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：洛阳理工学院

队伍名称：洛阳理工摄像头一队

参赛队员：王秉鑫

韩　鹏

陈灿灿

带队教师：姚惠林

路　纲

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邀请赛关保留、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

摘　　要

在“飞思卡尔”杯全国大学生智能汽车竞赛成功举办六届的背景下，我们总结前几届的经验，积极备战了第七届“飞思卡尔”杯，在这期间我们经历了很多挫折，也积累了一些经验，希望通过技术报告能够总结出我们的一些经验和教训。我们在 CodeWarrior IDE 开发环境中对智能车进行软件开发，我们用 CMOS 摄像头进行赛道图像的采集，对图像进行软件二值化。提取出赛道两边黑色引导线位置，求出中线位置，之后根据中线的趋势变化，来控制舵机和电机。采用光电编码器检测模型车当前的车轮转速给以反馈，使用 PID 控制算法调节驱动电机的转速和转向舵机的角度，实现了对模型车运动速度和运动方向的闭环控制。智能车系统主要由核心板，电源模块，电机驱动，图像采集模块，速度反馈模块。为了提高模型车的速度和稳定性，使用无线模块、键盘调试模块等调试工具等。我们用飞思卡尔公司的 16 位单片机 MC9S12XS128 作为控制核心芯片，结合图像的识别与处理，通过控制算法驱动转向机构与行驶机构，达到稳定且快速行驶的目的。本文详细叙述了智能车系统各个模块子系统的原理，设计目标，设计方法与过程，以及其所发挥的作用。主要分为机械结构设计，硬件电路设计和软件系统设计三大部分。为了提高智能汽车的行驶速度、可靠性稳定性，我们对比了各种方案的优缺点。实验结果表明，我们的智能车系统设计方案稳定可行，机械结构与控制算法经过长时间的调试均达到优化的状态，系统的鲁棒性较强。经测试，目前智能车可以稳定的完成一段路况复杂的赛道，平均速度达到 3m/s。运行稳定，达到了设计目标。

关键词：智能车、MC9S12XS128、传感器、PID 控制、无线

第一章 引言

1.1 引　　言

智能汽车是未来汽车的发展方向，在减少交通事故，能够稳定快速高效等许多方面发挥很重要的作用；智能汽车是一个集通信技术、计算机技术、自动控制、信息融合技术、车辆工程技术、传感器技术等于一身的行业，它的发展势必促进其他行业的迅速发展，在一定程度上代表了一个国家在自动化智能方面的国际水平，随着科学技术的不断发展进步，智能控制的应用越来越广泛，几乎渗透到所有领域。智能车技术依托于智能控制，前景非常广阔而且发展迅速。汽车在走过的 100 多年的历史中，从未停止过智能化的步伐，进入 20 世纪 90 年代以来，随着汽车市场竞争激烈程度的日益加剧和智能运输系统地兴起，国际上对于智能汽车及其相关技术的研究成为热门，一大批有实力有远见的大公司、大学和研究机构开展了这方面的研究这一新兴学科正在吸引越来越多的研究机构和学者投入其中。

在我国大学生“飞思卡尔”杯智能汽车竞赛从 2006 年开始已经成功举办了六届，得到了各级领导及各高校师生的高度评价。也激起广泛学子的兴趣，为大学生提供了一个充分展示想象力和创造力的舞台，吸引着越来越多来自不同专业的大学生参与其中。全国大学生“飞思卡尔”杯智能汽车竞赛包括光电组、摄像头组和电磁组，其中摄像头组的智能车速度最快，可以对赛道进行更精确的判断。在准备比赛的过程中，我们小组成员接触了智能控制、模拟识别、机械设计、传感器技术、汽车电子等多个学科，几个月来的经历，培养了我们电路设计、软件编程、系统调试、机械设计等多方面的能力，让我们在课堂上学

的内容能够用到实际当中，培养了我们实践动手的能力，对今后的学习工作都有很大的实际意义。摄像头传感器具有其他传感器（在智能汽车方面的传感器）所不具有的优势，摄像头传感器视野宽广，采集信息量大，准确，但目前摄像头也有很多技术上的不足，比如受光的影响较大，但随着技术的不断发展，基于摄像头的智能小车系统的研究将推动智能汽车的快速发展。

1.2 报告概述

在这份技术报告中，包含了我们从认识智能车大赛到设计制作出整个智能车的全部经历，这份技术报告记录了我们对本项赛事的认识、编译软件、传感器的选择与使用、机械研究及改造、PCB设计、程序算法、PID控制、调试过程以及PID参数调试，阐述了我们的思想创意和经验教训，在电路的设计、机械的改造以及算法的不断优化。这份技术报告凝聚了我们团队的劳动成果和智慧。在准备此次比赛的整个过程中，我们不断补充各方面的知识，包括PID控制类、传感器技术、汽车电子、模型车机械结构等多个学科，这次比赛的整个准备过程对我们的各方面知识的应用、实践动手能力以及创新思维方面的培养都有极大的推动作用。

1.3 比赛背景

教育部为了加强大学生实践、创新能力和团队精神的培养，在全国举办全国了大学生数学建模、电子设计、机械设计、结构设计等4项竞赛，在此基础上，经研究决定，委托教育部高等学校自动化专业教学指导分委会主办每年一度的全国大学生智能汽车竞赛。“飞思卡尔”杯全国大学生智能汽车竞赛已经成功举办了六届，本届智能汽车竞赛所使用的是A型车模，由组委会统一提供。比赛跑道为表面白色，两边有连续黑线作为引导线，黑线宽为25mm±5。比赛规则限

定了跑道宽度45cm和跑道最小曲率半径不小于50cm，跑道可以交叉，交叉角为90°，交叉路口黑色边缘线，具体形状在比赛当天现场公布。各参赛队伍在严格遵守比赛规则的条件下，选择光电、摄像头、电磁三种检测道路信息方法之一来选择传感器，并设计电机驱动、转向舵机控制以及控制算法，以简洁但功能完善为出发点，以稳定为首要条件，在最短的时间内跑完全程，并能检测赛道的起跑线自动停止在起跑线3m范围之内。

1.4 文献综述

智能车大赛所涉及到的知识面非常广泛，通过指导老师的引导，看往届的技术报告，智能车官方网站和上几届师兄的介绍、以及在各论坛与网友进行交流学习，我们对所阅读的文献进行了一下总结，将文献中所受到的启发进行了以下重点总结。X12单片机开发资料从Freescale公司官方网站下载得到X12单片机开发技术手册。仔细阅读各个功能模块文件，了解所有寄存器功能。自己在图书馆借的的《Freescale 9S12十六位单片机原理及嵌入式开发技术》，熟悉X12单片机的内部结构，具备了开发所需要的基本知识——智能车制作知识阅读了《学做智能车》一书，又通过网络查询一些相关资料，对整个制作过程有了初步的了解，对制作智能车有了一定的理论认识。

关于PID控制算法的文章很多，对于此次比赛，重点在于车的稳定和速度的提高，所以我们控制车在不掉出赛道的前提下走最短路线作为最佳路线，在阅读一些文献时，重点参考模糊控制和算法容易实现的内容。还从网上看了很多往届的智能车比赛视频，研究他们走的路线，综合这些强队优点我们有的放矢，在转角控制中力求最佳路线，最终在小S弯可以直冲过去。

1.5 本文结构

本篇技术报告采用总分的写法，先对系统总体设计思路进行介绍，然后分别对各部分模块进行介绍，突出强调了车模的机械改造、硬件电路设计和软件编程。本文除引言外有6个章节，第一章为引言，第二到六章为主体部分，对机械、硬件和软件设计进行了详细介绍，并对调试方法和调试过程进行了说明；第七章为车模的主要技术参数和总结，最后附上软件代码。

第二章　　系统设计

2.1 系统方案介绍

我们的智能车系统采用飞思卡尔16位单片机MC9S12XS128为核心控制芯片，采用COMS数字摄像头采集信号，摄像头拍摄赛道图像信息输出到信号处理模块进行视频同步信号分离。分离出场同步信号，行同步信号，奇偶场信号，数据和同步信号同时输入到S12X控制核心，通过软件二值化进一步处理以获得图像信息。通过光电编码器来检测车速，并采用S12X的脉冲累加器进行脉冲计数计算车模的当前速度；以处理后的信号来控制PWM模块，通过输出不同占空比的PWM分别对转向舵机、直流电机进行控制，完成智能车的转向，前进和制动。为了使智能车能够快速行驶，必须把路径的判断、相应的转向伺服电机控制以及直流驱动电机的控制精密地结合在一起。不论是COMS部分数据的错误采集和识别，还是转向伺服电机控制的失当，都会造成模型车严重抖动甚至偏离赛道；如果直流电机的驱动控制效果不好，还会造成直线路段速度上不去，或者知道摆动，入弯道速度过快而使智能车冲出赛道，入小S不能直冲等问题。

2.2 系统总体结构

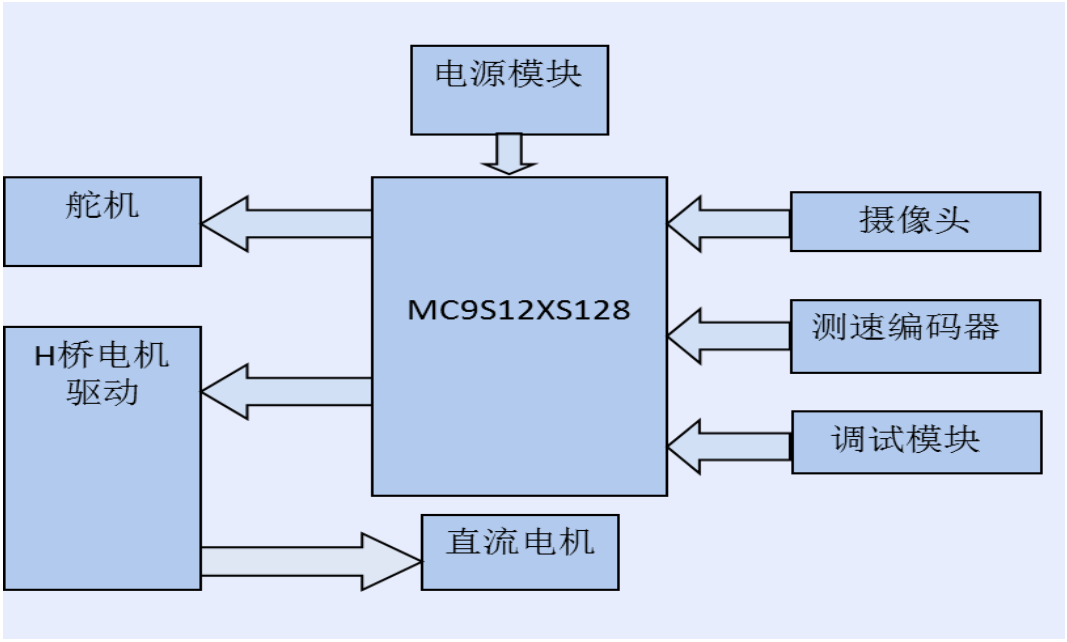


图2.2.1

系统结构图如图2.2.1

按照预先的设计，我们设计了整个系统的结构图。系统力求简单高效，在满足比赛要求的情况下，使硬件结构最简单，减少因硬件而出现的问题。
本硬件系统主要由几个部分组成：

（1）中央处理器单元

此次我们使用的核心单片机是MC9S12XS128,是飞思卡尔公司的MC9S12系列之一。内部Flash128KB，拥有2组各8路10位片内A/D、16路I/O口，有功能强大的8位PWM输出端口，以及8路16位增强型定时器（ECT）。通过对时钟合成寄存器和时钟分频寄存器的设置，我们将单片机的时钟频率提高到72MHz，经长期验证，在此频率下X12单片机能稳定工作！此芯片专门针对智能车开发的，完全能够胜任小车的检测和控制功能。所以我们选择了以S12X为核心的最小系统板！

（2）传感器模块

传感器我们选择了OV7620数字摄像头，摄像头先将路况信息传至单片机，然后用单片机来控制车模能够稳定的前行。使用通道PORTA来采集数据，同时，巧用16毫秒作为时间段，进行舵机及电机的控制。

（3）电源模块

为单片机和各个电路模块提供稳定电源，保证各模块正常工作，我们使用主委会提供的A车模电池。每次试车都量下电压，总结出理想电压。

（4）舵机驱动模块

对模型车上的S3010舵机直由电源电压供电，达到快速准确控制赛车方向，我们使用PWM23组合通道来控制舵机。

（5）电机驱动模块

对模型车上的电机进行驱动，控制赛车的速度，以CD4011和MOS管组成的H桥驱动电机，我们使用PWM45组合通道控制。

（6）速度检测模块

对模型车的速度进行实时检测，实现闭环控制，以便调整弯道和直道的速度，从而提高平均速度，使小车能更稳更快的跑完全程。

2.3 本章小结

本章通过论证，比较和讨论之后，我们确定了系统的整体方案。在接下来的章节里我们会详细介绍。

第三章　车模机械改装的设计与调节

一个完整的智能车系统，其最底层的就是智能车模型的机械结构。机械结构的设计关系到车的稳定性，尤其是速度在 3m/s 以上时，机械的好坏就决定着最高速度。所有的硬件电路，传感器，执行机构等都是架在车体机械结构上的，可以说机械结构对车模运行性能的影响是极其重要的，是影响速度的关键因素之一。一个良好的机械结构可以使智能车的控制算法部分变的更简单。鉴于这个原因，我们在机械设计方面花了好多功夫，进行了大量的调整。

3.1 前轮倾角的调整

3.1.1 主销内倾

主销内倾是指主销装在前轴略向内倾斜的角度，它的作用是使前轮自动回正。前轮内倾的优点是增加了前轮的稳定性，但同时也增加了舵机转向的负担。角度越大前轮自动回正的作用就越强烈，但转向也越费力，轮胎磨损增大；反之，角度越小则前轮自动回正的作用就越弱。为了保证前轮转向的灵活度，主销内倾不宜过大，故将其调节为3度左右。

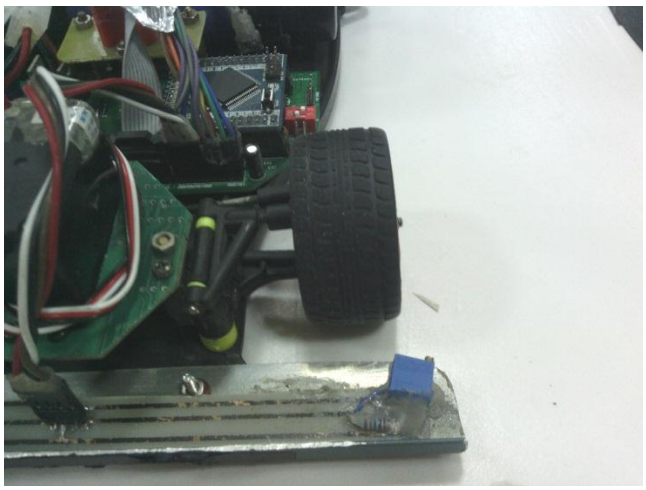


图3.1 主销内倾实物图

3.1.2 主销后倾

主销后倾是指主销装在前轴、上端略向后倾斜的角度。主销后倾主要影响的也是前轮稳定性和舵机转向负担。因此要在车子的稳定性和转向灵活性之间做一个权衡。主销后倾角越大，车速越高，前轮稳定性也愈好，但转向愈沉重。故选择了3度左右，是一个居中的值。

3.2 后轮差速器调节

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。现因所用车模配备的是后轮差速器，而差速器的特性是：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；此次使用的后轮差速器在过弯时，外侧前轮轮胎所遇的阻力较小，轮速较高；而内侧前轮轮胎所遇的阻力较大，轮速较低。因此，差速器的调整要注意滚珠轮盘间的间隙，过松过紧都会使差速器性能降低，转弯时阻力小的车轮会打滑，从而影响车模的过弯性能。好的差速机构，在电机不转的情况下，右轮向前转过的角度与左轮向后转过的角度之间误差很小，不会有迟滞或者过转动情况发生。为了保证过弯的平滑性，根据反复在圆形赛道上的实验，特地将差速器调至较理想的状态。

3.3 传动齿轮的调整

齿轮传动机构对车模的驱动能力有很大的影响。齿轮传动部分松紧度调节得不恰当，会大大增加电机驱动后轮的负载，从而影响到行驶速度。调整的原则是：两传动齿轮轴保持平行，齿轮间的配合间隙要合适，过松容易打坏齿轮，过紧又会增加传动阻力，白白浪费动力；传动部分要轻松、顺畅，容易转动，不能有卡住或迟滞现象。

判断齿轮传动是否调整好的一个依据是，听一下电机带动后轮空转时的声音。声音刺耳响亮，说明齿轮间的配合间隙过大，传动中有撞齿现象；声音闷而且有迟滞，则说明齿轮间的配合间隙过小，或者两齿轮轴不平行，电机负载

加大。现经多次反复实验，将传动齿轮调整至噪音较小，没有碰撞类的杂音状态。

3.4 舵机的安装方式

我们使用舵机倒置安装的方法，结构如图3.2所示，这样安的好处是可以降低重心。我们发现这样安效果很好。

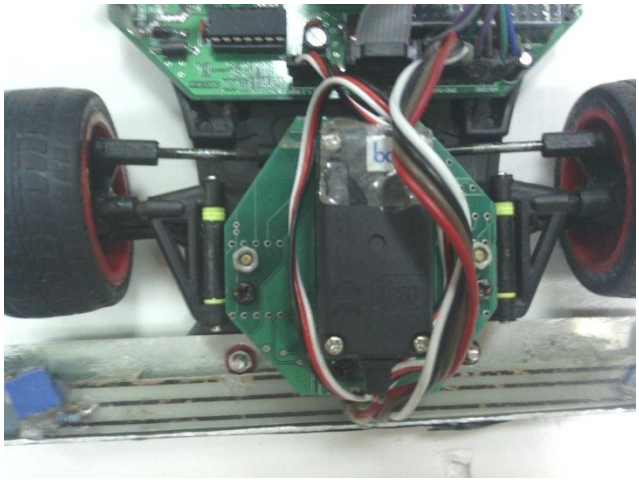


图3.2 舵机倒置安装

倒置安装可以增加连杆的长度，可以增加舵机的灵敏度，但这会相应地减小单位控制增量增加的力矩，此舵机单位输出力矩很大，完全可以满足比赛时对转弯的要求。所以我们选择了倒置安装的方法。

3.5 摄像头的选择与安装

3.5.1 摄像头的选择

在选择摄像头时，我们参照前几届比赛的经验和摄像头选择，对目前可选的各种摄像头进行了比较和实验。目前市面上常见的摄像头主要有 CMOS 和 CCD 两种， CCD 摄像头具有对比度高、动态特性好的优点，但需要工作在 12V 电压下，对于整个系统来说过于耗电，且图像稳定性不高；CMOS 摄像头体积小，耗

电量小，图像稳定性较高。因此，经过实验论证之后我们决定采用 CMOS 摄像头。对于 CMOS 摄像头又分数字和模拟两种。我们选用了 OV7620 进行实验，对数字摄像头的可行性进行论证。实验之后，得出结论：数字摄像头 OV7620 可以直接输出 8 路数字图像信号，使主板硬件电路的简化成为可能，且能够达到 60 帧/S 的帧速率，因此，最终我们采用了 CMOS 数字图像传感器方案。

3.5.2 摄像头支柱材料

好的支柱质量要轻，以免过度增加车身重量；要有一定钢性，以免摄像头固定不牢靠，发生松动；另外韧性也必不可少，太脆的材料经不起撞击，容易折断。综合考虑以上因素，我们选用了碳纤维材料的轻质杆，这种材料质地轻、刚性大，直径小而非常坚固耐用。具体外形见下图3.3：

图3.3 摄像头固定支架

3.5.3 摄像头安装位置

综合考虑车模质心、运动状态下摄像头与底盘的随动性能、整体安装的协调感，我们将摄像头装在了车体中间的位置，为了防止杆前后晃动，在它的后面又加了两个细一点的轻质杆加以固定。为了防止静电干扰摄像头我们在摄像头支架上包了一层锡箔纸。

3.5.4 调整摄像头的高度与角度

摄像头装到车上后，其自身的高度与镜头俯角将直接影响到车子的远瞻距离与黑线最少扫描点，因此要根据需要仔细调校。前瞻大概1.1m左右，高度26cm左右，在小车跑起来也很稳定。下图3.4为调校好的摄像头。

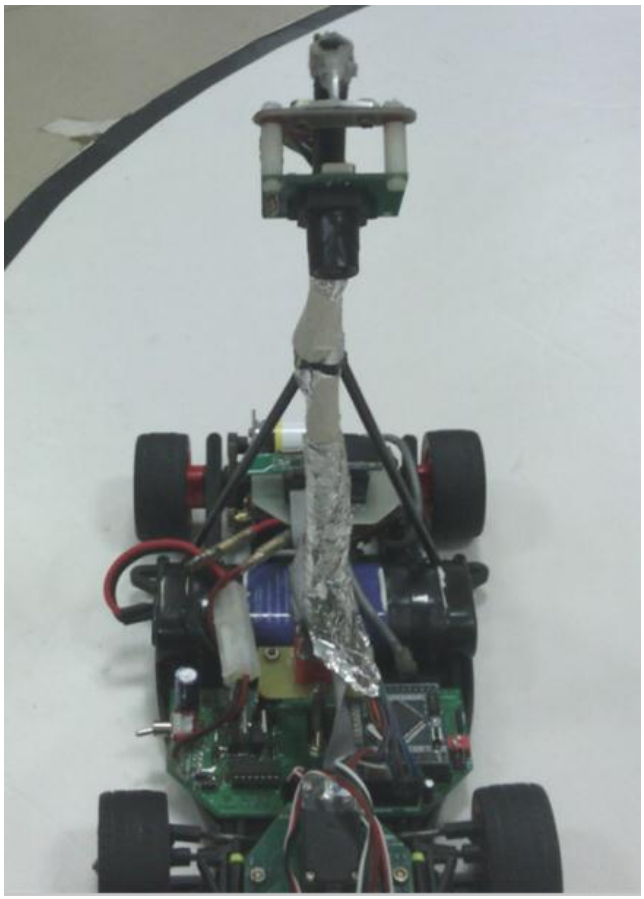


图3.4 调校好的摄像头

3.6 编码器的选择与安装

光电耦合管的发射端发射的红外光，其感光度达0.8mm，通过激光雕刻码盘的间隙到达接收端。当电机带动码盘转动时，接收端便以一定的频率接收到红外光线，经过整形电路输出频率与电机转速成正比的方波。由单片机进行数数测频，即可得到电机转速。由于激光雕刻码盘精度很高，所以用光电编码器测速非常准确。为了使整车安装紧凑，我们采用PCB板将光电码盘固定于赛车的最后

端。 编码器的安装如图3.5所示，编码器齿轮、电机齿轮和传动齿轮在一个水平线上，编码器的齿轮和传动齿轮的安装紧度如3.3传动齿轮的调整。

图3.5 编码器的安装

3.7 本章小结

本章主要介绍了我们对赛车机械结构进行的安装和改进。主销后倾角设为3度左右，使车轮具有自动回正的功能。为了缓解前轮的松动影响，设定了一定的主稍内倾。为了降低车的重心，我们将舵机倒置安装，使前轮的左右连接杆相等，保证左右转向相同，赛车过弯的性能也有了很大的提高。通过在前轮底盘加垫片的手段，降低了重心，有利于车的稳定性。通过对前轮、底盘、和舵机的一系列调整，提高了赛车的行驶性能。

第四章　硬件电路设计

我们将电路主要分成四部分，最小系统板、母版、红外对管板、调试板。最小系统是使用的助赢提供的 XS128 的 80 引脚的 PCB 版；母版上面包含了信号采集模块、驱动电路模块、测速模块，舵机驱动模块、各芯片的电源稳压部分，引出各种需要的引脚连接其它外围电路；电机驱动是采用 H 桥电路，加工成的 PCB 板。

4.1 最小系统板

为了为单片机提供稳定的电路，我们没有自己设计最小系统，网上的几家店面在这方面的制作已经很成熟了，设计的最小系统版轻巧，稳定，在组委会允许下，我们使用了现成的PCB版。我们采用freescale公司MC9S12XS128,16位、80引脚的单片机如图4-1,其集成了PWM,SPI,SCI,ECT,CAN,A/D,EEPROM,FLASH等模块，使用方便，功能强大。核心板主要用来对摄像头采集回来的数据进行读写，然后根据采集到的信息求得左右黑线的实际位置，算出中线位置，并根据一定的算法对舵机和电机进行控制。同时它还可以控制参数的设定等工作，因此最小系统板是整个系统的核心。

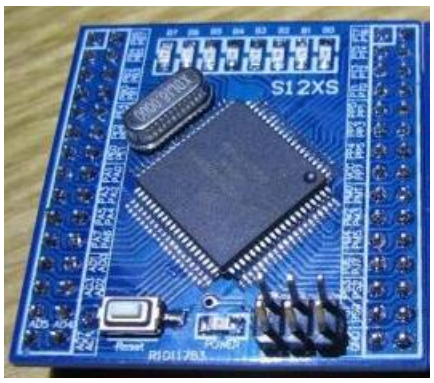


图4.1 最小系统板实物图

4.2 电源模块设计

电源是一个系统正常工作的基础，电源模块为系统其他各个模块提供所需

要的能源保证，因此电源模块的设计至关重要。模型车系统中接受供电的部分包括：传感器模块、单片机模块、电机驱动模块、伺服电机模块、无线模块、调试模块等。设计中，除了需要考虑电压范围和电流容量等基本参数外，还要在电源转换效率、噪声、干扰和电路简单等方面进行优化。可靠的电源方案是整个硬件电路稳定可靠运行的基础。智能车全部硬件电路的电源由7.2V，2A/h的可充电镍镉电池提供。由于电路中的不同电路模块所需要的工作电流容量各不相同，因此电源模块应该包含多个稳压电路，将充电电池电压转换成各个模块所需要的电压。

整个系统中+5V电路功耗较小，为了降低电源纹波，考虑使用线性稳压电路。另外，后轮驱动电机工作时，电池电压压降较大，为提高系统工作稳定性，必须使用低压降电源稳压芯片。常用的低压降串联稳压芯片主要有LM2940、LM1117等。我们使用LM2940提供+5V稳压电压，给最小系统、编码器和摄像头供电；使用LM1117提供5V稳压给调试模块和红外供电；LM1117提供+3.3V稳压电压为防撞的无线模块和调试模块中的OLED显示屏供电。舵机由电源电压供电。 电源模块如图4.2所示。

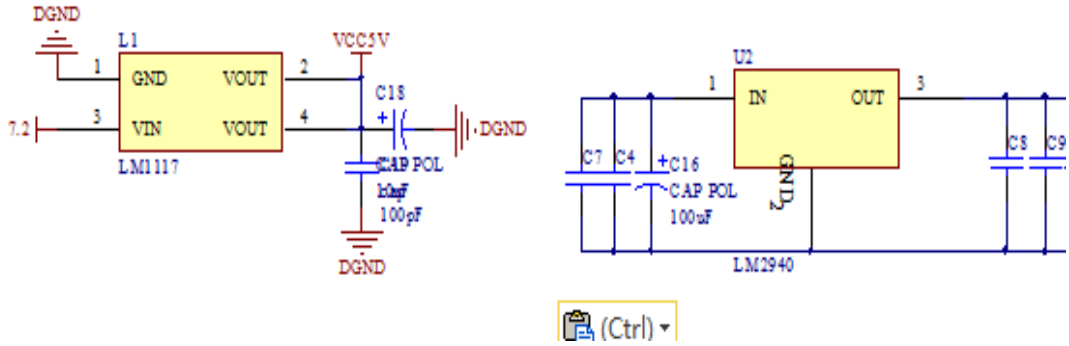


图4.2

4.3 驱动电路设计

今年的摄像头组用的是A车模型，它与以往车模都不相同，功率较大，所以我们的电机驱动板为一个由分立元件制作的直流电动机可逆双极型桥式驱动器，其功率元件由两支N 沟道功率MOSFET 管和两支P沟道功率MOSFET管组成，

大大提高了电动机的工作转矩和转速。该驱动器主要由以下部分组成： PWM信号输入接口、逻辑换向电路、电源电路、桥式功率驱动电路、缓冲保护电路等。实际使用中要求电机可以正反转，故使用两片接成全桥驱动。驱动电流基本能满足，损耗更小，实际证明效果不错。电路如图4.2 示

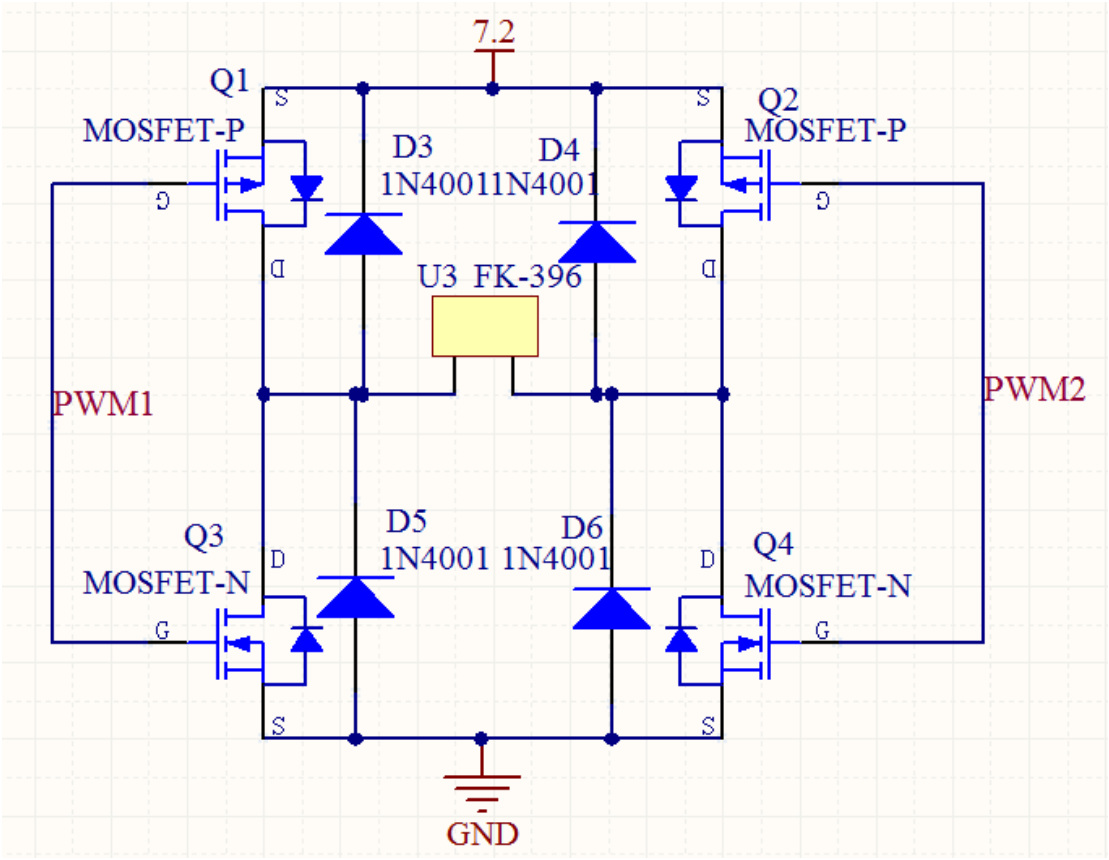


图4.2 驱动电路原理图

4.4 测速模块

为了使得模型车能够平稳地沿着赛道运行，除了控制前轮转向外，还需要控制小车的车速，形成闭环控制，使模型车能在急转弯时减速，使速度不至于过快而冲出赛道，同时可以使小车在直线段时以较快的速度行驶。所以要实时检测当前小车的速度，并根据小车所处的位置来实时调整小车的速度,使得小车在赛道上运行得更精确、更稳定，测速用PT7口。

电路如图4.3示

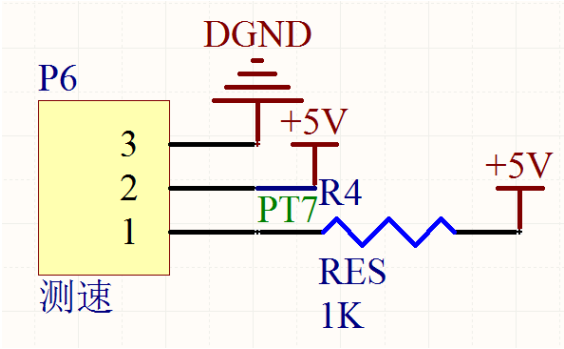


图4.3

4.4.1 测速原理及其分析

速度采用脉冲计数器，光电编码器的输出接在定时器PT7接口上。将该脉冲信号经电压比较器输入到MC9S12XS128芯片的计数器端口上，可得出相应于脉冲数值。然后通过计算得出当前转速的数值。在软件里，场中断处理程序读取一次脉冲次数，读完后立即清零，等待下一次的中断。并将读取数据经过处理，除去偶然误差，作为瞬时速度值。将该数字量于期望的转速相比较，来调整小车的速度。

4.4.2 采样周期的和光电编码盘线数的选择

我们使用欧姆龙E6A2-CW3C光电编码器，我们的采样周期设定为16ms，与摄像头的采样周期保持一致。

图4.4 编码器实物图

4.5 起始线和停车线的识别

由于摄像头组的起始线是中心线两端长10cm宽25mm的黑线，具体如图4.5所示。而对于摄像头组来说可以直接来判断也可以通过红外对管来辅助检测。我们最后使用两个红外对管来检测起始线，具体方法在下一章软件系统设计中会讲到。红外对管硬件如图4.6。

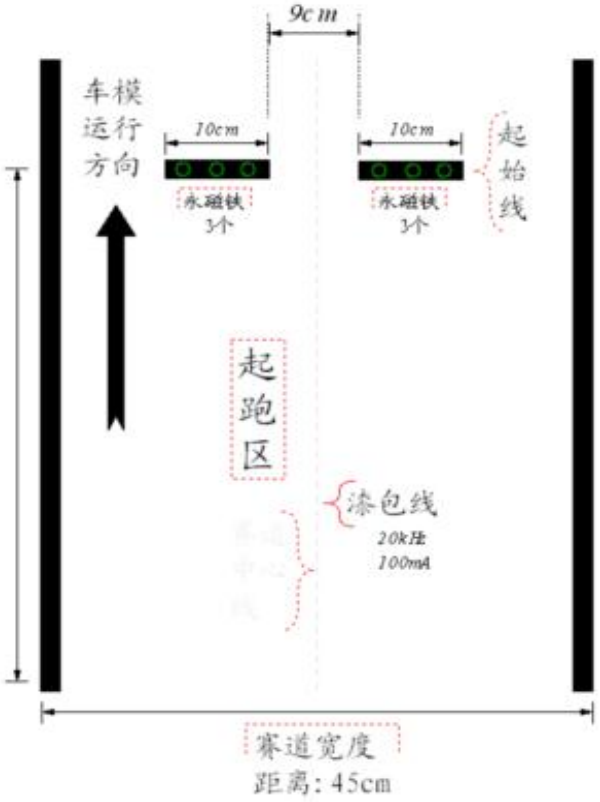


图4.5、起跑线示意图

红外对管如图4.6

图4.6

4.6 本章小结

本章介绍了系统的硬件模块，包括电源模块，核心板，视频采集模块，速度检测模块，检测停车模块和驱动模块。使用 LM1117 和 LM2940 线性稳压芯片，通过合理的布局设计，为赛车系统构建了一个稳定的电源系统，摄像头采用数 CMOS 视频芯片，稳定可靠。驱动电路采用 H 桥式电路的方案，既提高了驱动的能力，又减少了芯片的发热量。经测试整个硬件系统能可靠稳定的工作，为软件系统搭建了一个稳定可靠的硬件平台。

第五章　软件系统设计

5.1 MC9S12XS 控制软件主要理论

智能车开发环境采用了飞思卡尔MC9S12XS系列单片机开发软CodeWarrior 5.0版本。今年我们使用了5.0版本来完成对单片机个模块的初始化，使用起来非常方便，避免了之前需要很多代码来初始化的繁琐过程，X12单片机该软件具有支持多种语言、开发环境界面统一、交叉平台开发以及支持插件工具等特点。在CodeWarrior界面完成编译后，通过BDM工具，在CodeWarrior环境下向MC9S12XS模块下载程序。BDM工具使用简单，十分方便。开发环境如图5.1

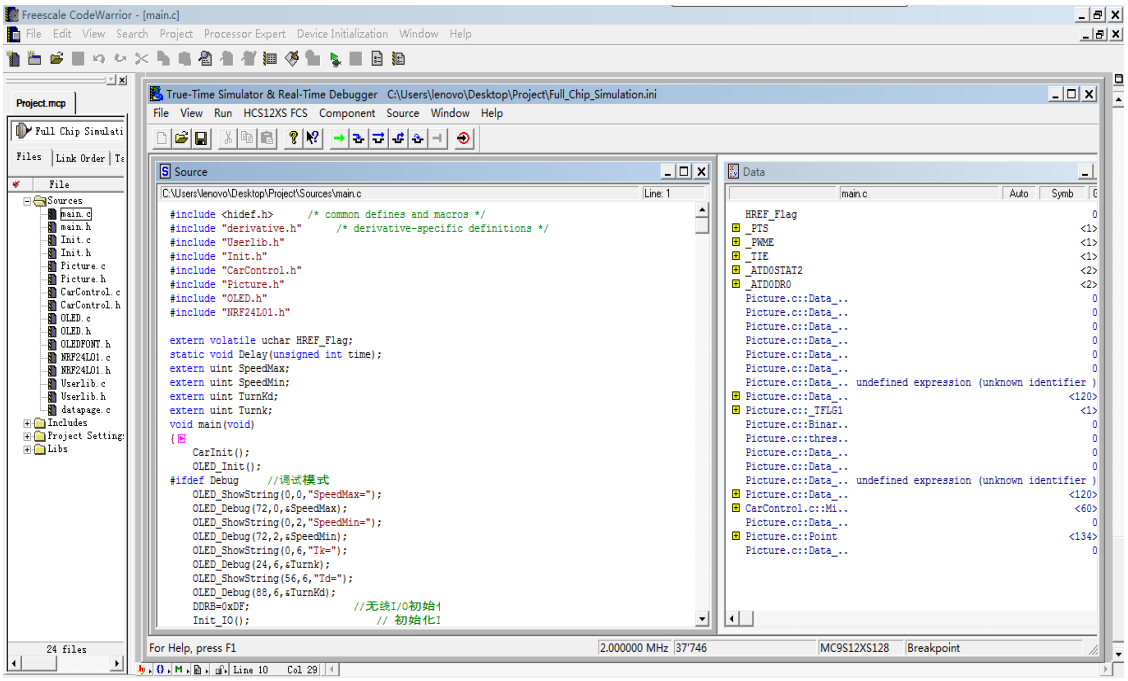


图5.1

5.2 软件系统总体设计

在整个系统设计中，用到了该单片机的4个基本功能模块： PWM输出模块、ECT模块、普通IO端口、外部中断口，先进入初始化模块，对各个模块进行初始

化，设置好后便可进入调试模块，进行模块参数的调整。然后进入主程序，为实现所期望的功能，系统通过在主程序内循环调用信号检测、信号处理、路径计算、路径判断、起始线检测、舵机控制、直流电机控制等功能子模块，程序的执行为先对各个模块进行初始化，进入调试模块，然后在主程序中完成相应的功能。高效稳定的软件程序是智能车平稳快速寻线的基础。在整个智能车中软件程序是最主要的，速度的提升很大一部分取决于程序的书写。本智能车采用CMOS摄像头作为寻线传感器，图像采集处理就成了整个软件的核心内容。我们在图像处理中主要包括以下几个方面的内容：黑线提取、二值化算法、求左右黑线，求中线、算左右偏差、求斜率、判断赛道类型、边缘干扰的处理。算法流程图如图5.2所示

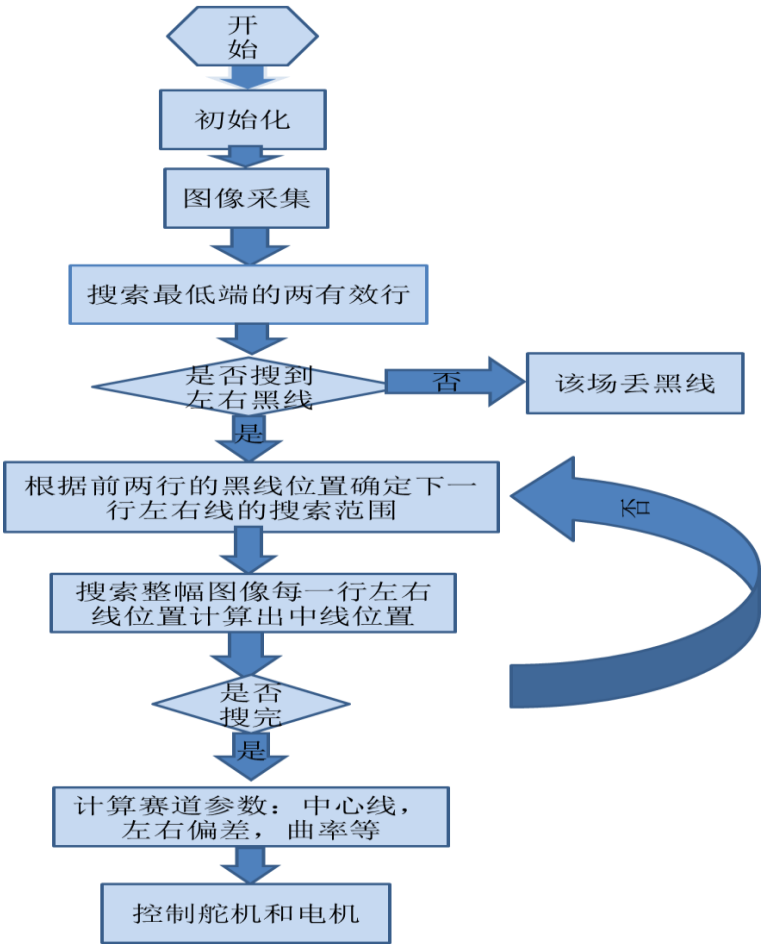


图5.2

5.3 各功能模块设置

5.3.1 时钟模块

时钟基本脉冲是CPU工作的基础。MC9S12XS128微控制器的系统时钟信号，由时钟振荡电路或专用时序脉冲信号提供。MCU内部的所有时钟信号都来源于EXTAL引脚，也为MUC与其他外接芯片之间的通信提供了可靠的同步时钟信号。X12的总线时钟是整个MCU系统的定时基准和工作同步脉冲，其频率固定为晶体频率的1/2。对于X12，可以利用寄存器SYNR、REFDV来改变晶振频率FOSCCLK，可以选用8MHz或16MHz外部晶体振荡器作外时钟。对SYNR 和REFDV进行设置，可以得到72MHz的总线频率。

5.3.2 PWM 输出模块

MC9S12XS128集成了8个8位独立的PWM通道，通过相应设置可变成4个16位PWM通道，每个通道都有专用的计数器，PWM输出极性和对齐方式可选择，8个通道分成两组，共有4个时钟源控制。PWM0、PWM1、PWM4、PWM5为一组，使用时钟源ClockA和ClockSA；PWM2、PWM3、PWM6、PWM7构成另一组，使用时钟源ClockB和ClockSB。ClockA和ClockB均是由总线时钟经过预分频后得到，分频范围1~128，通过寄存器PWMPRCLK来设置，ClockSA和ClockSB是分别通过ClockA和ClockB进一步分频后得到的，分频范围为1~512，分别通过寄存器PWMSCLA和PWMSCLB来设置，计算公式为：ClockSA=ClockA/（2*PWMSCLA）ClockSB=ClockB/（2*PWMSCLB）通过寄存器PWME来控制PWM0~PWM7的启动或关闭。为了提高精度，我们将PWM0和PWM1，PWM2和PWM3，PWM6和PWM7级联，构成16位的PWM通道，级联时，2个通道的常数寄存器和计数器均连接成16位的寄存器， 3个16位通道的输出分别使用通道7、3、1的输出引脚，时钟源分别由通道7、3、1的时钟选择控制位决定。级联时，通道7、3、1的引脚变成PWM输出引脚，通道6、2、0的时钟选择没有意义。但是通过PE模式设置就相当方便了，不用再去写代码控制寄存器，直接在窗口里面设置就可以了。

5.3.3 ECT 模块

X12得ECT具有8个输入（IC）捕捉/输出(OC)比较通道，可以通过设置TIOS寄存器选择输入或输出比较功能。ECT既可以作为一个时基定时产生中断，也可以用来产生控制信号。在PE模式里一样很方便，将所用到的P7口添加进来并做相应的设置就可以了。通过ECT模块，我们实现了对脉冲进行计数，检测智能车的速度，对速度进行闭环控制。

5.3.4 外部中断

对于摄像头图像的采集我们需要用到两个中断，所以我们需要添加两个中断口到PE模式中来，对其捕捉脉冲的方式选择、使能、端口的选择进行设置，设置好后编译后便可直接使用。

5.3.5 A/D

我们用一个AD模拟键盘进行参数的调整，这样节省了三个引脚，可以简化电路，两个AD口用来起始线红外的检测。

5.4 路径识别与算法

路径的提取和识别为控制算法的核心内容，模型车先通过摄像头检测路径，就是把摄像头看到的路况提取出来。但是这里包括噪点等等其他因素的一些干扰，所以要先对黑线的提取，这是一个图像处理的重要过程。关键是把背景图像跟黑线分开，我们采用的方法是利用简单的阈值判定作为黑白赛道的识别。既对当前某一路采集电压进行判断，我们采集回来的数据阀值是0—255，数值越高表示是白色，数值越低就是黑线所在的部分，当然也不排除一些噪点的干扰。所以接下来就是要对噪点的排除。阈值分割法的优点：它是一种基于区域的分割技术，它对图像有较强对比的景物的分割特别有用，计算简单，而且总能用封闭且连通的边界定义不交叠的区域。阈值分割法的关键在于阈值的确定。

如果阈值是不随时间和空间而变的，称为静态阈值；如果阈值随时间或空间而变化，称为而动态阈值。基于静态阈值的分割方法算法简单，计算量小，但是适应性差而采用了动态阈值的方法，针对本智能车系统，动态阈值就能适合绝大部分情况，因为智能车的运行赛道黑白分明，而且就只有黑白两种颜色，比赛背景也较简单。但是当有反光时就很不利对图像的的分析，特别是远景处，一般有反光的话这个阀值就不会有很大的差异，我们的跑道虽然布置在室内，远端的反光任然较厉害。如果环境好调试起来相对要容易的多。所以每次调车的时候都要拉上窗帘，用稳定的电灯的光源。

5.4.1 图像的二值化

图像的二值化的基本原理：图像的二值化处理就是以采到的点的灰度值为根据灰度值的大小分为黑线和白色底板，即将256个亮度等级的灰度图像通过适当的阀值选取而获得仍然可以反映图像整体和局部特征的二值化图像。在数字图像处理中，二值图像占有非常重要的地位，特别是在实用的图像处理中，以二值图像处理实现而构成的系统是很多的，要进行二值图像的处理与分析，首先要把灰度图像二值化，得到二值化图像，这样子有利于再对图像做进一步处理时，图像的集合性质只与像素值为0或255的点的位置有关，不再涉及像素的多级值，使处理变得简单，而且数据的处理和压缩量小。为了得到理想的二值图像，一般采用封闭、连通的边界定义不交叠的区域。所有灰度大于或等于阀值的像素被判定为属于特定物体，其灰度值为255表示，否则这些像素点被排除在物体区域以外，灰度值为0，表示背景或者例外的物体区域。如果某特定物体在内部有均匀一致的灰度值，并且其处在一个具有其他等级灰度值的均匀背景下，使用阀值法就可以得到比较的分割效果。如果物体同背景的差别表现不在灰度值上（比如纹理不同），可以将这个差别特征转换为灰度的差别，然后利用阀值选取技术来分割该图像。动态调节阀值实现图像的二值化可动态观察其分割图像的具体结果。黑线提取算法的基本思想：

1） 直接对原始图像逐行扫描，根据每行的实际情况设定每行的阈值提取黑白跳变点；

2)一旦摄像头的高度和倾角固定后，摄像头所看到的黑线宽度也就落在了一个范围内，在确定的黑线宽度范围内提取有效黑点，这样可以排除不在宽度范围

内的黑点干扰；
3)利用黑线的连续性，根据上一行左右黑线的位置来确定本行搜左右黑线的参考中心；
4) 图像是远处小近处大，但是远处看到的弧立黑点数也比较多，所以黑线宽度范围和前后行黑线中心的位置差别都要动态调整；
5) 图像数据量大，全部扫描一遍会浪费很多时间，利用前面已经求出的黑线中心位置判断出黑线的趋势，从而推断出下一行的黑线大概位置，确定出扫描范围，避免整行逐点扫描。
6) 提取出整场所有有效行的左右黑线后，根据预先设定好的权重行计算出中心线的加权平均，作为本场的黑线重心，由此也可以看出黑线的走线的大致趋势。

5.4.2 赛车走线路径的选择

根据观察前几届比赛以及本届西部赛区比赛获得的经验，赛车能否以最少的时间完成比赛，与赛车的速度和路径都有着密切关系，因此，如何能够使赛车以一个最合理、最高效的路径完成比赛是提高平均速度的关键。对于赛车路径的优化，我们从两方面来完成：
1、增加视角的长度和宽度
根据我们的分析，当赛车采集到的图像能够覆盖一个比较完整的大S弯道时，通过加权算法计算出来的黑线中心就会处于视场中央附近，此时赛车会以一个比较好的路径快速通过S弯道；相反，如果视场无法覆盖一个完整的S弯道，赛车就会误处理为普通的单向弯道，这样赛车的速度就会大大减慢。因此，尽量增大视场的长度和宽度就很有必要了。视场的长度与单片机可以处理的图像行数成正比。为了增加视场宽度，增加每行采集的图像点数之外，我们采用了广角镜头，从而有效地增加了视场宽度。
2、加权算法的优化
对整场有效行的黑线中心求加权平均的算法，在低速情况下可以有效地优化赛车路径，但在赛车速度提高到一定程度之后由于过弯时的侧滑，路径不是很好，究其原因是由于图像分布不均，三分之二的行分布于车体前方50cm的范围内，求出的加权平均受车体近处的图像影响较大，因此整场图像求加权的算法对于高速情况下的路径优化效果不是很明显。为了解决这个问题，我们对采样行进行了重新分配，做到近远端黑线分布赛道上大致均匀分布，这样远端黑

线的位置对转角的影响更大，适合高速情况下提前转弯，合理利用摄像头的大前瞻。

3、识别赛道

在识别赛道的时候我们尝试了很多方法，曲率的求法尝试了几种、最后选择了一种比较准确的算法，可以判断出直道、入弯、在弯道、出弯。为准确的入弯减速，出弯加速打下坚实的基础。判断小S时可以把它判断为直道，这样可以在小S时直冲过去。

5.4.3 起跑线检测

开始尝试过用摄像头检测起始线，由于我们检测30行，小车速度低时起始线图像可以检测一到两行，停车准确率挺高，但速度达到2.8m/s以上时，发现车停不了了，经过试验检测发现这时候一行起始线也检测不到了，速度过快扫描不到起始线了！再加上为了减少控制器的运算量，我们决定用红外对管来辅助检测起始线和停车线，以达到停车目的。我们用两个红外对管，如图4.6所示，开始在上坡时可能误判为起始线，后来再加上图像的处理，试验表明这种方法停车的准确率挺高的。

5.4.4 抗干扰处理

因为比赛时赛道外的背景色一致，所以赛道外的干扰处理的比较少，道内干扰主要是噪点跟反光，对于反光，只要实验室的条件够好，就可以避免很多软件上的处理，但是我们实验室的反光较厉害，而对于一些赛道上的一些噪点的话，本身因为这些噪点并不是很多，跟主要的黑线有明显的差别，主要不是连续的黑点并且跟图像的前后进行比较，要是偏差较大那么就直接剔除掉那些点。

5.5 转向和速度控制

将偏差的比例（Proportion）、积分（Integral）和微分（Differential）通过线性组合构成控制量，用这一控制量对被控对象进行控制，这样的控制器称PID控制器。在智能车的转向和速度控制方面，我们使用了鲁棒性很好的经典PID控制算法，配合使用理论计算和实际参数补偿的办法，使在寻线中的智能车达到了稳定快速的效果。PID控制算法主要用来控制智能车的快速加速、减速和速度的平稳。PID 控制是工程实际中应用最为广泛的调节器控制规律。问世至今70多年来，它以其结构简单、稳定性好、工作可靠、调整方便而成为工业控制的主要技术之一。单位反馈的PID 控制原理框图如图5.3：

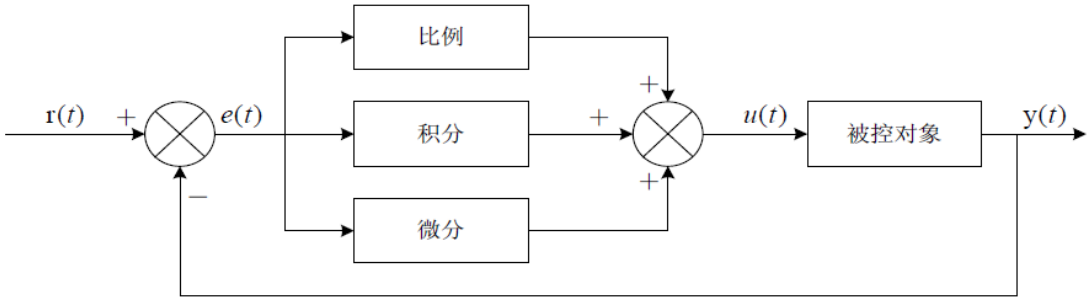


图5.3

常规的模拟PID控制系统原理框图如图5.3所示。该系统由模拟PID控制器和被控对象组成。图中r(t)是给定值，是系统的实际输出值，给定值与实际输出值构成控制偏差e(t)。　　$e(t)=r(t)-y(t)$

e(t)作为PID控制的输入，u(t)作为PID控制器的输出和被控对象的输入。所以模拟PID控制器的控制规律为

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

其中：　*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

PID 各个参数作用基本介绍：

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，则过渡过程越快，控制过程的静态偏差也就越小；但是*Kp*越大，也越容易产生振荡，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，静差小而又稳定的效果。积分环节的调节作用虽然会消除静态误差，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，这时系统在过渡时不会产生振荡；但是增大积分常数*Ti*会减慢静态误差的消除过程，消除偏差所需的时间也较长，但可以减少超调量，提高系统的稳定性。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，特别对高阶系统非常有利，它加快了系统的跟踪速度。但微分的作用对输入信号的噪声很敏感，对那些噪声较大的系统一般不用微分，或在微分起作用之前先对输入信号进行滤波。

5.5.1 位置式 PID 控制算法

由于计算机控制是一种采样控制，它只能根据采样时刻的偏差计算控制量，而不能像模拟控制那样连续输出控制量量，进行连续控制。由于这一特点上式积分项和微分项不能直接使用，必须进行离散化处理。离散化处理的方法为：以T作为采样周期，*K*作为采样序号，则离散采样时间*KT*对应着连续时间*t*，用矩形法数值积分近似代替积分，用一阶后向差分近似代替微分，可以得到离散的PID表达式为:

$$u_k = Kp[e_k + \frac{T}{Ti}\sum_{j=0}^{k} e_j + Td\frac{e_k - e_{k-1}}{T}]$$

其中 k—— 采样序号，k=0，1，2，……；

u_k—— 第k次采样时刻的计算机输出值；

e_k—— 第k次采样时刻输入的偏差值；

e_{k-1}—— 第k－1次采样时刻输入的偏差值；

Ki——积分系数，*Ki=Kp*T/Ti*;

Kd——微分系数，*Kd=Kp*Td/T*;

5.5.2 增量式 PID 算法

对位置式加以变换，可以得到PID 算法的另一种实现形式（增量式）：

$$\Delta u_n = u_n - u_{n-1} = k_p[(e_n - e_{n-1}) + \frac{1}{T_i}e_n + \frac{T_d}{T}(e_n - 2e_{n-1} + e_{n-2})]$$

增量式PID 具有以下优点：

(1) 由于计算机输出增量，所以误动作时影响小，必要时可用逻辑判断的方法关掉。

(2) 手动/自动切换时冲击小，便于实现无扰动切换。此外，当计算机发生故障时，由于输出通道或执行装置具有信号的锁存作用，故能保持原值。

(3) 算式中不需要累加。控制增量△u(n)的确定仅与最近k次的采样值有关，所以较容易通过加权处理而获得比较好的控制效果。

但增量式PID也有其不足之处：积分截断效应大，有静态误差；溢出的影响大。使用时，常选择带死区、积分分离等改进PID控制算法。

5.5.3 PID 参数整定

运用 PID 控制的关键是调整 KP、KI、KD 三个参数，即参数整定。PID 参数的整定方法有两大类：一是理论计算整定法。它主要是依据系统的数学模型，经过理论计算确定控制器参数；二是工程整定方法，它主要依赖工程经验，直接在控制系统的试验中进行，且方法简单、易于掌握，在工程实际中被广泛采用。由于智能车系统是机电高耦合的分布式系统，并且要考虑赛道的具体环境，要建立精确的智能车运动控制数学模型有一定难度，而且我们对车身机械结构经常进行修正，模型参数变化较为频繁，理论计算整定法可操作性不强，最终我们采用了工程整定方法。此外，我们先后实验了几种动态改变 PID 参数的控制方法。

5.6 转向舵机的 PID 控制算法

我们采用了位置式 PID 控制算法对舵机进行闭环控制，根据往届的技术资料和实际测试，将每场图像的黑线中心加权平均值与舵机 PID 参考角度值构成一次线性关系。

经过反复测试，我们选择的 PID 调节策略是：

(1) 将积分项系数置零，只用 PD，我们发现相比稳定性和精确性，舵机在这种随动系统中对动态响应性能的要求更高。更重要的是，在 KI 置零的情况下，我们通过合理调节 Kp，发现车能够在直线高速行驶时仍能保持车身非常稳定，没有震荡，基本没有必要使用 KI 参数；

(2) 微分项系数 KD 使用定值，不同的赛道使用不同的 KD 值。舵机在一般赛道中都需要较好的动态响应能力。

(3) 对 Kp，我们使用了二次函数曲线，Kp 随黑线位置与中心值的偏差呈二次函数关系增大，在程序中具体代码如下：

```
TurnKp = 8.0 + (float)(Aver * Aver) / Turnk;
```

其中 Aver 是偏差值加权平均平均值，TurnKp 是转向比例，Turnk 为定值。

经不断调试，最终我们选择了一组 PID 参数，得到了较为理想的转向控制效果。

5.7 驱动电机的 PID 控制算法

对于速度控制，我们采用了增量式 PID 控制算法，基本思想是直道加速，弯道减速。经过反复调试，将每场图像得到的黑线位置与速度 PID 参考速度值构成二次曲线关系。在实际测试中，我们发现小车直道和弯道相互过渡时加减速比较灵敏，与舵机转向控制配合得较好。

在程序中具体代码如下：

```
uk = error * Kp + (error - error1) * Kd;
```

其中uk为输出速度值，error为本场误差，error1为上一场误差，Kp为比例系数，Kd为微分系数。

开始调的PID在直道加速太慢，经过试验调整我们的PID调整的较好，直道能迅速加速。

5.8 本章小结

本章详细介绍了本智能车设计中的软件内容，从摄像头信号采样，信号读取，信号处理，信号分析到给出控制参量深化的给出了本设的程序设计理念和内容。PID 的设定，软件内容是小车的重点，我们在这方面花了好多功夫，也得出了很多经验。PID 的调试是后来调整的重点，决定着车的稳定性。

第六章　系统调试

6.1 开发工具

程序的开发是在组委会提供的CodeWarrior IDE下进行的，包括源程序的编写、编译和链接，并最终生成可执行文件。CodeWarrior for S12X 是面向以HC1和S12X为CPU的单片机嵌入式应用开发软件包。包括集成开发环境IDE、处理器专家库、全芯片仿真、可视化参数显示工具、项目工程管理器、C 交叉编译器、汇编器、链接器以及调试器。现代单片机具有很强的在线编程和调试功能，Freescale S12系列单片机就具有BDM（Background Debug Mode）功能，可以实现在线程序下载和在线背景调试功能。使用清华大学Freescale MCU/DSP 应用开发研究中心开发的BDM for S12，配合CodeWarrior的hiwave.exe在线调试器，即可实现从程序下载到在线调试的，使用BDM 来下载程序完整功能。把编译好的程序下载到单片机里运行。因此它可以通过使用两种开发工具：简单串行电缆或低成本的BDM，来完成调试功能。编译器是用的免费发放的CodeWarrior5.0版本编译器。

赛车的硬件开发工具主要为AltiumDesignerWinter09，通过该软件来完成车模固定板的绘制，电路原理图的绘制以及PCB板制作，功能非常强大。

6.1.1 软件开发平台

此次智能车大赛的软件开发平台为Code Warrior开发软件。其使用界面如图6.1.1所示。

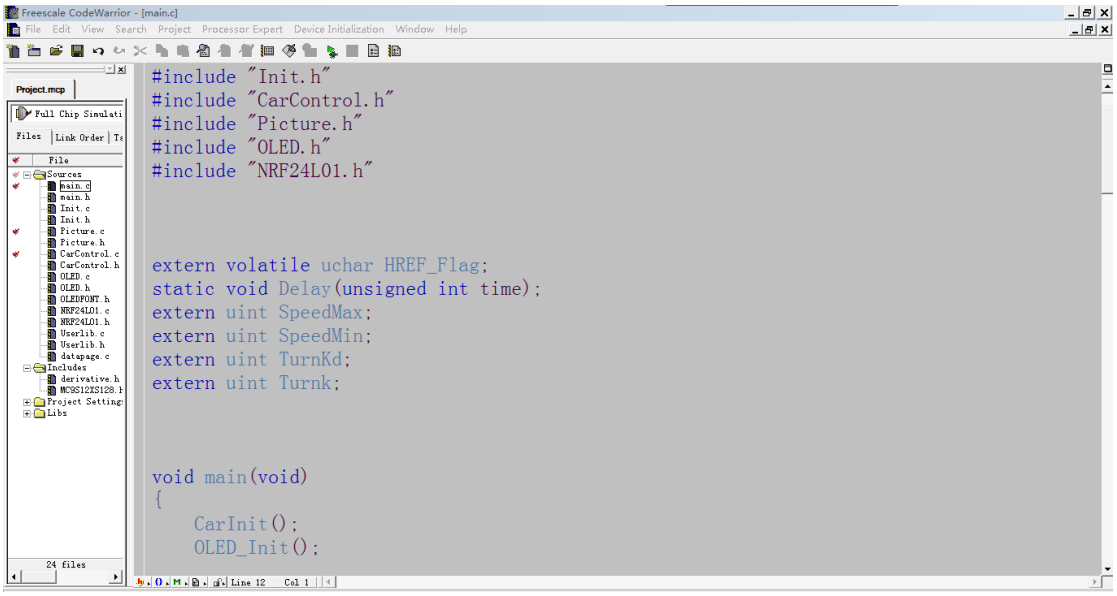


图6.1.1

在CodeWarrior软件中可以使用汇编语言或C语言，以及两种语言的混合编程。用户可在新建工程时将芯片的类库添加到集成环境开发环境中，工程文件一旦生成就是一个最小系统，用户无需再进行繁琐的初始化操作，就能直接在工程中添加所需的程序代码。该版本的PE模式非常方便，不用再去自己写一大堆的初始化程序，各模块的初始化只需要在相应的窗口进行设置即可。程序的下载也比较方便，不用像以前那样需要先擦除，通过设置后便可方便的直接下载。

6.1.2 串口调试模块

串口调试模块可以收集二值化图像的分析，和软件内部运行情况等等，用处很广。如图6.1.2。

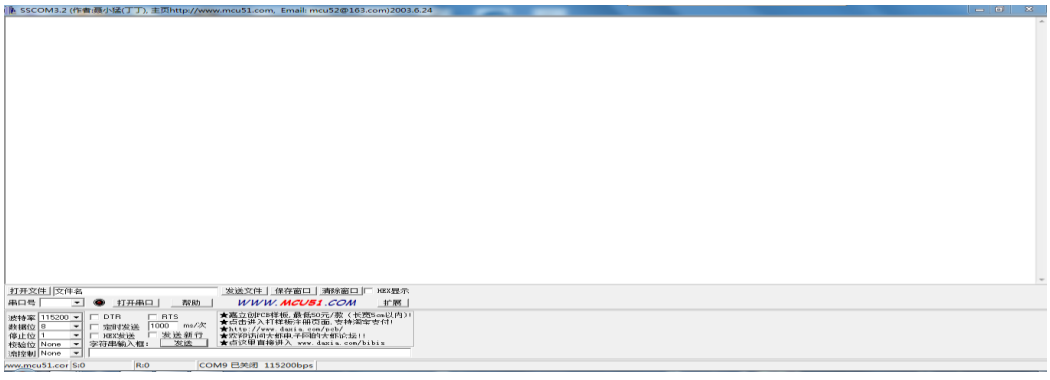


图6.1.2

6.2 调参数工具

我们使用OLED和按键的方法来进行参数的调整。OLED显示屏小巧，重量轻，可以显示汉字，用的AD键盘，有四个按键，一个加一个减还有一个确认，另一个备用。可以对参数的调整，还可以通过OLED看软件内部运行的情况，非常方便。OLED如图6.1.3所示，无线模块主要用来防撞，无线用的NRF24L01，如图6.14所示，调试模块如图6.1.5所示。

图6.1.3

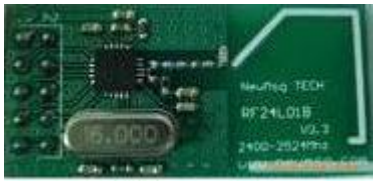


图6.14

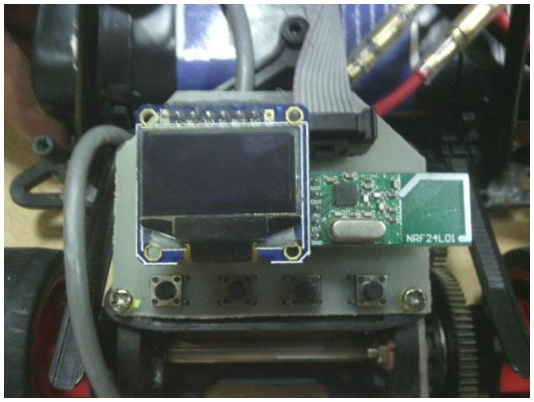


图6.1.5

6.3 计时器

为了方便显示时间和速度，我们用51单片机芯片，红外对管和一个1602显示屏自制了一个计时器

定时器精度为0.1秒。虽然说精确度不高，但在调试过程中已经够用了。可以完成计时器程序大致分为两个部分，一个是计时功能实现，另一个就是计时器初始化（端口初始化和显示），计时完成后自动计算出速度，有利于我们进一步调试，为了防止小车通过时多次触发，加入了延时程序（在计时开始后程序主体经过1s 后才再次响应其他触发），最终的程序相当稳定。如图6.1.5

图6.1.5

6.4 本章小结

本章主要介绍了系统调试用到的工具及算法的调试和实验结果。选CodeWarrior5.0forS12X作为软件开发平台， 通过串口对图像采集和图像处算法进行辅助调试， 参数工具的调节大大方便了系统的调试。

第七章　模型车主要技术参数说明

7.1 改造后的车模总体重量，长宽尺寸

表7-1 改造后车模参数

类别	参数
车模总体重量(kg)	1.2
车长（mm）	292
车宽（mm）	191
电路功耗(W)	15
电容总量（uF）	1557
传感器个数	4
传感器种类	一个COMS摄像头，两个红外对管，一个编码器
附加伺服电机个数	0
赛道信息检测精度	1cm
信息检测频率	60Hz

结 论

自从组队参加“飞思卡尔”杯智能汽车竞赛以来，我们小组从刚开始接触飞思卡尔芯片到查阅相关资料、初步设计结构和组装车模、编写程序、结构改造，最后车模的确立，最后终于完成了最初的设计目标，定下了现在这个设计方案。 在这份技术报告中，我们主要介绍了在准备比赛时的基本思路，包括机械设计、电路以及最重要的控制算法。在机械设计方面，我们分析了舵机转向系统的改进办法，主销内倾和后倾的调整以及在其他细节方面的优化。在电路方面，主板设计、调试等模块的设计，经过不断实验，最后确立了我们最终的电路图。在程序方面，利用比赛推荐的开发工具调试程序，经过小组成员不断讨论、改进、优化、终于设计出一套比较稳定快速的程序。在这套算法中，我们判断赛道然后结合路况调整车速，做到直道加速、弯道减速，保证在最短时间内跑完全程。

现在，面对即将到来的全国总决赛，在历时近九个月的充分准备，经过西部赛成功进入总决赛，我们有信心在全国比赛中取得优异成绩。

这份技术报告是我们小组辛勤劳动的结晶， “飞思卡尔”杯智能汽车竞赛将成为我们美好的回忆，经过这场比赛我相信每个参加比赛的成员都会有不少的收获。

致谢

在这几个月的备战过程中，我们遇到了许多困难，从最开始的机械设计、摄像头调试到后来的程序烧写，从电路板设计到系统搭建，一个个问题的解决见证了我们这支队伍的成长。期间离不开老师、同学们的大力支持，是他们的帮助让我们做得更好。

感谢学校在场地、器材和经费等方面的大力支持，在此特别感谢一直支持和关注智能车比赛的学校和学院领导以及姚惠林老师和路纲老师的悉心指导、没有他们在思路上的指点，我们将进展缓慢。同时也感谢比赛组委会能组织这样一项有意义的比赛。

参考文献

【1】 孙同景主编.《Freescale 9S12 十六位单片机原理及嵌入式开发技术》.北京-机械工业出版社 2008。

【2】 卓晴，黄开胜，邵贝贝．《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，2007.3

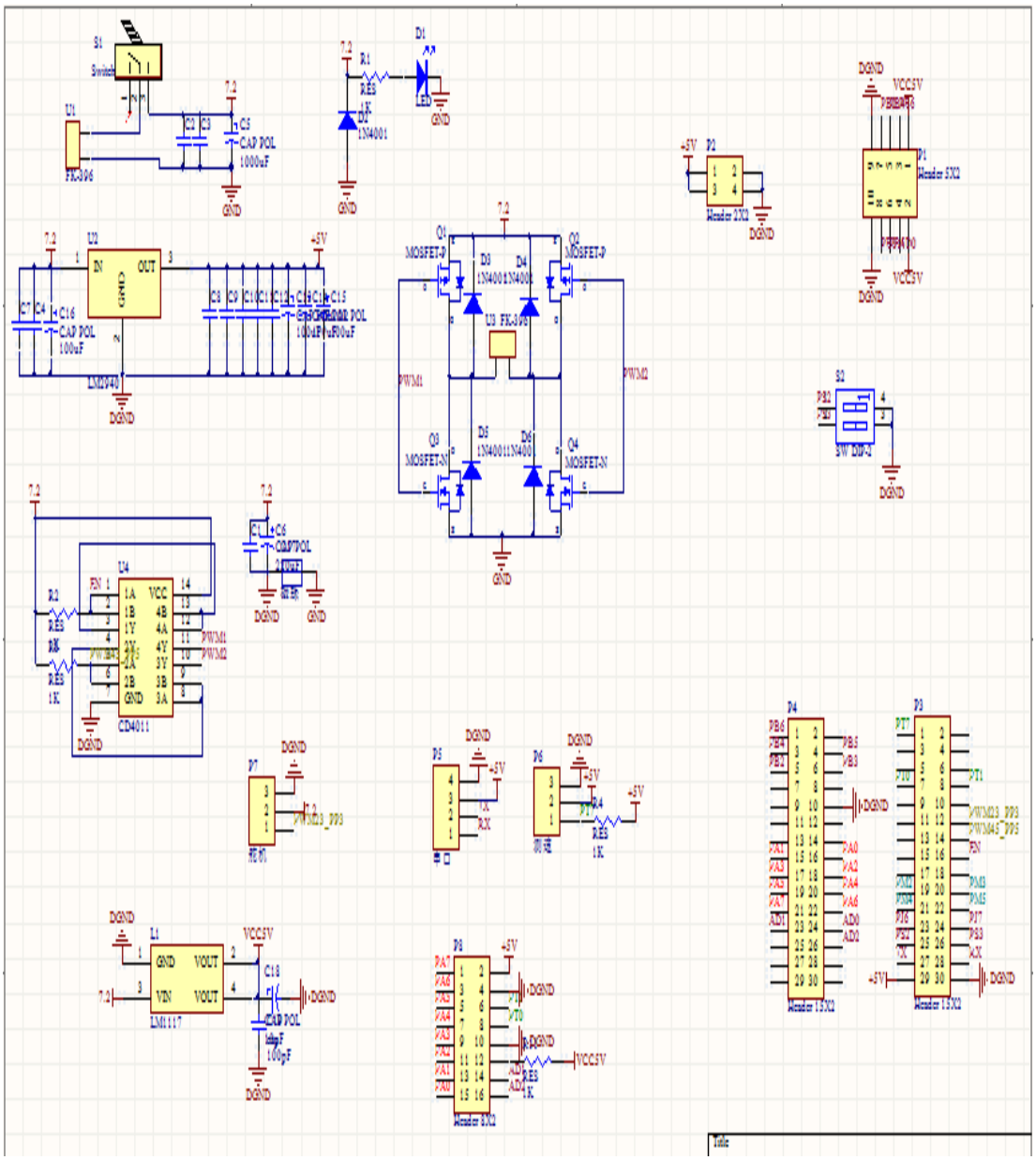
【3】 刘金琨.《先进 PID 控制及其 MATLAB 仿真》. 电 子 工 业 出 版 社

【4】 仲志丹，张洛平，张青霞．PID 调节器参数自寻优控制在运动伺服中的应用[J] ．洛阳工学院学报，2000，21（1）：57～60.

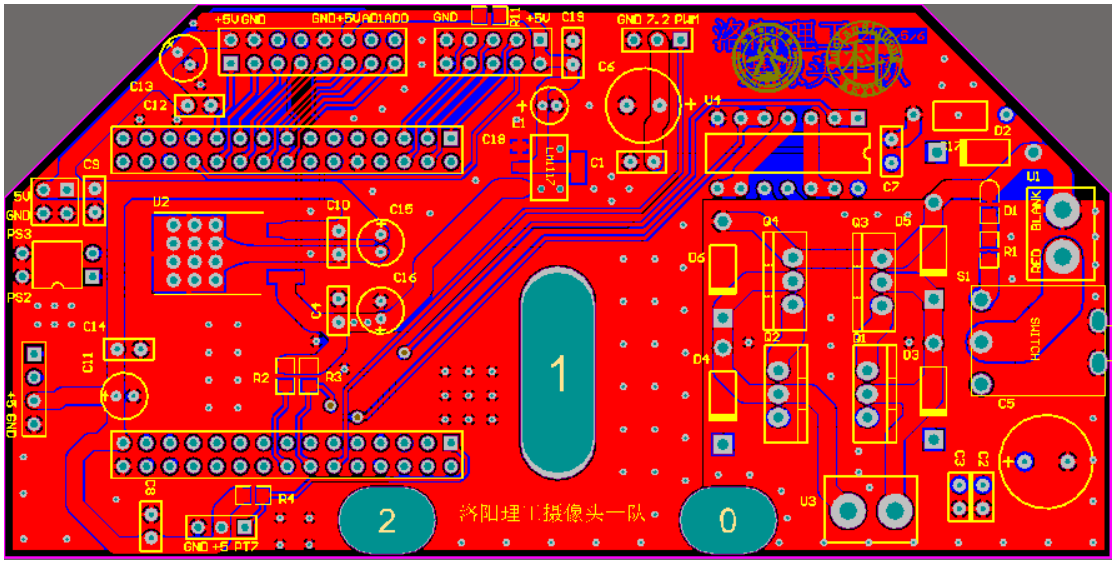
【5】 谭浩强著．C 程序设计．北京：清华大学出版社，2003.

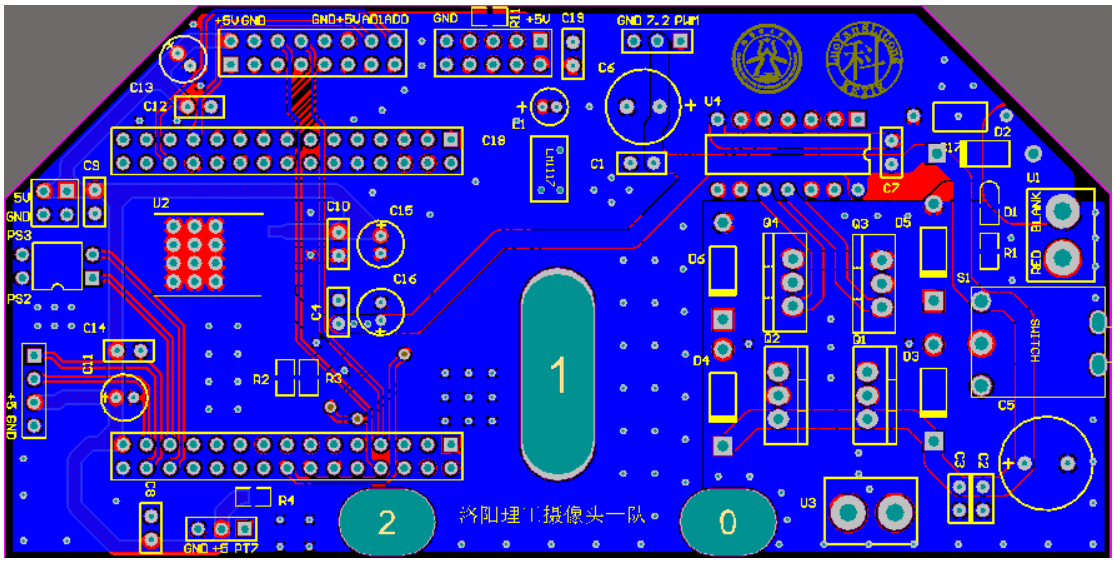
【6】 童诗白，华成英．模拟电子技术基础[M]．北京．高等教育出版社．2000

附录 A 核心电路板原理图



附录 B 核心电路板 PCB 图





附录 C　程序代码

```
/*********************************************************************
//函数名:初始化函数
//功能:XS128各模块的初始化
//说明:总线时钟设置为72M,场中断和行中断分别使用PT0进入和PT1,舵机PWM
       100kHz,电机PWM 20kHz,测速使用脉冲累加器 PT7,使用4ms定时器调节
       PID.PA口为图像数据输入口,PM口与拨码开关连接.串口通讯波特率为
       115200.
*********************************************************************/

#include"Init.h"

/*************************************************************/
//                      总线时钟设置
/*************************************************************/
void BusClock_Init(void)
{
    CLKSEL=0x00;              //关闭 PLLCLK
    PLLCTL_PLLON=1;           //打开 IPLL

    SYNR =0xC0 | 0x08;        //Busclock=PLLclock/2    72MHz
    REFDV=0x80 | 0x01;
```

```
//Pllclock=2*fosc*(1+SYNDIV)/(1+REFDIV)=144MHz fosc=16MHz

    POSTDIV=0x00;             //PLLCLK=VCOCLK

    while(CRGFLG_LOCK!=1);    //等待Pll时钟稳定
    CLKSEL_PLLSEL=1;          //PLLCLK 使能
}

/*************************************************************/
//                    AD采集设置
/*************************************************************/
void ATD_Init(void)
{
    ATD0CTL1=0x20;//10位精度
    ATD0CTL2=0x40;//禁止外部触发,标志位快速清零,中断禁止
    ATD0CTL3=0x98;//右对齐,转换序列长度为3,通道0,1,2, No FIFO模
式,Freeze模式下继续转换
    ATD0CTL4=0x43;//4AD采样周期,ATDClock=[BusClock*0.5]/[PRS+1]  ;
PRS=15, divider=32 ?
    ATD0CTL5=0x30;//特殊通道禁止,扫描模式连续采样,多通道采样 从 AN0
                  //改这可以不连续任选几个通道
    ATD0DIEN=0x00;//禁止数字输入
}                     //左对齐，无符号
```

```
/*************************************************************/
//                      捕捉通道和中断设置
/*************************************************************/
void ECT_Init(void)
{
    TIOS=0x00; //外部输入捕捉0,1,2,3通道

    TSCR1_TFFCA=0; // FAST FLAG CLEAR  读取即清零

    TCTL4_EDG0B = 0;    //上升沿
    TCTL4_EDG0A = 1;
    TCTL4_EDG1B = 1;    //下降沿
    TCTL4_EDG1A = 0;

    TIE_C0I = 0;        //行中断先关闭,在场中断中延时开启
    TIE_C1I = 1;        //场中断开启

    PACTL = 0x40;       //脉冲计数开启 下降沿有效 PT7
    PACNT=0x0000;       //计数器清零
    TSCR1_TEN=1; //系统时钟开启
}

/*************************************************************/
```

```
//                      串口设置
/*************************************************************/
void SCI_Init(void)
{
    SCI0BDL=0x28;     //BUSClock 72M  波特率115200
    SCI0BDH=0x00;
    SCI0CR1=0x00;
    SCI0CR2_TE=1;    //使能发射
    SCI0CR2_RE=1;    //使能接收
}

void SCI_Output(uchar ch)         //SCI数据发送
{
    while(!SCI0SR1_TDRE);
    SCI0DRL=ch;
}

byte SCI_Input(void)              //SCI数据接收
{
    uchar ch1;
    while(!SCI0SR1_RDRF);
    ch1=SCI0DRL;
    return ch1;
}
```

```
/*************************************************************/
//                    PWM设置
/*************************************************************/
void PWM_Init(void)
{
    PWMCTL_CON45=1;   //通道45组合    电机
    PWMCTL_CON23=1;   //通道23组合    舵机

    PWMCLK=0x00;      // 通道 3 选择时钟 B, 通道 5  选择时钟 A
    PWMPRCLK=0x52;    // 时钟预分频寄存器 ,时钟 B 频率为 Bus Clock/8 ,
时钟 A 频率为 Bus Clock/16

    PWMPOL=0x28;      //通道5周期开始为高电平输出 通道3 低电平
    PWMCAE=0x00;      //左对齐输出

    PWMPER45=900;
    PWMDTY45=0;       // 通道 3 输出一直为低低电平 , 和通道 2 形成电
压差 , 用于驱动电机

    PWMPER23=37500;   // S3010使用60Hz
    PWMDTY23=Median;   // 为了设定占空比 = PWMDTY45/PWMPER45*100%(左
对齐)
```

```
    PWME=0x28;          // 通道使能寄存器 ， 电机 5通道 ， 舵机 3通道使能
}

/*************************************************************/
//                    I/O设置
/*************************************************************/
void IO_Init(void)
{
    DDRM=0xFB;
    DDRP=0xFF;
    DDRB=0xFF;
    DDRA=0x00;
    DDRJ=0x00;
    DDRS=0x00;
    DDRT=0x00;

    PERS=0xFF;
    PERP=0xFF;
    PERM=0XFF;       //PM口上拉配置
    PUCR=0xFF;
}

/*************************************************************/
//                    初始化函数
```

```
/************************************************************/
void CarInit(void)
{
     DisableInterrupts
     BusClock_Init();
     PWM_Init();
     ECT_Init();
     SCI_Init();
     ATD_Init();
     IO_Init();
}
//主函数及调试函数
#include <hidef.h>      /* common defines and macros */
#include "derivative.h"      /* derivative-specific definitions */
#include "Userlib.h"
#include "Init.h"
#include "CarControl.h"
#include "Picture.h"
#include "OLED.h"
#include "NRF24L01.h"

extern volatile uchar HREF_Flag;
static void Delay(unsigned int time);
extern uint SpeedMax;
```

```
extern uint SpeedMin;
extern uint TurnKd;
extern uint Turnk;
extern uint Kp;
extern uint Kd;

void main(void)
{
    CarInit();
    OLED_Init();
#ifdef Debug     //调试模式
    OLED_ShowString(0,0,"SpeedMax=");
    OLED_Debug(72,0,&SpeedMax);
    OLED_ShowString(0,2,"SpeedMin=");
    OLED_Debug(72,2,&SpeedMin);
    OLED_ShowString(0,4,"Kp=");
    OLED_Debug(25,4,&Kp);
    OLED_ShowString(60,4,"Kd=");
    OLED_Debug(83,4,&Kd);
    OLED_ShowString(0,6,"Tk=");
    OLED_Debug(24,6,&Turnk);
    OLED_ShowString(56,6,"Td=");
    OLED_Debug(88,6,&TurnKd);
    DDRB=0xBF;                    //无线I/O初始化
```

```
    Init_IO();                          // 初始化IO
    TX_Mode();                          // 设置为接收模式
#endif
    EnableInterrupts;
    for(;;)
    {
        if(HREF_Flag==1)
        {
            HREF_Flag = 0;
            Binary();
            Edge();
            Midline();
            CarTurn();
            #ifndef Debug      //调试模式
            if(PTS_PTS3==0)    //拨上 拨码开关3进入调试
            {
                PWME=0x08;
                Picture_SCI();//发送串口数据
            }
            TIE_C1I=1;         //开中断
            #endif
        }
        _FEED_COP();
    }
```

```
}
void OLED_Debug(uchar x,uchar y,uint *p)
{
    uint temp;
    temp = *p;
    do
    {
        Delay(70);
        while(ATD0STAT2 == 0);
        if(ATD0DR0 > 150 && ATD0DR0 < 250)
        {
            temp++;
        }
        else if(ATD0DR0 > 320 && ATD0DR0 < 600)
        {
            temp--;
        }
        OLED_DebugShow(x,y,p);
        *p = temp;
        Delay(70);
        while(ATD0STAT2 == 0);
    }while(ATD0DR0 > 10);
}
void OLED_DebugShow(uchar x,uchar y,uint *p)
```

```
{
    uint temp;
    temp = *p;
    OLED_ShowChar(x,y,temp / 1000 + '0');
    OLED_ShowChar(x+8,y,temp / 100 % 10 + '0');
    OLED_ShowChar(x+16,y,temp / 10 % 10 + '0');
    OLED_ShowChar(x+24,y,temp % 10 + '0');
}
void OLED_DebShowChar(uchar x,uchar y,uchar *p)
{
    uint temp;
    temp = *p;
    OLED_ShowChar(x,y,temp / 1000 + '0');
    OLED_ShowChar(x+8,y,temp / 100 % 10 + '0');
    OLED_ShowChar(x+16,y,temp / 10 % 10 + '0');
    OLED_ShowChar(x+24,y,temp % 10 + '0');
}

int abs_(int x)        //绝对值函数
{
    if(x > 0)  return x;
    else return(-x);
}
static void Delay(unsigned int time) //延时函数
```

```
{
    uint x,y;
    for(x=0;x<time;x++)
        for(y=0;y<10000;y++);
}
/*********************************************************************
**                          OLED库函数
**XS128智能车专用版本   去掉了显示的缓存节省RAM 只能显示8*16的字符
*********************************************************************/
#include "OLED.h"
#include "OLEDFONT.h"

#define DC  PORTB_PB3   //命令/数据标志
#define RST PORTB_PB2   //复位
#define D1  PORTB_PB4   //时钟线
#define D0  PORTB_PB5   //数据线
#define OLED_CMD 0      //标志位
#define OLED_DATA 1

void OLED_Delay(unsigned int time);//延时函数

/**********************************************************
                    写数据/命令
**********************************************************/
```

```
void OLED_Write(unsigned char data,unsigned char cmd)//cmd为0写命令 为1写数据
{
    unsigned char i,temp;
    DC = cmd;
    for(i = 0;i < 8;i++)
    {
        D0 = 0;
        temp = data & 0x80;
        if(temp > 1)  temp = 1;//该编译器不为1的全为假(0)
        D1 = temp;
        data <<= 1;
        D0 = 1;
    }
}
/*************************************************************
                        OLED初始化
*************************************************************/
void OLED_Init(void)
{
    unsigned char temp_l,temp_k;  //for循环变量
    RST = 0;    //复位
    OLED_Delay(1000);
    RST = 1;
```

```
    OLED_Write(0x81,OLED_CMD);    //设置对比度命令
    OLED_Write(0xff,OLED_CMD);    //对比度值 0-0xff  值越大对比度越高
    OLED_Write(0xAF,OLED_CMD);     //开启显示
    OLED_Write(0x8D,OLED_CMD);     //设置是否开启电荷泵
    OLED_Write(0x14,OLED_CMD);     //开启电荷泵

    for(temp_l = 0;temp_l < 8;temp_l++) //清屏循环
    {
        OLED_Write(0xB0 + temp_l,OLED_CMD);    //设置页地址0-7，其低三位的值对应着 GRAM 的页地址
        OLED_Write(0x00,OLED_CMD);          //设置显示时的起始列地址低四位   00 - 0f
        OLED_Write(0x10,OLED_CMD);         //设置显示时的起始列地址高四位   10 - 1f
        for(temp_k = 0;temp_k < 128;temp_k++)
        {
            OLED_Write(0x00,OLED_DATA);
        }
    }
}
/*************************************************************
                        打点函数
*************************************************************/
```

```
void OLED_DrawPoint(unsigned char x,unsigned char y,unsigned char
data)/*d = 1显示点,d = 0不显示*/
{
    unsigned char pos;
    unsigned char tempL,tempH;
    if(x > 127 || y > 63) return;//超出范围了
    pos = 7 - y;
    tempH = (unsigned char)((127 - x) / 16);
    tempL = (unsigned char)((127 - x) % 16);
    OLED_Write(0xB0 + pos,OLED_CMD);   //设置页地址0-7，其低三位的值对
应着 GRAM 的页地址
    OLED_Write(0x00 + tempL,OLED_CMD);        //设置显示时的起始列地址
低四位   00 - 0f
    OLED_Write(0x10 + tempH,OLED_CMD);        //设置显示时的起始列地址
高四位   10 - 1f
    OLED_Write(data,OLED_DATA);
}
/*************************************************************
                        显示字符
*************************************************************/
void OLED_ShowChar(unsigned char  x,unsigned char  y,unsigned char
chr)
{                       /*字符显示为起始位置  起始位置x<121 y<55*/
    unsigned char  temp,temp1,t;
```

```
    chr = chr - ' ';//得到偏移后的值
    temp1 = x;
    for(t = 0;t < 16;t++)
    {
        temp = asc2_1608[chr][t];        //调用16*8字体
       OLED_DrawPoint(x,y,temp);
       x++;
       if((x - temp1) == 8)
       {
            y++;
            x = temp1;
       }
    }
}
/*****************************************************************
                    显示字符串
*****************************************************************/
void OLED_ShowString(unsigned char  x,unsigned char  y,char  *p)
{
    while(*p != '\0')
    {
        OLED_ShowChar(x,y,*p);
        p++;                              /*屏幕左上角为坐标(0,0)*/
        x += 8;  //调用12*6字体
```

```
    }
}
/**************************************************************
                        显示汉字
**************************************************************/
void OLED_ShowHanZi(unsigned char  x,unsigned char  y,char  *p)
{
    unsigned char  temp,i,l;
    unsigned char  y0 = y;
    for(i = 0;i < 32;i++)
    {
         temp = *(p++);
        for(l = 0;l < 8;l++)
          {
            if(temp & 0x80)
            {
                  OLED_DrawPoint(x,y,1);
            }
            else
            {
                OLED_DrawPoint(x,y,0);
            }
              temp <<= 1;
              y++;
```

```
                if((y - y0) == 16)
                {
                        y = y0;
                        x++;
                        break;
                }
            }
    }
}

void OLED_Delay(unsigned int time)
{
    for(;time>0;time--)
      {
          asm("nop");
      }
}
```