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：东北大学
队伍名称：一叶一世界
参赛队员：薛喜地
母德东
滕璞骏
带队教师：闻时光

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，比赛组委会和恩智浦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：薛嘉旭

陆东

滕璞骏

带队教师签名：闻财光

日期：2016.08.15

摘　　要

本文设计的智能车系统以 MKL26Z256VLH4 微控制器为核心控制单元，通过 CMOS 摄像头检测赛道信息，使用模拟比较器对图像进行硬件二值化，提取黑色双边界，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，使用上位机、蓝牙模块、按键模块等调试工具，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：MKL26Z256VLH4，CMOS，PID

Abstract

In this paper, the design of the smart car system to MKL26Z256VLH4 micro controller as the core control unit, through CMOS camera to detect the track information, using analog comparator hardware binarization of the image, extraction double black border and used to identify the track; through the real-time speed of encoder detection model car, using PID control algorithm to adjust the drive motor speed and steering angle, the realization of the closed-loop control on the model of vehicle movement speed and direction of movement. In order to improve the speed and stability of the model car, a lot of hardware and software testing are carried out by using the debugging tools such as host computer, Bluetooth module, key module and so on. Experimental results show that the design of the system is feasible

Keywords: MKL26Z256VLH4, CMOS, PID

目录

引　　言.....1
第一章 系统总体方案设计.....3
1.1 系统总体结构.....3
1.2　整车布局.....4
第二章 机械结构设计.....5
2.1 智能车机械参数调节.....5
2.1.1 前轮调整.....5
2.1.2 其他部分调整.....7
2.2 底盘高度的调整.....7
2.3 编码器的安装.....7
2.4 舵机转向结构的调整.....8
2.5 舵机的安装.....10
2.6 摄像头的安装.....10
第三章 硬件电路设计与实现.....11
3.1 硬件方案设计.....11
3.2 MCU 主控模块.....11
3.2.1 KL26 最小系统.....11
3.2.2 KL26 系列的片上资源.....11
3.2.3 单片机片上资源.....12
3.3 传感器.....13
3.3.1 数字摄像头 鹰眼 OV7725.....13
3.3.2 编码器.....14
3.4 电源管理模块.....14
3.5 电机驱动模块.....15
3.5.1 H 桥电机驱动芯片.....15
3.5.2 升压电源变换.....16
3.6 PCB 板设计.....16
第四章 系统软件设计及实现.....17
4.1 赛道中心线提取及优化处理.....17
4.1.1 原始图像的特点.....17
4.1.2 赛道边沿提取.....18
4.1.3 图像校正.....19
4.1.4 推算中心.....20
4.1.5 路径选择.....21
4.2 PID 控制算法介绍.....22
4.2.1 位置式 PID.....23
4.2.2 增量式 PID.....24
4.2.3 PID 的参数整定.....24
第五章 系统开发及调试工具.....25
5.1 开发工具.....25
5.2 仿真工具.....25
5.3 上位机图像显示.....26
5.3.1 山外多功能上位机.....26

5.4 蓝牙调试工具..... 27
5.5 拨码开关调试工具..... 27
第六章　模型车的主要技术参数..... 29
参 考 文 献..... 31
附录：程序源代码..... 33

引　　言

随着科学技术的不断发展进步，智能控制的应用越来越广泛，几乎渗透到所有领域。智能车技术依托于智能控制，前景广阔且发展迅速。目前，掌握着汽车工业关键技术的发达国家已经开发了许多智能车的实验平台和商品化的车辆辅助驾驶系统。有研究认为智能汽车作为一种全新的汽车概念和汽车产品， 在不久的将来会成为汽车生产和汽车市场的主流产品。

面向大学生的智能汽车竞赛最早始于韩国，在国内，全国大学生“飞思卡尔”杯智能汽车竞赛从 2006 年开始已经举办了十届，得到了各级领导及各高校师生的高度评价。大赛为智能车领域培养了大量后备人才，为大学生提供了一个充分展示想象力和创造力的舞台,吸引着越来越多来自不同专业的大学生参与其中。

全国大学生“飞思卡尔”杯智能汽车竞赛包括光电组、摄像头组和电磁组，其中数摄像头组的智能车速度最快，备受关注。

本技术报告主要包括机械系统、硬件系统、软件系统等，详尽地阐述了我们的设计方案，具体表现在硬件电路的创新设计以及控制算法的独特想法。智能车的制作过程包含着我们的辛勤努力，这份报告凝聚了我们智慧，是我们团队共同努力的成果。

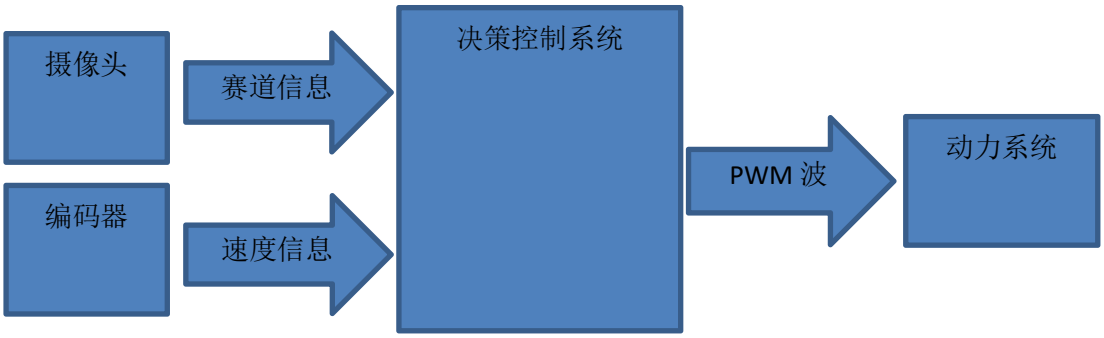
在准备比赛的过程中，我们小组成员涉猎控制、模式识别、传感器技术、汽车电子、电气、计算机、机械等多个学科，几个月来的经历，培养了我们电路设计、软件编程、系统调试等方面的能力，锻炼了我们知识融合、实践动手的能力，对今后的学习工作都有着重大的实际意义。

第一章 系统总体方案设计

1.1 系统总体结构

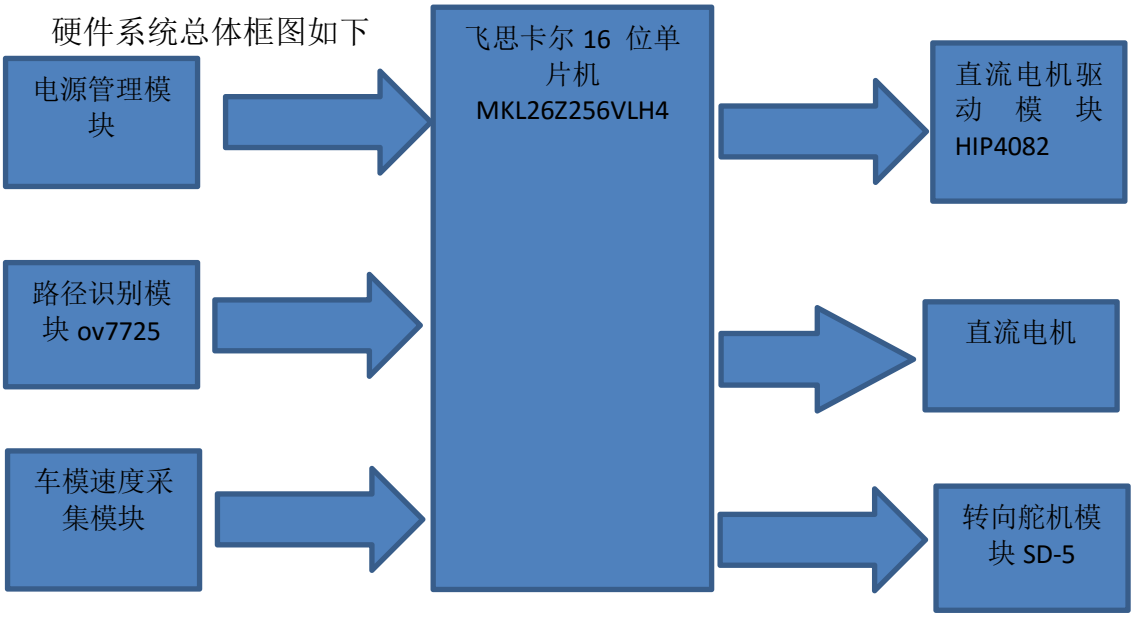
智能车主要由三个部分组成：检测系统，控制决策系统，动力系统。其中检测系统采用 CMOS 数字摄像头鹰眼 OV7725，控制决策系统用 KL26Z 作为主控芯片，动力系统主要控制舵机的转角和直流电机的转速。整体的流程为，通过视觉传感器来检测前方的赛道信息，并将赛道信息发送给单片机。同时，通过光电编码器构成的反馈渠道将车体的行驶速度信息传送给主控单片机。根据所取得的赛道信息和车体当前的速度信息，由主控单片机做出决策，并通过 PWM 信号控制直流电机和舵机进行相应动作，从而实现车体的转向控制和速度控制。

系统总体结构框图如下



赛车的硬件电路主要有七个部分组成：MKL26Z256VLH4 最小系统板，电源管理模块，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，辅助调试模块。

硬件系统总体框图如下



（1）MKL26Z256VLH4 最小系统板是系统的核心部分，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模块进行控制。

（2）电源管理模块给整个系统供电，保障系统安全稳定运行。

（3）图像采样处理模块采用数字摄像头鹰眼 OV7725，用于获得前方道路情况以供单片机处理，是智能小车的“视觉系统”。

（4）速度检测电路采用欧姆龙公司的 512 线光电编码器。

（5）电机驱动电路采用 1 个 HIP4082 构成全桥芯片驱动，通过输出 PWM 信号产生的平均电压来驱动直流电机。

（6）舵机驱动模块控制舵机的转向。

（7）辅助调试模块有蓝牙,串行通信,J-link 等，主要用于赛车系统的程序烧写、功能调试、赛车状态监控等方面。

1.2 整车布局

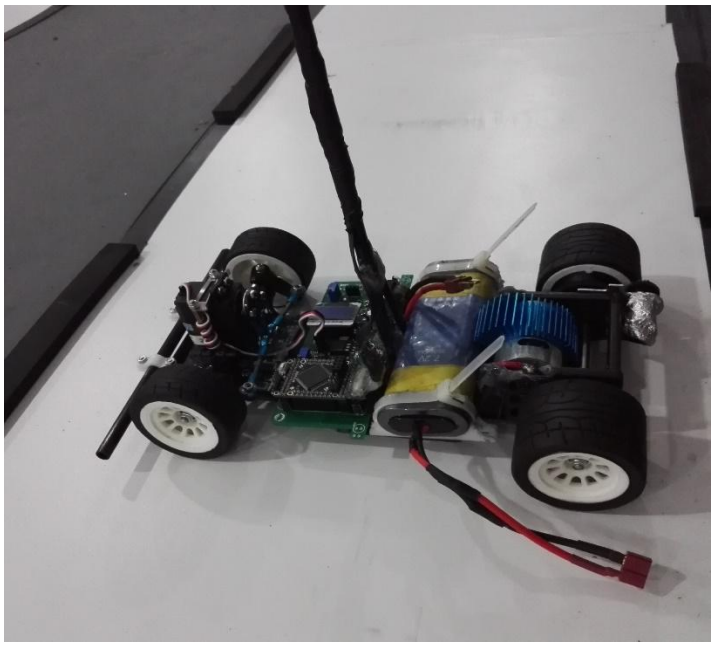
模型车的整车布局本着轻量化设计，如图具有以下特点：

（1）架高舵机并直立安装，以提高舵机响应速度；

（2）主板低位放置，降低赛车重心；

（3）采用强度高、质量轻的材料制作摄像头支架；

（4）摄像头后置于模型车的中间部分，减少赛车前方盲区。



第二章 机械结构设计

根据组委会的相关规定，今年摄像头组比赛车模更换为B型车模。针对不同的车模，必然会有不同的调整方式。在比赛备战之初，我们就对该车模进行了详细的系统分析。新型的B型车模精度相当差，因此在规则允许范围内尽量改造车模，提高车模整体精度是很必要的。本章将主要介绍智能汽车车模的机械结构及调整方案。

2.1 智能车机械参数调节

为保证智能小车直线行驶稳定,转向轻便灵活并尽可能的减少轮胎磨损，需要对小车的四轮定位参数进行调整。四轮定位内容主要有：主销后倾角，主销内倾角，前轮外倾角，前轮前束，外侧车轮二十度时，内外转向轮转角差，后轮外倾角，后轮前束。其中，前轮定位的参数对小车性能有着至关重要的影响，这四个参数反映了前轮、主销和前轴三者之间在车架上的位置关系。本节将对这四个参数做详细阐述如下：

2.1.1 前轮调整

小车在调试过程中，转向轮定位参数是很重要的因素，它通常不易被察觉，但是却有着较大的危害。如果取得不恰当，那么将造成转向不灵活，效率低以及转向轮侧滑等问题，使得小车性能下降，加速轮胎的磨损。

转向轮定位参数包括：主销内倾角、主销后倾角、转向轮外倾角及转向轮前束。这其中最重要的就是转向轮外倾角和转向轮前束。

主销内倾是指主销装在前轴略向内倾斜的角度，它的作用是使前轮自动回正，如图3.1.1。内倾角度越大前轮自动回正的作用就越强烈，但转向时也越费力，轮胎磨损增大；反之，角度越小前轮自动回正的作用就越弱。我们发现，主销内倾一定角度可以提高过弯上限。

主销后倾(Caster)是指主销装在前轴,上端略向后倾斜的角度,如图3.1.1。它使车辆转弯时产生的离心力所形成的力矩方向与车轮偏转方向相反,迫使车轮偏转后自动恢复到原来的中间位置上。由此主销后倾角越大，车速越高，前轮稳定性也愈好。主销内倾和主销后倾都有使汽车转向自动回正，保持直线行驶的功能。不同之处是主销内倾的回正与车速无关，主销后倾的回正与车速有关，因此高速时后倾的回正作用大，低速时内倾的回正作用大。

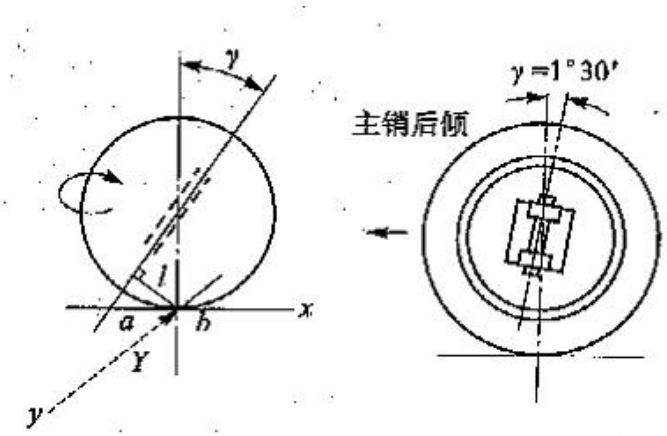


图 3.1.1 主销后倾角

车轮外倾角（Camber）是指从前放看前轴时，轮胎的中心平面不是垂直的，而是上面向外倾斜一个角度，如图 3.1.2。设置转向轮的外倾角是为了平衡和协调因为车重造成的前轮内倾倾向，使轮胎和路面呈垂直接触的最佳状态。

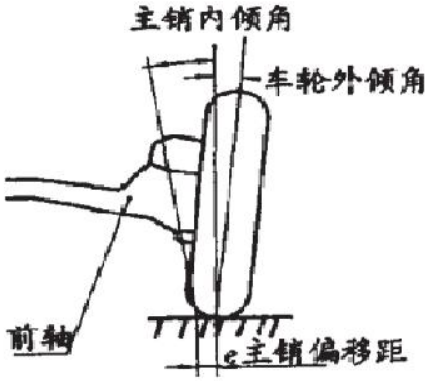


图 3.1.2 主销内倾角和车轮外倾角

转向轮前束(Toe out)是指同一轴两端车轮轮辋内侧轮廓线的水平直径的端点为等腰梯形的顶点，底边为车轮轴线。等腰梯形两底边长度之差为前束。如图 3.1.3 所示，当梯形前低边小于后底边时，前束为正（A<B），反之为负。车轮的水平直径与纵向平面之间的夹角为前束角。正的前束角在车轮中心产生向内的侧向力，而正的外倾角在车轮中心产生向外的侧向力，因此前束角的作用是与外倾角协调，保持车轮做纯滚动和直线行驶，从而减少轮胎磨损，提高汽车的操纵稳定性。

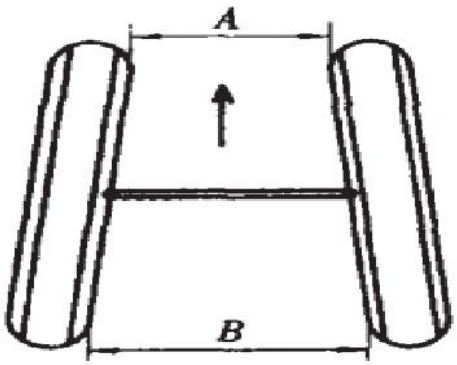


图 3.1.3 前束[4]

2.1.2 其他部分调整

其他部分调整主要涉及到小车底盘高度、小车重心位置、减震器、齿轮咬合、差速器等。具体调整如下：

1、底盘高度调整：底盘高度可以影响重心。适当降低底盘高度可以使小车重心降低，有利于过弯稳定。实际调整可以通过调整前轮高度、后轮轴高度调节块等方式来调节。

2、重心位置：重心位置同样影响小车性能。重心过前，增加转向阻力，引起转向迟滞。另外，如果小车速度很快的情况下，上下坡道的时候会造成前轮首先着地，很可能造成小车意外事故。重心过后，则会使小车前轮抓地不足，造成过弯非常不稳定。实际调整以重心在电池处为准，保持各部分重量均衡。

3、前轮悬挂系统的调整：前轮的悬挂系统弹簧对前轮抓地力有一定影响，但是为了防止摄像头抖动过于剧烈，我们选择将弹簧锁紧。

4、减震器弹簧调整：在此我们直接采用硬链接。

5、齿轮咬合调整：调整齿轮咬合，以不松动，无卡滞，松紧合适为准。另外还要保证齿轮间咬合有足够的接触面积。

6、差速器调整：合适的差速器调整能够提高小车入弯速度，提高弯道性能。差速器调整可以通过右后轮轮轴上面的螺丝。注意调整过松，会严重影响直道加速性能；调整过紧则会使差速器处于无效状态，影响小车在过弯时的速度，并可能会照成甩尾。

2.2 底盘高度的调整

在保证顺利通过坡道的前提下,底盘尽量降低,从整体上降低模型车的重心,可使模型车转弯时更加稳定、高速。

2.3 编码器的安装

选用编码器进行速度的测量。根据编码器的形状，我们自制了一个支架，速

度传感器用螺钉通过支架固定在后轮支架上，这样固定好之后，就有了较高的稳定性。然后调节编码器齿轮，使其与差速齿轮紧密咬合，增大测速的精确性，但是咬合过紧也增大了摩擦，减小了对电机做功的利用率，影响小车的快速行驶，因此减小摩擦同时增强齿轮间的咬合是我们主要考虑的因素。如图 3.3 所示。

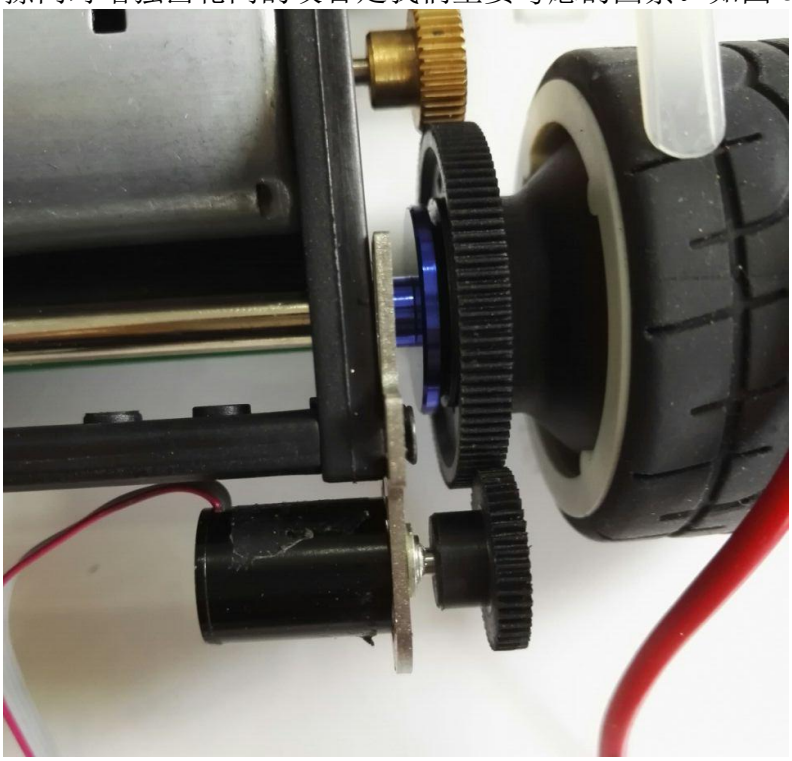


图 3.3　编码器安装

2.4 舵机转向结构的调整

车模转向由前轮舵机经过连杆将转动转变为平行四边形双摇杆结构的转动，最后由平行四边形双摇杆机构带动前轮的左右摆动实现转向。经过分析和测试，发现此转向机构结构相对复杂，效率底下。转向机构原理图 2.4.1。

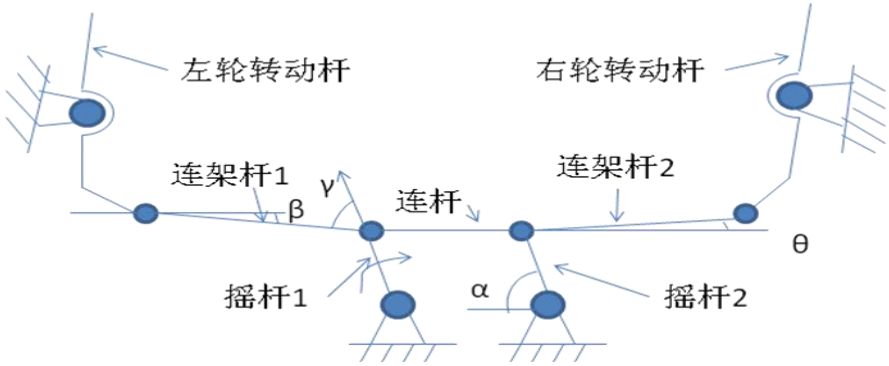


图 2.4.1　转向机构原理图

图 2.4.1 取车体为参考系，各固定铰相对车体静止。舵机连杆输出推动平行四边形双摇杆机构转动进而带动前轮转动。 β 角为压力角， γ 角为传动角，他们共同决定了舵机效率的高低， α 角反映了舵机的转角。

由于对称性，故只讨论 α 在 0-90 度的情况，在舵机处于中心位置时角为 0 度，角为 90 度，α 角为 90 度，此时舵机效率最高。随着 α 角的减小，传动角减小，而压力角则增大，根据上式可以看出，效率将迅速减小。当角到达 0 度，即平行四边形双摇杆机构摇杆 1 与连接杆 1 在同一直线上时，此时左轮转动杆转角达到最大值；而当连架杆 2 和右轮转动杆的夹角等于 180 度时，α 角将不能在减小，此时到达舵机的左极限，同理如果 α 增大也会存在上述两个极限点。由于上述两个极限点并不是同时到达，甚至还会造成一小段时间内左右轮转向不同的问题。

对于 B 车来说，转向角度越大越好，但是从对平行四边形双摇杆机构的分析来看，随着 α 角的减小， θ 角将增大，当到达极限位置时， θ 角也到达极限，也就是右轮的右极限。而对于现有的安装方式角是比较大的，因此这也就限定了理论上的转角。而车模本身的限制不仅仅是理论上的，由于各个部件尺寸并不是很合适，造成了转向机构的机械限位，因此在不影响车模强度的情况下，修改和更换掉一些部件，将这些机械限位去掉。

为了最大限度的增大转向范围，设计了一种简单且效率较高的方案—长连杆方案，如图 2.4.2。

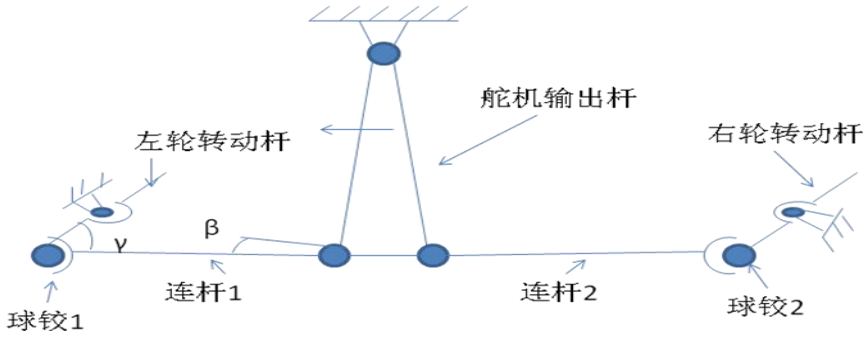


图 2.4.2 长连杆方案原理图

长连杆方案的优点是：

一、可以根据需要选择舵机输出杆的长度，从而获得所需要的灵敏度，但是舵机输出杆的长度也不能太长，因为这会对舵机的输出力矩有较高的要求，太长会烧坏舵机。

二、效率较高，舵机的单边（例如取左边）效率，对于长连杆方案来说，左轮角在增大的同时，右轮角在减小，而且角是在 0-45 度之间，同时的变动也比较小（45 度-135 度），因此长连杆的效率变动较小且效率较高。

三、转角大，由于长连杆方案中舵机输出杆的转动在同一个平面内，当其到达极限位置时，转角比平行四边形方案要大。

四、转向更灵敏，因为放大倍数较平行四边形方案要大。

2.5 舵机的安装

舵机安装我们采用较为经典的立式安装，在力矩一定的情况下，响应速度较为快。

舵机的安装如图：

2.6 摄像头的安装

为了降低整车重心，需要严格控制 CMOS 摄像头的安装位置和重量，我们采用了碳纤维管作为安装 CMOS 的主桅，这样可以获得最大的刚度质量比，整套装置具有很高的定位精度和刚度，使摄像头便于拆卸和维修，具有赛场快速保障能力。摄像头的安装如图 2.5 所示。

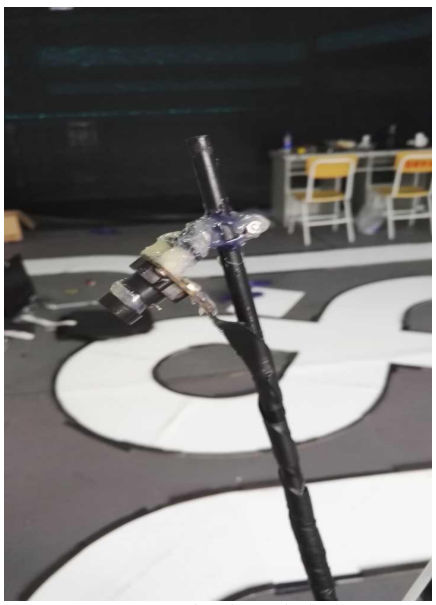


图 2.5 摄像头安装

第三章 硬件电路设计与实现

3.1 硬件方案设计

我们主要从系统的稳定性、可靠性、高效性、实用性、简洁等方面来考虑硬件的整体设计。从最初方案设定到最终方案的敲定，我们经历各种讨论与大的改动才有了如下的硬件方案。

可靠性是系统设计的第一要求,我们对电路设计的所有环节都进行了电磁兼容性设计，做好各部分的接地、屏蔽、滤波等工作，将高速数字电路与模拟电路分开，使本系统工作的可靠性达到了设计要求。

3.2 MCU 主控模块

本智能小车系统采用的是飞思卡尔 16 位单片机 MKL26Z256VLH4。

3.2.1 KL26 最小系统

2012 年 6 月， 飞思卡尔半导体公司率先推出了业内首款基于 ARMCortex-M0+处理器内核的超低功耗单片机——Kinetis L 系列产品。Cortex-M0+内核的功耗大约是现在任意一款 8 位或 16 位处理器的三分之一,但性能却提高了 2-40 倍。Kinetis L 系列单片机在超低功耗运行（VLPR）模式下的功耗仅为 50μA/MHz。该系列单片机还提供了多种睡眠模式，单片机可根据不同的中断唤醒源，从睡眠模式迅速切换到工作模式，待处理完数据之后再迅速返回到睡眠状态，从而延长了电池的使用寿命。因此 Kinetis L 系列单片机是小家电，游戏机配件，个人电脑外设，便携式医疗系统，音频系统，智能电表，照明和数字电源等产品开发的不二选择。

3.2.2 KL26 系列的片上资源

（1）16 位 ADC，可设置分辨率，采样时间和转换速度及功耗，具有单端和差分模式供用户使用；

（2）片上集成有温度传感器，可测量芯片运行时的温度；

（3）高速电压比较器，并内置一个 6 位 DAC，可作为电压比较器的输入源；

（4）支持 DMA 功能的 12 位 DAC；

（5）1 个 6 通道和两个 2 通道的 16 位低功耗定时器 PWM 模块，并支持 DMA 功能；

（6）2 通道 32 位周期中断定时器（PIT）可为实时操作系统任务调度提供时基（tick），或为 ADC 转换提供触发信号源；

（7）低功耗定时器（low power timer）能够在除 VLLS0 外的所有睡眠或

工作模式下运行，可用于在睡眠模式下的周期性唤醒单片机；

（8）带日历的实时时钟（RTC）；

（9）电容式触摸传感器接口支持多达 16 个外部电极，并支持 DMA 功能；

（10）通用 I/O 均带有外部中断触发和 DMA 请求功能；

（11）两个支持 DMA 功能的 I2C 通信接口，最高通信率可达到 100kbps，并与 SMBus V2 兼容；

（12）一个支持 DMA 功能的低功耗 UART；

（13）两个通用型 UART 通信接口；

（14）两个支持 DMA 功能的 SPI 通信接口；

（15）一个面向音频应用的 I2S 模块；

（16)USB2.0 On-The-Go 全速接口，并集成了 USB 电压稳压器，可提供 120mA 的输出电流。

3.2.3 单片机片上资源

单片机的硬件资源是有限的，合理的利用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，既涉及到信号的采集，同时涉及到控制部分。在采集部分，使用的是中断捕捉方式进行，而在控制部分，使用的是单片机自带的 PWM 控制波。以下列出单片机硬件资源和 I/O 口分配表，如表 4.1 所示。

图像采集模块	I/O	A 口用于数据接收，PTC12，PTC10 和PTC13 用于中断信号的捕捉
	硬件资源	其中A 作为普通I/O，只负责数据接收，PT0 和PT1采用输入捕捉方式，捕捉摄像头的行、场同步信号，完成对图像数据的采集
速度反馈	I/O	PTC5 与 PTE23 接入编码器
	硬件资源	PTC5 口是LPTMR 脉冲计数输入管脚选项用于捕获脉冲个数，PTE23用于检测正反转
舵机控制	I/O	PTB0
	硬件资源	选用 TPM1 模块的 TPM_CH0 通道，输出 16 位的 PWM。
电机驱动	I/O	PTE24，PTE30
	硬件资源	PTE24， PTE30分别接TPM0 模块的 TPM_CH0， TPM_CH3，通道，TPM_CH1 与TPM_CH3 通道控制直流电机电机正转反转PWM 信号输出。

辅助调试	I/O	蓝牙模块，J_link 仿真模块
	硬件资源	
串口通讯模块	I/O	RXD0/PS0、TXD0/PS1
	硬件资源	串口接收、串口发送
拨码开关	I/O	PTA1，PTA2，PTA5，PTA16,PTC1，PTC6，PTC8，PTE22
	硬件资源	

3.3 传感器

3.3.1 数字摄像头 鹰眼 OV7725

COMS 与 CCD

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

经过各种参数比对和以往制作的经验，决定采用 CMOS 摄像头，而 CMOS 摄像头又分为数字和模拟两种。我们选用了鹰眼 OV7725 进行实验，对数字摄像头的可行性进行论证。

鹰眼 OV7725 摄像头速率可达 150 帧每秒,通过特殊的硬件结构完成二值化操作，图像处理效果极佳,采集图像速度极快,可以减小车的控制周期，

摄像头由八位数据位 D0 至 D7 向单片机传输图像的二值化数据，单片在摄像头场信号与像素信号的影响下采集图像数据，经解压后处理图像数据。

3.3.2 编码器

光电式脉冲编码器可将机械位移、转角或速度变化转换成电脉冲输出，是精密数控采用的检测传感器。光电编码器的最大特点是非接触式，此外还具有精度高、响应快、可靠性高等特点。缺点是体积较大。

光电编码器采用光电方法，将转角和位移转换为各种代码形式的数字脉冲，在发光元件和光电接收元件中间,有一个直接装在旋转轴上的具有相当数量的透光扇形区的编码盘,在光源经光学系统形成一束平行光投在透光和不透光区的码盘上时，转动码盘，在码盘的另一侧就形成光脉冲，脉冲光照射在光电元件上就产生与之对应的电脉冲信号。

光电编码器的精度和分辨率取决于光电码盘的精度和分辨率，取决于刻线数。目前，已能生产径向线宽为 6.7×10-8 rad 的码盘，其精度达 1×10-8，比接触式的码盘编码器的精度要高很多个数量级。如进一步采用光学分解技术，可

获得更多位的光电编码器。

光电编码器按其结构的转动方式可分为直线型的线性编码器和转角型的轴角编码器两种类型，按脉冲信号的性质可分为有增量式和绝对式两种类型。

增量式编码器码盘图案和光脉冲信号均匀，可将任意位置为基准点，从该点开始按一定量化单位检测。该方案无确定的对应测量点，一旦停电则失掉当前位置，且速度不可超越计数器极限相应速度，此外由于噪声影响可能造成计数积累误差。该方案的优点是其零点可任意预置，且测量速度仅受计数器容量限制。

绝对式编码器的码盘图案不均匀，编码器的码盘与码道位数相等，在相应位置可输出对应的数字码。其优点是坐标固定，与测量以前状态无关，抗干扰能力强，无累积误差，具有断电位置保持，不读数时移动速度可超越极限相应速度，不需方向判别和可逆计数，信号并行传送等；其缺点是结构复杂、价格高。要想提高光电编码器的分辨率,需要提高码道数目或者使用减速齿轮机构组成双码盘机构，将任意位置取作零位时需进行一定的运算。

我们采用欧姆龙增量式 500 线光电编码器，其供电电压为 5V，输出为小幅值的正弦信号。为了将此信号放大整形，设计了信号调理电路，其基本原理是使用一个运放做成比较器电路，调节参考电压，使输出变为 0/5V 的方波信号，进入单片机进行计算。

3.4 电源管理模块

首先了解一下不同电源的特点，电源分为开关电源和线性电源，线性电源的电压反馈电路是工作在线性状态,开关电源是指用于电压调整的管子工作在饱和

和截至区即开关状态的。线性电源一般是将输出电压取样然后与参考电压送入比较电压放大器，此电压放大器的输出作为电压调整管的输入，用以控制调整管使其结电压随输入的变化而变化，从而调整其输出电压，但开关电源是通过改变调整管的开和关的时间即占空比来改变输出电压的。

从其主要特点上看：线性电源技术很成熟，制作成本较低，可以达到很高的稳定度，波纹也很小，而且没有开关电源具有的干扰与噪音，开关电源效率高、损耗小、可以降压也可以升压，但是交流纹波稍大些。

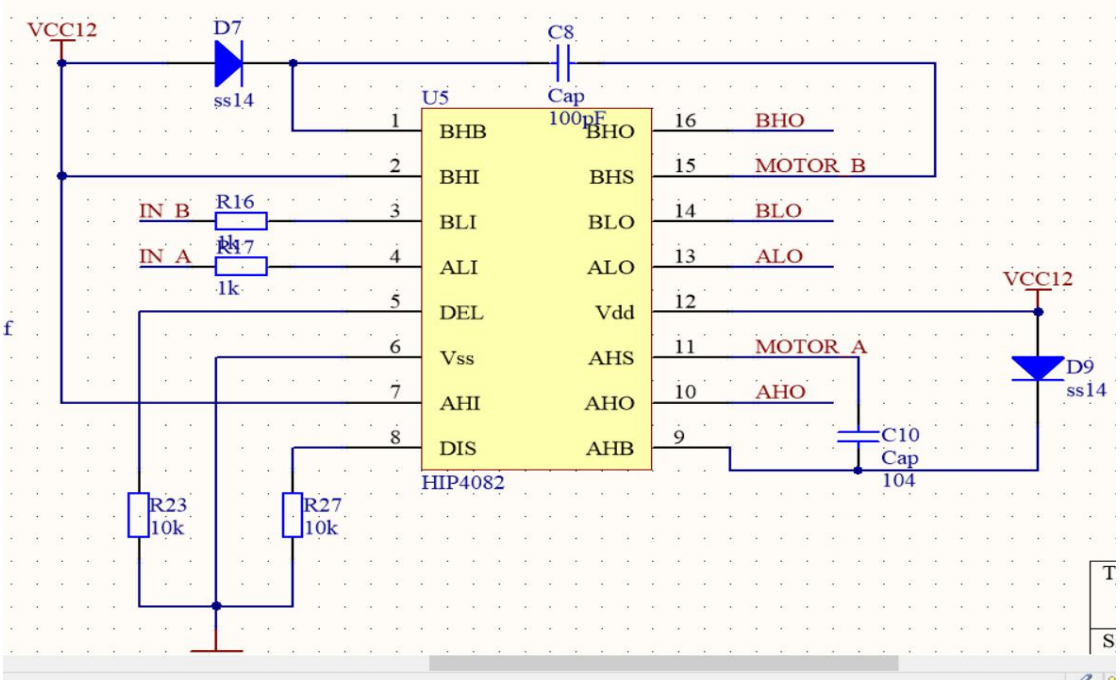
在电源管理模块中，我们选用了 LM2940、LM1084-adj、LM1117、MC34063，其中 3.3V 电压，给单片机、鹰眼摄像头 ov7725 以及 OLED 模块提供电源，电压要求稳定、噪声小，电流容量大于 500mA；5V 电压给编码器、lm1117 以及其余数字芯片供电；LM1084-adj 稳压至 5.5V 对舵机供电；DC-DC 芯片对 H 桥驱动芯片 hip4082 供电。

3.5 电机驱动模块

3.5.1 H 桥电机驱动芯片

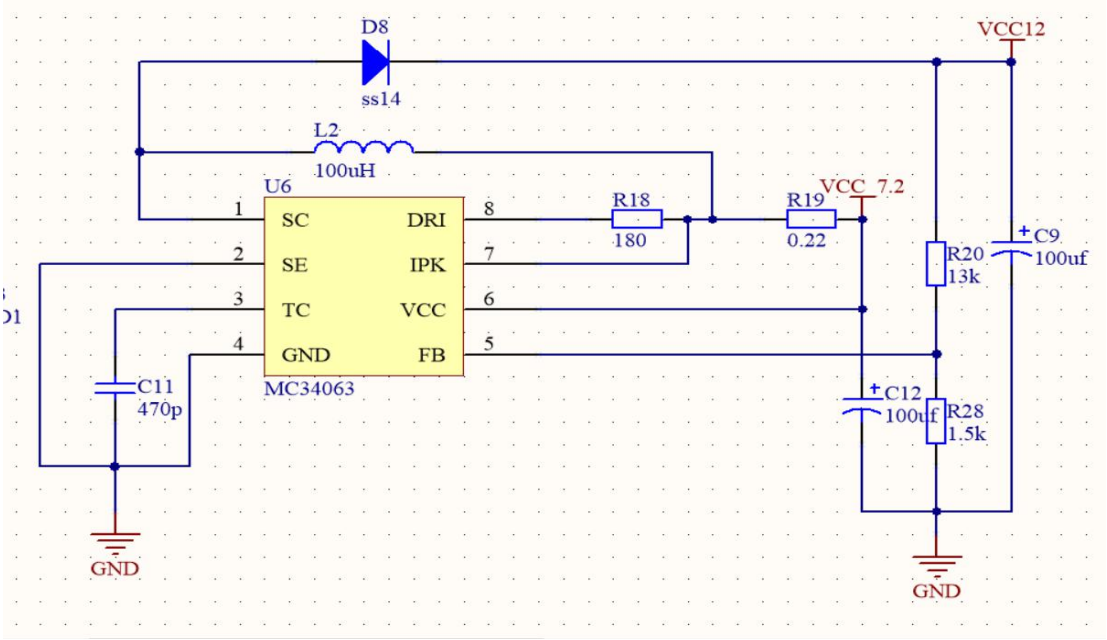
HIP4082 具有击穿保护,欠压保护功能,同与其相似的 HIP4081 相比,HIP4082 有一个更灵活的输入协议来驱动 MOSFET,HIP4082 降低了驱动电流来允许使用一个更小的封装，它拥有一个更广泛的可编程死区时间（0.1-4.5us），使其理想开关频率高达 200KHZ。

用 HIP4082 构成一个 H 桥驱动电路,通过 10,13,14,16 脚输出信号来控制电机。



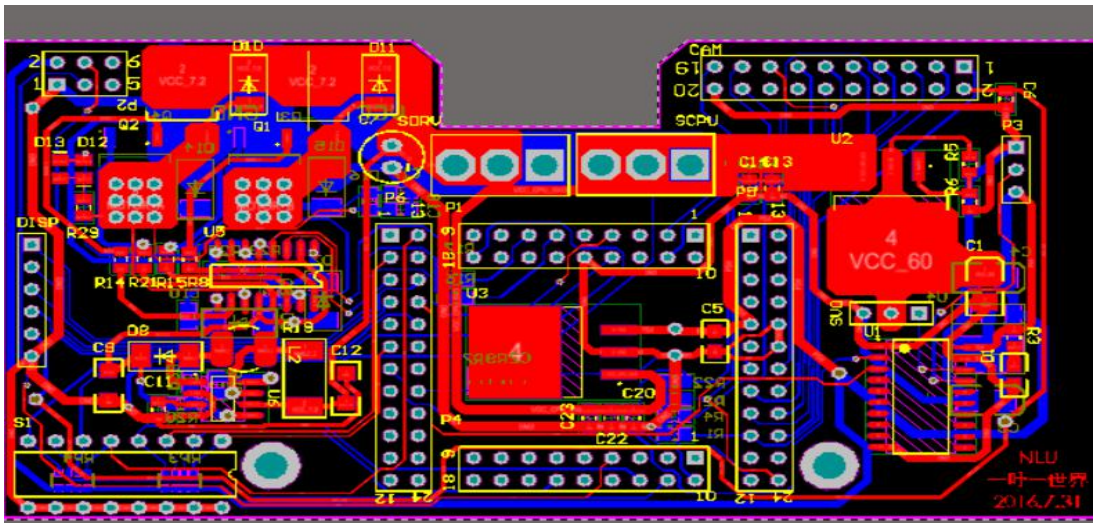
3.5.2 升压电源变换

HIP4082 推荐工作电压为 12V，因此需要进行升压电源变换来获得较强的栅极驱动能力，选用 MC34063 进行升压变换对 HIP4082 供电。



3.6 PCB 板设计

我们为了使小车在行驶的过程中更加稳定，我们不仅从机械上尽量减轻整车重量，降低车体重心位置，同时使电路设计尽量简洁，尽量减少元器件使用数量，缩小电路板面积，使电路部分重量轻，易于安装。我们在对电路进行详细、彻底的分析后，对电路进行了大量简化，并合理设计元件排列和电路走线，使得单片机在超频条件下能够稳定工作，这样使本系统硬件电路部分能够紧紧贴在车模底盘上面。具体 PCB 设计如下：



第四章 系统软件设计及实现

高效的软件程序是智能车高速平稳自动寻线的基础。我们设计的智能车系统采用 CMOS 摄像头进行赛道识别，图像采集及校正处理就成了整个软件的核心内容。在智能车的转向和速度控制方面，我们使用了鲁棒性很好的经典 PID 控制算法，配合使用理论计算和实际参数补偿的办法，使智能车能够稳定快速寻线。

4.1 赛道中心线提取及优化处理

4.1.1 原始图像的特点

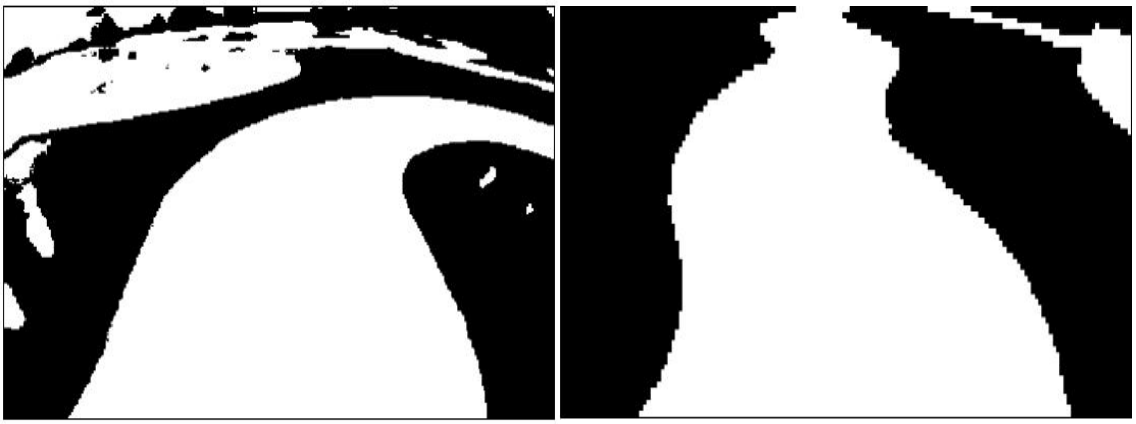
在单片机采集图像信号后需要对其进行处理以提取主要的赛道信息，同时，由于交叉道、起点线的存在，光线、杂点、赛道远处图像不清楚的干扰，图像效果会大打折扣。因此，在软件上必须排除干扰因素，对赛道进行有效识别，并提供尽可能多的赛道信息供决策使用。

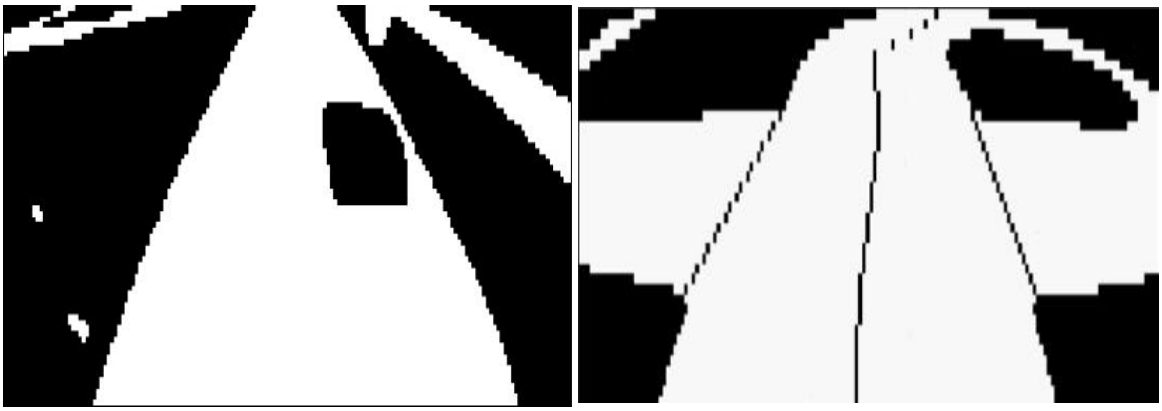
在图像信号处理中我们提取的赛道信息主要包括：赛道两侧边沿点位置、通过校正计算的赛道中心位置，中心点规划面积，赛道变化幅度，赛道类型判别。

由于摄像头自身的特性，图像会产生梯形式变形，这使得摄像头看到的信息不真实。因此我们利用赛道进行测量，创建函数还原出了真实赛道信息。原始图像是一个将模拟图像经模拟电路转换得到的二维数据矩阵，矩阵的每一个元素对应一个像素点，图像的第一行对应最远处，大约 190cm，图像的最底部一行对应最近处，大约 5cm。远处的图像小，近处的图像大，黑线为梯形状。

单片机通过比较器电路将每一行的黑白跳变点（跳变点按从右到左的顺序）记录下来，保存到两个二维数组里（分别表示上升沿、下降沿）。通过遍历上升沿和下降沿可以完成赛道边沿的提取。

摄像头采集到几种典型赛道图像如图





4.1.2 赛道边沿提取

边沿提取算法的基本思想如下：

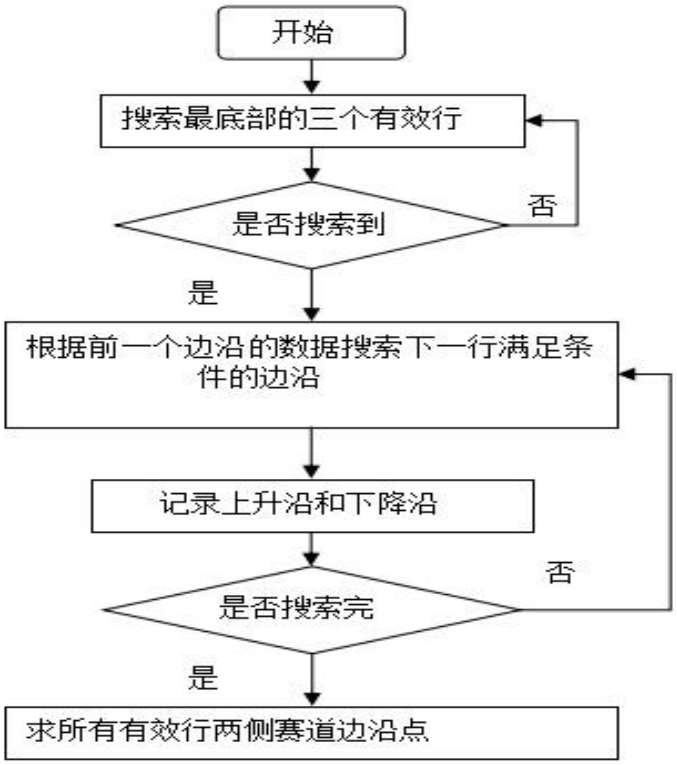
（1）直接逐行扫描原始图像，根据设定的阈值提取黑白跳变点；

（2）赛道宽度有一个范围，在确定的赛道宽度范围内提取有效赛道边沿，这样可以滤除不在宽度范围内的干扰；

（3）利用赛道的连续性，根据上一行白块的位置和边沿的位置来确定本行的边沿点；

（4）求边沿点时，因为近处的图像稳定，远处图像不稳定，所以采用由近及远的办法；

（5）进出十字的时候，通过校正计算出边沿角度可较好的滤除十字并补线；



4.1.3 图像校正

图像校正的实现如下：

（1）调整好摄像头位置、前瞻，固定好；

（2）将摄像头对准黑白相间的赛道板，然后用电视观看摄像头图像，用照相机对准电视拍照(见图 5.10)；

图 5.10 电视中的图像

（3）从照片中截取出赛道部分，然后用 matlab 编写程序，载入图片并进行相应的桶形变换、透视变换，调整好参数，生成校正表和反校正表；

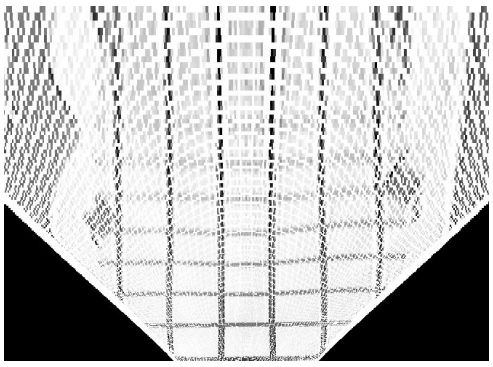
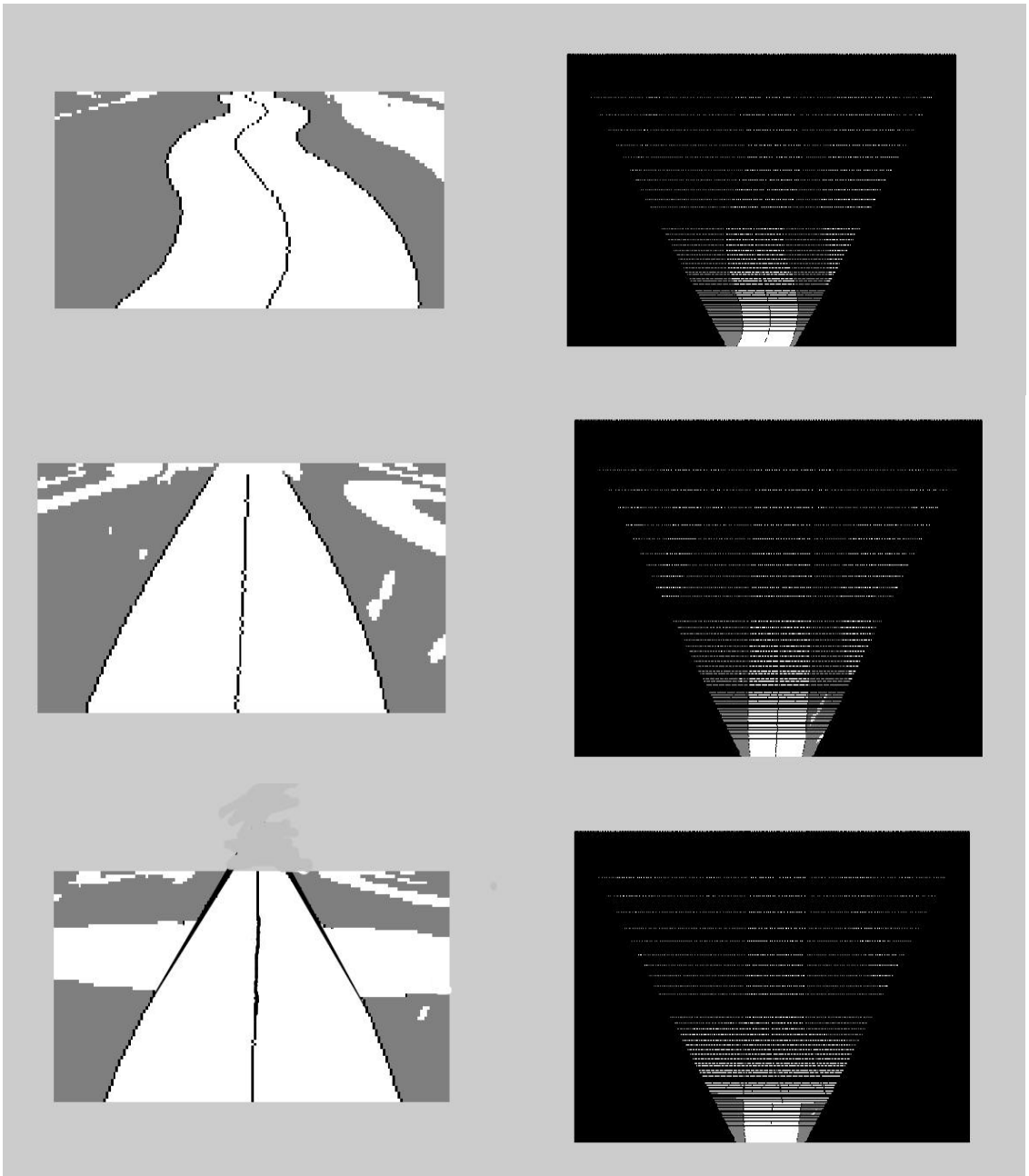


图 5.11 校正后的效果

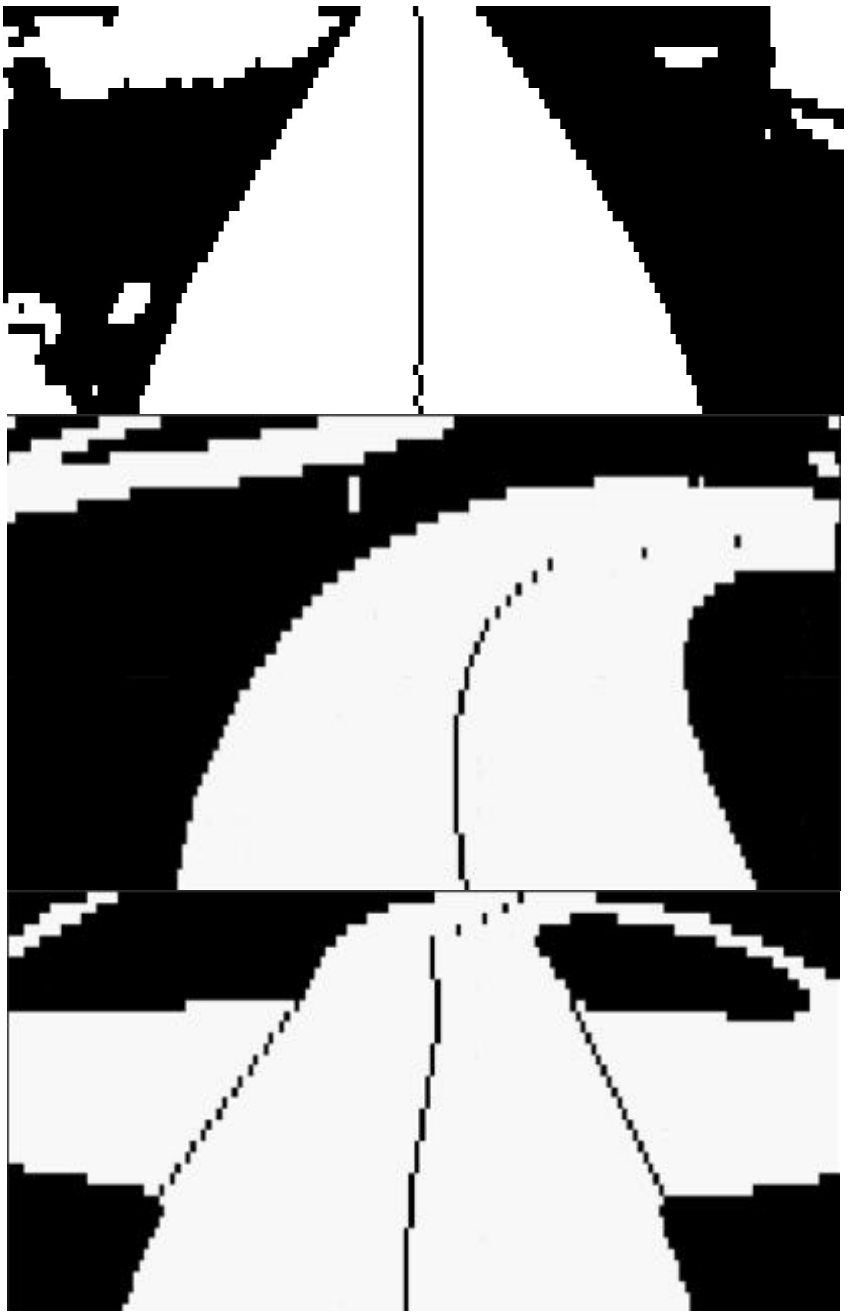
（4）在单片机程序中加入常量表，然后就可进行相应的校正和反校正变换了。

校正效果如下：



4.1.4 推算中心

通过之前提取的赛道边沿数据推算中心：当左右边沿点总数较少时返回；若只有单边有边沿点数据，则通过校正对单边数据按法线平移赛道宽度一半的距离；当能找到与一边能匹配上的另一边沿点时则直接求其中心作为中心点。推算完中心点后，对中心点进行均匀化，方便之后的控制。计算出的中心点效果如下：



4.1.5 路径选择

根据往届比赛以及本届的经验，赛车能否以最短的时间完成比赛，与赛车的速度和路径都有着密切的关系，因此，如何使赛车以一个最合理、最高效的路径完成比赛是提高平均速度的关键。

对于赛车路径的优化，我们从以下三个方面来完成：

1）增加视场的长度和宽度

根据我们的分析，当赛车采集到的图像能够覆盖一个比较完整的 S 弯道时，

通过加权算法计算出来的中心就会处于视场中央附近,此时赛车会以一个比较好的路径快速通过 S 弯道；相反，如果视场无法覆盖一个完整的 S 弯道，赛车就会误处理为普通的单向弯道，这样赛车的速度就会大大减慢。因此，尽量增大视场的长度和宽度就很有必要了。

视场的长度与单片机可以处理的图像行数成正比。我们采用由运算放大器制作的模拟比较器进行图像二值化，处理速度较 A/D 转换有了很大提高，大大增加了单片机处理的图像行数，最终处理行数为 70 行(隔 3 行提取一行)，达到的视场长度为 190 多 cm。为了增加视场宽度，增加每行采集的图像点数之外，我们采用了广角镜头，从而有效地增加了视场宽度。

2）优化加权算法

对整场有效行的中心求加权平均值的算法,在低速情况下可以有效地优化赛车路径，但在赛车速度提高到一定程度之后由于过弯时的侧滑，路径不是很好。而由于图像分布不均，三分之二的行分布于车体前方 40cm 的范围内，求出的加权平均值受车体近处的图像影响较大,因此整场图像求加权的算法对于高速情况下的路径优化效果不是很明显。

为了解决这个问题，我们对于参与加权计算的图像行数及权重进行了处理，减小了车体前部 50cm 范围内的图像参与加权的行数和权重，同时增大视场前部图像的权重。在经过长期调试之后，得到了一套比较合适的参数，能够有效优化高速情况下的赛车路径。

3）对不合理的中心点进行处理

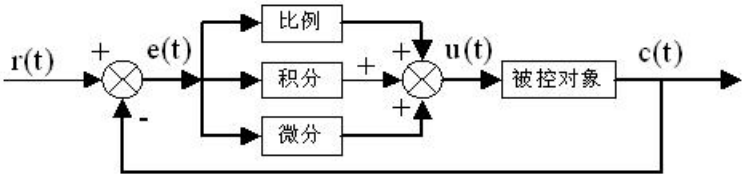
对于在校正后的图像数据中求得的中心线,反校正到原始图像后存在一行中含有多个中心点的情况。在通常情况下，这种情况出现在较远的视野中，但由于我们增大了视场前部图像的权重，这些中心点对权重的影响极大，导致车模容易出现掉轮甚至冲出赛道的现象。

4.2 PID 控制算法介绍

在工程实际中，应用最为广泛的调节器控制规律为比例、积分、微分控制，简称 PID 控制，又称 PID 调节。PID 控制器问世至今已有近 70 年历史，它以其结构简单、稳定性好、工作可靠、调整方便而成为工业控制的主要技术之一。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，控制理论的其它技术难以采用时,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。即当我们不完全了解一个系统和被控对象，或不能通过有效的测量手段来获得系统参数时，最适合用 PID 控制技术。PID 控制，实际中也有 PI 和 PD 控制。

PID 控制器是一种线性控制器，它根据给定值与实际输出值构成控制偏差。

将偏差的比例(P)、积分(I)和微分(D)通过线性组合构成控制量，对被控对象进行控制，故称 PID 控制器，原理框图如图所示。



在计算机控制系统中，使用的是数字 PID 控制器，控制规律为:

e(k)=r(k)-c(k)

式中

k——采样序号，k = 0，1，2…； r(k)——第 k 次给定值；

c(k)——第 k 次实际输出值； u(k)—— 第 k 次输出控制量；

e(k)—— 第 k 次偏差； e(k-1)—— 第 k-1 次偏差；

KP——比例系数； TI——积分时间常数；

TD——微分时间常数； T——采样周期。

简单说来，PID 控制器各校正环节的作用如下：

比例环节：及时成比例地反映控制系统的偏差信号，偏差一旦产生，控制器立即产生控制作用，以减少偏差。

积分环节：主要用于消除静差，提高系统的无差度。积分作用的强弱取决于积分时间常数，越大，积分作用越弱，反之则越强。

微分环节：能反映偏差信号的变化趋势(变化速率)，并能在该偏差信号变得太大之前，在系统中引入一个有效的早期修正信号，从而加快系统的动作速度，减小调节时间。 数字 PID 控制算法通常分为位置式 PID 控制算法和增量式 PID 控制算法。

4.2.1 位置式 PID

位置式 PID 中，由于计算机输出的 u (k) 直接去控制执行机构(如阀门)，u(k) 的值和执行机构的位置(如阀门开度)是一一对应的,所以通常称公式(4.5)为位置式 PID 控制算法。

位置式 PID 控制算法的缺点是：由于全量输出，所以每次输出均与过去的状态有关，计算时要对过去 e(k)进行累加，计算机工作量大；而且因为计算机输出的 u(k)对应的是执行机构的实际位置，如计算机出现故障，u(k)的大幅度变化， 会引起执行机构位置的大幅度变化，这种情况往往是生产实践中不允许

的，在某些场合，还可能造成严重的生产事故。因而产生了增量式 PID 控制的控制算法，所谓增量式 PID 是指数字控制器的输出只是控制量的增量△u(k)。

4.2.2 增量式 PID

当执行机构需要的是控制量的增量(例如：驱动步进电机)时，可由式(4.5)推导出提供增量的 PID 控制算式。由式(4.5)可以推出式(4.6)，式(4.5)减去式(4.6)可得式(4.4)。

式中；；

公式(4.7)称为增量式 PID 控制算法，可以看出由于一般计算机控制系统采用恒定的采样周期 T，一旦确定了 KP、TI 、TD，只要使用前后三次测量值的偏差，即可由式(4.7)求出控制增量。

增量式 PID 具有以下优点：

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

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

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

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

4.2.3 PID 的参数整定

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

第五章 系统开发及调试工具

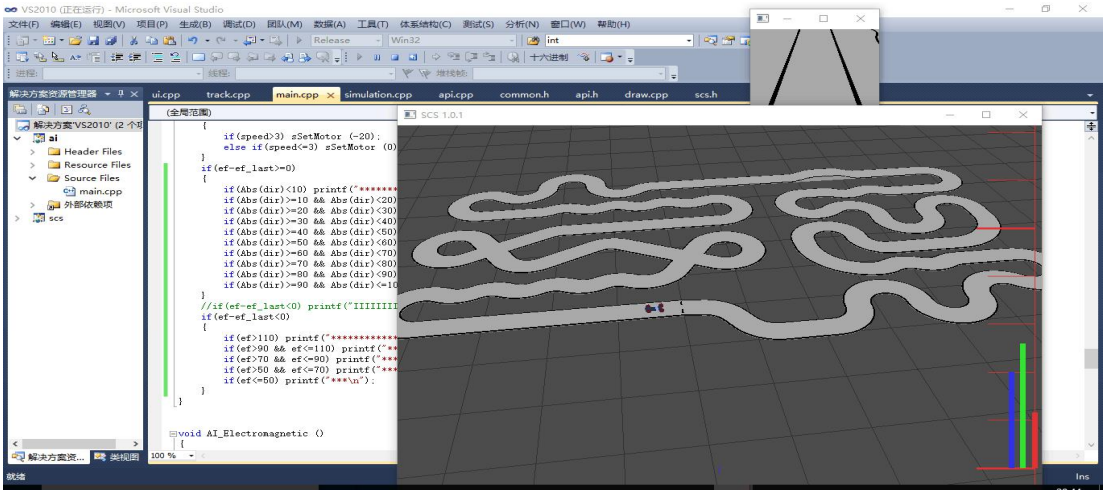
5.1 开发工具

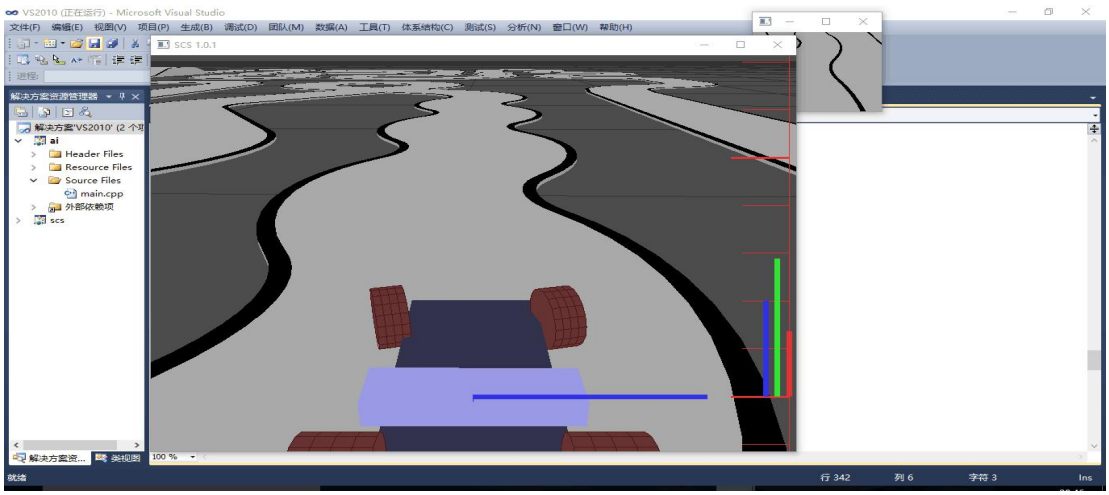
程序开放在 IAR Embedded Workbench IDE 下进行， Embedded Workbench for ARM 是 IAR Systems 公司为 ARM 微处理器开发的一个集成开发环境(下面简称 IAR EWARM)。比较其他的 ARM 开发环境，IAR EWARM 具有入门容易、使用方便和代码紧凑等特点。

EWARM 中包含一个全软件的模拟程序(simulator)。用户不需要任何硬件支持就可以模拟各种 ARM 内核、外部设备甚至中断的软件运行环境。从中可以了解和评估 IAR EWARM 的功能和使用方法。

5.2 仿真工具

便于赛道的设计、算法的验证、路径的优化、加减速的实现，我们采用了实用性较好的智能车仿真软件(SCS)进行了仿真。智能车仿真软件(SCS)是基于 VS2010 环境下运行的软件，软件运行图如下：



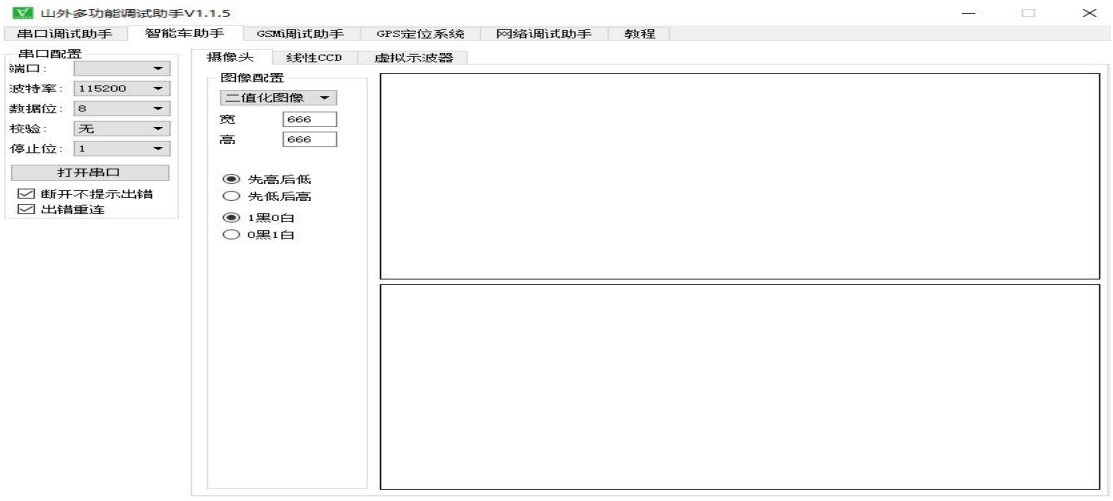


5.3 上位机图像显示

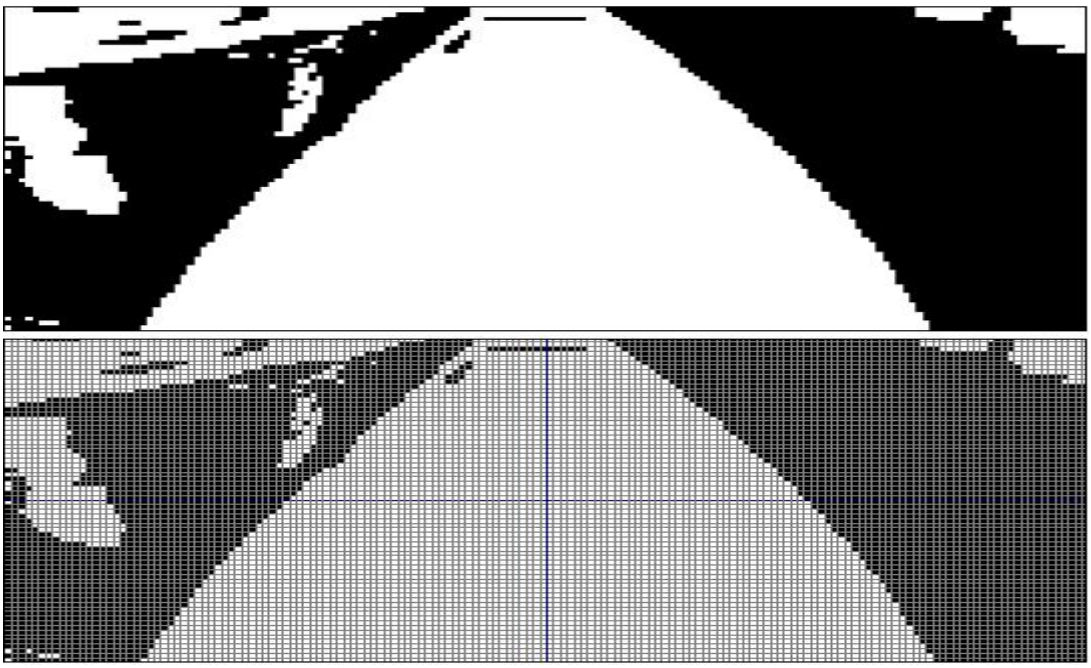
5.3.1 山外多功能上位机

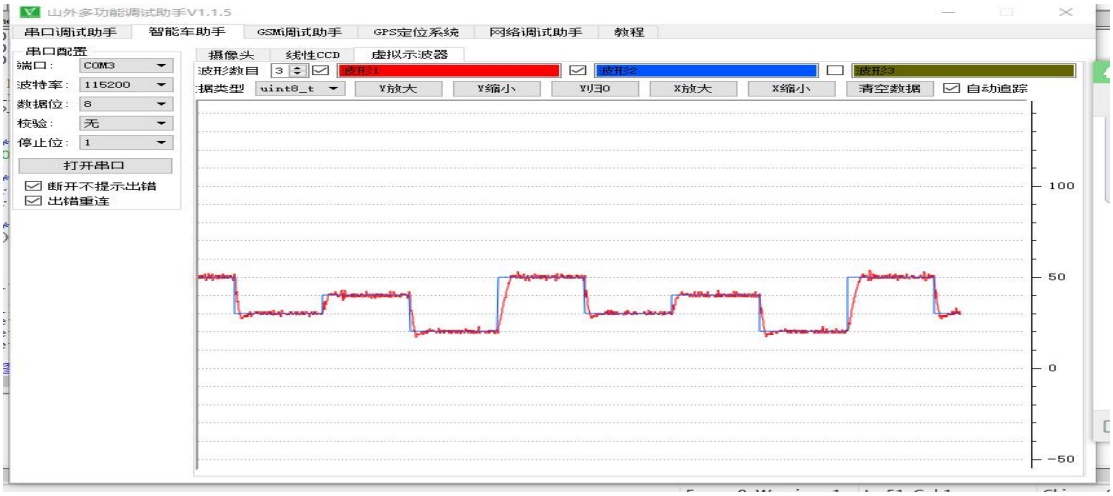
为了观察摄像头采集图像的直观效果,我们还采用了山外多功能上位机作为辅助开发调试工具。

我们设计的智能车系统采用 CMOS 摄像头采集赛道信息，分析处理之后用来编写黑线识别及控制算法。我们在山外多功能上位机环境下进行图像显示与处理程序，可完成赛道显示及相关参数的实时反馈，以及速度波形，运行界面如图 6.1 所示。



显示区域可原始图像及处理后的中心点,这为控制算法的编写提供了非常好的依据，也大大减少了调试者的工作量。





5.4 蓝牙调试工具

为了方便智能车调试，利用 CH340 芯片开发的 USB 转串口(UART 转 TTL)模块配合蓝牙，进行数据传输。通过蓝牙模块可以方便图像的采集与显示。蓝牙如下：

5.5 拨码开关调试工具

为了方便智能车调试，以及比赛试车，比赛出现意外方便换挡，我们采用了拨码开关，电路如图 6.4 所示：

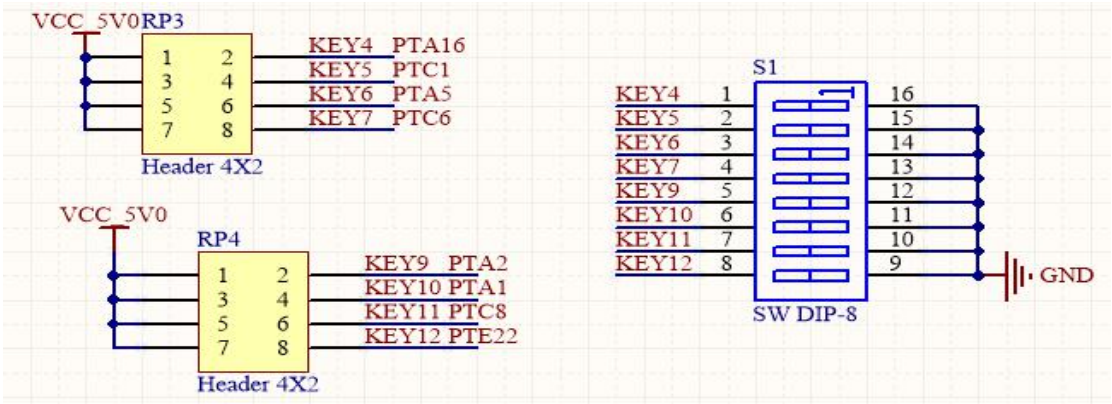


图 6.4 拨码开关电路图

第六章　模型车的主要技术参数

赛车基本参数	长	30cm
	宽	20cm
	高	35cm
车重		1350g
功耗	空载	10W
	带载	大于 12W
电容总容量		1800uF
传感器	光电编码器	1 个
	CMOS 摄像头	1 个
除了车模原有的驱动电机、舵机之外伺服电机个数		0
赛道信息检测	视野范围（近瞻/远瞻）	5/170cm
	精度(近/远)	2/12.5mm
	频率	112Hz

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]张军.AVR 单片机应用系统开发典型实例.北京：中国电力出版社，2005.

[3]王晓明．电动机的单片机控制［M］．北京：北京航空航天大学出版社.2002.

[4]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． 162-163.

[5]张文春．汽车理论[M]．北京．机械工业出版社．2005.

[6]童诗白,华成英.模拟电子技术基础［M］.北京: 高等教育出版社,2001.

[7]阎石．数字电子技术基础［M］．北京: 高等教育出版社，2000.

[8]谭浩强著．C 程序设计．北京：清华大学出版社，2003.

[9]尹勇.Protel DXP 电路设计入门与进阶［M］.北京: 科学出版社,2004.

[10]Park K.H ，Bien Z，Hwang D.H. A study on the robustness of a PID - type iterative learning controller against initial state error [J]. Int. J. Syst. Sci. 1999, 30(1) ，102～135.

[11]殷剑宏,吴开亚.图论及其算法［M］.中国科学技术大学出版社,2003.

[12]夏克俭．数据结构及算法［M］．北京：国防工业出版社， 2001.

[13]尹怡欣,陶永华.新型 PID 控制及其应用.北京：机械工业出版社,1998 年.

[14]李太福.基于在线参数自整定的模糊 PID 伺服控制系统[J]．交流伺服系统，2005，4：203～215.

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

附录：程序源代码

```
/*************************主函数*******************************
* 函数名称：main
* 功能说明：
* 备注：
*************************************************************/
void main()
{


    LCD_Init();

    LCD_Fill(0x00);
    NVIC_SetPriority(PORTC_PORTD_IRQn,1);      //配置优先级
    NVIC_SetPriority(DMA1_IRQn,0);             //配置优先级
    NVIC_SetPriority(PIT_IRQn,2);              //配置优先级

    //采集图像
    Camera_get_img_init();
    enable_irq(PORTC_PORTD_IRQn);
    enable_irq(DMA1_IRQn);



//      tpm_pwm_init(TPM0, TPM_CH3,20*1000,0);          //电机初始化
PWM     TPM0_CH3 对应为 PTE30     电机 IN2

    gpio_init(PTE22,GPI,0);//左红外管
    gpio_init(PTE23,GPI,0);//右红外管

    gpio_init(PTA16,GPI,0);    //拨码开关
    gpio_init(PTC1,GPI,0);
    gpio_init(PTA5,GPI,0);
    gpio_init(PTC6,GPI,0);
    gpio_init(PTA2,GPI,0);
    gpio_init(PTA1,GPI,0);
    gpio_init(PTC8,GPI,0);
    adc_init(ADC0_SE9);    //加速度计
    adc_init(ADC0_SE12);   //陀螺仪
```

```
        gpio_init(PTA13,GPO,1);     //蜂鸣器

        Key_Init();

        tpm_pwm_init(TPM1, TPM_CH0,300,S_D5_centervalue);          //初始化
舵机 PWM
        tpm_pwm_init(TPM0, TPM_CH3,20*1000,0);                //电机初始化
PWM     TPM0_CH3 对应为 PTE30     电机 IN2
        tpm_pwm_init(TPM0,TPM_CH0,20*1000,0);                //电机初始化
PWM     TPM0_CH0 对应为 PTE24      电机 IN1
        Speed_get_init();
        DELAY_MS(1000);
            while(1)
            {
              if(photo_status==photo_finish)
              {
                Camera_img_deal();
                Camera_get_img();
              }
              if(photo_status==photo_fail)
              {
                Camera_get_img();
              }

            }
    }


    /************************* 初始化采集图像函数
*****************************
    * 函数名称：Camera_get_img_init
    * 功能说明：初始化函数
    * 备注：
    ************************************************************************
*********/
    void Camera_get_img_init(void)
    {
      DisableInterrupts;

      while(ov7725_eagle_reg_init() == 0);
```

```
        dma_portx2buff_init(DMA_CH0, (void *)&PTD_B0_IN, (void *)buff, PTC13,
DMA_BYTE1, OV7725_EAGLE_SIZE);
        DMA_DIS(DMA_CH0);
        DMA_IRQ_DIS(DMA_CH0);

        disable_irq(PORTC_PORTD_IRQn);                          //关闭 PTA 的
中断

        port_init(PTC13, ALT1 | DMA_FALLING | PULLDOWN );          //PCLK
        port_init(PTC12, ALT1 | IRQ_FALLING   | PULLDOWN | PF);      //场中断，
上拉，上降沿触发中断，带滤波

        set_vector_handler(PORTC_PORTD_VECTORn ,PORTC_PORTD_IRQHandler);
        set_vector_handler(DMA0_VECTORn ,DMA0_IRQHandler);

        EnableInterrupts;
    }

    /*************************        获        取        图        像
*****************************
    * 函数名称：Camera_get_img
    * 功能说明：获取图像
    * 备注：仅在查看图像时调用
    ************************************************************************
*/
    void Camera_get_img(void)
    {
        photo_status = photo_start;                         //开始采集图像
        PORTC_ISFR = ~0;                                   //写 1 清中断标志位(必须的，
不然回导致一开中断就马上触发中断)
        enable_irq(PORTC_PORTD_IRQn);
    }

    /*************************        处        理        图        像
*****************************
    * 函数名称：Camera_img_deal
    * 功能说明：处理图像
    * 备注：
    ************************************************************************
*/
    void send(void)
    {
```

```
    int i=0;
    uint8 a[4]={1,254,254,1};
    uart_putbuff(UART0, a, 4);
    uart_putchar(UART0, (uint8)effective_line);
    for(i=0;i<H/GER;i++)
    {
        uart_putchar(UART0, (uint8)left_line_temp[i]);
        uart_putchar(UART0, (uint8)centerline[i]);
        uart_putchar(UART0, (uint8)right_line_temp[i]);
    }
}
void Camera_img_deal(void)
{
    img_copy();
    Center_line();
    S3010_control();

    PIT_IRQHandler();
//    int i=0;
//        for(i=5;i<65;i++)
//       picture[W*i+centerline[H/GER-1]] = black;
//      printf("centerline[H/GER-1]      %d      centerline_last      %d
\n",centerline[H/GER-1],centerline_last);
//    send();
//    show_line();
//    int i=0;
//    for(i=0;i<70;i++)
//       printf("%d,",right_line_temp[i]-left_line_temp[i]);
//    printf("\n\n\n\n");
   /*if(car_speed==0)*/
//    IMG_OLED_diplay();
//       printf("%d\n",right_line_temp[30]-left_line_temp[30]);

#if 0
    img_reduce(picture,send_buff,H*W/GER);          //图像滤波
    send_picture(send_buff,H*W/GER/8);   //发送压缩图像
#endif




}
```

```
    /************************* 行 中 断 、 场 中 断 函 数
*****************************
    * 函数名称：PORTC_PORTD_IRQHandler
    * 功能说明：摄像头行、场中断函数
    * 备注：
    ************************************************************************
*********/
    void PORTC_PORTD_IRQHandler(void)
    {                                                    //场中断
      if(PORTC_ISFR & (1 << 12))                                    //PTA6 触
发中断
      {
        PORTC_ISFR |= 1<<    PT12;
        VSYNC_handler();
      }
    }

    /************************* 场 中 断 服 务 函 数
*******************************
    * 函数名称：VSYNC_handler
    * 功能说明：场中断服务函数
    * 备注：
    ************************************************************************
******/
    void VSYNC_handler(void)
    {
      if(photo_status==photo_start)
      {
        photo_status = photo_capture;                    //标记图像采集中
        disable_irq(PORTC_PORTD_IRQn);
        dma_repeat(DMA_CH0,          (void          *)&PTD_B0_IN,          (void
*)(buff),OV7725_EAGLE_SIZE);
        DMA_IRQ_EN(DMA_CH0);
        DMA_EN(DMA_CH0);
      }
      else
      {
        disable_irq(PORTC_PORTD_IRQn);                          //关闭 PTA
的中断
        photo_status = photo_fail;                       //标记图像采集失败
      }
```

```
}

/************************* 场 中 断 服 务 函 数 ****************************
* 函数名称：DMA0_IRQHandler
* 功能说明：DMA 中断服务函数
* 备注：
*************************************************************************/
void DMA0_IRQHandler()
{
  DMA_IRQ_DIS(DMA_CH0);
  photo_status = photo_finish ;
  DMA_IRQ_CLEAN(DMA_CH0);
}



/************************* 山 外 上 位 机 发 送 图 像 函 数 *************************
* 函数名称：send_picture
* 功能说明：山外上位机发送图像函数
* 备注：
*************************************************************************/
void send_picture(uint8 *aim,int num)
{
//   uint8 a[1]={2};
  DisableInterrupts;
//   uart_putbuff(UART0, a, 1);
  uart_putbuff(UART0, cmdf, sizeof(cmdf));
  uart_putbuff(UART0, aim, num);
  uart_putbuff(UART0, cmdr, sizeof(cmdr));
  PORTC_ISFR    |= (1 << PT12);
#if 0
  PIT_Flag_Clear(QEP_pit);
#endif
  EnableInterrupts;
}
```

```
    /*************************          图          像          解          压
*********************************
    * 函数名称：img_extract
    * 功能说明：
    * 备注：
    ***************************************************************************
******/
    void img_extract(uint8 *dst, uint8 *src, uint32 srclen)
    {
        uint8 colour[2] = {white, black}; //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;
        while(srclen --)
        {
            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 ];
        }
    }



    /*************************          滤          波          函          数
*********************************
    * 函数名称：img_reduce
    * 功能说明：
    * 备注：
    ***************************************************************************
******/
    void    img_reduce(uint8 *aim,uint8 *get,uint32 length)
    {
      while(length>0)
      {
        *get = 0;
        *get += ((*aim++) & 0x80) >> 0;
        *get += ((*aim++) & 0x80) >> 1;
```

```
            *get += ((*aim++) & 0x80) >> 2;
            *get += ((*aim++) & 0x80) >> 3;
            *get += ((*aim++) & 0x80) >> 4;
            *get += ((*aim++) & 0x80) >> 5;
            *get += ((*aim++) & 0x80) >> 6;
            *get += ((*aim++) & 0x80) >> 7;
            *get =~ *get;
            get++;
            length -= 8;
        }

    }
    /************************** 滤 波 函 数
*******************************
    * 函数名称：img_copy
    * 功能说明：
    * 备注：
    ************************************************************************
******/
    void img_copy()
    {
        int copy_count = 0;
        for(copy_count = 0;copy_count < H/GER;copy_count++)
        {
            img_extract(&picture[copy_count*W],&buff[((28                    +
3*copy_count)*OV7725_EAGLE_W)/8],W/8);
        }
    }


    void debug_speed(void)
    {
    printf(" 向 左 =%d left_speed=%d left_pwm=%d right_speed=%d
right_pwm=%d\n",S3010_value-ServoParam[0],left_speed,left_motor_forward_pw
m,right_speed,right_motor_forward_pwm);
    }


    /************************** 滤 波 函 数
*******************************
    * 函数名称：Key_Init()
    * 功能说明：
```

```
    * 备注：
    ************************************************************************
******/
    void Key_Init()
    {
    #define    KEY1       gpio_get(PTA16)
    #define    KEY2       gpio_get(PTC1)
    #define    KEY3       gpio_get(PTA5)
    #define    KEY4       gpio_get(PTC6)
    #define    KEY6       gpio_get(PTA2)
    #define    KEY7       gpio_get(PTA1)
    #define    KEY8       gpio_get(PTC8)

                /*********    K1    ***********/
                if(KEY1 == 0 && KEY2 == 1 && KEY3 == 1 && KEY4 == 1 && KEY6
== 1 && KEY7 == 1 && KEY8 == 1)
                {
                    DELAY_MS(10);
                    if(KEY1 == 0 && KEY2 == 1 && KEY3 == 1 && KEY4 == 1 &&
KEY6 == 1 && KEY7 == 1 && KEY8 == 1)
                    {
                        mode1=1;
                        mode2=0;
                        mode3=0;
                        mode4=0;
                        mode5=0;
                        mode6=0;
                        mode7=0;
                    }
                }

                /*********    K2    ***********/
                if(KEY1 == 1 && KEY2 == 0 && KEY3 == 1 && KEY4 == 1 && KEY6
== 1 && KEY7 == 1 && KEY8 == 1)
                {
                    DELAY_MS(10);
                    if(KEY1 == 1 && KEY2 == 0 && KEY3 == 1 && KEY4 == 1 &&
KEY6 == 1 && KEY7 == 1 && KEY8 == 1)
                    {
                        mode1=0;
                        mode2=1;
                        mode3=0;
                        mode4=0;
```

```
                    mode5=0;
                    mode6=0;
                    mode7=0;
                }
            }

            /*********     K3     ***********/
            if(KEY1 == 1 && KEY2 == 1 && KEY3 == 0 && KEY4 == 1 && KEY6
== 1 && KEY7 == 1 && KEY8 == 1)
            {
                DELAY_MS(10);
                if(KEY1 == 1 && KEY2 == 1 && KEY3 == 0 && KEY4 == 1 &&
KEY6 == 1 && KEY7 == 1 && KEY8 == 1)
                {
                    mode1=0;
                    mode2=0;
                    mode3=1;
                    mode4=0;
                    mode5=0;
                    mode6=0;
                    mode7=0;
                }
            }

            /*********     K4     ***********/
            if(KEY1 == 1 && KEY2 == 1 && KEY3 == 1 && KEY4 == 0 && KEY6
== 1 && KEY7 == 1 && KEY8 == 1)
            {
                DELAY_MS(10);
                if(KEY1 == 1 && KEY2 == 1 && KEY3 == 1 && KEY4 == 0 &&
KEY6 == 1 && KEY7 == 1 && KEY8 == 1)
                {
                    mode1=0;
                    mode2=0;
                    mode3=0;
                    mode4=1;
                    mode5=0;
                    mode6=0;
                    mode7=0;
                }
            }

            /*********     K5     ***********/
```

```
                if(KEY1 == 1 && KEY2 == 1 && KEY3 == 1 && KEY4 == 1 && KEY6
== 0 && KEY7 == 1 && KEY8 == 1)
                {
                    DELAY_MS(10);
                    if(KEY1 == 1 && KEY2 == 1 && KEY3 == 1 && KEY4 == 1 &&
KEY6 == 0 && KEY7 == 1 && KEY8 == 1)
                    {
                        mode1=0;
                        mode2=0;
                        mode3=0;
                        mode4=0;
                        mode5=1;
                        mode6=0;
                        mode7=0;
                    }
                }

                /*********     K6     ***********/
                if(KEY1 == 1 && KEY2 == 1 && KEY3 == 1 && KEY4 == 1 && KEY6
== 1 && KEY7 == 0 && KEY8 == 1)
                {
                    DELAY_MS(10);
                    if(KEY1 == 1 && KEY2 == 1 && KEY3 == 1 && KEY4 == 1 &&
KEY6 == 1 && KEY7 == 0 && KEY8 == 1)
                    {
                        mode1=0;
                        mode2=0;
                        mode3=0;
                        mode4=0;
                        mode5=0;
                        mode6=1;
                        mode7=0;
                    }
                }

                /*********     K7     ***********/
                if(KEY1 == 1 && KEY2 == 1 && KEY3 == 1 && KEY4 == 1 && KEY6
== 1 && KEY7 == 1 && KEY8 == 0)
                {
                    DELAY_MS(10);
                    if(KEY1 == 1 && KEY2 == 1 && KEY3 == 1 && KEY4 == 1 &&
KEY6 == 1 && KEY7 == 1 && KEY8 == 0)
                    {
```

```
                    mode1=0;
                    mode2=0;
                    mode3=0;
                    mode4=0;
                    mode5=0;
                    mode6=0;
                    mode7=1;
                }
            }


    }

    /******************编码器**********************/
    uint8 left_speed=0;   //左轮速度
    uint8 right_speed=0;   //右轮速度
    uint8 car_speed=0;    //平均速度

    /************************** 初 始 化 速 速 获 取 函 数
*****************************
    * 函数名称：Speed_get_init
    * 功能说明：初始化函数
    * 备注：
    ****************************************************************
*********/
    void Speed_get_init(void)
    {
    #define                 INT_COUNT                                   0xFFFF
//LPTMR 产生中断的计数次数
        lptmr_pulse_init(left_QEP,INT_COUNT, LPT_Rising);                   //
初始化脉冲计数器，用 LPT0_ALT2，即 PTC5 输入
        tpm_pulse_init(right_QEP_tpm,right_QEP,TPM_PS_1);
//初始化 TPM2 为脉冲累加，输入管脚为 TPM_CLKIN0_PIN ，分频系数为 1
        set_vector_handler(PIT_VECTORn ,PIT_IRQHandler);                    //
设置 PIT0 的中断服务函数为 PIT_IRQHandler
        pit_init_us(QEP_pit,                                        QEP_time);
//初始化 PIT0，定时时间为： QEP_time
        enable_irq (PIT_IRQn);                                              //
使能 PIT0 中断
    }
    /**************************PIT        中        断        函        数
*********************************
```

```
    * 函数名称：PIT_IRQHandler
    * 功能说明：PIT 中断服务函数
    * 备注：
    ************************************************************************
********/
    uint32 stime=0;      //
    uint32 right_old_stime=0;    //系统时间
    uint32 left_old_stime=0;     //系统时间
    uint8 right_start_flag=0;                                   //起跑标志位
    int left_start_flag=0;                                     //起跑标志位
    void PIT_IRQHandler(void)
    {
    /***PIT0 和 PIT1 共用相同中断号，所以两者都共用相同中断函数，需要根
据标志位来判断是由哪个 PIT 触发中断***/
      //printf("6\n");
      if(PIT_TFLG(QEP_pit) == 1)                                //判断是否 PIT0
进入中断
      {
        stime++;                                                 //系统时间变量
自增
    //    if(right_start_flag>=0&&left_start_flag>=0)
        if(left_start_flag>=0)
        {
          Speed_get();                                           //获取当前速
度
        }
        PIT_Flag_Clear(QEP_pit);                                //清中断标志位
      }
    }


    #include "include.h"
    #include "common.h"

    #define SIZE_OF_KEYBD_MENU   (7)

    SmartCarParameter theparameter;

    static uint8 key_value = 0;

    static uint32 keymenu_curr_state = 0;
```

```
void main_window_proc(void);
void run_window_proc(void);
void image_debug_proc(void);
void vision_parameter_proc(void);
void speed_pid_window_proc(void);
void dir_pid_window_proc(void);
void servo_window_proc(void);

KeyMenuTableStruct KeyMenuTable[SIZE_OF_KEYBD_MENU] =
{
    {0,0,0,0,main_window_proc},
        {1,0,0,0,run_window_proc},
    {2,0,0,0,image_debug_proc},
    {3,0,0,0,vision_parameter_proc},
    {4,0,0,0,speed_pid_window_proc},
    {5,0,0,0,dir_pid_window_proc},
    {6,0,0,0,servo_window_proc},
};

#define MAIN_MENU_LEN    (6)
char * MainMenuStrings[] = {"Run!!!","DebugImage","PictureDeal", "Speed",
"Direction", "ServoValueSet"};

char * RunMenuStrings[] = {" Ready ?? "};
uint32 RunMenuParam[VISIONDBG_MENU_LEN]={745};
uint32 RunMenuParamMin[VISIONDBG_MENU_LEN]={666};
uint32 RunMenuParamMax[VISIONDBG_MENU_LEN]={830};
uint32 RunMenuParamAdj[VISIONDBG_MENU_LEN]={1};

char * VisionDebugStrings[] =
{"zhijiao_start","Delay_Time","Delay_YesOrNo","Stop_YesOrNo","block_point"};
uint32 VisionDebugParam[VISIONDBG_MENU_LEN]={42,1600,0,1,10};
uint32 VisionDebugParamMin[VISIONDBG_MENU_LEN]={22,400,0,0,0};
uint32   VisionDebugParamMax[VISIONDBG_MENU_LEN]={55,4000,1,1,20};
uint32   VisionDebugParamAdj[VISIONDBG_MENU_LEN]={1,400,1,1,1};

char * SpeedPIDMenuStrings[] = {"SpeedP", "SpeedI",
"Speed_Set_Max","Speed_Mid","Speed_Set_Min"};
uint32 SpeedParam[SPEED_MENU_LEN]={25,6,60,60,50};//180,72,0(bu jia
bang bang kong zhi);324,28,0
uint32   SpeedParamMin[SPEED_MENU_LEN]={0,0,0,0,0};
uint32   SpeedParamMax[SPEED_MENU_LEN]={1200,512,200,200,200};
uint32   SpeedParamAdj[SPEED_MENU_LEN]={1,1,1,1,1};
```

```
    char * DirPIDMenuStrings[] = {"DirectionP", "DirectionI",
"DirectionD","S3010Row_K"};
    uint32  DirectionParam[DIR_MENU_LEN]={3,4,140,21};
    uint32  DirectionParamMin[DIR_MENU_LEN]={0, 0, 50,1};
    uint32  DirectionParamMax[DIR_MENU_LEN]={12,12,250,100};
    uint32  DirectionParamAdj[DIR_MENU_LEN]={1,1,1,1};

    char * ServoMenuStrings[] = {"ServoMid","ServoRight","ServoLeft"};
    uint32  ServoParam[SERVO_MENU_LEN]={1640,1480,1800};//中值 1400
    uint32  ServoParamMin[SERVO_MENU_LEN]={100,100,100};
    uint32  ServoParamMax[SERVO_MENU_LEN]={10000,10000,10000};
    uint32  ServoParamAdj[SERVO_MENU_LEN]={1,1,1};


    ////////////////////////////////////////////////////////////////////////////
    //              主屏幕
    ////////////////////////////////////////////////////////////////////////////
    void main_window_proc(void)
    {
        uint8   i;
        switch (key_value)
        {
          case 0x01:
            {
                //keymenu_curr_state                                    =
KeyMenuTable[keymenu_curr_state].KeyEscState;
            }
            break;
          case 0x04:
            {
                KeyMenuTable[keymenu_curr_state].MenuSubstate++;
                if      (KeyMenuTable[keymenu_curr_state].MenuSubstate      >
(MAIN_MENU_LEN-1))
                    KeyMenuTable[keymenu_curr_state].MenuSubstate = 0;
            }
            break;
          case 0x02:
            {
                if (KeyMenuTable[keymenu_curr_state].MenuSubstate == 0)
                    KeyMenuTable[keymenu_curr_state].MenuSubstate           =
MAIN_MENU_LEN;
                KeyMenuTable[keymenu_curr_state].MenuSubstate--;
```

```
            }
            break;
        case 0x08:
            {
                // 确认键
                keymenu_curr_state                                          =
KeyMenuTable[keymenu_curr_state].MenuSubstate+1;
                KeyMenuTable[keymenu_curr_state].MenuSetup = 0;
                LCD_clear();
            }
            break;
        default:
            {
                LCD_set_XY(16, 0);
                LCD_write_english(">> Main Menu <<");
                LCD_set_XY(16, 1);

LCD_write_int(VisionDebugParam[1]/400-stime/400);
                uint32 battval = adc_once      (ADC0_SE13, ADC_8bit);
                LCD_set_XY(80, 1);
                float battvolt = (float)((battval)*0.0689f-0.5476f);
                LCD_write_float(battvolt, 2);
                if ( battvolt > 7.7f )
                {
                    for (i=0; i<MAIN_MENU_LEN; i++)
                    {
                        LCD_set_XY(8, i+2);
                        LCD_write_english(MainMenuStrings[i]);
                    }
                    for (i=0; i<MAIN_MENU_LEN; i++)
                    {
                        LCD_set_XY(0, i+2);
                        if  (KeyMenuTable[keymenu_curr_state].MenuSubstate  ==
i)
                            LCD_write_english(">");
                        else
                            LCD_write_english(" ");
                    }
                }
                else
                {
                    for (i=0; i<MAIN_MENU_LEN; i++)
                    {
```

```
                    LCD_set_XY(8, i+2);
                    LCD_write_english("                    ");
                }
            }
        }
        break;
    }
}

//////////////////////////////////////////////////////////////////////////////
//              开车
//////////////////////////////////////////////////////////////////////////////
uint8 run_state = 0;

void run_window_proc(void)
{
    uint8_t i;
    switch (key_value)
    {
      case 0x01:
        {
            keymenu_curr_state                                    =
KeyMenuTable[keymenu_curr_state].KeyEscState;
            KeyMenuTable[keymenu_curr_state].MenuSetup = 0;
            LCD_clear();
        }
        break;
      case 0x04:
        {
            if (0 == KeyMenuTable[keymenu_curr_state].MenuSetup)
            {
                KeyMenuTable[keymenu_curr_state].MenuSubstate++;
                if    (KeyMenuTable[keymenu_curr_state].MenuSubstate    >
(SERVO_MENU_LEN-1))
                    KeyMenuTable[keymenu_curr_state].MenuSubstate = 0;
            }
        }
        break;
      case 0x02:
        {
            if (0 == KeyMenuTable[keymenu_curr_state].MenuSetup)
            {
                if (KeyMenuTable[keymenu_curr_state].MenuSubstate == 0)
```

```
                        KeyMenuTable[keymenu_curr_state].MenuSubstate =
SERVO_MENU_LEN;
                    KeyMenuTable[keymenu_curr_state].MenuSubstate--;
                }
            }
            break;
        case 0x08:
            {
                if (0 == run_state)
                {
                    run_state = 1;
                    LCD_clear();
                    LCD_set_XY(8, 0);
                    LCD_write_english("          Runing          ");
                }
                else
                {
                    run_state = 0;
                }
            }
            break;
        default:
            {
                if (0 == run_state)
                {
                    LCD_set_XY(8, 0);
                    LCD_write_english("          Run          ");

                    for (i=0; i<SERVO_MENU_LEN; i++)
                    {
                        LCD_set_XY(8, i+2);
                        LCD_write_english(RunMenuStrings[i]);
                    }
                }
            }
            break;
    }

}

//////////////////////////////////////////////////////////////////////////////
//              图像调试
//////////////////////////////////////////////////////////////////////////////
```

```
#define IMG_DBG_TOTAL_FUN    (3)
void image_debug_proc(void)
{
    switch (key_value)
    {
      case 0x01:
        {
            keymenu_curr_state = KeyMenuTable[keymenu_curr_state].KeyEscState;
            LCD_clear();
        }
        break;
      case 0x02:
        {
            KeyMenuTable[keymenu_curr_state].MenuSubstate++;
            if (KeyMenuTable[keymenu_curr_state].MenuSubstate > (IMG_DBG_TOTAL_FUN-1))
                KeyMenuTable[keymenu_curr_state].MenuSubstate = 0;

        }
        break;
      case 0x04:
        {
            if (KeyMenuTable[keymenu_curr_state].MenuSubstate == 0)
                KeyMenuTable[keymenu_curr_state].MenuSubstate = IMG_DBG_TOTAL_FUN;
            KeyMenuTable[keymenu_curr_state].MenuSubstate--;
        }
        break;
      case 0x08:
        {
            KeyMenuTable[keymenu_curr_state].MenuSetup = !KeyMenuTable[keymenu_curr_state].MenuSetup;
        }
        break;
      default:
        {
            // 根据不同的KeyMenuTable[keymenu_curr_state].MenuSubstate 绘制不同的图像
            // 根据 KeyMenuTable[keymenu_curr_state].MenuSetup 决定是不是显示提示信息
            if (KeyMenuTable[keymenu_curr_state].MenuSubstate == 0) // 显示二值化图
```

```
                {
                    display_binary_image(0, 0, picture, H, W);
                    if (0 != KeyMenuTable[keymenu_curr_state].MenuSetup)
                    {
                        //extern uint8_t  threshold;

                        LCD_set_XY(48, 7);
                        LCD_write_english("th:");
                        LCD_set_XY(64, 7);
                        LCD_write_int(threshold);
                    }
                }
                else if (KeyMenuTable[keymenu_curr_state].MenuSubstate == 1) //
显示边界
                {
                    //extern uint8    u8image_edge[picture_height*W];

                    display_binary_image(0, 0, picture, H, W);
    //              if (0 != KeyMenuTable[keymenu_curr_state].MenuSetup)
    //              {
    //                  LCD_set_XY(48, 7);
    //                  LCD_write_english("th:");
    //                  LCD_set_XY(64, 7);
    //                  LCD_write_int(threshold);
    //              }
                }
                else if (KeyMenuTable[keymenu_curr_state].MenuSubstate == 2) //
显示边界
                {
                    //extern uint8    u8image_mid[picture_height*W];

                    display_binary_image(0, 0, picture, H, W);
    //              if (0 != KeyMenuTable[keymenu_curr_state].MenuSetup)
    //              {
    //                  LCD_set_XY(48, 7);
    //                  LCD_write_english("th:");
    //                  LCD_set_XY(64, 7);
    //                  LCD_write_int(threshold);
    //              }
                }
            }
            break;
        }
```

```
}

//////////////////////////////////////////////////////////////////////////////
//                图像参数
//////////////////////////////////////////////////////////////////////////////

void vision_parameter_proc(void)
{
    uint8    i;
    switch (key_value)
    {
       case 0x01:
         {
              keymenu_curr_state                                    =
KeyMenuTable[keymenu_curr_state].KeyEscState;
              KeyMenuTable[keymenu_curr_state].MenuSetup = 0;
              LCD_clear();
         }
         break;
       case 0x04:
         {
              if (0 == KeyMenuTable[keymenu_curr_state].MenuSetup)
              {
                   KeyMenuTable[keymenu_curr_state].MenuSubstate++;
                   if      (KeyMenuTable[keymenu_curr_state].MenuSubstate      >
(VISIONDBG_MENU_LEN-1))
                        KeyMenuTable[keymenu_curr_state].MenuSubstate = 0;
              }
              else
              {
                   if
(VisionDebugParam[KeyMenuTable[keymenu_curr_state].MenuSubstate]              >
VisionDebugParamMin[KeyMenuTable[keymenu_curr_state].MenuSubstate] )
                                        {

VisionDebugParam[KeyMenuTable[keymenu_curr_state].MenuSubstate]               -=
VisionDebugParamAdj[KeyMenuTable[keymenu_curr_state].MenuSubstate];
                                   zhuwei_write_flash();
                                        }
                              }
         }
         break;
       case 0x02:
```

```
            {
                if (0 == KeyMenuTable[keymenu_curr_state].MenuSetup)
                {
                    if (KeyMenuTable[keymenu_curr_state].MenuSubstate == 0)
                        KeyMenuTable[keymenu_curr_state].MenuSubstate        =
VISIONDBG_MENU_LEN;
                    KeyMenuTable[keymenu_curr_state].MenuSubstate--;
                }
                else
                {
                    if
(VisionDebugParam[KeyMenuTable[keymenu_curr_state].MenuSubstate]              <
VisionDebugParamMax[KeyMenuTable[keymenu_curr_state].MenuSubstate] )
                                                {

VisionDebugParam[KeyMenuTable[keymenu_curr_state].MenuSubstate]              +=
VisionDebugParamAdj[KeyMenuTable[keymenu_curr_state].MenuSubstate];
                                        zhuwei_write_flash();
                                                }
                                    }
            }
            break;
          case 0x08:
            {
                KeyMenuTable[keymenu_curr_state].MenuSetup
= !KeyMenuTable[keymenu_curr_state].MenuSetup;
            }
            break;
          default:
            {
                LCD_set_XY(8, 0);
                LCD_write_english("ImageProc Settings");

                for (i=0; i<VISIONDBG_MENU_LEN; i++)
                {
                    LCD_set_XY(8, i+2);
                    LCD_write_english(VisionDebugStrings[i]);
                    LCD_set_XY(96, i+2);
                    LCD_write_int(VisionDebugParam[i]);
                }

                for (i=0; i<VISIONDBG_MENU_LEN; i++)
                {
```

```
                    LCD_set_XY(0, i+2);
                    if (KeyMenuTable[keymenu_curr_state].MenuSubstate == i)
                    {
                        if (0 == KeyMenuTable[keymenu_curr_state].MenuSetup)
                        {
                            LCD_write_english(">");
                        }
                        else
                        {
                            LCD_write_english("*");
                        }
                    }
                    else
                        LCD_write_english(" ");
                }
            }
            break;
        }

    }

    ////////////////////////////////////////////////////////////////////////////
    //                速度环参数
    ////////////////////////////////////////////////////////////////////////////

    void speed_pid_window_proc(void)
    {
        uint8   i;
        switch (key_value)
        {
           case 0x01:
             {
                 keymenu_curr_state                                          =
KeyMenuTable[keymenu_curr_state].KeyEscState;
                 LCD_clear();
             }
             break;
           case 0x04:
             {
                 if (0 == KeyMenuTable[keymenu_curr_state].MenuSetup)
                 {
                     KeyMenuTable[keymenu_curr_state].MenuSubstate++;
                     if     (KeyMenuTable[keymenu_curr_state].MenuSubstate     >
```

```
(SPEED_MENU_LEN-1))
                        KeyMenuTable[keymenu_curr_state].MenuSubstate = 0;
                }
                else
                {
                    if
(SpeedParam[KeyMenuTable[keymenu_curr_state].MenuSubstate]                    >
SpeedParamMin[KeyMenuTable[keymenu_curr_state].MenuSubstate] )
                                        {

SpeedParam[KeyMenuTable[keymenu_curr_state].MenuSubstate]                    -=
SpeedParamAdj[KeyMenuTable[keymenu_curr_state].MenuSubstate];
                                    zhuwei_write_flash();
                                        }
                                }
            }
            break;
        case 0x02:
            {
                if (0 == KeyMenuTable[keymenu_curr_state].MenuSetup)
                {
                    if (KeyMenuTable[keymenu_curr_state].MenuSubstate == 0)
                        KeyMenuTable[keymenu_curr_state].MenuSubstate      =
SPEED_MENU_LEN;
                    KeyMenuTable[keymenu_curr_state].MenuSubstate--;
                }
                else
                {
                    if
(SpeedParam[KeyMenuTable[keymenu_curr_state].MenuSubstate]                    <
SpeedParamMax[KeyMenuTable[keymenu_curr_state].MenuSubstate] )
                                        {

SpeedParam[KeyMenuTable[keymenu_curr_state].MenuSubstate]                    +=
SpeedParamAdj[KeyMenuTable[keymenu_curr_state].MenuSubstate];
                                            zhuwei_write_flash();
                                        }
                }
            }
            break;
        case 0x08:
            {
                KeyMenuTable[keymenu_curr_state].MenuSetup
```

```
= !KeyMenuTable[keymenu_curr_state].MenuSetup;
                }
                break;
            default:
                {
                    LCD_set_XY(8, 0);
                    LCD_write_english("Speed PID Settings");

                    for (i=0; i<SPEED_MENU_LEN; i++)
                    {
                        LCD_set_XY(8, i+2);
                        LCD_write_english(SpeedPIDMenuStrings[i]);

                        LCD_set_XY(96, i+2);
                        LCD_write_int(SpeedParam[i]);
                    }

                    for (i=0; i<SPEED_MENU_LEN; i++)
                    {
                        LCD_set_XY(0, i+2);
                        if (KeyMenuTable[keymenu_curr_state].MenuSubstate == i)
                        {
                            if (0 == KeyMenuTable[keymenu_curr_state].MenuSetup)
                            {
                                LCD_write_english(">");
                            }
                            else
                            {
                                LCD_write_english("*");
                            }
                        }
                        else
                            LCD_write_english(" ");
                    }
                }
                break;
        }

    }

    ////////////////////////////////////////////////////////////////////////////
    //              方向环参数
    ////////////////////////////////////////////////////////////////////////////
```

```
    void dir_pid_window_proc(void)
    {
        uint8    i;
        switch (key_value)
        {
           case 0x01:
             {
                  keymenu_curr_state                                    =
KeyMenuTable[keymenu_curr_state].KeyEscState;
                  LCD_clear();
             }
             break;
           case 0x04:
             {
                  if (0 == KeyMenuTable[keymenu_curr_state].MenuSetup)
                  {
                       KeyMenuTable[keymenu_curr_state].MenuSubstate++;
                       if      (KeyMenuTable[keymenu_curr_state].MenuSubstate      >
(DIR_MENU_LEN-1))
                            KeyMenuTable[keymenu_curr_state].MenuSubstate = 0;
                  }
                  else
                  {
                       if
(DirectionParam[KeyMenuTable[keymenu_curr_state].MenuSubstate]                >
DirectionParamMin[KeyMenuTable[keymenu_curr_state].MenuSubstate] )
                                                {

    DirectionParam[KeyMenuTable[keymenu_curr_state].MenuSubstate]              -=
DirectionParamAdj[KeyMenuTable[keymenu_curr_state].MenuSubstate];
                                                        zhuwei_write_flash();
                                                }
                                       }
             }
             break;
           case 0x02:
             {
                  if (0 == KeyMenuTable[keymenu_curr_state].MenuSetup)
                  {
                       if (KeyMenuTable[keymenu_curr_state].MenuSubstate == 0)
                            KeyMenuTable[keymenu_curr_state].MenuSubstate        =
DIR_MENU_LEN;
```

```
                    KeyMenuTable[keymenu_curr_state].MenuSubstate--;
                }
                else
                {
                    if
(DirectionParam[KeyMenuTable[keymenu_curr_state].MenuSubstate]                    <
DirectionParamMax[KeyMenuTable[keymenu_curr_state].MenuSubstate] )
                    {

DirectionParam[KeyMenuTable[keymenu_curr_state].MenuSubstate]                    +=
DirectionParamAdj[KeyMenuTable[keymenu_curr_state].MenuSubstate];
                                    zhuwei_write_flash();
                                        }
                                }
            }
            break;
          case 0x08:
            {
                KeyMenuTable[keymenu_curr_state].MenuSetup
= !KeyMenuTable[keymenu_curr_state].MenuSetup;
            }
            break;
          default:
            {
                LCD_set_XY(8, 0);
                LCD_write_english("Dir PID Settings");

                for (i=0; i<DIR_MENU_LEN; i++)
                {
                    LCD_set_XY(8, i+2);
                    LCD_write_english(DirPIDMenuStrings[i]);
                    LCD_set_XY(96, i+2);
                    LCD_write_int(DirectionParam[i]);
                }

                for (i=0; i<DIR_MENU_LEN; i++)
                {
                    LCD_set_XY(0, i+2);
                    if (KeyMenuTable[keymenu_curr_state].MenuSubstate == i)
                    {
                        if (0 == KeyMenuTable[keymenu_curr_state].MenuSetup)
                        {
                            LCD_write_english(">");
```

```
                        }
                        else
                        {
                            LCD_write_english("*");
                        }
                    }
                    else
                        LCD_write_english(" ");
                }
            }
            break;
        }

    }


    ////////////////////////////////////////////////////////////////////////////
    //              舵机参数
    ////////////////////////////////////////////////////////////////////////////


    void servo_window_proc(void)
    {
        uint8   i;
        switch (key_value)
        {
          case 0x01:
            {
                keymenu_curr_state                                          =
KeyMenuTable[keymenu_curr_state].KeyEscState;
                KeyMenuTable[keymenu_curr_state].MenuSetup = 0;
                LCD_clear();
            }
            break;
          case 0x04:
            {
                if (0 == KeyMenuTable[keymenu_curr_state].MenuSetup)
                {
                    KeyMenuTable[keymenu_curr_state].MenuSubstate++;
                    if      (KeyMenuTable[keymenu_curr_state].MenuSubstate      >
(SERVO_MENU_LEN-1))
                        KeyMenuTable[keymenu_curr_state].MenuSubstate = 0;
                }
```

```
                else
                {
                    if
(ServoParam[KeyMenuTable[keymenu_curr_state].MenuSubstate]                    >
ServoParamMin[KeyMenuTable[keymenu_curr_state].MenuSubstate] )
                                        {

    ServoParam[KeyMenuTable[keymenu_curr_state].MenuSubstate]                -=
ServoParamAdj[KeyMenuTable[keymenu_curr_state].MenuSubstate];
                                                zhuwei_write_flash();
                                        }

                                    }
            }
            break;
          case 0x02:
            {
                if (0 == KeyMenuTable[keymenu_curr_state].MenuSetup)
                {
                    if (KeyMenuTable[keymenu_curr_state].MenuSubstate == 0)
                        KeyMenuTable[keymenu_curr_state].MenuSubstate        =
SERVO_MENU_LEN;
                    KeyMenuTable[keymenu_curr_state].MenuSubstate--;
                }
                else
                {
                    if
(ServoParam[KeyMenuTable[keymenu_curr_state].MenuSubstate]                    <
ServoParamMax[KeyMenuTable[keymenu_curr_state].MenuSubstate] )
                                        {

    ServoParam[KeyMenuTable[keymenu_curr_state].MenuSubstate]                +=
ServoParamAdj[KeyMenuTable[keymenu_curr_state].MenuSubstate];
                                                zhuwei_write_flash();
                                        }
                }
            }
            break;
          case 0x08:
            {
                KeyMenuTable[keymenu_curr_state].MenuSetup
= !KeyMenuTable[keymenu_curr_state].MenuSetup;
            }
```

```
            break;
          default:
            {
                LCD_set_XY(8, 0);
                LCD_write_english("Servo Settings");

                for (i=0; i<SERVO_MENU_LEN; i++)
                {
                    LCD_set_XY(8, i+2);
                    LCD_write_english(ServoMenuStrings[i]);
                    LCD_set_XY(96, i+2);
                    LCD_write_int(ServoParam[i]);
                }

                for (i=0; i<SERVO_MENU_LEN; i++)
                {
                    LCD_set_XY(0, i+2);
                    if (KeyMenuTable[keymenu_curr_state].MenuSubstate == i)
                    {
                        if (0 == KeyMenuTable[keymenu_curr_state].MenuSetup)
                        {
                            LCD_write_english(">");
                        }
                        else
                        {
                            LCD_write_english("*");
                        }
                    }
                    else
                        LCD_write_english(" ");
                }
            }
            break;
        }

    }

    static uint8    key_history[KEY_BOARD_HISTORY_LEN];

    void init_kbd(void)
    {
        uint8_t i;
        gpio_init(PTC16, GPI, 0);      //kGPIO_Mode_IPU
```

```
            port_init_NoALT(PTC16,PULLUP);
    gpio_init(PTC17, GPI, 0);
            port_init_NoALT(PTC17,PULLUP);
    gpio_init(PTE6, GPI, 0);
            port_init_NoALT(PTE6,ALT1|PULLUP);
    gpio_init(PTE0, GPI, 0);
            port_init_NoALT(PTE0,PULLUP);
    for (i=0; i<KEY_BOARD_HISTORY_LEN; i++)
        key_history[i] = 0;
}

void keyboard_scan(void)
{
    uint8   i, j;
    uint8   ckey = 0;
    if (!gpio_get( PTC17   ))
            {
               DELAY_MS(10);
               if (!gpio_get( PTC17   ))
                  ckey |= 0x01;
            }
    if (!gpio_get( PTC16   ))
            {
               DELAY_MS(10);
               if (!gpio_get( PTC16   ))
                  ckey |= 0x02;
            }
            if (!gpio_get( PTE0   ))
            {
               DELAY_MS(10);
               if (!gpio_get( PTE0   ))
                  ckey |= 0x04;
            }
            if (!gpio_get( PTE6   ))
            {
               DELAY_MS(10);
               if (!gpio_get( PTE6   ))
                  ckey |= 0x08;
            }
    for (i=1; i<KEY_BOARD_HISTORY_LEN; i++)
    {
        key_history[i-1] = key_history[i];
    }
```

```
    key_history[KEY_BOARD_HISTORY_LEN-1]=ckey;
    for (i=0; i<KEY_BOARD_NUM; i++)
    {
        if (key_value & 0x01 << i)
        {
            for (j=0; j<KEY_BOARD_HISTORY_LEN; j++)
            {
                if (1 == (key_history[j] & (0x01 << i)))
                    goto nopress;
            }
            key_value &= ~(0x01 << i);
          nopress:;
        }
        else
        {
            for (j=0; j<KEY_BOARD_HISTORY_LEN; j++)
            {
                if (0 == (key_history[j] & (0x01 << i)))
                    goto nopress2;
            }
            key_value |= 0x01 << i;
          nopress2:;
        }
    }
}

void display_proc(void)
{
    KeyMenuTable[keymenu_curr_state].WindowProc();
    if (0 != key_value)
    {
        key_history[KEY_BOARD_HISTORY_LEN-1] = 0;
    }
}


/*************************函数*******************************
* 函数名称：zhuwei_write_flash
* 功能说明：写入 flash 中的有关数据
* 备注： 重要，不能轻易更改
*************************************************************/
void zhuwei_write_flash(void)
{
```

```
    flash_erase_sector(flash_sector_number);                    //擦除扇
区

    flash_write(flash_sector_number, ServoMid_SECTOR_NUM, ServoParam[0]);

    flash_write(flash_sector_number,          ServoRight_SECTOR_NUM,
ServoParam[1]);

    flash_write(flash_sector_number, ServoLeft_SECTOR_NUM, ServoParam[2]);

    flash_write(flash_sector_number,          DirectionParam_P_SECTOR_NUM,
DirectionParam[0]);

    flash_write(flash_sector_number,          DirectionParam_I_SECTOR_NUM,
DirectionParam[1]);

    flash_write(flash_sector_number,          DirectionParam_D_SECTOR_NUM,
DirectionParam[2]);

    flash_write(flash_sector_number,
DirectionParam_S3010RowK_SECTOR_NUM, DirectionParam[3]);

    flash_write(flash_sector_number,          SpeedParam_P_SECTOR_NUM,
SpeedParam[0]);

    flash_write(flash_sector_number,          SpeedParam_I_SECTOR_NUM,
SpeedParam[1]);

    flash_write(flash_sector_number, SpeedParam_SpeedSetMax_SECTOR_NUM,
SpeedParam[2]);

    flash_write(flash_sector_number,          SpeedParam_D_SECTOR_NUM,
SpeedParam[3]);

    flash_write(flash_sector_number,  SpeedParam_SpeedSetMin_SECTOR_NUM,
SpeedParam[4]);

    flash_write(flash_sector_number,          VisionDebugParam_SECTOR_NUM,
VisionDebugParam[0]);

    flash_write(flash_sector_number,
VisionDebugParam_Stop_YesOrNo_SECTOR_NUM, VisionDebugParam[3]);
```

```
        flash_write(flash_sector_number,
VisionDebugParam_Delay_Time_SECTOR_NUM, VisionDebugParam[1]);

        flash_write(flash_sector_number,
VisionDebugParam_Delay_YesOrNo_SECTOR_NUM, VisionDebugParam[2]);

        flash_write(flash_sector_number,
VisionDebugParam_QuiAcc_YesOrNo_SECTOR_NUM, VisionDebugParam[4]);
    }


    /*************************函数*******************************
    * 函数名称：FLASH_READ
    * 功能说明：读取 flash 中的有关数据
    * 备注： 重要，不能轻易更改
    ************************************************************/
    void FLASH_READ(void)
    {
        ServoParam[0] =                 flash_read(flash_sector_number,
ServoMid_SECTOR_NUM, uint32);   //读取 4 字节
        ServoParam[1] =                 flash_read(flash_sector_number,
ServoRight_SECTOR_NUM, uint32);
        ServoParam[2] =                 flash_read(flash_sector_number,
ServoLeft_SECTOR_NUM, uint32);
        DirectionParam[0] =             flash_read(flash_sector_number,
DirectionParam_P_SECTOR_NUM, uint32);
        DirectionParam[1] =             flash_read(flash_sector_number,
DirectionParam_I_SECTOR_NUM, uint32);
        DirectionParam[2] =             flash_read(flash_sector_number,
DirectionParam_D_SECTOR_NUM, uint32);
        DirectionParam[3] =             flash_read(flash_sector_number,
DirectionParam_S3010RowK_SECTOR_NUM, uint32);
        SpeedParam[0] =                 flash_read(flash_sector_number,
SpeedParam_P_SECTOR_NUM, uint32);
        SpeedParam[1] =                 flash_read(flash_sector_number,
SpeedParam_I_SECTOR_NUM, uint32);
        SpeedParam[2] =                 flash_read(flash_sector_number,
SpeedParam_SpeedSetMax_SECTOR_NUM, uint32);
        SpeedParam[3] =                 flash_read(flash_sector_number,
SpeedParam_D_SECTOR_NUM, uint32);
        SpeedParam[4] =                 flash_read(flash_sector_number,
SpeedParam_SpeedSetMin_SECTOR_NUM, uint32);
        VisionDebugParam[3] =           flash_read(flash_sector_number,
```

```
VisionDebugParam_Stop_YesOrNo_SECTOR_NUM, uint32);
        VisionDebugParam[0]  =                flash_read(flash_sector_number,
VisionDebugParam_SECTOR_NUM, uint32);
        VisionDebugParam[1]  =                flash_read(flash_sector_number,
VisionDebugParam_Delay_Time_SECTOR_NUM, uint32);
        VisionDebugParam[2]  =                flash_read(flash_sector_number,
VisionDebugParam_Delay_YesOrNo_SECTOR_NUM, uint32);
        VisionDebugParam[4]  =                flash_read(flash_sector_number,
VisionDebugParam_QuiAcc_YesOrNo_SECTOR_NUM, uint32);
    }
```