。积分常数 Ti 越大，积分的积累作用越弱，但系统在过渡时不会产生振荡；积分常数 Ti 越小则积分的累积作用越强，但系统在过渡时却有可能产生振荡。所以必须根据具体要求来确定 Ti。

3.微分部分

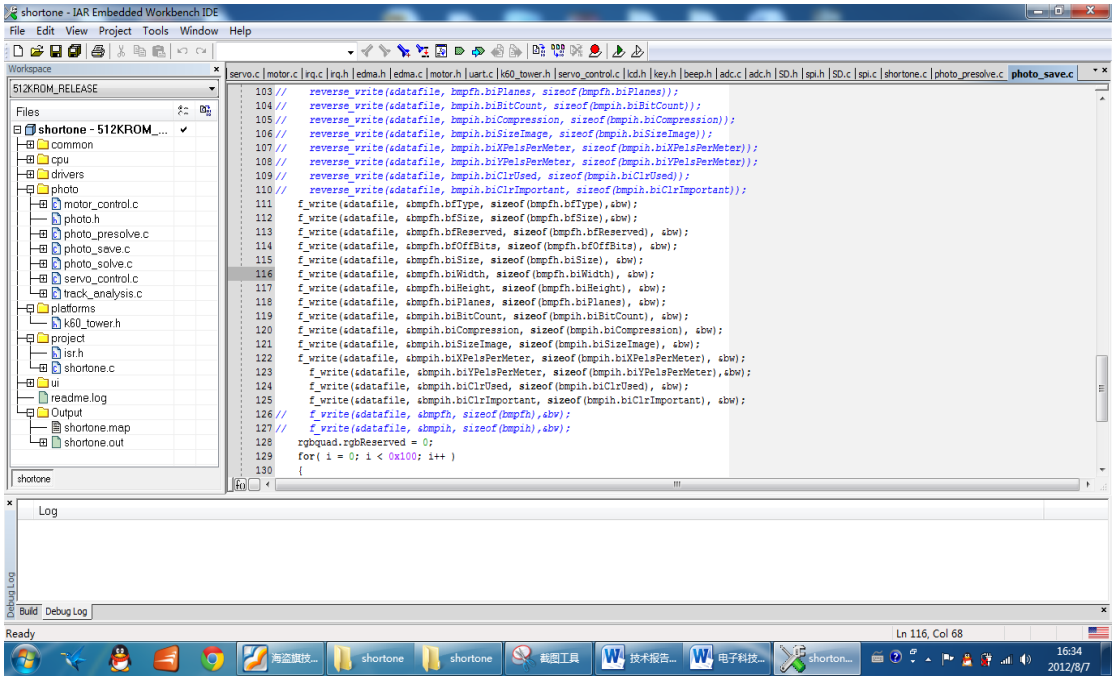
实际的控制系统除了希望消除静态误差外，还要求加快调节过程。在偏差出现的瞬间，或在偏差变化的瞬间，不但要对偏差量做出立即响应（比例环节的作用），而且要根据偏差的变化趋势预先给出适当的纠正。为了实现这一作用，可在 PI 控制器的基础上加入微分环节，形成 PID 控制器。

微分环节的作用使阻止偏差的变化。它是根据偏差的变化趋势（变化速度）进行控制。偏差变化的越快，微分控制器的输出就越大，并能在偏差值变大之前进行修正。微分作用的引入，将有助于减小超调量，克服振荡，使系统趋于稳定。

微分部分的作用由微分时间常数 Td 决定。Td 越大时，则它抑制偏差变化 e（t）的作用越强；Td 越小时，则它反抗偏差变化的作用越弱。微分部分显然对系统稳定有很大的作用。适当地选择微分常数 Td，可以使微分作用达到最优。

鉴于 PID 控制器具有良好地动态、稳态性能，我们选择对舵机进行 PD 控制，电机进行 PID 控制。

5.4 软件调试界面 ： IAR Embedded Workbench



IAR Embedded WorkBench 功能强大， 界面简洁严整，内置的 C 编译器编译产生的代码优化度高，执行效率很高。同时 IAR Embedded Workbench 具有强大的在线调试功能，可以充分满足智能车软件系统开发的需要。

第六章 报告总结

文中介绍了如下内容，智能车机械结构的设计和制作方法，以及如何将智能车优化为适合竞速的模型；智能车各个模块的工作原理和设计电路，如电源模块、电机驱动模块、测速模块等；智能车速度的 bang-bang 控制及 PID 控制算法和角度的模糊控制算法；系统开发过程中所用到的开发工具、软件以及调试过程。综合来看，智能车包括硬件和软件部分，是综合多学科知识的平台，对于我们专业课的学习和知识面的扩展有极大的帮助。

通过本次比赛，我们都学到了很多知识，对于汽车的机械结构、光电传感器、软件设计有了深入的了解，最关键的是掌握了一套较为完善的系统开发流程。

致谢

经过了近五个月的努力，我们终于完成了赛车制作。在此，对帮助过我们的老师同学表示深深的谢意。感谢飞思卡尔公司和主办方为我们提供了一个展示自我的舞台。感谢我们的指导老师田雨老师，是他们在后面默默无闻的支持我们，在参加比赛的过程中，为我们安排好一切，对我们关爱有加。感谢其他车队的同学，在和他们的交流中，使我们学到了很多东西。感谢电子科技大学为我们提供经费，场地，设备。最后，向审阅本报告的各位专家教授表示深深的谢意。

第七章 参考文献

【1】MEASUREMENTS & CONTROL1995 年 12 月作者：TAMARA BRATLAND ，ROBERT BICKING 和 BHARAT B. PANT
【2】新型 PID 控制及其应用 2002 年 作者：陶永华主编机械工业出版社
【3】控制系统仿真及 MATLAB 语言 2009 年 作者：吴忠强等电子工业出版社
【4】智能车专题培训 张树波
【5】学做智能车 2004 年 作者：邵贝贝等 北京航空航天大学出版社.

第八章 程序

```
/*servo_control.c*/
#include "servo.h"
#include "photo.h"
#include "irq.h"
#include "uart.h"
#include "common.h"
#include "lcd.h"
#include "motor.h"
#include "ui.h"
#include "beep.h"
//下面是舵机 PD 控制数据，第一个为 P，第二个为 D
int32 servo_P = 18;
int32 servo_D = 40;

static int32 get_deviation(int32 row, int32*find)
{
    int32 right_edge_valid = 0;
    int32 left_edge_valid = 0;
    int32 deviation;

    *find = 1;
    if( right_valid_line_num > 0 && right_edge_end >= row && right_edge_start
<= row )
    {
        right_edge_valid = 1;
    }

    if( left_valid_line_num > 0 && left_edge_end >= row && left_edge_start <=
row)
    {
        left_edge_valid = 1;
    }

    if( left_edge_valid == 1 && right_edge_valid == 1 )
    {
        deviation = (right_edge[row] + left_edge[row])/2 - PHOTO_CENTER;
    }
    else if( left_edge_valid == 1 )
    {
        deviation = left_edge[row] + half_plus[row]*5/4 - PHOTO_CENTER;
    }
    else if( right_edge_valid == 1 )
    {
```

```
            deviation = right_edge[row] - half_plus[row]*5/4 - PHOTO_CENTER;
        }
        else
        {
            *find = 0;
            deviation = 0;
        }
        return deviation;
}

void servo_control(int32 track_type)
{
        static int32 deviation[3] = {0};///0 为当前偏差，1 为前一次偏差, 2 为前两次偏
差
        int32 diff; ///偏差的微分
                int32 pwm_calculate = 0;
        int32 find = 0;
        int32 dev;
        int32 dev_cal = 2*deviation[1] - deviation[2];
                int32 start_row = 0;

                if (track_type == TRACK_TURN_S )
                {
           dev = get_deviation(control_row + 4, &find);
                }
                else if (track_type == TRACK_SLOPE)
                {
                                start_row = right_edge_start > left_edge_start ?
right_edge_start : left_edge_start;
                                if( start_row != -1 )
                                   dev                                          =
get_deviation(start_row,&find)*half_plus[control_row]/half_plus[start_row];
                }
                else
                {
                   dev = get_deviation(control_row, &find);
                }

                //lcd_cls();

        if( find == 0 )
        {
            deviation[0] = dev_cal;
        }
        else
```

```
    {
        deviation[0] = dev;
    }
    //      lcd_p8x16int(0, 2, deviation[0], '+');

            if( track_type == TRACK_START_STOP )
            {
                    deviation[0] = deviation[0]*2/3;
            }

            deviation[0] = (track_type == TRACK_TURN_S ) ? ((deviation[0]/3)*2):
deviation[0];
       //lcd_p8x16int(0, 0, deviation[0], '+');
     diff = deviation[0] - deviation[1];

            if( find != 0 )
            {
                deviation[2] =    deviation[1];
                deviation[1] = deviation[0]; ///更新前一次偏差的值
            }


            pwm_calculate = (servo_P*deviation[0] + servo_D*diff)/10 ;
//              if( pwm_calculate < 0 )
//                  pwm_calculate = pwm_calculate*7/6;
            pwm_calculate += SERVO_PWM_CENTER;

          // lcd_puttracktype(0, 0, track_type, '+');
    //        printf("%d ", pwm_calculate-SERVO_PWM_CENTER);
     pwm_calculate = pwm_calculate > SERVO_PWM_RIGHTMOST ?
SERVO_PWM_RIGHTMOST : pwm_calculate;
     pwm_calculate = pwm_calculate < SERVO_PWM_LEFTMOST ?
SERVO_PWM_LEFTMOST : pwm_calculate;

     SERVO_DRIVER((pwm_calculate));
        //    lcd_p8x16int(0, 6, pwm_calculate - SERVO_PWM_CENTER, '+');

}
```

```
/* motor_control.c */
#include "photo.h"
#include "motor.h"
```

```
#include "uart.h"
#include "encoder.h"
#include "lcd.h"
#include "common.h"

int32 motor_P = 90;
int32 motor_I = 4;
int32 motor_D = 15;


int32 track_speed[TRACK_TYPE_NUM] =
{
    320,                //speed for straight track
        220,                //speed for turn_ sharp
    150,                //speed for slope
    280,                //speed for turn_smooth
    250             //speed for turn_s
};
/*function name: motor_driver
*purpose：转速 pid 调节
*input parameter: 期望转速，P，I，D
                        y(t) = P*(delta+sigma*I+D*（delta-last_delta))
*output parameter：none
*return parameter：none
*/
void motor_control(int32 target_speed)
{

    static int32 last_sigma = 0;   //偏差累计项
    static int32 last_pwm = 0;      //上次控制值
    static int32 last_delta = 0;   //上次偏差值
    int32 current_speed = ENCODER_DATA;

    int32 current_delta = target_speed - current_speed;   //当前偏差
    int32 current_sigma;
    int32 current_pwm;

      //     uart_putchar(TERM_PORT,current_speed);
    if( last_pwm == MOTOR_PWM_MAX )
    {
        if( current_delta < 0 )
        {
            current_sigma = last_sigma + current_delta;
        }
        else
```

```
            {
                current_sigma = last_sigma;
            }
        }
        else if( last_pwm == MOTOR_PWM_MIN )
        {
            if( current_delta > 0 )
            {
                current_sigma = last_sigma + current_delta;
            }
            else
            {
                current_sigma = last_sigma;
            }
        }
        else
        {
            current_sigma = last_sigma + current_delta;
        }
        current_pwm = ( motor_P*current_delta + current_sigma*motor_I
                    + (current_delta - last_delta)*motor_D );
       current_pwm = current_pwm > MOTOR_PWM_MAX ? MOTOR_PWM_MAX :
current_pwm;
        current_pwm = current_pwm < MOTOR_PWM_MIN ? MOTOR_PWM_MIN :
current_pwm;

        motor_driver(current_pwm);
              ENCODER_CLEAR;
        last_sigma = current_sigma;
        last_pwm = current_pwm;
        last_delta = current_delta;
}
```

```
/*main.c*/
#include "common.h"
#include "sccb.h"
#include "irq.h"
#include "edma.h"
#include "servo.h"
#include "motor.h"
#include "uart.h"
```

```
#include "gpio.h"
#include "beep.h"
#include "ui.h"
#include "lptmr.h"
#include "photo.h"
#include "lcd.h"
#include "k60_tower.h"
#include "encoder.h"
#include "key.h"
#include "adc.h"
#include "sd.h"
#include "infrared.h"
#include "gyroscope.h"

int32 loop_counter = 0;
int32 sd_enable = 0;
void main(void)
{
        DisableInterrupts;
    beep_init();
    encoder_init();
        gyro_init();
        infrared_init();
    //  SD_Init();
          photosave_init();
        ui_init();
        lcd_p8x16str(0, 2, "ov initializing",'+');

    while( ov7620_reg_config() == 0)
    {
      beep(100);
    }

    //  sccb_test();
        lcd_p8x16str(0, 2, "ov init done",'+');
    //  time_delay_ms(200);

    main_setting();

    lcd_p8x16str(0,2,"running....",'+');
    time_delay_ms(2000);//delay for 2 sec
    beep(1);
        lcd_cls();
        motor_init();
    servo_init();
```

```
    dma_init();
        ENCODER_CLEAR;
    irq_init();
        EnableInterrupts;
        loop_counter = 0;
    for(;;)
    {
//      lcd_cls();
//                  if( GPIO_BIT_GET(PTC_BASE_PTR, INFRARED1_PIN) )
//                    lcd_p8x16int(0, 0, INFRARED1_PIN, '+');
//                  if( GPIO_BIT_GET(PTC_BASE_PTR, INFRARED2_PIN) )
//                    lcd_p8x16int(0, 2, INFRARED2_PIN, '+');
//                  if(   GPIO_BIT_GET(PTC_BASE_PTR, INFRARED3_PIN) )
//                    lcd_p8x16int(0, 4, INFRARED3_PIN, '+');
//                  if( GPIO_BIT_GET(PTC_BASE_PTR, INFRARED4_PIN) )
//                    lcd_p8x16int(0, 6, INFRARED4_PIN, '+');
                  loop_counter++;
                  beep_check();
        photo_presolve();
        photo_solve();

                  if( sd_enable ==   1 )
                    photo_saveasfile();
              /*
                lcd_cls();
                lcd_p8x16int(0, 0, adc_once(0, 1, 16), '+');
                *// printf("%d ", adc_once(0, 1, 16));
              //    photo_saveasfile();



        if( control_lost > 4 )
        {
            break;
        }
                  if( start_stop_counter >= 70 )
        {
            break;
        }
        irq_reset();
    }

    DisableInterrupts;
        BEEP_OFF;
        motor_driver(0);
        //time_delay_ms(1000);
```

```
    SERVO_DRIVER(0);

          sccb_test();
    lcd_cls();
    if( control_lost > 1 )
    {
        lcd_p8x16str(0, 0, "control lost!!!",'+');
    }

    if( start_stop_counter >= 70 )
    {
        loop_counter -= 70;
    }
    lcd_p8x16int(0, 2, loop_counter*100/60, '+' );
          while(key_get() != KEY_F )
             ;
          main_setting();
    //lcd_p8x16str(8*3, 2, "seconds", '+' );
    for(;;)
    ;

}
```