第十一届“恩智浦”杯全国大学生
智能汽车竞赛
技 术 报 告

学　　校：　合肥工业大学
队伍名称：　庞门左道队
参赛队员：　庞　博
　　李秀琦
　　熊宇龙
带队教师：　张　阳
　　史久根

关于技术报告和研究论文使用授权的说明

本人完全了解第十一届“恩智浦”杯全国大学生智能汽车邀请赛关于保留、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，比赛组委会和恩智浦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：李秀琦 熊宇龙 成博

带队教师签名：

日　　期：2016.8.15

摘要

本文设计的智能车系统以 MK60FN1M0VLQ15 微控制器为核心控制单元，通过 OV7725 摄像头检测赛道信息，对图像进行硬件二值化，提取黑色引导线，用于赛道识别；通过编码器检测模型车的实时速度，使用 PID 控制算法调节驱动电机的转速和转向舵机的角度，实现了对模型车运动速度和运动方向的闭环控制。为了提高模型车的速度和稳定性，使用虚拟示波器、上位机、键盘模块等调试工具，进行了大量硬件与软件测试。实验结果表明，该系统设计方案确实可行。

关键字： MK60FN1M0VLQ15 PID OV7725

Abstract

In this paper we will design a smart car system based on MK60FN1M0VLQ15 as the micro-controller unit. We use a OV7725 image sensor to obtain lane image information. Then convert the original image into the binary image by the analog comparator circuit in order to extract black guide line for track identification. An inferred sensor is used to measure the car`s moving speed. We use PID control method to adjust the rotate speed of driving electromotor and direction of steering electromotor, to achieve the closed-loop control for the speed and direction. In order to increase the speed and the reliability of the car, a great number of the hardware and software tests are carried on and the advantages and disadvantages of the different schemes are compared by using the、MFC simulation platform and the keyboard module. The results indicate that our design scheme of the smart car system is feasible.

Keywords: MK60FN1M0VLQ15, PID, OV7725

第一章　引言

1.1　背景介绍

智能车是一种高新技术密集型的新型汽车，它涵盖的范围广泛包括模式识 别、传感器技术、自动化控制实现、电力电子技术、计算机技术等多个领域。 在国际上已经形成智能汽车研究、设计、开发、竞赛的热潮。

在我国，教育部为了加强大学生实践、创新能力和团队合作精神的培养，

委托教育部高等学校自动化专业教学指导分委员会主办了每年一度的全国大学生智能汽车竞赛。全国大学生智能汽车竞赛是在竞赛组委会提供的统一汽车模型平台上，使用恩智浦半导体公司的 8 位、16 位微控制器作为核心控制模块，通过设计道路识别传感器和电机驱动电路、编写相应软件及装配模型车，制作一个能够自主识别道路的模型汽车，按照规定路线（路线赛前未知）行进，以完成时间最短者为优胜。智能车竞赛目前已经发展有摄像头、光电组、电磁组、信标组、双车组、电轨组和节能组七个组别比赛，赛车速度与比赛质量也越来越高，竞争更是日趋激烈。

1.2　文献综述

在智能车制作过程中，参考了大量的相关资料文献。关于控制算法的文章很多，对于此次比赛，重点在于速度的提高，因此在阅读这些文献时，重点参考模糊控制和算法容易实现方面得内容。很多算法都是通过实验测试，最后得到了一个比较优越的速度控制方案。具体参考文献在文后参考文献中列出。

1.3　本文主要结构

本文针对第十一届“恩智浦”杯智能车比赛准备阶段的各个方面做了一个详细的总结。采用先总后分的结构，先对系统总体设计进行介绍，然后分别对各部分进行介绍，突出强调了系统机械设计、硬件电路设计和软件编程。

本文有 7 个章节，第一章为引言，简单介绍了智能车比赛的背景；第二到六章为主体部分，对机械结构、硬件和软件设计系统进行了详细介绍，并对调试方法和过程进行了详细说明。第七章为总结，最后是参考文献。

第二章　系统总体结构设计

根据大赛的统一要求，须在组委会统一提供的车模平台上，自主选择传感 器类型，设计系统硬件电路，开发软件算法。所以本系统主要包括三个大部分，分别为车模平台、硬件电路系统、软件算法。每一个部分又由各个小模块构成。所以要构建一个完整的小车系统，必须先对各个模块进行论证和设计，再将整个系统组合成一个完整系统，进行系统联调。

2.1 系统总体结构

从智能汽车设计规则及其自主寻迹运行的要求进行分析，可将智能汽车系统设计在结构上分为两个部分，即智能车车载系统设计和上位机系统设计。从智能车功能需求上分析，智能车车载系统设计需完成赛道信息采集、赛道元素判断、直流电机测速及转速控制、舵机转向控制、无线通信等基本功能。系统总体结构如图 2.1 所示

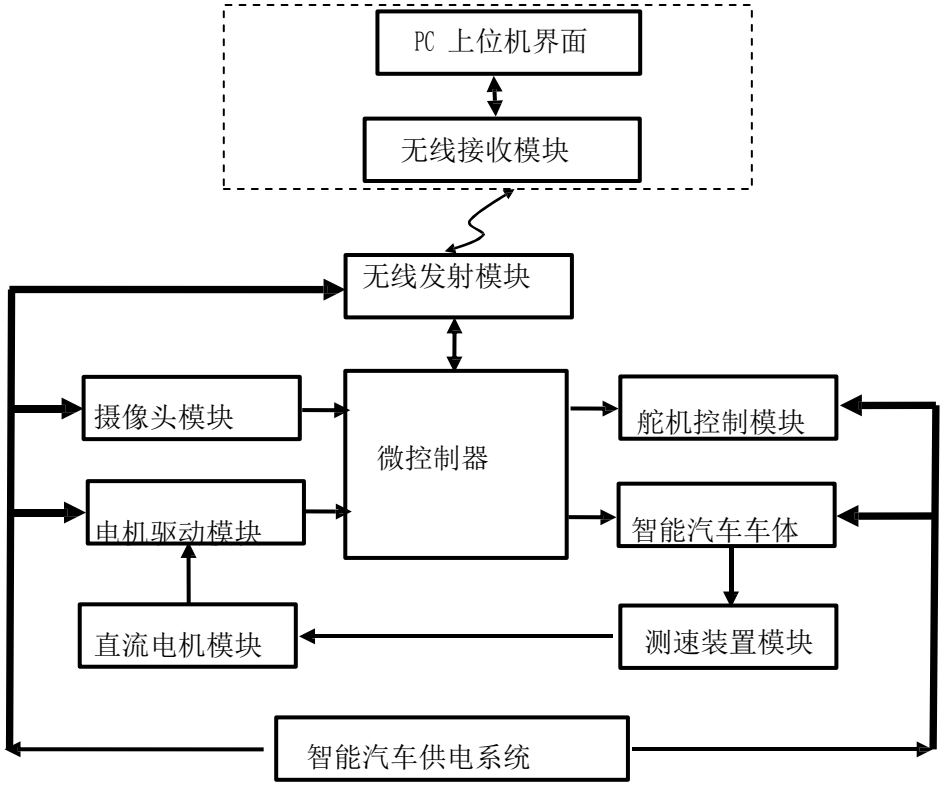


图 2.1 系统总体结构

2.2 智能车硬件电路总体设计

采用鹰眼摄像头进行道路识别，车模的硬件电路主要有七个部分组成：MK60FN1M0VLQ15 最小系统，图像采样处理模块，速度检测电路，电机驱动电路，舵机驱动模块，电源管理模块，辅助调试模块。

（1） MK60FN1M0VLQ15 最小系统是系统的核心部分，负责接收赛道图像数据，赛车 速度等反馈信息，并对这些信息进行恰当的处理，形成合适的控制量来对舵机 与驱动电机进行控制。

（2）图像采样处理模块由鹰眼摄像头组成，是智能小车的“视觉系统”，用于获得前方道路情况以供单片机处理。

（3）速度检测电路由欧姆龙旋转旋编码器组成，通过机械调整差速。

（4）电机驱动电路使用 HIP2104 与专用 MOS 管搭建的 H 桥全桥驱动，可以实现电机的正反转。

（5）舵机驱动模块控制舵机的转向。

（6）电源管理模块给整个系统供电，保障系统安全稳定运行。

（7）辅助调试模块有串行通信、蓝牙等，主要用于赛车系统的程序烧写，功能调试和测试，赛车状态监控，赛车系统参数和运行策略设置等方面。

2.3 智能车软件系统总体设计

系统硬件对于赛车来说是最基础的部分，软件算法则是赛车的核心部分。如果把一辆车和一个人做个类比的话，我们可以说，赛车的硬件结构相当于人的身体；赛车的软件算法相当于人的思想。只有“身体健康，思想进步”，才会取得好成绩。所以软件系统对于赛车来说至关重要。首先，赛车系统通过图像采样处理模块获取前方赛道的图像数据，同时通过速度传感器模块实时获取赛车的速度。然后利用边缘检测方法从图像数据中提取赛道中线，求得赛车位置与中线的偏差，接着采用 PD 算法对舵机进行反馈控制，最终赛车根据检测到的速度，结合我们的速度控制策略，对赛车速度不断进行恰当的控制调整，使赛车在符合比赛规则情况下沿赛道快速前进。

2.4 本章小结

本章主要介绍了赛车整体结构设计的概述。硬件、软件和机械部分的有效 融合是赛车跑出好成绩的关键因素。赛车采用组委会统一提供的车模，由主控 芯片 MK60FN1M0VLQ15 最小系统，图像采样模块，速度传感模块，舵机驱动模块，电

机驱动模块和辅助调试模块组成，通过图像采集、黑线提取、速度控制等环节 使赛车在规则下沿赛道快速前进。

第三章　车体机械结构设计

在小车的制作和调试过程中，随着车速的进一步提高，由机械引发的问题日益变得严重，经过理论分析和实践调试效果，得到了一套比较适合本小车的机械结构。本章主要介绍赛车车模的机械特点和调整方案。从理论方面进行论证，在结合具体实践成果，通过不同的方案效果对比，为读者提供一个全面的认识。

3.1 汽车行驶的数学模型

汽车是现代社会的主要交通工具，在对汽车研究过程中，形成了一大批研究成果。我们在查阅了一大堆资料的前提下，形成了自己对汽车原理的理解，首先建立汽车行驶的数学模型（见图 3.1）

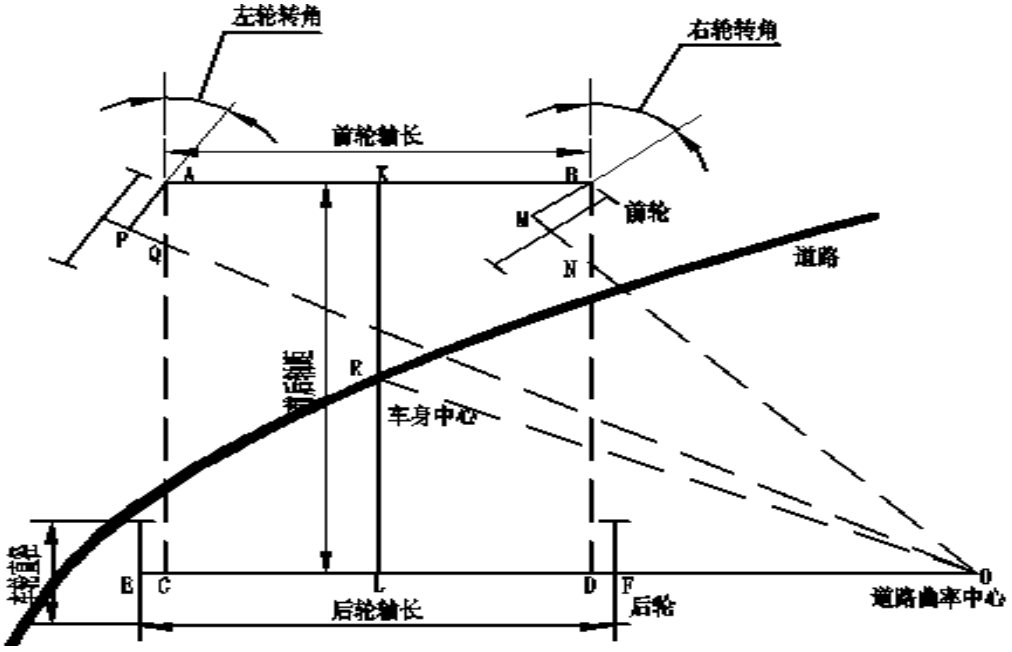


图 3.1 汽车行驶数学模型

采用第十一届“恩智浦”智能车大赛组委会提供的韩国 Matiz 系列 1：10 模型车的参数对公式（1），公式（2）进行仿真。得到车模行驶时理论转弯半径与车轮转角的关系（见图 2.2）和右轮也左轮的转角关系（见图 3.3）

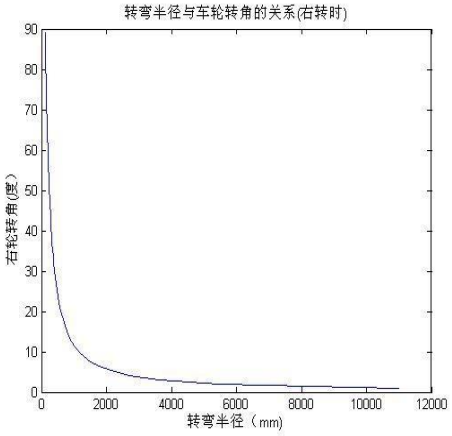


图 3.2 弯半径与车轮转角关系

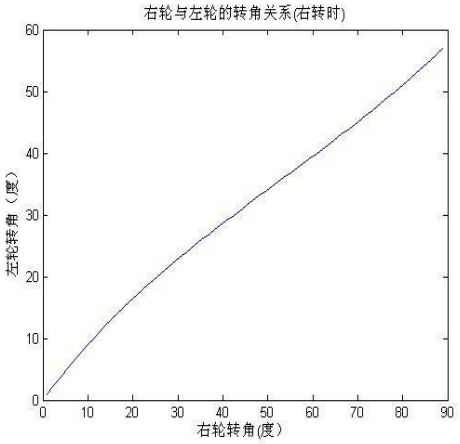


图 3.3 右、左轮的转角关系

在模型车结构参数一定的情况，小车左右两轮的转角存在一定的函数关系，当向右过弯时，右轮转向比左轮转向大，同理向左转弯时，左轮转向较右轮转向大，同时，随着道路曲率半径的越大，车轮所需的转角越小。在实际调试过程中，要以理论为基础，配合以上理论计算公式，寻找小车的轮速参数。

3.2 车模机械模型

官方给出的要求为使用 B 车模，车模如下图。传感器允许使用面阵 CCD 或者 CMOS 摄像头以及超声传感器进行赛道检测。禁止使用激光发射管；禁止使用线阵 CCD 摄像头。

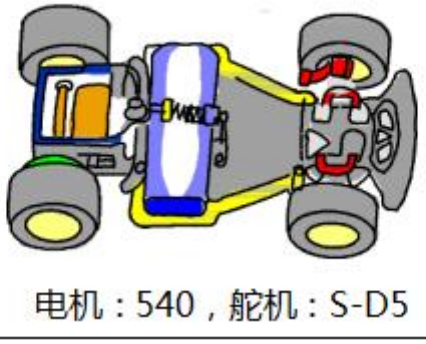


图 3.4 车模机械结构图

3．3　舵机的安装

舵机最早出现航模运动中，控制航模上的发动机，翼舵转向。在“恩智浦”智能汽车大赛中，主办方提供的舵机内部具有位置反馈电路，使它的舵盘输出转角正比于给其的脉冲信号的宽度。

舵机是具有较大延迟特性的器件，其延迟与其转角大小成正比，但如果能使舵机转过一个越小的角度而使车轮转过一个越大的角度，则会大大提高舵机过弯的响应速度。这不仅与舵机的安装方式有关，而且也与舵机输出臂的长度有关。

3.3.1 舵机的安装方式

对比各种舵机安装方式，我们决定采用最经典的输出臂朝下的输出方式，对于此种安装方式，同样以主销为 Z 轴，前轴为 Y 轴建立空间直角坐标系如图 3.5 所示。为了使小车左右转向一致，在舵机的安装中，一般将舵机前轮轴的中垂线上，并使舵机的转轴与 OA 杆同处于 XOY 平面上，且当模型车直线行驶时，输出臂 AB 垂直与 OA 杆，并使之保持水平。

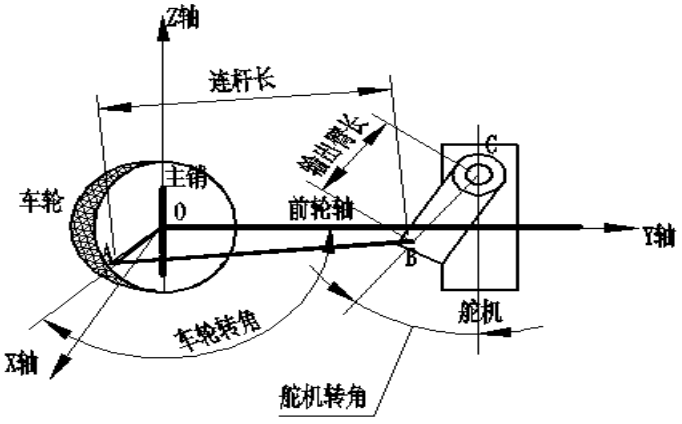


图 3.5　舵机安装方式关系图

根据理论计算和实际效果对比，得到最终舵机安装图，如图 3.6 所示。

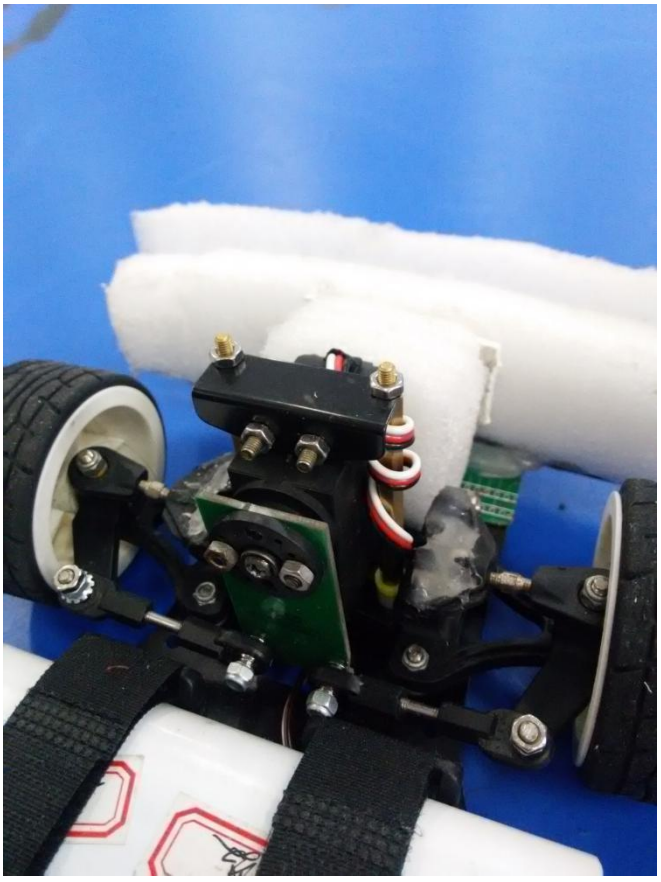


图 3.6 舵机的实际安装图

3.3.2 舵机臂的调整

由于舵机本身的滞后特性，对于诸如小车这样实时系统，必须尽量提高舵机的响应速度（及舵机的灵敏度）。舵机的灵敏度和舵机的安装方式有关，而且还与舵机臂的长短有关系，在安装方式一定的前提下。舵机输出臂越长，则舵机的灵活性越大，但是舵机的转矩会随着输出臂的增加而减小。在实际调整过程中，既要寻找最优的舵机长度，又要兼顾舵机的输出转矩。最后舵机臂的长短还应考虑到前轮的调整。

3.4 前轮调节

现代汽车在正常行驶的过程中，为了使汽车沿着直线行驶稳定，转向稳定、轻便，转向后能及时回正，并减少轮胎和转向系零件的磨损等，在转向轮，转向节和前轴之间须形成一定的相对安装位置，叫车轮定位，其主要参数包括：主销内倾，主销后倾，前轮前束，前轮外倾。智能汽车大赛提供的车模的四项前轮定位参数均可调，并给规格不同配件。

3.4.1 主销后倾

主销后倾是指在汽车的纵向平面内（汽车的侧面）有一个向后的倾角，即主销轴线与地面垂直线在汽车纵向平面内的夹角。采用主销后倾角的原因是由于汽车在车轮偏转后会产生一回正力矩，纠正车轮的偏转。所以主销后倾角越大，车速越高，前轮自动回正能力越强，但是过大的回正力矩会使车模转向沉重。经过调试，我们将主销后倾调节为如图 3.7 所示。

图 3.7 主销后倾角调整

3.4.2 主销内倾

主销内倾角是指在横向平面内主销轴线与地面垂直线之间的夹角，它的作用也是使前轮自动回正，角度越大前轮回正能力就越强，但转向时也就越费力，轮胎磨损增大；反之，角度越小前轮自动回正的能力就越弱。通常汽车的主销内倾角不大于 8 度，主销内倾的调整应保持在一个合适的范围，一般 0 度到 8 度范围内皆可。在实际的调整中，只要将角度调整为 5 度左右就会对于过弯性能有明显的改善。如果赛道比较滑，可以将这个角度再调节的大一些。

对于模型车，通过调整前桥的螺杆的长度可以改变主销内倾角的大小，由于过大的内倾角也会增大转向阻力，增加轮胎磨损，所以在调整时可以近似调整为 0 度到 3 度左右，不宜太大。主销内倾和主销后倾都有使车模转向自动回正，保持直线行驶的功能，不同之处是注销内请的回正与车速无关，主销后倾的回正与车速有关，因此高速时主销后倾的回正作用大，低速时主销内倾的回正作用大。

3.4.3 前轮外倾

前轮外倾角是指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，对汽车的转向性能有直接的影响，它的作用是提高前轮的转向安全性和转向操纵的轻便性。在汽车的横向平面内，轮胎呈“八”字形张开时称为“负外倾”，而呈现“V”字形张开时称为“正外倾”。如果车轮垂直地面一旦满载就易产生变形，可能引起车轮上部向内倾侧，导致车轮连接件的损坏。

所以事先将车轮校偏一个正外倾角度，一般这个角度约在 1° 左右，以减少承载轴承负荷，增加零件使用寿命，提高汽车的安全性能。前轮外倾角可以减少转向阻力，使汽车转向轻便。

3.4.4 前轮前束

当车轮有了外倾角后，在滚动时就类似与圆锥滚动，从而导致两侧车轮向外滚开。由于转向横拉杆和车桥的约束使车轮不可能向外滚开，车轮将在地面上出现边滚边向内滑移的现象，从而增加了轮胎的磨损。在安装车轮时候，为了消除车轮外倾带来的这种不良后果，可以使汽车两前轮的中心面不平行，并使两轮前边缘距离小于后面的边缘距离，两者之差称为“前轮前束”。模型车是由舵机带动左右横拉杆实现转向的。主销在垂直方向的位置确定后，改变左右横拉杆的长度即可改变前束的大小。

3.5 编码器的安装

编码器是测速用的传感器，必须将其和电机牢牢咬合才能准确测量实际速度值。我们选择 500 线欧姆龙编码器，将其安装在后轮前，也采用一个支架和车身固定在一起，这样便使得编码器能非常牢的和电机齿轮咬合，另外，为了更加牢固，我们又加入热熔胶进行加固。具体安装图如图 3.8 所示。

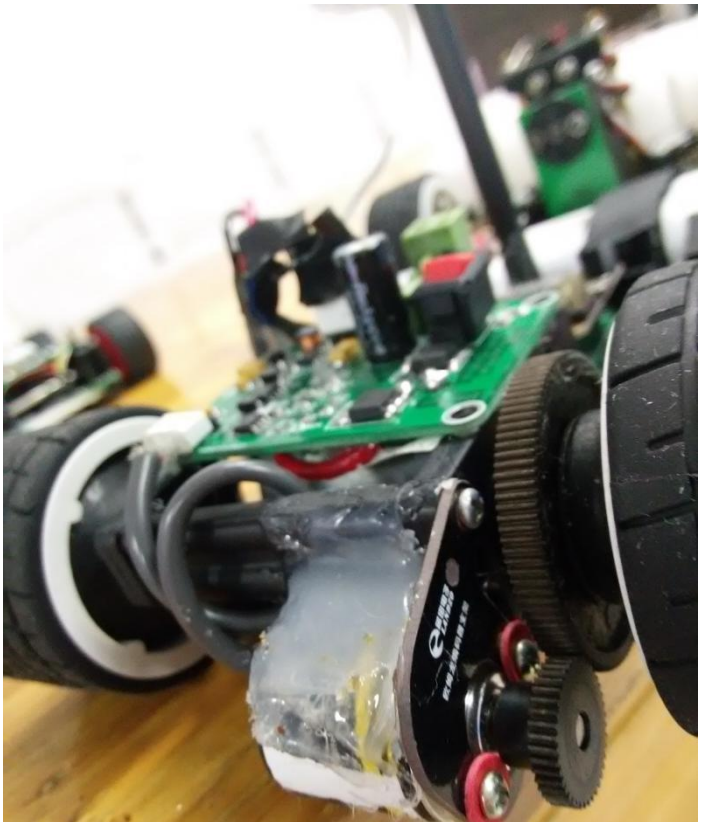


图 3.8　编码器的安装

3.6 图像传感器安装

信号采集传感器我们采用的是鹰眼摄像头。摄像头安装主要考虑三个方面，一是图像的失真要小，二是整车重心要低，三是前瞻要合适、盲区要小。

3.6.1 摄像头底座安装

在摄像头的安装位置方面，如果摄像头安装在车体太后面，将会导致图形畸变严重，以致给处理图像带来太多的复杂因素，但要是安装在车体太前面就会在保证一定的前瞻的情况下，导致小车车前盲区的过大，这样在弯道的识别控制时，是相当不利的，所以在摄像头位置的安装上，我们选择了距车体的中间位置进行固定安装，这样使得采集会的图像不至于畸变过于厉害，也使得小车的盲区控制在 7cm 左右。

3．6.2 摄像头支撑杆的选择

在摄像头支架杆上的选择有两种，一种是碳素杆非常轻，但很容易破裂，后一种采用轻质铝材，牢固耐用，但比起碳素杆还是重了许多。对比两种材质的支撑杆，结合本届车模的具体情况，决定采用碳素杆材料，从而减轻车模重量。

3．6.3 摄像头固定

要获得稳定可靠的信息，摄像头的固定必须相当牢固。在摄像头固定方面，其可以上下左右进行调节，在调节好视角之后，为了使其更加牢固采用了热熔胶塑胶进行二次定型。如图 3.9 所示。

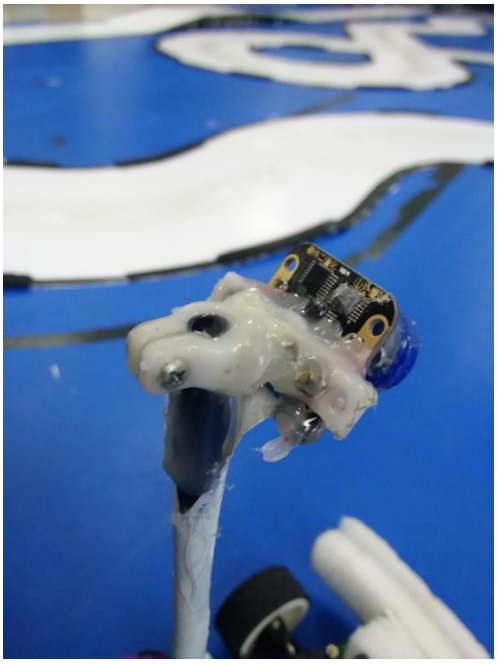


图 3.9　摄像头安装方式

3.7　整车效果图

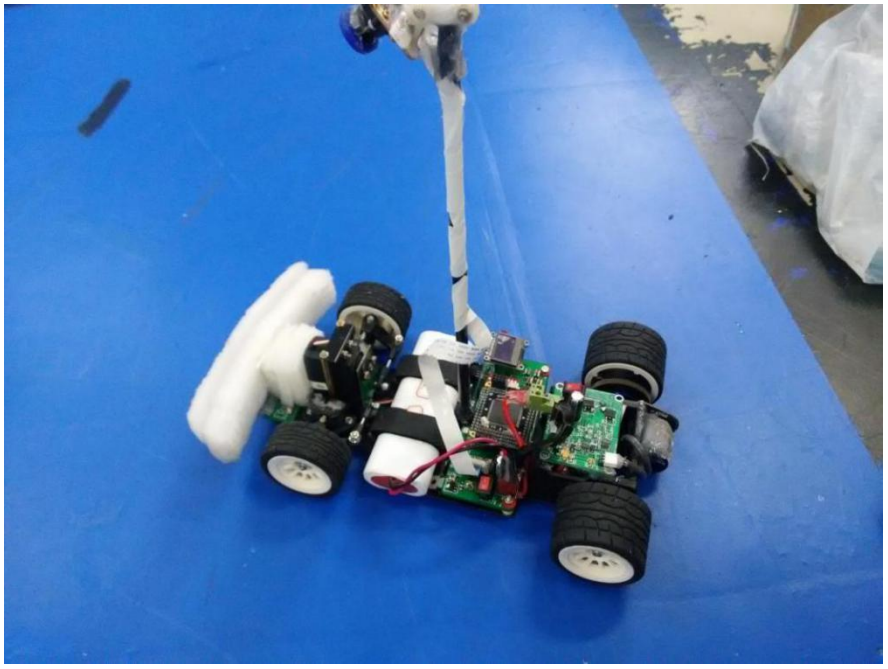


图 3.10 整车效果图

3.8　本章小结

本章主要介绍了小车的安装制作和调试过程中机械方面的问题。由于前期过于注重控制算法的研究，而没太注重整车的机械结构。所以随着调试的进一步进行，机械结构问题已经严重影响到了车速的进一步提高。特别是小车的过弯速度，即使速度很低，过弯也存在一些问题。后来在多次调整小车机械结构后，小车个过弯速度在同样的控制算法基础上都有了很到的改观。对了这样一个车载系统，要实现机械与控制的完美结合，才能达到最佳的速度效果。

第四章　硬件系统设计

4.1 硬件系统总体设计

系统的硬件电路是整个系统的基础，也是软件平台得以稳定运行的基础，所以硬件电路的设计是非常重要的。在硬件系统的设计过程中，本着系统安全、稳定的原则，电路设计尽量模块化，主要包括单片机最小系统、图像采集模块、电源管理模块、速度检测反馈模块、辅助调试模块（包含人机交互模块、蓝牙模块）、电机驱动模块等。（具体硬件系统框图见图 4.1）

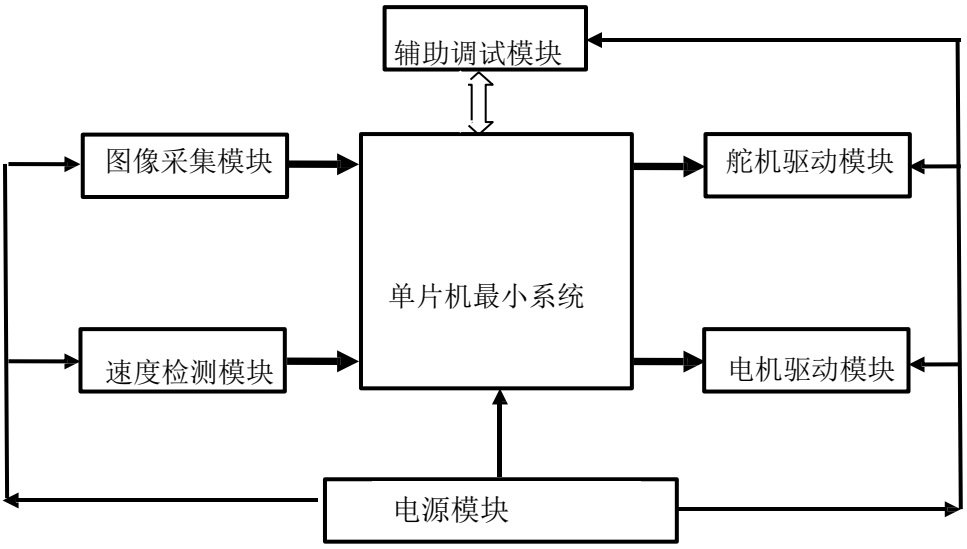


图 4.1 硬件系统框图

在本系统中，以恩智浦半导体公司的微控制器 MK60FN1M0VLQ15 做为主控制器。系统工作过程为：微控制器按照一定时序接收来自图像采集模块的摄像头图像信息，采集的同时进行解压缩、存储。每场采集结束后，对图像进一步处理，（即微控制器对存储的图像信息进行解码处理，提取出黑线的信息，并拟合中线）之后根据求出的偏差以位置式 PD 控制舵机打角，从而完成对智能车运行方向的调整。在图像采集处理的同时，微控制器根据此时测速装置反馈而回的电机实际转速大小和当前轨道引线状况下电机期望转速大小比较，采用 PID 控制器和 Bang-bang 控制器相结合，以控制直流电机转速来控制智能车自主寻迹运行速度。此外，通过智能车车载辅助调试模块，将智能车运行参数及采集的轨道引线信息，可以通过人机交互中的小液晶显示出来，或者以无线通信方式发送于上位机通信系统，

由上位机监控界面对智能车自主寻迹运行情况进行实时跟踪，方便调试。根据第十一届比赛的规则，硬件电路的电源统一由 7.2V、2A/h 的可充电镍镉电池提供。由于电路中的不同电路模块所需要的工作电压和电流容量各不相同，因此电源模块应该包括多个稳压电路，将充电电池电压转换成各个模块所需要的电压。

单片机的硬件资源是有限的，合理的利用单片机的硬件资源是保证系统稳定的前提。在本系统中，既涉及到传感器数据的获取，同时涉及到根据获取的传感器数据控制执行机构。在图像采集部分，使用的是中断捕捉方式进行，而在控制部分，主要使用的是单片机自带的 PWM 控制波。

4.2 单片机系统

单片机最小系统，采用大赛组委会统一提供的 MK60FN1M0VLQ15 单片机，16 位微处理器。为了节省成本，将最小系统直接画在主板上，该小系统工作稳定，只需要 3.3V 电源供电。MK60FN1M0VLQ15 最小系统是系统的核心部分，负责接收赛道图像数据，赛车速度等反馈信息，并对这些信息进行恰当的处理，在控制算法的控制下，形成合适的控制量来对舵机与驱动电机进行控制。从而控制整个小车的稳定、快速行驶。单片机小系统原理图如图 4.2 所示。

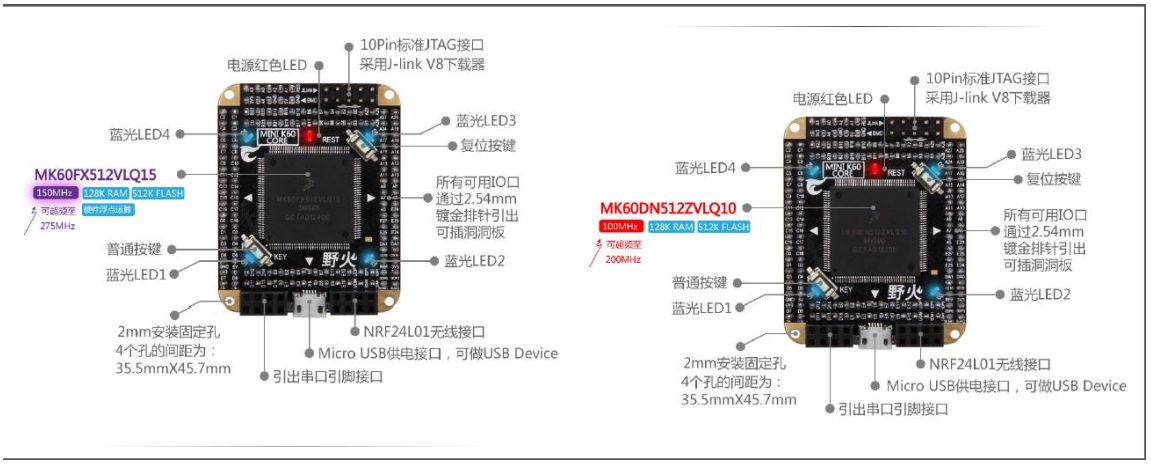


图 4.2 单片机最小系统原理图

4.3 图像采集模块

OV7725 摄像头的选择

CCD 摄像头具有对比度高、动态特性好的优点，但需要工作在 12V 电压下，对于整个系统来说过于耗电，而且 CCD 体积大，质量重，会抬高车体的重心，这对

于高速情况下小车的行驶非常不利。

与之相比，COMS 摄像头具有体积小、质量轻、功耗低，图像动态特性好等优点，因为小车队图像的清晰度，分辨率要求并不高，所以选用 COMS 摄像头。

对于摄像头的选择，主要考虑以下几个参数：

1 芯片大小

2 自动增益

3 分辨率

4 最小照度

5 信噪比

6 标准功率

7 扫描方式

其中芯片大小主要会对视场的范围会有影响，扫描方式主要有追行扫描以及隔行扫描，像 ov5116 就是隔行扫描。

市面上的摄像头主要分为数字和模拟两种，数字摄像头主要有 OV7620,OV6620，OV7670,OV7725,模拟摄像头主要有 OV5116，BF3003，MT9V136。大多数摄像头都支持 sccb 通信，可以很好的实现单片机与摄像头之间的交互。

智能车的摄像头对图像的分辨率要求并不高，但是对动态特性要求非常高，特别是小车在高速行驶入弯或者出弯的时候，图像变化较大，这就对摄像头的自动增益有较高的要求。一般来说，在摄像头图像发生突变时，感光芯片本身会有一段适应时间，这段时间要求越小越好。

OV5116，BF3003 比较

OV5116 具有成像稳定，技术成熟等优点，但是同时也有供货不稳定（已停产），图像动态特性差等缺点。于是我们选用了新的摄像头 bf3003。经试验证明，与 ov5116 相比，bf3003 具有动态特性好，反应快，曝光时间短等优势，且可以进行 sccb 通信，可以实现动态调节，但是 BF3003 功耗大，对电源芯片要求较高，复合信号纹波较大，稳定性不如 OV5116。

OV5116 是 Omni Vision 公司生产的较为典型的 CMOS 图像传感器模块，芯片阵列大小为 352×288，有效光敏面为 312×215 像素，电源是 5V(DC)，28 个引脚的 PLCC 型封装。摄像头输出的黑白全电视信号为 PAL 制式模拟信号，每秒 25 帧，电视扫描线为 625 线，奇场在前，偶场在后。

我们的智能模型车自动控制系统中使用野火鹰眼 OV7725 摄像头采集赛道信息。

ov7725 信噪比高、速度快、稳定性好和微光灵敏度高，其硬件二值化效果非常好。由于 K60 抗干扰能力比较弱，因此需在 PCLK 和 VSYNC 信号线接下拉 150 欧电阻增强电路的抗干扰能力。

综上所述，我们最终选择了 OV7725 摄像头。

4.4 速度反馈模块电路设计

由于对车模采用的是闭环控制系统，所以必然涉及到反馈量的检测。在车模行驶过程中，最重要的一个反馈量就是车模的行驶速度。通过采用欧姆龙 500 线编码器直接测量车轮的速度。

光电式旋转编码器通过光电转换，可将输出轴的角位移、角速度等机械量转换成相应的电脉冲以数字量输出（REP）。它分为单路输出和双路输出两种。技术参数主要有每转脉冲数（几十个到几千个都有），和供电电压等。单路输出是指旋转编码器的输出是一组脉冲。

4.5 舵机供电电源设计

在模型飞机和四轮模型车中，舵机是必备的部分，舵机的基本工作原理为内部包括了一个小型直流马达，一组变速齿轮组，一个反馈可调电位器及一块电子控制板。其中，高速转动的直流马达提供了原始动力，带动变速（减速）齿轮组，使之产生高扭力的输出，齿轮组的变速比愈大，伺服马达的输出扭力也愈大，也就是说越能承受更大的重量，但转动的速度也愈低减速齿轮组由马达驱动，其终端（输出端）带动一个线性的比例电位器作位置检测，该电位器把转角坐标转换为一比例电压反馈给控制线路板，控制线路板将其与输入的控制脉冲信号比较，产生纠正脉冲，并驱动马达正向或反向地转动，使齿轮组的输出位置与期望值相符，令纠正脉冲趋于为 0，从而达到使伺服马达精确定位的目的。

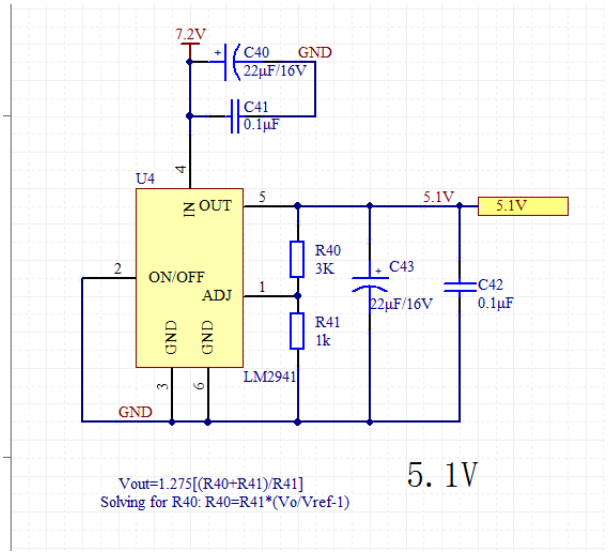


图 4.3 舵机供电电源电路图

B 车配置的转向舵机采用的是型号为 S-D5 的数字舵机，舵机的供电电压变化范围为 5V 左右，为了使舵机有更快的响应速度，采用 5.1V 的供电电压。

4.6 驱动模块设计

小车的驱动模块是小车的动力之源。我们采用 IRLR7843 搭建的 H 桥全桥电来驱动电机，可以控制电机的正反转。

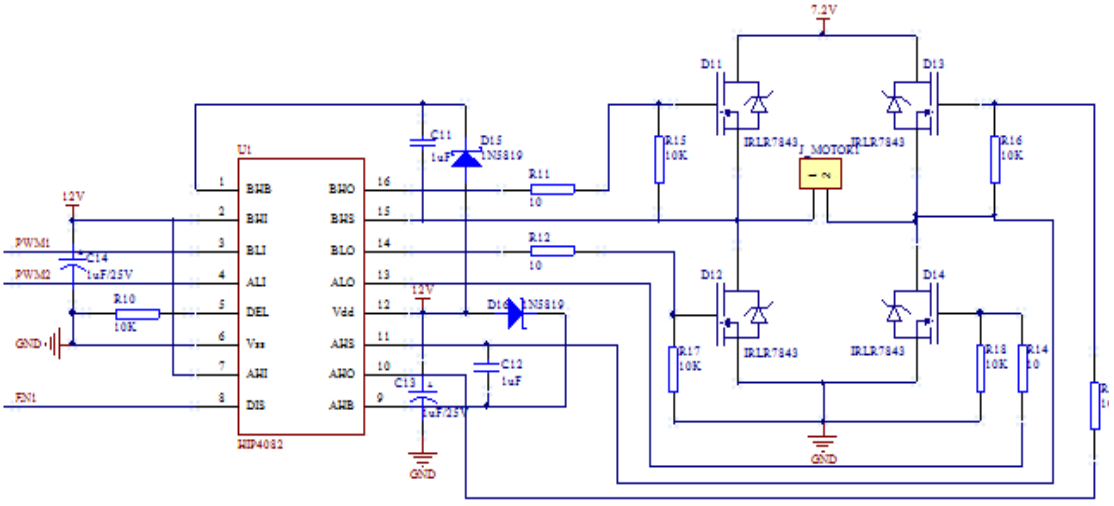


图 4.4 电机驱动电路

4.7 电源供电模块设计

电源模块是整个系统的能源动力机构，所以良好的电源供给是系统稳定运行的前提。使用两个 5V 电源模块分别给主芯片和其他需要 5V 供电的芯片、模块供电。芯片使用贴片式的 LM2940 供电，更稳定。其他模块使用插件式的 LT1086 供电，

能提供更大的电流。原理图如下图 4.7。

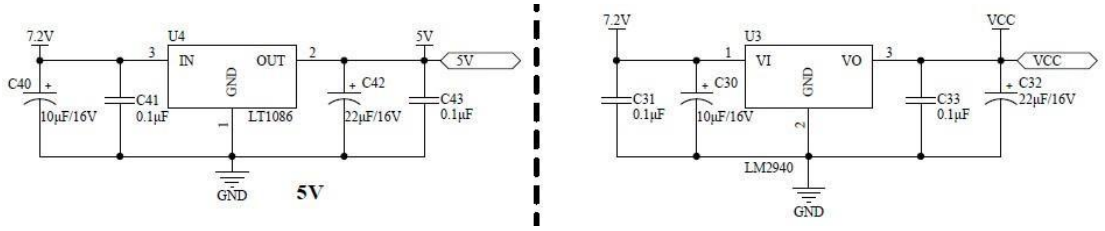


图 4.5 系统电源模块

4．8 辅助调试模块

辅助调试模块虽然不是系统的必要部分，但是在小车的调试过程中却起着至关重要的作用，我们采用的调试模块包括人机交互、拨码开关、指示灯、蓝牙模块。人机方便修改参数，并可以实时显示赛道图像；拨码开关用于调速，及控制是否使用灯塔发车停车；指示灯用于显示赛道元素；蓝牙模块用于将赛道图像及控速波形等信息发送到上位机。

4．9 本章小结

本章详细介绍了车载系统的硬件电路结构和工作过程，采用鹰眼摄像头搭建系统，系统硬件电路简单，工作稳定。在机械和硬件电路搭建完毕后，就需要配合软件平台，经过测试发现，该硬件平台可支持软件平台的稳定运行。

第五章　软件系统设计

5.1 软件系统整体设计方案

高效稳定的程序是智能小车平稳快速行驶的保证。我们设计的智能车系统采用鹰眼摄像头进行赛道识别，图像采集、处理成了整个软件的核心内容之一。在小车转向和速度控制方面，智能车用仍然采用的经典 PID 控制算法，不断把理论和实际结果比较，再进行适当的补偿，使智能车在赛道上达到平稳快速的效果。具体过程为：使用编码器监测智能车的实际速度，再根据赛道信息给定智能车运行速度，运用增量式 PID 算法调节驱动电机转速，实现了电机的快速响应。

5.2 提取边线

边线怎么去快速获取，怎么能获得正确的边线，避免其他赛道的影响，是小车可以沿赛道正确运行的前提。因为摄像头视角的原因，距小车最近的行最可靠，不用担心旁边赛道的影响。所以，找边线时从近向远开始找，近处行的边线作为远处的基准，赛道的边线不会突变，所以下一行的边线的范围以这一行的边线做一个限制。若在这个范围内没找到边线，则可能是十字弯，或者是弯道的尽头。所以往后找不到边线时，会加入一个判断，判断是以上哪种情况，然后根据不同的情况，采用不同的寻线的方式。

5.3 赛道特殊元素的识别

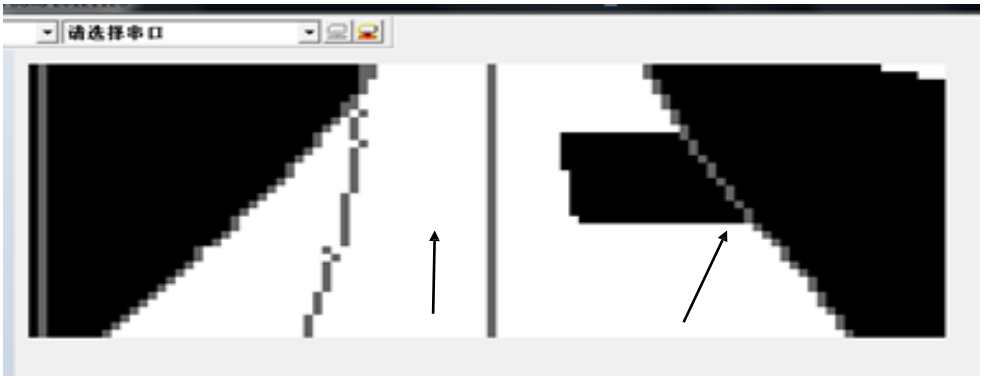
第十一届摄像头组涉及到的一些特殊元素有十字弯、障碍、坡道。下面分别介绍不同元素的识别方法。

5.3.1 十字弯

十字弯的识别其实只是个副产物，我们真正想要的是，十字弯后的边线怎么找。我们的方法是，当连续丢线时，并且连续三行出现全行都是白点的行，然后去判断是否是十字弯，如图所示：

我们的依据是进行纵向判断，箭头 1.3 处是比较平的，即纵向变化很小，然而箭头 2 处纵向的边线变化很大，这样即满足十字弯的条件，然后从左到右找纵向边线变化由平缓到剧烈的突变点，从右向左一样，即可找到箭头 1.3 所指的点，往后的边线，就在这个点左右一个限制范围内找。

5.3.2 障碍



如上图所示：判断障碍的时间是：左边线可以正常寻线，但右边线突然找不到边线，开始判断是否有障碍，识别障碍的方法是：在丢线那一刻时的左右

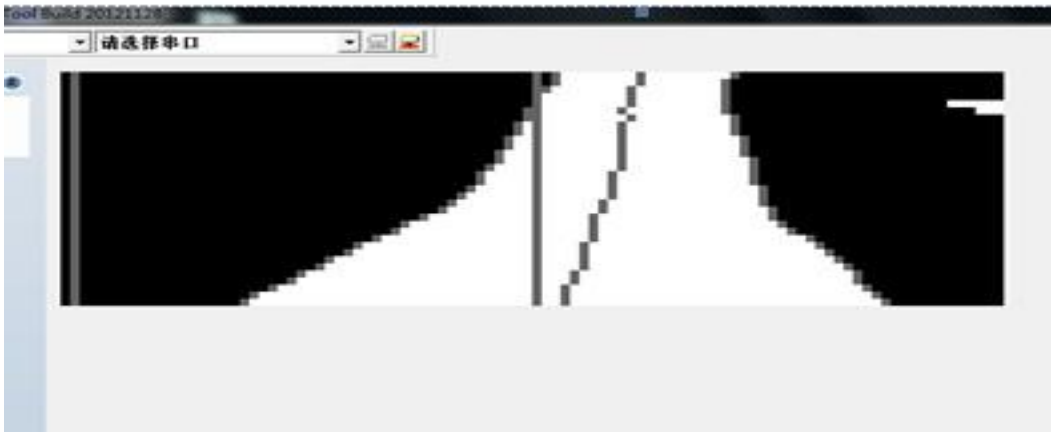
边线之间出现箭头 2 纵向边线变化缓慢，基本是平的，而箭头 1 处的纵向边线相对于丢线时刻所对应的行变化很大，这样即判断为障碍。识别到障碍时，假如障碍在右，则只依赖左边线拟合中线，将中线拟合的偏离实际赛道的中线。障碍的状态解开的时间是，箭头 2 所指的那段很平的面出现在最近行，表示小车已即将过障碍了。

5.3.2 坡道

坡道属于比较好识别的元素，只要判断赛道大宽度增大，且比弯道处增大的厉害，便判断成上坡。

下坡的识别与上坡相反，识别到赛道宽度变窄。因为在坡道上有些图像是未知的，所以在坡道上，我们限制了对直角弯和障碍的识别，速度控制方面，为了尽量快速稳定的通过，在上坡时我们不做限制，在刚识别下坡时给了一个很低的

速度，让其速度减缓一下，只要小车度过坡道的那条棱角，便可恢复正常。



图像处理方面大概就这些，概况一下：第一步是图像采集，把摄像头的采集时序弄清楚是基础，第二步便是边线搜寻，怎么能把所有正确的边线找到，把其他赛道的影响排除掉很重要，第三步是赛道特殊元素的识别，应该所第三步和第二步相辅相成，元素识别需要依赖找的边线，边线找寻的方法又要由赛道不同元素而决定，第四步便是中线的拟合，当两条边线都找到时可简单的求平均值，但只有一条边线时怎么更好的拟合中线也很重要。

5.4 控制算法

5.4.1 电机控制策略

整体思想：用 PWM 波的等面积效应来控制电流大小，即控制电机的转速，用全桥来控制电流的方向，从而可控制电机的正反转。根据赛道的情况，通过比例或者建立的模型进行给定速度，再通过光电编码器，对轮子实际速度的测量，通过反馈偏差进行 PID 校正控制。

B 车采取的是机械差速，通过调整差速器来达到左转右转时两个轮子速度不同的控制策略，由于 B 车跑起来较为优雅，加速快减速慢的特性，我们整个比赛过程中采取了匀速的控制策略，但在入弯时会有被动减速的可能。

5.4.2 电机的 PID 控制

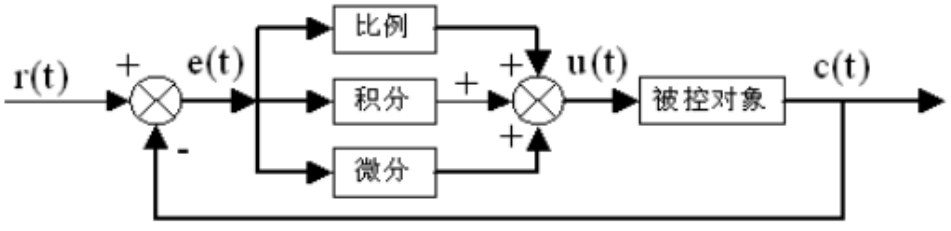


图 5.1 PID 控制器原理框图

PID 控制以其结构简单、稳定性好、工作可靠、调整方便而成为工业控制的主要技术之一。当被控对象的结构和参数不能完全掌握，或得不到精确的数学模型时，系统控制器的结构和参数必须依靠经验和现场调试来确定，这时应用 PID 控制技术最为方便。

PID 控制就是根据系统的误差，利用比例(P)、积分(I)、微分(D)计算出控制量进行控制的。目前 PID 控制在工业控制系统中无处不见，随着控制效果的要求不断提高，PID 逐渐向智能化发展，但形形色色的现代控制理论最终还是源自经典 PID 理论。

PID 解决了自动控制理论所要解决的最基本问题，既系统的稳定性、快速性和准确性。调节 PID 的参数，可实现在系统稳定的前提下，兼顾系统的带载能力和抗扰能力，同时，在 PID 调节器中引入积分项，系统增加了一个零积点，使之成为一阶或一阶以上的系统，这样系统阶跃响应的稳态误差就为零。

由于自动控制系统被控对象的千差万别，PID 的参数也必须随之变化，以满足系统的性能要求。因此 PID 参数的确定是 PID 控制中重要的部分之一。[10]

PID 控制的基本原理如式 1 所示：

$$u(t) = K_p\left[e(t) + \frac{1}{T_i}\int_0^t e(t)\,\mathrm{d}t + T_d\,\frac{\mathrm{d}e(t)}{\mathrm{d}t}\right] \quad (1)$$

式中，$u(t)$ -- 控制量

K_P -- 比例系数

T_i -- 积分时间常数

T_d -- 微分时间常数

将式 1 离散化，即将描述连续系统的微分方程代之以等效的描述离散系统的差分方程，就可以得到相应的数字 PID 调节器。当控制周期也就是积分时间足够小的时候，利用矩形法进行数值积分，即以求和代替积分，以差分代替微分，可得到数字形式的 PID 控制方程如式 2 所示：

$$u(k)=K_p\left[e(k)+\frac{1}{T_i}\sum_{n=0}^{k}e(n)T+T_d\frac{e(k)-e(k-1)}{T}\right] \qquad (2)$$

式中 T 为控制周期，当 T 足够小时离散化的控制模型可以逼近连续式的 PID 控制模型。另外，可以将式 2 中的 T 提取出来，把 K_D 分配进去，分别与 $1/T_i$ 和 T_d 组成比例项系数 K_P，积分项系数 K_I 和微分项系数 K_D，式 2 可简化为式 3：

$$u(k)=K_Pe(k)+K_I\sum_{n=0}^{k}e(n)+K_D[e(k)-e(k-1)] \qquad (3)$$

式 3 也就是常说的位置式 PID 控制方程，但是由于积分项累加计算比较复杂，而且在程序计算中极容易出现溢出现象，因此人们考虑每一次不是给出绝对的控制量，而是根据上一次的控制，给出本次需要增加的控制量即可。根据这一想法，我们首先考虑上一次(即 k-1 时)给出的控制量 $u(k-1)$ 为：

$$u(k-1)=K_Pe(k-1)+K_I\sum_{n=0}^{k-1}e(n)+K_D[e(k-1)-e(k-2)] \qquad (4)$$

式 3-式 4 可得 k 时刻需要给出的增量

$$u(k)-u(k-1)=K_P[e(k)-e(k-1)]+K_Ie(k)+K_D[e(k)-2e(k-1)+e(k-2)] \quad (5)$$

由式 5 得输出 $u(k)$ 也可表示为：

$$u(k)=u(k-1)+K_P[e(k)-e(k-1)]+K_Ie(k)+K_D[e(k)-2e(k-1)+e(k-2)] \quad (6)$$

式 6 就是增量式 PID 控制方程，由此可以看出，增量式 PID 消去了积分项的累加部分，这样便于程序编写。

5.4.3 舵机控制策略

我们的舵机采用位置式的 PD 的变形，整场加权平均作为 P 项，远处某段与近处某段偏差作为 D_Space 项，前后两场偏差作为修正量 D_Time 项。

5.4.3.1 逐差法求出远近某行的斜率

对全场中线加权平均，可以减小因未知状况导致的错误打角的发生可能，同时通过修改权重，可以实现提前打角的程度，与不同的速度进行匹配。

5.4.3.2 前后某两行的偏差，作为 D_Space 项

前后某两行的偏差作为其中一个 D 项控制舵机，可以使小车入弯提前开始打角，同时可以促使其在急弯中打死角。

5.4.3.3 前后两场的偏差的偏差，作为 D_Time 项

前后两场的偏差的偏差作为另一个 D 项控制舵机，对打角值进行修正。

5.5 本章小结

本章详细介绍了系统软件算法，包括图像处理算法和控制算法。

第六章　调试方法

通过调试工具才能知道单片正在做的事，从而人通过分析才能知道单片机是否在按照人的意图去执行，如果没有调试工具是不能做好车的，甚至是不可能完成小车的制作。

6.1 本系统中所使用调试工具简介

在本系统的完成中，主要使用到了以下两种调试组合工具：

（1）IAR 在线调试

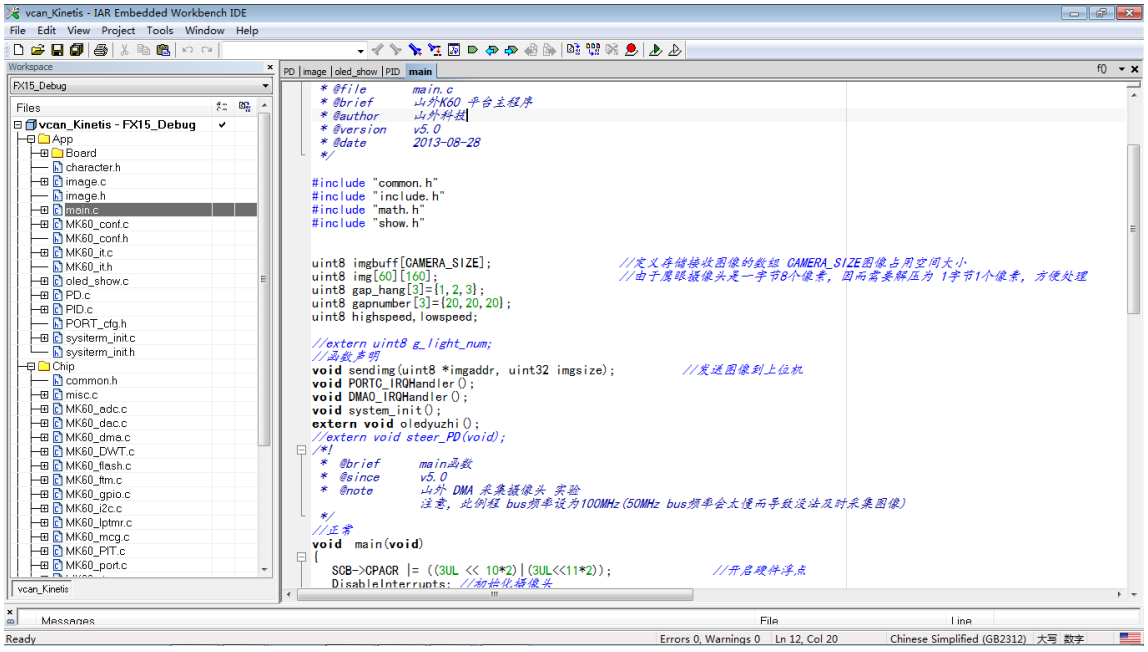


图 6.1　IAR 程序界面

IAR 是最基本的调试工具，在没有其他调试工具的时候这个已经能完成很多事，在后面会单独讲解利用这套调试工具可以做的事，并且熟练使用这个最基础的调试工具是做智能车最基本的要求。

（2）串口示波器

串口示波器是直接使用串口实现单片机和 PC 机的通讯，通过这个调试工具可以查看静态和动态的波形，包括预期速度波形、实际速度波形等等。其上位机的人机界面如下图，一共有四个通道，操作简单，功能强大。

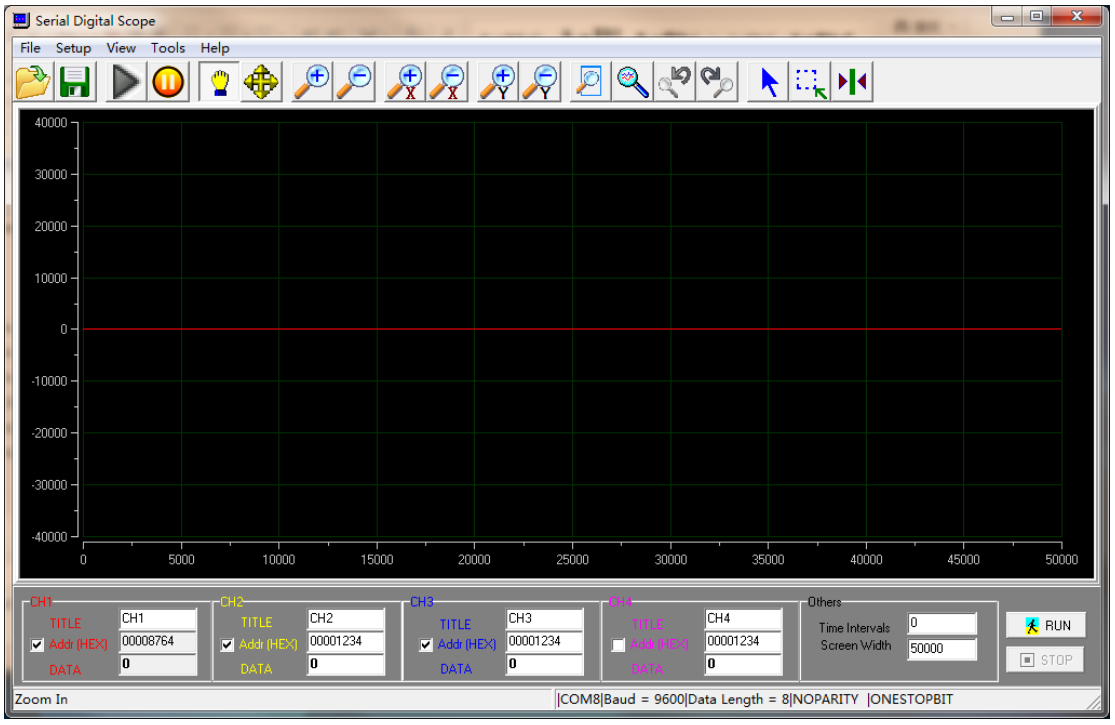


图 6.2 串口示波器调试界面

6.2 IAR 在线调试工具使用

通过在线调试，可以静态的查看定义的所有变量值，可以看到摄像头在某一时刻的采集到的数据值，可以用这种调试方法做到：

（1）摄像头的安装固定，摄像头前瞻的确定，每两采集行见距离的测量；

（2）摄像头采集数据正误的检查；

（3）在静态时，观测黑线提取的是否正确；

（4）可以检查各个硬件模块是否工作在正常情况下，并对程序的执行正误做出分析；

（5）可进行测速模块的反馈值检测；这套调试工具是在前期软件编写过程中最常使用的一套工具，也是和小车机械结构安装的辅助工具。对于这套调试工具的使用之处还有许多地方，在此就不一一说明。

第七章　结论

7.1　主要内容总结

首先根据竞赛规则及功能要求，设计以恩智浦半导体公司的MK60FN1M0VLQ15单片机和鹰眼摄像头为核心的智能车系统。通过摄像头采集的赛道信息送入单片机，在单片机内计算出赛道的轨迹状况，选择最优行进路线。整个系统包括车体机械结构设计和系统的软／硬件系统设计。车体机械结构设计主要包括编码器和图像传感器的安装、舵机和前轮调节等；硬件系统设计完成了电源管理模块、电机驱动模块、速度测量模块、辅助调试模块、图像采集处理模块、舵机控制模块和单片机最小系统模块等的电路设计和调试软件系统设计完成了各功能模块的算法及程序设计，包括图像采集算法设计、以及舵机和电机的PID算法设计。所设计的系统经过测试，赛车能够快速安全行驶。

7.2 存在问题及解决方案

模型车速度和转弯的协调问题。速度提高了，转弯就不稳，要顺利的转弯，速度又上不去。为了协调这个问题，在算法上多次改进，但是效果不好。后来在机械上加以改进，从而最后取得比较好的效果。从而认识到了机械对于小车的重要性，同时也认识到系统各个部分都是不可分离的。

一个良好的系统仍然需要不断完善和更新，为了保证车子能够稳定达到最高平均速度，智能车系统的设计还是在不断的改进的过程。但是限于水平有限，我们的小车仍然有很多需要改进的地方。

7.3　心得体会

在小车的制作过程中，前期把主要精力都放在了控制方面，但是随着速度的提高，机械方面的问题便暴露出来了，但由于在机械方面缺乏经验，在借鉴了别人的经验和自己的多次尝试后，小车的机械问题也得到了比较好的解决，但是小车还是有很大的调整空间。

在智能车制作过程中，遇到过很多问题，在老师和师兄们的帮助下，再加上我们的不懈努力调试，终于在不断的改进中，完成了智能小车的制作。在智能小车的制作过程中，不仅丰富了我们的知识，并且锻炼了动手能力，同时培养了团队合作精神。在此感谢恩智浦组委会对本次比赛的大力支持，同时感谢合肥工业大学恩智

浦智能车工作室的指导老师张阳老师和师兄师姐们的帮助。

参考文献

[1]卓晴、黄开胜、邵贝贝等编. 学做智能车—挑战“飞思卡尔”杯. 北京：北京航空航天大学出版社，2007. 366~369

[2]黄开胜、金华民、蒋狄南著. 韩国智能模型车技术方案分析. 北京：北京航空航天大学出版社，2007. 45~51

[3]卓晴著. 智能汽车自动控制器方案设计. 北京：北京航空航天大学出版社，2007. 5

[4]邹朗豪、李文昌、苏启健著. Rapid prototype and smartcar design. 北京：北京航空航天

大学出版社，2007. 71~86

[5]黄开胜、陈宋著. 汽车理论与智能模型车机械结构调整方法. 北京：北京航空航天大学出版社，2007. 23

[6]西南科技大学第六届摄像头组西科超音速队技术报告

[7]合肥工业大学第九届摄像头组斛兵一队技术报告

附录：源程序

```
#include "common.h"
#include "include.h"
#define WanDaoXiShu 1
#define HalfRoad 71
#define R_lim 200
#define L_lim -60
#define RoadX 15
#define Block 35
/***********变量声明*****************************/
extern uint8 img[hang][160];
extern int16 g_i16_count_seconds;
extern  uint8 ramp_seconds;
int16 error_center;
uint8 Road[50]={   29,29,30,30,31,31,32,32,33,33,
                34,35,36,37,37,38,39,40,41,42,
                43,44,45,46,46,47,48,49,50,51,
                52,53,54,55,56,57,58,59,60,61,
                61,64,65,66,67,69,70,71,72,73};

int16 leftblack[hang];  //存放每行左边黑线的位置
int16 rightblack[hang]; //存放每行右边黑线的位置
int16 lastcenter[hang]; //存放拟合中线的位置
uint8 ramp=0,state=0;
uint8 stop_flag=0,stop_flag1=0,zhangai_flag_left=0,zhangai_flag_right=0;
extern uint8 hairpin;
extern uint16 speedsta;
extern uint8 chuwan,ruwan;
extern float kP,kD,kPP,kDD;
extern float E0;
//extern int16 OutData[4];
extern int16 think_speed;
extern uint16 speedmax;
extern uint8 x,x0,x1,x2;

void display_big_numberstr(unsigned char x,unsigned char page, uint16 num, unsigned char
bit_number);

//坡道/******************进行二值化处理******************/
/*
void img_extract(uint8 *dst, uint8 *src, uint32 srclen)
{
   uint8 colour[2] = {255, 0}; //0 和 1 分别对应的颜色
   //注：山外的摄像头 0 表示 白色，1 表示 黑色
   uint8 tmpsrc;
   uint8 i,j;
  // src=src+(120*20);
   for(j=0;j<50;j++)
```

```
  {
    for(i=0;i<20;i++)
    {
      tmpsrc = *src++;
      *dst++ = colour[ (tmpsrc >> 7 ) & 0x01 ];
      *dst++ = colour[ (tmpsrc >> 6 ) & 0x01 ];
      *dst++ = colour[ (tmpsrc >> 5 ) & 0x01 ];
      *dst++ = colour[ (tmpsrc >> 4 ) & 0x01 ];
      *dst++ = colour[ (tmpsrc >> 3 ) & 0x01 ];
      *dst++ = colour[ (tmpsrc >> 2 ) & 0x01 ];
      *dst++ = colour[ (tmpsrc >> 1 ) & 0x01 ];
      *dst++ = colour[ (tmpsrc >> 0 ) & 0x01 ];
    }
    src=src+20;
  }
}
*/
void img_extract2(uint8 *dst,uint8 *src, uint8 *gap,uint8 *gapnum)
{
  uint8 colour[2] = {255, 0}; //0 和 1 分别对应的颜色
  //注：山外的摄像头 0 表示 白色，1 表示 黑色
  uint8 tmpsrc;
  uint8 i,j,x;
  for(x=0;x<3;x++)
  {
    for(i=0;i<*gapnum;i++)
    {
     for(j=0;j<20;j++)
     {
       tmpsrc = *src++;
       *dst++ = colour[ (tmpsrc >> 7 ) & 0x01 ];
       *dst++ = colour[ (tmpsrc >> 6 ) & 0x01 ];
       *dst++ = colour[ (tmpsrc >> 5 ) & 0x01 ];
       *dst++ = colour[ (tmpsrc >> 4 ) & 0x01 ];
       *dst++ = colour[ (tmpsrc >> 3 ) & 0x01 ];
       *dst++ = colour[ (tmpsrc >> 2 ) & 0x01 ];
       *dst++ = colour[ (tmpsrc >> 1 ) & 0x01 ];
       *dst++ = colour[ (tmpsrc >> 0 ) & 0x01 ];
     }
     src=src+(*gap-1)*20;
    }
    gapnum++;
    gap++;
  }
}

void Get_line(void)
{
  uint8
RoadState[6],middle_use,left_use,right_use,E_R1,E_R2,E_L1,E_L2,leftblock,square_hang,squar
```

```
e_left,square_right,square_lie,ramp_temp,rightblock,right_H,left_H,LK,RK,tenleft,double_start
_miss,tenright,outtrack,left_start_miss_hang[2],left_start_miss_lie[2],right_start_miss_hang[2]
,right_start_miss_lie[2],left_end_miss_hang,right_end_miss_hang,left_end_miss_lie,right_end
_miss_lie,increase;
    int8 left_state,right_state,temp_error_left,temp_error_right;
    uint16 average_left,average_right;
    int16
temp_error_center,average_center,error_left,error_right,j,i,ave,f_ave,c_ave,left_i,right_i,left_j
,right_j,z,black_num_left,black_num_right;
    int16 temp_temp,temp_temp2,stop_line_right,stop_line_left;
    uint8 left_miss_number,right_miss_number,double_miss,k;
    int8 WEL0,WER0,WEL1,WEL2,WEL3,WER1,WER2,WER3, shuzheshu[80][2];
    int16 rightblack2[hang];
    static uint8 top=0;
    average_center=0;
    E_R1=0;
    WEL0=0;WEL1=0;WEL2=0;WEL3=0;
    WER0=0;WER1=0;WER2=0;WER3=0;
    left_use=0;
    right_use=0;
    middle_use=0;
    E_R2=0;
    E_L1=0;
    E_L2=0;
    temp_error_center=0;
    stop_line_right=0;
    stop_line_left=0;
    z=0;
    temp_temp2=0;
    black_num_left=0;
    black_num_right=0;
    zhangai_flag_left=0;
    zhangai_flag_right=0;
    left_i=0;
    right_i=0;
    left_j=0;
    right_j=0;
     LK=0;
    ramp_temp=0;
    RK=0;
    leftblock=0;
    rightblock=0;
    square_hang=0;
    square_lie=0;
    square_left=0;
    square_right=0;
    left_H=0;
    right_H=0;
    double_start_miss = 0;
    left_start_miss_lie[0] = 0;
```

```
left_start_miss_hang[0] = 0;
left_start_miss_lie[1] = 0;
left_start_miss_hang[1] = 0;
left_end_miss_hang = 0;
left_end_miss_lie = 0;
right_start_miss_hang[0] = 0;
right_start_miss_lie[0] = 0;
right_start_miss_hang[1] = 0;
right_start_miss_lie[1] = 0;
right_end_miss_lie = 0;
right_end_miss_hang = 0;

k = 0;
increase = 0;
double_miss = 0;
left_miss_number = 0;
right_miss_number = 0;
temp_error_left = 0;
temp_error_right = 0;
left_state = 0;
right_state = 0;
average_left = 0;
average_right = 0;
error_left = 0;
error_right = 0;
outtrack = 0;
ave=0;
for(temp_temp=0;temp_temp<6;temp_temp++)
{
   RoadState[temp_temp]=0;
}
/*******************************
最低三行，从两边向中间扫定出基准
49,48,57 行
*******************************/
if(E0<-30)
{
  for(i=50-1; i>50-4;i--)
  {
      j = 0;
      while(img[i][j] == black && j<=159)
      {
         j++;
      }
      leftblack[i] = j;
      while(img[i][j] == white && j<=159)
      {
         j++;
      }
      rightblack[i] = j;
```

```
        lastcenter[i] = (leftblack[i]+rightblack[i])>>1;

    }
  }
  if(E0>30)
  {
    for(i=50-1; i>50-4;i--)
    {
       j = 159;
       while(img[i][j] == black && j>2)
       {
          j--;
       }
       rightblack[i] = j;
       while(img[i][j] == white && j>2)
       {
          j++;
       }
       leftblack[i] = j;
       lastcenter[i] = (leftblack[i]+rightblack[i])>>1;
      }
  }
 if(E0<30||E0>-30)
  {
  for(i=50-1; i>50-4;i--)
  {
    j = 0;
    while(img[i][j] == black && j<=159)
    {
        j++;
    }
    leftblack[i] = j;
    j = 159;
    while(img[i][j] == black && j>2)
    {
        j--;
    }
    rightblack[i] = j;
    lastcenter[i] = (leftblack[i]+rightblack[i])>>1;
     }
  }
 /*********************************
  自第 47 行向上寻找边界，
  1.左边,在上一行的基础上右移 3 点，若为白则向左找黑点，
  2.若为黑则向右找白点,同时计数，若黑点次数(tenleft)大于等于 5，用 leftblock 记住此
   处行数，一幅图只会记住第一个 leftblock。
  3.若找到的边线比上一行向外扩了 3 个像素，则判定为突变。将其上一行置于图像两边，
   这一行取上一行的值，并计次数（left_miss_number/right_miss_number），完成一次
   置换，并记录一幅图第一次置换的行数（left_start_miss_hang[]/right_start_miss_hang[]）
   和第二次置换的行数，以及列数（left_start_miss_lie[]）,LK 和 RK 辅助记录当前置换状
```

```
态
  4.outtrack 出赛道的判定，当左线大于右线，判定为 outtrack
  5.double_miss 当左右同时靠近边线，计次数。第一次双丢行数为 double_start_miss
       **********************************/
  for(i=50-4;i>0;i--)
       {
         j = leftblack[i+1] + 3;   //在上一行的基础上右移 3 点，若为白则向左找黑点，若为黑则向右找白点
    if(white ==img[i][j])
        {
      while(img[i][j]==white && j>0)//若到边界自动跳出
                {j--;}
      tenleft=0;
        }
        else
        {
      tenleft=0;
          while(img[i][j]==black && j<159)
              { j++;  tenleft++;  }
      if(tenleft>=5 && leftblock ==0)
      {
        leftblock = i;//记右移较多的行数
      }
        }
        leftblock[i] = j;
    WEL3=WEL2;
    WEL2=WEL1;
    WEL1=WEL0;
    WEL0=leftblock[i]-leftblock[i+1];
    //置换
    if(leftblock[i]<(leftblock[i+1]-3) || ((WEL0+WEL1<0) && WEL2>0 &&WEL3>0 ))
        {
            leftblock[i] = leftblock[i+1];
      leftblock[i+1] = 0;
            left_miss_number++;
      if(LK==0 && left_start_miss_hang[0]==0)
      {LK=1;left_start_miss_hang[0]= i;
      left_start_miss_lie[0] = leftblock[i];
      left_miss_number--;
      leftblock[i+1]=leftblock[i];}
      else if(left_start_miss_hang[1]==0 && LK==2)
      {LK=3;left_start_miss_hang[1]= i+1;
      left_start_miss_lie[1] = leftblock[i+2];}
       }
      else if(LK==1)
      {LK=2;}
        //右边
        j = rightblack[i+1] - 3;   //在上一行基础上左移 5 点，若为白则向右找黑点，若为黑则向左找白点
        if(white ==img[i][j])     //若白
```

```
    {
       while(img[i][j]==white && j<159)  //若到边界自动跳出
              {j++;}
tenright=0;
    }
    else
    {
tenright=0;
       while(img[i][j]==black && j>0)
              {  j--;tenright++;   }
  if(tenright>5 && rightblock ==0)
  {
     rightblock = i;
  }
    }
    rightblack[i] = j;
WER3=WER2;
WER2=WER1;
WER1=WER0;
WER0=rightblack[i+1]-rightblack[i];

    if(rightblack[i]>(rightblack[i+1]+3) || ((WER0+WER1<0) && WER2>0 &&WER3>0))
    {
  rightblack[i] = rightblack[i+1];
  rightblack[i+1] = 159;
          right_miss_number++;
  if(RK==0 && right_start_miss_hang[0]==0)
  {
   RK=1;//RK=1 为第一次找到初始丢失行
   right_start_miss_hang[0] = i;
   right_start_miss_lie[0] = rightblack[i];
   right_miss_number--;
   rightblack[i+1]=rightblack[i];
  }
  else if(RK==2 && right_start_miss_hang[1]==0)
  {
   RK=3;//RK=3 第二次找到丢失行
   right_start_miss_hang[1] = i+1;
   right_start_miss_lie[1] = rightblack[i+2];
  }
     }
else if(RK==1)
{
 RK=2;//RK=2 第一次补线结束
}
if(leftblack[i+1]==0 && rightblack[i+1]==159)
{
  double_miss++;
  if(double_start_miss==0)
  {double_start_miss = i;}
```

```
        }
            if(i<45)
    {
      if(leftblack[i]>rightblack[i] || (rightblack[i]<30 && tenright>6 && E_R1>4 &&E_R2>4)
||(leftblack[i]>130 && tenleft>6 && E_L1>4&&E_L2>4))
      {
                  outtrack = i;
                  break;
             }
      E_R2=E_R1;
      E_R1=tenright;
      E_L2=E_L1;
      E_L1=tenleft;
    }
         }//End Of          for(i=50-3;i>5;i--)
  for(i=0;i<49;i++)
  {
   rightblack2[i]=rightblack[i];
  }
  /*************************
   起跑线识别
  *************************/
  if(g_i16_count_seconds>100)
  {
    if(outtrack == 0 &&leftblack[49]<leftblack[1] && leftblack[1]<rightblack[1] &&
rightblack[1]<rightblack[49]&&( right_miss_number==0 || left_miss_number==0)&&(E0>-
10&&E0<10)&&ramp==0)
    {
     for(i=30;i<50;i++)
     {
      for(z=0;z<5;z++)
      {
        if(black ==img[i][leftblack[i]+(Road[i]/2)-2+z])
        {
          for(j=0;j<3;j++)
          {
            if(black==img[i-j][rightblack[i]-(Road[i]/2)+2-z])
            {
              black_num_left++;
              break;
            }
          }
         }
      }
      for(z=0;z<5;z++)
      {
       if(black ==img[i][rightblack[i]-(Road[i]/2)+2-z])
        {
          for(j=0;j<3;j++)
          {
```

```
            if(black==img[i-j][leftblack[i]+(Road[i]/2)-2+z])
            {
               black_num_right++;
               break;
            }
          }
       }
    }
    if(black_num_left==5||black_num_right==5)
    {
       black_num_left=0;
       black_num_right=0;
       z=0;
       stop_flag1=1;
    }
  }
  if(stop_flag1==1)
  {
     for(i=28;i<48;i++)
     {
        if(white ==img[i][leftblack[i]+25])
        {
           for(j=0;j<3;j++)
           {
              if(white ==img[i-j][rightblack[i]-25])
              {
                 stop_line_left++;
                 break;
              }
           }
        }
     if(white ==img[i][rightblack[i]-25])
     {
        for(j=0;j<3;j++)
        {
           if(white ==img[i-j][leftblack[i]+25])
           {
              stop_line_right++;
              break;
           }
        }
     }
    }
    if( stop_line_right==20|| stop_line_left==20)
    {
       stop_line_right=0;
       stop_line_left=0;
       stop_flag=1;
       stop_flag1=0;
       gpio_init (PTA14, GPO, 0);
```

```
    }
   }
  }
}

/*************************
  坡道
*************************/
if(outtrack != 0 )
{
 ramp_temp=ramp;
 ramp = 1;
 for(i=1;i<15;i++)
 {
  if(leftblack[outtrack+i] == 0 || rightblack[outtrack+i] == 159)//判别是否是弯道
  {
   ramp=ramp_temp;
   break;
  }
 }
}
if(ramp == 1 && top==0)
{
 for(i=49;i>outtrack+3;i--)
 {
  if(leftblack[i]-leftblack[i+1]>5 && leftblack[i]<80)
  {
   for(j=0;j<3;j++)
   {
    if(rightblack[i+1-j]-rightblack[i-j]>5 &&rightblack[i]>80)
    {
     top=1;
     break;
    }
   }
  }
  if(rightblack[i+1]-rightblack[i]>8)
  {
   for(j=0;j<3;j++)
   {
    if(leftblack[i-j]-leftblack[i+1-j]>8)
    {
     top=1;
     break;
    }
   }
  }
  if(top==1)
  {
  //   gpio_init (PTA14, GPO, 0);
```

```
      RoadState[0] = 1;//检测到坡道
      break;
    }
   }
  }
 if(top==1)
 {
  for(i=48;i>outtrack+1;i--)
  {
    if(leftblack[i]-leftblack[i+1]>3 || rightblack[i+1]-rightblack[i]>3)
    {
      break;
    }
  }
  if(i<outtrack+4)
  {
    top=0;
    ramp=0;
    ramp_seconds = 0;
  }
 }
 if(top==0)
    {
    //  gpio_init (PTA14, GPI, 1);

    }
 /************************************************
 1.求大趋势走向,求各边线（除了置换了的行）的平均值，
 并与第 37 行比较，判断相对于 37 行，赛道总体是收缩还是扩张，
 2.求弯曲程度，所有边线（除了置换了的行）与平均值之差的绝对值之和
 3.赛道宽度，求第 16 行的左右边线之差，为了防止急弯处 outtrack 之后的路况不可控
 影响判断，当 outtrack 离得比 16 行近的时候，看 outtrack 行的宽度
************************************************/
 //平均值
 for(i=49;i>outtrack;i--)
 {
    average_left += leftblack[i];
    average_right += rightblack[i];
 }
 average_left = average_left/(49-outtrack-left_miss_number);
 average_right = (average_right-right_miss_number*159)/(49-outtrack-right_miss_number);
 //弯曲程度
 for(i=49;i>outtrack;i--)
 {
    temp_error_left = leftblack[i]-average_left;
    temp_error_right = rightblack[i]-average_right;
    if(temp_error_left>=0)
    {
       error_left += temp_error_left;
    }
```

```
      else
      {
        error_left -= temp_error_left;
      }
      if(temp_error_right>=0)
      {
        error_right += temp_error_right;
      }
      else
      {
        error_right -= temp_error_right;
      }
    }
    //趋势
    error_left = (error_left-left_miss_number*average_left)/(49-outtrack-left_miss_number);
    error_right = (error_right-(159-average_right)*right_miss_number)/(49-outtrack-
right_miss_number);
    //前方赛道宽度
    if(outtrack>15)
    {
          temp_temp = rightblack[outtrack]-leftblack[outtrack];
    }
    else
    {
          temp_temp = rightblack[15]-leftblack[15];
    }

    left_state = average_left-leftblack[36];  //负值左趋，正值右趋
    right_state = average_right-rightblack[36];
/*******************************************************
  1.补线，之前已经记录了置换的起点行和列，求终点并连线。
  终点是起点向上找到黑点，并找寻附近不会有突变的连续的点，
  条件：补右线的时候，outtrack 较远且左右都有丢线（正入十字弯）或者不存在
outtrack 且只有一边丢线（斜入十字弯），右边有丢线，且左线要么向外趋势，向右趋的
话，
    则需要弯曲程度较小
      补左线类似
  2.急弯也存在补线，此处应用二次方程，二次方程的系数是宏定义的，写在本 C 文件最
上面
  条件：outtrack 不会太远，右边有丢线
*******************************************************/

    if(((right_miss_number!=0)&&(left_miss_number!=0)&&(outtrack<16))||(outtrack<10  &&
right_miss_number!=0 &&(left_state<=0 ||(left_state>0 && error_left<25))))//补右边
    //不可存在出赛道，且右边丢失多
    {
      for(k=0;k<2;k++)
      {
        j = right_start_miss_lie[k];
        temp_temp=0;
```

```
        while(temp_temp<80 && j>0)//从突变起点向上找终点
        {
          i = right_start_miss_hang[k];
          while(img[i][j]==white && i>1)//竖着向上找黑边
          {i--;}
          if(i==1)
          {
            shuzheshu[temp_temp][0]=50;
          }//没找到，记为 50
          else
          {
            shuzheshu[temp_temp][0]=i;
          }//找到了，记下行数
          shuzheshu[temp_temp][1]=j;
          if(temp_temp>3 && shuzheshu[temp_temp-2][0]!= 50 &&
(shuzheshu[temp_temp][0]-shuzheshu[temp_temp-1][0])<0 &&(shuzheshu[temp_temp-1][0]-
shuzheshu[temp_temp-2][0])<0  )
          {
            left_j=0;//随便写一句，方便设断点
            break;
          }
          j--;
//          if(temp_temp>8&&shuzheshu[temp_temp][0]==50 && shuzheshu[temp_temp-
1][0]==50&&shuzheshu[temp_temp-2][0]==50)//若连续三个点没找着
//          {
//            left_j=temp_temp;
//            temp_temp=80;
//            break;
//          }
          temp_temp++;
        }
        if(temp_temp>=78 || j==0)//没找到
        {
          temp_temp2=0;
//          for(i=right_start_miss_hang[k];i>outtrack;i--)
//          {
//            if(leftblack[i]<3)//数一下左边为边界的个数
//            {temp_temp2++;}
//          }
          for(i=7;i>outtrack;i--)
          {
            if(leftblack[i]<3)//数一下左边为边界的个数
            {temp_temp2++;}
          }
          if(temp_temp2<3 && k==0 &&right_start_miss_hang[k]>5)//左边无边线
          {
            RoadState[2]=2;//没找到终点且左边线不在边界（右转）
            right_H=right_start_miss_hang[k];
            for(i=right_H;i>0;i--)
            {
```

```
            j = rightblack[i+1] - 3;       //在上一行基础上左移 5 点，若为白则向右找黑点，
若为黑则向左找白点
             if(white ==img[i][j])  //若白
             {
               while(img[i][j]==white && j<159)  //若到边界自动跳出
               {j++;}
             }
             else
            {
               while(img[i][j]==black && j>0)
              {j--;}
             }
             rightblack[i] = j;
          }
        }
        else//左边在边界处有值
        {
          if(j==0||outtrack!=0||temp_temp!=80)
          {
            RoadState[2]=3;//出十字当做弯道处理
            outtrack=right_start_miss_hang[k];
          }
          else if(error_left<17 && error_right<17)//入十字
          {
            RoadState[2]=4;//入十字，斜率为 1，盲补（待修，行数应较远）
            for(i=right_start_miss_hang[k];i>0;i--)
            {
                rightblack[i]=rightblack[i+1]-1;//+(rightblack[i+2]-rightblack[i+5])/3;
                 if(rightblack[i]>158){rightblack[i]=159;}
                 else if(rightblack[i]<2){rightblack[i]=0;}
            }
          }
//          else if(error_right>error_left)
//          {
//            left_H=right_start_miss_hang[k];
//          }
//          else
//          {
//            right_H=right_start_miss_hang[k];
//          }
//
          /*
          else
          {
           right_H=right_start_miss_hang[k];
           i=temp_temp2;
           for(i=right_H;i>0;i--)
            {
              j = rightblack[i+1] - 3;    //在上一行基础上左移 5 点，若为白则向右找黑点，
若为黑则向左找白点
```

```
                if(white ==img[i][j])          //若白
                {
                   while(img[i][j]==white && j<159)  //若到边界自动跳出
                   {j++;}
                }
                else
                {
                   while(img[i][j]==black && j>0)
                  {j--;}
                }
                rightblack[i] = j;
              }
            }*/
          }
          k=1;
        }
        else
        {
          RoadState[2]=1;//找到终点并补线
          right_end_miss_hang=shuzheshu[temp_temp][0];
          right_end_miss_lie=shuzheshu[temp_temp][1];
          for(i=right_start_miss_hang[k];i>right_end_miss_hang;i--)
          {
            increase = right_start_miss_lie[k]-right_end_miss_lie;
            rightblack[i] = right_start_miss_lie[k]-(right_start_miss_hang[k]-
i)*increase/(right_start_miss_hang[k]-right_end_miss_hang);
            if(rightblack[i]>158){rightblack[i]=159;}
            else if(rightblack[i]<2){rightblack[i]=0;}
          }
        }
        if(RK!=3)
        {
          k=1;
        }
      }
    }
   else if(right_miss_number>4)
   //如果存在出赛道，且右边丢失多，认为是急右弯
   {
      RoadState[2]=5;//右突变急弯
      right_H=right_start_miss_hang[0];
      for(i=right_H;i>0;i--)
      {
        j = rightblack[i+1] - 3;       //在上一行基础上左移 5 点，若为白则向右找黑点，若为
黑则向左找白点
        if(white ==img[i][j])          //若白
        {
          while(img[i][j]==white && j<159)  //若到边界自动跳出
          {j++;}
        }
```

```
          else
          {
             while(img[i][j]==black && j>0)
             {j--;}
          }
          rightblack[i] = j;
       }
   }
   if(((left_miss_number!=0)&&(right_miss_number!=0)&&(outtrack<16))||(outtrack<10 &&
left_miss_number!=0 &&(right_state>=0 ||(right_state<0 && error_right<25)))) //补左线
   //不可出赛道，且左边丢失多
   {
      for(k=0;k<2;k++)
      {
         j = left_start_miss_lie[k];
         temp_temp=0 ;
         while(temp_temp<80 && j<159)//从突变起点向上找终点
         {
          i = left_start_miss_hang[k];
          while(img[i][j]==white && i>1)//竖着向上找黑边
          {i--;}
          if(i==1)
          {shuzheshu[temp_temp][0]=50;}//没找到，记为 50
          else
          {shuzheshu[temp_temp][0]=i;}//找到了，记下行数
          shuzheshu[temp_temp][1]=j;
          if(temp_temp>8 && shuzheshu[temp_temp-2][0]!= 50 && (shuzheshu[temp_temp][0]-
shuzheshu[temp_temp-1][0])<0 &&(shuzheshu[temp_temp-1][0]-shuzheshu[temp_temp-
2][0])<0)
          {break;}
          j++;
//          if(shuzheshu[temp_temp][0]==50 && shuzheshu[temp_temp-
1][0]==50&&shuzheshu[temp_temp-2][0]==50)//若连续三个点没找着
//          {
//             temp_temp=80;
//             break;
//          }
          temp_temp++;
         }
        if(temp_temp>=78 || j==159)
         {
          temp_temp2=0;
//          for(i=left_start_miss_hang[k];i>outtrack;i--)
//          {
//             if(rightblack2[i]>157)//数一下右边为边界的个数
//             {temp_temp2++;}
//          }
          for(i=7;i>outtrack;i--)
          {
             if(rightblack2[i]>157)//数一下右边为边界的个数
```

```
            {temp_temp2++;}
        }
        if(temp_temp2<3&& k==0 &&left_start_miss_hang[k]>5)//右边存在突变，但是没找到终点，且对应右边是不存在边线的
        {
            RoadState[3]=2;//没找到终点且右边线不在边界（左转）
            left_H=left_start_miss_hang[k];
            for(i=left_H;i>0;i--)
            {
               j = leftblack[i+1] + 3;   //在上一行的基础上右移 3 点，若为白则向左找黑点，若为黑则向右找白点
                if(white ==img[i][j])
                {
                  while(img[i][j]==white && j>0)//若到边界自动跳出
                  {j--;}
                }
                else
                {
                  while(img[i][j]==black && j<159)
                  {j++;}
                }
                leftblack[i] = j;
             }
        }
        else//左边存在突变，但是没有找到终点，且右边存在边线
        {
           if(j==0||outtrack!=0||temp_temp!=80)//左弯
           {
              RoadState[3]=3;
              outtrack=left_start_miss_hang[k];
           }
           else if(error_left<17 && error_right<17)//入十字
           {
               RoadState[3]=4;//待修，行数应较远
               for(i=left_start_miss_hang[k];i>0;i--)
               {
                 leftblack[i]=leftblack[i+1]+1;
                 if(leftblack[i]>158){leftblack[i]=159;}
                 else if(leftblack[i]<2){leftblack[i]=0;}
               }
           }
//           else if(error_left>error_right)//出十字当做弯道处理
//           {
//              right_H=left_start_miss_hang[k];
//
//           }
//           else
//           {
//              left_H=left_start_miss_hang[k];
//           }
```

```
            /*
            else
            {
               left_H=left_start_miss_hang[k];
                for(i=left_H;i>0;i--)
                {
                   j = leftblack[i+1] + 3;    //在上一行的基础上右移 3 点，若为白则向左找黑点，
若为黑则向右找白点
                   if(white ==img[i][j])
                   {
                      while(img[i][j]==white && j>0)//若到边界自动跳出
                      {j--;}
                   }
                   else
                   {
                      while(img[i][j]==black && j<159)
                      {j++;}
                   }
                   leftblack[i] = j;
                }
            }*/
          }
          k=1;
        }
        else//找到终点
        {
           RoadState[3]=1;//找到终点并补线
           left_end_miss_hang=shuzheshu[temp_temp][0];
           left_end_miss_lie=shuzheshu[temp_temp][1];
           for(i=left_start_miss_hang[k];i>left_end_miss_hang;i--)
           {
              increase = left_end_miss_lie-left_start_miss_lie[k];
              leftblack[i] = left_start_miss_lie[k]-(left_start_miss_hang[k]-
i)*increase/(left_end_miss_hang-left_start_miss_hang[k]);
              if(leftblack[i]<2){leftblack[i]=0;}
           }
        }
        if(RK!=3)
        {
           k=1;
        }
     }
   }
   else if(left_miss_number>4)
   //存在出赛道，且左边丢失多，认为急左弯
   {
      RoadState[3]=5;//急左弯
      left_H=left_start_miss_hang[0];
      for(i=left_H;i>0;i--)
      {
```

```
        j = leftblack[i+1] + 3;   //在上一行的基础上右移 3 点，若为白则向左找黑点，若为黑则向右找白点
        if(white ==img[i][j])
        {
          while(img[i][j]==white && j>0)//若到边界自动跳出
          {j--;}
        }
        else
        {
          while(img[i][j]==black && j<159)
          {j++;}
        }
        leftblack[i] = j;
      }
    }
  /****************************************
    出十字 ,第一次双丢的行数在 51-58 行，丢的行数大于 9 行
    且左右弯曲程度小于 25
    ****************************************/
     if(double_start_miss>40 && double_miss>15 && (rightblock==0||leftblock==0))
     {
      temp_temp=0;
     //找到赛道中间方块的角点
     i=49;
     while(i>0 && square_lie==0)
     {
       for(j=40;j<120;j++)
       {
         if(img[i][j]==0)
         {
           if(square_lie==0)
           {
             square_hang=i;
             square_lie=j;
           }
           temp_temp++;
         }
       }
       if(square_lie==0)
       {
         i--;
       }
       else
       {
         break;
       }
     }
     square_lie+=temp_temp>>1;
     j=square_lie;
     //找出该点左边还是右边斜率大
```

```
while(img[i][j-10]==255 && i>0)
{i--;}
square_left=i;
i=square_hang;
j=square_lie;
while(img[i][j+10]==255 && i>0)
{i--;}
square_right=i;
if(square_left==square_right)
{
   j=square_lie;
   while(img[i][j-15]==255 && i>0)
   {i--;}
   square_left=i;
   i=square_hang;
   j=square_lie;
   while(img[i][j+15]==255 && i>0)
   {i--;}
   square_right=i;
}
if(square_left<square_right)//左边斜率大
{
 RoadState[4]=1;//赛道在方块左边
 //向上
 i=square_hang;
 j=square_lie;
 rightblack[i]=j;
 i--;
 for(;i>0;i--)
 {
  temp_temp=4;
  j=rightblack[i+1]-4;
  if(white ==img[i][j])          //若白
         {
    while(img[i][j]==white && j<159 && temp_temp>0)  //若到边界自动跳出
    {j++;temp_temp--;}
  }
  else
  {
    while(img[i][j]==black && j>0)
    {
      j--;
    }
  }
  if(temp_temp>0)
  {
   rightblack[i]=j;
  }
  else
  {
```

```
    rightblack[i]=rightblack[i+1]-1;
   }
  }
  //向下
  i=square_hang;
  j=square_lie;
  for(;i<49;i++)
  {
   if((rightblack[i]-rightblack[i-1])>1 || (rightblack[i]-rightblack[i-1])<0)
   {
     rightblack[i+1]=rightblack[i]+1;
   }
   else
   {
     rightblack[i+1]=rightblack[i]+rightblack[i]-rightblack[i-1];
   }
  }
}
  else//右边斜率大
  {
    RoadState[4]=2;//赛道在方块右边
    //向上
    i=square_hang;
    j=square_lie;
    leftblack[i]=j;
    i--;
    for(;i>0;i--)
    {
     temp_temp=4;
     j=leftblack[i+1]+4;
     if(white ==img[i][j])       //若白
     {
       while(img[i][j]==white && j>0 && temp_temp>0)  //若到边界自动跳出
       {j--;temp_temp--;}
     }
     else
     {
       while(img[i][j]==black && j>0)
       {
         j++;
       }
     }
     if(temp_temp>0)
     {
      leftblack[i]=j;
     }
     else
     {
      leftblack[i]=leftblack[i+1]+1;
     }
```

```
    }
    //向下
    i=square_hang;
    j=square_lie;
    for(;i<49;i++)
    {
     if((leftblack[i]-leftblack[i-1])>1 || (leftblack[i]-leftblack[i-1])<0)
     {
        leftblack[i+1]=leftblack[i]-1;
     }
     else
     {
        leftblack[i+1]=leftblack[i]+leftblack[i]-leftblack[i-1];
     }
    }
  }
 }
else if(double_start_miss>40 && double_miss>15)
{
  RoadState[4]=3;//出十字补线
 if(rightblock)
 {
      i=rightblock;
      while(rightblack[i+1]-rightblack[i]>2)
      {
       i--;
      }
      right_end_miss_hang=i;
      right_end_miss_lie=rightblack[i];
      for(i=49;i>right_end_miss_hang;i--)
      {
        increase = 159-right_end_miss_lie;
        rightblack[i] = 159-(49-i)*increase/(49-right_end_miss_hang);
        if(rightblack[i]>158){rightblack[i]=159;}
        else if(rightblack[i]<2){rightblack[i]=0;}
      }
 }
 if(leftblock)
 {
      i=leftblock;
      while(leftblack[i]-leftblack[i+1]>2)
      {
       i--;
      }
      left_end_miss_hang=i;
      left_end_miss_lie=leftblack[i];
      for(i=49;i>left_end_miss_hang;i--)
      {
        increase = left_end_miss_lie-0;
        leftblack[i] = 0-(49-i)*increase/(left_end_miss_hang-49);
```

```
            if(leftblack[i]<2){leftblack[i]=0;}
          }
     }
    }


 /************************************
 确定弯道参数
 ************************************/
//  if(outtrack>=3 && left_H==0 && right_H==0)
  if( left_H==0 && right_H==0)
  {
    for(j=0;j<8;j++)
    {
      if(leftblack[outtrack+j]<3)
      {
       left_H++;
      }
      if(rightblack[outtrack+j]>157)
      {
       right_H++;
      }
    }
    if(left_H>2 && right_H>2)//如果左右都有边界，认为不需要二次补线
    {
      left_H=0;
      right_H=0;
    }
    else if(left_H>2)
    {
      i=outtrack;
      while(leftblack[i]>=3)
      {
       i++;
      }
      while(leftblack[i]<3 && i<47)
      {
       i++;
      }
      left_H=i;
      right_H=0;
    }
    else if(right_H>2)
    {
      i=outtrack;
      while(rightblack[i]<=157)
      {
       i++;
      }
      while(rightblack[i]>157 && i<47)
```

```
        {
          i++;
        }
        left_H=0;
        right_H=i;
      }
    }
    /**********************
      障碍
    **************************/
    temp_temp=error_left;
    temp_temp=error_right;
    // if(img[48][0]!=white &&img[48][159]!=white &&outtrack == 0 &&leftblack[49]<leftblack[1]
&& leftblack[1]<rightblack[1] && rightblack[1]<rightblack[49] &&( right_miss_number==0 ||
left_miss_number==0) /*&& left_H==0 && right_H==0*/)
    if(double_miss<3 &&error_left<17 && error_right<17&& left_state>0 && right_state<0
&&outtrack == 0 &&leftblack[49]<leftblack[1] && leftblack[1]<rightblack[1] &&
rightblack[1]<rightblack[49] &&( right_miss_number==0 || left_miss_number==0))
    {
      for(i=49;i>10;i--)
      {
        for(z=1;z<30;z++)
        {
          temp_temp=0;
          temp_temp2=0;
          if(black ==img[i][leftblack[i]+z])
          {
            for(j=0;j<5;j++)
            {
              if(white==img[i-j][rightblack[i]-z]&&black==img[i-3][leftblack[i]+z+2])
              {
                temp_temp++;
              }
            }
            if(temp_temp==5)
            {
              temp_temp=0;
              zhangai_flag_left=1;
              gpio_init (PTA16, GPO, 0);
              left_i=i;
            }
          }
          if(black ==img[i][rightblack[i]-z])
          {
            for(j=0;j<5;j++)
            {
              if(white==img[i-j][leftblack[i]+z]&&black==img[i-3][rightblack[i]-z-2])
              {
                temp_temp2++;
              }
```

```
            }
            if(temp_temp2==5)
            {
               temp_temp2=0;
               zhangai_flag_right=1;
               gpio_init (PTA16, GPO, 0);
               right_i=i;
            }
         }
      }
      if( zhangai_flag_left==1)
      {
         for(j=leftblack[i]+z;j<150;j++)
         {
            if(white==img[i][j])
            {
               left_j=j;
               break;
            }
         }
         break;
      }
      if( zhangai_flag_right==1)
      {
         for(j=rightblack[i]-z;j>10;j--)
         {
            if(white==img[i][j])
            {
              right_j=j;
              break;
            }
         }
         break;
      }
    }
}
if(zhangai_flag_right==0&&zhangai_flag_left==0)
{
  gpio_init (PTA16, GPI, 1);
}
///障碍处理开始
    if(left_i)
    {
      RoadState[1]=1;
       for(i=0;i<50;i++)
       {
          leftblack[i] = leftblack[i]+(Block+(170-speedsta)/5);
          lastcenter[i] = (leftblack[i] +rightblack[i])/2;
       }
    }
```

```
      else if(right_i)
      { RoadState[1]=2;
        for(i=0;i<50;i++)
        {
           rightblack[i] = rightblack[i]-(Block+(170-speedsta)/5);
           lastcenter[i] = (leftblack[i] +rightblack[i])/2;
        }
      }
//障碍处理结束
   //寻找左右可用行
   i=49;
   while((leftblack[i]==0 || rightblack[i]==159) && i>0)
   {
      i--;
   }
   middle_use=i;
   if(middle_use<left_H|| middle_use<right_H)
   {
      middle_use = 0;
   }
   //算中线
   if(left_i==0 && right_i==0)
   {
   if(left_H==0 && right_H==0)
   {
      temp_temp=0;
      temp_temp2=0;
      RoadState[5]=0;
      for(i=middle_use;i>0;i--)
        {
           /*if(leftblack[i]==0)
           {
              lastcenter[i]=lastcenter[i+1] + rightblack[i] - rightblack[i+1];
           }
           else if(rightblack[i]==159)
           {
              lastcenter[i]=rightblack[i]-Road[i]-RoadX;
            //   lastcenter[i]=lastcenter[i+1]+leftblack[i]-leftblack[i+1];
           }
           else
           {
              lastcenter[i]=leftblack[i]+Road[i]+RoadX;
              //lastcenter[i]=(leftblack[i]+rightblack[i])>>1;
           }*/
           lastcenter[i]=(leftblack[i]+rightblack[i])>>1;
          /*  if(double_miss<3 && error_left<17 && error_right<17&& left_state>0 &&
right_state<0)
           {
            ;
           }
```

```
            else if(temp_temp>15)
            {
              lastcenter[i]=lastcenter[i]-RoadX;
              gpio_init (PTA14, GPO, 0);
            }
            else if(temp_temp2>15)
            {
              lastcenter[i]=lastcenter[i]+RoadX;
              gpio_init (PTA14, GPO, 0);
            }*/
          }
        if(middle_use>5)
        {
          for(i=middle_use+1;i<50;i++)
          {
            if(leftblack[i]==0 && rightblack[i]==159)
            {
              lastcenter[i]=lastcenter[i-1]+lastcenter[i-2]-lastcenter[i-3];
            }
            else if(leftblack[i]==0)
            {
              //  lastcenter[i]=rightblack[i]-Road[i]-RoadX;
               lastcenter[i]=lastcenter[i-1]+(rightblack[i]-rightblack[i-1])*0.6;
               //temp_temp++;
            }
            else
            {
               //lastcenter[i]=leftblack[i]+Road[i]+RoadX;
               lastcenter[i]=lastcenter[i-1]+(leftblack[i]-leftblack[i-1])*0.6;
              //  temp_temp2++;
            }
          }
        }
        else
        {
          for(i=49;i>0;i--)
          {
            if(leftblack[i]==0 && rightblack[i]==159)
            {
              lastcenter[i]=(leftblack[i]+rightblack[i])>>1;
            }
            else if(leftblack[i]==0)
            {
              lastcenter[i]=lastcenter[i-1]+(rightblack[i]-rightblack[i-1])*0.6;//rightblack[i]-Road[i]-
RoadX;
            }
            else
            {
              lastcenter[i]=lastcenter[i-1]+(leftblack[i]-leftblack[i-
1])*0.6;//leftblack[i]+Road[i]+RoadX;
```

```
            }
          }
        }
      }
      else if(left_H==0)//右弯
      {
        RoadState[5]=2;
        if(middle_use!=0)
        {
          for(i=middle_use;i>right_H;i--)
          {
              lastcenter[i] = (leftblack[i]+rightblack[i])>>1;
          }

          if(middle_use==right_H)
          {
            lastcenter[i]= (leftblack[i]+rightblack[i])>>1;
            i--;
          }

          for(;i>outtrack+3;i--)
          {
           //////////////
              lastcenter[i] = (int16)(lastcenter[i+1] + leftblack[i] - leftblack[i+1]);
            // lastcenter[i] = leftblack[i]+Road[i]+WanDaoXiShu*(right_H-i);
             if(lastcenter[i]>R_lim)
             {lastcenter[i]=R_lim;}
          }
          if(middle_use<49)
         {
            for(i=middle_use+1;i<50;i++)
            {
               if(leftblack[i]==0 && rightblack[i]==159)
                {
                   lastcenter[i] = lastcenter[i-1]+(lastcenter[i-2]+lastcenter[i-3]-lastcenter[i-4]-
lastcenter[i-5])/4;
                }
                else if(leftblack[i]==0)
                {
                   lastcenter[i] = lastcenter[i-1]+(rightblack[i]-rightblack[i-1])*0.6;
                }
                else if(rightblack[i]==159)
                {
                   lastcenter[i] = lastcenter[i-1]+(leftblack[i]-leftblack[i-1])*0.6;
                }
            }
          }
        }
        else
        {
```

```
        i=49;
        while(leftblack[i]==0 && i>0)
        {
           i--;
        }
        middle_use=i;
        right_H=i-1;

        lastcenter[i]=leftblack[i]+Road[i]+5;
         i--;
        for(;i>outtrack+3;i--)
        {
          ///////

           //lastcenter[i] = leftblack[i]+Road[i]+WanDaoXiShu*(right_H-i);
           lastcenter[i] = (int16)(lastcenter[i+1] + leftblack[i] - leftblack[i+1]);//-temp_temp);
           if(lastcenter[i]>R_lim)
           {lastcenter[i]=R_lim;}
        }
        if(middle_use<49)
       {
          for(i=middle_use+1;i<50;i++)
          {
             if(leftblack[i]==0 && rightblack[i]==159)
              {
                 lastcenter[i] = lastcenter[i-1]+(lastcenter[i-2]+lastcenter[i-3]-lastcenter[i-4]-
lastcenter[i-5])/4;
              }
              else if(leftblack[i]==0)
              {
                 lastcenter[i] = rightblack[i]-Road[i];//lastcenter[i-1]+rightblack[i]-rightblack[i-1];
              }
              else if(rightblack[i]==159)
              {
                 lastcenter[i] = leftblack[i]+Road[i];//lastcenter[i-1]+leftblack[i]-leftblack[i-1];
              }
          }
       }
    }
  }
  else if(right_H==0)//左弯
  {
     RoadState[5]=1;
     if(middle_use!=0)
     {
        for(i=middle_use;i>left_H;i--)
        {
            lastcenter[i] = (leftblack[i]+rightblack[i])>>1;
        }
        if(middle_use==left_H)
```

```
        {
           lastcenter[i] = (leftblack[i]+rightblack[i])>>1;
           i--;
        }
        for(;i>outtrack+3;i--)
        {
          ////////////////
          lastcenter[i] = (int16)(lastcenter[i+1] + rightblack[i] - rightblack[i+1]);//-temp_temp);
         //   lastcenter[i] = rightblack[i]-Road[i]-WanDaoXiShu*(left_H-i);
           if(lastcenter[i]<L_lim)
           {lastcenter[i]=L_lim;}
        }
         if(middle_use<49)
       {
          for(i=middle_use+1;i<50;i++)
          {
             if(leftblack[i]==0 && rightblack[i]==159)
             {
                lastcenter[i] = lastcenter[i-1]+(lastcenter[i-2]+lastcenter[i-3]-lastcenter[i-4]-
lastcenter[i-5])/4;
             }
             else if(leftblack[i]==0)
             {
                lastcenter[i] = lastcenter[i-1]+(rightblack[i]-rightblack[i-1])*0.6;
             }
             else if(rightblack[i]==159)
             {
                lastcenter[i] = lastcenter[i-1]+(leftblack[i]-leftblack[i-1])*0.6;
             }
          }
       }
     }
     else
     {
        i=49;
        while(rightblack[i]==159 && i>0)
        {
           i--;
        }
        middle_use=i;
        left_H=i-1;
        lastcenter[i]=rightblack[i]-Road[i]-5;
        i--;
        for(;i>outtrack+3;i--)
        {
        ////////////////////
            lastcenter[i] = (int16)(lastcenter[i+1] + rightblack[i] - rightblack[i+1]);//-temp_temp);
          // lastcenter[i] = rightblack[i]-Road[i]-WanDaoXiShu*(left_H-i);
           if(lastcenter[i]<(L_lim))
           {lastcenter[i]=L_lim;}
```

```
        }
         if(middle_use<49)
        {
          for(i=middle_use+1;i<50;i++)
          {
            if(leftblack[i]==0 && rightblack[i]==159)
            {
              lastcenter[i] = lastcenter[i-1]+(lastcenter[i-2]+lastcenter[i-3]-lastcenter[i-4]-
lastcenter[i-5])/4;
            }
            else if(leftblack[i]==0)
            {
              lastcenter[i] = rightblack[i]-Road[i];//lastcenter[i-1]+rightblack[i]-rightblack[i-1];
            }
            else if(rightblack[i]==159)
            {
              lastcenter[i] = leftblack[i]+Road[i];//lastcenter[i-1]+leftblack[i]-leftblack[i-1];
            }
          }
        }
      }
    }
    }
    //补全中线
    if(outtrack)
    {
        temp_temp=(lastcenter[outtrack+4]-lastcenter[outtrack+7])/3;

        for(i=outtrack+3;i>0;i--)
        {
          lastcenter[i]=lastcenter[i+1]+temp_temp;
        }
     /* if(error_left>error_right)
      {
        for(i=outtrack+3;i>0;i--)
        {
          temp_temp2=(uint8)(speedmax*(right_H-i)*(right_H-i)/WanDaoXiShu);
          lastcenter[i]=lastcenter[i+1]+temp_temp-temp_temp2;
        //  lastcenter[i]=lastcenter[i+1]+1;
        }
      }
      else if(error_right>error_left)
      {
        for(i=outtrack+3;i>0;i--)
        {
          temp_temp2=(uint8)(speedmax*(left_H-i)*(left_H-i)/WanDaoXiShu);
          lastcenter[i]=lastcenter[i+1]+temp_temp-temp_temp2;
    //       lastcenter[i]=lastcenter[i+1]-1;
        }
      }*/
```

```
  }


  //display

//  //计算远近分段弯曲程度
//   ave=0;
//   for(i=6;i>3;i--)
//   {
//
//    ave=lastcenter[i]-79;
//    if(ave<0)
//    {ave=-ave;
//    }
//    ave+=ave;
//   }
//   f_ave=ave/3;
//   ave=0;
//   for(i=45;i>40;i--)
//   {
//    ave=lastcenter[i]-79;
//    if(ave<0)
//    {
//     ave=-ave;
//    }
//    ave+=ave;
//   }
//   c_ave=ave/5;
//
//  display_small_numberstr(70, 3, f_ave,4);
//display_small_numberstr(70, 4, c_ave,4);
//    if(f_ave>15 && f_ave<90 && c_ave<5 )
//    {
//     chuwan=0;
//     ruwan++;
//     if(ruwan>3)
//     {
//        state = 2;
//       // think_speed = speedmin;
//        ruwan=0;
//     }
//    }
//    else if((state==3&&f_ave<=50&&c_ave>10)||(f_ave<15&&c_ave<5))
//    {
//       ruwan=0;
//       chuwan++;
//       if(chuwan>3)
//       {
//         state = 1;
//        // think_speed = speedmax;
```

```
//          chuwan=0;
//        }
//     }
//     else
//    {
//       state=3;
//      // think_speed = speedsta;
//    }
    error_center=0;
   //计算 error_center,用于弯道增强
    if(outtrack<40)
    {
      for(i=49;i>39;i--)
      {
         average_center += lastcenter[i];
      }
      average_center /= 10;

       for(i=49;i>40;i--)
      {
         temp_error_center = lastcenter[i]-average_center;
         if(temp_error_center>=0)
         {
            error_center += temp_error_center;
         }
         else
         {
            error_center -= temp_error_center;
         }
      }
    }
    else
    {
       for(i=49;i>outtrack;i--)
      {
         average_center += lastcenter[i];
      }
      average_center /= 49-outtrack;
       for(i=49;i>outtrack;i--)
      {
         temp_error_center = lastcenter[i]-average_center;
         if(temp_error_center>=0)
         {
            error_center += temp_error_center;
         }
         else
         {
            error_center -= temp_error_center;
         }
      }
```

```
    }

  // error_center /= 15;
   //障碍

    //上色
   for(i=50-1;i>outtrack;i--)
   {
      img[i][lastcenter[i]] = 0;
      if(lastcenter[i]>0)
      {
         img[i][lastcenter[i]-1] = 0;
      }
      if(lastcenter[i]<159)
      {
         img[i][lastcenter[i]+1] = 0;
      }
      img[i][leftblack[i]] = 170;

      img[i][rightblack[i]] = 170;
  //    img[i][79] = 0;
 //     img[i][80] = 170;

   }
   for(j=0;j<159;j++)
   {
     img[20][j]=170;img[40][j]=170;img[45][j]=170;
     img[25][j]=170;
   }

   for(i=0;i<6;i++)
   {
      temp_temp=right_H;
      left_j=RoadState[i];
   }

 //display_small_numberstr(70, 2, error_center,4);
   //发夹弯补偿
   if((RoadState[5]==1 && error_left>error_right/* && error_left>=5*/)||(RoadState[5]==2 &&
error_right>error_left/* &&
error_right>=5*/)||(RoadState[5]==0&&(error_left>(error_right+20)||(error_right>(error_left+
20)))))
   {
     hairpin=1;
         gpio_init (PTA16, GPO, 0);
   }
   else
   {
     hairpin=0;
     gpio_init (PTA16, GPO, 1);}
```

```
  /*
RoadState[0]坡道：
      0 无
      1 检测到坡道
RoadState[1]障碍：
      0 无
      1 左障碍
      2 右障碍
RoadState[2]右边突变：
      0 无突变
      1 找到终点并补线
      2 没找到终点且左边线不在边界（右转）
      3 j==0||outtrack!=0||temp_temp!=80(出十字当做弯道处理)
      4 入十字，斜率为 1，盲补（待修，行数应较远）
            5 右突变急弯
RoadState[3]左边突变：
      0 无突变
      1 找到终点并补线
      2 没找到终点且右边线不在边界（左转）
      3 j==0||outtrack!=0||temp_temp!=80(出十字当做弯道处理)
      4 入十字，斜率为 1，盲补（待修，行数应较远）
            5 左突变急弯
RoadState[4]出十字方块/补线：
      0 无此路况
      1 方块，且赛道在方块左边
      2 赛道在方块右边
      3 出十字补线
RoadState[5]算中线：
      0 直道
      1 左弯
      2 右弯
   */
}
```