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：　　华中科技大学

队伍名称：　　华中科技大学三队

参赛队员：　　熊　枭

　　刘　洁

　　宋生乾

带队教师：　　彭　刚

　　何顶新

关于技术报告和研究论文使用授权的说明

本人完全了解第五届全国大学生“飞思卡尔”杯智能汽车竞赛关保留、使用技术报告和研究论文的规定，即：参赛作品著作权归参赛者本人，比赛组委会和飞思卡尔半导体公司可以在相关主页上收录并公开参赛作品的设计方案、技术报告以及参赛模型车的视频、图像资料，并将相关内容编纂收录在组委会出版论文集中。

参赛队员签名：_____

带队教师签名：_____ 日　期：___

目录

第 1 章　引言

全国大学生智能汽车比赛是经全国高等教育司研究，委托高等学校自动化 专业教学指导分委会主办的，旨在培养创新精神、协作精神，提高工程实践能 力的大学生科技创新活动。

参赛选手须使用竞赛秘书处统一指定并负责采购竞赛车模套件，采用飞思卡尔半导体公司的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、动力电机驱动、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。

本文从整体车模的机械、电路、硬件等方面，详细地阐述了基于数字摄像 头的智能巡线小车的设计。

1.1　概述

在历时近半年的智能车的准备过程中，我们经过多次尝试和实验以及理论分析，最终确定了使用 MS9S12XS128 作为智能车核心处理芯片，同时使用数字摄像头 OV5017 作为图像传感器对道路进行识别，并在电路、机械等方面实现创新，最终成功实现了基于数字摄像头图像采集系统，并确定了适当的摄像头视野以及合理的控制算法，使智能车在高速运行的状态下能够平稳、顺利的完成比赛。

1.2 智能车系统框架图

智能车系统在大方向上主要分为四部分：电路部分、程序部分、机械结构 部分、辅助部分。智能车系统结构框如图1.1所示：

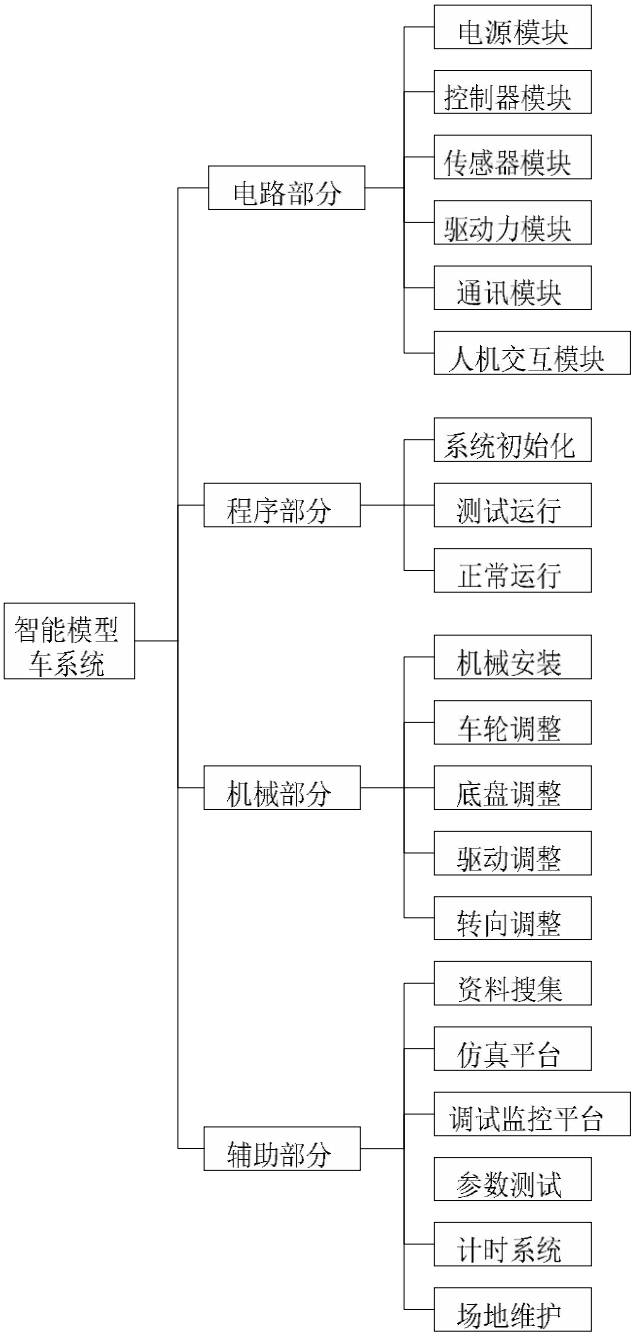

智能模型车系统
电路部分
电源模块
控制器模块
传感器模块
驱动力模块
通讯模块
人机交互模块
程序部分
系统初始化
测试运行
正常运行
机械部分
机械安装
车轮调整
底盘调整
驱动调整
转向调整
辅助部分
资料搜集
仿真平台
调试监控平台
参数测试
计时系统
场地维护

图 1.1 智能车系统框图

1.3 智能车设计

1.3.1 电路设计

为了实现高分辨率、硬件周期短的智能车图像识别系统，在智能车设计的过程中，对于图像的处理，使用外部存储器——FIFO 存储图像像素点的坐标，设 计的中实现了图像的硬件二值化，并合理的处理了分辨率，分辨率外部设置， 使单片机能够合理的采集图像进行处理。

主要处理芯片使用了比较经典的 MS9S12XS128 作为核心处理单元，外围使用 BTS7960 作为电机的驱动电路，实现了智能车优良的加减速性能和稳定的工作环境。

1.3.2 机械设计

机械方面，首先对舵机的安装进行了改装，加长了舵机的连杆，以增加反应 速度。另外，对于智能车的地盘以及转向系统、减震系统进行了仿真，使机械 设计尽量科学，合理。

在安装方面，摄像头的安装部件都采用了先设计再用铝材料进行加工的方式，使得安装更加简介、稳固。摄像头的安装杆则采用超轻的碳纤维材料，尽 量降低智能车重心，减少智能车的重量。

1.3.3 控制决策

图像处理方面，采用了图像压缩的方法，对于采集到的原始图像进行处理， 减少了图像的数据量，减轻了单片机的负担，方便了单片机对于整个赛道信息 的处理。
智能车控制方面通过机械的设计安装，智能车的控制主要从转向、加减速等 方面着手考虑。对于转向系统，通过相应的理论计算，我们得到了舵机转角与 图像偏差的关系，并利用此关系实现了智能车的转向的控制，得到了良好的效 果。

1.3.4 调试平台

为了实现良好的调试效果，并在制作的过程中减少控制决策的工作量，我们 用 VB设计了相应了图像和数据调试工具，方便的对于采集 的图像进行处理，并对小车的数据信息进行实时监控，达到了事半功倍的效果。

1.4 本文结构

本文从智能车设计的电路、机械、软件、调试平台等方面，详细地介绍了智能车的整个设计过程。其中，第二章为智能车硬件电路的设计，第三章为智能车机械安装方案的的详细描述，第四章为智能车控制方案的设计，第五章为调试工具的设计和使用，第六章为智能车技术参数的说明，最后为智能车调试心得和总结。

第 2 章　智能车硬件电路设计

智能车的硬件设计在整个智能车的设计过程中占有相当重要的分量,稳定合理的硬件是智能车告诉运动的必要条件。我们智能车硬件的实际遵循电路简单、功能强大、方便调试等原则，使用了自己制作的单片机最小系统，并设计了相 应的图像处理硬件，极大地方便了单片机对于道路信息的采集和处理，并实施 相应的控制策略。

2.1　单片机最小系统

freescale公司MC9S12XS128,16bit、112pin的单片机其集成了PIM，TIM，PWM,SPI,SCI,ECT,CAN,AD,PIT,CAN等模块，使用方便功能强大。Freescale公司K60P144M100SF2RM单片机是ARM内核的单片机，功能比MC9S12XS128强大了不少，其eDMA模块较好的提高了程序的效率，同时在硬件上也减少了不少器件，包括FIFO等。我们在linux下开发K60，由于方案不是很成熟，我们设计了兼容两套最小系统的主板，MC9S12XS128(图1.2)和K60（图1.1）最小系统板如下：

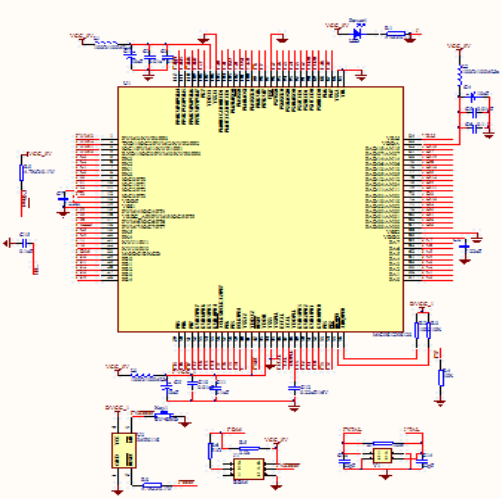


图1.1

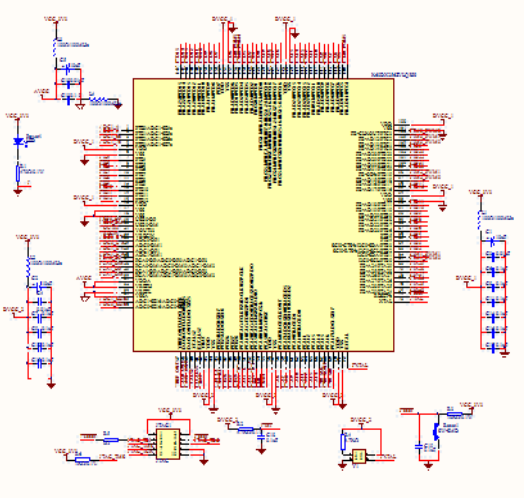
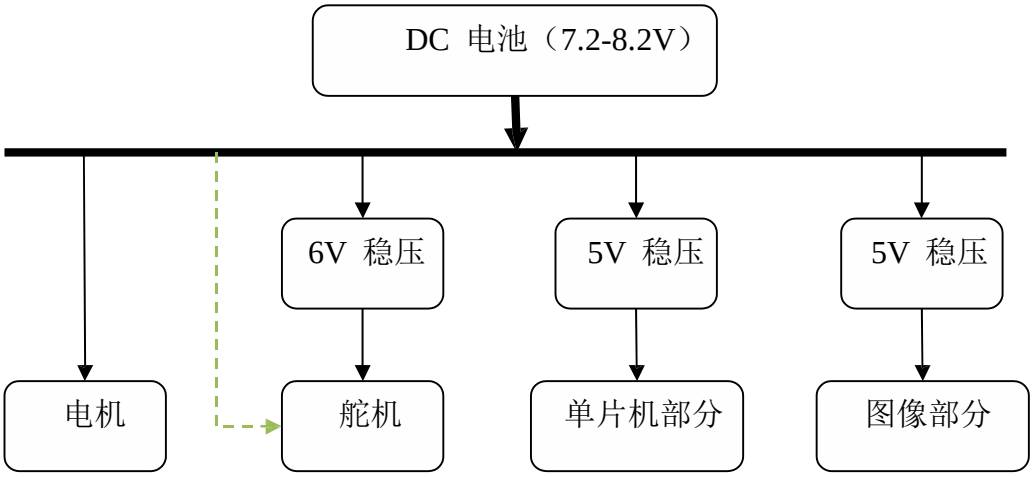


图1.2

2.2 电源管理

系统里面的电源由组委会提供的电池，并且禁止使用 DC-DC 升压电路直接 为驱动电机以及舵机提供动力。本系统里面需要给电机、舵机、单片机部分、 摄像头及图像处理部分供电。

其电能分配如下所示（图表2-1）：



图表 2-1 电源分配

2.2.1 电机供电

由于电池采用组委会统一提供的电池,同时又不允许采用升压电路,为了给 电机最大的驱动能力，因此采用电池电压直接供电的方式。

由于中间未加稳压环节,因此电机的供电电压波动范围是比较大的。刚充满电的电池有8.2V 左右，电量不多的时候会降低到7.2V 以下（更低就建议立即 停止使用进行充电）如何回避电池电压给驱动电机带来的影响是一个没有去解 决的问题。

2.2.2 舵机供电

舵机的手册中推荐的供电是5V 或6V,可能在5V 条件下,会延长舵机使用 寿命，但是由于比赛性质，追求的更多的是性能。因此，我们首先考虑使用6V电压给舵

机供电。

对于6V 的供电：我们并没有对舵机的电流等进行测试,只是在我们的了解范围内寻找了最好的稳压芯片。能够找到的一些稳压芯片如下：

名称	输出电压(V)	最大输出电流（A）	压差（V）	备注
LM7806	6±0.12	2.2	2.0	线性稳压
LT1084	可调	5	1.2	线性稳压
LM1117ADJ	可调	0.8	1.2	线性稳压

由于电池电压一般为7.8V 左右，最低到7V，要稳压到6V，对稳压芯片的 压差要求还是比较高的，我们用的是LT1084，其最大电流能达到5A，能比较好的满足要求。电路如下（图1.3）：

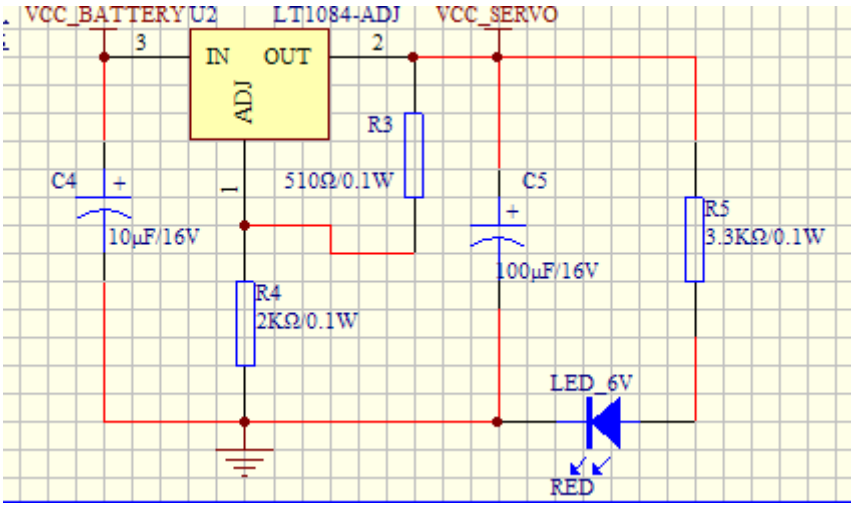


图1.3

2.2.3 单片机部分供电

单片机及部分外设工作时，需要 5V 电压。常见的可以提供 5V 的稳压芯片如下：

名称	输 出 电 压（V）	最大输出电流（A）	压差（V）	备注
LM7805	5+0.2	1.0	2.0	线性稳压
LM2940	5±0.15	1.0	0.5	开关稳压

两者相比较而言，LM2940 具有超低压差的优点，可以避免赛车启动、制动等过程中电池电压下拉之后单片机的重启。同时，开关稳压的效率高，耗电更少，使电池充电后使用更长久。所以我们优先选择LM2940-5V。其电路如下（图1.4）：

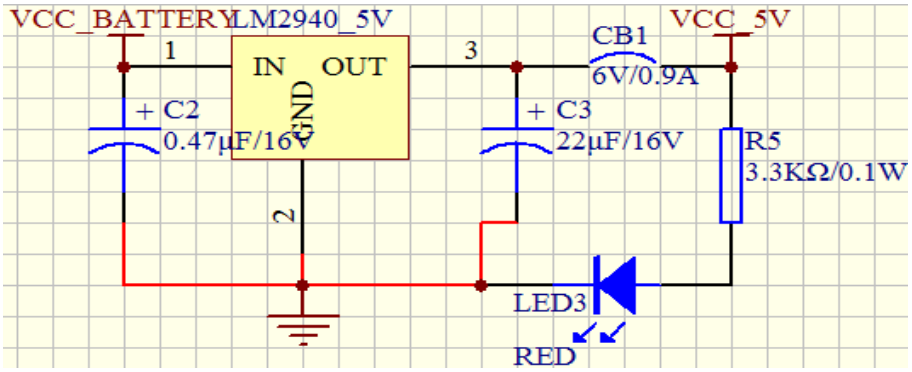


图1.4

2.2.4 摄像头及图像处理部分供电

摄像头部分包括摄像头（OV5017）以及信号处理板上所有元件的供电，均 为5V 电压，由于都是数字芯片，其耗电也不大，不足0.2A。因此，摄像头部分我们采用一片LM2940 的方案。同上，这里 不再赘述。

2.2.5 防反插二极管

我们设计了防反插二极管，宁愿牺牲电池，保护电路元件。如下（图1.5）所示：

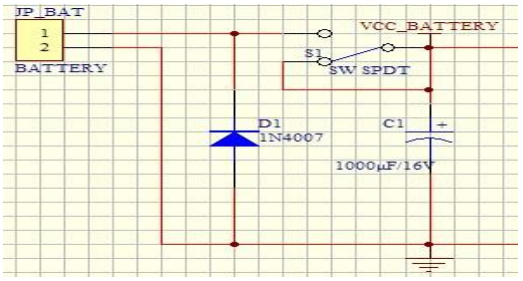


图1.5

2.3 电机驱动部分

英飞凌的BTS7960B 半桥驱动芯片，负载电流可以达到 43A，而内阻只有 16 mΩ。

BTS7960 是一款针对电机驱动应用的完全集成的大电流半桥芯片。它是 NovalithICTM 系列的成员之一，它的一个封装中集成了一个 P 通道场效应管在上桥臂和一个 N 通道场效应管在下桥臂以及一个控制集成电路。由于上桥臂采用的是 P 通道开关，对于电荷泵的需求也就不复存在了，因此电磁干扰减至了最小。由于集成在内驱动集成电路具有逻辑电平输入，与微控制器的连接变 得非常简单，且该驱动集成电路还具有电流检测诊断、转换率调整、死区时间 生成以及过热、过压、欠压、过流和短路保护。BTS7960的特点如下：

在 25℃时导通电阻的典型值为 16 mΩ
低静态电流，在 25℃时的典型值仅为 7 μA
与主动续流相结合的脉宽调制能力高达 25kHz
开关电流限制降低功耗的过流保护
电流限制在典型的 43A
具有电流检测能力的状态标志诊断
具有锁定行为的过热关断
过压锁定
欠压关断
带有逻辑电平输入的驱动电路
用于优化电磁干扰的可调节转换率

2.4 摄像头处理

对数字式的摄像头模块进行的选型，要求黑白的、分辨率不能太高，否则无 法处理其数据。最终我们选定了黑白数字摄像头OV5017，它基本数据如下：

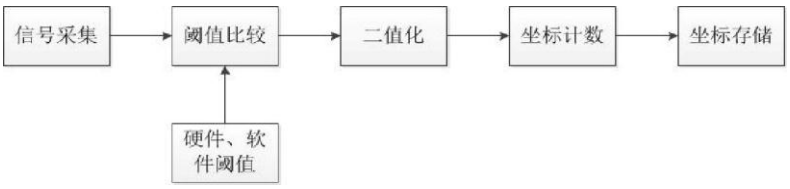
图像速度：50 帧/s---0.5 帧/s（即：20ms/帧----500ms/帧）；
图像大小：（最大）384*288，可开窗调节大小； 感光能力：
最小0.5 勒克斯（光圈：1.4，速度：50 帧/s） 一般配置为：
50 帧/s 不开窗、自动光圈

2.4.1 数字式摄像头设计思想

由于设计时以分辨率高为原则，并在设计初期就为了实现硬件对于摄像头图像的二值化，并且考虑到了单片机处理的数据量等多方面问题，最终使用了 OV5017 分辨率192*72，50 帧/s，并使用FIFO 对于摄像头图像进行坐标存储。

2.4.2 图像采集流程图

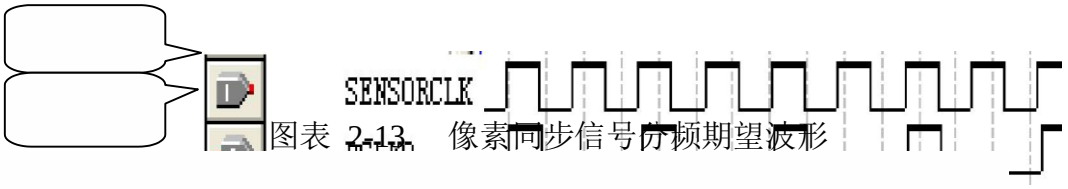
设计中，主要的硬件处理流程如下：



图表 2-12 硬件二值化流程图

2.4.3 像素同步信号处理

像素同步信号一直存在,摄像头数据总线上的数据是在像素同步信号的上升 沿更新，下降沿锁存，设计需求将像素同步信号处理成以下：



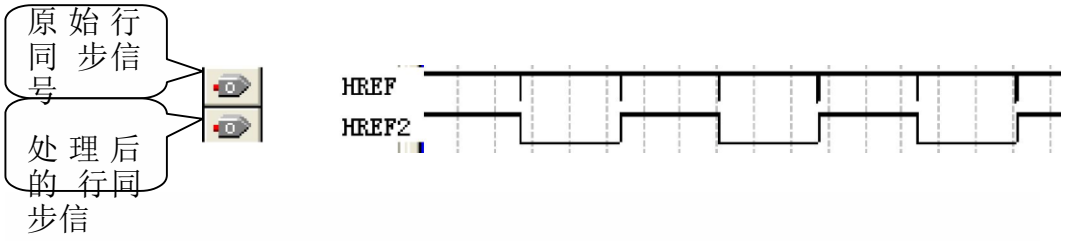
图表 2-13 像素同步信号分频期望波形

使用普通的逻辑器件，就能实现上述波形。这样，横向分辨率处理为192

2.4.4 行同步信号处理—HREF

对于分辨率，我们的需求是192*72，因此，对于行像素，要进行分频处理， 如果使用二分频，则最终的分辨率为192*144，四分频为192*72。

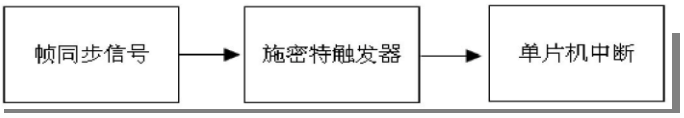
以二分频为例：



图表 2-14 行同步信号分频期望波形

2.4.5 帧同步信号处理

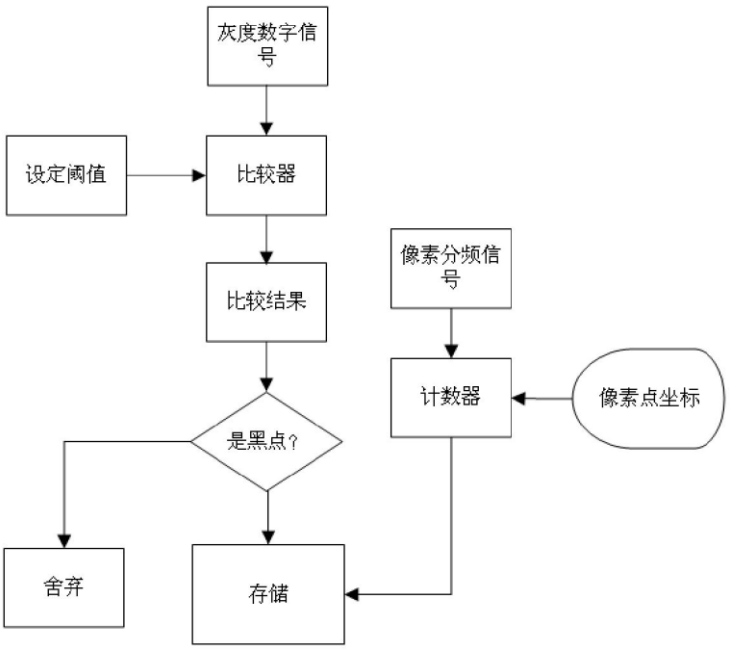
帧同步信号的处理，我们将其处理之后直接接到单片机中断，接收帧同步中 断，并以此为信号单片机处理上一帧图像的数据。



图表 2-14 帧同步信号处理

2.4.6 阈值比较和黑点存储部分

阈值的比较是整个系统工作的关键，在摄像头工作时，对于像素同步信号进 行计数，转化为黑点的坐标，将坐标进行存储，主要的模块流程如下：



图表 2-14 阈值比较和黑点坐标存储

第三章 机械设计

3.1 车体结构

智能车竞赛所使用的车模是一款带有差速器的四轮驱动模型赛车，它由大赛组委会统一提供，其机械结构简单、材料轻便强硬、零件加工精度较高。下面具体介绍 A 型车模的基本参数。

基本参数	尺寸
轴距	197mm
前轮距	140mm
后轮距	140mm
车轮直径	50mm
传动比	18/76
车模重量（不含电池）	446.55g
电池（净重）	300g
舵机（净重）	41.74g

表 3.1 智能车模基本参数

组委会规定改装后车模的最大尺寸为：宽 250mm，长 400mm，高度不限。不得随意更换车模的零件，如有损坏可以用相同的材质零件代替。

3.2 前轮定位

前轮定位对赛车的速度有很大的影响，虽然车模比较小，但是前轮定位作用还是不容忽视的。前轮定位的内容有：主销后倾、主销内倾、主销外倾、前轮前束。

1）主销

转向轮（前轮）围绕主销进行旋转，前轴的轴荷通过主销传给转向轮，具备这两点的就叫做主销。主销内倾角和车轮外倾角度主要是由转向节决定。如图 3.1 所示。

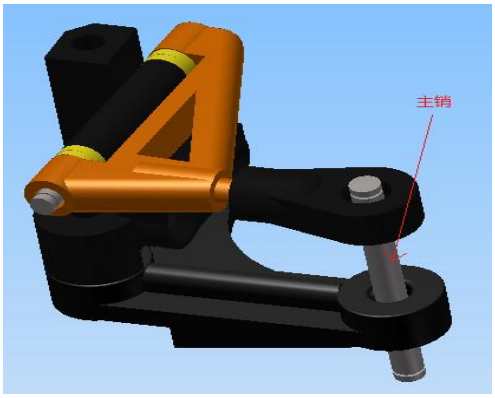


图3.1

2）主销后倾角

主销向后倾斜，主销轴线与地面垂直线在赛车纵向平面内的夹角称为主销后倾角。

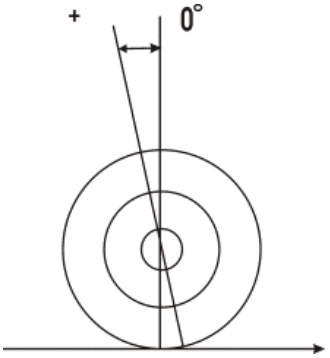


图 3.2 主销后倾角示意图

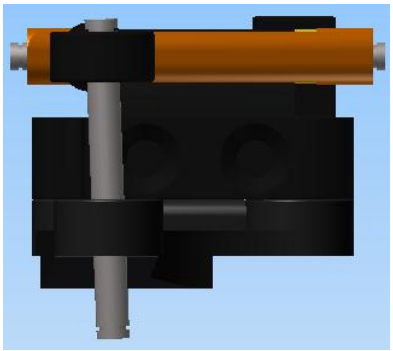


图 3.3 主销后倾角三维图

前轮重心在主销的轴线上，由于主销向后倾斜使前轮的重心不在车轮与地面的接触点上，于是产生了离心力，主销后倾形成的离心力，可以保证汽车直线行驶的稳定性，还可以帮助车轮自动回正。主销后倾角延长线离地面实际接触越远，即主销后倾角越大，车速越高，离心力越大，行驶就更加稳定。

主销后倾角在车轮偏转后形成一回正力矩，阻碍车轮偏转，主销后倾角越大，车速愈高，车轮偏转后自动回正力矩越强。但回正力矩过大，将会引起前轮回正过猛，加速前轮摆振，并使转向沉重，通常后倾角为1～4度，主要由四块黄色垫片进行调节，经过反复试验，我们采用了2:2的数量比安装方式。

3）主销内倾角

主销在横向平面内向内倾斜，主销轴线与地面垂直线在赛车横向断面内的夹角

称为主销内倾角。

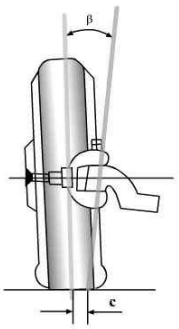


图 3.4 主销内倾角示意图

图片来源：百度图片

主销内倾角也有使轮胎自动回正的作用，当汽车转向轮在外力作用下发生偏转时，由于主销内倾，则车轮连同整个汽车的前部将被抬起一定高度，在外力消失后，车轮就会在重力作用下力图恢复到原来的中间位置。但主销内倾角不宜过大，否则在转弯时轮胎将与赛道间产生较大的滑动，从而会增加轮胎与路面间的摩擦阻力，使转向变得沉重，同时会加速轮胎的磨损。

4）前轮外倾角

通过车轮中心的汽车横向平面与车轮平面的交线与地面垂线之间的夹角，称为前轮外倾角。

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。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。由于我们所使用的车模的前轮零件是固定的，所以我们无法对前轮外倾角做修改，但是我们可以改变主销内倾/外倾角来改变前轮外倾角。

5）前轮前束

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 车体重心

1）车体重心的测定

确定车体重心可以通过实测和软件两种方法，实测法就是用吊线法来确定重心，先估计重心大概在何处，然后用线绑在该处绑起来，向上提起赛车，根据赛车倾斜的方向，重新挪动线的位置再提起，反复的移动测试，这样能大概地测出赛车的重心位置。软件法即用三维软件将车体完整的表示出来，通过给各个零件输入其密度等信息，由软件自动计算得出车体的重心。利用 Inventor 对设计好的赛车做重心分析。

2）车体重心位置调整

车体重心的位置对赛车加减速性能、转向性能和稳定性都有较大的影响。重心调整主要包括重心高度和前后位置的调整。

理论上，车体重心越低稳定性越好，重心低有利于赛车在高速转弯的贴地性，可以有效防止发生侧翻，因此在车体底盘高度、舵机安装、电路板的安装等上尽量使重心放低。

车体重心前后方向的调整对赛车的行驶性能也有很大的影响。本赛车为后轮驱动，驱动力的大小直接跟轮胎与地面的附着力有关，而附着力与路面附着系数和驱动轴的轴荷有关，驱动轴的轴荷又取决于重心的水平位置，如果仅从这方面考虑，重心应靠近后轴。

根据车辆运动学理论，车身重心前移，大部分重量压在前轮，转向负荷增大，会增加转向，对模型车的制动性能和操纵稳定性有益，但降低转向的灵敏度，同时降低后轮的抓地力；重心后移，会减少转向，但增大转向灵敏度，后轮抓地力也会增加。但综合起来看，重心应靠近后轴一点。

因此，在设计今年的方案时，将重心位置调整作为重点。

3.4 离地高度及底盘刚度调整

1) 底盘高度的调整

车模底盘的高度主要由赛道中的坡决定，在顺利过坡的前提下底盘越低越好。这样会使得重心更低，赛车的稳定型更强。车模底盘的高度可由前后悬架来调，可以在底盘与前悬架之间垫上不同高度的垫块。还可以通过调整后悬架

中的一个零件来改变底盘的高度。

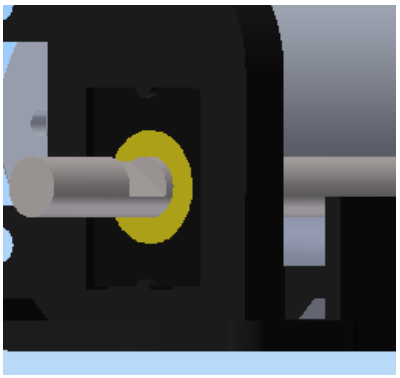


图 3.5

2)底盘刚度调整

A 型原装车模为了增加赛车在坎坷的路面上的适应性，设计了很好的减震系统，利用　　弹簧来减缓赛车在行驶过程中的震动，而且同时给左右晃动留有很大的余量。

对于 A 型车模，主要调整前后悬挂纵向减震刚度。适当增大底盘的刚度有利于车体走直线的稳定性，可通过增加垫片来增大弹簧的预紧力，另外还可以通过调整弹簧的另一个

支点的位置来改变预紧力，从而提高底盘的刚度。虽然增加弹簧的预紧力的效果确实是很好，但是考虑到弹簧占的空间有点大，阻碍了传感器和电路板的安装，对整车的布局影响很大，所以最终还是决定将其去掉，直接将前后悬架用螺钉紧固定起来。

3.5 差速机构的调整

差速机构的部件为：左右两片滑片、带滚珠的大齿轮（76 齿）、转轴和放松螺母，其结构是左侧的轮子与转轴为插入连接，两滑片和大齿轮通过黄色的垫块套在转轴中，而且两滑片紧贴着大齿轮的滚珠，右轮与右侧的滑片紧贴，也是通过黄色垫块套在转轴中，并有放松螺母固定。通过调节放松螺母的松紧度来调节两滑片与大齿轮的滚珠之间的滑动摩擦力度，从而提供左右的驱动力，当左右车轮受到的阻力不同时，将会产生差速。

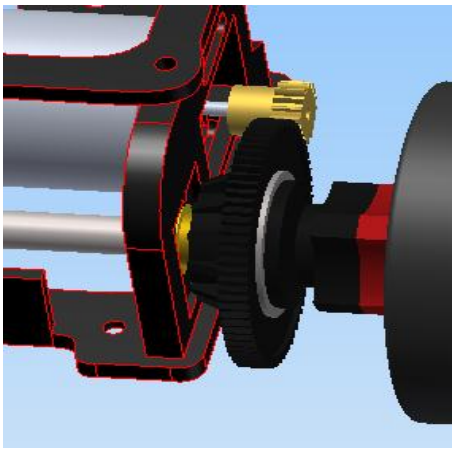


图3.6

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。

差速器的特性是：阻力越大的一侧，驱动齿轮的转速越低；而阻力越小的一侧，驱动齿轮的转速越高。以此次使用的后轮差速器为例，在过弯时，因外侧前轮轮胎所遇的阻力较小，轮速便较高；而内侧前轮轮胎所遇的阻力较大，轮速便较低。

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。

对于差速器的调整和改进一直是参赛队员研究重点，在第六届的时候提出了一种方案，在放松螺母固定处加上滚动轴承以代替原车模的铁片，这样使得车模本身的阻力减小，而且提高了差速性能。但是，今年组委会规定，不得使用滚动，这就使得差速器部分没什么可改进的

了，只能在其里面上多点润滑脂增加润滑效果。还有就是多研究下放松螺母的松紧度，找到最适中的位置。

3.6 舵机的固定

舵机有立式和卧式两种安装方案（这里以舵机转轴为参考对象，立式：转轴处于水平方向；卧式：转轴处于竖直方向），如图 3.7 和图 3.8 所示：

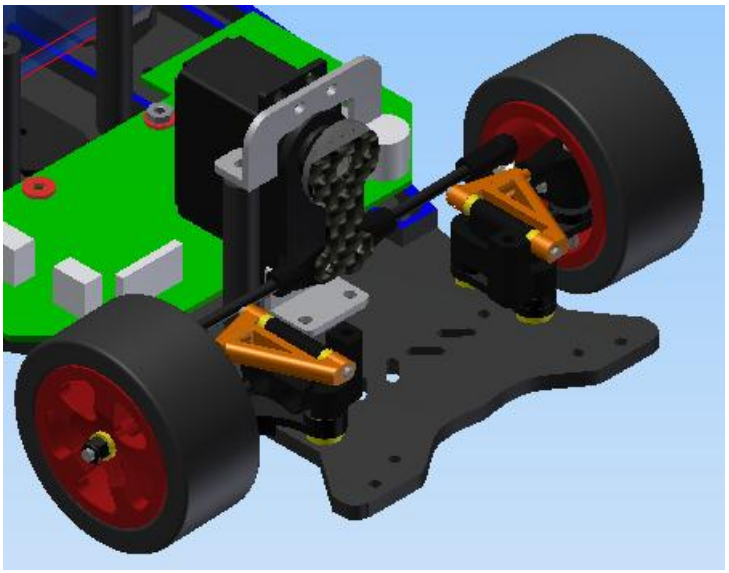


图 3..7 舵机立式安装

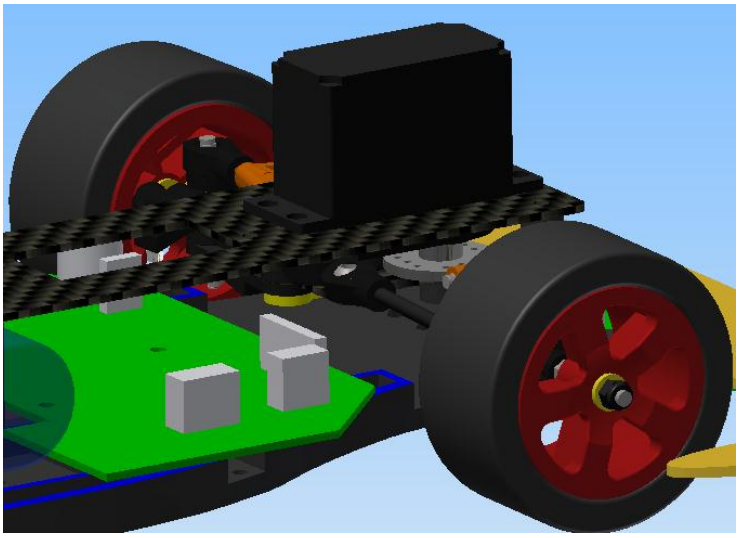


图3.8 舵机卧式安装

舵机立式安装方式的优点：（1）转向响应速度快，转向角较为符合阿克曼转向原理，它由舵机臂竖直平面的运动转化为拉杆水平方向的运动，减少了在同一平面上运动的死区；（2）方便安装舵机臂，有利于调节赛车转向的中心值。

舵机立式安装方式的缺点：（1）不好安装固定；（2）安装后较高，占用竖直方向的空间，会挡到摄像头（3）重心较高。

舵机卧式安装方式的优点：（1）转向响应速度较快；（2）高度较低；（3）重心低。

舵机卧式安装方式的缺点：转向角部分符合阿克曼转向原理（转角小时），舵机臂和拉杆都在水平平面内运动，当舵机臂长度与转角臂长度相等时会导致内、外侧

轮不符合阿克曼转角。

经过仿真分析后得出舵机立式安装方式较好，立式安装方式的转角较符合阿克曼转角。很多实力非常强的学校都采用了立式安装方式，如北京科技大学，图 3.9 为北科在第 5 届光电组所用赛车，图 3.10 为北科摄像头组所用赛车。

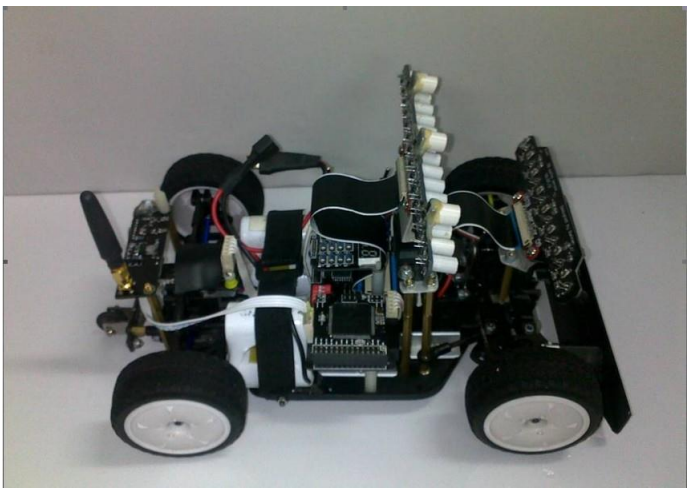


图 3.9

图片来源：第五届北京科技大学的光电赛车

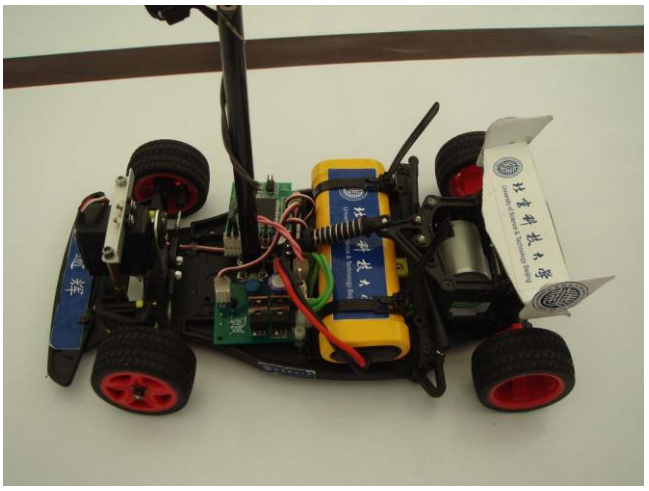


图 3.11

图片来源：北京科技大学的摄像头赛车

为此，可利用 Inventor 和 Excel 对车模进行仿真，并根据阿克曼原理算出理论转角，然后进行比较分析。

首先，将车模简化为简单的几何模型进行分析。然后根据此模型分析出车模参数的几何关系：$V / \tan \alpha = V / \tan \beta + H$。再用控制变量法，改变 α（外侧车轮的角度），算出 β（内侧车轮的角度）。最后，将算出的值和利用 Inventor 实测的数据导入到 Excel 中进行分析。分析结果如下表 3.2 和表 3.3。

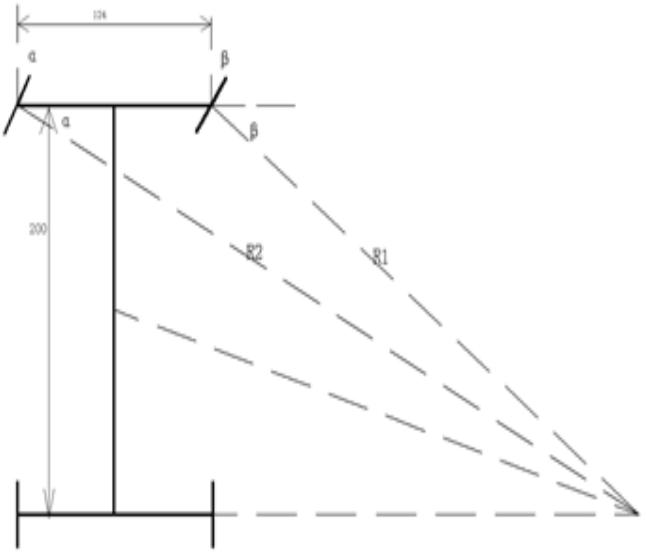


图3.12

图片来源：基于SoildWorks的机械分析报告（华中科技大学智能车队 姚叶明）

	A	B	C	D	E
1	左轮实际值	右轮实际值	右轮理论值	理论与实际差值	内外车轮的差角
2					
3	33.69	27.16	24.4664373	2.693562702	9.223562702
4	30.74	25.31	22.79734985	2.512650151	7.942650151
5	27.3	23	20.78099047	2.219009534	6.519009534
6	24.71	21.16	19.20478176	1.955218235	5.505218235
7	21.45	18.73	17.13832115	1.591678855	4.311678855
8	18.2	16.18	14.97267432	1.207325681	3.227325681
9	15.41	13.89	13.01687109	0.873128914	2.393128914
10	12.45	11.38	10.83049712	0.549502885	1.619502885
11	9.24	8.54	8.31208467	0.22791533	0.92791533
12	6.31	5.86	5.861348102	-0.001348102	0.448651898
13	3.4	3.11	3.264984533	0.154984533	0.135015467
14	0	0	0	0	0

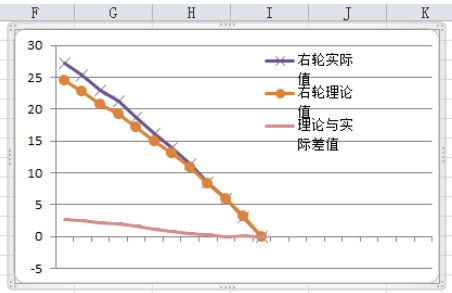


表 3.2 卧式安装

	A	B	C	D	E
1	左轮实际值	右轮实际值	右轮理论值	理论与实际差值	内外轮的差角
2					
3	33.06	25.14	24.11417623	1.025823774	8.945823774
4	30.21	23.68	22.49190864	1.188091356	7.718091356
5	27.31	22.06	20.78697397	1.273026029	6.523026029
6	24.78	20.55	19.24810484	1.301895162	5.531895162
7	21.79	18.63	17.3585219	1.271478103	4.431478103
8	18.28	16.22	15.02737545	1.192624546	3.252624546
9	15.13	13.9	12.81518206	1.084817937	2.314817937
10	12.39	11.77	10.78489037	0.985109628	1.605109628
11	9.52	9.43	8.538384692	0.891615308	0.981615308
12	6.56	6.88	6.076585954	0.803414046	0.483414046
13	3.3	3.91	3.172653257	-0.737346743	0.127346743
14	0	0	0	0	0

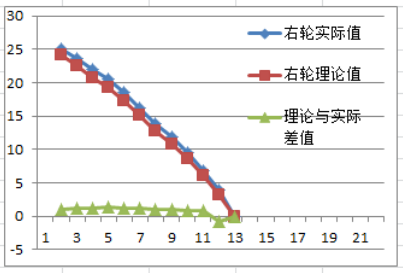


表 3.3 立式安装

根据分析的结果，可以得出：（1）立式和卧式安装方式在转角较小时，跟阿克曼转角比较吻合；（2）在转角较大时，立式比卧式更接近阿克曼转角。故立式安装方式

是较为理想的安装方式。

3.7 编码器的固定

编码器是用来对车速进行测算的仪器，目前很多学校使用的是通过光电编码的编码器，其精度较高、安装较为方便，但体积有点大。光电编码器的安装精度较高，要求编码器轴与赛车后轴同轴，且编码盘必须与赛车的中心线共线。对于 A 型车模来说，编码器的安装一般采取两种方式：一、直接装在车上，利用齿轮与差速器上的大齿轮相咬合。这种安装方式实际上是在测电机的转速，而不是真正的车速。二、在编码器上安装一个码盘，将码盘拖在地上，这样可
以测出赛车的实际速度，这里指的实际速度也是相对的，是一种理想情况，是在赛车没有发生侧滑、漂移和离地的情况下测得的速度。
以上两种安装方式都有各自的优缺点：

	方式一（固定在车上）	方式二（拖在地上）
优点	1、安装简单； 2、占用空间小，空间利用率高； 3、缩短赛车的长度；	1、测得的速度为赛车实际速度；
缺点	1、测得的是电机速度而不是实际速度； 2、增加了传动的阻力；	1、安装复杂； 2、对编码盘的加工精度要求较高； 3、在赛车行驶过程中容易弹起来； 4、加长了赛车的长度；

表 3.4

经过比较分析，方式一的整体效果要比方式二好，故近几年编码器的安装几乎都是采用方式一，只有去年电磁的赛车采用方式二，因为去年组委会规定有长度的限定，为了最大限度的
增加前瞻，不得不用方式一的安装方法来缩短赛车车体所占的长度。采用方式一进行安装方式，可以测出的真实速度，而且这种安装方式比较成熟。

基于 A 车摸方式一的安装方式如图 3.13。 基于 A 车摸方式二的安装方案如下图 3.14。

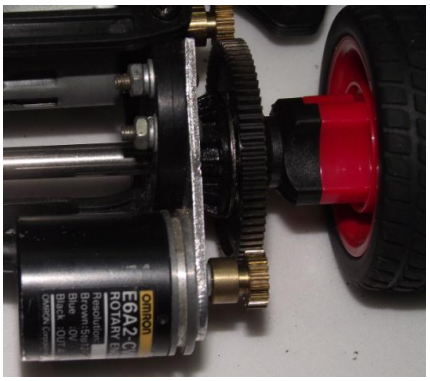


图 3. 13

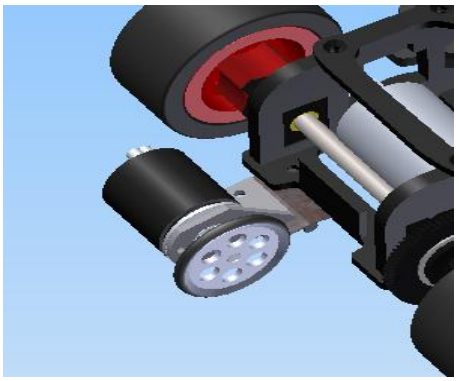
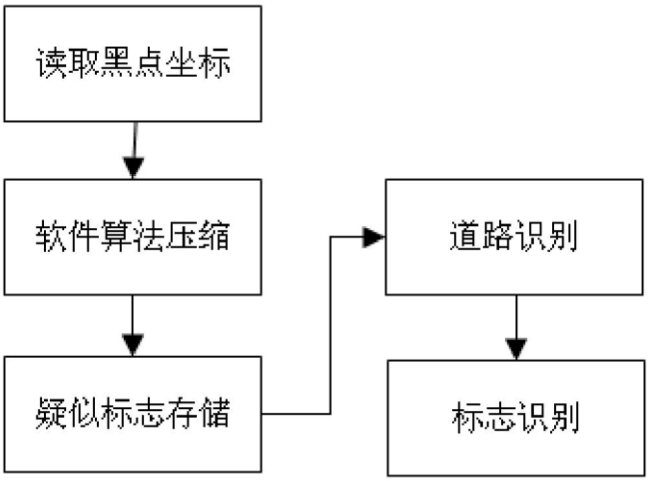


图 3.14

第 4 章　智能车软件设计

4.1　图像采集与道路识别

图像的采集，主要由硬件完成并实现二值化，这样，单片机对于图像的处 理，只需要从外部FIFO 中读取黑点坐标的位置，然后对于图像进行压缩、简化， 最终提取出比较真实的道路，主要的；流程如下：

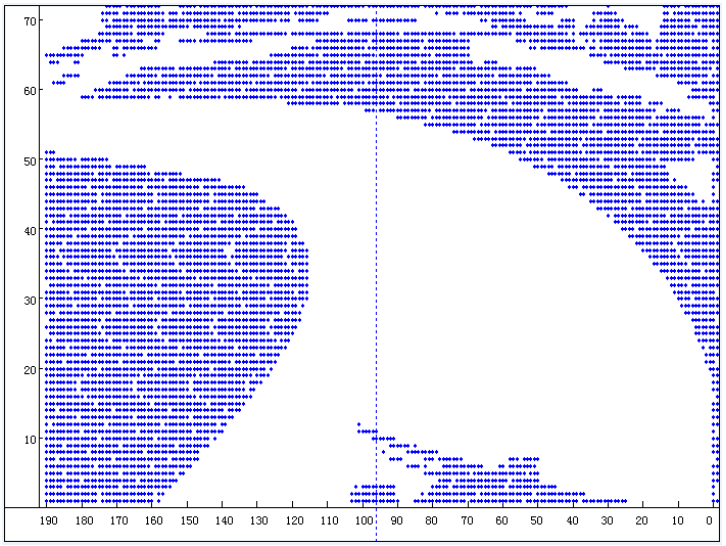


图表 4-1　图像处理流程

4.1.1　完整图像的读取

完整图像的读取，是单片机直接从外部存储器中，读取完整的图像，单片 机中，这幅图像是不能完整存储的，因此我们设计了相应的上位机软件（稍后 介绍），用来从串口接收单片机读取的图像。

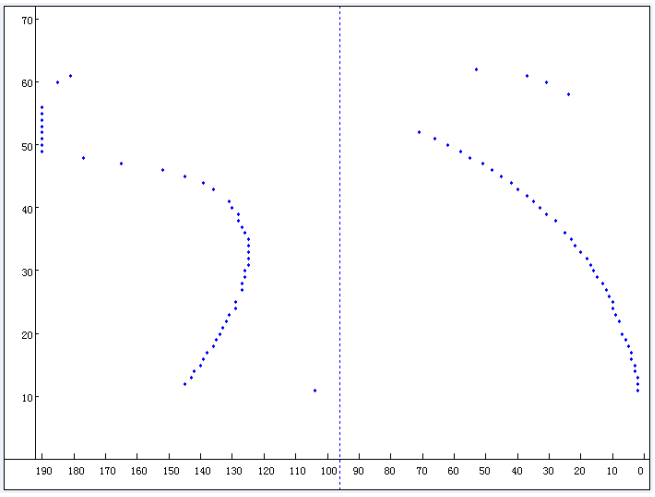
如图所示，摄像头经过图像二值化得到的完整图像：



图表 4-2 完整图像识别

4.1.2 图像压缩

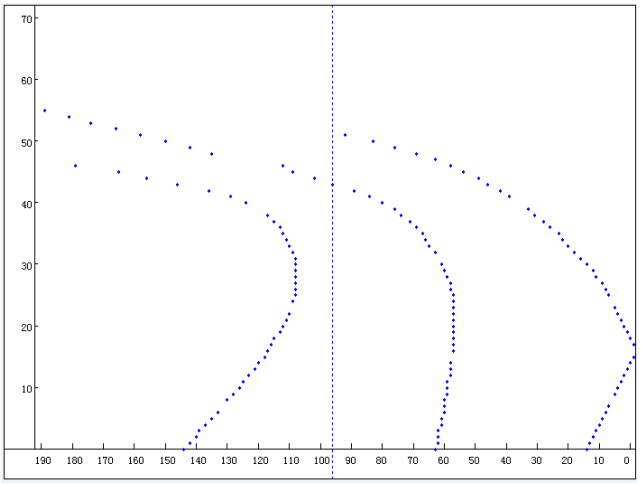
由于完整图像在的分辨率为 192*72，在单片机中，这样的数据时无法存储的，更别说处理了。因此，我们在算法中，对于完整的图像进行滤波处理，最 终将左右两边的道路分别存储为长度是 72 的数组，其下标作为行数，内容为本行跑道的坐标。如图4-3：



:

图表 4-3 压缩算法执行之后的赛道

通过两边的路最后得到中间的一条路如图 4-4：



图表 4-4　根据两边的赛道得到中间的路

4.2　特殊标志识别

在比赛中，特殊标志主要是起始线、直道、弯道、以及十字线等。

4.2.1　起始线的识别

起始线在赛车运行过程中，是至关重要的，起始线的识别直接决定了最终的成绩的好坏，因此，我们详细地研究了起始线的特征，从特征入手，最终解决 起始线的识别。

起始线的识别主要通过外围设备数字红外传感器，由于数字摄像头动态时在图像上很难识别到起始线。

4.3 控制策略

小车的控制,是为了小车能够在不冲出赛道的前提下,告诉稳定的运行,小 车的控制主要体现在速度控制和角度控制方面，同时配合速度和角度的控制。 速度的控制，我们使用了经典的增量式 PID 算法，角度的控制，考虑到角度的 模糊性，我们最终使用了角度偏差的线性控制，并且取得了比较好的效果。

4.3.1 速度控制

（1）速度控制概述

速度控制，我们最初选了了三套方案：PID 控制，bang-bang 控制，PID+bang-bang 控制，在实验的过程中，我们发现，PID 参数调节适当之后，其速度控制效果可以超越bang-bang 控制：

控制算法	带载调节时间（ms）	超调量
PID	365	5%
bang-bang	320	12%
PID+bang-bang	348	9%

图表 4-6 各种控制比较

可以看出，PID 参数调节适当，可以在速度控制上面取得很好的效果，Bang-Bang 控制虽然加速时间很短,但超调量比较大,在实际的小车行驶过程中，会极不稳定，因此，最终放弃了这种方案。

PID+bang-bang 控制在调节时间上面有所优势,总体效果而言,和单纯的PID 算法差别不大，最终也没有使用此方案。

（2）增量式PID 算法

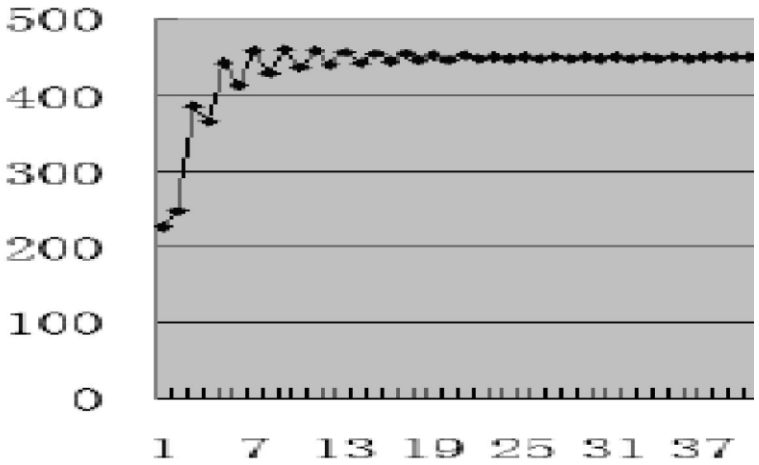
PID 算法中，我们最初尝试使用位置式PID 算法，其核心思想如下：

$$u(n)=K_P\left\{e(n)+\frac{T}{T_I}\sum_{i=0}^{n}e(i)+\frac{T_D}{T}\left[e(n)-e(n-1)\right]\right\}+u_0$$
$$=u_P(n)+u_I(n)+u_D(n)+u_0$$

图表 4-7 位置式PID 数字实现

位置式PID 在使用的过程中，编写程序时需要充分考虑各种参数，并且调试的过程中，会经常出现异常，对于积分项的处理比较麻烦，最终，我们尝试了增量式PID 算法，彻底解决了积分项的饱和问题。

在程序的设计中，按照上述公式编写程序，在速度控制方面，取得了很不错的效果。 最终的控制效果如图



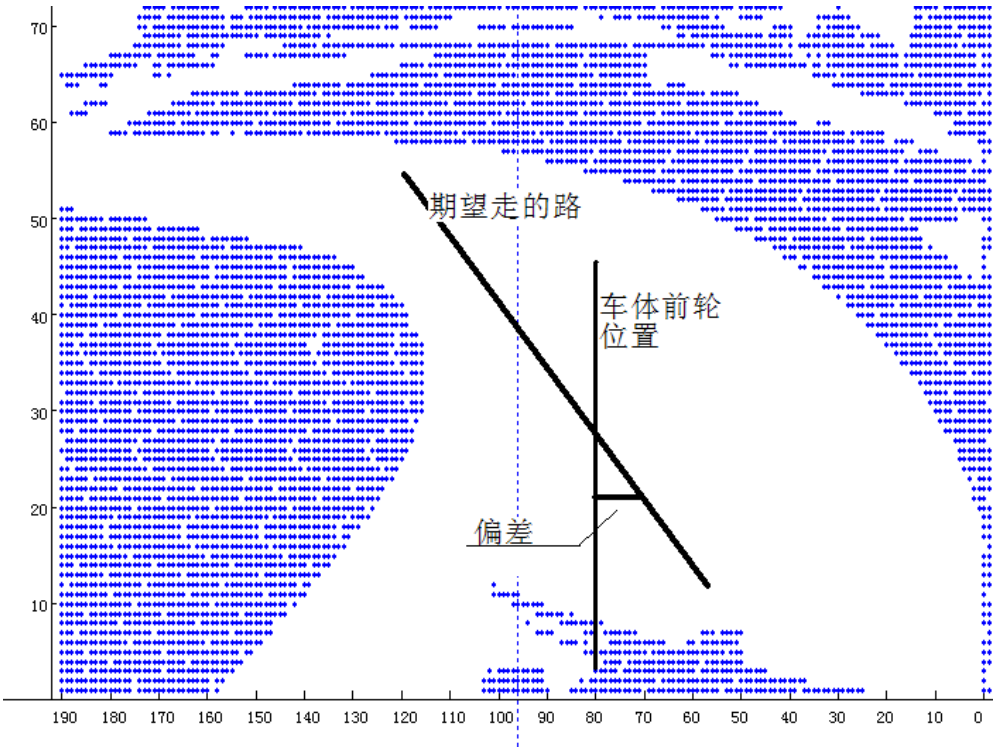
图表4-8 增量式PID 实际测量

其中，横坐标是时间度量（*10ms，纵坐标为速度度量（cm/s）

4.3.2 方向控制

智能车的方向控制决定了车体运行过程中的姿态,同时在很大程度上也决定了智能车在实际运行速度，在进行了理论分析的基础上，我们使用了比较简单 的对于偏差的线性控制，即通过图像得到小车与实际跑道之间的偏差，通过线 性控制得到舵机的转角。

如图所示：



图表 4-9 小车期望路线与中心偏差

从图像中，经过对于跑道图像的处理，得到小车行走的期望路线，与小车前轮所处的中心线，即图像中心比较得到一个偏差，根据偏差直接线性控制舵机转角。

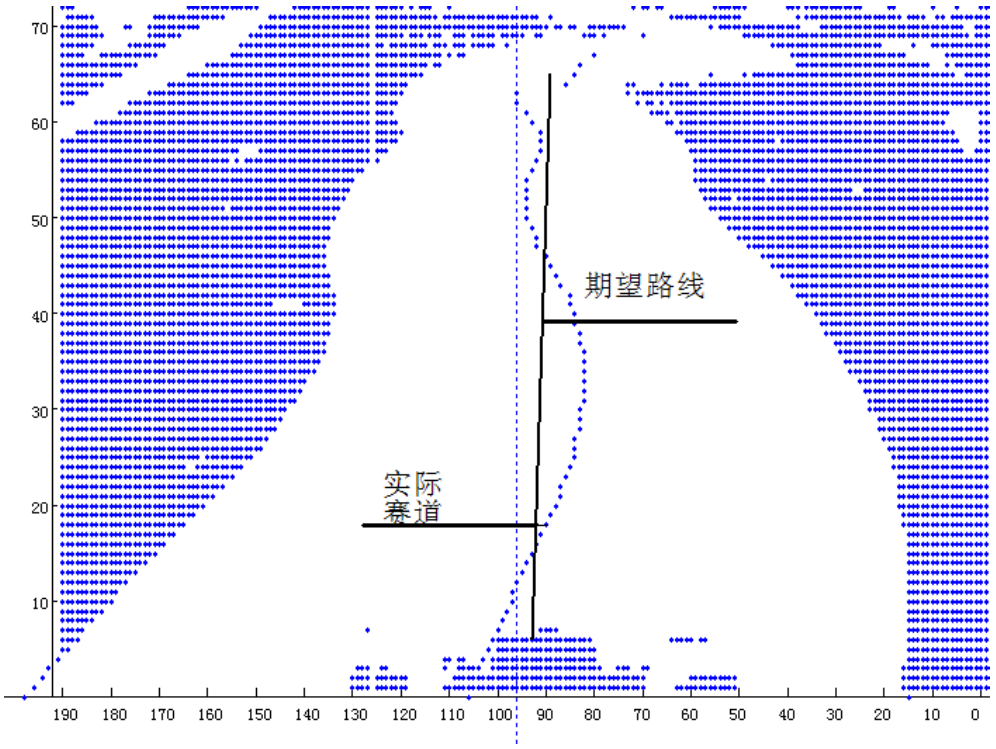
在这个过程中，所选取参考位置的远近影响到小车的转弯性能，因而，我们 也采取了根据速度的不同，动态地选取参考位置，充分考虑到了小车提前拐弯 的性能。

4.3.3 赛道优化

整个比赛，摄像头组最大的优势在于其“视觉”通过“视觉”小车能够判 断前方赛道的信息，并能够做出相应的判断措施，“抄近道”不仅是必然，要是小车能够告诉运行，也是必须！

小车对于赛道的优化，主要是对于波浪形赛道的优化和普通弯道的优化，对于超越小车直线行走的弯道，可以按照普通弯道处理。

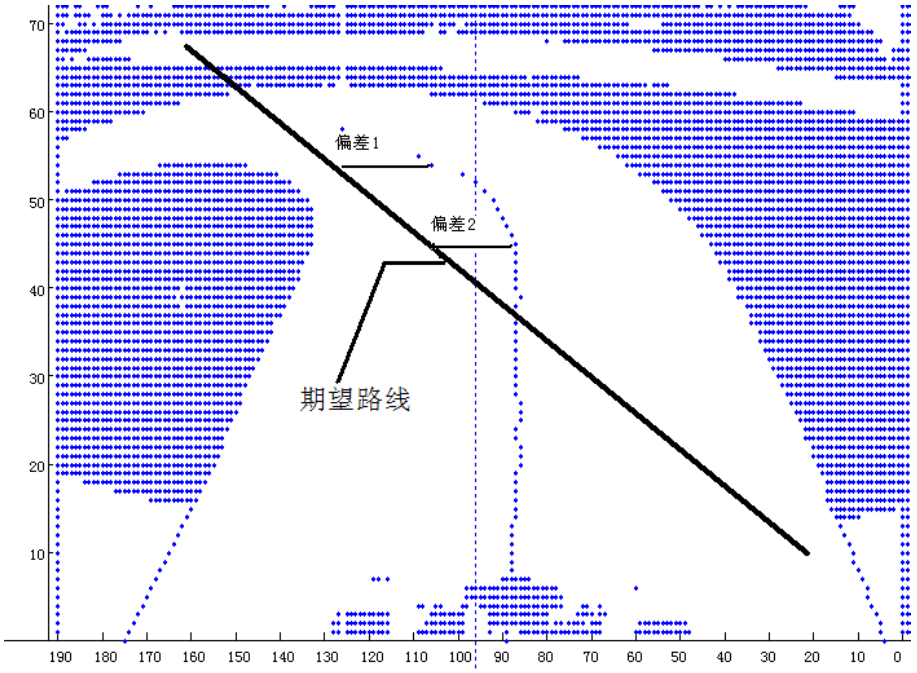
（1）对于小波浪形赛道：



图表 4-10　波浪形赛道处理

对于这种赛道的处理，我们直接利用了摄像头的“畸变”效应，由于摄像头的畸变效应，远处的像素点单位距离所代表的世界距离会很大，在图像中看到远处赛道与中心的偏差很小，所以在控制的思路在于取较远的参考行，计算偏差进行处理。这样也与上述的对于赛车的整体控制理论相符合。

（2）对于普通弯道赛道： 处理时只要将选取的参考行向远的方向移动，即可调节。 如图：



图表 4-10 普通弯道处理

这样，选择不同的参考行作为控制行可以实现入弯姿态的调整。

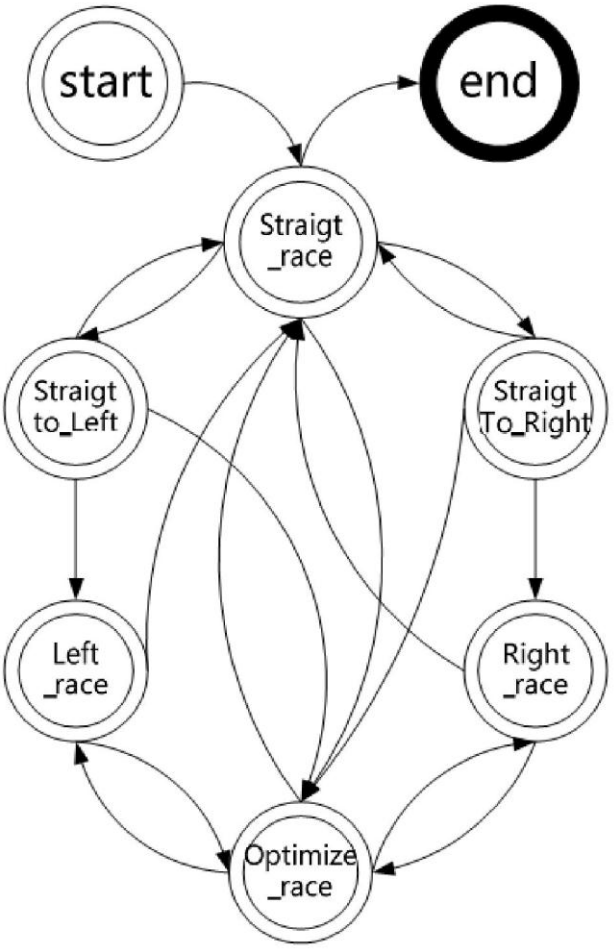
（3）对于大型波浪弯道的处理： 大型波浪弯的处理直接作为普通弯道，即可实现优化，这里不在赘述。

4.4 赛道模式

我们的控制策略是基于赛道模式的识别,即状态机。那么,模式的选择与跳 转的控制就极为重要。

经过长期的讨论、总结与归纳,我们最终确定了赛道的几种基本类型,同时， 针对不同的赛道类型，我们还详细的归纳了其检测函数，最终得到了适应各种 赛道类型的检测系统。

4.4.1 赛道模式转换图



4.5 单片机总体控制

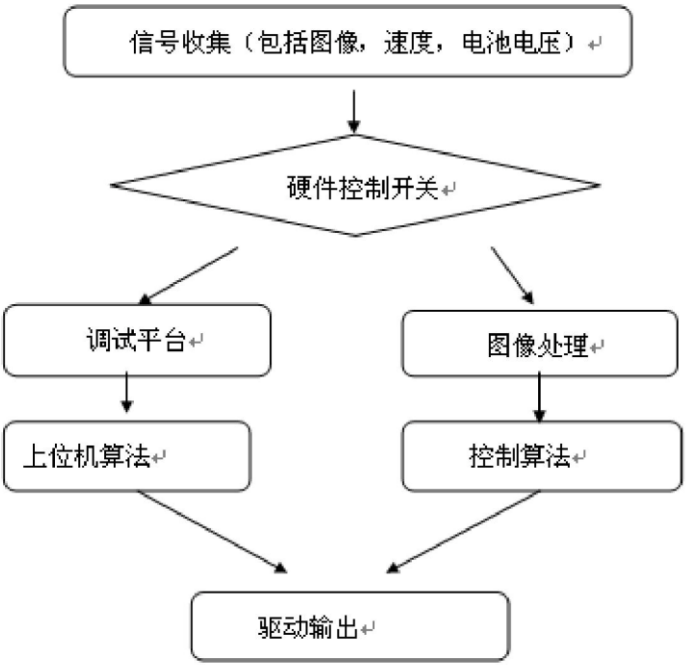
单片机对于整个车体的控制,在于实现各个模块的整合。在整个系统中,不 仅仅实现的是对于图像、速度、角度的调节，更重要的是实现了一个能够“稳 定”工作的系统，在总体控制中，我们还加入了调试工具的单片机通讯代码， 使得调试过程简洁，方便。

总体而言,单片机的信号输入主要有以下几个来源:摄像头、编码器、按键， 输

出主要为电机、舵机的控制信号，以及各种状态指示灯。

单片机的程序分设计到的一些控制参数，可以通过上位机输入并进行控制，这在调试的过程中是极为方便的！

整个系统工作模块图如下：



图表 4-11 单片机软件

在小车的整个运行过程中，人为的输入主要是模式开关的选择，实现了整个小车在不同模式状态下运行，尽量将小车的速度发挥到极致。

第 5 章　智能车调试和调试平台设计

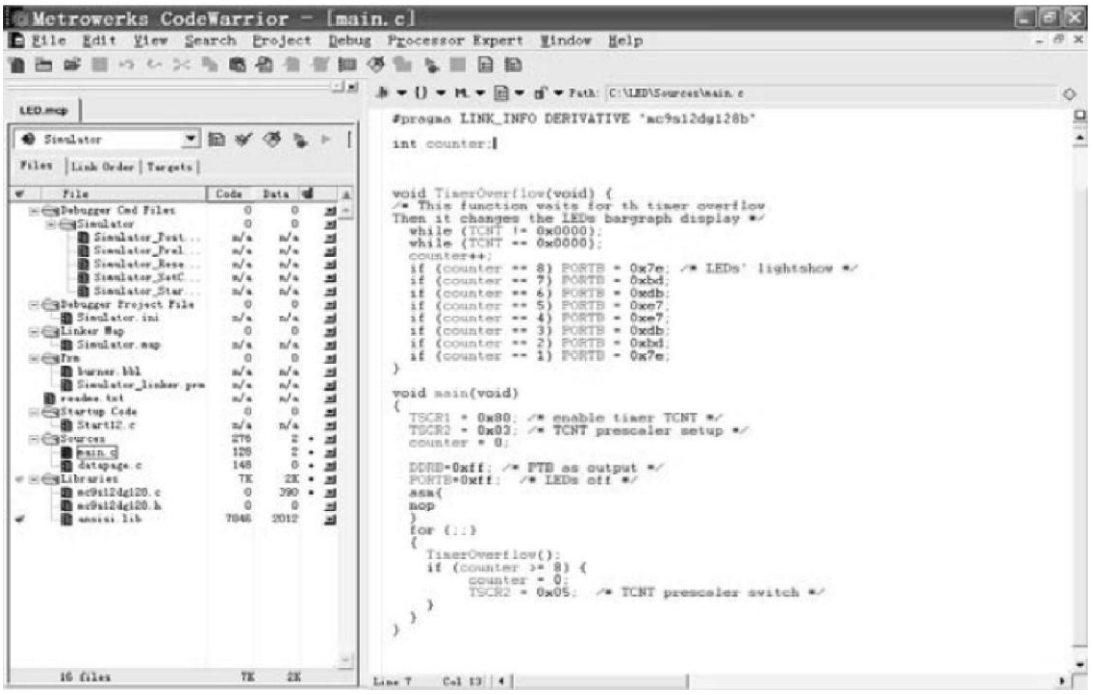
5.1　调试工具

系统的开发调试用到了Metroworks　公司的Code Warrior 3.1 开发软件，同时，我们使用了组委会推荐使用了CW4.7 作为主要的单片机程序下载和调试工具。

（1）编译工具

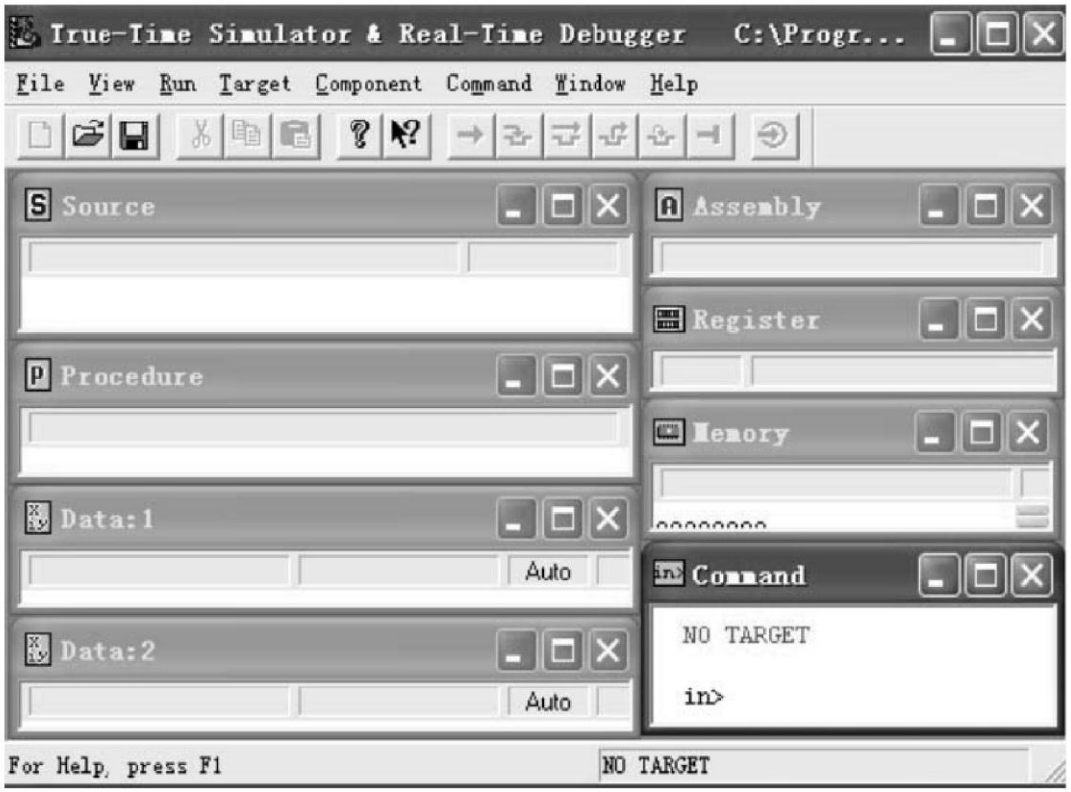
CW4.7 可以很方便的对单片机进行选择，并且在编写程序的过程中，强大的功能使得程序员编写的代码紧凑，高效。

使用界面如下：



图表 5-1　CW4.7　界面

（2）下载调试工具



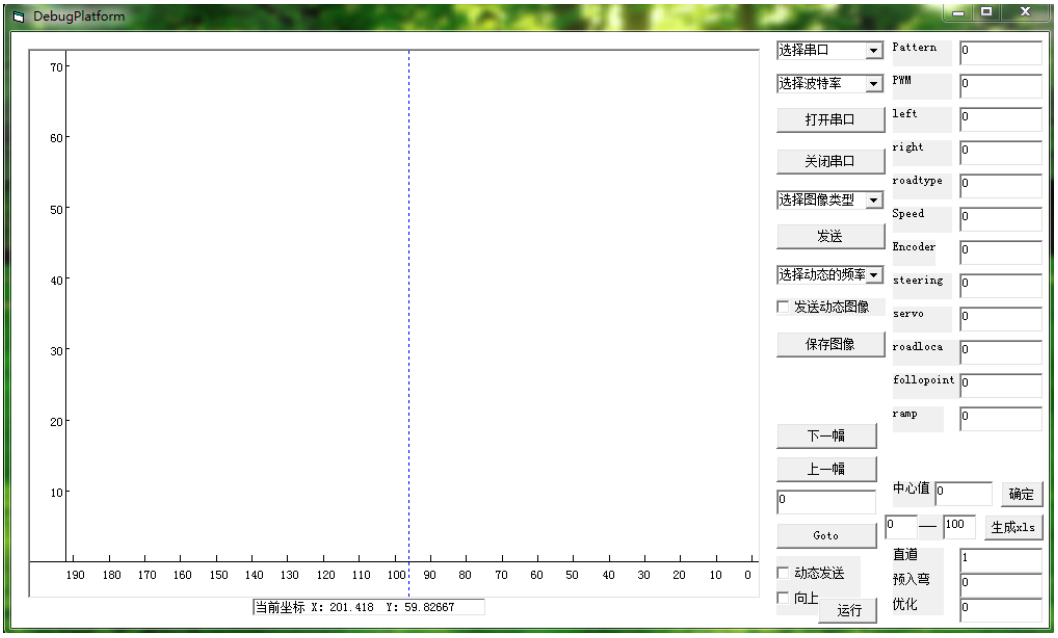
在cw 编译器中，集成了单片机下载和调试软件hiwave，在这个环境下，程序员可以使用BDM 对于单片机进行调试。使得单片机的调试极为方便，这种状况下， 我们可以随意的看到程序运行过程中的各个状态，极大的方便了调试。

5.2 调试平台的设计

为了顺利得到单片机实时运行的数据,并在调试的过程中方便的得到一些单片机运行信息,我们用VB语言,设计了针对智能车的上位机调试软件。 方便地对于数据进行采集、处理，同时，使用无线串口实现无线传输，方便地 进行远程控制。

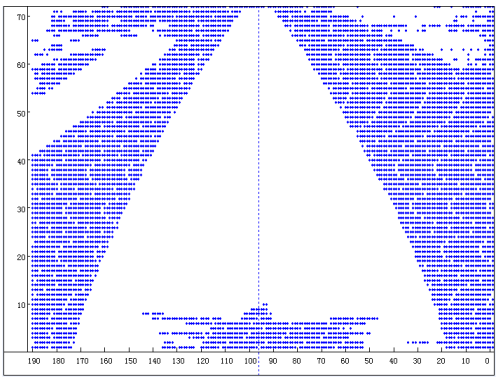
调试平台主要由以下几部分构成:图像调试模块,速度监测模块、数据传输 设置模块。

5.2.1 图像采集模块

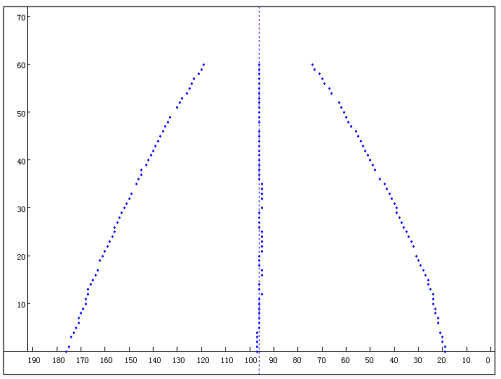


图表 5-3 图像采集模块

这个模块主要实现了对于单片机采集到的图像进行显示，其中包括一些附加功能包括舵机、电机的测试。以及单片机通讯测试等。同时，该模块能够实现 图像和数据的动态显示，一般状况下，能够使用串口对图像进行传输。



图表 5-4 完整图像

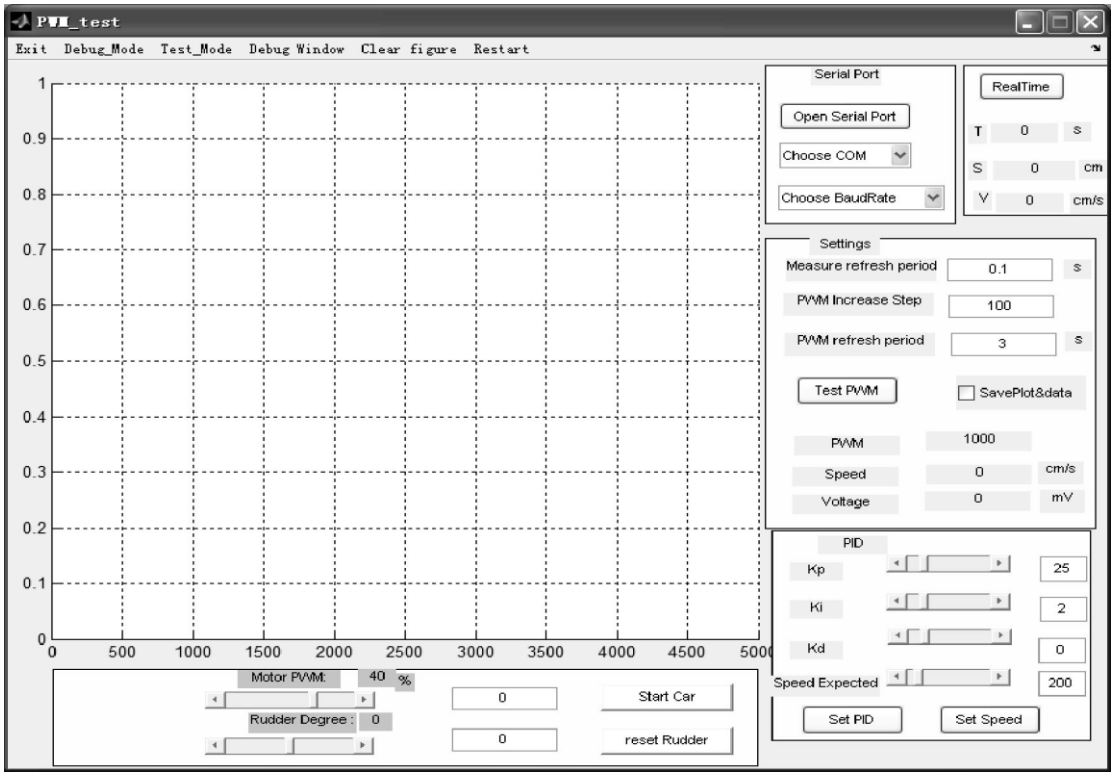


图表 5-5 压缩后图像

5.2.2 速度监测模块

速度监测模块主要实现了对于速度的实时监测，用来测试 PID 参数，以及小车的动态性能，速度监测模块极大的方便了小车在速度方面的调试过程。

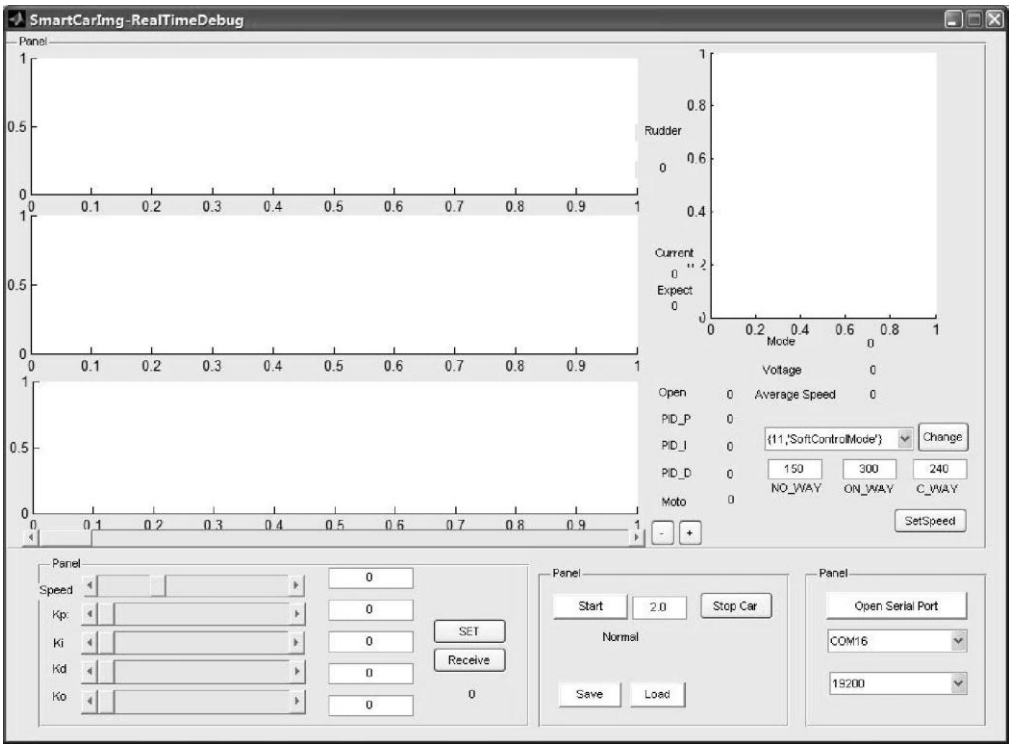
为了测试速度和电压的关系，在速度监视的同时，能够同时对小车的电池电压进行检测。



图表 5-6 速度检测模块

5.2.3 数据传输设置模块

在调试的过程中,难免有一些参数,不能在静态的时候依靠单片机BDM 功能，因此，我们设计出了这个模块，发送和接收单片机的相关状态数据， 在调试中，根据这些状态数据，得到相应的控制策略。



图表 5-6　数据传输设置模块

在这个模块中，可以设定将一些需要显示的数据以曲线的方式显示出来，直观地看到参数的变化规律。

第 6 章　全文总结

6.1　技术参数说明

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）	320*210*300
车模轴距/轮距（毫米）	轴距 205/　前 141/后 143
车模平均电流（匀速行驶）（毫安）	2600
电路电容总量（微法）	1728
传感器种类及个数	摄像头 编码器各一个
新增加伺服电机个数	0
赛道信息检测空间精度（毫米）	3
赛道信息检测频率（次/秒）	50
主要集成电路种类/数量	电机驱动/2，摄像头处理/11
车模重量（带有电池）（千克）	0.98

图表6. 1

6.2　存在的问题和改进

经过近半年的智能车的设计，制作、调试，在队员的分工和配合下，最终顺利完成了智能车电路、机械、程序等方面的工作。在这个过程中，学到了很多，积累了很多，能够做到最好，靠的是整个团队的协作，但是在整个制作过程中，也出现了一些问题：

（1）硬件设计的功能完整性，在电路设计的过程中，端口分配是一个很重要的问题，设计时没有将无用的IO口留出，造成之后需要扩展时扩展困难。

（2）在程序设计的过程中，过于模糊化，没有进行实际的建模和仿真，因此，只能从量的角度看到车跑的好不好，而没有从质的角度理解。

（3）在准备过程中，对于其他学校的参考和学习是很有必要的。

6.3 致谢

作为一个融合多学科交叉的的复杂系统，完成整智能车的设计、制作和调试是一个非常庞大的工程，仅靠几名队员是很难完成的，它需要一个高效运作、规范管理的团队的紧密合作才能完成。

感谢指导老师们的指导、支持和帮助。没有他们的指导和帮助，整个智能车队就很难高效稳定运行下去，设计的方向也很难把握得这么准确。

感谢华中科技大学启明学院和控制科学与工程系在调试场地,物资上的支持和帮助，正是有了学校的大力支持，才使我们能专心做事，仔细研究。

还要感谢教育部,自动化分教指委给大家提供的这个锻炼团队合作和创新能力的竞赛平台，以及清华大学和本届竞赛中给大家提供赛场的湖南大学、杭州电子科技大学以及其他赛区承办学校，感谢各个院校为大家提供优良的竞赛环境，使大家能够在竞赛中发挥出高水平。

整个比赛的过程，磨练了每个队员的意志，提高了队员的素质，总体而言，我们学到了很多，提高了很多，这将对以后的学习和工作起到很到的作用。

相信技术永无止境，创新永无止境！

参考文献

[1] 邵贝贝. 单片机嵌入式应用的在线开发方法[M]. 北京. 清华大学出版社. 2004

[2] 沈长生. 常用电子元器件使用一读通[M]. 北京. 人民邮电出版社. 2004

[3] 安鹏，马伟. S12单片机模块应用及程序调试[J]. 电子产品世界. 2006. 第211期

[4] 卓晴 黄开胜 .学做智能车. 北京：北京航空航天出版社，2007年3月第1版.

附录A 源代码

```
/*****************************************************
*File Name:              Arithmetic.c
*01a 2012-2-23 16:03:08
*02a 2012-7-13 10:24:44
*****************************************************************************
***/
#include "Arithmetic.h"

/*    global parameter declare */
volatile int gl_racePattern =
STRAIGHT_RACE; volatile unsigned char
gl_imageCenter = 81; volatile int
gl_protectSteeringAngle = 0;


/*=============================================================
=== =======

   *    Function
=*/ ============================================================
=======

/*-----------------------------------------------------------------
  *    Function:
  *    Return :    none
  *    Modification History
  *    01a 2012-2-23 16:32:32
-----------------------------------------------------------------*/
void CoreControl(void)
{
    static unsigned long OptimizeDistance = 0;
    int StraightFlag = 0;
    int CurveFlag = 0;
    unsigned char OptimizePoint = 0;
    int preOptimizePointY[15] = {0};
/*****************************************************************
***/
    static int PlaceConuter = 0 ;
static unsigned char LastEndY = 0 ;
    int again = 1;
    unsigned char end_y = 0;
    int D_X = 0;
```

```
        int table_y = 0,table_x = 0;
    /*****************************************************************
    ***/
        if(!IFlag.NoWay)
        {
            FindRealRoad();
            StraightFlag = CheckStraightRace();                    //StraightFlag == 0 :
not StraightRace
//StraightFlag == 1 : StraightRace
        }
    switch(gl_racePattern)
        {
    /*-------------------------------------                      STRAIGHT_RACE
    -------------------------------------*/
                    case
            STRAIGHT_RACE:

                if(!IFlag.NoWay)
                {
                    if(!StraightFlag)
                    {
                        PreCurveFlag = CheckPreCurveRace();
                        if(PreCurveFlag != NOTTURN)
                        {
                            if(PreCurveFlag == TURNLEFT)
                            {
                                gl_racePattern      =      PRE_LEFTCURVE;
                                CtrlOnStraightToLeft();
                            }                                                  //Contro
                            else                                               l
                            {


                            }
                        }   gl_racePattern      =      PRE_RIGHTCURVE;
                        else    CtrlOnStraightToRight();
                        {                                                      //Contro
                                                                               l
                            OptimizeFlag = CheckOptimizeRace();
                            if(OptimizeFla
                                  g)
                            {
                                gl_racePattern                  =
                                OPTIMIZE_RACE;
                                OptimizeDistance                =
                                CtrlOnOptimize();
//Contro                          gl_steeringAngle        =       -
l                               gl_servoAngle;
```

Control
HeaderO
ptimiz();

```
else                                    //unknown pattern
{                                       //Control

                        }
                    }//if(PreCurveFlag != NOTTURN)
                }
                else                                        //if it's Straight race
                {                                           //then the pattern
not
changed             CtrlOnStraight();
                                                            //Contro
                    l gl_servoAngle = 0;
                }
            }
            else //protect Control
            {
                ProtectCtrl();
            }
            break;
    /*-----------------------------------                    PRE_LEFTCURVE
    ---------------------------------------*/
                case
            PRE_LEFTCURVE:

            if(!IFlag.NoWay)
            {
                if(!StraightFlag)
                {
                    RepairRoad();
                    OptimizeFlag = CheckOptimizeRace();
                    if(OptimizeFlag)                    //if it's not Optimize
race
                    {                                       //then      the
not
changed                 pattern gl_racePattern = OPTIMIZE_RACE;

                        OptimizeDistance            =
                        gl_encoderTotal;
                        CtrlOnOptimize();
//Control                 gl_steeringAngle = - gl_servoAngle;

                        ControlHeaderOptimize();
                        don't change the pattern == 1 :

got to the LEFTCURVE pattern
```

```
                    }//if(CurveFlag == LEFTRACE)
                }//if(OptimizeFlag)
            }
            else
            {
                gl_racePattern = STRAIGHT_RACE;
                CtrlOnStraight();
                                                        //Contro
            }   l gl_servoAngle = 0;
        }
        else //protect Control
        {
            ProtectCtrl();
        }
        break;
/*-------------------------------------                PRE_RIGHTCURVE
---------------------------------------*/

            case
    PRE_RIGHTCURVE:

        if(!IFlag.NoWay)
        {
            RepairRoad();
            if(!StraightFlag)
            {
                OptimizeFlag = CheckOptimizeRace();
                if(OptimizeFlag)
                {
```

```
//Control


 don't change the pattern
gl_racePattern = OPTIMIZE_RACE; OptimizeDistance =
gl_encoderTotal; CtrlOnOptimize();
gl_steeringAngle = - gl_servoAngle;

ControlHeaderOptimize();



CurveFlag = CheckCurveRace();
if(CurveFlag == RIGHTRACE)          //RightFlag == 0 :
```



```
                                    CtrlOnStraightToRight();        //if it's not Right
then the pattern not                race,
changed
                                }//if(CurveFlag == RIGHTRACE)

                            }//if(OptimizeFlag)
                        }
                        else
                        {   gl_racePattern                  =
                            STRAIGHT_RACE;
                            CtrlOnStraight();
                            gl_servoAngle = 0;                      //Control
                        }//if(!StraightFlag)
                    }
                    else //protect Control
                    {
                        ProtectCtrl();
                    }
                    break;
    /*-----------------------------------                     IN_LEFTCURVE
    -----------------------------------*/
                        case
                IN_LEFTCURVE:
                    if(!IFlag.NoWay)
                    {
                        if(!StraightFlag)
```

```
{
    RepairRoad();
    OptimizeFlag = CheckOptimizeRace();
    if(OptimizeFlag)
    {
```

```
 //Control
gl_racePattern = OPTIMIZE_RACE; OptimizeDistance =
gl_encoderTotal; CtrlOnOptimize();
gl_steeringAngle = - gl_servoAngle;
ControlHeaderOptimize();
```

```
PreCurveFlag =
```

```
                    }
                    else
                    {
                        CurveFlag = CheckCurveRace();
                        if(CurveFlag == RIGHTRACE)
//RightFlag == 0 : don't change the pattern

{                                   //RightFlag == 2 : got to the
RIGHTCURVE
pattern
                                gl_racePattern          =
                                IN_RIGHTCURVE;
                        }       CtrlOnRightCurve();
                        else
                        {

                                CtrlOnLeftCurve();      //if it's not
preCurve race, then the pattern not changed
                        }
                    }//if(PreCurveFlag       ==
                    TURNLEFT)
                }
                else }//if(OptimizeFlag)
                {

                    gl_racePattern          =
                    STRAIGHT_RACE;              //Control
                    CtrlOnStraight();
                    gl_servoAngle = 0;
                }//if(!StraightFlag)
            }
            else //protect Control
            {
                gl_steeringAngle            =
                LEFT_MAX_ANGLE; ProtectCtrl();
            }//if(!IFlag.NoWay)
            break;
    /*-----------------------------------          IN_RIGHTCURVE
    ---------------------------------------*/
                case
            IN_RIGHTCURVE:

            if(!IFlag.NoWay)
            {
                RepairRoad();
                if(!StraightFlag)
                {
                    OptimizeFlag = CheckOptimizeRace();
```

```
if(OptimizeFlag)
```

```
                    }
                    else
                    {
                        PreCurveFlag = CheckPreCurveRace();

                        if(PreCurveFlag == TURNRIGHT)
                        {
                            gl_racePattern = PRE_RIGHTCURVE;
                            CtrlOnStraightToRight();

                        }
                        else if(PreCurveFlag == TURNLEFT)
                        {
                            gl_racePattern                    =
                            PRE_LEFTCURVE;
                            CtrlOnStraightToLeft();           //Control
                        }
                        else
                        {

                            CurveFlag = CheckCurveRace();
                            if(CurveFlag == LEFTRACE)
                            {
                                gl_racePattern                =
                                IN_LEFTCURVE;
                                CtrlOnLeftCurve();
                            }   gl_servoAngle = 0;
                            else
                            {

                                CtrlOnRightCurve();           //if it's
not preCurve race, then the pattern not changed
                                gl_servoAngle =
                                    0;
                            }
                        }//if(PreCurveFlag == TURNRIGHT)
                    }//if(OptimizeFla
                }       g)
                else
                {


//Contro                gl_racePattern                =
l                     STRAIGHT_RACE;
                      CtrlOnStraight();

                      gl_servoAngle = 0;

                  }//if(!StraightFlag)
              }
              else //protect Control
```

```
        break;
    /*-------------------------------------                    OPTIMIZE_RACE
    -------------------------------------*/
                    case
              OPTIMIZE_RACE:
                OptimizePoint = GetOptimizePoint();
                if(!IFlag.NoWay)
                {
                    if(!StraightFlag)
                    {
                        RepairRoad();

                        OptimizeFlag = CheckOptimizeRace();
                        if(OptimizeFlag)
                        {
                            OptimizeDistance                =
                            gl_encoderTotal;
                            gl_steeringAngle                =
                        }
                        else gl_servoAngle;
                        {   ControlHeaderOptimize();



                            if(gl_encoderTotal - OptimizeDistance > 15)
                            {
                                CurveFlag = CheckCurveRace();
                                if(CurveFlag == RIGHTRACE)
//RightFlag == 0 : don't change the pattern

{                                           //RightFlag    ==    2    :    got    to    the
RIGHTCURVE
pattern
                                    gl_racePattern                  =
                                    IN_RIGHTCURVE;
                                    CtrlOnRightCurve();
                                    gl_servoAngle =
                                            0;
                                }
                                else if(CurveFlag == LEFTRACE)
                                {
```

```
                }
                else
                {
                    gl_racePattern                    =
                    STRAIGHT_RACE;
                    CtrlOnStraight();
                    gl_servoAngle = 0;                              //Control
                }
            }
            else //protect Control
            {
                gl_steeringAngle                =
                gl_servoAngle; ProtectCtrl();
            }
            break;
/*-------------------------------------                          UNKWON_PATTERN
-------------------------------------*/
            default:
            break;
    }
      if(gl_racePattern != OPTIMIZE_RACE)
    {
        gl_servoAngle = 0;
    }

    ProtectImage();
}
/*---------------------------------------------------------------------
  *    Function:   ProtectImage
  *    Return  :    none
  *    Notes    :     if in 1s, 80% image are Noway, STOP
  *    Modification History
  *    01a 2012- 4-21 20:05:41
---------------------------------------------------------------------*/
void ProtectImage(void)
{
    static unsigned char Noway = 0;
    static unsigned char Counter = 0;

    if(IFlag.NoWay == 1)
    {

    }
    else Counter++;
    {

    }
```

```
            if(Counter >= 50)
        {
            Counter = 0;
            if(Noway >= 20)
            {
                gl_car_Mode = STOP_MODE;
            }
        }
    }

/*****************************************************************************
    *    Control.c-----
    *          (c)Copyright 2012 HUST IA Studio 2012
    *                    All Rights Reserved
    *
    *File Name:              Control.c
    *Purpose:
    *Programmer(s):
    *Modification History
    *01a 2012-7-23 16:03:08 Written
    ******************************************************************************
    ***/
    #include "Control.h"

    const  int  tableSin[77]
    = {                             0,2,3,5,7,9,10,12,14,16,
                                    17,19,21,22,24,26,28,29,31,33,
                                    34,36,37,39,41,42,44,45,47,48,
                                    50,52,53,54,56,57,59,60,62,63,
                                    64,66,67,68,69,71,72,73,74,75,
                                    77,78,79,80,81,82,83,84,85,86,
                                    87,87,88,89,90,91,91,92,93,93,
                                    94,95,95,96,96,97,97
                                  };
    const int tableBodyWidth[72] = {42,42,41,41,40,40,40,39,39,38,
                                        38,38,37,37,36,36,36,35,35,34,
                                        34,34,33,33,32,32,32,31,31,31,
                                        30,30,29,29,29,28,28,27,27,27,
                                        26,26,25,25,25,24,24,23,23,23,
                                        22,22,21,21,21,20,20,19,19,19,
                                        18,18,17,17,17,16,16,15,15,15,
                                        14,14};

    const int tableRoadSideWidth[72] = {107,106,105,104,103,102,101,100,98,97,
                                            96,95,94,93,92,91,90,89,88,87,
                                            86,85,83,82,81,80,79,78,77,76,
                                            75,74,73,72,71,70,69,67,66,65,
```

```
                                        54,52,51,50,49,48,47,46,45,44,
                                    43,42,41,40,39,37,36,35,34,33,
                                    32,31};

    /*    global parameter declare */

    /*=========================================================
=== =======

      *    Function
=*/ ===========================================================
    =======

    /*-------------------------------------------------------------------
      *    Function:    control car on straight way
      *    Return  :     none
      *    Notes    :
      *    Modification History
      *    01a 2012-4-18 10:38:23 Written
----------------------------------------------------------------------*/
    void CtrlOnStraight(void)
    {
        static int Err0 = 0, Err1 = 0 ,Derro = 0,Da = 3,LastD = 0;
        int AngleX = 0,AngleY = 0;
        int ImageCenter = 0,RealRoadEndy = 0,RealRoadEndx = 0,RealRoadStry =
0,RealRoadStrx = 0;
        int DeathZoomFlag = 0,lowSpeedFlag = 0;
        int Kp = 0,Kd = 0;
        int nowP = 0,nowD = 0;
        int StraigtAngle = 0,FarestAngle = 0;
        unsigned char RoadEnd = 0,RoadStart = 0,i = 0;

        RoadEnd   =
        Road_End_Y;   RoadStart
        = Road_Start_Y;

        ImageCenter =            (int)gl_imageCenter;

        RealRoadEndx                              =

        (int)realRoad[RoadEnd].CenterX;
        RealRoadEndy = (int)realRoad[RoadEnd].CenterY;

        RealRoadStrx                              =
        (int)realRoad[RoadStart].CenterX;
        RealRoadStry                              =
        (int)realRoad[RoadStart].CenterY;
        Death
        Zoom
```

```
        nowP = Kp * Err0;
        nowD = (Kd * Derro * (10 - Da) + Da * LastD) / 10;
        LastD = nowD;
        StraigtAngle = (nowP + nowD) / 70;


        gl_steeringAngle = StraigtAngle ;
        if(gl_curSpeed < 400)
        {
            gl_speed = 80;
        }
        else
        {
            gl_speed = 60;
        }
    }

    void CtrlOnUnknownRace(void)
    {
        int AngleX = 0,AngleY = 0;
        int ImageCenter = 0,RealRoadEndy = 0,RealRoadEndx = 0,RealRoadStry =
0,RealRoadStrx = 0;
        int BasicAngle = 0;
        unsigned char RoadEnd = 0,RoadStart = 0,i = 0;
        int RoadStartX = 0;

        RoadEnd   =
        Road_End_Y;   RoadStart
        =
        Road_Start_Y;RoadStart
        X = (int)Road[RoadStart];
        ImageCenter    =
        (int)gl_imageCenter;
        RealRoadEndx                                =
        (int)realRoad[RoadEnd].CenterX;
        RealRoadEndy                                =
        (int)realRoad[RoadEnd].CenterY;

        RealRoadStrx                                =
        (int)realRoad[RoadStart].CenterX;
        RealRoadStry                                =
        (int)realRoad[RoadStart].CenterY;

        AngleX      =      RealRoadStrx      -
        RealRoadEndx;        AngleY        =
        RealRoadEndy - RealRoadStry;

        BasicAngle = MyArcTan(AngleX,AngleY);

        gl_steeringAngle = BasicAngle;
```

```
    *    Return  :     none
    *    Notes    :
    *    Modification History
    *    01a 2012-4-18 10:38:23 Written
    ------------------------------------------------------------------*/
    void CtrlOnRightCurve(void)
    {
        static int Err0 = 0, Err1 = 0 ,Derro = 0,Da = 2,LastD = 0;
        int AngleX = 0,AngleY = 0;
        int ImageCenter = 0,RealRoadEndy = 0,RealRoadEndx = 0,RealRoadStry =
0,RealRoadStrx =
        0;
        int Kp = 0,Kd = 0;
        int nowP = 0,nowD = 0;
        int BasicAngle = 0,DeltaAngle = 0;
        unsigned char RoadEnd = 0,RoadStart = 0,i = 0;
        int RoadStartX = 0;
        RoadEnd   =
        Road_End_Y;   RoadStart
        = Road_Start_Y;

        RoadStartX                    =

        (int)Road[RoadStart];

        mageCenter  =

        (int)gl_imageCenter;

        RealRoadEndx                          =
        (int)realRoad[RoadEnd].CenterX;
        RealRoadEndy                          =
        (int)realRoad[RoadEnd].CenterY;

        RealRoadStrx                          =
        (int)realRoad[RoadStart].CenterX;
        RealRoadStry                          =
        (int)realRoad[RoadStart].CenterY;

        Err1 = Err0;
        Err0    =    BEST_RIGHT_PX    -
        RoadStartX; Derro = Err0 - Err1;

       if(Road[RoadStart] > ImageCenter)
        {
            AngleX = RealRoadStrx - RealRoadEndx; AngleY =
```

```
            BasicAngle += DeltaAngle;
        }
        else
        {
            BasicAngle = RIGHT_MAX_ANGLE;
        }

        gl_steeringAngle = BasicAngle;

        if(gl_curSpeed < 300)
        {
            gl_speed = 70 - (BasicAngle >> 1);
        }
        else
        {
            gl_speed = 40;
        }
    }
    /*-----------------------------------------------------------------
      *    Function:    control car on left curve way
      *    Return  :     none
      *    Notes    :
      *    Modification History
      *    01a 2012-4-18 10:38:23 Written
    -----------------------------------------------------------------*/
    void CtrlOnLeftCurve(void)
    {
        static int Err0 = 0, Err1 = 0 ,Derro = 0,Da = 2,LastD = 0;
        int AngleX = 0,AngleY = 0;
        int ImageCenter = 0,RealRoadEndy = 0,RealRoadEndx = 0,RealRoadStry =
0,RealRoadStrx = 0;
        int Kp = 0,Kd = 0;
        int nowP = 0,nowD = 0;
        int BasicAngle = 0,DeltaAngle = 0;
        unsigned char RoadEnd = 0,RoadStart = 0;
        int RoadStartX = 0;

        RoadEnd   =
        Road_End_Y;   RoadStart
        = Road_Start_Y;

        RoadStartX       =       (int)Road[RoadStart];

        ImageCenter =             (int)gl_imageCenter;

        RealRoadEndx                                =

        (int)realRoad[RoadEnd].CenterX;
        RealRoadEndy = (int)realRoad[RoadEnd].CenterY;
```

```
        Err1 = Err0;
        Err0 = BEST_LEFT_PX - RoadStartX;
        Derro = Err0 - Err1;

       if(Road[RoadStart] < ImageCenter)
        {
            AngleX = RealRoadEndx -
            RealRoadStrx; AngleY =
            RealRoadEndy - RealRoadStry;
            BasicAngle = 0 - MyArcTan(AngleX,AngleY);
            Kp = 5;
            Kd =
            5;
            nowP = Kp * Err0;
            nowD = (Kd * Derro * (10 - Da) + Da * LastD)
            / 10; LastD = nowD;
            DeltaAngle = (nowP + nowD) / 10;
            BasicAngle += DeltaAngle;
        }
        else
        {
            BasicAngle = LEFT_MAX_ANGLE;
        }

        gl_steeringAngle = BasicAngle;

        if(gl_curSpeed < 300)
        {
            gl_speed = 70 + (BasicAngle >> 1);
        }
        else
        {
            gl_speed = 40;
        }
}
/*-----------------------------------------------------------------
   *    Function:   control car on optimize way
   *    Return  :    none
   *    Notes    :
   *    Modification History
   *    01a 2012-4-18 10:38:23
-----------------------------------------------------------------*/
/*void CtrlOnOptimize(void)
{
        int i = 0,t = 0,Futherp = 0;
        int AngleX = 0,AngleY  = 0 , error;
        int OptimizePointx = 0,OptimizePointy = 0,RealPoint[2] = {0,0};
```

```
int RoadTurnPointY[10] = {0};
int OptimizeAngle[10]  = {0};
int RoadEndy = 0,RoadStarty = 0,RoadEndx = 0;
int EndAngle = 0;
int BestAngle = 0;

RoadEndy  =  (int)Road_End_Y;
RoadEndx  =
(int)Road[RoadEndy];

RoadStarty = (int)Road_Start_Y; */

/* Caculate the Futher point Angle */
                        /*
  GetRealPoint(RoadEndx,RoadEndy,RealP
                    oint);

AngleX  =  (int)gl_imageCenter  -
RealPoint[0]; AngleY = RealPoint[1];

if(AngleX > 0)
{
    EndAngle = MyArcTan(AngleX,AngleY);
}
else
{
    AngleX = 0 - AngleX;
    EndAngle = 0 - MyArcTan(AngleX,AngleY);
}

FindTurnPointY(RoadTurnPointY);                    // Caculate the
Protect point Angle
while(RoadTurnPointY[i] != 255)
{
    OptimizePointy = RoadTurnPointY[0];
    CaculateMappingXErro(&error, RoadStarty, RoadEndy, OptimizePointy);
    if(error > 0)
    {
        OptimizePointx = (int)Road[OptimizePointy]
                        + tableRoadSideWidth[OptimizePointy] -
tableBodyWidth[OptimizePointy];
      // OptimizePointx = (int)Road[OptimizePointy] +
tableBodyWidth[OptimizePointy];
    else
```

```
            // OptimizePointx = (int)Road[OptimizePointy] - tableBodyWidth[OptimizePointy];
            }

            GetRealPoint(OptimizePointx,OptimizePointy,RealPoint);AngleX = (int)gl_imageCenter - RealPoint[0]; AngleY = RealPoint[1];
            if(AngleX > 0)
            {
                OptimizeAngle[i] = MyArcTan(AngleX,AngleY);
            }
            else
            {
                AngleX = 0 - AngleX;
                OptimizeAngle[i] = 0 - MyArcTan(AngleX,AngleY);
            }
            i++;
        }

        Futherp = i - 1; */

    /*  gl_steeringAngle = EndAngle;

        if(EndAngle > 0)
        {
            BestAngle = 255;
            for(i=0;i<Futherp;i++)
            {
                if(OptimizeAngle[i] < 0)
                {
                    OptimizeAngle[i] = 255;
                }
                else if(BestAngle > OptimizeAngle[i])
                {
                    BestAngle = OptimizeAngle[i];
                }
            }

            if(EndAngle < BestAngle)
            {
                gl_steeringAngle = EndAngle;
            }
            else
```

```
        {
            gl_steeringAngle = BestAngle;
        }
    }


    else if(EndAngle < 0)
    {
        BestAngle = -255;
        for(i=0;i<Futherp;i++)
        {
            if(OptimizeAngle[i] > 0)
            {
                OptimizeAngle[i] = 255;
            }
            else if(BestAngle < OptimizeAngle[i])
            {
                BestAngle = OptimizeAngle[i];
            }
        }

        if(EndAngle > BestAngle)
        {
            gl_steeringAngle = EndAngle;
        }
        else
        {
            gl_steeringAngle = BestAngle;
        }
    }
    else
    {
        gl_steeringAngle = EndAngle;
    }*/
/*  if(OptimizeAngle[0] > 0)
    {
        for(t=1;t<i;t++)
        {
             if((OptimizeAngle[i] < OptimizeAngle[0]) && (OptimizeAngle[i] >
            0))
            {
                gl_steeringAngle = OptimizeAngle[i];
            }
        }//for(t=1;t<i;t++)
    }
    else if(OptimizeAngle[0] < 0)
    {
        for(t=1;t<=i;t++)
```