第七届“飞思卡尔”杯全国大学生智能汽车竞赛

技 术 报 告

学　　校：　武汉大学

队伍名称：　IRIS-Foresight

参赛队员：　区金鹏

　　蒋旦

　　王振华

带队教师：　杨飞

关于技术报告和研究论文使用授权的说明

本人完全了解第七届“飞思卡尔”杯全国大学生智能汽车竞赛关保留、使用技术报告和研究论文的规定，即：参赛作品著作权归参赛者本人，比赛组委会和飞思卡尔半导体公司可以在相关主页上收录并公开参赛作品的设计方案、技术报告以及参赛模型车的视频、图像资料，并将相关内容编纂收录在组委会出版论文集中。

参赛队员签名：_____

带队教师签名：_____

日　　期：_____

摘要

本文主要阐述一个能沿两边有黑线的白道循迹的智能小车系统的设计过程。

本小车以飞思卡尔公司生产的MK60DN512为系统控制器,利用CCD摄像头获取前方黑线信息，利用偏差计算，插值，曲线拟合，S 弯判断等方法对图像进行处理，根据图像处理得到的黑线偏差，黑线拟合斜率，黑线曲率半径等关键信息，用 PID 算法对舵机进行控制，PID+Bangbang 对电机进行控制。

本文将按硬件到软件的顺序，重点从模型车机械机构的调整与改造，硬件电路的设计与调试，图像的获取与处理算法，舵机，电机控制策略的设计方面进行介绍，在明确其中各方面的优点与不足的基础上，提出了改进的方法与设想。

目录

第一章：绪论

1.1 课题背景

受教育部高等教育司委托，高等学校自动化专业教学指导分委员负责主办全国大学生智能车竞赛。该项比赛已列入教育部主办的全国五大竞赛之一。该项赛事从 2006 年开始，在清华大学等学校举办过六届，取得了良好的效果。比赛规定“参赛选手须使用大赛组委会统一提供的竞赛车模，采用飞思卡尔微控制器作为核心控制单元，自主构思控传感器信号采集处理、控制算法及执行、动力电机驱动、转向舵机控制等，完成智能车工程制作及调试，于指定日期与地点参加场地比赛。参赛队伍之名次（成绩）由赛车现场成功完成赛道比赛时间为主，技术方案及制作工程质量评分为辅来决定[1]。

第七届全国大学生智能车竞赛决赛将于 2012 年 8 月中下旬在南京师范大学举行，本文所介绍的智能车设计工作即是围绕着这一比赛主题。

1.2 国外情况

智能车竞赛最早始于韩国，2000 年韩国汉阳大学承办了第一届智能车竞赛，并由飞思卡尔（Freescale）公司赞助，每年大约有 100 余支大学生队伍参加比赛。智能车的设计制作，其专业知识设计控制、模式识别、传感技术、汽车电子、电器、计算机和机械等多个学科，对学生的知识融合和实践动手能力的培养，以及高等学校控制和汽车电子学科学术水平的提高，具有良好的产期推动作用[2]。

第二章：智能车系统硬件的设计与调试

如果说把一个系统比作一个人的话，那么软件即是这个系统大脑里的思想，而硬件则是这个系统的躯体，倘若一个强大的思想没有一个健康的躯体支撑的话，那么它也发挥不了其强大。因此一个稳定，可靠的硬件系统对于整个系统功能的实现有着至关重要的作用。本章将重点介绍智能车系统硬件的设计与调试，包括模型车车机械结构与硬件电路两部分。前部分又细分为汽车行驶的数学模型，舵机的安装方式与输出臂长的选取，模型车

前轮结构的调整，摄像头支架的设计与安装。后部分又分为 MK60DN512 单片机，电机驱动模块，车速检测模块，电池供电模块，图像采集模块，辅助调试模块。

2.1 模型车机械结构的设计与调整

2.1.1 汽车行驶的数学模型

根据汽车理论， 假设轮胎不打滑，并忽略轮胎所受的重力作用产生的形变以及左右两侧轮胎由于受力不均产生的形变，即可得到理想的汽车转向模型：如图 2.1 所示，即左右两轮的轴线与后轮的主轴，三点交于车身中心所处道路位置的曲率中心. 不失一般性，这里只讨论右转的情形.

对于图 2.1，设左轮转向为θ_L，右轮角分别为θ_R，对于以上模型，显然有如下关系：

$$\theta_L = \angle PAQ = \angle POC, \theta_R = \angle MAN = \angle NOD$$

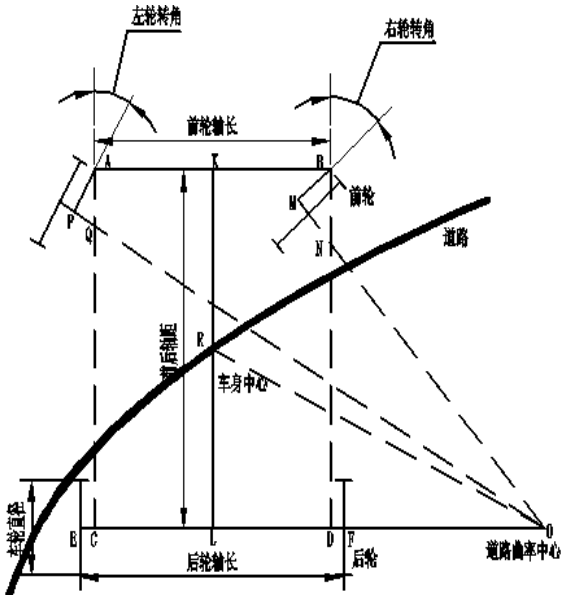
$$OC - OD = CD$$

于是得到在理想状态下，汽车的过弯时的转角方程：

$\frac{(L_{轴距} - L_{杆}/\cos\theta_L)}{\tan\theta_L} - \frac{(L_{轴距} - L_{杆}/\cos\theta_R)}{\tan\theta_R} = L_{前}$	公式(一)

同样也可以得到右转时转弯半径R与右轮转角θ_R之间的关系：

$R = \sqrt{OL^2 + (\frac{L_{轴距}}{2})^2} = \sqrt{(\frac{L_{轴距} - \frac{L_{杆}}{\cos\theta_R}}{\tan\theta_R} + \frac{L_{前}}{2})^2 + (\frac{L_{轴距}}{2})^2}$	公式（二）



图中：

AP, BN为连接杆与主销之间的杆长$L_{杆}$

AB, CD为前轮轴长$L_{前}$，后轮轴长$L_{后}$

KL为前轴与后轴的轴距$L_{轴距}$

图 2.1

采用第一届“飞思卡尔”智能车大赛组委会提供的韩国 Matiz 系列 1：10 模型车的参数对方程（1），方程（2）进行仿真.

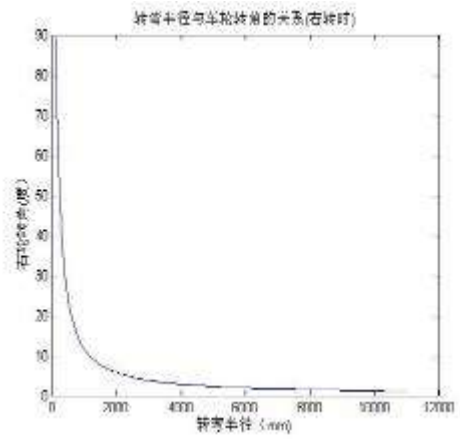


图 2.2

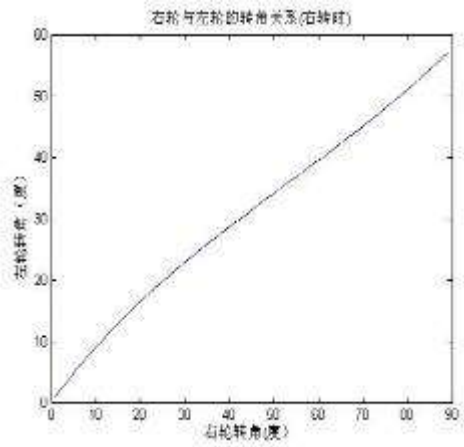


图 2.3

结论：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。

2.1.2 舵机安装方式与输出臂长的选取

2.1.2.1 舵机的工作原理

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“飞思卡尔”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.然而舵机是具有较大延迟特性的对象，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，则会大大提供舵机过弯的响应速度。而这与不仅与舵机的安装方式有关，也与舵机输出臂的长度有极大的关系，输出臂越长越有利提高响应速度，但舵机在输出力矩相同的条件下，力臂越长，作用力越小，所以当转轮遇到较大转向阻力时，会影响舵机对转向轮控制的精度，甚至反而使转向轮的反应速度变慢。因此找到一个较好的舵机安装方式及其最优的输出臂长对提高舵机的反应速度十分重要。

图 2.4

设舵机的工作速度为$v(rad/s)$，额定电压下的堵转力矩为$M_{\max}$，在比赛中拉动前轮转动所需的最大驱动力为$f_{\max}$.因此在一定舵机安装方式下，忽略其他次要因素，舵机的转角$\theta_{舵}$与车轮转角$\theta_{轮}$必定存在一定的函数关系.

$$\theta_{舵}=f(\theta_{轮},L_{臂}) \qquad L_{臂}\text{为舵机输出臂长}$$

因此在安装方式一定的情况下下，最优输出臂$L_{臂}$应该满足如下关系

$$Min\ \frac{\delta\overline{\theta_{舵}}}{\delta\overline{\theta_{轮}}}$$

$$S.T \quad (\vec{f} \times \vec{L}_{臂})_{max} \leq M_{m}$$

由于在模型车空间较小，因此舵机的安装方式受到很大局限，下面讨论舵机两种的典型安装方式并给各自的最优输出臂长。

2.1.2.2 舵机的安装方式一（输出臂朝上）

对于此种安装方式，以主销为Z轴，前轴为Y轴建立空间直角坐标系如图 2.6. 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.

图 2.5

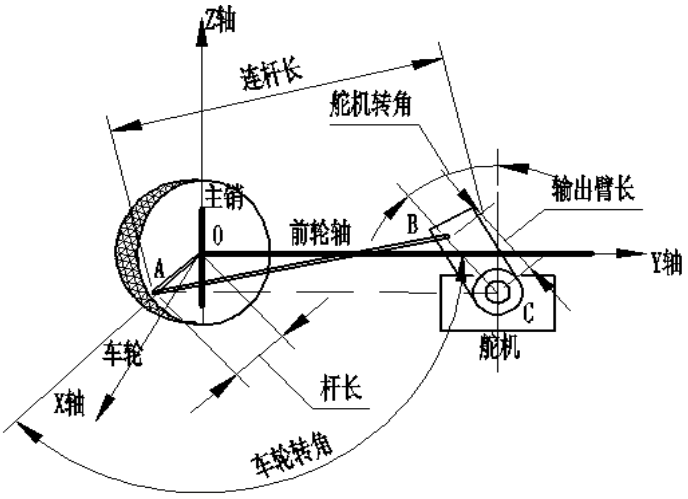


图 2.6

显然对于图 2.6 坐标系，A, B两点的坐标为

$$A(L_{杆}\sin\theta_{轮}, L_{杆}\cos\theta, 0) \qquad B(L_{杆}, \frac{L_{前}}{2} - L_{臂}\sin\theta_{舵} \quad L_{臂} \quad \theta_{舵}\cos$$

即 $L_{AB} = \sqrt{(L_{杆} - L_{杆}\sin\theta_{轮})^2 + (\frac{L_{前}}{2} - L_{臂}\sin\theta_{舵} - L_{杆}\cos\theta)^2 + L_{臂}^2\cos^2\theta_{舵}}$

而 $L_{AB} = \sqrt{\frac{L_{前}^2}{4} + L_{臂}^2}$， 于是得到如下$\theta_{轮}$与$L_{臂}$的关系方程：

$\sqrt{\frac{L_{前}^2}{4}+L_{臂}^{\ 2}}=\sqrt{\begin{matrix}(L_{杆}-L_{杆}\sin\theta_{轮})^2+(\frac{L_{前}}{2}-L_{臂}\sin\theta_{舵}-L_{杆}\cos\theta_{舵})^2\\+L_{臂}^{\ 2}\cos^2\theta_{舵}\end{matrix}}$	公式（三)

2.1.2.3 舵机的安装方式二（输出臂朝下）

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 2.8 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 2.7

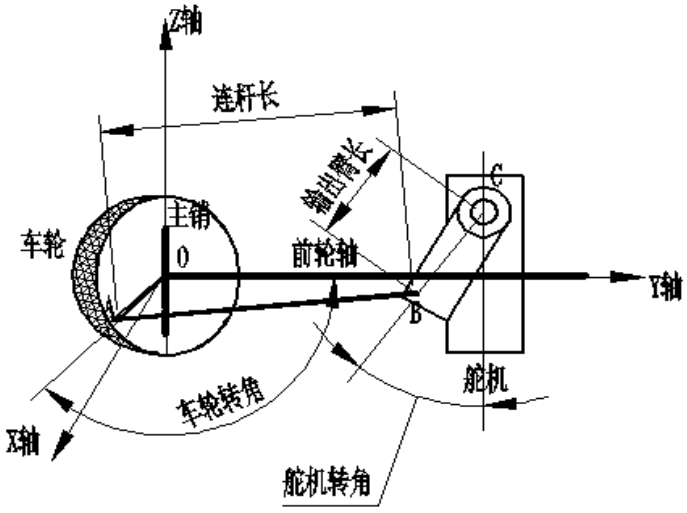


图 2.8

显然对于图 2.8 坐标系， A,B 两点的坐标

$A(L_{杆}\sin\theta_{轮},L_{杆}\cos\theta,0)$　　$B(L_{杆},\frac{L_{前}}{2}-L_{臂}\sin\theta_{舵}\ \ L_{臂}-(1\ \ \theta_{舵}\cos$

即 $L_{AB}=\sqrt{(L_{杆}-L_{杆}\sin\theta_{轮})^2+(\frac{L_{前}}{2}-L_{臂}\sin\theta_{舵}-L_{杆}\cos\theta)^2+L_{臂}^{\ 2}(1-\cos\ \theta_{舵})^2}$

而 $L_{AB}=\frac{L_{前}}{2}$，于是得到如下 $\theta_{轮}$ 与 $L_{臂}$ 的关系方程：

$\frac{L_{前}}{2}=\sqrt{\begin{matrix}(L_{杆}-L_{杆}\sin\theta_{轮})^2+(\frac{L_{前}}{2}-L_{臂}\sin\theta_{舵}-L_{杆}\cos\theta)^2\\ +L_{臂}^{\ 2}(1-\cos\theta_{舵})^2\end{matrix}}$	公式（四)

2.1.2.4 两种舵机的安装方式的比较

由于模型车的循迹要求不是十分精密，因此车轮的转角可以不是连续的，一度的精度足以满足巡线要求，因此对于以上两种安装舵机的方式，给定 $L_{臂}$，让 $\theta_{轮}$ 以一度为步长从 1 度递增到 120 度，利用公式（三），公式（四）求出相应的 $\theta_{舵}$，然后对这 120 组（$\theta_{轮},\theta_{舵}$）进行线性拟合，拟合所得的直线斜率越小，说明舵机的反应越快。

即 $Min\quad slope=\frac{N\sum_{i=1}^{N}\theta_{舵i}\theta_{轮i}-\sum_{i=1}^{N}\theta_{舵i}\sum_{i=1}^{N}\theta_{轮i}}{N\sum_{i=1}^{N}\theta_{舵i}^2-(\sum_{i=1}^{N}\theta_{舵i})^2}$（$\theta_{轮i}=(1,2,\mathrm{L}\ ,120,N=120)$

$$S.T\qquad (\vec{f}\times\vec{L}_{臂})_{max}\le M_{m}$$

采用智能车大赛提供的 Futaba S3010 舵机的参数，利用 Matlab 编程，最终算得安装方式（一）和安装方式（二）的最优输出臂长 $L_{臂1}$，$L_{臂2}$,以及在最优输出臂长下的拟合直线斜率 $slope_1$，$slope_2$，最终结果如下：$L_{臂1}=40mm$ $L_{臂2}=30mm$，$slope_1=0.3097$，$slope_2=0.2613$ 显然安装方式（二）要优于安装方式（一）

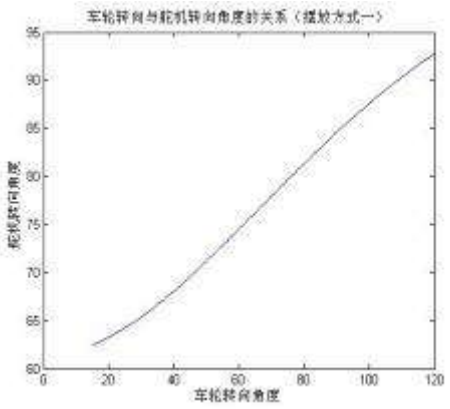


图 2.9

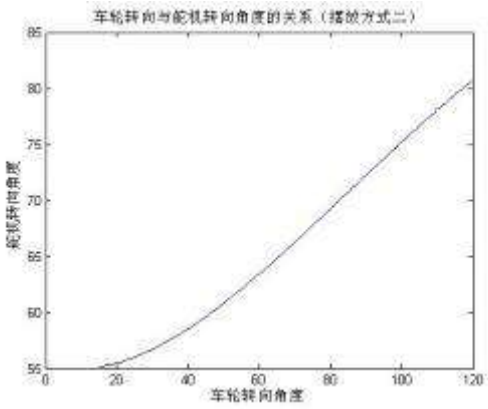


图 2.10

2.1.3 模型车前轮结构的调整

现代汽车在正常行驶的过程种，为了使汽车直线行驶稳定，转向稳定，转向轻便，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：主销后倾，主销内倾，车轮外倾和前束。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。

2.1.3.1 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。模型车可以通过增减黄色垫片的数量来改变主销后倾角：每侧有 4 片垫片，前 2 后 2，后倾角为 0 度；前 1 后 3，后倾角为 2~3 度，前 0 后 4，后倾角为 4~6 度，欲使模型车转向灵活，一般将后倾角设定为 0 度。

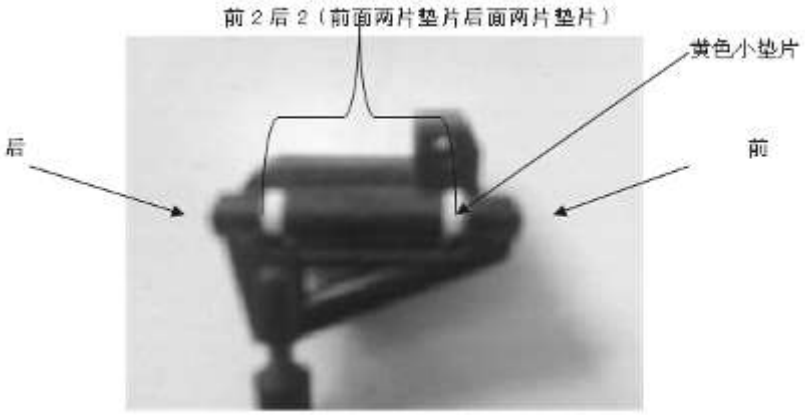


图 2.11

2.1.3.2 主销前倾

主销前倾是指在汽车的横向平面内向内倾斜的一个角度，即主销轴线与地面垂直线在汽车横向断面内的夹角。主销前倾也有使车轮自动回正的作用。模型车可以通过调整如图 2.12 所示螺杆的长度来改变主销的内倾角，调整范围为 0~10 度时对模型车的性能影响不大，可根据在赛道上的调试情况自行调节。

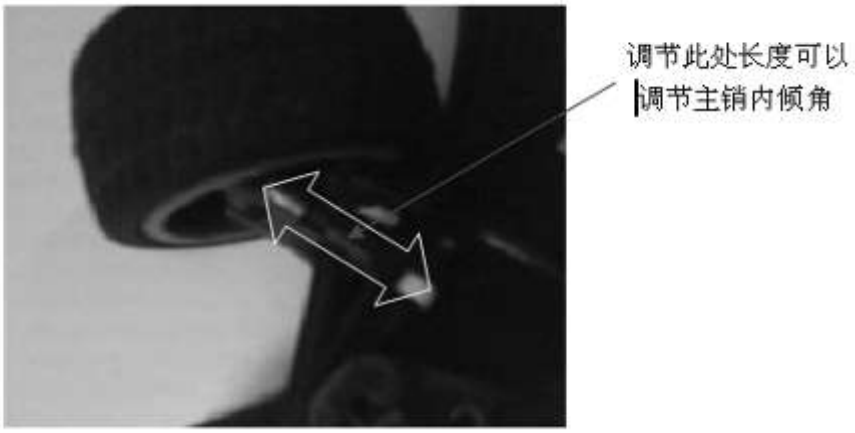


图 2.12

2.1.3.3 前轮外倾角与前束

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。当轮胎呈“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”。前轮外倾角可以减少转向阻力，使汽车转向轻便。模型车提供了序号为 EX-19 的配件用来调节前轮外倾的角度。

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”。如图 2.13 所示，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。

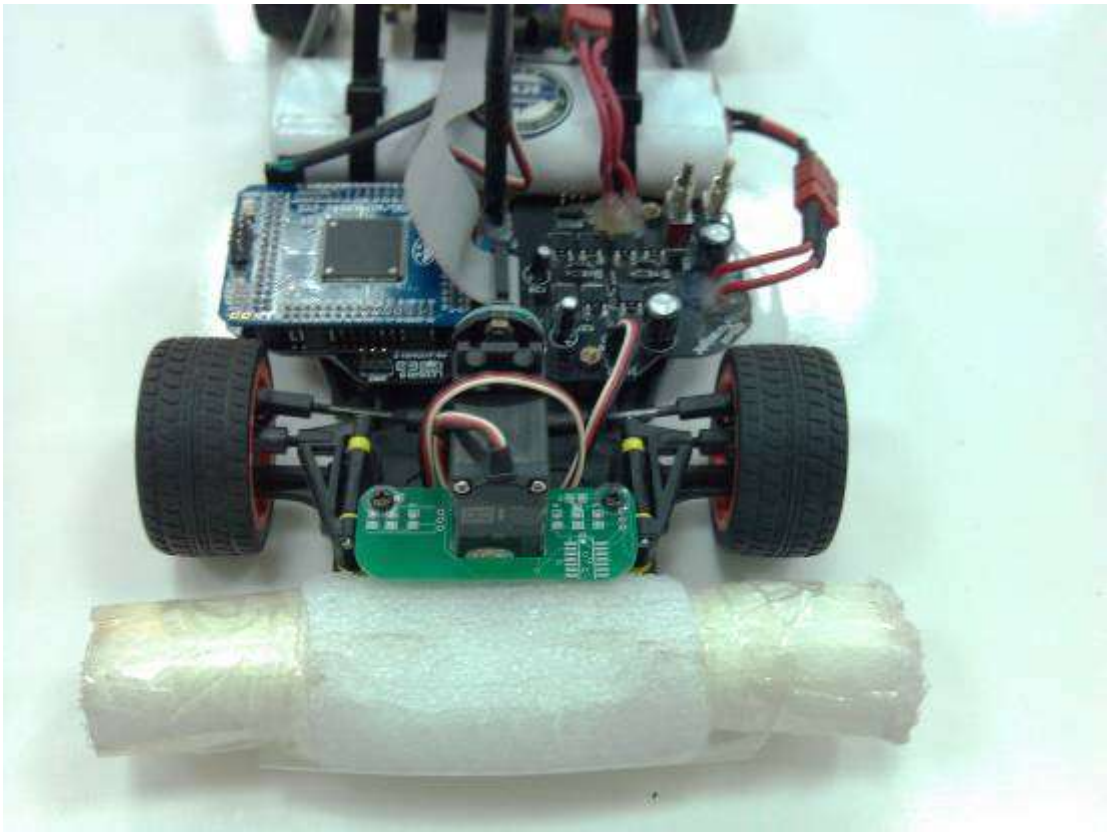


图 2.13 前轮调节

2.1.4 摄像头支架的设计与安装

由于智能车在调试过程中需要对各种硬件参数进行调整,其中就包括摄像头的安装高度，摄像头与铅垂线的夹角。因此摄像头的安装方式既要保证安装牢靠，也应使安装位置可调， 同时应该减轻重量，使重心降低，依据以上原则，分别设计了摄像头底座（固定于车模底盘），摄像头支杆，摄像头固定座（安放摄像头）和整车效果图。

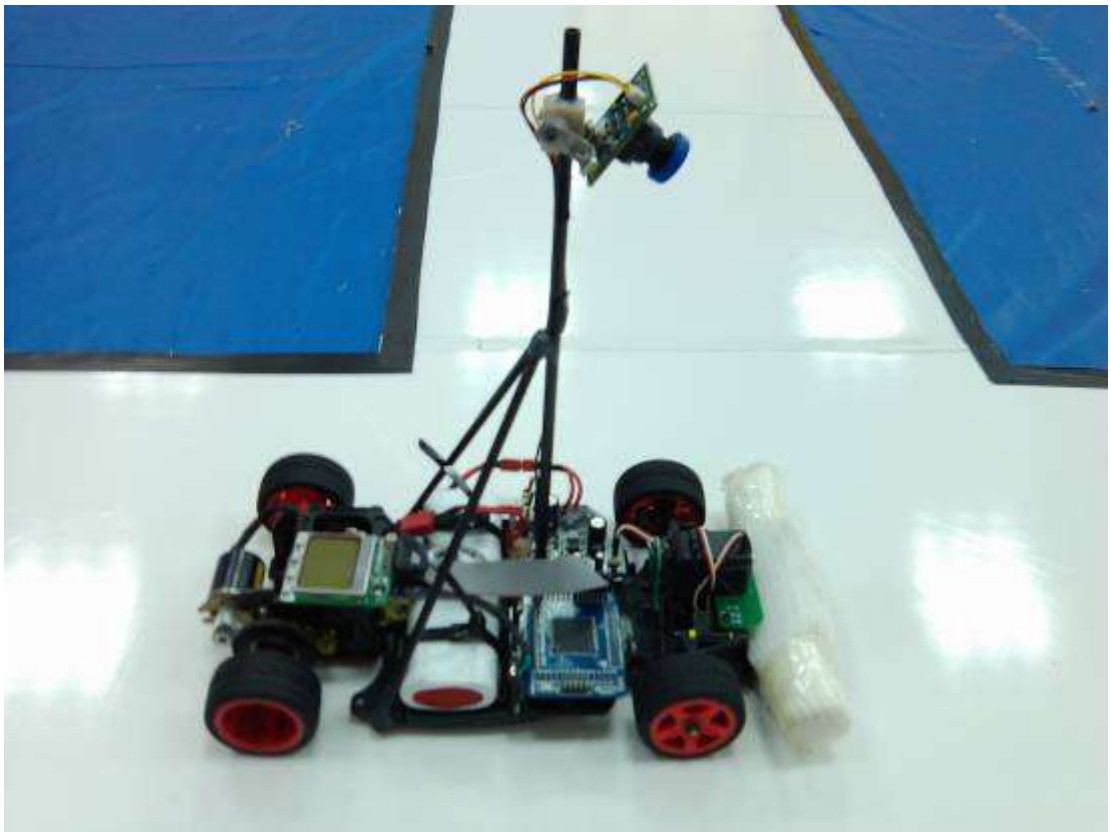


图 2.15　摄像头支杆

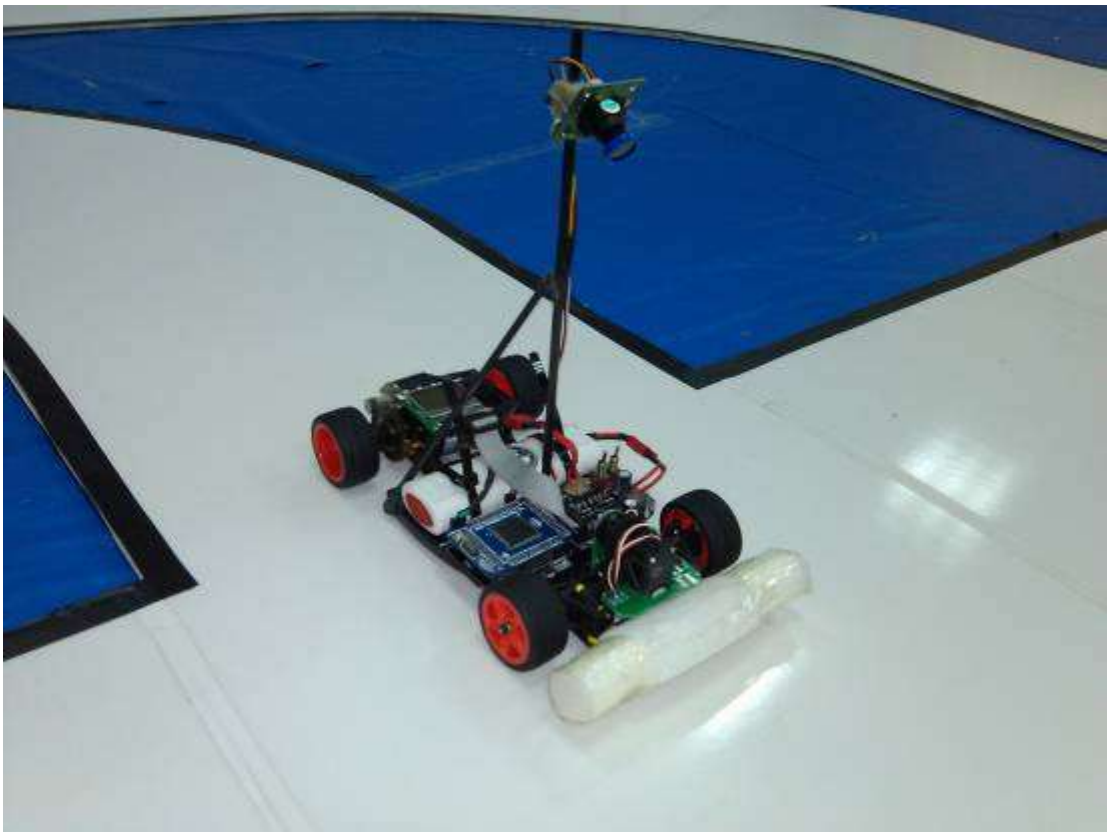


图 2.17　整车效果图

2.2 硬件电路的设计与调试

智能车系统在硬件电路上主要包括电机内部电路，舵机内部电路，电源模块，主控制器模块，摄像头及其采集模块，电机驱动模块，测速模块，辅助调试模块等，其中大赛组委会规定严禁改动舵机，电机的内部结构，因此发挥空间主要集中在其他几个模块。

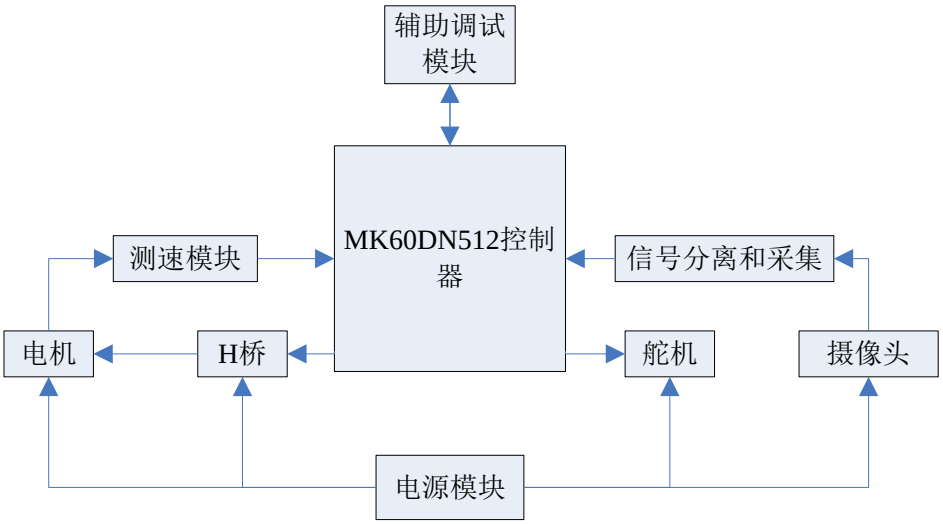


图 2.18：硬件电路结构图

2.2.1 Freescale MK60DN512 单片机

智能汽车大赛由美国飞思卡尔半导体公司（前摩托罗拉半导体部）赞助，“必须采用飞思卡尔半导体公司的 8 位、16 位、32 位处理器(单核)作为唯一的微控制器”，MK60DN512 是 Kinect 系列单片机中的一款增强型 32 位单片机，片内资源丰富，接口模块包括 SPI、SCI、IIC、A/D、FTM 等，在汽车电子应用领域具有广泛的用途。

Kinetis 系列微控制器是飞思卡尔公司于 2010 年下半年推出的，是业内首款基于 ARM Cortex-M4 内核的微控制器。

目前，Kinetis 系列的微控制器有 K10、K20、K30、K40、K50、K60 和 K70 等几个子系列。其中 K50 子系列是专用于医疗器械方面的专用系列，具有独特的应用特性。

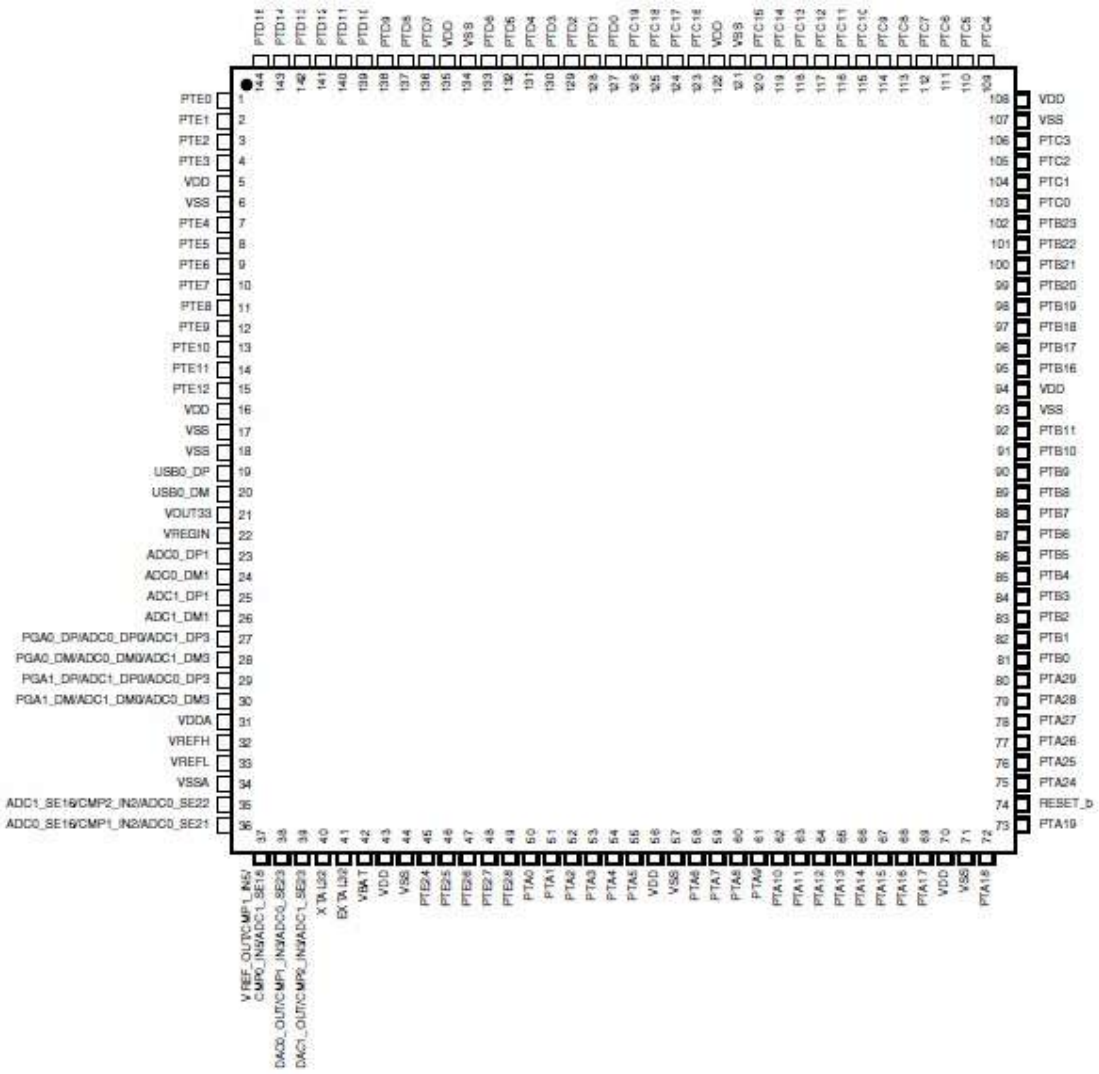


Figure 29. K60 144 LQFP Pinout Diagram

图 2.19：主控单片机引脚图

对于本小车系统，144 脚已经足够满足需要，但为了提高单片机处理速度，缩短控制周期，本小车系统除了从软件上下功夫，降低算法的复杂度，提高代码的执行效率外，也将单片机的时钟周期提高到了 180M。

2.2.2 电机驱动模块

模型车后轮驱动电机的型号为 RS-380 电机，工作在 7.2V 电压下，空载电流为 0.5A，转速为 16200r/min。在工作电流为 3.3A 时，转速达到 14060r/min，工作效率最大。

由于比赛中，模型车不需要进行倒车，因此设计的电机驱动模块只需要实现模型车加速前行和制动即可，所以电机只工作在正转方向的做功和发电两个状态。

以前模拟量驱动器使用比较广泛，现在由于 PWM 技术的不断发展和日趋完善，具有调速范围宽，效率高，易于数字控制，应用范围广等特点，脉宽调制控制方法（PWM 和 SPWM）、变频技术在直流调速和交流调速中获得广泛的使用。而 MK60DN512 单片机上就有集成的 PWM 模块，因此可以使用大功率晶体管，全桥或者半桥电路，输出 PWM 波形实现对电机的控制。在上一届的比赛中了，为了简化驱动电路的设计，多数的队伍采用了集成电机驱动芯片 MC33886 来完成对电机的控制，尽管内置短路保护、欠压保护、过温保护等功能[1] ，性能优越，但经过实际的试验发现，33886 在使用过程种发热明显，两片并联加装上散热片之后，发热有所改善，但重量较大，不符合小车制作简洁轻便的原则。权衡比较之后，在小车系统上采用自制的 H 桥的电路。

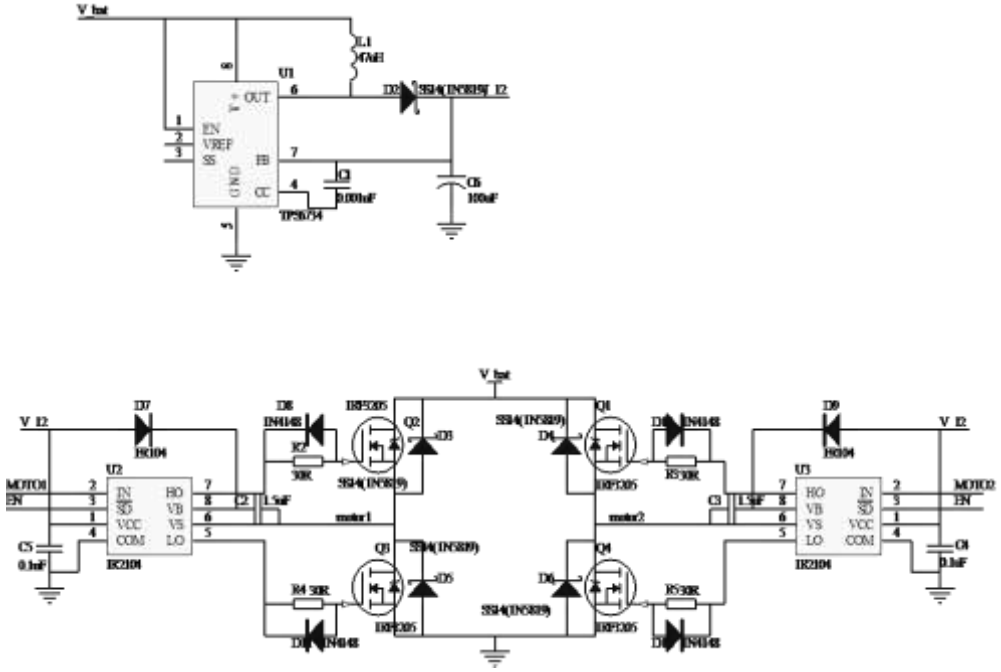


图 2.20：电机驱动模块

2.2.3 车速检测模块

根据前几届比赛队伍的情况，车速检测器件大致有测速发电机，旋转编码器，对射式光电开关，霍尔元件等。综合重量、价格、测量精度和安装方面各方面因素，微型旋转编码器比较适合小车系统。具体实现如图所示：利用车模后轴的齿轮将转速传动至编码器，使编码器输出一定频率的方波信号，在速度不太慢的情况下输出与实际速度的线性关系较好。设定单片机的 FTM 模块为脉冲累加功能，检测信号的上升沿，就能采集这些脉冲，便得到得到当前的车速。

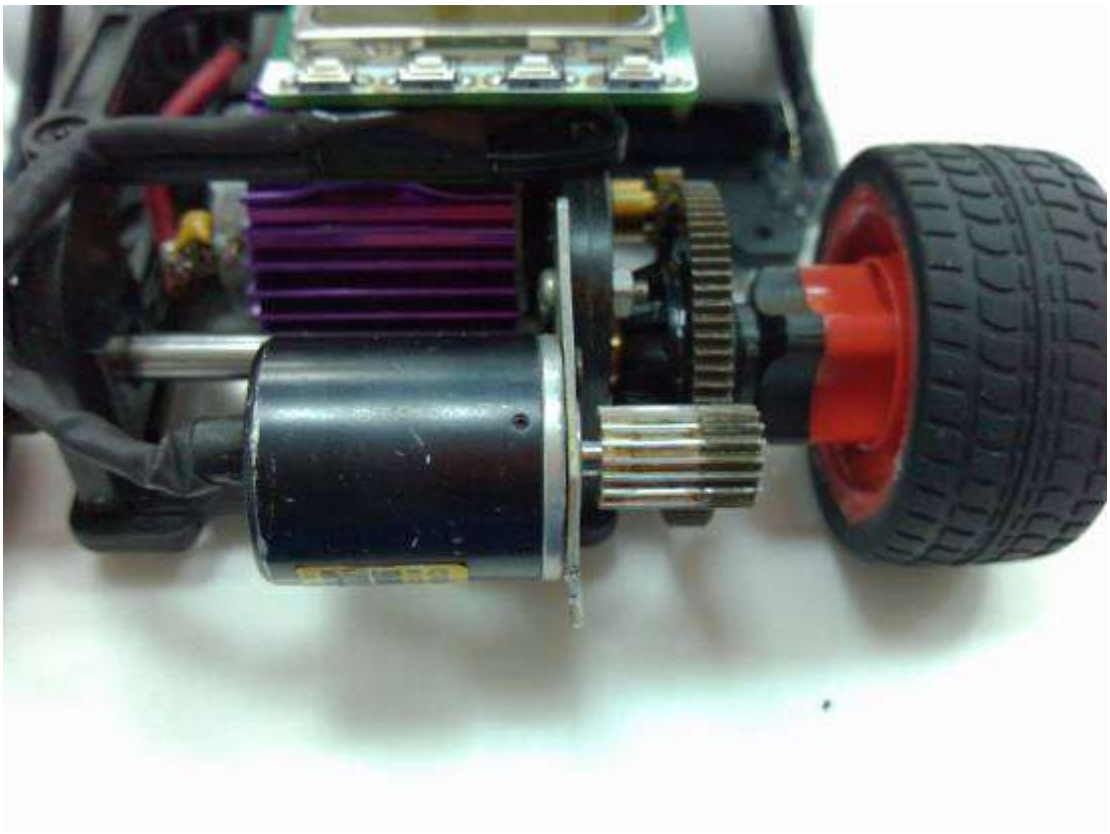


图 2.21：车速检测模块

2.2.4 电池供电模块

电源模块为小车系统的其他各模块提供所需要的电源。设计中，除了需要考虑电压的范围和电流容量等基本参数外，还要在电源转换效率，降低噪声，防止干扰和电路简洁方面进行优化。可靠的电源方案是整个硬件电路稳定可靠运行的基础。

由于带动整个小车系统工作的能量均来自一块由 6 颗 1.2V 镍镉充电电池串联而成的 7.2V 电池，容量为 2000mAh。而系统中各模块所需的工作电压和工作电流各不相同，因此电源模块应该包括多个稳压电路，将电池电压转换为各个模块所需的电压，具体的电路结构如图所示

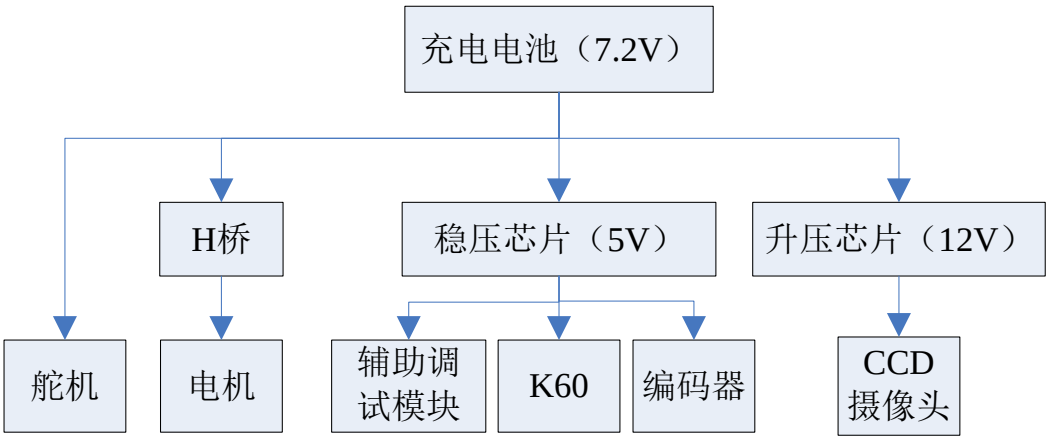


图 2.22：电源供电模块结构图

目前市场上的电源稳压芯片很多，最常用的是线性稳压器,，如78xx系列的三段稳压器件，虽然这种线性稳压器具有输出电压成可调、稳压精度高的优点，但是由于其线性调整工作方式在工作中会产生较大的“热损失”，导致电源利用率不高，不易达到低功耗的要求。而低压差的稳压芯片TPS7350则避开了这些缺陷。TPS7350具有输入电压范围大，过热、过流及电压反接保护等特点，特别是当其输出电流为100mA时压差仅仅为0．035V，只要输入电压在+7V～+5，1V范围内变化，TPS7350能可靠的保证输出电压稳定在5V，从而显著的提高了电源的利用效率。

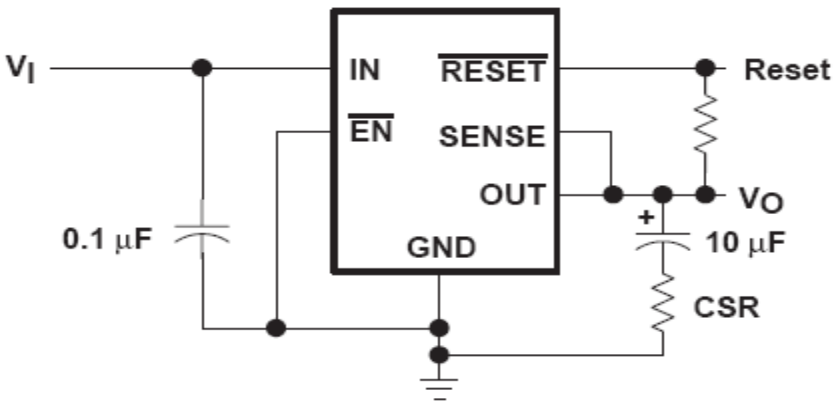


图 2.23：TPS7350 典型电路图

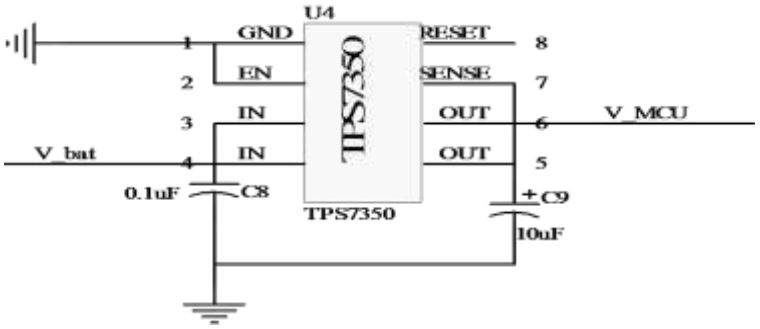


图 2.24：最终设计电路

2.2.5 图像采集模块

智能小车赛道识别方案一般有两种：一、基于红外光电传感器；二、基于摄像头。两者各有所长 ，本小车系统采用摄像头设别方案。

一般说来，单片机采集图像传感器的数据有两种方法，模拟式和数字式，数字式的图像采集则是利用 CMOS 图像传感器的数字输出特性，直接读取传感器的数字输出。该方式性能稳定，不需要 A/D，图像采集工作在单片机就变成了按照一定顺序将外部数据并行读入，因此程序简单，采集速度快。但是数字式摄像头只有 CMOS 型，没有找到 CCD 型的，而且使用起来外围电路和软件部分的寄存器设置较为繁琐，故我们选用模拟式的具有较高成像质量的 CCD 作为图像采集模块。

2.2.5.1 CCD 视频信号。

摄像头主要由镜头，图像传感芯片和外围电路构成。图像传感芯片又是其最重要的部分，摄像头的指标（如黑白或彩色，分辨率）就取决于图像传感芯片的指标，该芯片要配以合适的外围电路才能工作，将它们制作在一块电路板上，称为“单板”，若给单板配上镜头、外壳、引线和接头，这就构成了通常所见的摄像头。

摄像头通常引出三个端子，一个为电源端，一个为地端，另一个就为视频信号端（有的摄像头多出一个端子，那是音频信号端）。电源接多大要视具体的单板而定，目前而言，一般有两种规格，5-9V，或 9-12V，本小车系统采用 5~9V 的规格。摄像头的视频信号电压一般位于 0.5V-2V 之间。

按一定的分辨率，以隔行扫描的方式采样图像上的点，当扫描到某点时，就通过图像传感芯片将该点处图像的灰度转换成与灰度成一一对应关系的电压值，然后将此电压值通过视频信号端输出。具体而言（参见图 6.1），摄像头连续地扫描图像上的一行，就输出

一段连续的电压视频信号，该电压信号的高低起伏正反映了该行图像的灰度变化情况。当扫描完一行，视频信号端就输出一低于最低视频信号电压的电平（如 0.3V），并保持一段时间。这样相当于，紧接着每行图像对应的电压信号之后会有一个电压“凹槽”，此“凹槽”叫做行同步脉冲，它是扫描换行的标志。然后，跳过一行后（因为摄像头是隔行扫描的方式），开始扫描新的一行，如此下去，直到扫描完该场的视频信号，接着就会出现一段场消隐区。此区中有若干个复合消隐脉冲（简称消隐脉冲），在这些消隐脉冲中，有个脉冲，它远宽于（即持续时间长于）其他的消隐脉冲，该消隐脉冲又称为场同步脉冲，它是扫描换场的标志。场同步脉冲标志着新的一场的到来，不过，场消隐区恰好跨在上一场的结尾部分和下一场的开始部分，得等场消隐区过去，下一场的视频信号才真正到来。PAL 制式的摄像头每秒扫描 25 幅图像，每幅又分奇、偶两场，先奇场后偶场，故每秒扫描 50 场图像。每场图像有 312.5 行，行周期为 64us.奇场时只扫描图像中的奇数行，偶场时则只扫描偶数行。具体的时序图如下：

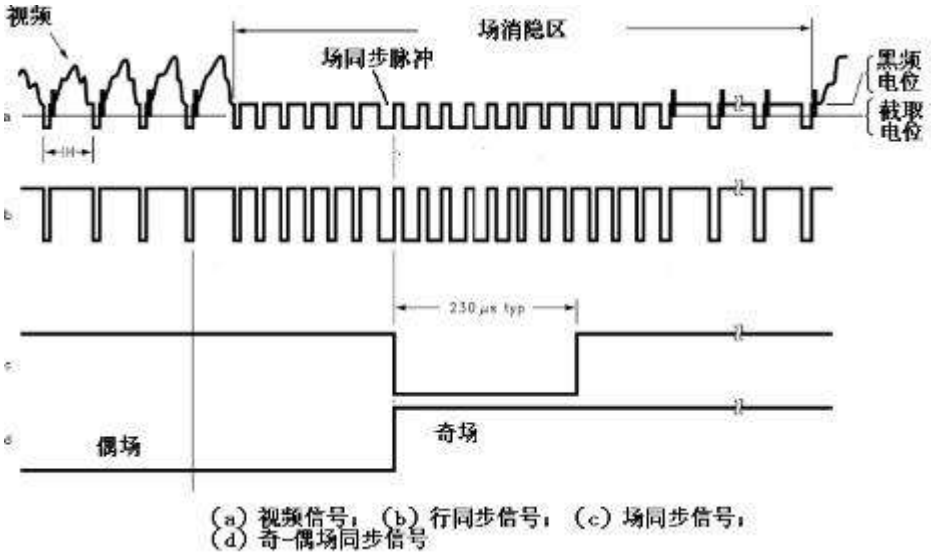


图 2.25：摄像头信号时序图

2.2.5.3 视频信号分离

视频信号每场是是不同的行组成，如图 1 所示，场与场之间，行与行之间都存在同步信号，单片机通过对这些同步信号的捕捉，来控制图像采集的时序，保证图像采集的正确性。视频信号分离芯片 LM1881 能将视频信号中的行同步脉冲、消隐脉冲和场同步脉冲提取出来，并将它们转换成数字信号交给单片机的 I/O 口。视频信号分离电路如图 2.26 所示。

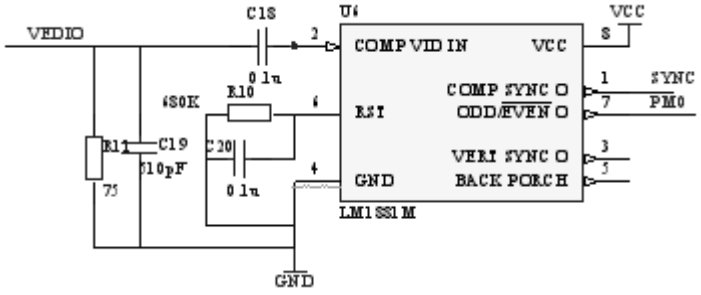


图 2.26：视频信号分离电路

2.2.5.4 视频信号数字化。

由于模拟摄像头采集的图像信号为模拟信号，而计算机系统为离散的数字系统，故需要转换为数字信号才能交付 MCU 处理。大赛组委会所推荐的芯片 K60 系列片内集成了 8 路 AD 口，同时还具有丰富的 IO 接口资源，而且大赛中小车赛道色彩构成的简单性，使得视频信号数字化方案变得多样化。小车以模拟 CCD 摄像头为传感器，模拟式的采集先将一路视频信号引用图 2 电路，通过其可以将摄像头输出的复合视频信号进行分离，得到独立的同步信号和视频模拟量信号，然后通过逐行采样来完成整幅图像的采集。下面主要介绍三种对行信号的数字化的方法。

1). 片内 A/D 转换+DMA 读取

由于 MK60DN512 本身就含有 24 路 AD，故只需引入一路视频信号至任意一个 AD 口，然后软件上，先对 AD 口进行相应的初始化，再在行同步中断函数中开启 DMA 中断即可完成数字化：

此方法的优点在于，不需要进行额外的外围电路设计，直接引入视频信号，利用 MK60DN512 的片内资源进行 AD 转换即可得到图片灰度数据。K60 在超频到 60M 的情况下，能每行采集 190 个点，如果对图片横向精度要求不高，选择此方法最为简单。但如果 CCD 看得比较远，由于图像的几何畸变，会造成远方的黑线最后 AD 结果只有一个黑点，这样在黑线提取时造成了较高的误判率，此时此方法就不再适用。

2). 片外 A/D 转换

为了每行信号得到更高的采样精度，在 K60 还有多余的 IO 口前提下，我们可以考虑用片外 A/D 法。TCL5510 为一款 8 位并行高速 AD 转换芯片，如果采用独立时钟其 AD 速度能达到 20MSPS，然后将数字信号通过 8 位数据总线并行输出，直接引入到 K60 的一组 IO 口上，在软件设计上只需对此 IO 口进行读取即可获得数字信号。相应 AD 转换电路如图 2.27。

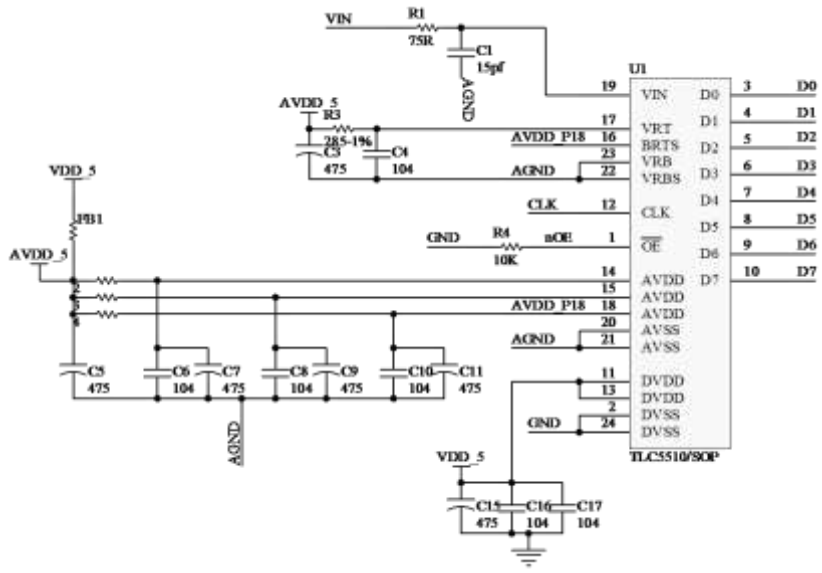


图 2.27：片外 AD 转换电路

芯片对端口进行读取时，为了防止读取到 AD 转换的跳变沿，故需要进行时钟同步，K60 含有 3 路 PWM 输出，通过对一路 PWM 进行翻转输出一定频率的脉冲，即可模拟一个同步时钟信号至片外 AD 模块，较方便地解决了时钟同步问题。在实际情况下，K60 超频到 180M 时每行能采集 240 个点，从而远方黑线消失的现象能得到很好的解决。

除了采样精度高，更重要的是此方法保存的是灰度信息，能极大程度上防止了图像信息丢失，但是外围电路比较复杂，而且占用了较多的 IO 口资源（需要 K60 额外提供一组 8 位的 IO 口进行数字信号读取）。

3). 电压比较器转换

以上两种方法最后得到的都是图像的灰度数据，能够比较逼真地反应 CCD 所见情景。但是由于大赛中，赛道仅由黑白两色组成（如图 2.28），所以即使是灰度数据，我们最后处理时也一般要在软件上进行二值化将图像分割成黑白

二色图片。所以，我们可以考虑直接用硬件进行图像二值化，将视频信号转换为一组方波信号，然后直接输入到一位 IO 口中，对这一位的端口进行读取，高电平表示 1，低电平表示 0。在软件上支灰度图片进行黑白分割时，比较常用的方法是固定阈值法，即高于此阈值电压，即认为是 1，否则是 0，然后再通过软件进行黑线边缘检测。

图 2.29.固定参考电压二值电路

采用固定参考电压的二值电路设计起来比较简单，对参考阈值电压调结也比较方便－只需调结一个电位器阻值即可（为系统增加一个 LCD，可直接在调结电位器后采集到的图像，或者利用串口成像软件观测采集到的二值图像），故具有一定的场地适应性。

采用硬件二值检测电路，通过对两个电位器大小的调整，便能适应不同比赛场地光线，而且能适应 CCD 的不同视野，解决了固定电压比较器远视野无法分割的缺点，具有更强的场地适应性。而对比片外 AD，此方案具有以下优点：

1. 电路设计更简单。

2. 占用芯片 IO 口只有一位，为片外 AD 的 1/8。

3. 不用考虑时钟同步的问题。

4. 横向精度可以视为无限，在满足要求的情况下，MCU 不用超频甚至还需要分频，8M 频率读端口便能每行采集 120 多个点。具体采集多少点依用户需求而定。

5. 采用硬件二值，减少了 CPU 的运算量。

显然，对于色彩简单的赛道环境，此方案拥有极大的优势，但是由于此方案直接给出的是 0，1 二值数据，导致在图像的软件处理上便没有了处理灰度数据的灵活性了。故此方案也成为了我们图像采集的最终方案。

2.2.6 辅助调试模块

在小车的调试过程，经常需要查看某些变量的值，通过串口线与上位机之间进行通讯显然不现实，于是给小车设计了无线串口模块，使小车在行驶过程种将某些重要变量通过串口发送到上位机进行分析处理。

STR-36 型微功率无线通信模块，是一款全 ISM 信道,高性价比，微功率无线模块，发射频率在-20 到 10dBm 之间用户可调,接收灵敏度达-115dBm,通信速率最高可达 500kbit/s，是无线燃气表，无线水表和无线电表等其他 RF 无线通信设备的理想选择。STR-36 采用无铅环保生产工艺，完全符合欧美产品环保出口标准，十分适合小车的调试。

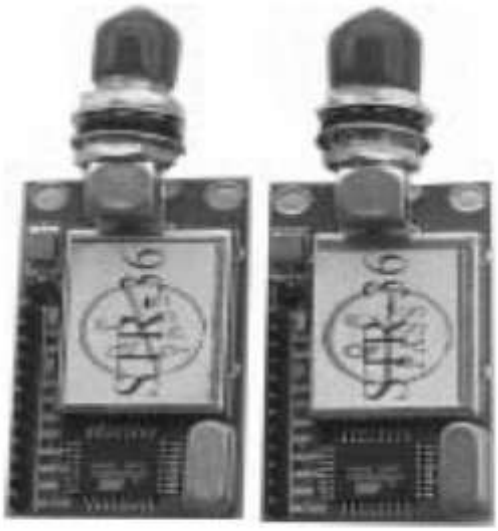
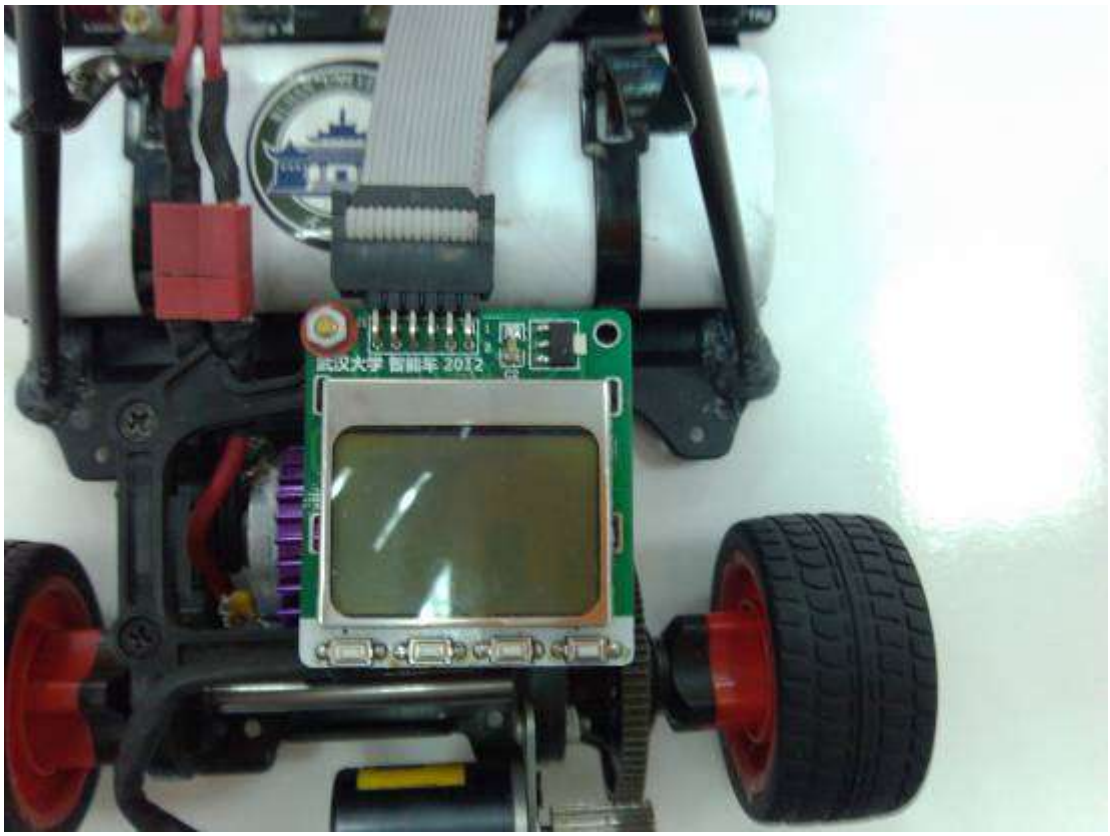


图 2.27 ：无线串口


武汉大学 智能车 2012

第三章：智能车系统软件的设计与调试

智能车系统的软件系统包括赛道信息提取，赛道参数的分析与计算，舵机转向控制，电机速度控制以及辅助调试等模块，由于小车系统控制具有单任务，周期性的特点，采用传统的程序流程控制方法，将图像采集到小车控制的任务放在一个死循环中顺序执行。

3.1 路径识别

在硬件系统稳定可靠的情况下，小车得以稳定，快速的运行，并以较为优化的路径通过赛道，前提是必须精确快速的提取出黑线信息，并准确滤出干扰信号，为了舵机和电机的提供良好的控制信息。

3.1.1 目标黑线的提取

由于小车运行在底色为白色的环境下，只是在跑道两边有宽度为 25mm 的黑线指引。于是可以采用相对简单的图像分割方式，找到一个合适的分割阈值，将图像二值化处理，当某点的像素值高于该阈值时，为白色点，反之则为黑点，然后找出这些黑点的中心，即为目标黑线所在的位置，此种方法称为中心点提取算法。但考虑到比赛场地外部环境的不确定性，为了提高抗干扰能力，可以采用更为稳定的边缘提取算法，其基本原理是利用黑白交界处像素值将会出现较大跳变，利用此跳变找出黑线的左右边缘。具体实现如图 3.2：

为了方便后面的叙述，先明确以下变量的含义 ：

表 3.1

row：摄像头扫描一幅图像上各点的行坐标 *column*：摄像头扫描一幅图像上各点的纵坐标 *fbegin*：左侧黑线开始的行 *fend*：左侧黑线结束的行 *fbeign*2：右侧黑线开始的行 *fend*2：右侧黑线结束的行 *left*：某行左侧黑线的位置 *right*：某行右侧黑线的位置 *data*[*row*][*line*]：一幅图像各点的像素值数组 *diff*[*row*]：各行黑线的中心位置 *diff* 2[*row*]：各行黑线中心偏差	各变量含义

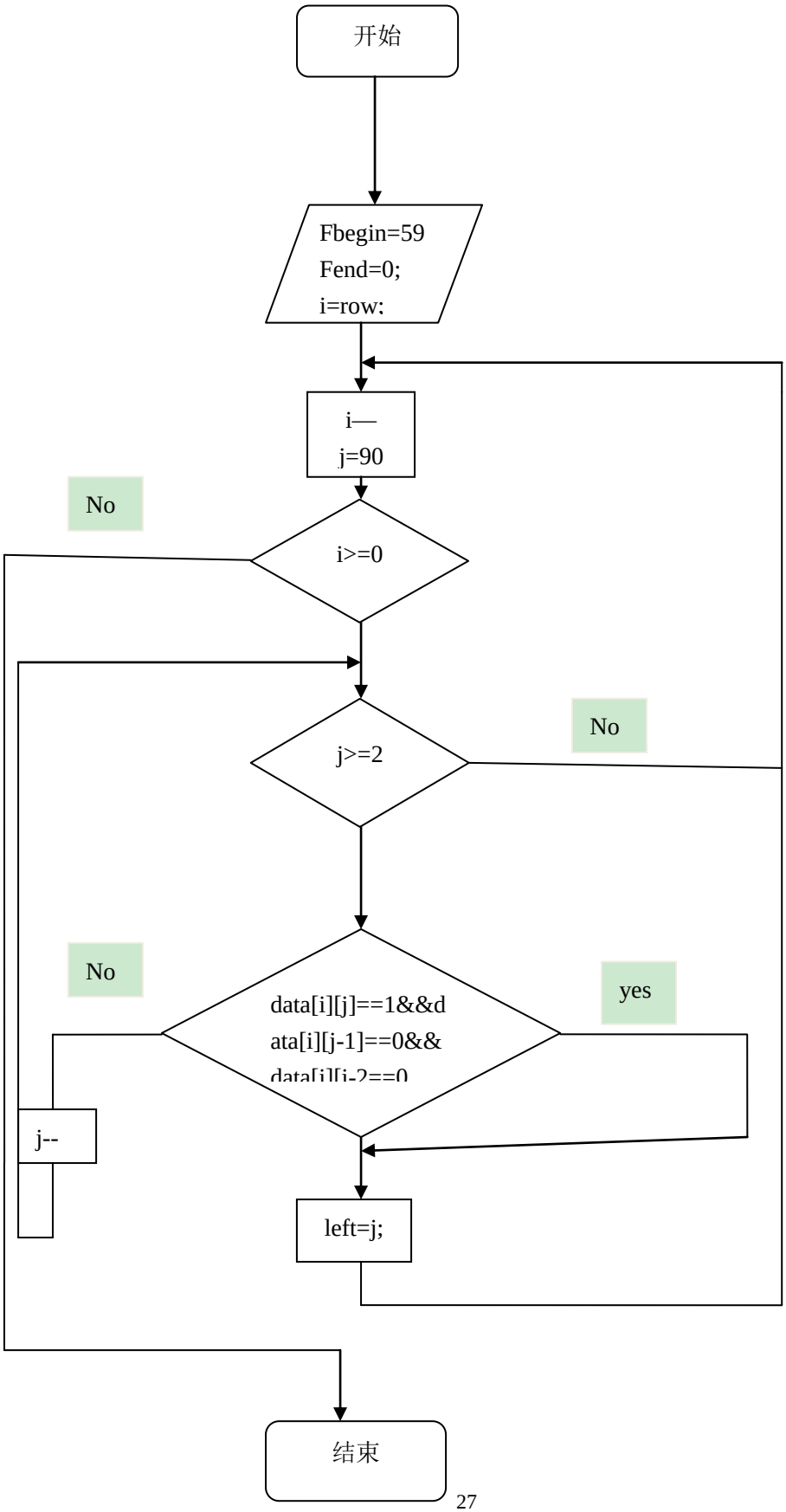
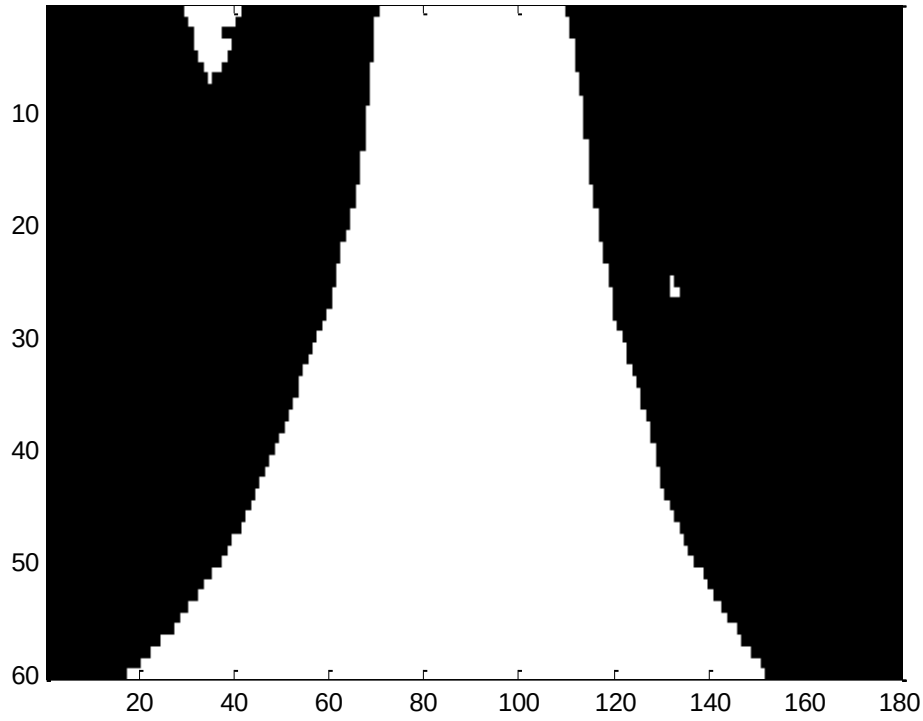

开始
Fbegin=59
Fend=0;
i=row;
i—
j=90
No
i>=0
No
j>=2
No
yes
data[i][j]==1&&d
ata[i][j-1]==0&&
j--
left=j;
结束

图 3.2：左侧目标黑线提取程序流程图

右侧黑线提取的方法相同，只是 j 从 90 到 179。当找到每行的左侧和右侧的黑线的位置后， *diff*[i]=(left+right)/2;因此对于最左最右就出现黑点时，采用中心提取算法，其他情况采用边缘提取算法。图 3.3 为原始采集到的图像，图 3.4 为最终提取效果图，效果良好。



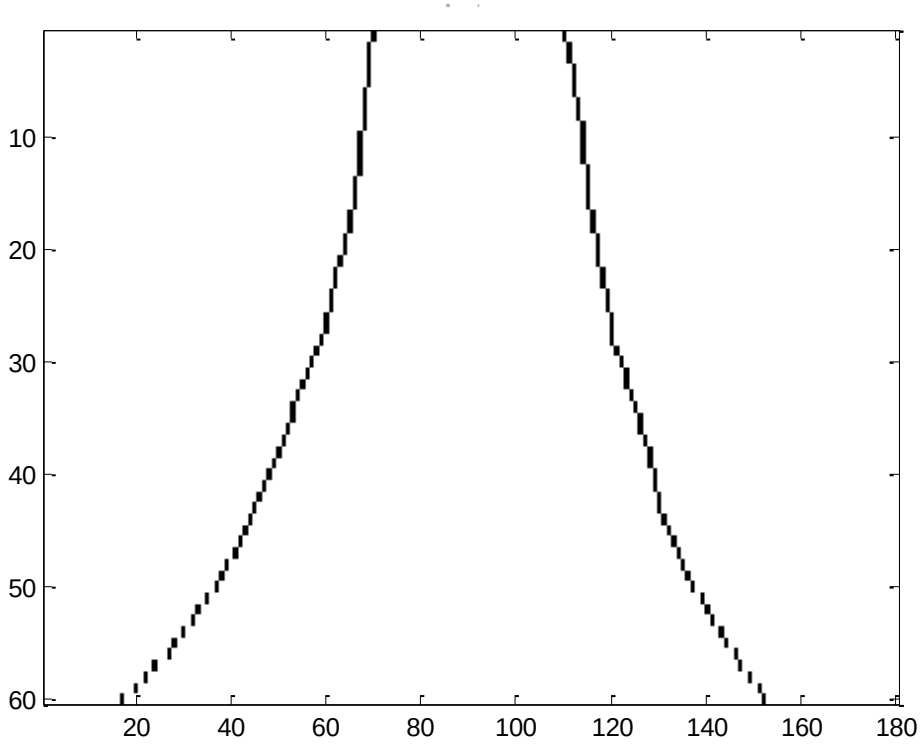


图 3.3：提取效果图，黑色部分为找到的边缘

3.1.2 抗干扰处理

考虑比赛场地的不确定，为了防止比赛场地出现一定黑色干扰，导致控制出错，因此有必要对整幅图像进行抗干扰处理滤波，考虑到黑色指引线是一条平滑连续的曲线，因此对于图像中出现的那些不在黑线趋势上的点即可视为干扰，加以去除。

目前软件抗干扰滤波方法很多，多数需要进行卷积运算，单片机开销较大，于是本小车系统采用对整幅图像进行较为简单的插值处理。由于摄像头提取到的近处黑线信息出错的概率非常小，因此利用近处黑线的中心偏差，对前方的黑线中心偏差进行插值预测，将插值所得到的黑线中心偏差与提取出来的黑线中心偏差进行比较，若相差较小，则保留原来的黑线中线偏差值，若相差较大，则用插值所到的值将其替换。同时考虑到小车在过急弯时摄像头图像会出现如图3.4的情况，因此有必要对插值后的黑线中心进行限幅处理，以防止插值后的黑线中心偏差偏出视野范围太多。

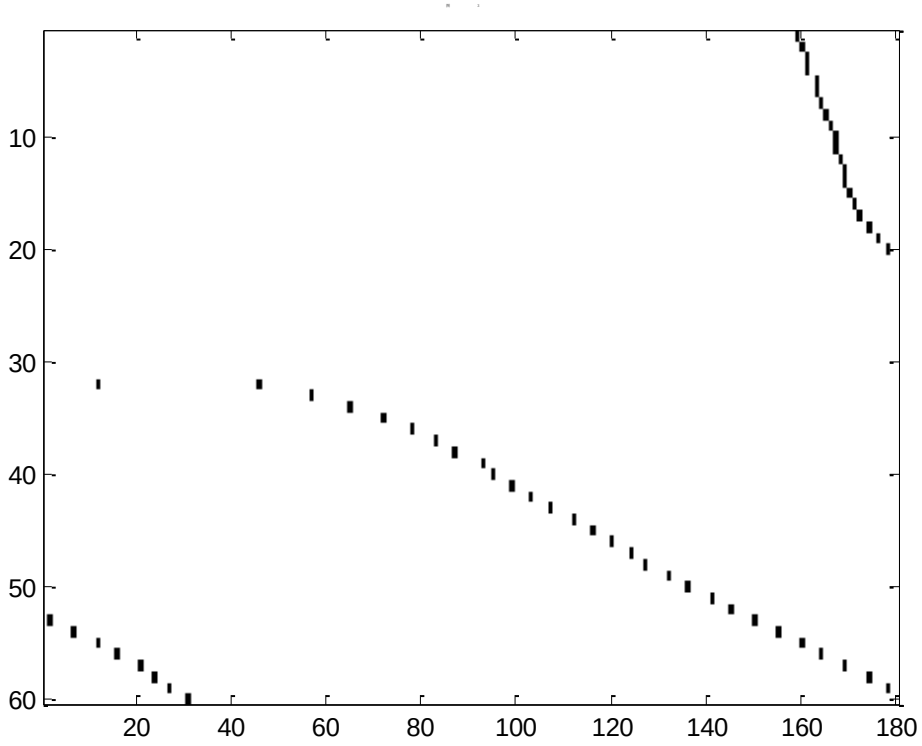


图 3.4：过急弯时出现的图像

插值在数学上方法很多，常用的有牛顿差值，拉格朗日插值等，但对于智能小车这个控制不要求十分精确的系统，二次插值足以达到要求，具体实现如图 3.5 所示：

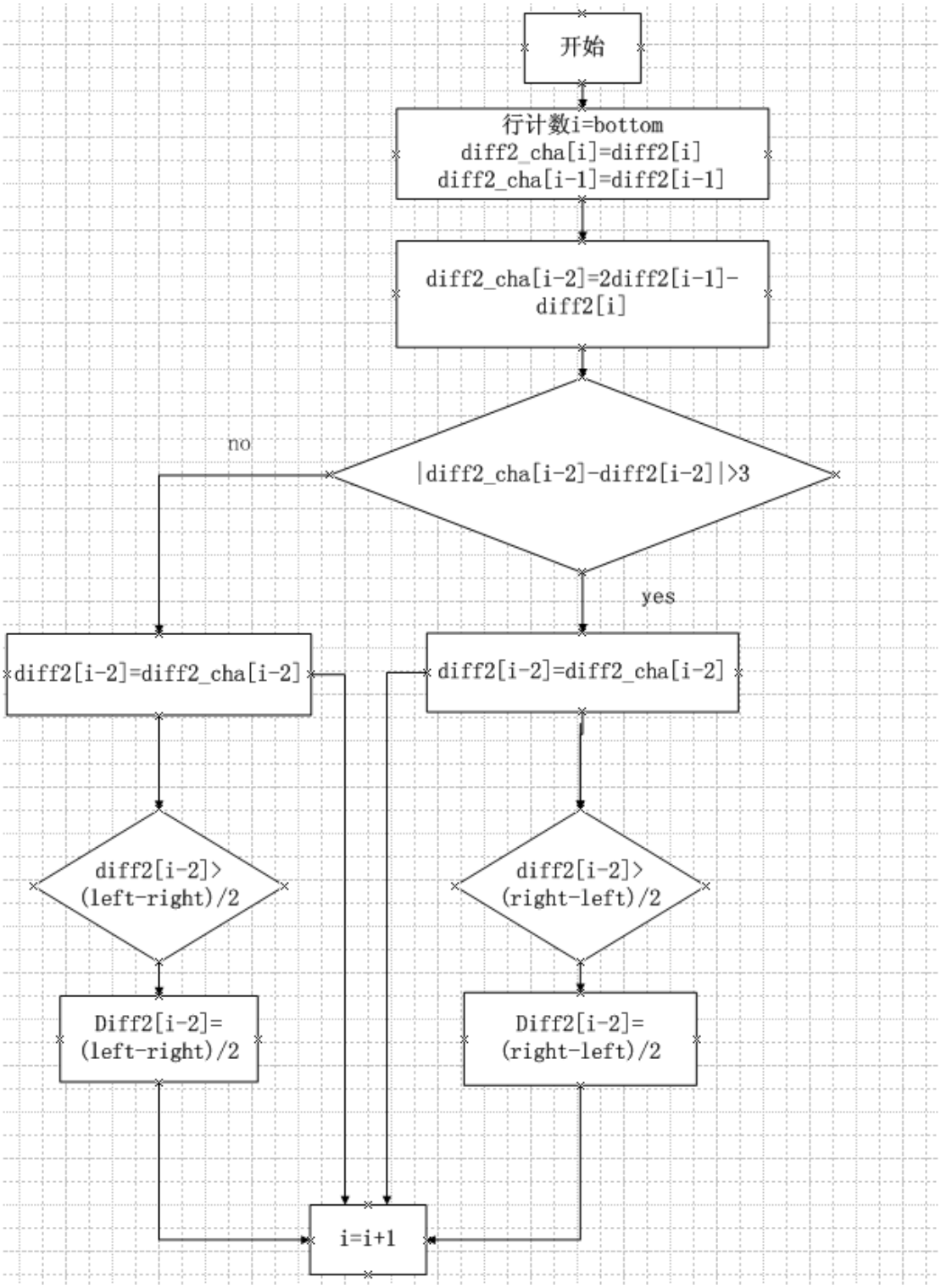


图 3.5：插值滤波程序流程图

这样的插值有两个好处，第一：有效的去除了随机干扰；第二：对于那些
如图 3.4 不完整的图像，黑线的中心全部根据黑线的走向趋势补齐在摄像头视野的两侧，走向为向右的补齐在最右侧，走向为向左的补齐在最左侧，这在某种程度上也是一种赛道的趋势预测。

3.1.3 赛道趋势预测

对于摄像头采集到整幅图像，如果仅仅提取到每行的黑线中心偏差，显然对于图像信息的利用率是十分低下的，对小车的控制来说也是不够的。由于在本小车的控制中，控制量主要以图像前方某行黑线中心偏差为主，但一行的偏差，并不能反应当前道路走向趋势，因此必须对整幅图形进行分析，提取出某些重要的参数，利用这些参数与单行偏差识别出赛道类型，给舵机一个最优的控制，使小车既能安全稳定的行驶，不会冲出赛道，同时又以一个较为优化的路径通过赛道。

在智能汽车的比赛中，赛道类型主要分为：大 S 弯，小 S 弯（蛇形弯），直道。整体上，对于小车的控制目标是：对于直道和小 S 弯，力求小车以较小的舵机控制量，同时以较高的速度通过，即直道舵机不抖，小 S 直冲。对于大 S 则要以最高的安全速度切内道驶过。

3.1.3.1 赛道线性拟合

对于一条曲线，反应一条曲线缓急程度的几何参数是曲线的曲率半径，然而摄像头采集到的一幅图像不是一条连续的曲线，只是赛道上一些离散的点，尽管任然可以求出各点的曲率半径，但求曲率不可避免的要求二次导数，算法十分复杂。因此采用对这些离散的点进行线性拟合的方法，用拟合所得的斜率和残差绝对值之和来区分赛道类型不仅简单而且效果明显。

为了的叙述讨论方面，以行数为 X 轴，黑线中心偏差为 Y 轴，图像最远出赛道中心为原点建立如下坐标系：

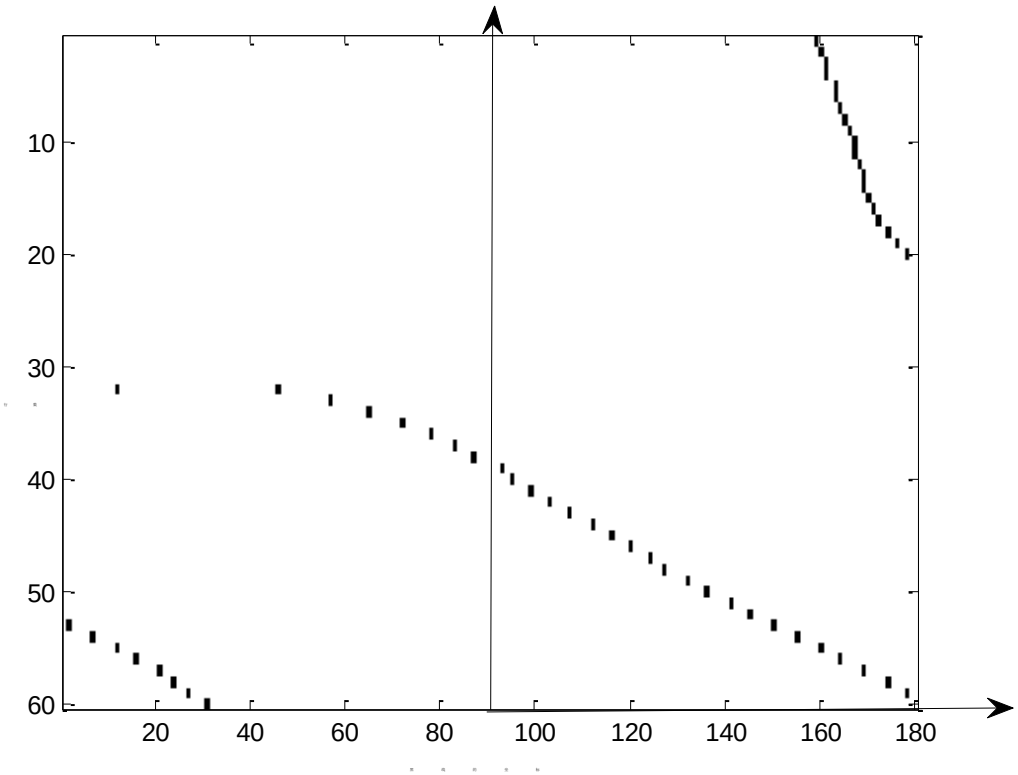


图 3.6：赛道坐标系

采用最小二乘法进行拟合，设拟合直线的方程为：

$$diff2 = a \times line + b$$

残差的平方和表达式为：

$$e^2 = \sum_{line=top}^{bottom} (aline + b - diff2[line])^2$$

拟合的各点为：$(fend, diff2[fend]), ..., (fbegin, diff2[fbegin])$

欲使得 $e^2 = \sum_{line=top}^{bottom} (aline + b - diff2[line])^2$ 最小

a, b 应该满足：

$$\frac{\partial_{e^2}}{\partial_a} = \sum_{line=top}^{bottom} (aline + b - diff2[line]) \times line = 0$$

$$\frac{\partial_{e^2}}{\partial_b} = \sum_{line=top}^{bottom} (aline + b - diff2[line]) = 0$$

解得：

$$a=\frac{N\sum_{line=top}^{bottom}(line\times diff2[line])-\sum_{line=top}^{bottom}line\times\sum_{line=top}^{bottom}diff2[line]}{N\sum_{line=top}^{bottom}line^2-(\sum_{line=top}^{bottom}line)^2}$$ $$b=\frac{\sum_{line=top}^{bottom}diff2[line]-a\sum_{line=top}^{bottom}line}{N}$$ $$e^2{}_{\min}=\sum_{line=top}^{bottom}(aline+b-diff2[line])^2$$ 其中$N=fbegin-fend+1$	公式（五）

其中 a 为赛道的斜率，$e^2{}_{\min}$ 为拟合后残差的平方和，根据两者可以有效的区分出赛道类型,，具体如下：

表 3.2

赛道类型	斜率	残差平方和
直道	小	小
大弯	大	大
小 S 弯	小	大

然而以上以上判据只有在小车运行得十分正的时候才能有效的将赛道有效的区分开，然而实际操作中发现，如果跑道在大 S 弯之后紧接着出现小 S 弯，则小 S 在图像中出现的形态是一个斜着的小 S，此时算得的小 S 的斜率和残差平方和也同样会很大，从而与大弯混淆。同时比赛中经常出现一个曲率半径较大的 3/4 圆的赛道，对于此种赛道，小车应该持续以一个较小的舵机控制量，以较高的速度快速通过，为了把小 S 和这种曲率半径较大的圆圈赛道区分开来，对整幅图像需要做进一步的分析计算。

3.1.3.2 小 S 弯判断

对于整幅图像，黑线的中心偏差定会出现一个最大值最小值，但对于直道和大弯，最大值最小值都出现在图像的顶部或是底部，而对于小 S，最大值或最小值一般会出现在图像的中部，也就是说小 S 弯在图像中会出现一个极值点，即存在一个导数为 0 的点，然而求导数精确度不高，于是我们避开求导，采用下面的方法：首先从图像的最顶部到最底

部寻找黑线中心偏差的最大值，最小值，如果最大值或是最小值出现在中部的某个范围内，则认为这是小 S 弯。　具体实现流程图如下：

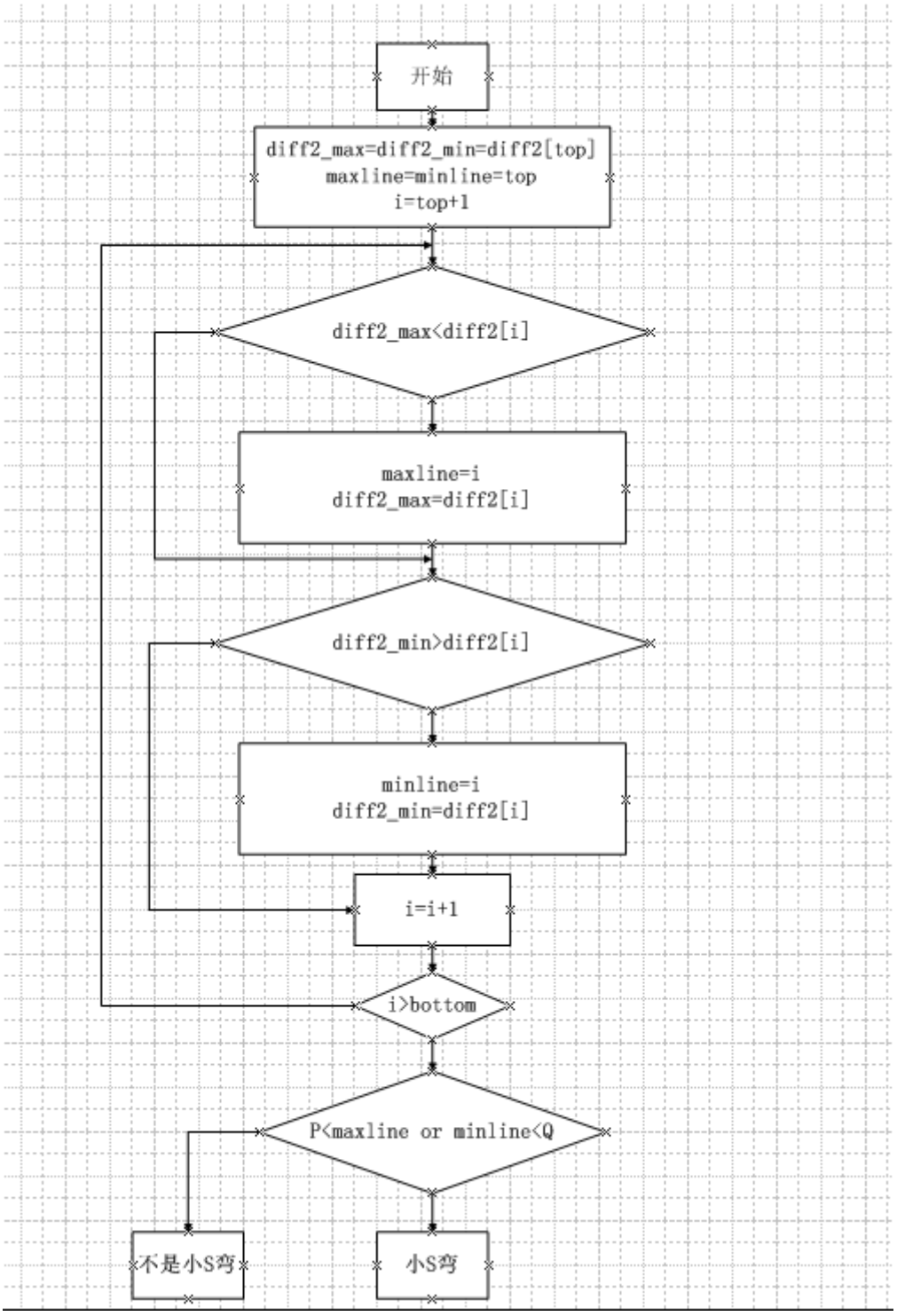


图 3.7

3.1.3.3 大圈判断

3.2 舵机控制

对舵机的前轮控制，采用以较为简单直接的位置式 PID 进行控制为主，同时辅以坏帧图像和小 S 弯道的控制策略。力求舵机控制使前轮转向快速准确。

3.2.1 数字 PID 舵机控制

在提取到图像信息较为完整的情况下，以图像前方某行的黑线中心偏差和整幅图像的拟合直线斜率作为控制量，总的算式为：

$PID\_diff2 = K_P diff2[row](n) + I\_enK_i \sum_{I=1}^{N} diff2[row](n) +$ $K_d(diff2[row](n) - diff2[row](n-1))$ .....单行偏差控制部分 $PID\_a = K_{pa} a$ .....斜率控制部分 $PID\_out = \frac{M \times PID\_diff2 + N \times PID\_a}{M + N}$ .....两者加权平均 其中： K_p，K_i，K_d分别为单行偏差控制部分的P参数，I参数，D参数 I_en为D参数使能，$0/1$变量，为1启用I控制，为0则不启用 $diff2[row](i)$为第i个控制周期某行的黑线中心偏差 K_{pa}为斜率控制部分P参数	公式（六）

根据摄像头拍摄的视野范围，我们选取采集图像中第 5 行的偏差作为单行偏差控制部分的 $diff\ 2[row]$ 考，虑到模型车在高速运行的时候，舵机的延时环节可以等效成一个积分环节，该环节对道路中的蛇形线和路径检测中的干扰进行了平均，保证了模型车的稳定性，在偏差较大的情况下，使 $I\_en = 0$，不启用 D 控制，在偏差较小的情况下启用 D 控制。同时如果采用恒定的 P 参数对小车进行控制，在很容易出现直道上小车来回震荡、弯道转向不足的现象。为此有必要对 P 参数进行非线性化处理，即采用非线性 P 调节。

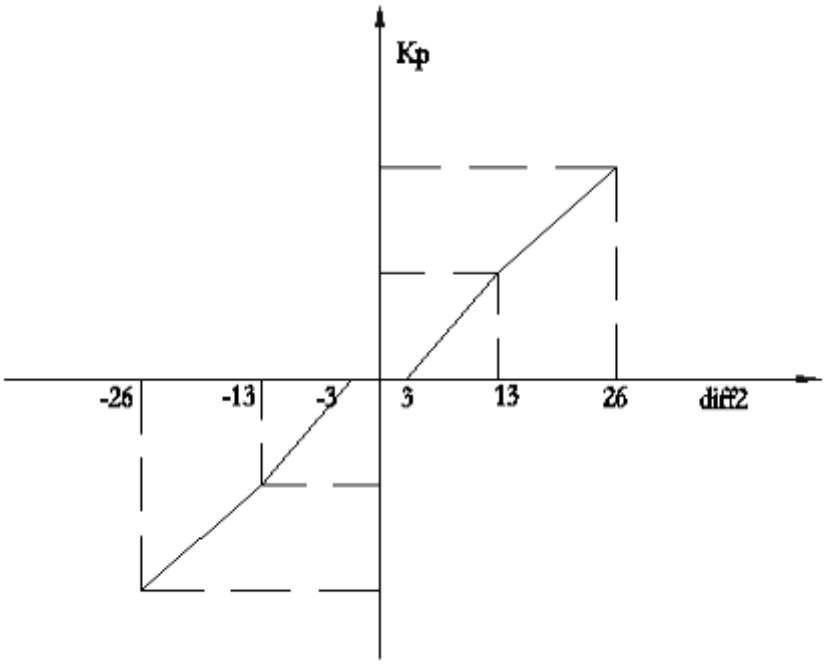


图 3.8：非线性 P 调节

如图所示，Kp 为比例项输出，偏差在-3～3 时为比例区间，输出为 0；当偏差绝对值在 3～13 时，输出一个较小的数值，对赛车产生的位移进行微调；当偏差绝对值大于 13 时，比例系数增大，P_out 输出较大值，使舵机快速动作，以纠正大偏差。这样即实现了系统在直道上时缓慢调节，遇到弯道时迅速动作。

3.2.2 错帧和小 S 控制

由于小车前瞻较大，同时摄像头的视野范围有效，这样在小车高速行驶到急弯时候，就会出现摄像头采集不到黑线的情况，这种图像情况称为错帧。对于错帧情况，除了从硬件上努力，提高摄像头的视野范围外，在软件上采取的策略为：当第一次出现错帧时，舵机保持上一周期的状态，连续出现错帧时候则逐渐在原来的基础上逐渐增大舵机控制量，当错帧连续出现 7 次时，将舵机达到极限位置，同时降低速度。

同时对于赛道出现的小 S 弯情况，首先根据 3.2.1 的控制方法算出舵机控制量，然后去这个控制量的 1/4 作为实际的舵机控制量。

即：

$PID\_out = \frac{swan\_flag \times PID\_out}{4} + (1 - swan\_flag) \times PID\_out$ 其中： $swan_flag$为小S弯判断标志,0/1变量,1表示小S弯,0则不是	公式（七）

这样即不会使转向失控，有保证了小 S 弯直冲。

3.3 速度控制

3.3.1 速度的闭环控制

小车得以快速稳定的运行，速度控制和舵机转向控制应该相互配合得很好，也就是说速度控制和转向控制应该是相互耦合的，然而这种耦合的数学模型十分复杂，实现起来十分复杂，因而在本小车的速度控制与舵机控制还是相互独立的，其基本思想是：对路径设别出来各种赛道类型，通过试验找出各自的安全通过的极限速度，将这些速度设定为设为预定值，用 PWM 波进行调速，

各种赛道摩擦力各不相同，加上电池变化等等因素的影响，如果单纯的用 PWM 波进行开环调速，系统响应时间较长，调速效果比较差。

在硬件的实际中，小车上安装了车速传感器，通过单片机 ECT 模块我们可以计算出赛车当前的车速，然后采用 PID 闭环控制，就及时快速地调节车速达到预定值。

同时为了提高电机的响应速度，依据 Bang-Bang 控制理论，当当前速度与速度预设值之间相差较大时，用电机满占空比或电机停转的方式调速，当当前速度与设定值之间相差不大时，则通过 PID 调速。即：

$Motor\_out = \begin{cases} 100\% & speed\_cnt - speed\_set < -S \\ P \times (speed\_cnt - speed\_set) & \lvert speed\_cnt - speed\_set \rvert \le S \\ 0 & speed\_cnt - speed\_set > S \end{cases}$	公式（八）

同时由于控制周期不能足够短，并且电机加速性能比较明显，这种调速方式使得速度大起大落，为了是速度调节变得平滑一些，当对一幅图像分析计算出速度设定值后，在后一幅图像的采集过程中，对速度进行两次控制，这样就使得小车在直道上加速迅猛，同时

在弯道上速度变化得稳定平滑。

3.3.2 刹车功能的实现

理想的赛车速度控制应当是直道上赛车以极限速度全速运行，在入弯的瞬间将速度降至通过该弯道安全车速，出弯后立即以极限速度运行。

为了尽量提高直道上的车速并保持入弯后车速足够安全，赛车必须能在最短的时间内刹车。

在现有的硬件上实现刹车有三种方法：

（1）使用机械结构刹车。在赛车左右后轮分别安装一个伺服电机，控制刹车片抱死后轮，以达到刹车的功能，和实际车辆的刹车原理相似。使用该方法需要增加两个伺服电机，如果左右刹车量不一致，赛车就会发生偏移，所以需要对左右轮车速进行分别测量，并结合赛车转向情况综合计算，系统较为复杂，实现困难。

（2）暂停驱动电路输出。本方式为常用的刹车功能，具体操作可以断开驱动电路或给电机 0 占空比实现，实际效果是使赛车向前滑行，依靠轮胎与地面的摩擦力减速。但由于刹车距离较长，高速运行的赛车需要相当长的一段滑行才能达到转弯的安全车速，所以效果并不理想。

（3）电机反转。该方式综合利用电机电磁制动和赛车后轮差速器结构进行刹车。高速运行的赛车刹车时，对电机施加一个反向电流，在电机内部磁场反向，使转子受到很大的磁力矩作用，迅速减速并反转。由于本模型车后轮装有差速器，允许左右后轮和传动齿轮分别以不同速度转动，当电机与赛车后轮转向相反时，他们之间摩擦力能使赛车很快减速。实际使用倒转功能进行刹车效果非常明显，但频繁对电机和驱动电路施加反向电流，两者的发热相当明显。

综合以上三种刹车方式，制定了快速减速的方案：在速度差较大时电机反转，速度差较小时赛车滑行，效果良好。

在现有的速度控制策略基础上加入刹车功能，我们将智能车速度控制策略设定为：

$$Motor\_out=\begin{cases}100\% & speed\_cnt-speed\_set<-S_1\\ P\times(speed\_cnt-speed\_set) & -S_1\le peed\_cnt-speed\_set\le S_2\\ -20\% & speed\_cnt-speed\_set>S_2\end{cases}$$	公式（九）

3.4 开发与调试手段

智能车的设计和开发是一个复杂的系统工程，为了提高了工作效率，在其调试与开发过程使用到了综合使用各种开发平台开发单片机和上位机程序，下面一一介绍。

3.4.1 IAR 与在线调试工具 BDM

IAR Embedded Workbench IDE 是由 IAR SYSTEMS 公司提供的面向 FreescaleK60 系列的 MCU 与 DSP 嵌入式应用开发的软件工具。其中包括集成开发环境 IDE、处理器专家、全芯片仿真、可视化参数显示工具、项目工程管理、C 交叉编译器、汇编器、链接器以及调试器。其中在本设计中重要的部分就是集成开发环境和调试器。

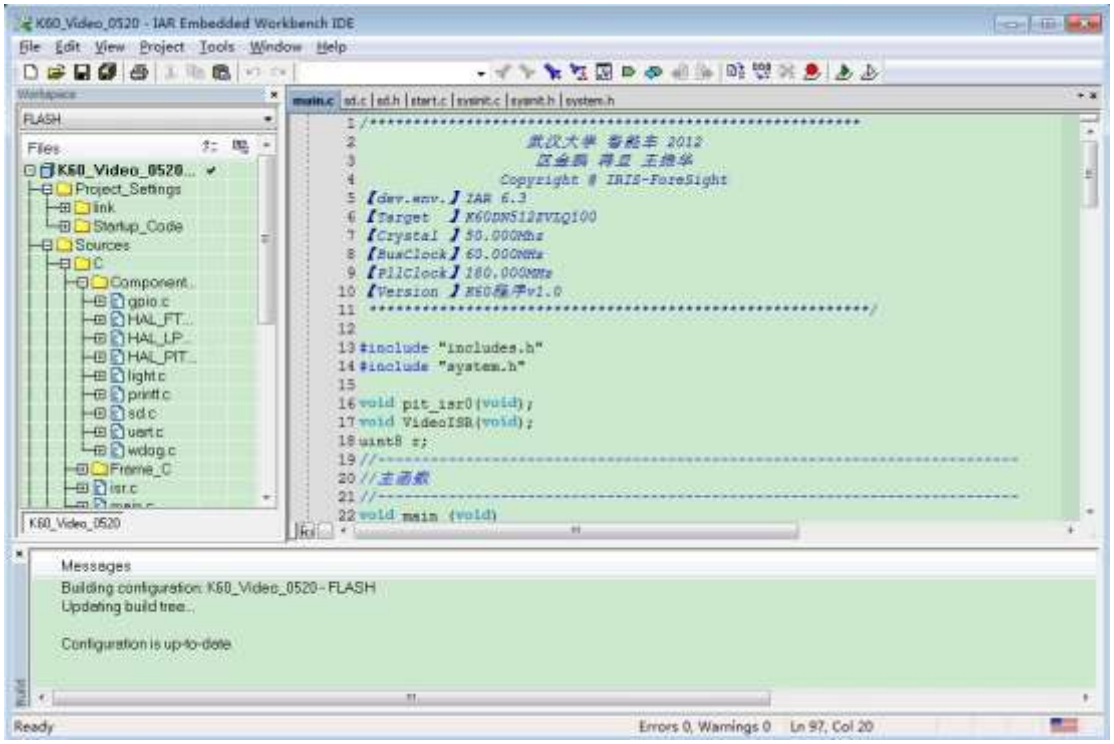


图 3.9：使用 IAR Embedded Workbench IDE 开发智能车程序

现代单片机具有很强的在线编程和调试功能，Freescale K60 系列单片机就具有 BDM（Background Debug Mode）功能，可以实现在线程序下载和在线背景调试功能。

3.4.2 上位机软件

为了在上位机上能控制赛车，实时监控其各参数和采集图像，我们开发了各种上位机程序。

（1） VC++开发的 CCD 智能小车监控软件

在智能车开发初期，我们使用 VC++开发了第一版的智能车监控程序，主要目的是为了观察单片机采集到的图像。

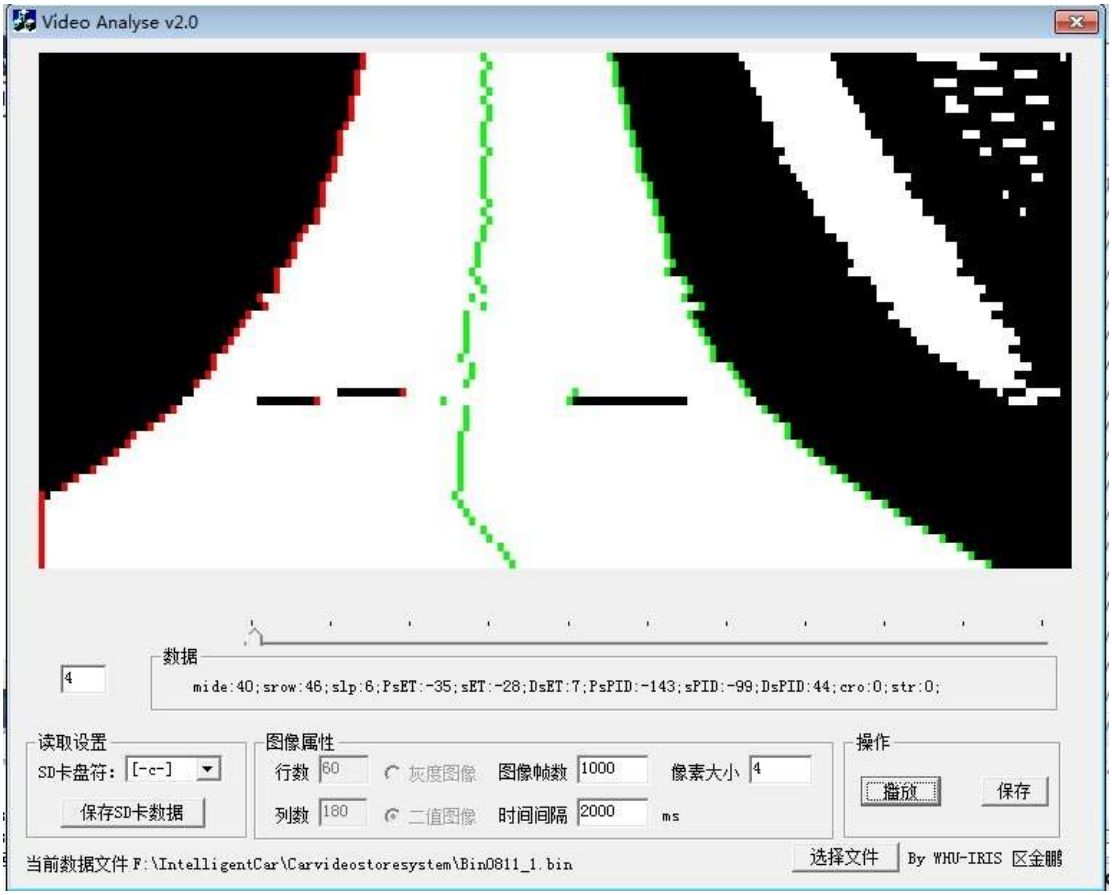


图 3.10：CCD 智能小车监控软件

（2）Matlab 开发的智能车监控程序

本软件的开发是为了在上位机进行各种图像处理算法和黑线位置计算的仿真。通过 Matlab 的 Serial 对象对计算机串口进行操作[2] ，和单片机通讯。

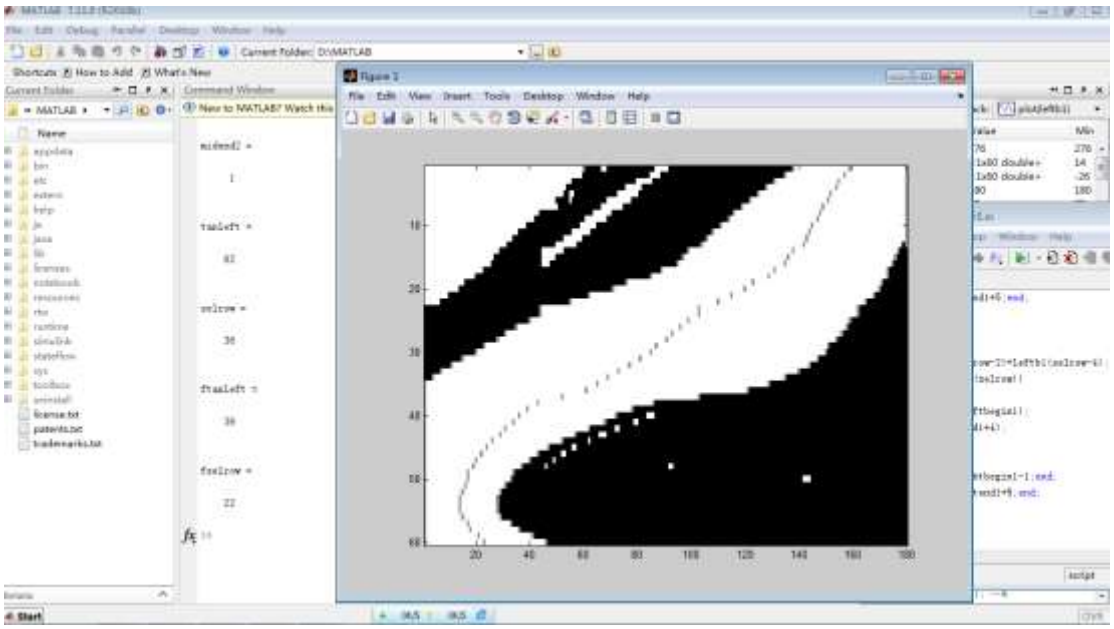


图 3.11　基于 Matlab 的智能车监控程序

（3）LabView 开发的小车监控程序

本程序实现监控赛车在运行中各种参数，如速度、设定速度、偏差、转角等的变化，并提供控制小车运行的功能。程序具有速度和偏差两个趋势图，可以进行实时曲线显示。

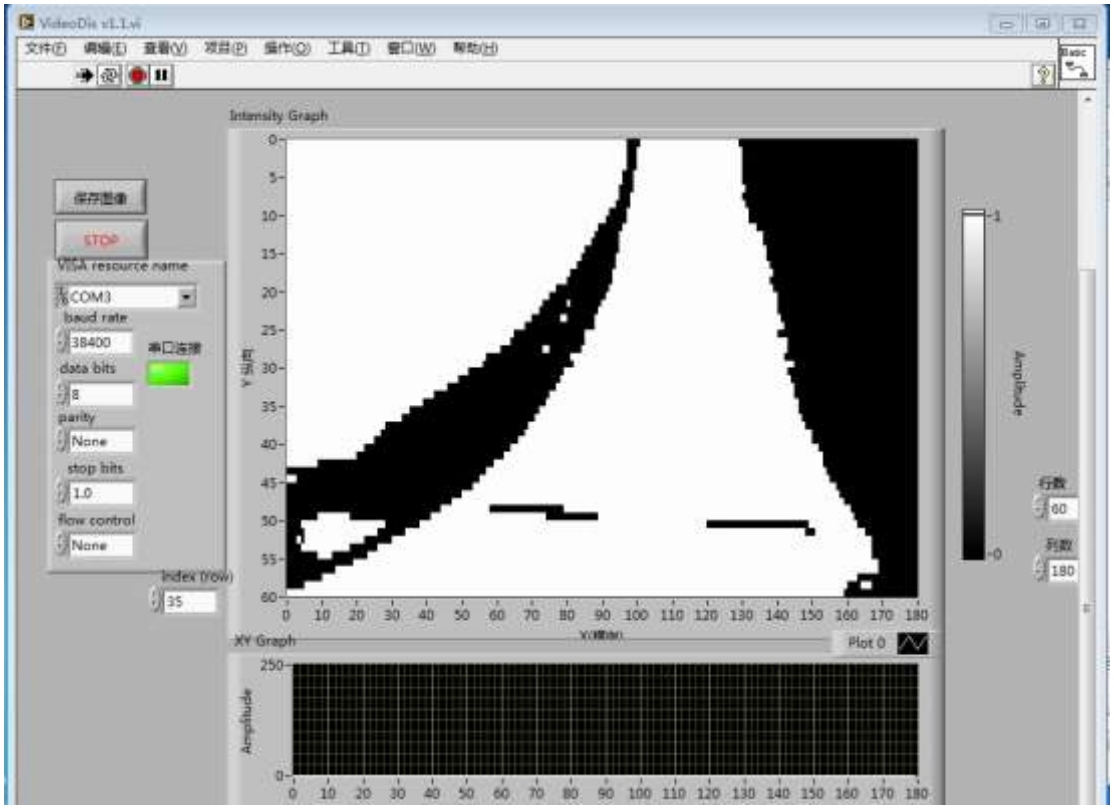


图 3.12 LabView 开发的智能车监控系统

第四章：模型车主要技术参数说明

本智能车主要技术参数如下：

表 4.1 智能车技术参数统计

项目	参数
路径检测方法（赛题组）	摄像头
车模几何尺寸（长、宽、高）（毫米）	280*160*310
车模轴距/轮距（毫米）	115
车模平均电流（匀速行驶）(毫安)	800
电路电容总量（微法）	1122.472597
传感器种类及个数	2
新增加伺服电机个数	0
赛道信息检测空间精度（毫米）	5
赛道信息检测频率（次/秒）	50
主要集成电路种类/数量	1
车模重量（带有电池）（千克）	1.010

参考文献

[1] “第一届‘飞思卡尔’杯全国大学生智能汽车邀请赛”组委会.“第一届‘飞思卡尔’杯全国大学生智能车邀请赛”比赛规则．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．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，2007．45~51

[3]卓晴著．智能汽车自动控制器方案设计．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，2007．5

[4]邹朗豪、李文昌、苏启健著．Rapid prototype and smartcar design．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,2007.71~86

[5]卓晴、王珌、王磊著．基于面阵 CCD 的赛道参数检测方法．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，2007．35~40

[6]Freescale, Inc．MC9S12DT128B Device User Guide V01.09．Freescale Semiconductor, Inc，2002

[7]Freescale, Inc．MC9S12DT128B CRG Block User Guide V03.08．Freescale Semiconductor, Inc，2002

[8]Freescale, Inc．MC33886 Datasheet．U.S.A:Freescale Semiconductor, Inc, 2005

[9]李立国、刘旺、郝杰等著．基于大前瞻光电识别和道路记忆方法的智能车．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，2007．338~350

[10] ON Semiconductor．NCP630 3.0A Fast Linear Voltage Regulators

[11] 余灿键、程东成、李伟强著．PID 算法在智能汽车设计上的应用．基于大前瞻光电识别和道路记忆方法的智能车．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，2007．128~137

[12] 黄开胜、陈宋著．汽车理论与智能模型车机械结构调整方法．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，2007．23

[13]广州周立功单片机发展有限公司著．I^2C 总线规范．广州：广州周立功单片机发展有限公司，2000

[14]李绍民著．图像传感器 OV7620 在自主足球机器人中的应用．国外电子元器件，2004 年第 10 期　　：11~14

[15]OmniVision Technologies, Inc．OV7620/OV7120 Specification．OmniVision, Inc，2001

[16]林辛凡,李红志,黄颖．第二届“飞思卡尔”杯全国大学生智能汽车邀请赛 技 术报告[R]， 2007 清华三角洲队（原清华一队）

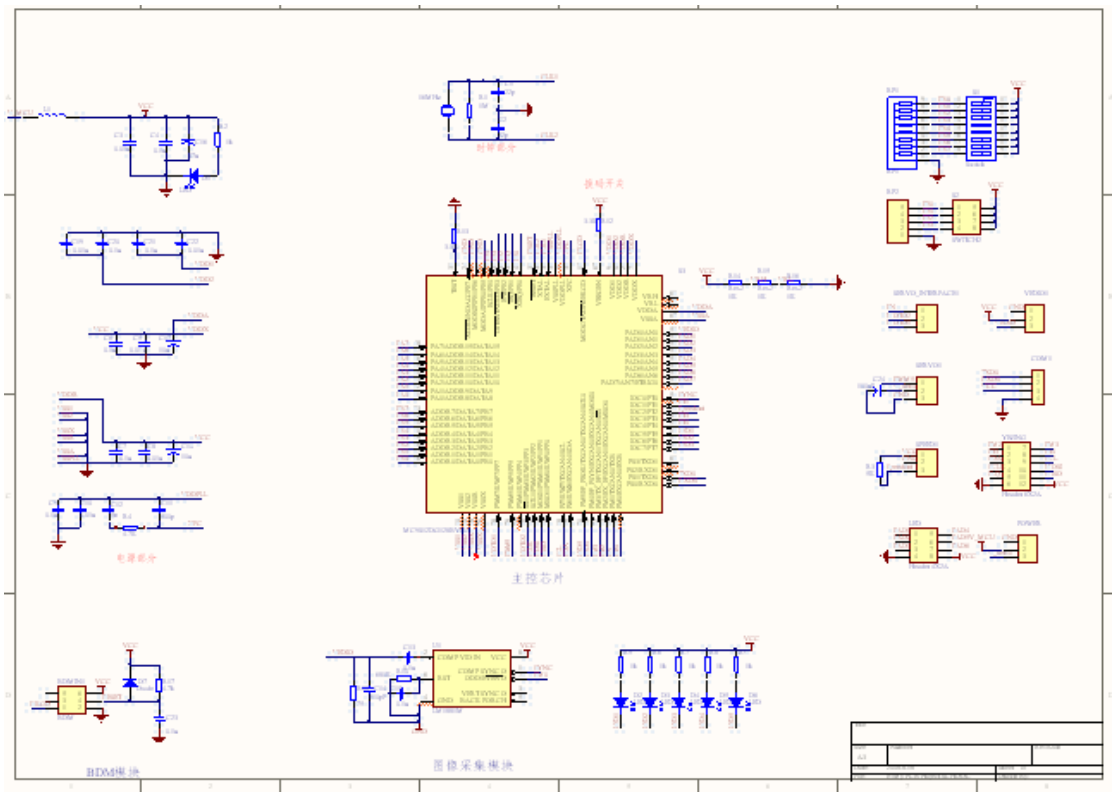
[17]吴小平、刘万春、朱玉文著．基于 TMS320C32 的视觉图像处理系统[J]．探测与控制学报．2000 年第 22 卷第 1 期：18~22

[18] 徐济仁、牛纪海、陈家松等著．FIFO 存储器 IDT7205 及其在接收机采样处理板中的应用．有线电视技术，2004 年第 24 期：62~64

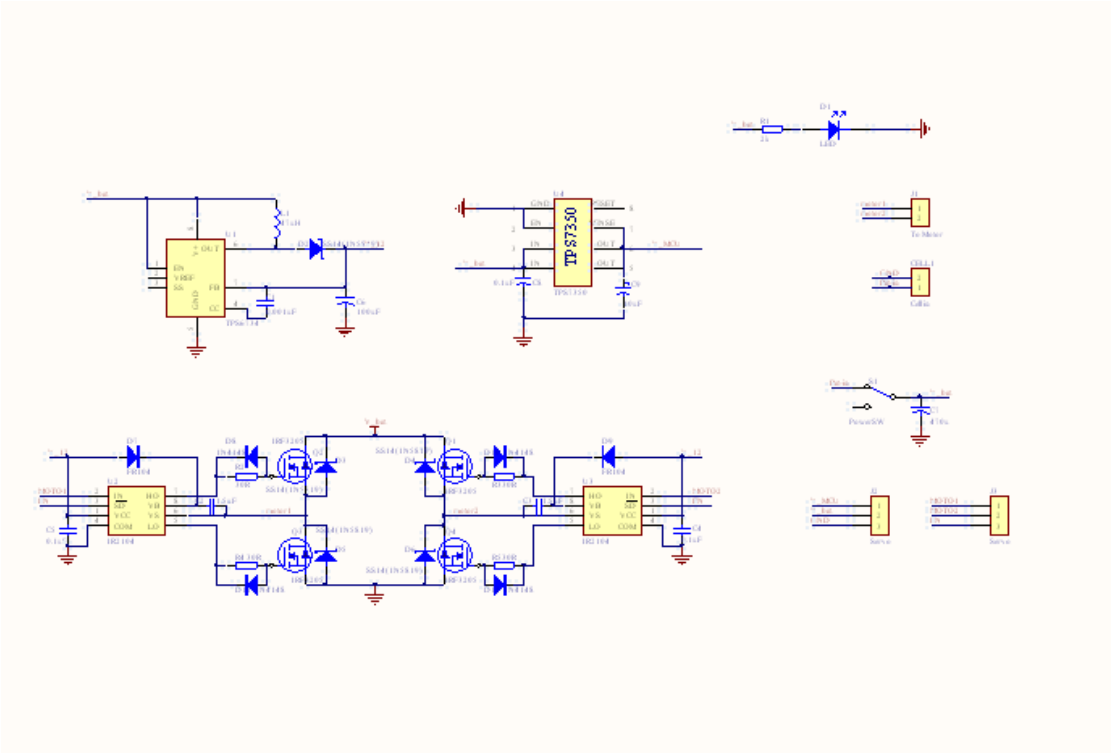
[19]沈庭芝、方子文著．数字图像处理及模式识别．北京：北京理工大学出版社，1998

附录：

附录 A：主控板原理图



附录 B：电源模块和驱动模块原理图



附录 C：主控程序

```
/*********************************************************
【dev.env.】IAR 6.3
【Target  】K60DN512
【Crystal 】50.000Mhz
【busclock】60.000MHz
【pllclock】180.000MHz

                    武汉大学 智能车 2012
                Copyright @ IRIS-ForeSight
 *********************************************************/

#include "includes.h"
#include "system.h"

void pit_isr0(void);
void VideoISR(void);

void main (void)
{
    DisableInterrupts;
    pllinit180M();
    periph_clk_khz = 180000/3;
    //打开所有 IO 时钟
    SIM_SCGC5 = SIM_SCGC5_PORTA_MASK | SIM_SCGC5_PORTB_MASK |
SIM_SCGC5_PORTC_MASK | SIM_SCGC5_PORTD_MASK | SIM_SCGC5_PORTE_MASK;
    //串口初始化
    uart_init (UART0,periph_clk_khz,38400);
    enableuartreint(UART0,UART0irq);          //开串口 3 接收中断

    //电机 10KHz PWM
    LPLD_FTM0_PWM_Init(10000);//frequency = 10 000
    LPLD_FTM0_PWM_Open(3,0);//duty = 0
    LPLD_FTM0_PWM_Open(4,0);//duty = 0;
    //舵机 100Hz PWM
   // LPLD_FTM1_PWM_Init(100);//frequency = 100
    //LPLD_FTM1_PWM_Open(0,1455);//duty = 15%
    hw_FTM1_init();
```

```
    //定时 10ms 中断初始化
    LPLD_PIT_Init(0, 10000, pit_isr0);
    //脉冲累加初始化
    LPLD_LPTMR_Init(1,0,2,0,NULL);
    //LED 初始化
    light_init(PORTA, 14, 1);
    light_init(PORTA, 15, 1);
    light_init(PORTA, 16, 1);
    light_init(PORTA, 17, 1);

    //GPIO 初始化
    PORTE_PCR24 = ( PORT_PCR_MUX(1) );//电机使能
    DDRE24 = 1;
    PTE24_OUT = 1;

    PORTE_PCR28 = ( PORT_PCR_MUX(1) );//视频读取 IO 设置
    DDRE28 = 0;
/*
    PORTE_PCR5      =      (      PORT_PCR_MUX(1)|PORT_PCR_IRQC(9)      |
PORT_PCR_PE_MASK );//行中断
    DDRE5 = 0;
    PORTE_PCR9      =      (      PORT_PCR_MUX(1)|PORT_PCR_IRQC(9)      |
PORT_PCR_PE_MASK );//场中断
    DDRE9 = 0;
    enable_irq(91);
*/
        SIM_SCGC5|=SIM_SCGC5_PORTE_MASK;// 启用 PORTE 时钟
        PORTE_PCR5=1<<8 | 0x9<<16| 1;   // GPIO , 下降边沿触发 ,开启上拉电阻
        GPIOE_PDDR&=~(1<<5);                    // 中断必须是输入模式
        PORTE_PCR9=1<<8 | 0xb<<16| 1;   // GPIO , 下降边沿触发 ,开启上拉电阻
        GPIOE_PDDR&=~(1<<9);
        enable_irq(91);

    Initvediodata();
    enFieldInt = 1;
    //uart_sendN (UART0 , (uint8*)"IRIS-ForeSight", 14);
    SetSpeed    = 80;
    type=60;
    send=0;
    last_rowcount=1;
```

```
    enFieldInt=1;
    enRowInt = 0;
    DelaymS(10000);
    light_change( PORTA, 17);
    light_change( PORTA, 16);
    DelaymS(1000);
    light_control(PORTA,17,1);
    light_control(PORTA,16,1);
    EnableInterrupts;

//    enRowInt=1;
    //主循环
    while(1)
    {/*
        if (startline_dete_flag==1)
        {
            if (startline_count2<5000)
                startline_count2++;

            if (startline_count2<4500&&startline_count2>300)
                SetSpeed=(-150);
            else
                SetSpeed=type;
            SpeedControl();
        }
*/
        SetSpeed=type;
        if(SpdFlag==1)
        {
            SpdFlag=0;
            SpeedControl();
      //        light_change(PORTA,16);
       /*        if (send==0)
            {
                uart_send1(UART0,Pulse>>8);
                uart_send1(UART0,Pulse);
                light_change(PORTA,17);
                DelaymS(100);
            }*/
        //            uart_send1(UART0,(uint8)(servoPID/10));
```

```
            }

            if
((row_ready==1)&&(rowcount<=ROW)&&(rowcount>=1)&&(last_rowcount<=rowcount))
            {
                searchline(ROW-last_rowcount);
                last_rowcount++;
            }//&&(fbegin!=0||fbegin2!=0)
            if (last_rowcount==(ROW+1)&&(servo_flag==0)&&(fieldcomplete==1))
            {
                light_control(PORTA,17,1);
                light_control(PORTA,16,1);
                light_control(PORTA,15,1);
                light_control(PORTA,14,1);
            /*  light_change(PORTA,17);*/
                servo_flag=1;//
                chuli();
              /*                                                                    if
((startline_count1>=150)&&(startline_dete_flag==0)&&(intabs(servoPID)<250))
                startline_detection();*/
                servocontrol();
                enFieldInt = 1;
            }
        }
}

//视频采集场行中断
void VideoISR(void)
{
    if(PORTE_ISFR & (1<<5))//行中断
    {
        PORTE_ISFR  = (PORTE_ISFR|(1<<5));          //写 1 清中断标志位
        if(enRowInt==1)
        {
            Row_ISR_PTE5();
        }
    }
    if(PORTE_ISFR & (1<<9))//场中断
    {
        PORTE_ISFR  = (PORTE_ISFR|(1<<9));          //写 1 清中断标志位
```

```
            if(enFieldInt==1)
            {
                Field_ISR_PTE9();
            }
        }
}

//10ms 测速中断程序
void pit_isr0(void)
{
    PIT_TFLG(0)|=PIT_TFLG_TIF_MASK;          //清标志
    Speed = LPTMR0_CNR;
    SpdFlag = 1;
    LPLD_LPTMR_Reset();          //Reset LPTMR0
    LPLD_LPTMR_Init(1,0,2,0,NULL);
}
//-------------------------------------------------------------------------*
//函数名: uart3_isr                                                        *
//功   能: 串口 3 数据接收中断例程                                          *
//说   明: 无                                                              *
//-------------------------------------------------------------------------*
void uart0_isr(void)
{
    uint8 ch;
    DisableInterrupts;        //关总中断
    //接收一个字节数据
    if(uart_re1 (UART0,&ch))
        {
          if (ch=='g')
        uart_send1(UART0,'y');
          send=1;
        }
    EnableInterrupts;         //开总中断
}

//10ms 测速中断程序
void pit_isr0()
{
    PIT_TFLG(0)|=PIT_TFLG_TIF_MASK;          //清标志
    Speed = LPTMR0_CNR;
```

```
    SpdFlag = 1;
    LPLD_LPTMR_Reset();             //Reset LPTMR0
    LPLD_LPTMR_Init(1,0,2,0,NULL);

}

//锁相环频率为 50/15*54=180M 函数
void pllinit180M(void)
{
    uint32_t temp_reg;
        //使能 IO 端口时钟
    SIM_SCGC5 |= (SIM_SCGC5_PORTA_MASK
                                    | SIM_SCGC5_PORTB_MASK
                                    | SIM_SCGC5_PORTC_MASK
                                    | SIM_SCGC5_PORTD_MASK
                                    | SIM_SCGC5_PORTE_MASK );
    //这里处在默认的 FEI 模式
    //首先移动到 FBE 模式
    MCG_C2 = 0;
    //MCG_C2 = MCG_C2_RANGE(2) | MCG_C2_HGO_MASK | MCG_C2_EREFS_MASK;
    //初始化晶振后释放锁定状态的振荡器和 GPIO
    SIM_SCGC4 |= SIM_SCGC4_LLWU_MASK;
    LLWU_CS |= LLWU_CS_ACKISO_MASK;

    //选择外部晶振，参考分频器，清 IREFS 来启动外部晶振
    //011 If RANGE = 0, Divide Factor is 8; for all other RANGE values, Divide Factor is 256.
    MCG_C1 = MCG_C1_CLKS(2) | MCG_C1_FRDIV(3);

    //等待晶振稳定
    //while (!(MCG_S & MCG_S_OSCINIT_MASK)){}                   //等待锁相环初始化
结束
    while (MCG_S & MCG_S_IREFST_MASK){}                         //等待时钟切换到外
部参考时钟
    while (((MCG_S & MCG_S_CLKST_MASK) >> MCG_S_CLKST_SHIFT) != 0x2){}

    //进入 FBE 模式,
    //0x18==25 分频=2M,
    //0x08==15 分频=3.333M
    //0x09==16 分频=3.125M,
    //0x10==17 分频=2.94M
```

```
    //0x11==18 分频=2.7778M
    //0x12==19 分频=2.63M,
    //0x13==20 分频=2.5M
    MCG_C5 = MCG_C5_PRDIV(0x0e);

    //确保 MCG_C6 处于复位状态，禁止 LOLIE、PLL、和时钟控制器，清 PLL VCO 分频器
    MCG_C6 = 0x0;

    //保存 FMC_PFAPR 当前的值
    temp_reg = FMC_PFAPR;

    //通过 M&PFD 置位 M0PFD 来禁止预取功能
    FMC_PFAPR |= FMC_PFAPR_M7PFD_MASK | FMC_PFAPR_M6PFD_MASK | FMC_PFAPR_M5PFD_MASK
                | FMC_PFAPR_M4PFD_MASK | FMC_PFAPR_M3PFD_MASK | FMC_PFAPR_M2PFD_MASK
                | FMC_PFAPR_M1PFD_MASK | FMC_PFAPR_M0PFD_MASK;
    ///设置系统分频器
    //MCG=PLL, core = MCG, bus = MCG/3, FlexBus = MCG/3, Flash clock= MCG/8
    SIM_CLKDIV1 = SIM_CLKDIV1_OUTDIV1(0) | SIM_CLKDIV1_OUTDIV2(2)
                | SIM_CLKDIV1_OUTDIV3(2) | SIM_CLKDIV1_OUTDIV4(7);

    //从新存 FMC_PFAPR 的原始值
    FMC_PFAPR = temp_reg;

    //设置 VCO 分频器，使能 PLL 为 100MHz, LOLIE=0, PLLS=1, CME=0, VDIV=26
    MCG_C6 = MCG_C6_PLLS_MASK | MCG_C6_VDIV(30);   //VDIV = 31 (x54)
                                                   //VDIV = 26 (x50)
    while (!(MCG_S & MCG_S_PLLST_MASK)){}; // wait for PLL status bit to set
    while (!(MCG_S & MCG_S_LOCK_MASK)){}; // Wait for LOCK bit to set

    //进入 PBE 模式
    //通过清零 CLKS 位来进入 PEE 模式
    // CLKS=0, FRDIV=3, IREFS=0, IRCLKEN=0, IREFSTEN=0
    MCG_C1 &= ~MCG_C1_CLKS_MASK;

    //等待时钟状态位更新
    while (((MCG_S & MCG_S_CLKST_MASK) >> MCG_S_CLKST_SHIFT) != 0x3){};
    //SIM_CLKDIV2 |= SIM_CLKDIV2_USBDIV(1);
```

```
    //设置跟踪时钟为内核时钟
    SIM_SOPT2 |= SIM_SOPT2_TRACECLKSEL_MASK;
    //在 PTA6 引脚上使能 TRACE_CLKOU 功能
    PORTA_PCR6 = ( PORT_PCR_MUX(0x7));
    //使能 FlexBus 模块时钟
    SIM_SCGC7 |= SIM_SCGC7_FLEXBUS_MASK;
    //在 PTA6 引脚上使能 FB_CLKOUT 功能
    PORTC_PCR3 = ( PORT_PCR_MUX(0x5));
}


//-------------------------------------------------------------------------*
//  文件名: gpio.c                                                         *
//  说    明: gpio 驱动程序文件                                            *
//-------------------------------------------------------------------------*

#include "gpio.h"        //包含 gpio 头文件

//-------------------------------------------------------------------------*
//函数名: gpio_init                                                       *
//功    能:  初始化 gpio                                                  *
//参    数: port:端口名                                                   *
//              index:指定端口引脚                                        *
//              dir:引脚方向,0=输入,1=输出                                *
//              data:初始状态,0=低电平,1=高电平                           *
//返    回:  无                                                          *
//说    明:  无                                                          *
//-------------------------------------------------------------------------*
void gpio_init (GPIO_MemMapPtr port, int index, int dir,int data)
{
    PORT_MemMapPtr p;
    switch((uint32)port)
    {
    case 0x400FF000u:
        p = PORTA_BASE_PTR;
        break;
    case 0x400FF040u:
        p = PORTB_BASE_PTR;
        break;
```

```
        case 0x400FF080u:
            p = PORTC_BASE_PTR;
            break;
        case 0x400FF0C0u:
            p = PORTD_BASE_PTR;
            break;
        case 0x400FF100u:
            p = PORTE_BASE_PTR;
            break;
        default:
            break;
        }
        PORT_PCR_REG(p,index)=(0|PORT_PCR_MUX(1));

        if(dir == 1)//output
        {
            GPIO_PDDR_REG(port) |= (1<<index);
            if(data == 1)//output
                GPIO_PDOR_REG(port) |= (1<<index);
            else
                GPIO_PDOR_REG(port) &= ~(1<<index);
        }

        else
            GPIO_PDDR_REG(port) &= ~(1<<index);

}

//-------------------------------------------------------------------------*
//函数名: gpio_ctrl                                                        *
//功    能: 设置引脚状态                                                    *
//参    数: port:端口名                                                    *
//          index:指定端口引脚                                             *
//          data: 状态,0=低电平,1=高电平                                   *
//返    回: 无                                                            *
//说    明: 无                                                            *
//-------------------------------------------------------------------------*
void gpio_ctrl (GPIO_MemMapPtr port, int index, int data)
{
    if(data == 1)//output
```

```
        GPIO_PDOR_REG(port) |= (1<<index);
    else
        GPIO_PDOR_REG(port) &= ~(1<<index);
}

//-------------------------------------------------------------------------*
//函数名: gpio_reverse                                                     *
//功    能: 改变引脚状态                                                   *
//参    数: port:端口名;                                                   *
//          index:指定端口引脚                                            *
//返    回: 无                                                           *
//说    明: 无                                                           *
//-------------------------------------------------------------------------*
void gpio_reverse (GPIO_MemMapPtr port, int index)
{
    GPIO_PDOR_REG(port) ^= (1<<index);
}

    #include "gpio.h"        //包含 gpio 头文件
void gpio_init (GPIO_MemMapPtr port, int index, int dir,int data)
    {
        PORT_MemMapPtr p;
        switch((uint32)port)
        {
        case 0x400FF000u:
            p = PORTA_BASE_PTR;
            break;
        case 0x400FF040u:
            p = PORTB_BASE_PTR;
            break;
        case 0x400FF080u:
            p = PORTC_BASE_PTR;
            break;
        case 0x400FF0C0u:
            p = PORTD_BASE_PTR;
            break;
        case 0x400FF100u:
```

```
            p = PORTE_BASE_PTR;
            break;
        default:
            break;
        }
        PORT_PCR_REG(p,index)=(0|PORT_PCR_MUX(1));

        if(dir == 1)//output
        {
            GPIO_PDDR_REG(port) |= (1<<index);
            if(data == 1)//output
                GPIO_PDOR_REG(port) |= (1<<index);
            else
                GPIO_PDOR_REG(port) &= ~(1<<index);
        }

        else
            GPIO_PDDR_REG(port) &= ~(1<<index);

    }

    void gpio_ctrl (GPIO_MemMapPtr port, int index, int data)
    {
        if(data == 1)//output
            GPIO_PDOR_REG(port) |= (1<<index);
        else
            GPIO_PDOR_REG(port) &= ~(1<<index);
    }

    void gpio_reverse (GPIO_MemMapPtr port, int index)
    {
        GPIO_PDOR_REG(port) ^= (1<<index);
    }
void chuli(void)
```

```
{
/***************率去偏离太大的点***************/

 if ((fbegin>2&&fbegin<44)&&(fbegin2>2&&fbegin2<44))
    {
        fbegin=fbegin-3;
        fbegin2=fbegin2-3;
    }

 if (fbegin-fend>=5)
  {
     for (i=(fbegin-1);i>=fend;i--)
        {
           if (intabs(left_heixian[i]-left_heixian[i+1])>=filter)
              {
                 fend=i+1;
                 break;
              }
        }
  }

    if (fbegin2-fend2>=5)
      {
        for (i=fbegin2-1;i>=fend2;i--)
           {
              if (intabs(right_heixian[i]-right_heixian[i+1])>=filter)
                 {
```

```
                    fend2=i+1;
                    break;
                }
            }
        }


 zhidao_flag=0;
 if (fbegin-fend>=40&&fbegin2-fend2>=40&&fend==0&&fend2==0)
     zhidao_flag=1;


/****************** 交叉路口 ********************/
crossing_flag=0;
if (fend>=5&&fend2>=5)
   {
     if (cCor[0][fend-1]==-1&&cCor[1][fend2-1]==500&&intabs(fend-fend2)<=20)
          crossing_flag=1;

     if
(cCor[0][fend-2]==-1&&left_heixian[fend-2]-left_heixian[fend-1]<-10&&cCor[1][fend2-2]==50
0&&left_heixian[fend-1]!=0&&(intabs(fend-fend2)<=20))
          crossing_flag=1;

     if
(cCor[0][fend-2]==-1&&right_heixian[fend2-2]-right_heixian[fend2-1]>10&&cCor[1][fend2-2]=
=500&&right_heixian[fend2-1]!=0&&(intabs(fend-fend2)<=20))
```

```
            crossing_flag=1;

    }
    /**/
    if (fbegin<=44&&fbegin>20&&fbegin2>20&&fbegin2<=44&&fend==0&&fend2==0)
         crossing_flag=1;



    crossing_count=0;
    crossing_fbegin=0;
    crossing_fend=0;
    crossing_fbegin2=0;
    crossing_fend2=0;

    if (fbegin-fend<=10&&fbegin2-fend2<=10)
       {
          for (i=fend-6;i>=0;i--)
             {
                if
(cCor[0][i]==-1&&cCor[0][i+1]==-1&&cCor[0][i+2]==-1&&cCor[0][i+3]==-1&&cCor[0][i+4]
==-1)
                      crossing_count=1;

                if (crossing_count==1&&cCor[0][i]!=-1&&cCor[0][i]!=0&&crossing_fbegin==0)
                      crossing_fbegin=i;

                if (crossing_fbegin!=0&&(cCor[0][i]==-1||cCor[0][i]==0))
```

```
                crossing_fend=i+1;

            if (cCor[0][i]==0)
                break;
        }


    crossing_fbegin=crossing_fbegin-3;
    if (crossing_fbegin-crossing_fend>=5)
        {
            for (i=crossing_fbegin-1;i>=crossing_fend;i--)
                {
                    if (intabs(left_heixian[i]-left_heixian[i+1])>=20)
                        {
                            crossing_fend=i+1;
                            break;
                        }
                }
        }

    crossing_count=0;
    for (i=fend2-6;i>=0;i--)
        {
            if
(cCor[1][i]==500&&cCor[1][i+1]==500&&cCor[1][i+2]==500&&cCor[1][i+3]==500&&cCor[1
][i+4]==500)
                crossing_count=1;
```

```
        if (crossing_count==1&&cCor[1][i]!=500&&cCor[1][i]!=0&&crossing_fbegin2==0)
            crossing_fbegin2=i;

        if (crossing_fbegin2!=0&&(cCor[1][i]==500||cCor[1][i]==0))
            crossing_fend2=i+1;

        if (cCor[1][i]==0)
            break;
    }

crossing_fbegin2=crossing_fbegin2-3;
if (crossing_fbegin2-crossing_fend2>=5)
    {
        for (i=crossing_fbegin2-1;i>=crossing_fend2;i--)
            {
                if (intabs(right_heixian[i]-right_heixian[i+1])>=20)
                    {
                        crossing_fend2=i+1;
                        break;
                    }
            }
    }

if (crossing_fbegin-crossing_fend>=20&&crossing_fbegin2-crossing_fend2>=20)
    {
        crossing_flag=1;
```

```
            fbegin=crossing_fbegin;
            fend=crossing_fend;
            fbegin2=crossing_fbegin2;
            fend2=crossing_fend2;
        }

    }
//0524_7.txt
crossing_biaoji1=0;
crossing_biaoji2=0;
crossing_m1=0;
crossing_m2=0;

if (fbegin-fend<10&&fbegin2-fend2<10&&crossing_flag==1)
{
   if (biaoji4>biaoji3)
        for (i=fend2;i>=0;i--)
        {
            if (crossing_m2!=0)
                crossing_m2=crossing_m2+1;

            if (intabs(cCor[1][i]-cCor[1][fbegin2])<20&&crossing_biaoji2==0)
            {
                fbegin2=i;
                crossing_biaoji2=crossing_biaoji2+1;
                crossing_m2=crossing_m2+1;
            }
```

```
            if (cCor[1][i]==0)
                break;
            if (crossing_m2!=crossing_biaoji2)
            {
                fend2=i+1;
                break;
            }
        }
else
        for (i=fend;i>=0;i--)
        {
            if (crossing_m1!=0)
                crossing_m1=crossing_m1+1;

            if (intabs(cCor[0][i]-cCor[0][fbegin])<20&&crossing_biaoji1==0)
            {
                fbegin=i;
                crossing_biaoji1=crossing_biaoji1+1;
                crossing_m1=crossing_m1+1;
            }
            if (cCor[0][i]==0)
                break;

            if (crossing_m1!=crossing_biaoji1)
            {
                fend=i+1;
                break;
```

```
                }
            }
        crossing_flag=0;
    }
    //***************  检测小 S ****************************//
    small_S_flag=0;

    if
((fbegin2-fend2>=30)&&(fbegin-fend>=30)&&(fend<5)&&(fend2<5)&&(fbegin>=34)&&(fbegi
n2>=34))
        small_S_flag=1;

    //********   滤去交叉路口中 90 度的点  ***************//
    if (fbegin-fend>5)
        {
          for (i=fbegin-5;i>=fend;i--)
            {
              if
(left_heixian[i+4]-left_heixian[i+5]<-2&&left_heixian[i+3]-left_heixian[i+4]<-2&&left_heixian[
i+2]-left_heixian[i+3]<-2&&left_heixian[i+1]-left_heixian[i+2]<-2&&left_heixian[i]-left_heixia
n[i+1]<-2)
                {
                  fend=i+5;
                  break;
                }
            }
        }
```

```
    if (fbegin2-fend2>5)
     {
        for (i=fbegin2-5;i>=fend2;i--)
           {
              if
(right_heixian[i+4]-right_heixian[i+5]>2&&right_heixian[i+3]-right_heixian[i+4]>2&&right_hei
xian[i+2]-right_heixian[i+3]>2&&right_heixian[i+1]-right_heixian[i+2]>2&&right_heixian[i]-rig
ht_heixian[i+1]>2)
                 {
                    fend2=i+5;
                    break;
                 }
           }
     }


    slope_flag=0;

    if (fbegin>fbegin2)
        slope_begin=fbegin2;
    else
        slope_begin=fbegin;

    if (fend>fend2)
        slope_end=fend;
    else
```

```
    slope_end=fend2;


if (slope_begin-slope_end>0&&fbegin>44&&fbegin2>44)
 {
    for (i=slope_begin;i>=slope_end;i--)
       {
          if (right_heixian[i]-left_heixian[i]<factor2[i]-9&&right_heixian[i]-left_heixian[i]>0)
                slope++;
          if (right_heixian[i]-left_heixian[i]>factor2[i]+3)
                break;
       }
    if (slope>=6)
          slope_flag=1;

    if (slope_end<=5)
       {
          for (i=slope_end;i<=slope_begin;i++)
             {
                if (right_heixian[i]-left_heixian[i]>factor2[i]+8)
                      slope2++;
                else
                      break;
             }

          if (slope2>=6)
             slope_flag=1;
       }
```

```
    }
}
#include "includes.h"

//锁相环频率为 50/15*54=180M 函数
void pllinit180M(void);
void delay(void);
void DelaymS(int delay);
void DelayuS( int delay);
float floatabs(float a) ;
int intabs(int a);
//采集视频
void GetVedio(void);
void Initvediodata(void);
void InitPara(void);
//发送一场数据
void uart_send_data(UART_MemMapPtr uartch);
void chuli(void);
void servocontrol(void);
//检测起跑线
void startline_detection(void);
//速度控制;
void SpeedControl(void);
void hw_FTM1_init(void);
extern int periph_clk_khz;//总线频率

//+++++++舵机的 PWM 的常数定义 +++++++++++++++//
```

```
#define SERVO_MID     5335// 舵机的中心位置   //B2782     ----2722
#define SERVO_R        540 //   // 最大右边位置，大概在 38 度左右  1800
#define SERVO_L       -540   //    // 最大左边位置，大概在 38 度左右
//vedio parameter
#define ROW 60
],*g_vedio;
int rowcount=0,last_rowcount=0,fieldcomplete=0,rowsyncnt=0;
int SpeedE1, SpeedE2, SpeedE3, SetPWM = 0, SpdFlag, SetSpeed, Speed;
int PID[3];
int filter=20;
const int hop[ROW]={30,45,58,69,79,88,
                    95,101,107,113,118,123,
                    128,133,137,141,145,149,
                    153,157,161,164,167,170,
                    173,176,179,182,185,188,
                    191,193,195,197,199,201,
                    203,205,207,209,211,213,
                    215,217,219,221,223,224,
                    225,226,227,228,229,230,
                    231,232,233,234,235,236,
                   };

int i,j,fbegin=0,fend=0,fbegin2=0,fend2=0,fact_fend,fact_fbegin;
int crossing_flag=0,small_S_flag=0,zhidao_flag=0;
int j_begin=0,j_end=0;
int count=0;
char left=0,right=0;
```

```
int servoPID = 0,last_servoPID=0,crossing_servoPID;
int mm=0,mm2=0,m_1=0,m_2=0,last_m_1=0,last_m_2=0;
unsigned char send=0;
int cCor[2][ROW],left_heixian[ROW],right_heixian[ROW];
int type=0,row_ready=0,kk=0;

//交叉路口变量
int crossing_biaoji1=0,crossing_biaoji2=0,crossing_m1=0,crossing_m2=0;


int slope=0,slope2=0,slope_begin=0,slope_end=0,slope_flag,bound=20;
float k_p=0;
int crossing_count=0,crossing_fbegin=0,crossing_fend=0,crossing_fbegin2=0,crossing_fend2=0;

int last_VS=1;
int zhongxian[ROW];

int all_white_flag=0,biaoji=0,biaoji2=0,biaoji3=0,biaoji4=0;
int black_biaoji=0;
int
black_begin=0,black_end=0,black_begin1=0,black_begin2=0,black_begin3=0,black_begin4=0;
int count1=0,count2=0,count3=0,count4=0;
int servo_flag=0;
int crossing_count1=0,crossing_count2=0;

int startline_dete_flag=0;
int startline1=0,startline2=0,startline3=0,startline4=0;
```

```
int startline_count1=0,startline_count2=0;
short int Pulse=0;

int shuaiwei_flag=0,shuaiwei_flag2=0;
//*****曲率计算用的变量****///
int a,b,c;
int kur;
int rev;
int enFieldInt = 0, enRowInt = 0;
const int delta[ROW]=
{
207,202,202,198,193,193,
189,185,185,185,178,178,
178,175,171,165,165,159,
159,153,148,148,144,144,
139,137,135,133,131,129,
125,122,119,117,114,113,
110,109,106,105,102,100,
98,96,94,92,90,88,
86,84,82,81,78,77,
75,73,71,68,66,64,
};

int factor2[ROW]= {
43,44,44,45,46,46,
47,48,48,48,50,50,
50,51,52,54,54,56,
```

```
56,58,60,60,62,62,
64,65,66,67,68,69,
71,73,75,76,78,79,
81,82,84,85,87,89,
91,93,95,97,99,101,
104,106,108,110,114,115,
119,122,126,130,135,140,
};


//锁相环频率为 50/15*54=180M 函数
void pllinit180M(void)
{
    uint32_t temp_reg;
        //使能 IO 端口时钟
    SIM_SCGC5 |= (SIM_SCGC5_PORTA_MASK
                                    | SIM_SCGC5_PORTB_MASK
                                    | SIM_SCGC5_PORTC_MASK
                                    | SIM_SCGC5_PORTD_MASK
                                    | SIM_SCGC5_PORTE_MASK );
    //这里处在默认的 FEI 模式
    //首先移动到 FBE 模式
    MCG_C2 = 0;
    //MCG_C2 = MCG_C2_RANGE(2) | MCG_C2_HGO_MASK | MCG_C2_EREFS_MASK;
    //初始化晶振后释放锁定状态的振荡器和 GPIO
    SIM_SCGC4 |= SIM_SCGC4_LLWU_MASK;
    LLWU_CS |= LLWU_CS_ACKISO_MASK;
```

```
    //选择外部晶振，参考分频器，清 IREFS 来启动外部晶振
    //011 If RANGE = 0, Divide Factor is 8; for all other RANGE values, Divide Factor is 256.
    MCG_C1 = MCG_C1_CLKS(2) | MCG_C1_FRDIV(3);

    //等待晶振稳定
    //while (!(MCG_S & MCG_S_OSCINIT_MASK)){}                    //等待锁相环初始化结束
    while (MCG_S & MCG_S_IREFST_MASK){}                          //等待时钟切换到外部参考时钟
    while (((MCG_S & MCG_S_CLKST_MASK) >> MCG_S_CLKST_SHIFT) != 0x2){}

    //进入 FBE 模式,
    //0x18==25 分频=2M,
    //0x08==15 分频=3.333M
    //0x09==16 分频=3.125M,
    //0x10==17 分频=2.94M
    //0x11==18 分频=2.7778M
    //0x12==19 分频=2.63M,
    //0x13==20 分频=2.5M
    MCG_C5 = MCG_C5_PRDIV(0x0e);

    //确保 MCG_C6 处于复位状态，禁止 LOLIE、PLL、和时钟控制器，清 PLL VCO 分频器
    MCG_C6 = 0x0;

    //保存 FMC_PFAPR 当前的值
```

```
    temp_reg = FMC_PFAPR;

    //通过 M&PFD 置位 M0PFD 来禁止预取功能
    FMC_PFAPR |= FMC_PFAPR_M7PFD_MASK | FMC_PFAPR_M6PFD_MASK |
FMC_PFAPR_M5PFD_MASK
                | FMC_PFAPR_M4PFD_MASK | FMC_PFAPR_M3PFD_MASK |
FMC_PFAPR_M2PFD_MASK
                | FMC_PFAPR_M1PFD_MASK | FMC_PFAPR_M0PFD_MASK;
    ///设置系统分频器
    //MCG=PLL, core = MCG, bus = MCG/3, FlexBus = MCG/3, Flash clock= MCG/8
    SIM_CLKDIV1 = SIM_CLKDIV1_OUTDIV1(0) | SIM_CLKDIV1_OUTDIV2(2)
                | SIM_CLKDIV1_OUTDIV3(2) | SIM_CLKDIV1_OUTDIV4(7);

    //从新存 FMC_PFAPR 的原始值
    FMC_PFAPR = temp_reg;

    //设置 VCO 分频器，使能 PLL 为 100MHz, LOLIE=0, PLLS=1, CME=0, VDIV=26
    MCG_C6 = MCG_C6_PLLS_MASK | MCG_C6_VDIV(30);   //VDIV = 31 (x54)
                                                   //VDIV = 26 (x50)
    while (!(MCG_S & MCG_S_PLLST_MASK)){}; // wait for PLL status bit to set
    while (!(MCG_S & MCG_S_LOCK_MASK)){}; // Wait for LOCK bit to set

    //进入 PBE 模式
    //通过清零 CLKS 位来进入 PEE 模式
    // CLKS=0, FRDIV=3, IREFS=0, IRCLKEN=0, IREFSTEN=0
    MCG_C1 &= ~MCG_C1_CLKS_MASK;
```

```
    //等待时钟状态位更新
    while (((MCG_S & MCG_S_CLKST_MASK) >> MCG_S_CLKST_SHIFT) != 0x3){};
    //SIM_CLKDIV2 |= SIM_CLKDIV2_USBDIV(1);

    //设置跟踪时钟为内核时钟
    SIM_SOPT2 |= SIM_SOPT2_TRACECLKSEL_MASK;
    //在 PTA6 引脚上使能 TRACE_CLKOU 功能
    PORTA_PCR6 = ( PORT_PCR_MUX(0x7));
    //使能 FlexBus 模块时钟
    SIM_SCGC7 |= SIM_SCGC7_FLEXBUS_MASK;
    //在 PTA6 引脚上使能 FB_CLKOUT 功能
    PORTC_PCR3 = ( PORT_PCR_MUX(0x5));
}
/*=====================================================================
======
FTM1c0 PWM 输出初始化函数
The edge-aligned mode is selected when (QUADEN = 0), (DECAPEN = 0), (COMBINE
= 0), (CPWMS = 0), and (MSnB = 1).
K60P144M100SF2RM.pdf    P1011 39.4.6 Edge-Aligned PWM (EPWM) Mode
The EPWM period is determined by (MOD - CNTIN + 0x0001) and the pulse width
(duty cycle) is determined by (CnV - CNTIN).
//=====================================================================
======*/
/*PMW 频率=系统频率/4/(2^FTM1_SC_PS)/FTM1_MOD
        =125 000 000/4/(2^5)/19531
        =50HZ*/
```

```
void hw_FTM1_init(void)
{
   //SIM_SOPT4|=SIM_SOPT4_FTM1FLT0_MASK;
   /* Turn on all port clocks */
   SIM_SCGC5 |= SIM_SCGC5_PORTA_MASK;

   // PTA8 K60P144M100SF2.pdf  第 68 页  8.1 K60 Signal Multiplexing and Pin Assignments
   PORTA_PCR8 = PORT_PCR_MUX(0x3)| PORT_PCR_DSE_MASK;; // FTM is alt3 function for this pin

   SIM_SCGC6|=SIM_SCGC6_FTM1_MASK;          //使能 FTM1 时钟

   //change MSnB = 1
   FTM1_C0SC |= FTM_CnSC_ELSB_MASK;
   FTM1_C0SC &= ~FTM_CnSC_ELSA_MASK;
   FTM1_C0SC |= FTM_CnSC_MSB_MASK;

   //FTM1_SC = FTM_SC_PS(0) | FTM_SC_CLKS(1);
   //FTM1_SC=0X0F;
   FTM1_SC = 0x2b; //not enable the interrupt mask,up-down counting mode,System clock,Divide by 32
   //FTM1_SC=0X1F;          //BIT5   0 FTM counter operates in up counting mode.
                            //1 FTM counter operates in up-down counting mode.
   //BIT43 FTM1_SC|=FTM1_SC_CLKS_MASK;
                            //00 No clock selected (This in effect disables the FTM counter.)
```

```
                                        //01 System clock
                                        //10 Fixed frequency clock
                                        //11 External clock
    //BIT210 FTM1_SC|=FTM1_SC_PS_MASK;
                                        //100M              MOD=2000;
                                        //000 Divide by 1---12KHZ
                                        //001 Divide by 2--- 6KHZ
                                        //010 Divide by 4--- 3K
                                        //011 Divide by 8--- 1.5K
                                        //100 Divide by 16---750
                                        //101 Divide by 32---375
                                        //110 Divide by 64---187.5HZ
                                        //111 Divide by 128--93.75hz

    FTM1_MODE |= FTM_MODE_WPDIS_MASK;
     //BIT1      Initialize the Channels Output
    //FTMEN is bit 0, need to set to zero so DECAPEN can be set to 0
    FTM1_MODE &= ~1;
     //BIT0      FTM Enable
     //0 Only the TPM-compatible registers (first set of registers) can be used without any restriction. Do not use the FTM-specific registers.
     //1 All registers including the FTM-specific registers (second set of registers) are available for use with no restrictions.

    FTM1_OUTMASK=0XFE;     //0 Channel output is not masked. It continues to operate normally.
                                        //1 Channel output is masked. It is forced to its inactive state.
```

```
    FTM1_COMBINE=0;          //Function for Linked Channels (FTMx_COMBINE)
    FTM1_OUTINIT=0;
    FTM1_EXTTRIG=0;          //FTM External Trigger (FTMx_EXTTRIG)
    FTM1_POL=0;              //Channels Polarity (FTMx_POL)
                             //0 The channel polarity is active high.
                             //1 The channel polarity is active low.
    //Set Edge Aligned PWM
    FTM1_QDCTRL &=~FTM_QDCTRL_QUADEN_MASK;
    //QUADEN is Bit 1, Set Quadrature Decoder Mode (QUADEN) Enable to 0,     (disabled)
    //FTM0_SC = 0x16; //Center Aligned PWM Select = 0, sets FTM Counter to operate in up counting mode,
    //it is field 5 of FTMx_SC (status control) - also setting the pre-scale bits here

    FTM1_INVCTRL=0;        //反转控制
    FTM1_SWOCTRL=0;          // 软 件 输 出 控 制  F  TM  Software  Output  Control (FTMx_SWOCTRL)
    FTM1_PWMLOAD=0;        //FTM PWM Load
                           //BIT9: 0 Loading updated values is disabled.
                           //1 Loading updated values is enabled.
    FTM1_CNTIN=0;          //Counter Initial Value
   //  FTM1_MOD=28125;          //Modulo value,The EPWM period is determined by (MOD - CNTIN + 0x0001)
                           //采用龙丘时钟初始化函数，可以得到 4 分频的频率，例如：系统 60M 频率时，PWM 频率是 15M,以此类推
                           //PMW 频率=系统频率/4/(2^FTM1_SC_PS)/FTM1_MOD
     FTM1_MOD=2003550;                                    //PMW          频          率
```

```
=180M/4/(2^4)/28125=180000000/4/16/28125=100Hz
        //PMW 分频=180M/4/(2^4)=2.8125M,一个脉冲是 1us,2109 个脉冲就是 1.5ms
        FTM1_C0V=SERVO_MID;          //设置 the pulse width(duty cycle) is determined by
(CnV - CNTIN).
        FTM1_CNT=0;                  //只有低 16 位可用-560
}

void FTM2_QUAD_init()
{
     /*开启端口时钟*/
    SIM_SCGC5 |= SIM_SCGC5_PORTA_MASK;

    /*选择管脚复用功能*/
    PORTA_PCR10 = PORT_PCR_MUX(6);
    PORTA_PCR11 = PORT_PCR_MUX(6);

    /*使能 FTM2 时钟*/
    SIM_SCGC3|=SIM_SCGC3_FTM2_MASK;

    FTM2_MOD = 65535;//可根据需要设置

    FTM2_CNTIN = 0;

    FTM2_MODE |= FTM_MODE_WPDIS_MASK; //禁止写保护
    FTM2_MODE |= FTM_MODE_FTMEN_MASK; //FTMEN=1,关闭 TPM 兼容模式，开启
FTM 所有功能
```

```
    FTM2_QDCTRL &= ~FTM_QDCTRL_QUADMODE_MASK; //选定编码模式为 A 相与 B 相编码模式
    FTM2_QDCTRL |= FTM_QDCTRL_QUADEN_MASK; //使能正交解码模式

    FTM2_SC |= FTM_SC_CLKS(3);//选择外部时钟
    FTM2_CONF |=FTM_CONF_BDMMODE(3);//可根据需要选择
    FTM2_CNT = 0;
}

//Delay 函数
void delay(void)
{
    int i =0;
    int j=0;
    for(i=0;i<1000;i++)
        for(j=0;j<1000;j++)
            asm("nop");
}

//Delay 1ms
void DelaymS(int delay)
{
    int   i, j;
    for ( i = delay; i != 0; i -- )
    {
        for ( j = 7000; j != 0; j -- ) ;
        for ( j = 6320; j != 0; j -- ) ;
```

```
        }
}
//Delay 1us
void DelayuS( int delay )
{
        int    i;
        int j;
        for ( i=delay; i!= 0; i-- )
        {
                for ( j = 1; j != 0; j -- ) ;
        }
}
float floatabs(float a)
{
   return (a>0)?a:-a;
}
int intabs(int a)
{
        if (a>=0)
           return a;
        else
           return (-a);
}


}

void Initvediodata(void)
```

```
{
   int i,j;
   for(i=0;i<ROW;i++)
   for(j=0;j<COLUMN;j++)
   data[i][j]=0;
}

void InitPara(void)
{
      rowcount=0;
      rowsyncnt=0;
      last_rowcount=1;
      m_1=0;

      all_white_flag=0;
      biaoji=0;
      biaoji2=0;
      biaoji3=0;
      biaoji4=0;
      black_biaoji=0;
      black_begin=0;
      black_end=0;
      black_begin1=0;
      black_begin2=0;
      black_begin3=0;
      black_begin4=0;
      crossing_flag=0;
```

```
j=0;
count1=0;
count2=0;
count3=0;
count4=0;
servo_flag=0;
m_1=0;
last_m_1=0;
m_2=0;
last_m_2=0;
j_begin=0;
j_end=COLUMN;
count=10;
left=0;
right=0;
mm=0;
mm2=0;
fbegin=0;
fbegin2=0;
fend=0;
fend2=0;
slope=0;
slope2=0;
for (i=0;i<ROW;i++)
{
   cCor[0][i]=0;
   cCor[1][i]=0;
```

```
            left_heixian[i]=0;
            right_heixian[i]=0;
            zhongxian[i]=0;
        }
        row_ready=0;
        fieldcomplete=0;
        type=200;
        rev=10;
}

//GPIO 中断服务函数，场同步中断
void Field_ISR_PTE9(void)
{
         servo_flag=0;
        if (startline_count1<=200)
                    startline_count1++;
        rowcount=0;
        rowsyncnt=0;
        last_rowcount=1;
        m_1=0;

        all_white_flag=0;
        biaoji=0;
        biaoji2=0;
        biaoji3=0;
        biaoji4=0;
        black_biaoji=0;
```

```
black_begin=0;
black_end=0;
black_begin1=0;
black_begin2=0;
black_begin3=0;
black_begin4=0;
crossing_flag=0;
j=0;
count1=0;
count2=0;
count3=0;
count4=0;

m_1=0;
last_m_1=0;
m_2=0;
last_m_2=0;
j_begin=0;
j_end=COLUMN;
count=10;
left=0;
right=0;
mm=0;
mm2=0;
fbegin=0;
fbegin2=0;
fend=0;
```

```
    fend2=0;
    slope=0;
    slope2=0;
    for (i=0;i<ROW;i++)
      {
        cCor[0][i]=0;
        cCor[1][i]=0;
        left_heixian[i]=0;
        right_heixian[i]=0;
      }
    row_ready=0;
    fieldcomplete=0;
}

//GPIO 中断服务函数，行同步中断
void Row_ISR_PTE5(void)
{
    rowsyncnt++;          //视频采集行同步 counter

      g_vedio=data[ROW-rowcount-1]+COLUMN-1;
      GetVedio();
      rowcount++;
      row_ready=1;
    if (rowcount>=ROW)
    {
      enFieldInt = 0;
      enRowInt = 0;
```

```
            fieldcomplete=1;
            if(fieldcomplete==1)
            {
                if (send == 1)
                {
                    DisableInterrupts;
                    uart_send1(UART0,'x');
                    uart_send_data(UART0);    //发送一场数据
                    EnableInterrupts;
                }/**/
            }
        }
}


//发送一场数据
void uart_send_data(UART_MemMapPtr uartch)
{
    int i,j;
    for(i=0;i<ROW;i++)
    {
       for (j=0;j<COLUMN;j++)
          uart_send1(uartch,data[i][j]);
    }
}

void searchline(int RC)
```

```
{
   if (RC==59)
      {
            black_biaoji=0;
            for (j=j;j<=(COLUMN-1);j++)
              {
                 if (data[RC][j]==1&&black_biaoji==0)
                    {
                       black_begin1=j;
                       black_biaoji=black_biaoji+1;
                    }
                 if (data[RC][j]==1&&black_biaoji!=0)
                       count1=count1+1;

                 if (data[RC][j]==0&&black_biaoji!=0)
                       break;
              }

             black_biaoji=0;
             for (j=j;j<=(COLUMN-1);j++)
              {
                 if (data[RC][j]==1&&black_biaoji==0)
                    {
                       black_begin2=j;
                       black_biaoji=black_biaoji+1;
                    }
```

```
    if (data[RC][j]==1&&black_biaoji!=0)
        count2=count2+1;

    if (data[RC][j]==0&&black_biaoji!=0)
        break;
  }
black_biaoji=0;
for (j=j;j<=(COLUMN-1);j++)
  {
    if (data[RC][j]==1&&black_biaoji==0)
       {
         black_begin3=j;
         black_biaoji=black_biaoji+1;
       }

    if (data[RC][j]==1&&black_biaoji!=0)
        count3=count3+1;

    if (data[RC][j]==0&&black_biaoji!=0)
        break;
  }
black_biaoji=0;
for (j=j;j<=(COLUMN-1);j++)
  {
    if (data[RC][j]==1&&black_biaoji==0)
       {
         black_begin4=j;
```

```
            black_biaoji=black_biaoji+1;
          }

        if (data[RC][j]==1&&black_biaoji!=0)
            count4=count4+1;

        if (data[RC][j]==0&&black_biaoji!=0)
            break;
      }

if (count1>=count2&&count1>=count3&&count1>=count4)
  {
    black_begin=black_begin1;
    black_end=black_begin+count1-1;
  }
if(count2>=count1&&count2>=count3&&count2>=count4)
  {
    black_begin=black_begin2;
    black_end=black_begin+count2-1;
  }
if (count3>=count1&&count3>=count2&&count3>=count4)
  {
    black_begin=black_begin3;
    black_end=black_begin+count3-1;
  }
if (count4>=count1&&count4>=count2&&count4>=count3)
  {
```

```
            black_begin=black_begin4;
            black_end=black_begin+count4-1;
          }

        black_begin=black_begin+20;
        black_end=black_end-20;
        if (black_begin>black_end)
            black_begin=black_end;

        if (count1>=(COLUMN-4))
          {
            black_begin=COLUMN/2;
            black_end=COLUMN/2;
          }
    }

if ((biaoji==0||biaoji2==0||all_white_flag==1)||(right!=0||left!=0||RC>=4))
    {
      if (all_white_flag==0)
      {
        last_m_1=m_1;
        last_m_2=m_2;
        if (biaoji3>=2)
          {
            if (left==1)
              {
                m_1=left_heixian[RC+1]-left_heixian[RC+2]+20;
```

```
            if (left_heixian[RC+1]+m_1>=(COLUMN-1))
               m_1=(COLUMN-1)-left_heixian[RC+1];

            if (left_heixian[RC+1]+m_1<=29)
               m_1=29-left_heixian[RC+1];

            j_begin=left_heixian[RC+1]+m_1;
          }
        else
          {
            m_1=last_m_1;
          }
  }
else
  {
     j_begin=black_begin;
  }


if (biaoji4>=2)
  {
     if (right==1)
        {
           m_2=right_heixian[RC+1]-right_heixian[RC+2]-20;

           if (right_heixian[RC+1]+m_2>=149)
```

```
                m_2=149-right_heixian[RC+1];

              if (right_heixian[RC+1]+m_2<=0)
                m_2=0-right_heixian[RC+1];

              j_end=right_heixian[RC+1]+m_2;
            }
           else
            {
              m_2=last_m_2;
            }
        }
       else
        {
          j_end=black_end;
        }
  }

else
  {
    j_begin=COLUMN/2;
    j_end=COLUMN/2;
  }

  if (j_begin>j_end)
    {
      if (biaoji3>biaoji4)
```

```
            j_end=j_begin-4;

        if (biaoji4>biaoji3)
            j_begin=j_end+4;

    }

if (j_end<0)
    j_end=0;

if (j_begin>COLUMN-1)
    j_begin=COLUMN-1;


    all_white_flag=0;
    crossing_count1=0;
    crossing_count2=0;
    left=0;
    right=0;
    count=0;

    for (j=j_begin;j>=2;j--)
     {
        if (data[RC][j]==1)
            crossing_count1=crossing_count1+1;
        if (data[RC][j]==1&&data[RC][j-1]==0&&data[RC][j-2]==0)
            {
```

```
                left=1;
                left_heixian[RC]=j;
                if (biaoji3==0)
                  {
                    if (left_heixian[RC]<=140)
                    {
                      cCor[0][RC]=(left_heixian[RC]-(COLUMN/2))*delta[RC];
                      cCor[0][RC]/=100;
                      cCor[0][RC]+=186;
                      biaoji3=biaoji3+1;
                      break;
                    }
                  }
                else
                  {
                      cCor[0][RC]=(left_heixian[RC]-(COLUMN/2))*delta[RC];
                      cCor[0][RC]/=100;
                      cCor[0][RC]+=186;
                      biaoji3=biaoji3+1;
                      break;
                  }
            }
        }
    for (j=j_end;j<=(COLUMN-3);j++)
        {
          if (data[RC][j]==1)
              crossing_count2=crossing_count2+1;
```

```
    if (data[RC][j]==1&&data[RC][j+1]==0&&data[RC][j+2]==0)
      {
        right=1;
        right_heixian[RC]=j+2;
        if (biaoji4==0)
          {
            if (right_heixian[RC]>=10)
              {
                cCor[1][RC]=(right_heixian[RC]-(COLUMN/2))*delta[RC];
                cCor[1][RC]/=100;
                cCor[1][RC]+=186;
                biaoji4=biaoji4+1;
                break;
              }
          }
        else
          {
                cCor[1][RC]=(right_heixian[RC]-(COLUMN/2))*delta[RC];
                cCor[1][RC]/=100;
                cCor[1][RC]+=186;
                biaoji4=biaoji4+1;
                break;
          }
      }

}
```

```
            if
(crossing_count1==j_begin-1&&crossing_count2==(COLUMN-2)-j_end&&(biaoji==1||biaoji2==1))
                all_white_flag=1;

            if (mm!=0)
                mm=mm+1;

            if (biaoji3==1&&mm==0&&left_heixian[RC]<=140)
              {
                fbegin=RC;
                mm=mm+1;
              }

            if (mm!=biaoji3&&biaoji==0)
              {
                fend=RC+1;
                biaoji=biaoji+1;
              }

            if (mm2!=0)
               mm2=mm2+1;

            if (biaoji4==1&&mm2==0)
              {
                fbegin2=RC;
```

```
                mm2=mm2+1;
              }

            if (mm2!=biaoji4&&biaoji2==0)
              {
                fend2=RC+1;
                biaoji2=biaoji2+1;
              }

            if (crossing_count1==j_begin-1&&left!=1)
                cCor[0][RC]=-1;

            if (crossing_count2==COLUMN-2-j_end&&right!=1)
                cCor[1][RC]=500;
        }
}

void chuli(void)
{
/***************率去偏离太大的点***************/

  if ((fbegin>2&&fbegin<44)&&(fbegin2>2&&fbegin2<44))
    {
        fbegin=fbegin-3;
        fbegin2=fbegin2-3;
    }
```

```
if (fbegin-fend>=5)
  {
     for (i=(fbegin-1);i>=fend;i--)
        {
           if (intabs(left_heixian[i]-left_heixian[i+1])>=filter)
              {
                 fend=i+1;
                 break;
              }
        }
  }

    if (fbegin2-fend2>=5)
      {
         for (i=fbegin2-1;i>=fend2;i--)
            {
               if (intabs(right_heixian[i]-right_heixian[i+1])>=filter)
                  {
                     fend2=i+1;
                     break;
                  }
            }
      }


zhidao_flag=0;
if (fbegin-fend>=40&&fbegin2-fend2>=40&&fend==0&&fend2==0)
```

```
        zhidao_flag=1;



/****************** 交叉路口 *********************/
crossing_flag=0;
if (fend>=5&&fend2>=5)
   {
      if (cCor[0][fend-1]==-1&&cCor[1][fend2-1]==500&&intabs(fend-fend2)<=20)
            crossing_flag=1;

      if
(cCor[0][fend-2]==-1&&left_heixian[fend-2]-left_heixian[fend-1]<-10&&cCor[1][fend2-2]==50
0&&left_heixian[fend-1]!=0&&(intabs(fend-fend2)<=20))
            crossing_flag=1;

      if
(cCor[0][fend-2]==-1&&right_heixian[fend2-2]-right_heixian[fend2-1]>10&&cCor[1][fend2-2]=
=500&&right_heixian[fend2-1]!=0&&(intabs(fend-fend2)<=20))
            crossing_flag=1;

   }
   /**/
   if (fbegin<=44&&fbegin>20&&fbegin2>20&&fbegin2<=44&&fend==0&&fend2==0)
        crossing_flag=1;



   crossing_count=0;
```

```
    crossing_fbegin=0;
    crossing_fend=0;
    crossing_fbegin2=0;
    crossing_fend2=0;

    if (fbegin-fend<=10&&fbegin2-fend2<=10)
       {
          for (i=fend-6;i>=0;i--)
             {
                if
(cCor[0][i]==-1&&cCor[0][i+1]==-1&&cCor[0][i+2]==-1&&cCor[0][i+3]==-1&&cCor[0][i+4]
==-1)
                      crossing_count=1;

                if (crossing_count==1&&cCor[0][i]!=-1&&cCor[0][i]!=0&&crossing_fbegin==0)
                      crossing_fbegin=i;

                if (crossing_fbegin!=0&&(cCor[0][i]==-1||cCor[0][i]==0))
                      crossing_fend=i+1;

                if (cCor[0][i]==0)
                      break;
             }

          crossing_fbegin=crossing_fbegin-3;
          if (crossing_fbegin-crossing_fend>=5)
```

```
        {
            for (i=crossing_fbegin-1;i>=crossing_fend;i--)
                {
                    if (intabs(left_heixian[i]-left_heixian[i+1])>=20)
                        {
                            crossing_fend=i+1;
                            break;
                        }
                }
        }

    crossing_count=0;
    for (i=fend2-6;i>=0;i--)
        {
            if
(cCor[1][i]==500&&cCor[1][i+1]==500&&cCor[1][i+2]==500&&cCor[1][i+3]==500&&cCor[1
][i+4]==500)
                    crossing_count=1;

            if (crossing_count==1&&cCor[1][i]!=500&&cCor[1][i]!=0&&crossing_fbegin2==0)
                    crossing_fbegin2=i;

            if (crossing_fbegin2!=0&&(cCor[1][i]==500||cCor[1][i]==0))
                    crossing_fend2=i+1;

            if (cCor[1][i]==0)
                    break;
```

```
        }

    crossing_fbegin2=crossing_fbegin2-3;
    if (crossing_fbegin2-crossing_fend2>=5)
        {
            for (i=crossing_fbegin2-1;i>=crossing_fend2;i--)
                {
                    if (intabs(right_heixian[i]-right_heixian[i+1])>=20)
                        {
                            crossing_fend2=i+1;
                            break;
                        }
                }
        }

    if (crossing_fbegin-crossing_fend>=20&&crossing_fbegin2-crossing_fend2>=20)
        {
            crossing_flag=1;
            fbegin=crossing_fbegin;
            fend=crossing_fend;
            fbegin2=crossing_fbegin2;
            fend2=crossing_fend2;
        }

  }
//0524_7.txt
crossing_biaoji1=0;
```

```
crossing_biaoji2=0;
crossing_m1=0;
crossing_m2=0;

if (fbegin-fend<10&&fbegin2-fend2<10&&crossing_flag==1)
{
   if (biaoji4>biaoji3)
        for (i=fend2;i>=0;i--)
        {
            if (crossing_m2!=0)
                crossing_m2=crossing_m2+1;

            if (intabs(cCor[1][i]-cCor[1][fbegin2])<20&&crossing_biaoji2==0)
            {
                fbegin2=i;
                crossing_biaoji2=crossing_biaoji2+1;
                crossing_m2=crossing_m2+1;
            }
            if (cCor[1][i]==0)
                break;
            if (crossing_m2!=crossing_biaoji2)
            {
                fend2=i+1;
                break;
            }
        }
   else
```

```
            for (i=fend;i>=0;i--)
            {
                if (crossing_m1!=0)
                    crossing_m1=crossing_m1+1;

                if (intabs(cCor[0][i]-cCor[0][fbegin])<20&&crossing_biaoji1==0)
                {
                    fbegin=i;
                    crossing_biaoji1=crossing_biaoji1+1;
                    crossing_m1=crossing_m1+1;
                }
                if (cCor[0][i]==0)
                    break;

                if (crossing_m1!=crossing_biaoji1)
                {
                    fend=i+1;
                    break;
                }
            }
        crossing_flag=0;
    }
    //***************  检测小 S ****************************//
    small_S_flag=0;

    if
((fbegin2-fend2>=30)&&(fbegin-fend>=30)&&(fend<5)&&(fend2<5)&&(fbegin>=34)&&(fbegi
```

```
n2>=34))
        small_S_flag=1;

   //********   滤去交叉路口中 90 度的点  ***************//
   if (fbegin-fend>5)
      {
        for (i=fbegin-5;i>=fend;i--)
           {
              if
(left_heixian[i+4]-left_heixian[i+5]<-2&&left_heixian[i+3]-left_heixian[i+4]<-2&&left_heixian[
i+2]-left_heixian[i+3]<-2&&left_heixian[i+1]-left_heixian[i+2]<-2&&left_heixian[i]-left_heixia
n[i+1]<-2)
                 {
                   fend=i+5;
                   break;
                 }
           }
      }

    if (fbegin2-fend2>5)
      {
        for (i=fbegin2-5;i>=fend2;i--)
           {
              if
(right_heixian[i+4]-right_heixian[i+5]>2&&right_heixian[i+3]-right_heixian[i+4]>2&&right_hei
xian[i+2]-right_heixian[i+3]>2&&right_heixian[i+1]-right_heixian[i+2]>2&&right_heixian[i]-rig
ht_heixian[i+1]>2)
```

```
                {
                    fend2=i+5;
                    break;
                }
            }
        }



slope_flag=0;

if (fbegin>fbegin2)
    slope_begin=fbegin2;
else
    slope_begin=fbegin;

if (fend>fend2)
    slope_end=fend;
else
    slope_end=fend2;

if (slope_begin-slope_end>0&&fbegin>44&&fbegin2>44)
  {
    for (i=slope_begin;i>=slope_end;i--)
        {
          if (right_heixian[i]-left_heixian[i]<factor2[i]-9&&right_heixian[i]-left_heixian[i]>0)
                slope++;
          if (right_heixian[i]-left_heixian[i]>factor2[i]+3)
```

```
                    break;
                }
            if (slope>=6)
                slope_flag=1;

            if (slope_end<=5)
                {
                for (i=slope_end;i<=slope_begin;i++)
                    {
                    if (right_heixian[i]-left_heixian[i]>factor2[i]+8)
                        slope2++;
                    else
                        break;
                    }

                if (slope2>=6)
                    slope_flag=1;
                }
        }
}

//检测起跑线
void startline_detection(void)
{
    for (i=39;i<=49;i++)
    {
     startline1=0;
```

```
startline2=0;
startline3=0;
startline4=0;
if
((right_heixian[i]-left_heixian[i]>=10)&&(fend==0&&fend2==0)&&left_heixian[i]!=0&&right_
heixian[i]!=0)
    {
      for (j=left_heixian[i]+2;j<=right_heixian[i]-4;j++)
        {
          if (data[i][j]==1&&data[i][j+1]==0&&data[i][j+2]==0)
            {
              startline1=j;
              break;
            }
        }
      for (j=j;j<=right_heixian[i]-4;j++)
        {
          if (data[i][j]==0&&data[i][j+1]==1&&data[i][j+2]==1)
            {
              startline2=j;
              break;
            }
        }

      for (j=j;j<=right_heixian[i]-4;j++)
        {
          if (data[i][j]==1&&data[i][j+1]==0&&data[i][j+2]==0)
```

```
                {
                    startline3=j;
                    break;
                }
            }

        for (j=j;j<=right_heixian[i]-4;j++)
            {
                if (data[i][j]==0&&data[i][j+1]==1&&data[i][j+2]==1)
                    {
                        startline4=j;
                        break;
                    }
            }

        if (startline2-startline1>=5||startline4-startline3>=5)
            {
                startline_dete_flag=1;
                break;
            }
        }
    }
}
```