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：　西华大学
队伍名称：　川工第一漂
参赛队员：　王超
　　杨明
　　赵梦竹
带队教师：　董秀成
　　古世甫

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：_____

目录

第一章 引言

为加强大学生实践、创新能力和团队精神的培养，促进高等教育教学改革，受教育部高等教育司委托，由教育部高等学校自动化专业教学指导分委员会主办全国大学生智能汽车竞赛。该竞赛是以智能汽车为研究对象的创意性科技竞赛，是面向全国大学生的一种具有探索性工程实践活动，是教育部倡导的大学生科技竞赛之一。该竞赛以“立足培养，重在参与，鼓励探索，追求卓越”为指导思想，旨在促进高等学校素质教育，培养大学生的综合知识运用能力、基本工程实践能力和创新意识，激发大学生从事科学研究与探索的兴趣和潜能，倡导理论联系实际、求真务实的学风和团队协作的人文精神，为优秀人才的脱颖而出创造条件。该竞赛由竞赛秘书处设计、规范标准硬软件技术平台，竞赛过程包括理论设计、实际制作、整车调试、现场比赛等环节，要求学生组成团队，协同工作，初步体会一个工程性的研究开发项目从设计到实现的全过程。该竞赛融科学性、趣味性和观赏性为一体，是以迅猛发展、前景广阔的汽车电子为背景，涵盖自动控制、模式识别、传感技术、电子、电气、计算机、机械与汽车等多学科专业的创意性比赛。该竞赛规则透明，评价标准客观，坚持公开、公平、公正的原则，力求向健康、普及、持续的方向发展。

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。

1.1 器材限制规定

1.1.1. 须采用统一指定的车模。本届比赛指定采用三种车模，分别用于三个赛题组。

各赛题组车模运行规则：

光电组，摄像头组：车模正常运行，车模使用 A 型车模（摄像头组）、B 型车模（光电组）。车模运行方向为，转向轮在前，动力轮在后。
电磁组：车模直立行走，使用 C 型车模。车模运行时只允许动力轮着地，车模直立行走。
1.1.2 须采用飞思卡尔半导体公司的 8 位、16 位、32 位处理器作为唯一的微控制器。

1.1.3. 三个赛题组使用传感器限制：

参加电磁赛题组不允许使用光学传感器获得道路的光学信息，但是可以使用光电码盘测量车速；

参加光电赛题组不允许使用图像传感器获取道路图像信息进行路径检测；

参加摄像头赛题组可以使用光电管作为辅助检测手段；

非电磁组的赛道不保证有电磁信号；

1.1.4. 其他事项

如果车模中禁止改动的部件发生损坏，需要使用相同型号的部件替换；摄像头组车模改装完毕后，车模尺寸不能超过：250mm 宽和 400mm 长。光电组车模改装完毕后，车模尺寸不能超过 350mm 宽和 400mm 长。电磁组车模改装完毕后，车模尺寸宽度不超过 250mm，长度没有限制。

1.2 有关赛场的规定

1. 赛道基本参数（不包括拐弯点数、位置以及整体布局）见附件三；

2. 比赛赛道实际布局将在比赛当日揭示，在赛场内将安排采用与制作实际赛道相同的材料所做的测试赛道供参赛队进行现场调试；

1.3 裁判及技术评判

竞赛分为分赛区（省赛区）和全国总决赛两个阶段。其中，全国总决赛阶段是在全国竞赛组委会秘书处指导下，与决赛承办学校共同成立竞赛执行委员会，下辖技术组、裁判组和仲裁委员会，统一处理竞赛过程中遇到的各类问题。

全国和分赛区（省赛区）竞赛组织委员会工作人员，包括技术评判组及现场裁判组人员均不得在现场比赛期间参与任何针对个别参赛队的指导或辅导工作，不得泄露任何有失公允竞赛的信息。在现场比赛的时候，组委会可以聘请参赛队伍带队教师作为车模检查监督人员。

在分赛区（省赛区）阶段中，裁判以及技术评判由各分赛区（省赛区）组委会参照上述决赛阶段组织原则实施。

1.4 分赛区、总决赛比赛规则

分赛区和总决赛的比赛规则相同，都具有电磁组、光电组和摄像头组三个赛题组比赛。三个赛题组比赛原则上在同一个场馆同时进行，所遵循的比赛规则也基本相同的。三个赛题组分别独立进行成绩排名。

分赛区和总决赛的现场比赛均包括初赛与决赛两个阶段。下面列出的现场预赛、决赛阶段的比赛规则适用于各分赛区及总决赛的三个赛题组。

1.4.1 初赛与决赛规则

1) 初赛阶段规则

参赛队根据比赛题目分为三个组，并以抽签形式决定组内比赛次序。

比赛分为两轮，三个赛题组同时在三个赛道上进行比赛，每支参赛队伍可以在每轮比赛之前有 10 分钟的现场调整时间。在此期间，参赛队伍只允许对赛车的硬件（不包括微控制器芯片）进行调整。第二轮比赛在同一赛道沿逆向进行。在每轮比赛中，选手首先将赛车放置在起跑区域内赛道上，赛车至少静止两秒钟后自动启动。对于电磁组不要求赛车静止两秒钟启动。

每辆赛车在赛道上跑一圈，以计时起始线为计时点，跑完一圈后赛车需要自动停止在起始线之后三米之内的赛道内，如果没有停止在规定的区域内，比赛计时成绩增加 1 秒。对于电磁组比赛不要求车模停止在起跑线之后三米之内的赛道上。每辆赛车以在两个单轮成绩中较好的一个作为赛车最终初赛成绩；计时由电子计时器完成并实时显示。根据参赛队伍数量，由比赛组委会根据成绩选取一定比例的队伍晋级决赛。晋级决赛的赛车在决赛前有 10 分钟的调整时间。在此期间，参赛队伍只允许对赛车的硬件（不包括微控制器芯片）进行调整。技术评判组将对全部晋级的赛车进行现场技术检查，如有违反器材限制规定的(指本规则之第一条)即可取消决赛资格，由后备首名晋级代替。由裁判组申报组织委员会批准公布决赛名单。全部车模在整个比赛期间都统一放置在车模的展示区内。

2）决赛阶段规则

参加决赛队伍按照预赛成绩进行排序，比赛顺序按照预赛成绩的倒序进行。决赛的比赛场地使用一个赛道。决赛赛道与预赛赛道形状不同，占地面积会增大，赛道长度会增加。电磁组可能另外单独铺设跑道。

每支决赛队伍只有一次比赛机会，在跑道上跑一圈，比赛过程与要求同预赛阶段。计时由电子计时器完成并实时显示。预赛成绩不记入决赛成绩，只决定决赛比赛顺序。没有参加决赛阶段比赛的队伍，预赛成绩为最终成绩，参加该赛题组的排名。

1.4.2 比赛过程规则

按照比赛顺序，裁判员指挥参赛队伍顺序进入场地比赛。同一时刻，一个场地上只有一支队伍进行比赛。在裁判员点名后，每队指定一名队员持赛车进入比赛场地。参赛选手有 30 秒的现场准备时间。准备好后，裁判员宣布比赛开始，选手将赛车放置在起跑区内，即赛车的任何一部分都不能超过计时起始线。赛车应在起跑区静止两秒钟以上，然后自动出发（电磁组直立行走车模不要求静止两秒钟）。赛车应该在 30 秒之内离开出发区，沿着赛道跑完一圈。由计时起始线传感器进行自动计时。赛车跑完一圈且自动停止后（电磁组直立行走车模不要求自动停止），选手拿起赛车离开场地，将赛车放置回指定区域。如果比赛完成，由计算机评分系统自动给出比赛成绩。

第二章：系统方案

2.1 系统策略

为了使赛车沿着赛道的中心更快更稳地行驶，我们把系统分为高精度路径检测模块、高精度速度控制模块、转向控制模块。通过把频率提高为 100MHz，通过合理的处理控制算法，同时加上各高精度模块的检测与控制，赛车能够很好地完成比赛任务。

2.2 系统设计方案

2.2.1 路径检测总体方案

摄像头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。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。因此我们采用方案一。

2.2.2 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，便于 K60 I/O 管脚直接获取。

综上所述，我们采用 CMOS 数字摄像头。

2.2.3 速度控制总体方案

在保证赛车稳定性的前提下，提高速度是我们设计的重点。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速度控制。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，实践证明开环控制在我们的智能车平台上精度不高，无法切实有效的提高我们小车的速度。

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。而且由于赛道上有斜坡，赛车安全通过斜坡的关键是控制上坡和下坡的速度，所以对于采用速度闭环控制的赛车来说通过斜坡就像是通过普通的直道一样的简单，不需要对斜坡进行专门的检测。

通过实际测试比较，采用速度闭环控制方案能显著提高我们小车的速度，因此我们采用速度闭环控制方案。

2.2.4 转向控制

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 波脉冲宽度有联系，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。调试过程中我们发现直接使用电池电压供电舵机也可以正常工作，我们把控制舵机的 PWM 波周期调整为 20 ms，把舵机臂长加长，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。

2.3 智能车技术参数

表 2.1 赛车主要技术参数

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）	280mm/170mm/350mm
车模轴距/轮距（毫米）	200mm/146mm
车模平均电流(匀速行驶)(毫安)	2100mA
电路电容总量（微法）	1500μF
传感器种类及个数	摄像头/1　编码器 /1
新增加伺服电机个数	0
赛道信息检测空间精度（毫米）	3mm（近端）、120mm(远端)
赛道信息检测频率（次/秒）	30
主要集成电路种类/数量	asm1117/2 ov7620/1 LM2940 LM2941 BTS7970
车模重量（带有电池）(千克）	1.5

第三章：机械系统的设计

车模的机械部分是影响智能车行驶性能最直接的部分。一个不良的机械系统会给智能车的控制带来难度。因此，车模的机械性能应该是优先考虑的问题。

3.1 舵机安装

本车模的默认舵机安装是卧式安装，为了使舵机重力在底盘上均匀分布，舵机被安装在车模前部的正中，舵机转盘没在舵机中部因此只有用一长一短两个拉杆连接，因此在转弯时会出现左转和右转有不同的效果，并且舵机较矮，力臂较短，力矩较小。

针对以上的问题，我们讨论决定采用立式安装，用自制的铝片作为舵机的固定支架，同时改用两个相同长度的拉杆使左右完全对称，并用腐蚀板制作一个较长的舵机柄。经过这些改造使舵机左右转动力平衡，增大车行进中的车轮转向速度。这样虽然在舵机转速不变的情况下加快了车轮的转角速度，但是舵机转向时增大了舵机转动负荷，可能会出现舵机齿轮被损坏的情况，但是，经过我们的测试没出现这个问题。安装效果如下图：

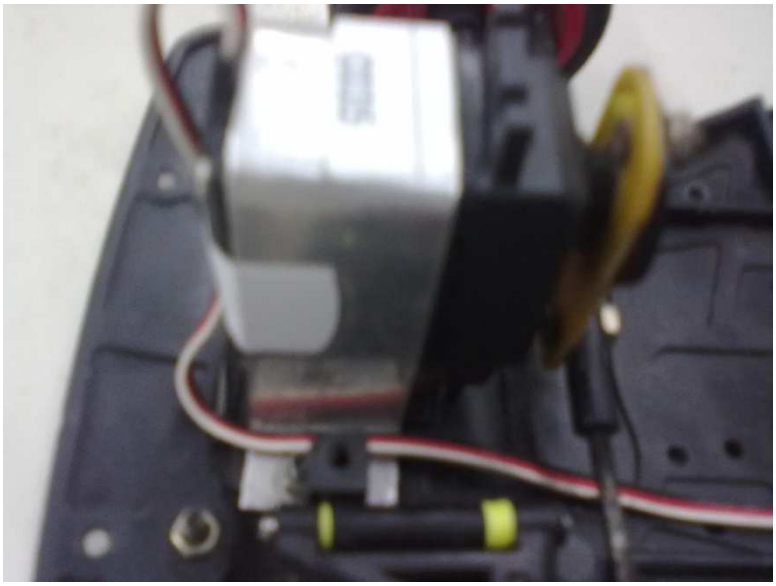

图 3.1 舵机安装

3.2 前轮的调整

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 4 个部分决定，反映了转向轮、主销和前轴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。在此部分中，我们通过改变主销上横向轴上的介子来改变主销后倾角。

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 编码器的安装

编码器在安装时主要考虑其能否和电机齿轮很好的咬合，速度较快时不至于出现速度检测不准的现象。我们把编码器齿轮通过差速轮传动和电机齿轮直接咬合，减小误差，同时使用自制的编码器安装架把其安装在靠近底板位置以达到降低重心的目的，增强后轮的抓地能力。安装效果如下图：

图 3.2 编码器安装

3.4 电路板的安装

电路板是智能车上的核心部分，在安装时，考虑到电路板的稳定性和模型车的重心等问题，我们把电路安装在舵机和电池支架中间的底板上。我们在底板上打 2 个螺孔，分别将电路板的 2 个固定孔固定在底板上，保证电路板的牢固性，同时还可以压低重心并使重心前移，提高模型车的过弯性能和稳定性。

3.5 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，重心高过弯时也容易翻车。安装过低，视野变小，图像变形严重，而且容易反光。过急弯时容易丢线，

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，用碳素管支撑摄像头，并通过车模自带的后轮宽度调节块固定碳素管，安装的高度大概在距底板 30cm 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。

图 3.3 摄像头支架及电路安装

第四章：硬件系统的设计

4.1 硬件系统框图

硬件系统主要包括电源模块、单片机模块、摄像头采集模块、速度传感模块和驱动模块。总体框图如下：

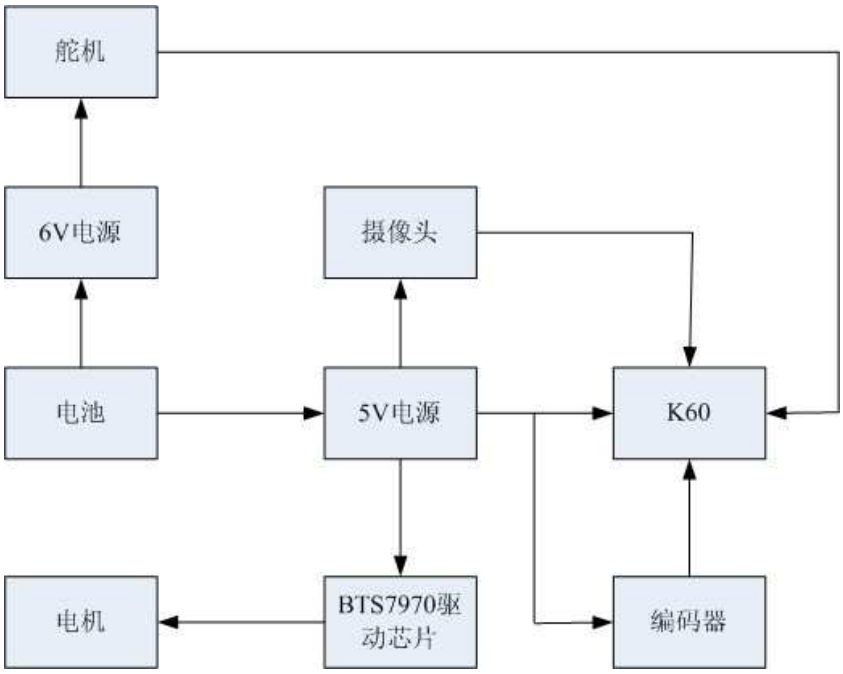

图 4.1 硬件系统总体框图

4.2 电源模块

电源管理模块为系统各个模块提供所需要的电源，可靠的电源方案是整个硬件电路稳定可靠运行的基础。设计中，除了需要考虑电压范围和电流大小等基本参数之外，还需要在电源转换效率、降低噪声、防止干扰等方面进行优化。

全部硬件电路的电源由 7.2V 2000mAh Ni-cd 蓄电池提供。由于电路中的不同电路模块所需要的工作电压和电流大小各不相同，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。主要包括以下不同的电压。

5V 电压。主要为单片机系统、摄像头、编码器以及部分接口电路提供电源，电压要求稳定、噪声小，电流容量大于 500mA

6V 电压。主要为舵机部分供电

4.2.1 5V 电源部分

稳压芯片主要有线性稳压芯片和开关稳压芯片两种。电机和舵机的突然启停会使电池电压骤变，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，我们采用的稳压芯片是 LM2940, LM2940 是一种低压差线性稳压集成电路，其最大输出电流达 1A,足以提供整个系统所需的功率。

此部分电源原理图如 4.2 所示：

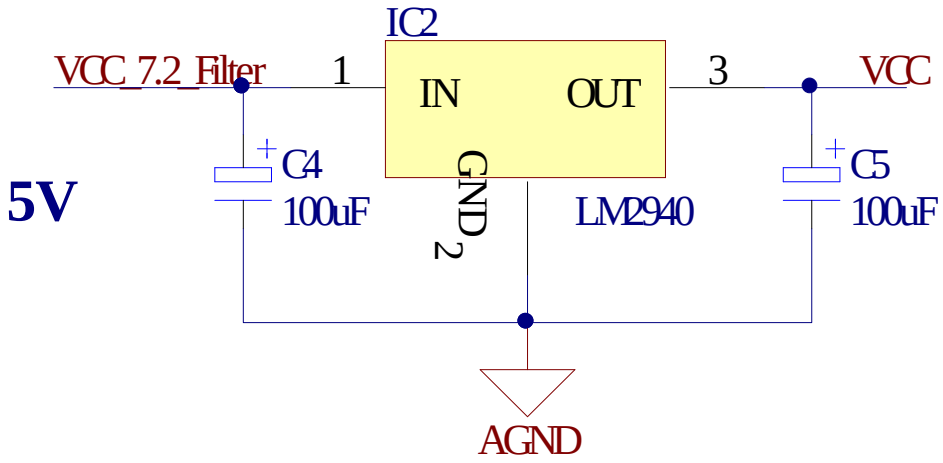

图 4.2 5V 电源

4.2.2 6V 电源部分

此部分电路为舵机提供电源，由于舵机对电源的稳定性较高，我们选择使用可调稳压器 LM2941。LM2941 也是一种低压差线性稳压集成电路，其最大输出电流达 1 A,输出精度高。

此部分电路原理图如下：

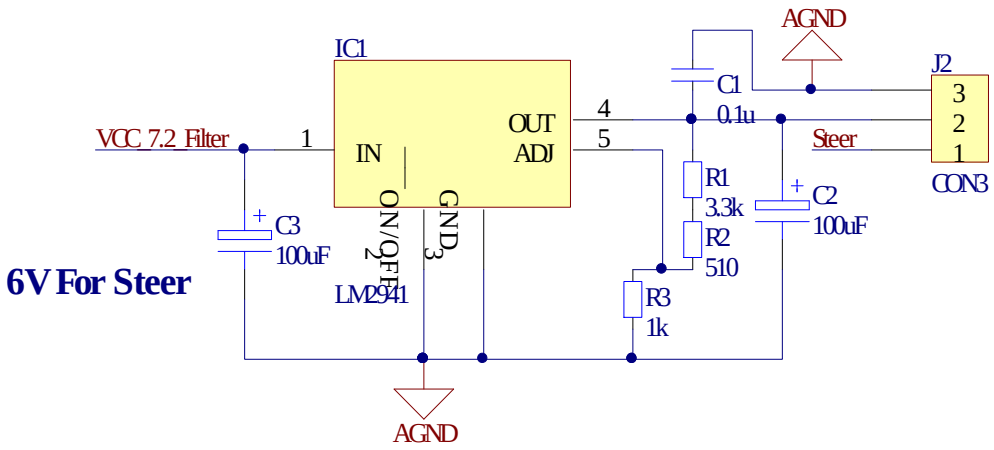

图 4.3、舵机电源

4.3 单片机系统模块

单片机是整个智能车系统的核心部分。我们所采用的单片机为飞思卡尔公司的 Kinetis ARM Cortex-M4 32 位系列的单片机 K60DN512，为 144 引脚封装。

K60DN512 芯片是 Kinetis 系列 K60 子系列的典型芯片，是 Kineits 系列中集成度最高的芯片。K60 系列 MCU 具有 IEEE1588 以太网、USB2.0 OTG 和硬件加密电路。K60DN512 具有丰富的通讯接口、AD 转换电路和外围控制电路。其特性如下：

（1）ARM Cortex- M4 内核，主频高达 100MHZ；
（2）32 路 DMA 供外设和存储器使用，大大提高 CPU 利用率；
（3）10 种低功耗模式，包括运行，等待，停止和断电；
（4）512K Flash 和 128K SRAM；
（5）集成硬件和软件看门狗，硬件加密电路和 CRC 电路；
（6）33 路单路和 4 路差分的 16 位 AD 转换，2 路 12 位 DA 转换；
（7）8 路电机控制，2 路方波解码，4 路可编程定时器；
（8）SD 卡主机控制器，6 路 UART，IIC，IIS ，SPI，CAN；
（9）USB2.0 全速和高速接口，支持 OTG；
（10）IEEE1588 以太网接口，支持 MII 和 RMII 通讯；
（11）工作电压 1.71V~3.6V，工作温度- 40° ~105° ；
（12）多达 100 路 GPIO 引脚，所有 GPIO 引脚兼容 5V 电平。

4.3.1 K60DN512 核心板设计

K60DN512 核心板实现 K60N512 芯片的最小系统电路。芯片的最小系统电

路是能够使芯片运行起来所需要的最小电路，通常包括：电源电路，晶振电路，复位电路，调试接口电路。K60DN512 的各模块功能引脚全部通过扩展接口引出。

4.3.2 电源电路

K60DN512 工作电压为 3.3V ，扩展接口中引入 5V 电压，然后通过线性稳压器件 ASM1117 提供 3.3V 电压。

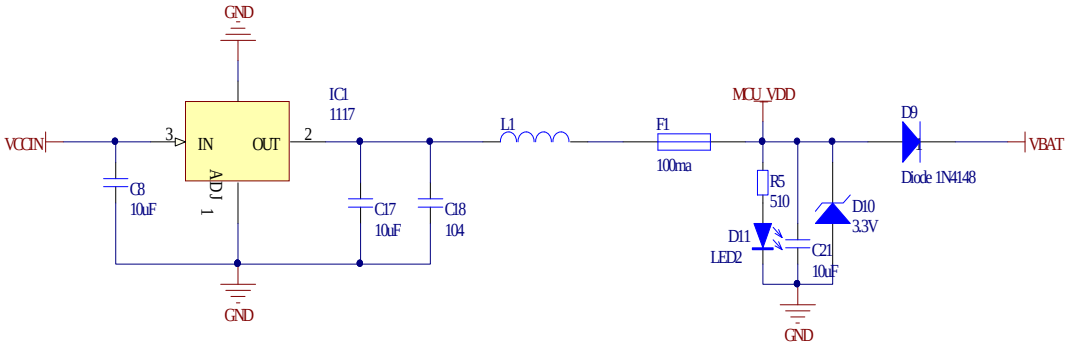

图 4.4 核心板电源

4.3.3 晶振电路

晶振用于产生芯片工作时钟。 K60DN512 内部集成多用途时钟产生器（Multipurpose Clock Generator，MCG）模块，用于将晶振输入时钟倍频至系统所需时钟。K60DN512 共需要两个晶振，一个是芯片的主晶振，用于产生芯片和外设的工作时钟，另一个是实时定时器（Real Timer Counter，RTC）的晶振。本系统芯片时钟使用 50MHz 有源晶振，RTC 时钟使用 32.768KHz 无源晶振。其晶振电路如下：

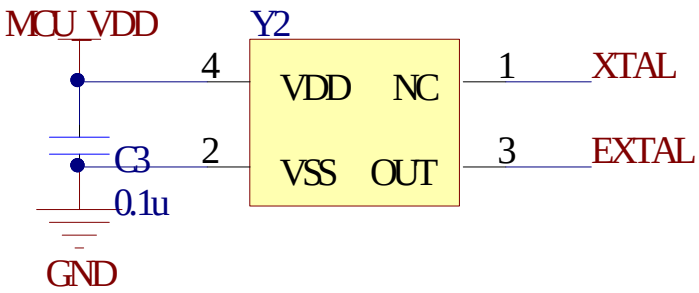

图 4.5 芯片主时钟电路

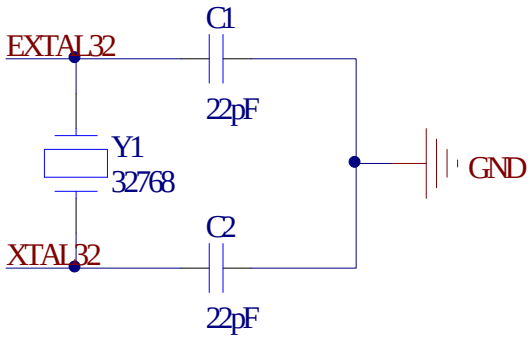

图 4.6 芯片实时时钟电路

4.3.4 JTAG 调试电路

Kinetis 芯片使用的是 ARM Cortex- M4 内核，该 内核内部集成了 JTAG（Joint Test Action Group ）接口，通过 JTAG 接口可以实现程序下载和调试功能。JTAG 调试电路如下图：

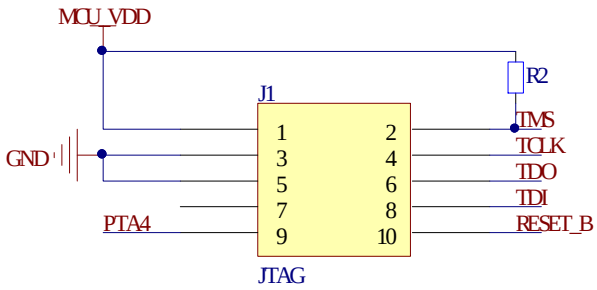

图 4.7 JTAG 调试电路

4.3.5 扩展接口电路

扩展接口将 K60DN512 所有功能引脚引出,是 K60DN512 核心板和 Kinetis 扩展板的接口。扩展接口还负责给核心板提供电源。

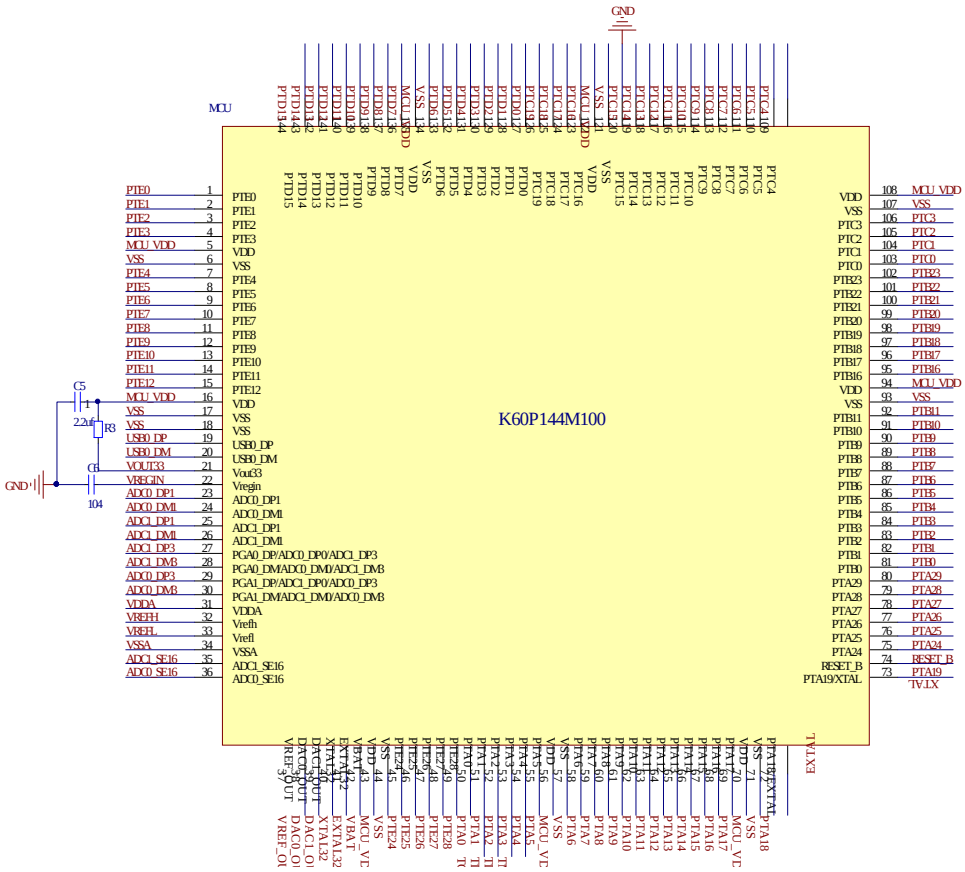

图 4.8 核心板接口扩展

4.4 驱动模块

本智能车系统的驱动电路由 2 片 BTS7960 构成 H 桥。通过控制 4 个 MOS 管的导通和关断来实现正反转，并通过控制输入的 PWM 波的占空比来调节电机两端的平均电压，达到控制电机的转速的目的。驱动芯片采用英飞凌公司的半桥式驱动芯片 BTS7960。其工作电压为 5-36V，驱动电流最大可达 41A，导通电阻典型值为 16mΩ ，输入信号为 TTL/CMOS，PWM 频率高达 25KHz。

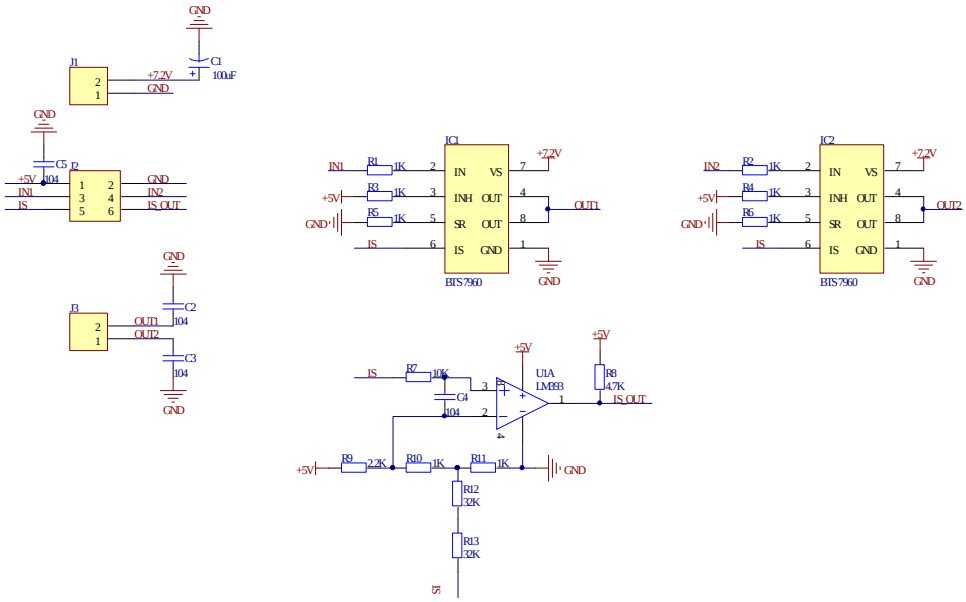
GND
J1
C1
100uF
+7.2V
GND
C5
104
+5V
IN1
IS
J2
GND
IN2
IS_OUT
J3
C2
104
OUT1
OUT2
C3
104
IC1
IN1
R1
1K
+5V
R3
1K
GND
R5
1K
IS
IN
VS
INH
OUT
SR
OUT
IS
GND
+7.2V
OUT1
BTS7960
IC2
IN2
R2
1K
+5V
R4
1K
GND
R6
1K
IS
OUT2
R7
10K
C4
104
U1A
LM393
R8
4.7K
IS_OUT
+5V
R9
2.2K
R10
1K
R11
1K
GND
R12
32K
R13
32K

图 4.9 驱动电路

第五章：图像采集处理模块

5.1 摄像头简介

摄像头分黑白和彩色两种，为达到寻找到黑线的目的，只需要提取画面的灰度信息，而不必提取彩色信息，所以我们使用 COMS 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 COMS 图像传感器 OV7620，总有效像素单元为 664×492 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、第三代网络摄像手机、手机、智能安全系统、汽车倒车雷达、玩具，以及工业、医疗等多种用途。

OV7620 是 1/3CMOS 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，能满足一般图像采集系统的要求。

5.2 黑白阀值提取

阀值选取是获取有效的赛道信息的关键，选择阀值过大，则可能部分白点处理为黑点，同样过小，则可能丢失黑塞边线。

阀值分为两种：动态阀值和静态阀值

动态阀值适应环境能力较强，静态阀值需要根据现场环境来设置阀值参数；

动态阀值处理有两种方法：

(1) 用概率统计获取黑白色差值，分别找到每行黑色白色峰值，做一定比例选取中间某一个值作为阀值。

(2) 比较每行中各点灰度值求取最大值和最小值，根据适当运算求得中间

某个值作为阀值。

静态阀值根据观察直接给出。

在光照多变或者环境条件较为复杂的情况下，动态阀值具有一定的优势，但是在高速行驶处理过程中，大量数据处理无疑降低了运算速度。在 K60 这块芯片中，高频下速度是能够保证的。因此我们采用了动态二值化。

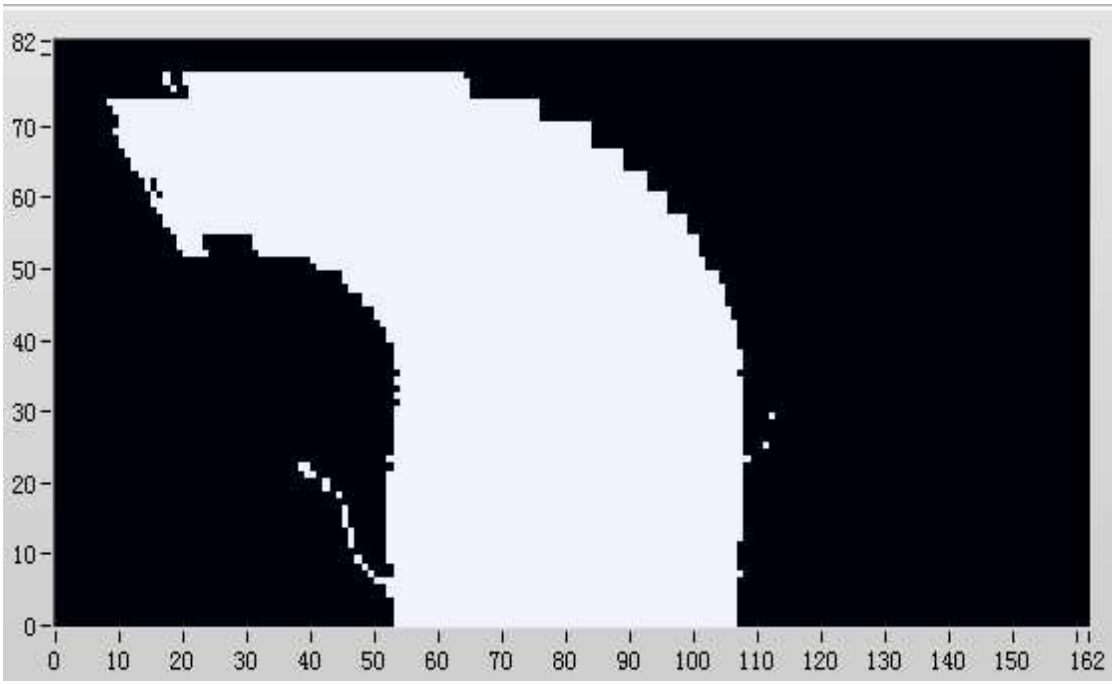

图 5.1 二值化后图像

5.3 图像滤波

由于反光或者杂点的干扰，一般男获取连续的边沿线段信息，因此图像二值化后需要进行滤波处理。

滤波方案有：高斯滤波，中值滤波、领域平均法滤波等

根据我们图像的需要我们采用中值滤波。

5.4 边沿黑线提取

根据第七届“飞思卡尔”杯智能车竞赛规则，赛道两边沿贴有宽度为 2.5CM

的黑色胶带，中间为白色 KT 板，因此，我们采用从赛道中心向两边寻找黑线的方案来寻找边沿黑线。

首先寻找前面十行的黑色变沿线，如果能够找到两边两条边沿，则该场图像有效，否则本场图像无效，跳过处理根据上一次图像来控制。

若该场图像有效，则继续处理，从图像中间分别向两边寻找黑线，如果有一边没有黑线则连续寻找下面三行都么有，则认为该边没有黑线了，同时记录下该行数。剩余行则只寻找另一边了，直到寻找到另一边也没有黑线了，记录下另一个边沿的丢失行。如果图像中两个边沿都同时完全存在，则两个边沿的最终有效行都为图像的行数。两边沿的最终行数确定后通过比较来给弯道标志（左弯道、右弯道、直道）。

5.5 边沿黑线提取校正

实际调试过程中由于环境的不理想，如赛道边沿黑线反光、跑到衔接的缝隙或者磨损等，会导致黑线提取不理想，有一定的杂点，因此要进行对黑线适当的修复以保证路径识别的准确性和稳定性。我们通过对连续相邻几行黑点进行差分处理，如果误差超过设定的值就认为该行黑线提取失败，于是人为赋值为上一行的黑点位置，通过这样的补偿，我们的黑线识别非常的成功，基本上赛车不会因为图像识别错误而跑出赛道。

5.6 十字判断

由于实际赛道十字处，都有小段直道，因此变沿线是相互垂直的，于是，当检测到变沿线上拐点两边黑线近似于相互垂直则认为是十字，或者是前面一些行中两边沿都没有找到黑线，从某一行开始又突然寻找到两条变沿线，则也同样为十字。基于此判定，我们的车十字口通过率为百分之八十。如果不是特别走偏，十字能全部通过。

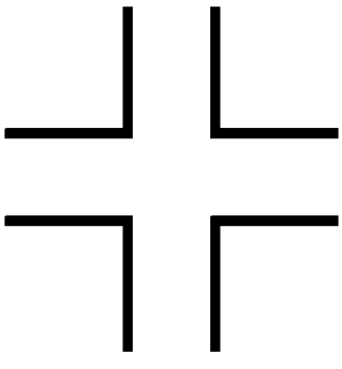

图 5.2 交叉路口

5.7 中心线提取

控制车子转向需要一个赛道信息的反应量，这个量通过赛道的边沿反应出来，而赛道中心线就是这个控制舵机量和赛道信息的中间量。通过前面提取的边沿黑线，经过一定的算法处理就能得到反应实际赛道中心对应的模型一维数组。

(1) 直道或者类似直道：两个边界的平均值

(2) 左转弯：从开始行到左线丢失行，中心值为两个边界平均值；从左线丢失行到右线丢失行，中心值为右边线减去一个固定值。

(3) 右转弯：从开始行到右线丢失行，中心值为两个边界平均值；从右线丢失行到左线丢失行，中心值为左边线减去一个固定值。

(4) 十字：从开始行处理到十字叉口行，中心值为两边沿黑线的平均值。或者从对面叉口行开始到图像最后一行，中心值为两边沿平均值，从开始行到对面叉口行给一特定中心值。

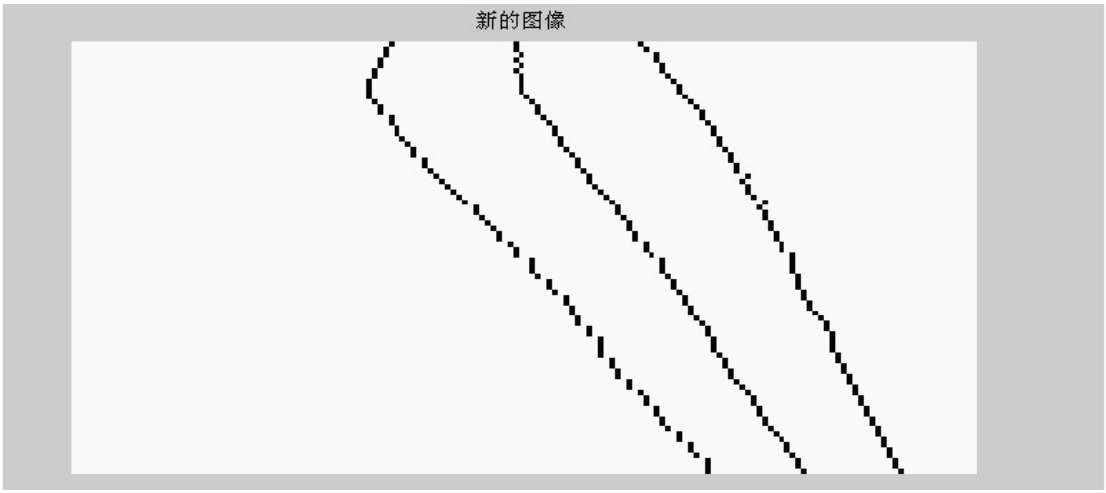

图 5.3 直道类似直道

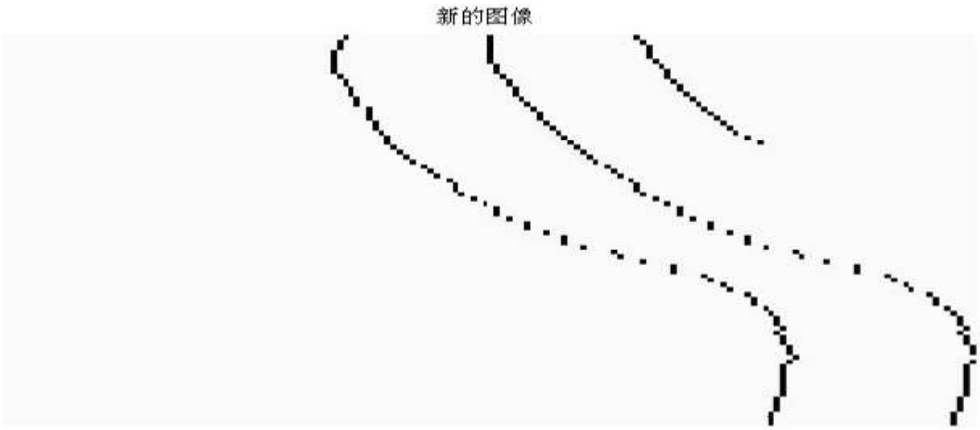

图 5.3 右转弯

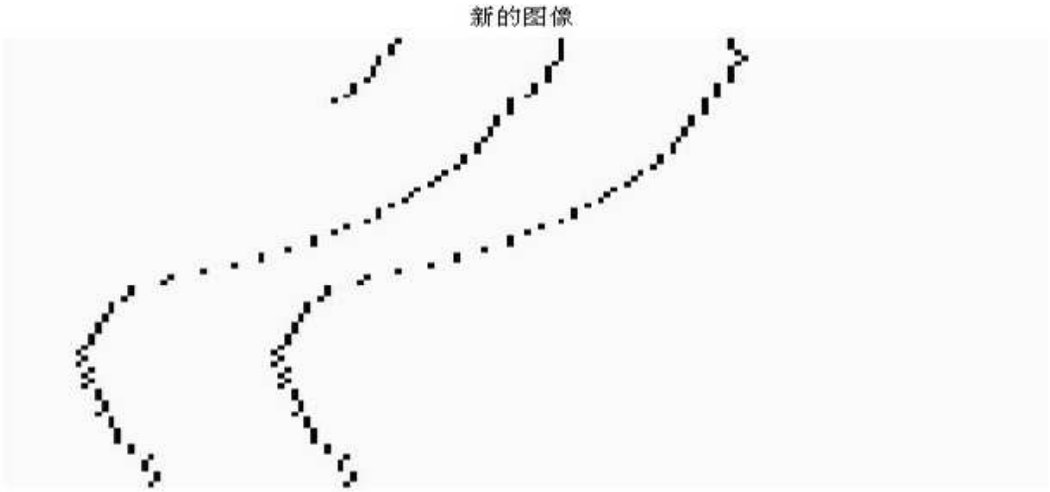

图 5.4 左转弯

第六章：控制策略

控制策略对智能车的速度和路径起决定性作用，它是智能车的控制核心，它直接决定着智能车的极限速度。本智能车的控制策略主要包括路径优化策略、舵机控制策略、电机控制策略。

6.1 路径优化策略

本文的路径优化是指将智能车识别出的路径进行处理变换，以方便智能车行驶，保证智能车能够不冲出跑道并以最快的速度通过各种弯道。

6.1.1 路径优化思想的确定

路径优化以确保智能车能够以走最短的路程通过各种道路。

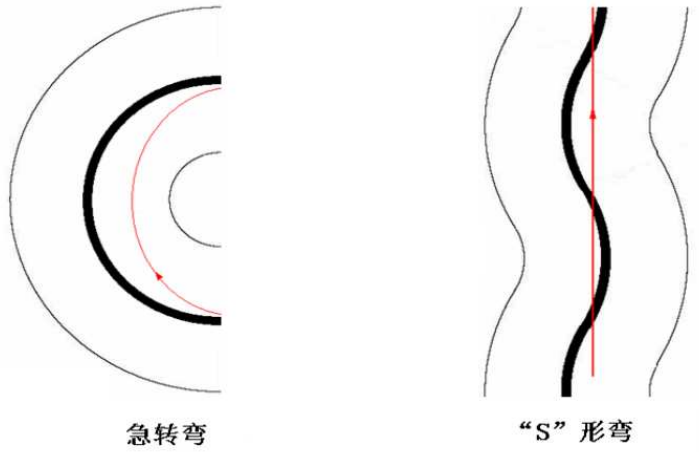

图 6.1 路径优化效果

6.1.2 路径优化的实现

最小二乘法（又称最小平方法）是一种数学优化技术。它通过最小化误差的平方和寻找数据的最佳函数匹配。利用最小二乘法可以简便地求得未知的数据，并使得这些求得的数据与实际数据之间误差的平方和为最小。所以我们采用最小二乘法拟合的方法来实现路径的优化，两点之间最短路程为其直线距离，所以本文采取将路径拟合为直线的方法来取得最短的优化路径。

最小二乘法原理如下：[公式推导过程引用大连理工大学—狩猎者]

最小二乘法就是将一组符合 Y=a+bX 关系的测量数据，用计算的方法求出最佳的 a 和 b。显然，关键是如何求出最佳的 a 和 b。

设直线方程的表达式为：

$$y = a+bx$$

要根据测量数据求出最佳的 a 和 b。对满足线性关系的一组等精度测量数据(xi，yi)，假定自变量 xi 的误差可以忽略，则在同一 xi 下，测量点 yi 和直线上的点 a+bxi 的

偏差 di 如下：

$$d_1 = y_1 - a - bx_1 \qquad \text{公式 1}$$

$$d_2 = y_2 - a - bx_2$$

$$d_3 = y_3 - a - bx_3$$

$$\cdots\cdots$$

$$d_n = y_n - a - bx_n$$

显然最好测量点都在直线上(即 d1=d2=……=dn=0)，求出的 a 和 b 是最理想的，但测量点不可能都在直线上，这样只有考虑 d1、d2、……、dn 为最小，也就是考虑 d1+d2+……+dn 为最小，但因 d1、d2、……、dn 有正有负，加起来可能相互抵消，因此不可取；而|d1|+ |d2|+……+ |dn|又不好解方程，因而不可行。现在采取一种等效方法：当 $d1^2+d2^2\wedge\cdots\cdots+dn^2$ 对 a 和 b 为最小时，d1、d2、……、dn 也为最小。取（ $d1^2+d2^2\wedge\cdots\cdots+dn^2$ ）为最小值，求 a 和 b 的方法叫最小二乘法。

令

$$D = \sum_{i=1}^{n} d_i^2 = \sum_{i=1}^{n} [y_i - a - b_i]^2$$

公式 2

D 对 a 和 b 分别求一阶偏导数为：

$$\frac{\partial D}{\partial a} = -2[\sum_{i=1}^{n} y_i - na - b\sum_{i=1}^{n} x_i]$$

公式 3

$$\frac{\partial D}{\partial b} = -2[\sum_{i=1}^{n} x_i y_i - a\sum_{i=1}^{n} x_i - b\sum_{i=1}^{n} x_i^2]$$

公式 4

再求二阶偏导数为：

$$\frac{\partial^2 D}{\partial a^2} = 2n$$

公式 5

$$\frac{\partial^2 D}{\partial b^2} = 2\sum_{i=1}^{n} x_i^2$$

公式 6

显然：

$$\frac{\partial^2 \mathrm{D}}{\partial \mathrm{a}^2} = 2\mathrm{n} \geq 0$$

公式 7

$$\frac{\partial^2 D}{\partial b^2} = 2\sum_{i=1}^{n} x_i^2 \geq 0$$

公式 8

满足最小值条件，令一阶偏导数为零：

$$\sum_{i=1}^{n} y_i - na - b\sum_{i=1}^{n} x_i = 0$$

公式 9

$$\sum_{i=1}^{n} x_i y_i - a\sum_{i=1}^{n} x_i - b\sum_{i=1}^{n} x_i^2 = 0$$

公式 10

引入平均值：

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

公式 11

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

公式 12

$$\overline{x^2} = \frac{1}{n}\sum_{i=1}^{n} x_i^2$$

公式 13

$$\overline{xy} = \frac{1}{n}\sum_{i=1}^{n} x_i y_i$$

公式 14

则：

$$\bar{y} - a - b\bar{x} = 0$$

公式 15

$$\overline{xy} - a\bar{x} - b\overline{x^2} = 0$$

公式 16

解得：

$$a = \bar{y} - b\bar{x}$$

公式 17

$$b = \frac{\overline{xy} - \bar{x}\bar{y}}{\overline{x^2} - \bar{x}^2}$$

公式 18

将 a、b 值带入线性方程 bx a y + = ，即得到回归直线方程。将计算公式离散化后为：

$$B = \left[\sum (y - \bar{y})(x - \bar{x})\right] / \left[\sum (x - \bar{x})^2\right]$$

公式 19

$$A = \bar{y} - B\bar{x}$$

公式 20

以下是相应程序：

```
for (i=0;i<finalrow;i++){
                    if(i<5)    Sum_center+=BlackCenter[i];
                    SumX=SumX+i;
                    SumX2=SumX2+i*i;
                    SumY=SumY+BlackCenter[i];
                    SumXY =SumXY +i*BlackCenter[i];
                }
Steer_Slope=((int32)100*(finalrow*SumXY - SumX*SumY))/(finalrow*SumX2 - SumX * SumX);
```

6.2 舵机控制策略

舵机的控制决定了小车运动的轨迹，为了尽可能提高车模运行速度，针对不同的路面需要对运动轨迹进行优化。对于急转弯路面，应尽量靠内道行驶，而小“S”弯应该取直通过。为了简化问题，可以将小车的运动规律进行简化：车体近似沿着道路中心线运行时，改变舵机输出转角即可改变小车前轮转向，并改变车体与道路中心线的相对位置。于舵机响应周期长，而积分项会使系统的瞬时响应性能减弱，不适合用在系统滞后的环境下，综合考虑之后，在舵机控制上我们采用了位置式 PD 算法。其简化公式：

steerangle=STEER_KP*steer+ STEER_KD*error_steer

其中，steerangle 是舵机输出量，STEER_KP 舵机 PD 控制的比例参数，STEER_KD 舵机 PD 控制的微分参数。

由于上式控制舵机的方法不能反映出智能车在当前赛道所处位置，所以我们对上式进行了改进，增加一项反映智能车所在赛道当前所处位置变量。由于图像近端和小车前轮轴线之间有一定的距离，所以图像近端不能反映出小车当前所处的位置，所以我们任然采用最小二乘法归算此变量。经过实践证明计算结果和实际距离有一定的误差，但是这个误差在我们能够接收的范围内。

```
Car_near =Sum_center- 3*Slope/100-VDEIO_CENTER;
if(Car_near >30)      Car_near =30;
if(Car_near <-30)    Car_near=30;
steer=Steer_Slope+5*Car_near;
```

其中，Car_near 是拟合出的小车前轮轴线上偏离赛道中心的距离。Sum_center 图像近端 5 行黑线的的平均值。Slope 是通过最小二乘法计算出的整场图像的斜率。VDEIO_CENTER 是视场中心。

6.3 电机控制策略

速度控制策略以跑道识别结果和摄像头有效前瞻作为输入量，输出期望速度值，与底层速度控制相结合控制模型车的加速或刹车。为了达到好的速度控制效果，对速度进行闭环控制是必须的。这里所说的速度控制策略是指设定速度的确定方法——设定速度主要由道路与直道的偏差来决定，道路越接近直道，设定速度越高，反之越低。

系统行进中的最低速度的确定：令系统以较低的速度匀速行使，在保证安全的前提下，逐渐提高匀速行使的速度，直到系统出现不稳定行为，此速度再减去一个安全量，即为所需的最低速度。也就是说，变速行驶的最低速度等于匀速行使的最高速度。

系统行进中的最高速度的确定：在确定最低速度以后，加入变速策略，不断提高最高速度的设定值，直到系统出现不稳定行为，此速度再减去一个安全

量，即为所需的最高速度。

系统行驶过程中难免出现“失去道路”的情况，对此需要采取一定的安全策略，防止系统“盲跑”而导致危险。

6.3.1 PID 控制算法思想

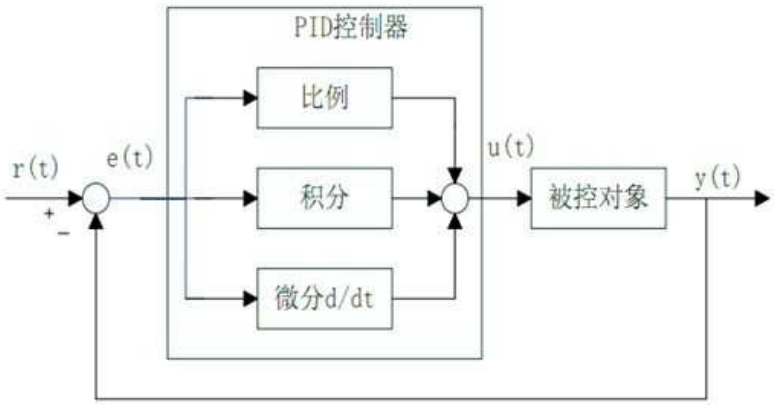

图 6.2 PID 控制器

我们采用增量式 PID，其计算公式如下：

$$\Delta u_n = K_p(e_n - e_{n-1}) + Ki * e_n + K_d(e_n - 2e_{n-1} + e_{n-2})$$

公式 21

三个基本参数 Kp,Ki,Kd 在实际控制中的作用：

比例调节作用（Kp）：是按比例反应系统的偏差，系统一旦出现了偏差，比例调节立即产生作用用以减少偏差。比例作用大，可以加快调节，减少误差，但是过大的比例，使系统的稳定性下降，甚至造成系统的不稳定。

积分调节作用（Ki）是使系统消除稳态误差，提高误差度。因为有误差，积分调节就进行，直至无差，积分调节停止，积分调节输出一常数。积分作用的强弱取决于积分时间常数 Ti,Ti 越小，积分作用就越强。积分作用常与另外两种调节规律结合，组成 PI 调节器或 PID 调节器

微分调节作用（Pd）：微分作用反应系统的误差信号的变化率，具有预见性，能预见偏差变化的趋势，因此能产生超前的控制作用，在偏差还没有形成之前，已被微分调节作用消除。因此，可以改善系统的动态性能。

6.4 电机控制流程图

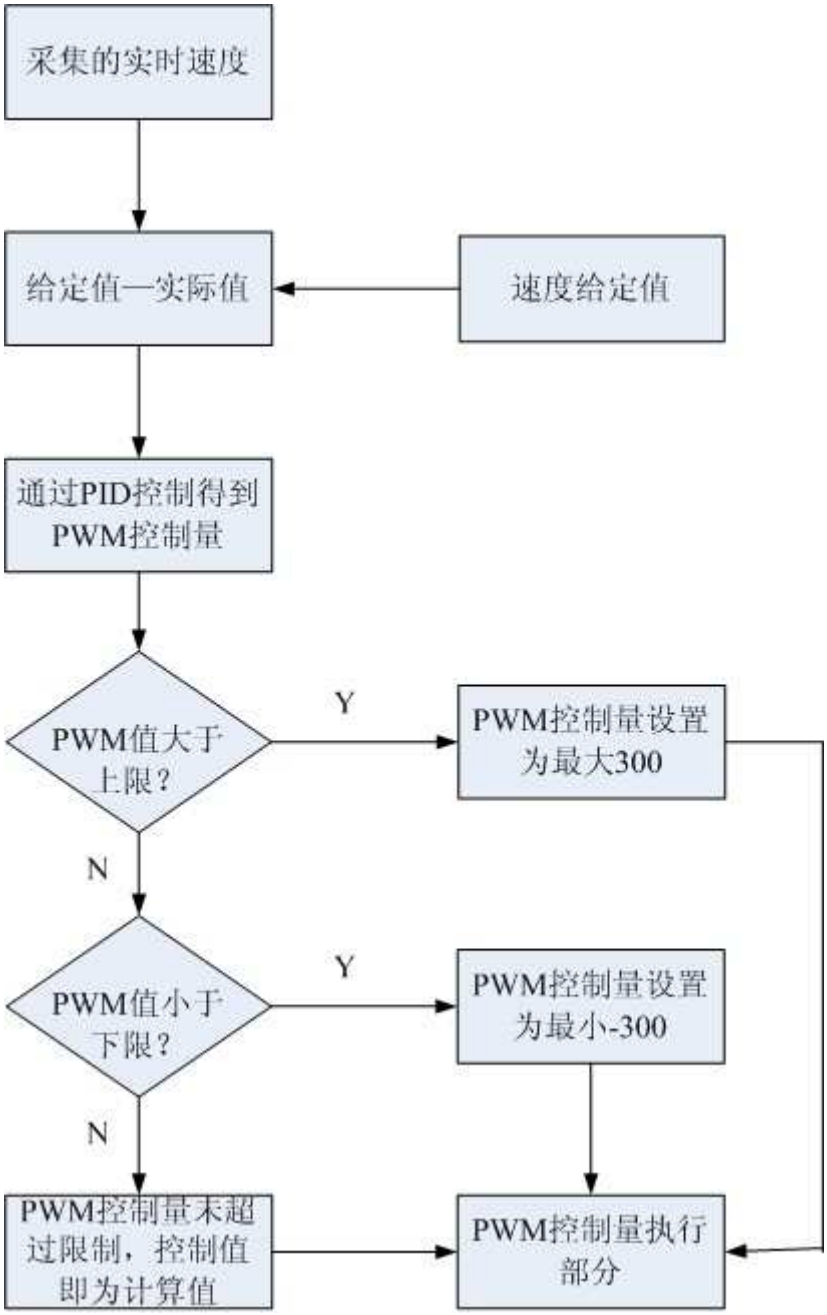

图 6.3 速度控制流程图

6.4.1 部分电机控制程序

```
  if(measure_pulse>(uint8)20)
                {

measure_speed=((int32)measure_pulse*263-(int16)5011)/(int16)1000;
                }else
                {
                    measure_speed=0;
                }

                Err3=Err2;
                Err2=Err1;
                Err1=Speed_Expect- measure_speed;

Speed_PID+=SPEED_KP*(Err1-Err2)+SPEED_KI*Err1+SPEED_KD*(Err1+Err3-
2*Err2);
                 if(Speed_PID>=SPEED_MAX_DTY)
                   Speed_PID=SPEED_MAX_DTY;
                 else if(Speed_PID<=0)Speed_PID=0;
                        MotorDTY=Speed_PID;
               if(measure_speed<5 && MotorDTY >=250)     //保护电机
               {
                  //MotorDTY=0;
                  Motor_Brake(0);
                  Motor_Run(0);
               }else
               {
                    Motor_Brake(0);
                    Motor_Run(MotorDTY);
               }
```

6.5 起跑线识别

根据比赛规则，起跑线只会出现在直道上，所以为了能够准确识别起跑线不至于在赛道弯道中识别错误，因此在程序中判断赛车在直道上行驶时才进行起跑线识别。起跑线的特点是两段横帖着的 10CM 的黑线，当赛车在直道上行驶时，能够看到整幅图像，因此两边沿黑线应该是最大行数的。又因为赛车高速运行中，起跑线采集可能只存在在一两行中，并且有可能不是一行中左右两段线都有，因此检测到一边也算作是检测到起跑线了。首先根据判定的三个条件中前两个直道，有效行数都满足的情况下，从图像的中间向两边寻找，如果某一行的黑点个数大于某一个值，小于某一个值，则认为检测到了起跑线。根据实际调试，能够比较准确的检测到起跑线。

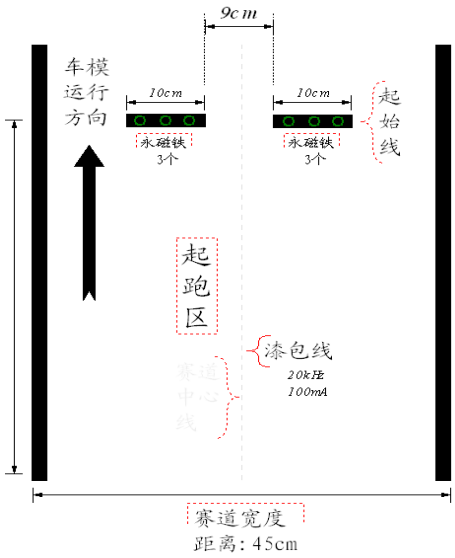

图 6.4 赛道出发区示意图

第七章：系统调试

7.1 编译开发环境

系统的调试开发用到了，IAR6.3 开发软件，另外为了调试方便，我们还用到 MATLAB,LABVIEW 开发了部分上位机，以及用无线调试模块和外部状态指示单元等。

7.2 上位机开发软件

由于我们选用的单片机是 MK60DN512，其内核是 ARM Cortex-M4 Core，可以用 IAR Systems 公司的 IAR6.3 作为其开发软件，也可以用 Metroworks 公司的 Code Warrior10.1 作为其开发软件。经过比较我们选择 IAR Systems 公司的 IAR6.3 开发软件。

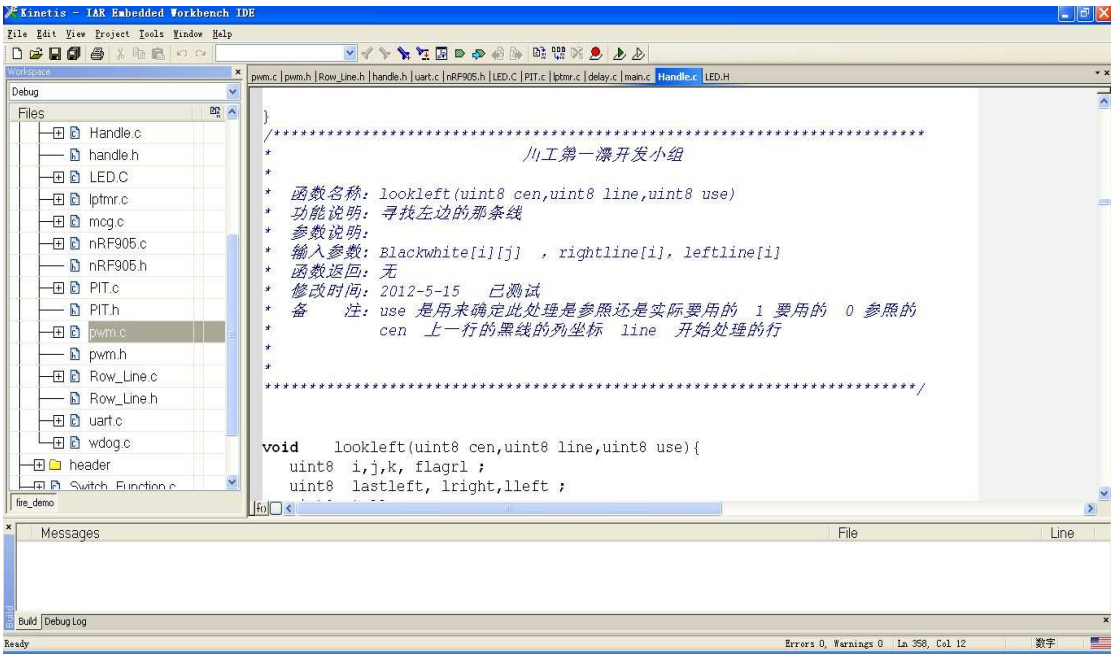

图 7.1 IAR6.3 运行界面

IAR Embedded Workbench 功能非常强大，支持众多知名半导体公司的

微处理器。许多全球著名的公司都在使用 IAR SYSTEMS 提供的开发工具，用以开发他们的前沿产品，从消费电子、工业控制、汽车应用、医疗、航空航天到手机应用系统.

7.3 智能车上位机开发软件

由于单片机的处理能力有限和不便于人的观察。我们专门设计了我们的智能车上位机。效果如下图。

前期用 MATLAB 设计的上位机效果图：

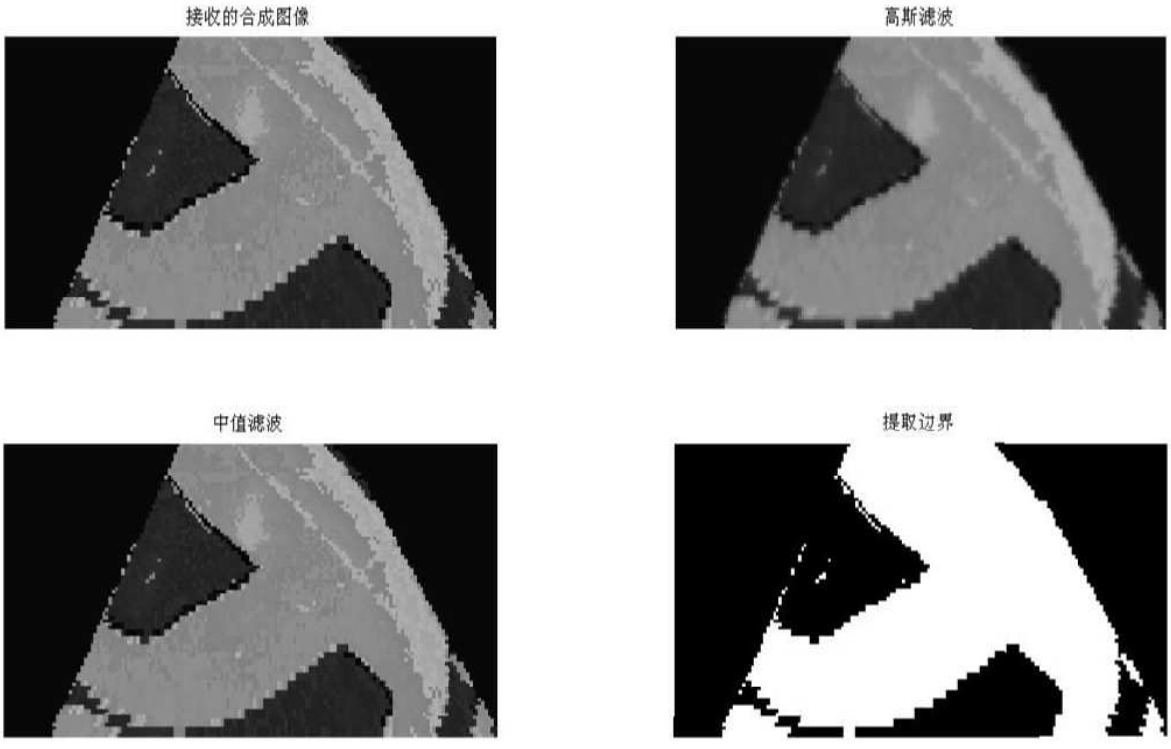

图 7.2 MATLAB 运行效果图

中后期用 LABVIEW 设计的上位机效果图：

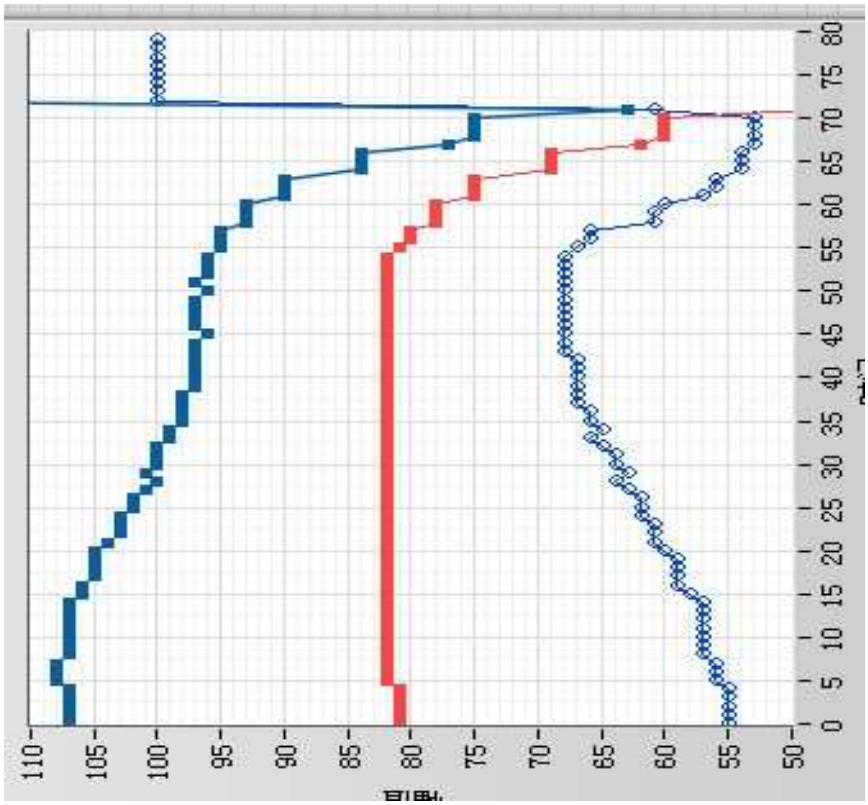

图 7.3 LABVIEW 运行效果图

7.4 智能车无线调试模块

在智能车调试过程中，需要将小车检测到的路面信息以及速度等参数及时的发给 PC，以便针对性的对算法进行优化和分析。由于小车在运行时不能通过有线的方式获取其运行的参数，我们用了无线模块。

无线模块介绍：nrf905,通讯距离可达 300 米，完全满足调试的最远距离。支持串口（UART）与 USB 口透明数据传输，适用于串口设备与计算机之间的数据传输。 使用此模块 K60 单片机可以方便地和 PC 通过 RS232 协议通信.

实物图如下：

图 7.4 nRF905 效果图

7.5 状态指示灯

为了使调试更加方便以及便于观察赛道运动考虑，我们在主控小系统上增加了 8 个 LED 作为智能车指示灯，大大提高了调试效率。

7.6 蜂鸣器指示

当赛车快速行驶时，预设的事件发生时（如检测到十字，急弯，起跑线等），蜂鸣器立即发出声音提示，在调试过程中非常适用。

第八章 总结与展望

8.1 总结

历时几个月，终于到了全国总决赛的时候了，无论最终结果如何，我们都将不会留有遗憾，在这半年多的时间里，我们车队队员，每天早出晚归，甚至在实验室熬通宵做车，讨论算法，其中的酸甜苦辣，坚持与放弃，成功后的兴奋以及失败后的沮丧，只有一起经历过的队友才能理解。这个过程中，我们不仅得到了专业知识的提高，更学会了团队精神与人合作，充分相信队友，也和队友建立了深厚的友谊。

在做车之前，我们只有少之又少的动手能力和做车经验，但是在这个智能车的制作过程中，我们学会了如何去查阅资料，如何去请教前辈，如何学习的方法。更加明白了学海无涯学无止境的道理。在以后人生的道路上，我们遇到困难要具有河里水一样的精神，遇到石头的阻碍，想法设法绕过去处理解决掉问题。

8.2 不足与展望：

虽然我们在西部赛区中取得了我校参加飞思卡尔比赛以来最好的成绩，并顺利进入全国总决赛。可是我们发现，我们与传统强队之间任然存在很大的差距，这种差距不仅仅体现在技术层面上，还有智能车文化底蕴和传承上。在全国赛中，还有很多我们值得学习和的地方。

其次，在智能小车的简洁化和轻量化方面还需要进一步改进，在规则的理解和把握上，还应该更大胆尝试些改装。

总之，我们抱着“不断学习不断超越自己”的态度一步一步走下去。

第九章 致谢

经过半年多的努力，我们终于完成了智能车的制作，在此，对帮助过我们的老师和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。感谢我们指导老师董秀成教授和古世甫老师，是他们在后面对我们的支持，在参加比赛的过程中，为我们安排好一切，对我们关爱有加，感谢梁海师哥对我们的引导和鼓励以及技术的支持。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。感谢西华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。

参考文献

1.孙鑫，余安萍.VC++深入详解[M].北京：电子工业出版社，2006.6

2.唐建文.智能小车控制系统的设计与实现[D].广东：广东工业大学硕士论文，2008

3.李正军.计算机控制系统[M].北京：机械工业出版社，2005.1

4.卓晴，黄开胜，邵贝贝.学做智能车——挑战“飞思卡尔”杯[M].北京：北京航天大学出版社，2007,3:35~40

5.宋敏，郤新凯，郑亚茹.CCD 与 CMOS 图像传感器探测性能比较.半导体光电，

2005,26(1):5~9

6.孙涵，任明武，唐振民等.基于机器视觉的智能车辆导航综述[J].公路交通科技，2005,22(5):132~135

7. 黄开胜，金华民，蒋狄南.韩国智能模型车技术方案分析[J].电子产品世界，

2006,3:150~152

8. 易大义 ，“ 计算方法 [M] ” ，浙江大学出版社， 1995

9. 谭浩强 ，“ C 程序设计（第二版） ” ，清华大学出版社 ，1999.12

10.余永权 ，“ 单片机在控制系统中的应用 ” ，电子工业出版社， 2003.10

11.电子科技大学海盗旗技术报告。

12.大连理工大学狩猎者技术报告

附件

A 智能车主控板原理图：

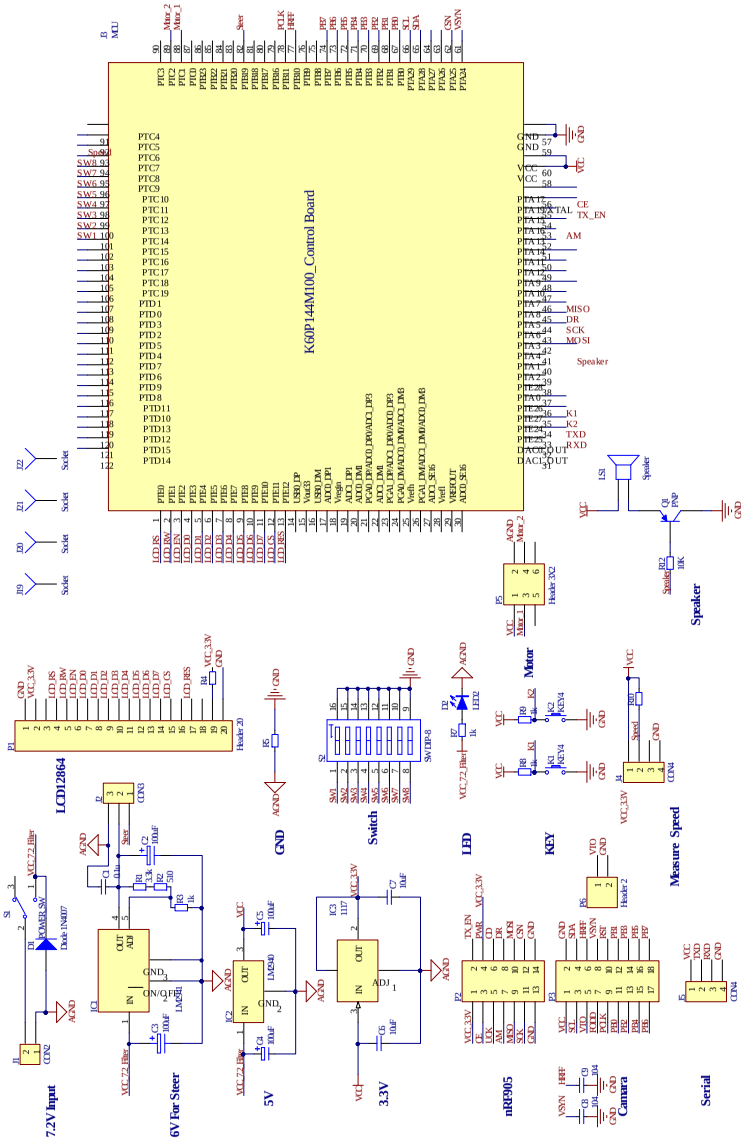

图 A-1 主控电路

B 智能车小系统原理图：

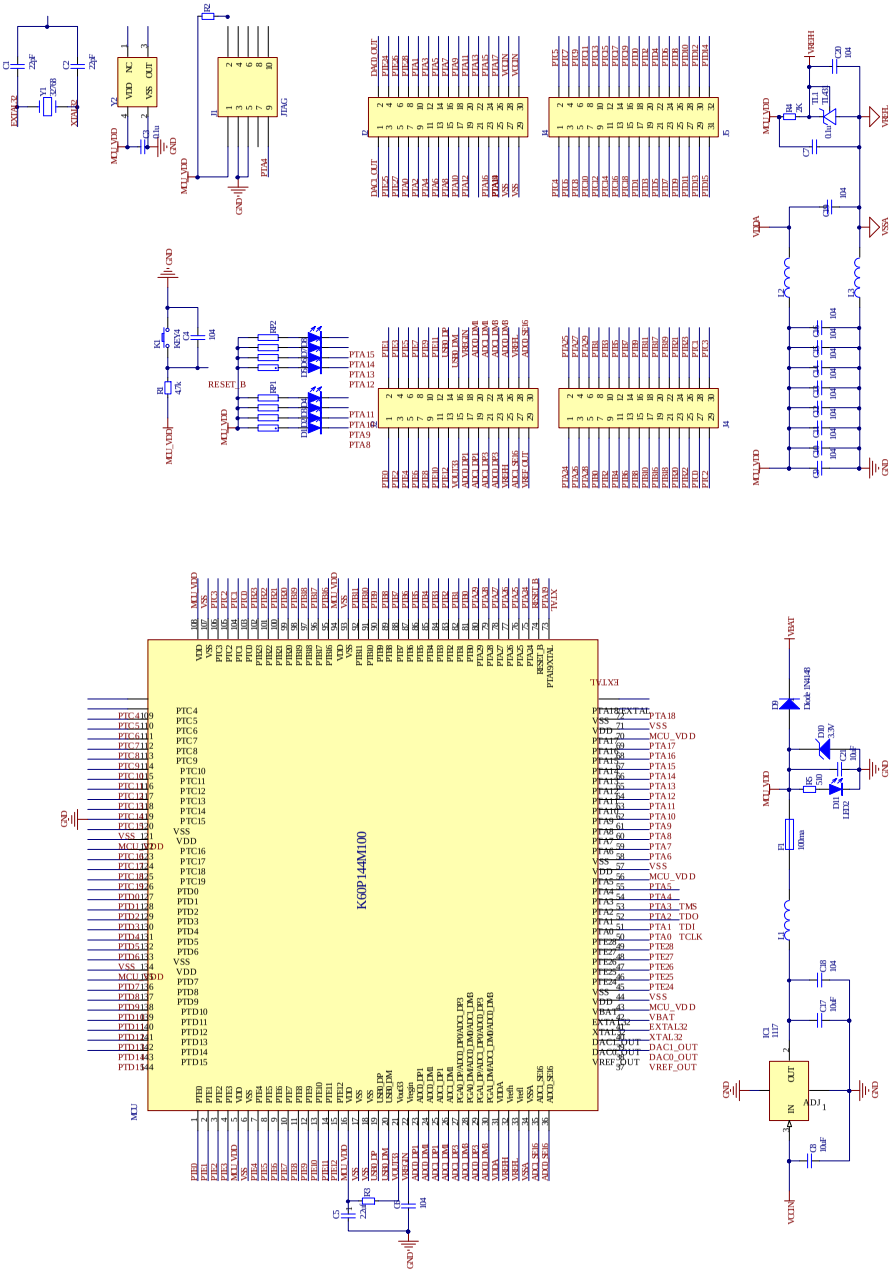

图 B-1 系统板电路

C 智能车小系统 PCB：

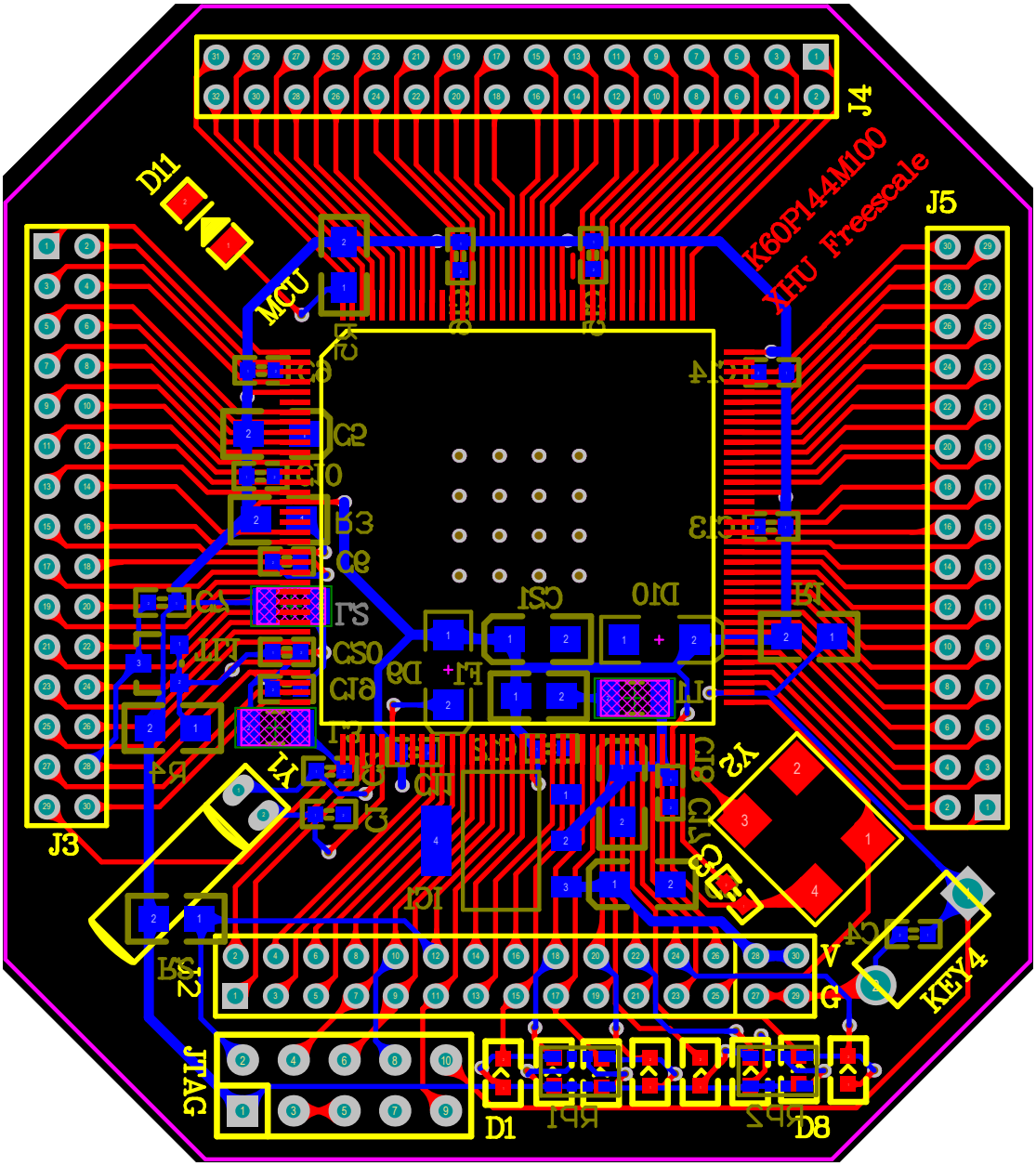

图 C-1 系统板 PCB

D 部分代码程序：

```
/*****************************************************************
************
* 函数名称：      DMA0_Init
* 功能说明：      DMA 初始化服务程序      每次有 PCLK 来触发一次 DMA 采集
中断
* 函数输入：      None
* 函数输出：      None
* 返 回 值：      None
* 标      签:
*****************************************************************
***********/
void DMA0_Init(void)
{
    SIM_SCGC6|=SIM_SCGC6_DMAMUX_MASK; //打开 DMA 多路复用器时
钟
    SIM_SCGC7|=SIM_SCGC7_DMA_MASK;        //打开 DMA 模块时钟
    DMAMUX_CHCFG0=DMAMUX_CHCFG_SOURCE(50);
//DMA 通道 0 对应 50 号 DMA 请求，即 PORTB
    DMA_TCD0_CITER_ELINKNO=DMA_CITER_ELINKNO_CITER(V);//  当
前主循环次数,采集点数
    DMA_TCD0_BITER_ELINKNO=DMA_BITER_ELINKNO_BITER(V);//  起
始主循环次数，采集点数
    DMA_TCD0_SADDR=(uint32)&GPIOB_PDIR;        //设置源地址 GPIO 口，
PORTB
    DMA_TCD0_SOFF=0;                                    //每次传送源地址不变
    //DMA_TCD1_NBYTES_MLOFFYES=DMA_NBYTES_MLOFFYES_NBYT
ES(1)+DMA_NBYTES_MLOFFNO_SMLOE_MASK+DMA_NBYTES_MLOFFY
ES_MLOFF(-4);//传送 4 字节
    DMA_TCD0_NBYTES_MLNO=DMA_NBYTES_MLNO_NBYTES(1);      //
每次读取一字节
    DMA_TCD0_SLAST=0;                          //主循环结束后源地址 0 回写
tcd
    DMA_TCD0_DLASTSGA=0;                  //主循环结束后目的地址 0 回写
tcd
```

```
    DMA_TCD0_DADDR=(uint32)video; //设置目的地址，video 数组第一个元素
    DMA_TCD0_DOFF=1;                                    //每次写目的地址加1
    DMA_TCD0_ATTR=DMA_ATTR_SSIZE(0)+DMA_ATTR_DSIZE(0);// 源 数据宽度 8bit，目的数据宽度 8bit
    DMA_TCD0_CSR=DMA_CSR_DREQ_MASK;          //DMA 通道 0 主循环结束后停止硬件请求
    DMA_TCD0_CSR|=DMA_CSR_INTMAJOR_MASK;        //使能 DMA0 中断
    DMAMUX_CHCFG0|=DMAMUX_CHCFG_ENBL_MASK;
//DMA 通道 0 使能

}

/*************************************************************************
************
* 函数名称：      DMA_CHO_ISR
* 功能说明：    DMA 中断服务程序        达到主循环设定次数后触发该中断表示结束
* 函数输入：      None
* 函数输出：      None
* 返 回 值：      None
* 标      签:
*************************************************************************
***********/
void DMA_CHO_ISR(void)
{
        DMA_INT|=DMA_INT_INT0_MASK;   //清除通道 0 中断

}

/*************************************************************************
************
* 函数名称：      porta_isr
* 功能说明：      场中断服务程序      场中断，A24，下降沿中断
* 函数输入：      None
* 函数输出：      None
* 返 回 值：      None
* 标      签:
```

```
**********************************************************************
**********/
void porta_isr(void)
{
        PORTA_PCR24|=PORT_PCR_ISF_MASK;   //清除中断标志
        DMA0_Init();                      //初始化 DMA 主要是初始化地址的
        row=0;                            //初始化行
        imagerow=0;                       //初始化采集行 类似于放在数组中
的
        disable_irq(87);                  //关闭 A 口场中断 ，24A 场中断
        PORTB_PCR10|=PORT_PCR_ISF_MASK;
        enable_irq(88);                   //使能 B 口行中断 ，B10 行中断
        DMA_INT|=DMA_INT_INT0_MASK;       //清除通道 0 中断
        enable_irq(0);                    //使能 DMA 通道 0 完成中断
}


/**********************************************************************
**********
* 函数名称：      portb_isr
* 功能说明：      行中断服务程序      行中断，B10，上升沿中断
* 函数输入：      None
* 函数输出：      None
* 返 回 值：      None
* 标      签:
**********************************************************************
**********/
void portb_isr(void)
{
        PORTB_PCR10|=PORT_PCR_ISF_MASK;   //清除中断标志位
        row++;                                  //行计数
        if(row==data_table[imagerow])     //如果当前行数据应该采集
        {
            DMA_TCD0_DADDR=(uint32)&video[imagerow][0];
            imagerow++;
            DMA_ERQ|=DMA_ERQ_ERQ0_MASK;
        }
      else if(row>=170)//ENDROW)              //一场完成，关闭行中断
         {
              disable_irq(88);                //一场完成，关闭行中断
```

```
                disable_irq(0);
                send_flag=1;                    //发送标志位置一
            }
}


void main(void)
{
    uint16 i,j,k,flagdelaystop,time, runtime,stoptime;
    uint8 flagstopstop=0,stopstoptime=0;

    flagstartline=0;time=0;stoptime=0;runtime=0;flagdelaystop=0;//常量初始化

    pll_init(PLL100);               //设置总线频率
    IO_Init();                      //初始化 IO 方向
    LED_init();                     //初始化小灯 IO 方向
    PIT_INIT_MS(0,10);
    LPTMR_INIT();                   //测速初始化
    Steer_Init();                   //舵机初始化
    Motor_Init();                   //电机初始化
    uart_init(UART4,115200);        //初始化串口
    Strategy_Select();
    Set_StopTime();
     enable_irq  (87);              //使能 A 口中断，A24 场中断准备采集图像
   for(;;)
   {
            if(send_flag) {         //这样写可以采集一副图像处理多次
              send_flag=0;          //图像采集完成标志清 0
               if(flagstart==0){
                  runtime++;
                }

                if(flagdelaystop==0){
                    if(flagstart==1){   // 停车起跑线检测延后时间
                        time++;
                        if(time>=Stop_Time)   {//
                          time=500;
                          flagdelaystop=1;
                        }
```

```
                }
            }

            if(runtime==70) {     //发车静止的两秒时间后直接开环加
速
                Motor_Run(150);
                Motor_Brake(0);
            }
           else  if(runtime>=80){  //开始一段时间直接开环加速后开始测
速
                enable_irq(68);         //开周期中断  开电机测速
                runtime=0;
                flagstart=1;
            }
            if(flagstopstop==1){
                    stopstoptime++;
                    if(stopstoptime==5){
                            flag_brakestopstop=1;   //刹车标志
                            Speed_Expect=10;
                    }
                    else if(stopstoptime>=6){
                            disable_irq(68);
                            Motor_Run(0);
                            Motor_Brake(0);
                    }
            }

            //主函数处理过程
             erzhihua();              //静态二值化
             Fandhandcenter();
             findtwoline();
             Cross();
             findcentercenter( );
             Calculat_Slope();

            //车子控制处理完成

if((Slope<=45)&&(Slope>=(-45))&&(finalrow>=73)&&(crossroad!=1))
             {
                  if((flagstartline==0)&&(time==500))
```

```
                {
                 Startline();
                }
            }


            if(flagstartline==1){ //关闭电机
              CONTROL_ROW=2;
              flagstopstop=1;

            }

            PORTA_PCR24|=PORT_PCR_ISF_MASK;
            enable_irq (87);

        }
    }

/************************************************
**函数(模块)名称:   erzhihua(void)
**功能:              定阀值    图像做二值化   寻找摄像头采集的实际图像左右边界
**输入参数:         recove_image[i][j]
**输出参数:         Blackwhite[i][j]   ，rightline[i]，leftline[i]
**函数返回值说明:   无
**使用的资源
**其它说明:

***优化建议：只循环一次，第 0 行单独处理，从第一行开始循环，在循环中加入 左右 rightline
***             leftline  处理  二值化处理   少了两次循环

************************************************/

void      erzhihua(void)
{
    uint8 i,j,k,flagzeroright1,flagzeroleft1,flagzeroleft2,flagzeroright2;
    uint8   rightlineflag,leftlineflag;
```

```
    THRSHVALUE=180;          //第 0 行的阀值

    for(j=1;j<image_line;j++){
          if(recove_image[0][j]==0){
               recove_image[0][j]=recove_image[0][j-1];
         }


     Blackwhite[0][j]=(recove_image[0][j]>THRSHVALUE)?WHITE:BLACK;
          }

    blr[0]=BLRINE; brl[0]=BRLINE;

    THRSHVALUE=140;       //第 1 行之后的的阀值
    for(i=1;i<image_row;i++){
        for(j=1;j<image_line;j++){
           if(recove_image[i][j]==0){
               recove_image[i][j]=recove_image[i][j-1];
            }


   Blackwhite[i][j]=(recove_image[i][j]>THRSHVALUE)?WHITE:BLACK;

        }
        blr[i]=BLRINE;
        brl[i]=BRLINE;
        BlackCenter[i]=5;

    }

 }


/************************************************
**函数(模块)名称:   Fandhandcenter(void)
**功能:                第一行做简单的去噪点处理，寻找处理的第 0 行图像处理的中心
** 输 入 参 数 :                  Blackwhite[i][j]    ，  rightline[i] ，  leftline[i]

**输出参数:           zerocenterpoint
```

```
**函数返回值说明:   无
**使用的资源
**其它说明:              第一行去噪处理可以找准处理的中心
*************************************************/

void      Fandhandcenter(void){
    uint8 i,j,flagzeroright1,flagzeroleft1,flagzeroleft2,flagzeroright2;

    uint8       zerowhite;
    uint8       leftzero1,leftzero2;
    uint8       rightzero1,rightzero2;
    uint8       teshupoint1,teshupoint2;
    uint8       zeroline;

    zerowhite=0;leftzero1=0;rightzero1=0;leftzero2=0;rightzero2=0;
    flagzeroright1=0;flagzeroleft1=0;flagzeroleft2=0;flagzeroright2=0;
    teshupoint2=0;teshupoint1=0;zerotwoteshupoint=0;

    zerocenterpoint=80;   //这个对后面寻找黑线产生了致命的影响
    zeroline=0;
    crossroad=0;

   //这一小段是对第一行的所有点做一个滤波处理，去掉单独的一个黑点的情
况
       for(j=rightline[0]+1;j<(leftline[0]-3);j++){

if((Blackwhite[0][j]==5)&&((Blackwhite[0][j-1]==5)||(Blackwhite[0][j+1]==5))){
                Blackwhite[0][j]=5;
          }
          else
if((Blackwhite[0][j]==5)&&((Blackwhite[1][j-1]==5)||(Blackwhite[1][j]==5)||(Black
white[1][j+1]==5))){
                Blackwhite[0][j]=5;
          }
          else
if((Blackwhite[0][j]==5)&&((Blackwhite[0][j-2]==5)||(Blackwhite[0][j+2]==5))){
                Blackwhite[0][j]=5;
          }
          else   {
                Blackwhite[0][j]=250;
```

```
            }
        }    // end for(j=rightline[0]+1;

    //寻找第一行的参考处理中心
    for(j=rightline[0]+2;j<leftline[0];j++){                    //能够寻找右边的了

if((Blackwhite[0][rightline[0]+2]==250)&&(Blackwhite[0][rightline[0]+3]==250)){
                rightzero1=rightline[0];

            if((Blackwhite[0][j]==250)&&(Blackwhite[0][j+1]==250)){
                    zerowhite++;
            }else {
                    zerowhite=0;

                    if(flagzeroleft1==0){
                        flagzeroleft1=1;
                        leftzero1=j;
                        zeroline=1;
                    }
            }
        }

    }   //for   j   结束


    //下面是寻找第一行的开始处理的中心
    for(j=leftline[0]-2;j>rightline[0];j--){


if((Blackwhite[0][leftline[0]-2]==250)&&(Blackwhite[0][leftline[0]-3]==250)){
            leftzero2= leftline[0];

            if((Blackwhite[0][j]==250)&&(Blackwhite[0][j-1]==250)){
                    zerowhite++;
            }else {
                    zerowhite=0;

                    if(flagzeroright2==0){
                        flagzeroright2=1;
                        rightzero2=j;          //这儿的 J 必须为列数  不然要错的
```

```
                    zeroline=1;
                }
            }
        }

    }  //for  j  结束


    if(zeroline==1){


if((rightzero2>78)&&(leftzero2==leftline[0])&&((leftzero2-rightzero2)>5) ){
                teshupoint2=2;
                zerocenterpoint=(leftzero2+rightzero2)/2;
            }
            if((leftzero1<82)&&(leftzero1>0)&&((leftzero1-rightzero1)>5) ){
                teshupoint1=1;
                zerocenterpoint=(leftzero1+rightzero1)/2;
            }
            if((teshupoint1==1)&&(teshupoint2==2)){

              zerotwoteshupoint=1;

              if((leftzero2-rightzero2)>(leftzero1-rightzero1 )){     //7.4 修改
                  zerocenterpoint=(leftzero2+rightzero2)/2;
              }else{
                  zerocenterpoint=(leftzero1+rightzero1)/2;
              }

            }
    } //end if(zeroline==1){

    zerocenterpointlast=zerocenterpoint;
    zerotwoteshupointlast=zerotwoteshupoint;
}

/*****************************************************************
******
*                                  川工第一漂开发小组
*
```

```
*   函数名称：lookleft(uint8 cen,uint8 line,uint8 use)
*   功能说明：寻找左边的那条线
*   参数说明：
*   输入参数: Blackwhite[i][j]  ，rightline[i]，leftline[i]
*   函数返回：无
*   修改时间：2012-5-15     已测试
*   备       注：use  是用来确定此处理是参照还是实际要用的   1  要用的   0  参照的
*                cen   上一行的黑线的列坐标   line   开始处理的行
*
*
**********************************************************************
*****/


void       lookleft(uint8 cen,uint8 line,uint8 use){
    uint8   i,j,k, flagrl ;
    uint8   lastleft, lright,lleft ;
    uint8   trll;
    uint8   t_offset;
        lastleft=cen;
        trll=0;
        t_offset=7;

    for(i=line;i<76;i++){

                flagrl=0;

                lleft=lastleft+t_offset;
                if(lleft>image_line) lleft=image_line;   //右边不能超过数组的边界了

                if(t_offset>lastleft){
                   lright=t_offset-lastleft;
                }
                else{
                   lright=lastleft-t_offset;
                }
                   if(lright<=rightline[i])   lright=rightline[i]+1;
                if(use==0){
                   lleft=leftline[i]+2;
```

```
                if(lright<=brl[i]){     //前提条件是先找到了右边边界
                    lright=brl[i]+2;
                }

            }


             for(j=lright;j<lleft;j++){
        //          for(j=lright;j<(leftline[i]+2);j++){
                if(j>6){

                    k=j-5;

                    if(k<=rightline[i])    k=rightline[i]+1;      //@@@可能这儿刚好是一个弊端的
                    if(use==0){
                       if(k<=brl[i]){
                           k=brl[i]+2;
                       }
                    }
                    if(Blackwhite[i][k]==5)  {          //单线寻找的时候这个很重要啊  不然就一直找到底去了

                       if(use==1){
                           leftrow=i+1;
                       }
                        blr[i]=k;
                        i=78;
                        flagrl=1;
                        j=image_line;
                    }
                }

                    if(flagrl==0){
                            if(Blackwhite[i][j]==5 ){
                                    blr[i]=j;
                                    flagrl=1;
                                    j=image_line;
                            }
                    }//flagrl  结束
```

```
}//for   j 结束

if(flagrl==0)
{
      blr[i]=blr[i-1];
      trll++;              //正常情况下是 1
      if(i<image_row-3){
            for(j=lright;j<lleft;j++){
        //      for(j=lright;j<(leftline[i+1]+2);j++){
                if(flagrl==0){
                      if(Blackwhite[i+1][j]==5 ){
                            blr[i+1]=j;
                            flagrl=1;
                            j=image_line;
                      }
                }//flagrl 结束
              }//for   j 结束

              if(flagrl==0){
                brl[i+1]=brl[i];
                trll++;       //正常情况下是 2
                for(j=lright;j<lleft;j++){
            //      for(j=lright;j<(leftline[i+2]+2);j++){
                  if(flagrl==0){
                        if(Blackwhite[i+2][j]==5 ){
                              blr[i+2]=j;
                              flagrl=1;
                              j=image_line;
                        }
                  }//flagrl 结束
                }//for   j 结束

                if(flagrl==0){
                      trll++;
                      if(use==1){
                        flagimage=2;      //左边出去了
                        leftrow=i;      //左边的有效行的最后一行
                      }
                      i=78;
```

```
                        }   //结束 if(flagrl==0) 第三次
                    }//结束   if(flagrl==0) 第二次
                }     //if(i<image_row-3)
            }   //结束   if(flagrl==0) 第一次
        lastleft=blr[i];

        if(use==1){
          if(i==75){
              leftrow=75;
          }

          //7.5 修改
          if(leftrow==0){
            if((blr[i]==leftline[i])&&(blr[i-1]<leftline[i-1])){
                  leftrow=i;
                  use=0;
              }
          }
        }

    }//结束 for   i
}

/*************************************************************************
******
*                                  川工第一漂开发小组
*
*   函数名称：lookright(uint8 cen,uint8 line,uint8 use)
*   功能说明：寻找右边的那条黑线
*   参数说明：
*   输入参数: Blackwhite[i][j]   ，rightline[i]，leftline[i]
*   函数返回：无
*   修改时间：2012-5-15   已测试
*   备      注：use 是用来确定此处理是参照还是实际要用的   1 要用的   0 参
照的
*               cen   上一行的黑线的列坐标   line   开始处理的行
*
*************************************************************************
*****/
 //寻找右边的那一条黑线
```

```
void       lookright(uint8 cen,uint8 line,uint8 use){
    uint8 i,j,k, flaglr ;
    uint8 lastright, tright,tleft ;
    uint8 tlrr ;
    uint8 t_offset;

        t_offset=7;
        lastright=cen;
        tlrr=0;
     for(i=line;i<76;i++){       //从传过来的当前行开始寻找


        flaglr=0;


          //寻找右边界

                if(lastright>=t_offset)      tright=lastright-t_offset;      //左边巡线的边
界
                else
                      tright=0;
                tleft=lastright+t_offset;
if(tleft>=(leftline[i]))    tleft=leftline[i]-1;        //跑到右边边界了
                if(use==0){
                   tright=1;
                   if(tleft>=blr[i]){
                        tleft=blr[i]-2;
                   }
                }
                for(j=tleft;j>tright;j--){    //从赛道最里面向最左边寻找
       //              for(j=tleft;j>1;j--){
                   if(j<image_line-6){
                      k=j+5;
                      if(k>=leftline[i])    k=leftline[i]-1;
                      if(use==0){
                         if(k>=blr[i]){
                              k=blr[i]-2;
                         }
                      }
                      if(Blackwhite[i][k]==5)  {          //单线寻找的时候这个很重要
```

啊 不然就一直找到底去了

```
                if(use==1){
                    rightrow=i+1;
                }
                brl[i]=k;
                flaglr=1;
                i=image_row;        //草你妹 的 对后面有影响的!

            }
        }

        if(flaglr==0){
            if(Blackwhite[i][j]==5 ){
                    flaglr=1;
                    brl[i]=j;
                    j=tright;     //为了快点跳出来的
            }
        }
    }//for   j 结束                    这个是寻找当前行的右边黑线

    if(flaglr==0)                          {
        brl[i]=cen;
        tlrr++;
        if(i<image_row-3){
            for(j=tleft;j>tright;j--){
        //      for(j=tleft;j>1;j--){
                if(flaglr==0){
                        if(Blackwhite[i+1][j]==5 ){

                                flaglr=1;
                                brl[i+1]=j;
                                j=tright;     //为了快点跳出来的
                        }
                }
            }//for   j 结束                          又没有找到黑线

            if(flaglr==0)                                  {
                    brl[i+1]=cen;
                    tlrr++;           //正常情况下应该是 2 了
                        for(j=tleft;j>tright;j--){
```

```
//          for(j=tleft;j>1;j--){
              if(flaglr==0){
                  if(Blackwhite[i+2][j]==5 ){

                          flaglr=1;
                          brl[i+2]=j;
                          j=tright;
                      }
                }
            }//for   j  结束
          if(flaglr==0)          //第三次
          {
              tlrr++;

              if(use==1){
                  flagimage=1;       //右边出去了
                  rightrow=i;       //右边的有效行的最后一行
              }
              i=76;

          }

        } //  结束   if(flaglr==0)     第二次
      }//  结束   if(i<image_row-3)
    } //   结束   if(flaglr==0)   第一次

lastright=brl[i];
if(use==1){
    if(i==75){
      rightrow=75;
    }

    //7.5 日修改
    if(rightrow==0){
      if((brl[i]==rightline[i])&&(brl[i-1]>rightline[i-1])){
          rightrow=i;
          use=0;
      }
    }
}
```

```
    }      //for   i   结束
}


/*************************************************************************
******
*                                  川工第一漂开发小组
*
*   函数名称：findtwoline(void)
*   功能说明：寻找图像的两条黑线 ，并判断左右偏转
*   参数说明：
*   输入参数: Blackwhite[i][j] ，rightline[i]，leftline[i],zerocenterpoint
*   函数返回：flagimage,finalrow,brl[i],blr[i],leftrow,rightrow
*   修改时间：2012-7-1     已测试
*   备      注：    flagimage=1；  右转                  brl[i]       右边线
*                  flagimage=2；  左转                  blr[i]       左边线
*                  flagimage=3；  右转                  leftrow      左边有效线的
最后一行
*                  flagimage=4；  左转                  rightrow     右边有效线的
最后一行
*                  flagimage=5 ；  类似直到或者小 S     finalrow     整张图像的
有效线最后一行
*
*************************************************************************
*****/
void      findtwoline(void){

    uint8 i,j,flagrl,leftzero,flaglr,centerlast;
    uint8 lastleft,lastright,tright,tleft,rright,rleft;
    uint8 tlr,trl,zeroleft,flagstartleft,flagstartright;

    uint8       t_offset;
    uint8       zerosingle;
    uint8       rightside,leftside;

    flagimage=0;t_offset=5;tlr=0;trl=0;rightside=0;leftside=0;
    leftrow=0;rightrow=0;zerosingle=0;firstfactrightrow=0;firstfactleftrow=0;
    flagstartleft=0;flagstartright=0;
    finalrow=0;
```

```
centerlast=zerocenterpoint;

for(i=0;i<=1;i++){              //前面十行的处理方法和后面不打一样

    flagrl=0;flaglr=0;          //每行寻找到左右边界的标志

  //寻找左边界
      for(j=centerlast;j<image_line;j++){
              if(flaglr==0)
               {
                    if(j==pointright)
                      {
                        blr[i]=pointright;      //到边界了还是么有找到
黑线的
                        flaglr=1;                //找到边界了
                        j=pointright;            //跳出循环
                      }
                    if(Blackwhite[i][j]==5 )    //找打黑点了
                      {
                        blr[i]=j;
                        flaglr=1;
                        j=image_line;
                      }
               }
      }//for    j 结束       向右寻找结束


       //寻找右边界
      leftzero=0;
      for(j=centerlast;j>0;j--)
      {
          if(flagrl==0)
          {
              if(j==pointleft)
                {
                  brl[i]=0;
                  flagrl=1;          //这个可以不要
                  j=0;               //这个也可以不要
                }
              if(Blackwhite[i][j]==5 )
```

```
                        {
                          brl[i]=j;
                          flagrl=1;
                          j=0;
                        }
                    }
            } //for    j 结束       左边寻找结束

            zeroleft=blr[0];
            centerlast=(blr[i]+brl[i])/2;   //暂时求取当前行的平均值

            BlackCenter[i]= centerlast;
            blr[0]=zeroleft;


    }// for   i 结束


    if(zerosingle==0){

          centerlast=(brl[1]+blr[1])/2;

          for(i=2;i<76;i++){
               lastleft=blr[i-1];lastright=brl[i-1];
               flaglr=0;flagrl=0;

              if(Blackwhite[i][centerlast]==250){

                  //寻找右边界
              if(rightside==0){
                  if(lastright>=t_offset)                    tright=lastright-t_offset;
      else
                      tright =0;
                  if(tright<(rightline[i]-2))   tright=rightline[i]-2;   //7.7 日修改
                  tleft=lastright+t_offset;
if(tleft>=leftline[i]) tleft=leftline[i]-1;              //跑到右边边界了

                  for(j=tleft;j>tright;j--){   //从赛道最里面向最左边寻找
                     if(flagrl==0){
                          if(Blackwhite[i][j]==5 ){
```

```
                flagrl=1;
                brl[i]=j;
                j=tright;     //为了快点跳出来的
            }
        }
    }//for   j 结束                    这个是寻找当前行的右边黑线

    if(flagrl==0)          /                    {
     brl[i]=brl[i-1];
     tlr++;
     if(i<image_row-3){
         for(j=tleft;j>tright;j--){
             if(flagrl==0){
                 if(Blackwhite[i+1][j]==5 ){

                     flagrl=1;
                     brl[i+1]=j;
                     j=tright;     //为了快点跳出来的
                 }
             }
         }//for   j 结束                        又没有找到黑线

         if(flagrl==0)                                    {
                 brl[i+1]=brl[i];
                 tlr++;            //正常情况下应该是 2 了
                  for(j=tleft;j>tright;j--){    //从赛道最里面向最左边寻
找
                     if(flagrl==0){
                         if(Blackwhite[i+2][j]==5 ){

                                 flagrl=1;
                                 brl[i+2]=j;
                                 j=tright;
                             }
                         }
                     }//for   j 结束
                     if(flagrl==0)          //第三次
                     {
                         tlr++;
```

```
                                rightrow=i;       //右边的有效行的最后一行
                                lookleft(blr[i-1],i,1);
                                lookright(brl[rightrow-1],rightrow,0);
                                 F_roadright(rightrow);

                                if( leftrow>rightrow) {
                                   flagimage=1 ;
                                   finalrow= leftrow;
                                }
                                else if ( leftrow==rightrow){
                                   flagimage=5;
                                   finalrow=rightrow;
                                }
                                else    if( leftrow<rightrow){
                                     flagimage=2;
                                     finalrow=rightrow;
                                }

                                i=76;
                                rightside=1;
                                leftside=1;
                            } //  结束    if(flaglr==0)     第三次
                        } //  结束    if(flaglr==0)     第二次
                    } //  结束    if(i<image_row-3)
                } //    结束    if(flaglr==0)    第一次
            }    //    结束  if(rightside==0){    整幅图像有效的右边都找到了


          //寻找左边界


        if(leftside==0){
            rleft=lastleft+t_offset;
            if(rleft>image_line)  rleft=image_line;    //右边不能超过数组的边界
了

            if(t_offset>lastright){
               rright=t_offset-lastleft;
            }
            else{
```

```
        rright=lastleft-t_offset;
    }

//    for(j=rleft;j<rright;j++){
     for(j=rright;j<rleft;j++){
        if(flaglr==0){
            if(Blackwhite[i][j]==5 ){
                blr[i]=j;
                flaglr=1;
                j=image_line;
            }
        }//flagrl 结束
    }//for   j 结束

    if(flaglr==0)                    {
        blr[i]=blr[i-1];
        trl++;              //正常情况下是 1
        if(i<image_row-3){
            for(j=rright;j<rleft;j++){
                if(flaglr==0){
                    if(Blackwhite[i+1][j]==5 ){
                        blr[i+1]=j;
                        flaglr=1;
                        j=image_line;
                    }
                }//flagrl 结束
            }//for   j 结束

            if(flaglr==0){
                blr[i]=blr[i-1];
                trl++;       //正常情况下是 2
                for(j=rright;j<rleft;j++){
                    if(flaglr==0){
                        if(Blackwhite[i+2][j]==5 ){
                            blr[i+2]=j;
                            flaglr=1;
                            j=image_line;
                        }
                    }//flagrl 结束
                }//for   j 结束
```

```
                    if(flaglr==0){
                        trl++;
                        leftrow=i;       //左边的有效行的最后一行

                        lookright(brl[i-1],i,1);
                        lookleft(blr[leftrow-1],leftrow,0);

                        F_roadleft(leftrow);

                        if( leftrow>rightrow) {
                           flagimage=1 ;
                           finalrow= leftrow;
                        }
                        else if ( leftrow==rightrow){
                           flagimage=5;
                           finalrow=rightrow;
                        }
                        else    if( leftrow<rightrow){
                           flagimage=2;
                           finalrow=rightrow;
                        }

                        i=76;
                        leftside=1;
                        rightside=1;
                    } //结束  if(flagrl==0)  第三次
                } //结束    if(flagrl==0)  第二次
            }    //if(i<image_row-3)
        }   //结束    if(flagrl==0)  第一次
    }   //  结束  if(leftside==0)

    if(i<76)                        {
            if((blr[i]<leftline[i])&&(flagstartleft==0)){
                firstfactleftrow=i;
                flagstartleft=1;
            }
            if((flagstartright==0)&&(brl[i]>rightline[i])){
                firstfactrightrow=i;
                flagstartright=1;
```

```
}

if(i>5){     //修改了哈位置，更真实的反应图上信息    7.3 日
   if(blr[i]<(leftline[i]-1)){
      if((brl[i]==rightline[i])||(brl[i]==(rightline[i]+1))){

            rightrow=i;       //右边的有效行的最后一行


            lookleft(blr[i-1],(i-1),1);
             lookright(brl[rightrow-1],rightrow,0);

            if( leftrow>rightrow) {
            flagimage=1 ;
            finalrow= leftrow;
            }
            else if ( leftrow==rightrow){
            flagimage=5;
            finalrow=rightrow;
            }
            else    if( leftrow<rightrow){
            flagimage=2;
            finalrow=rightrow;
            }

             i=image_row;

      }
   }
}

if((i<76)&&(i>5)){
   if(brl[i]>(rightline[i]+1)){
    if((blr[i]==leftline[i])||(blr[i]==(leftline[i]-1))){

          leftrow=i;       //左边的有效行的最后一行

          lookright(brl[i-1],i,1);
          lookleft(blr[leftrow-1],leftrow,0);
          if( leftrow>rightrow) {
```

```
                    flagimage=1 ;
                    finalrow= leftrow;
                    }
                    else if ( leftrow==rightrow){
                    flagimage=5;
                    finalrow=rightrow;
                    }
                    else    if( leftrow<rightrow){
                    flagimage=2;
                    finalrow=rightrow;
                    }


                    i=image_row;
                 }
                }
              }

             centerlast=(blr[i]+brl[i])/2;

             if(i==75)          //整张图像都处理了
              {
                flagimage=5;                    //整幅图像都完成了
                finalrow=75;
                leftrow=75;
                rightrow=75;
             }
       }// end if (i<76)
}
else{    //     if(Blackwhite[i][centerlast]==250)
          if(i>5){
            if(blr[i]<=blr[i-5]){

               flagimage=3;           //右丢线了 只有 左边一条线
            }else

               flagimage=4;           //左丢线了 只有 右边一条线
          }

          finalrow=i;
```

```
                i=image_row;
            }       //      if(Blackwhite[i][centerlast]==250)          }//for   i 结束
    }//zerosingle==0    结束


}


/*****************************************************************************
******
*                                   川工第一漂开发小组
*
*   函数名称：findcentercenter(void)
*   功能说明：寻找图像的中心线
*   输入参数: flagimage,finalrow,brl[i],blr[i],leftrow,rightrow
*   函数返回：BlackCenter[i] ,finalrow,
*   修改时间：2012-7-1     已测试
*   参数说明：    flagimage=0 ;    危险的              Straightrow   直道
*                 flagimage=1 ;   右转                 rightcrosspoint   十字的特
殊点
*                 flagimage=2 ;   左转                 lefhtcrosspoint   十字的特
殊点
*                 flagimage=3 ;   右转                 leftrow      左边有效线的
最后一行
*                 flagimage=4 ;   左转                 rightrow     右边有效线的
最后一行
*                 flagimage=5 ;   类似直到或者小 S     rightcontious=3  十字
*                 flagimage=10;   十字                 leftcontious=4   十字
*               //   offset_value     中线和边界线的一个偏移量 通过计算得出
*                 wuxiaoimage   = 1  则图像无效
*   备      注：    从开始到单线开始有效行(leftrow/rightrow)之间是两边都有线
的，
*                 中心值=左右边界平均值
*                   从单线有效行(leftrow/rightrow)到图像的有线的最后一行
finalrow
*                 中心值=单线+-offset_value
*                 当遇上十字的时候，最开始处理到十字的特殊点就截止了
*
*                 当任何一行中左右两个边界差值大于  70  则认为  此张图像是
无效图像不能够
```

```
*                   处理的 （十字的） wuxiaoimage  = 1 舵机处理的时候保持
上一次的值
*
**********************************************************************
*****/
void       findcentercenter(void){
           uint8 i,k, t,crossrow,zhongxin;

           uint8       offset_value;
           uint16      sum;

           sum=0; wuxiaoimage=0;Straightrow=0;zhongxin=0;t=0;

           if(flagimage==5){                  //整幅图像都有
               for(i=0;i<finalrow;i++){
                   BlackCenter[i]= (blr[i]+brl[i])/2;
                   sum=sum+BlackCenter[i];                //可以删掉
                }

               zhongxin=sum/finalrow;

               for(i=0;i<finalrow;i++){      // Straightrow;
                  if(((BlackCenter[i]-zhongxin)<6)&&((BlackCenter[i]
-zhongxin)>(-6)) ){
                        t++;
                  }else{
                        i=finalrow;
                  }

                }
               if(t==finalrow){
                     Straightrow=1;           //7.4 日加的
               }

           }else   if(flagimage==0){
                 wuxiaoimage=1;
           }
           else   if(flagimage==1){     //右边

              if((rightcontious==4)&&(rightcrosspoint>1)){
```

```
        finalrow=rightcrosspoint;

        if(rightrow>rightcrosspoint){
            rightrow=rightcrosspoint;
        }
    }

     for(i=0;i<rightrow;i++){
         BlackCenter[i]= (blr[i]+brl[i])/2;
         sum=sum+BlackCenter[i];                    //可以删掉

        if((blr[i]==leftline[i])&&(brl[i]==rightline[i])){
            if((blr[i]-brl[i])>70){
               wuxiaoimage=1;
            }
         }
     }
     offset_value=blr[rightrow-1]-BlackCenter[rightrow-1];

     for(i=rightrow;i<finalrow;i++){

         if(blr[i]>offset_value) {
              BlackCenter[i]= blr[i]-offset_value;                    }else {
            BlackCenter[i]=0;
         }
        sum=sum+BlackCenter[i];                    //可以删掉
         if((blr[i]==leftline[i])&&(brl[i]==rightline[i])){
            if((blr[i]-brl[i])>70){
               wuxiaoimage=1;
            }
         }

      }   //   end      for(i=rightrow;

}
else    if(flagimage==2){         //左边

   if((leftcontious==3)&&(leftcrosspoint>1)){
```

```
        finalrow=leftcrosspoint;

        if(leftrow>leftcrosspoint){
            leftrow=leftcrosspoint;
        }
    }

     for(i=0;i<leftrow;i++){
         BlackCenter[i]= (blr[i]+brl[i])/2;
         sum=sum+BlackCenter[i];                    //可以删掉

         if((blr[i]==leftline[i])&&(brl[i]==rightline[i])){
            if((blr[i]-brl[i])>70){
               wuxiaoimage=1;
            }
         }
      }
     offset_value=BlackCenter[leftrow-1]-brl[leftrow-1];

     for(i=leftrow;i<finalrow;i++){
         BlackCenter[i]= brl[i]+offset_value;
         sum=sum+BlackCenter[i];                //可以删掉

         if((blr[i]==leftline[i])&&(brl[i]==rightline[i])){
            if((blr[i]-brl[i])>70){
               wuxiaoimage=1;
            }
         }

      }
}
else if(flagimage==3){                  //只有左边一条单线      右转
   for(i=0;i<finalrow;i++){

      if(blr[i]>28) {
         BlackCenter[i]= blr[i]-28;     //让中心值等于左边边界
      }else {
         BlackCenter[i]=0;
       }
       sum=sum+BlackCenter[i];              //可以删掉
```

```
                if((rightcontious==4)&&(rightcrosspoint>1)){
                    if(i>= rightcrosspoint){
                      i=finalrow;
                    }
                  }

              }
            }
            else if(flagimage==4){            //只有右边一条单线     左转
              for(i=0;i<finalrow;i++){
                    BlackCenter[i]= brl[i]+28;     //让中心值等于右边边界
                    sum=sum+BlackCenter[i];                 //可以删掉
                    if((leftcontious==3)&&(leftcrosspoint>1)){
                     if(i>=leftcrosspoint){
                        i=finalrow;
                      }
                    }
              }
            }
            else if(flagimage==10){         //十字
              for(i=0;i<finalrow;i++){
                    sum=sum+BlackCenter[i];         //可以删掉
              }

            }

}


/*****************************************************************************
******
*                                    川工第一漂开发小组
*
*   函数名称：findcrosscenter(void)
*   功能说明：进入十字的判读
*   参数说明：
*   输入参数: Blackwhite[i][j] ，rightline[i]，leftline[i],zerocenterpoint
*   函数返回：flagimage,finalrow, BlackCenter[i],firstrowtwoline
*   修改时间：2012-7-1     已测试
*   备       注：
```

```
*                从第三行开始从第二行的中线开始处左右分别寻找两条线，若判断
*                出左右两条边线都不等于实际图像边界时候，则认为找到对面的有
*                双线，找到双线后的双线行中心值为左右边界值的平均值，然户从
*                开始出现双线的哪行 firstrowtwoline 开始向第 1 行赋值，这些行的
*                中心值都为  BlackCenter[firstrowtwoline+2]    flagimage=10
*
**********************************************************************
*****/
void      findcrosscenter(void){
            uint8 i,j,t,centerlast,flaglr,flagrl,ktime;

            uint8         firstrowtwoline;            //十字路寻找对面的双线

            centerlast=80;ktime=0;firstrowtwoline=2;finalrow=2;
            for(i=3;i<60;i++){
              finalrow++;
              flagrl=0;flaglr=0;
              for(j=centerlast;j<leftline[i];j++){
                 if(Blackwhite[i][j]==5 ){
                              blr[i]=j;
                              flaglr=1;
                              j=image_line;
                         }
              }
              if(flaglr==0)    blr[i]=leftline[i];

              for(j=centerlast;j>rightline[i];j--){
                 if(Blackwhite[i][j]==5 ){
                              brl[i]=j;
                              flagrl=1;
                              j=2;
                         }
              }
              if(flagrl==0)    brl[i]=rightline[i];
```

```
                    //下面是寻找特殊点的了
                      if(ktime==1){

if(   (brl[i]==rightline[i])&&(blr[i]<leftline[i])&&(brl[i-5]>rightline[i-5])&&(blr[i-5]
<leftline[i-5])){
                                finalrow=i;
                                t=i;
                                i=75;
                            }
                            else
if((blr[i]==leftline[i])&&(brl[i]>rightline[i])&&(brl[i-5]>rightline[i-5])&&(blr[i-5]<
leftline[i-5])){
                                finalrow=i;
                                t=i;
                                i=75;
                            }

                        }



if((brl[i]<(leftline[i]-2))&&(brl[i]>(rightline[i]+2))&&((blr[i]-brl[i])<70)){
                            centerlast=(blr[i]+brl[i])/2;
                            BlackCenter[i]=centerlast;
                             ktime=1;
                        }

                        if(ktime==0){
                            firstrowtwoline++;
                        }

                    }

                    for(i=(firstrowtwoline+2);i>0;i--){
                          BlackCenter[i]=BlackCenter[firstrowtwoline+2];
                    }
                   flagimage=10;

}
```

```
/*****************************************************************************
******
*                                  川工第一漂开发小组
*
*   函数名称：Startline(void)
*   功能说明：起跑线的的判读
*   参数说明：
*   输入参数: Blackwhite[i][j] ， rightline[i]，leftline[i], blr[i],brl[i]
*   函数返回：flagstartline
*   修改时间：2012-7-1     已测试
*   备       注：
*                   如果斜率在一定的直道范围类，并且有效行大于 70 则 从第
二行开始
*                   到第五十行从中间向两边处理
*
*                   如果找到某一行连续一段黑线长度在 8 到 18 之间，则是起跑
线的一半
*                   找到两段长度在 8 到 18 之间的线，且两段所在行数相差在一
行左右，且
*                   两段的近端边界值在 9 到 20 之间 则为起跑线 flagstartline=1
*****************************************************************************
*****/
void       Startline(void){
     uint8 i,j,m,m1,centerlast,leftstart1,rightstart1;

     //起跑线的变量
     uint8     startlineright,startlineleft,row1=0,row2=0;
     int     t=10,y=0;

     startlineright=0;startlineleft=0;flagstartline=0;m=0;m1=0;

     centerlast=(blr[1]+brl[1])/2;

        for(i=2;i<50;i++){
          if((blr[i]-brl[i])<61){

          centerlast=(blr[i-1]+brl[i-1])/2;

          for(j=centerlast;j<(blr[i]-2);j++){
```

```
    if((Blackwhite[i][j]==5)&&(Blackwhite[i][j+1]==5)){
            startlineleft++;
            m=j;
    }
} //end for(j=)

if((startlineleft>=5)&&(startlineleft<=18)){          //5  到 18 个点
        leftstart1=m-startlineleft;
        row1=i;                    //7.19 日添加
        flagstartline=1;    //7.15 日添加
    //    i=75;                    //7.15 日添加
}

if(i<75){                          //7.19 日添加

      for(j=centerlast;j>(brl[i]+2);j--){

        if((Blackwhite[i][j]==5)&&(Blackwhite[i][j-1]==5)){
              startlineright++;
              m1=j;
        }

      } //end for(j=)
       if((startlineright>=5)&&(startlineright<=18)){          //5 到 18 个点
              rightstart1=m1+startlineright;
              row2=i;
              flagstartline=1; //7.15 日添加
         //       i=75;                    //7.15 日添加
        }

      if((row1>0)&&(row2>0)){                //7.19 日添加
          t=row1-row2;
          y=leftstart1-rightstart1;
          if((t>(-6))&&(t<6)&&(y<25)&&(y>5)){
              flagstartline=1;
              i=75;
          }
      }
}
```