第六届“飞思卡尔”杯全国大学生
智能汽车竞赛

技 术 报 告

学　　校：大连理工大学
队伍名称：狩猎者
参赛队员：宋　曦
　　耿晓杨
　　李华龙

带队教师：吴振宇
　　李航

关于技术报告和研究论文使用授权的说明

本人完全了解第四届“飞思卡尔”杯全国大学生智能汽车邀请赛关保留、使用技术报告和研究论文的规定，即：参赛作品著作权归参赛者本人，比赛组委会和飞思卡尔半导体公司可以在相关主页上收录并公开参赛作品的设计方案、技术报告以及参赛模型车的视频、图像资料，并将相关内容编纂收录在组委会出版论文集中。

参赛队员签名：_____

带队教师签名：_____

日　　期：_____

目　　录

引　言

随着电子科技的不断发展，越来越多的自动化设备开始进入到人们的生产生活中，嵌入式的迅猛发展为智能研究提供了更广阔的平台。在工业生产、科学探索、救灾抢险、军事等方面，人工智能发挥着越来越重要的作用，在此背景下，智能控制策略变得尤为重要。

“飞思卡尔”杯全国大学生杯智能汽车竞赛是国家教学质量与教学改革工程资助项目，以飞思卡尔半导体公司生产的 16 位单片机为核心控制模块，通过增加道路传感器、电机驱动电路以及编写相应程序，制作一个能够自主识别道路的汽车模型。因而该竞赛是涵盖了智能控制、模式识别、传感技术、汽车电子、电气、计算机、机械等多个学科的比赛，对学生的知识融合和实践能力的提高，具有良好的推动作用。

本文采用第六届“飞思卡尔”杯全国大学生智能车竞赛的汽车模型作为研究平台，以 16 位单片机 MC9S12XS128 作为主控制单元，运用 Code Warrior 软件作为开发工具进行智能控制策略研究。道路信息检测模块普遍采用使用简单、速度快的数字类摄像头，但是 CCD 在工作稳定性、分辨率、价格等方面均优于数字类摄像头，基于研究需要，经过综合考虑，本设计采用 CCD 传感器采集道路信息。

本届车模后置双电机，因此需要对两个电机分别进行速度检测，而单片机系统只有一个脉冲累加器，本设计创新性采用锁存器对左右测速模块进行分时复用，从而实现了左右两轮的准确无冲突测速。

在电源模块设计中，由于 CCD 传感器对电压要求较高，要达到 12V，LM2940 和 LT1085 已经不能够满足要求，因此采用 LT1946 稳压芯片将电池电压稳压到 12V 来对 CCD 传感器进行供电。

CCD 传感器数据处理电路主要部分为 LM1881 和模数转换。由于 CCD 传感器输出的是复合信号，因此需要用 LM1881 进行视频信号的分离，主要为行信号与场信号的分离以及奇偶场的识别等，模数转换的作用是将 CCD 传感器输出的模拟信号转换成数字信号。

单片机软件算法部分为本文论述的重点，主要体现了智能车的智能控制策略，在经过大量的试验后，我们决定采用斜率法计算转向角度，在计算斜率时采用最小二乘法进行曲线拟合，计算出斜率，根据斜率及截距大小来确定转向角度大小。经过长达数月的调试，智能车各项功能均达到设计要求。

第一章　智能车机械部分安装与设计

本文在第六届“飞思卡尔”杯全国大学生智能车竞赛组委会提供的摄像头组车模基础上进行智能车的组装和设计。其中前轮倾角的调节关系到智能车高速行驶的稳定性，经过长时间调节，最终确定出利于智能车行驶的前轮倾角。

1.1　前轮倾角的调节

智能车前轮倾角主要分为三种，即前束角、上开倾角和后倾角。

1.1.1　前束角

前束角是指从车辆上方向下俯看轮胎时，左右两个轮胎的中心线所构成的夹角。前束角分为两种，分别为正前束和负前束。当两轮胎中心线所构成的夹角是前窄后宽时为正前束，如图 1.1 所示。当两轮胎中心线所构成的夹角是前宽后窄时为负前束，如图 1.2 所示。

图 1.1　正前束

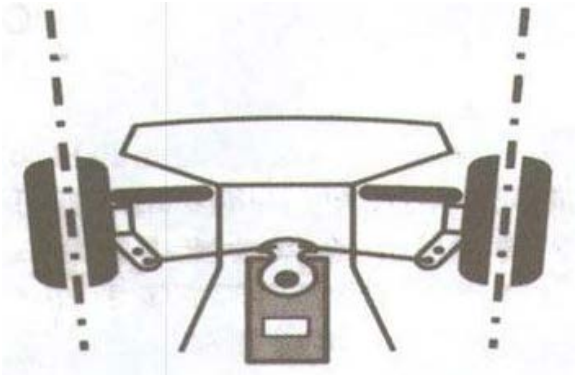

图 1.2　负前束

两者相比，正前束拥有较佳的直线稳定性，但是有一个缺点，就是在加速出弯时，容易发生车尾滑动的现象。相反，负前束在转弯后期的反应比较温和，此时即使大胆的操控转向，车体的姿态也会很稳定。但是与正前束相比，直线稳定性变差，跑直线就会变的很难。

在正常行驶时，智能车速度一般在 2 m/s 以上，直线行驶时会超过 3 m/s，在如此高速的情况下，如果车体姿态不稳定会导致智能车瞬间冲出跑道或者花费大量时间调整车体姿态，导致整体速度下降。基于对车体姿态稳定性要求较高的考虑，本智能车系统采用负前束倾角。

1.1.2 上开倾角

上开倾角是从正面看车子时，左右两个轮胎中心线所形成的夹角。当夹角上宽下窄时称为正上开倾角，如图 1.3 所示。反之，上窄下宽时称为负上开倾角，如图 1.4 所示。对上开倾角的调整一般是为了改变转弯时轮胎的抓地力。

转弯时，因为产生离心力，所以车辆会朝圆心外方倾斜。此时，若能设定出恰当的负上开倾角，将会在车辆倾斜时让轮胎全面接触路面。在抓地力完全发挥的情况下，整个转弯的过程就会变得更容易执行。另外，为了提高直线稳定性，往往也会设定为负上开倾角。车子的转弯特性，如果前轮的抓地力高于后轮的话，就会造成转向过度；如果后轮的抓地力高于前轮的话，就会造成转向不足。

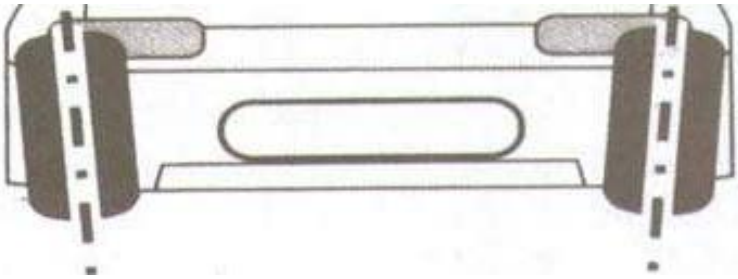

图 1.3　正上开倾角

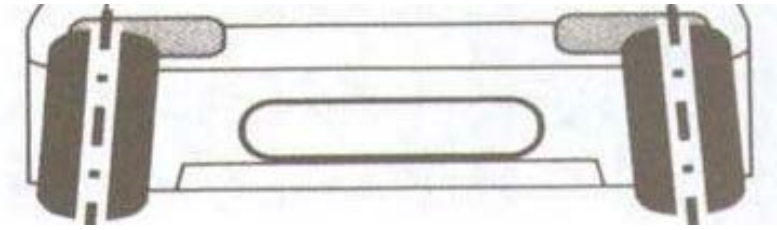

图 1.4　负上开倾角

由于摄像头车速度会很快，要求尽可能的在较快速度下转弯，而本届车模后置双驱动电机，加上电池位置在中后部位，使得车的整体重心靠后。在快速转弯时前轮抓地力不够，内侧前轮会有翘起情况，导致转弯比较困难，因此为了加大前轮抓地力，实现顺利过弯，本智能车系统采用负上开倾角。

1.1.3 后倾角

后倾角，就是指从侧面看，路面与转向销所形成的夹角。当转向销中心线垂直于路面时，后倾角最小。转向销上侧越往后倾，后倾角的角度越大，如图 1.5 所示。

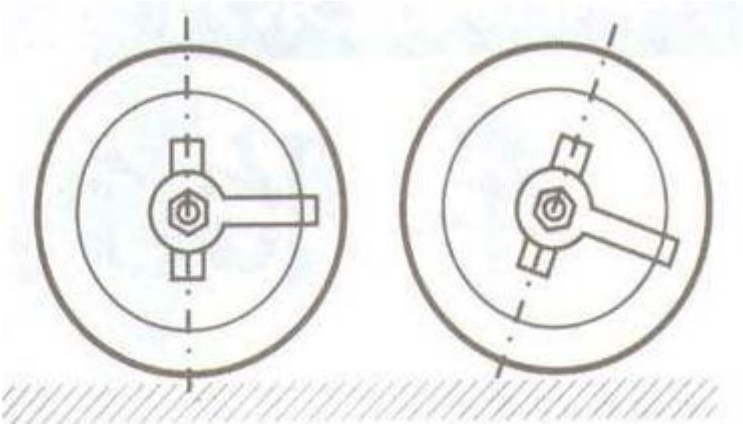

图 1.5　后倾角示意图

当后倾角的角度设定较大时，直线稳定性会变好。但是在弯道转弯时会产生前轮接地面积变小的情况，相反，将后倾角设定小的话，转弯的反应就会很灵敏。因此，转弯过度时，应将后倾角调大，转弯不足时，应将后倾角调小，这样就可以获得稳定的转弯特性。由于本智能车系统对转弯灵敏度要求较高，经反复试验后，最终将后倾角设置为最小。

1.2 CCD 传感器的安装

本智能车系统是依据摄像头采集的图像为依据行驶的，CCD 传感器采集图像的稳定性及图像包含的信息是智能控制策略的基础，而传感器的安装位置及角度则决定了采集的信息量的大小。

1.2.1 摄像头镜头的选择

不同型号的镜头的视角大小是不一样的，长焦镜头的焦距较长，成像较大，畸变较小，易于分析处理，当小车在直道或者小弯道上行驶时，可以很好的获得赛道信息。但

当跑道转弯角度较大时，由于视角小，会出现看不到赛道的情况，增大了道路信息判断的难度。而广角镜头则与长焦镜头相反，由于视角较大，广角镜头的成像较小，在弯道上可以采集到较多的赛道信息，但是在直道或者小弯道时，远处的赛道信息在图像中所占比例大大减少，分析赛道信息较为困难，即前瞻较短。在试验了多个镜头后，最终找到既能适当采集弯路信息，又能符合前瞻要求的镜头。

1.2.2 CCD 传感器的固定

对于智能车来讲，车体重量越小，速度控制越迅速，车体姿态调整也越快，整体性能也就越好，但是 CCD 数据采集要求尽量稳定也就是尽量减少摄像头的晃动，因此本智能车系统采用质量较轻但又符合系统的刚性要求的铝合金材料来固定 CCD 传感器。当摄像头位置处于智能车前面时车体近处视角过窄，转弯时采集赛道信息较为困难，但是由于车模本身的中心偏后，如果，摄像头后置又会加大前后重量差距，因此综合考虑后，本文将 CCD 传感器安装在智能车的中心位置，这样既符合采集要求，又减小了前后重量差距。

1.2.3 CCD 传感器的调节

固定好 CCD 传感器后，还需要对 CCD 传感器的高度和角度进行调节。摄像头位置越高，采集图像畸变越小，远处道路信息所占比例越大，但是过高会导致智能车行驶不稳定，严重时会导致翻车。角度可以调节摄像头的前瞻距离和智能车近处道路的采集盲区的大小，角度越小，前瞻距离越大，同时近处采集盲区越大。经过长时间试验，设计出可以调节 CCD 传感器角度和高度的摄像头固定装置，如图 1.6 所示，成功使 CCD 传感器采集图像符合系统要求。

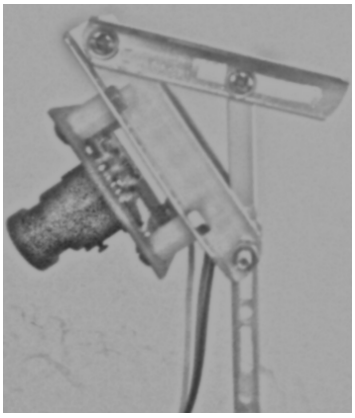

图 1.6　CCD 传感器固定装置

1.3 伺服器的安装设计

伺服器的安装方式决定了伺服器控制前轮转弯的连杆的长度，连杆长度越长，前轮转弯越迅速，但同时转矩会越小，反之，连杆长度越短，前轮转弯越缓慢，但同时转矩会越大。根据智能车转弯响应时间及力矩大小要求，本系统对伺服器的安装设计如图 1.7 所示。

图 1.7　伺服器安装设计

1.4 智能车重心调整

在满足各机械部件正常工作的前提下，应该考虑将智能车的整体性能调整至最好，其中主要调整为重心调整，重心对车性能影响主要为以下几个方面：

1、对动力性能的影响

汽车的驱动力必须大于等于坡度阻力、滚动阻力、空气阻力三者之和而等于汽车驱动轮的附着力。故重心位置必须保证驱动轮能够提供足够的附着力。仅从此方面考虑，重心越靠近驱动轴越好。

2、对转向性能的影响

重心前移，易发生后轴侧滑，而且增大舵机的转向力矩；重心后移，前轮摩擦力减小，丧失转向性能。

3、对稳定性能的影响

重心太高，汽车高速急转弯行驶时，会发生侧向倾覆，应尽量降低重心。同时，摄像头位置过低则会影响前瞻和图像畸变程度。

因此，智能车的重心调整在几何中心附近较为合适，尽量满足驱动、转向、稳定性、前瞻等的要求，并根据智能车实际行驶要求和情况进行适当调整。

第二章　智能车硬件电路方案设计

本系统的硬件电路采用模块化设计方式。主要包括单片机最小系统模块、电源模块、图像采集及处理模块、测速模块、电机驱动模块、无线通讯模块、显示模块等部分。其中，电源部分为整个系统稳定工作的基础，经过多次改进，最终确定了使用 3.3V 对单片机进行供电。此方案有效解决了系统各模块间电平匹配问题，同时也可以避免因为智能车加减速，造成电池电压瞬时降低，导致的单片机复位的问题。测速模块、电机驱动模块构成驱动电机闭环控制电路，实现电机转速的快速响应。在文献[1]和文献[2]的指导下，智能车的硬件电路模块均达到要求。智能车系统的硬件结构框图如图 2.1 所示。

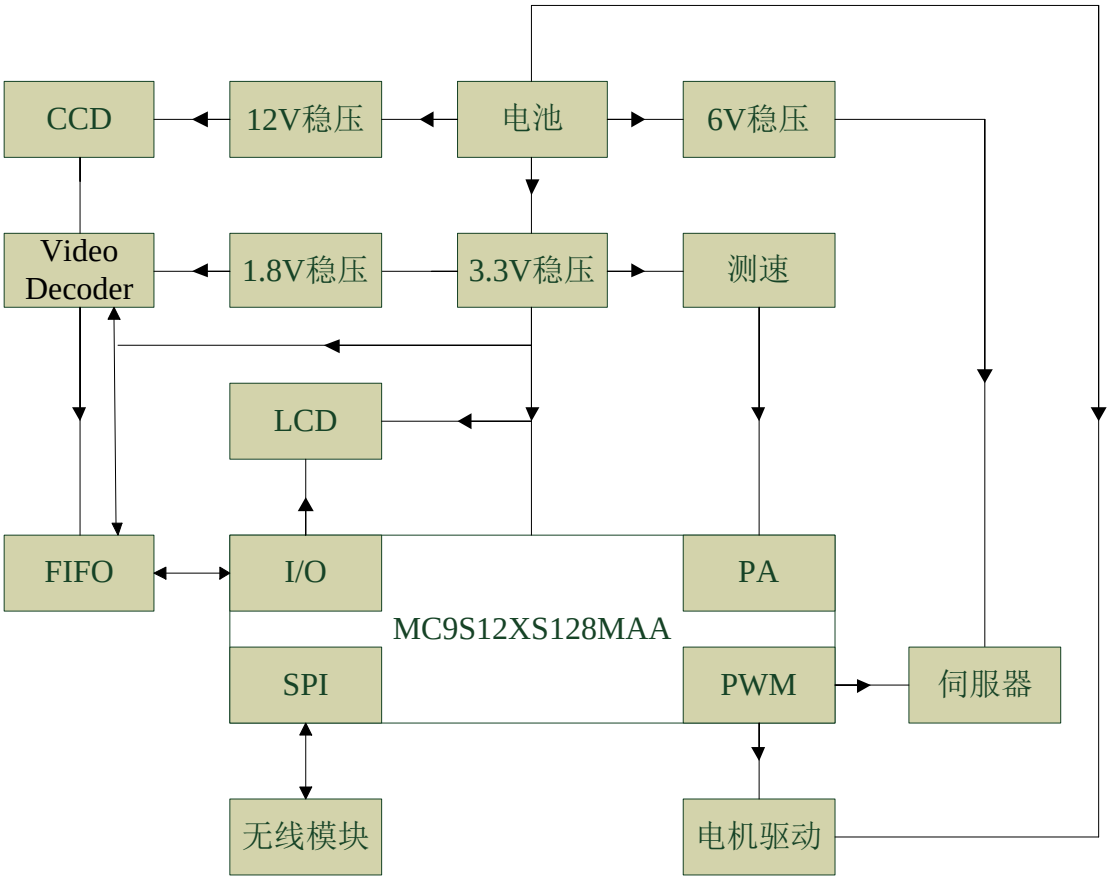

图 2.1　硬件结构框图

2.1　单片机系统板电路

本设计的核心控制器为飞思卡尔公司生产的 16 位单片机 MC9S12XS128。该单片机具有 80 引脚和 112 引脚两种封装，本设计采用 80 引脚封装。

MC9S12XS128 具有丰富的系统资源和方便的外部电路接口，其中包括 16 位中央处理单元，中断模块，RAM 存储器， FLASH 存储器，EEPROM 存储器，定时捕捉器，同步外设接口 SPI，异步串行总线接口 SCI，增强型捕捉定时器 ECT，可选精度的模数转换器 ATD，8 通道脉冲宽度调试器 PWM，离散数字 I/O 通道 A、B、K、E 等功能模块[3]。系统板电路模块如图 2.2 所示。

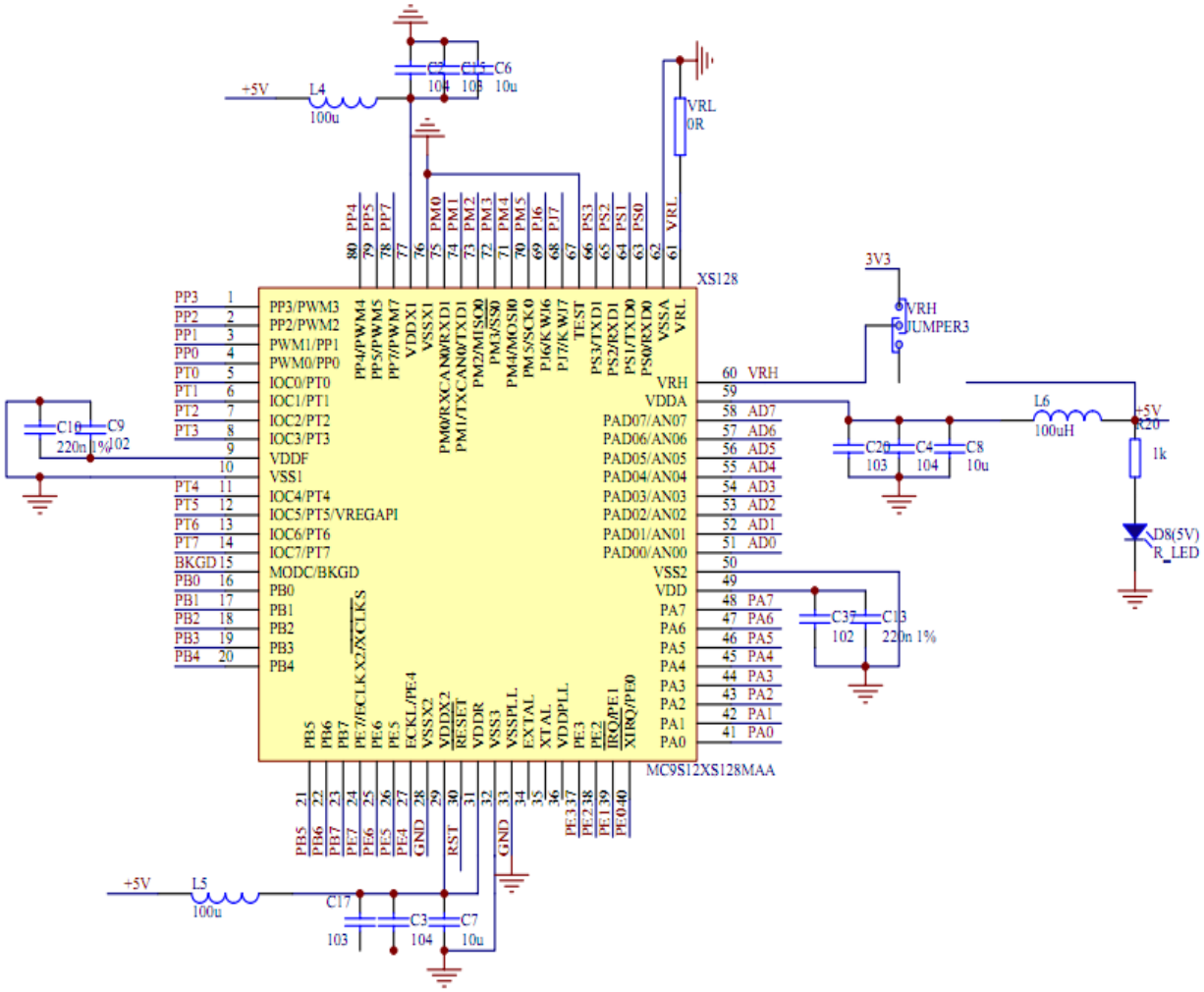

图 2.2　单片机系统板原理图

单片机系统版原理图包括晶振及 PLL 电路，复位电路，MCU 工作模式电路，BDM 电路以及扩展出的一些普通 I/O 与 PWM 等。其中，晶振电路对于整个系统的稳定性是至关重要的，本设计中使用的是 16MHz 无源晶振，无源晶振外围电路简单方便，而且价格也较便宜。通过锁相环可以将 CPU 时钟倍频到 60MHz，经过测试发现系统可以在此高频下稳定工作，这使得单片机指令执行周期缩短，大大改善了智能车的实时控制性能。

2.2 电源模块

电源模块是系统稳定工作的基础，因此，电源模块输出电压和电流的稳定性在整个智能车系统中起着非常重要的作用。

智能车的电池为 Ni-Cd(镍镉)电池，该类型电池具有高效的利用率和稳定的性能，一直被作为各种航模、电动车等的供电设备。为了获得最高的性能和最长的寿命，该充电电池必须以正确的方法来使用。

对镍镉电池充电时，通常采用电池容量值的大约两倍来充电，当电池是 1800mAh, 那么用 3 到 4 安培来充电是安全的，而且充得比较饱满。通常电流高，电池的爆发力会强些，但未必如电流低时饱满。同时我们也必须注意充电电流不能过高（>7A），当电流过高时，不仅不能提高电池性能，反而会损坏电池，严重时会导致电池起火、爆炸。根据经验，一般在 4.0 到 6.5 安培之间是效果较好。

电池充满电时，电压大约为 8V。在电池压小于 7V 时，应注意及时充电，电池过放会对其造成不可逆转的损害，电压低于 6V 会对电池造成毁灭性伤害。

2.2.1 电源升压电路

由于 CCD 传感器的额定工作电压为 12V，因此需要将电池电压升至 12V。

LT1946 是 LINEAR 公司 生产的高性能稳压芯片，其使用方便，通过改变外围电路电阻的阻值即可以调整输出稳压值。经过测试，当输入达到 2.1v 时，输出便可以稳定在 12V 左右，达到系统要求。因此，本系统采用 LT1946 作为电源升压芯片，电路原理图如图 2.3 所示。

经过升压，成功将系统输入电压稳定在 12V，并给 CCD 传感器供电，对整个系统的稳定起到了至关重要的作用。

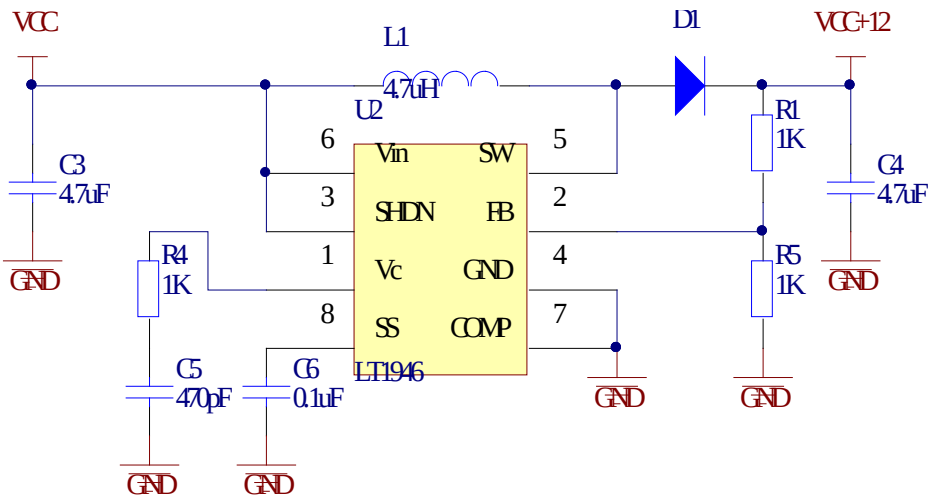

图 2.3　电源升压原理图

2. 2. 2　稳 3. 3V 电路模块

由于单片机的额定电压为 3.3V，同时电机驱动信号端、磁电编码器、CCD 信息处理等重要功能模块的电压均为 3.3V，因此，将电压稳定在 3.3V 并给各模块供电是必不可少的。电路的设计原理如图 2.4 所示。

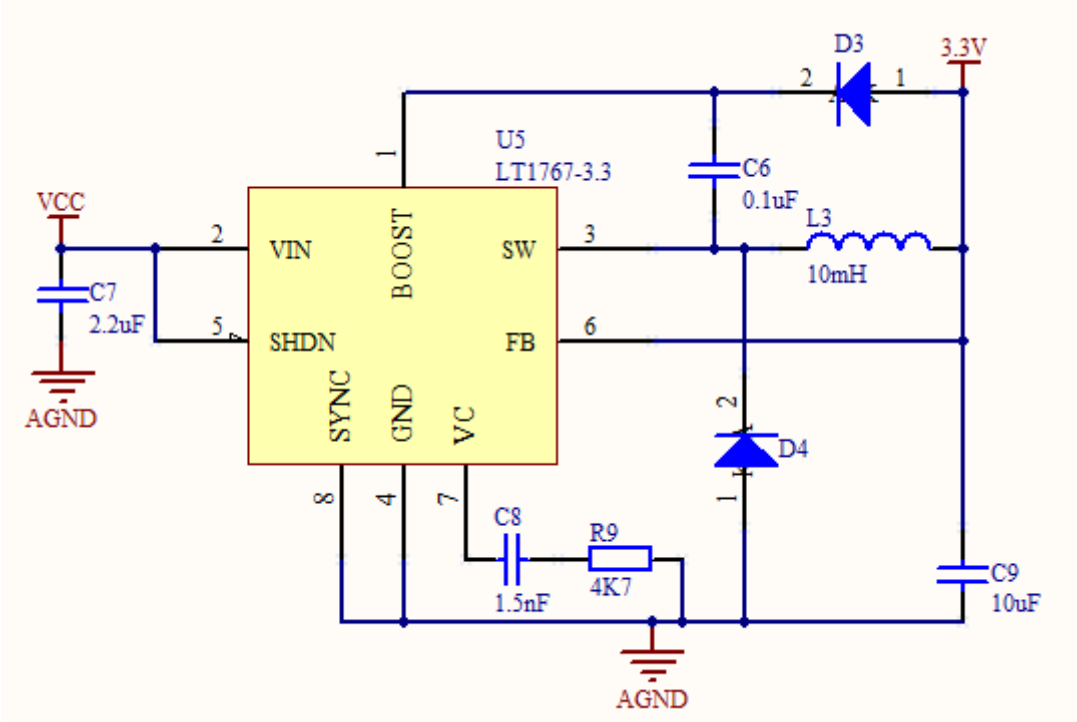

图 2.4　稳 3.3V 电路原理图

2.2.3 稳 6V 电路模块

由于伺服器的额定工作电压为 4~6V，而智能车在实际行驶过程中，转弯需要的转矩较大，为了保证伺服器的准确快速响应，本系统采用 6V 给伺服器供电，电路原理如图 2.5 所示。

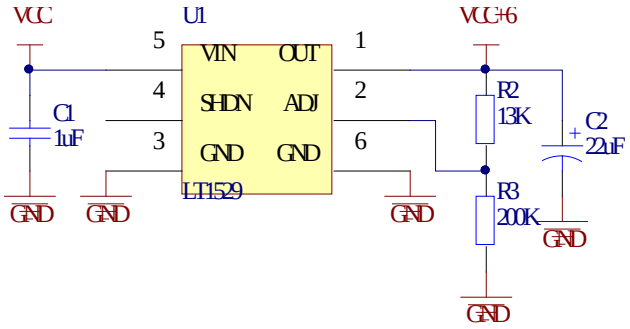

图 2.5　稳 6V 电路原理图

2.3　CCD 数据处理模块

微型飞行器空中侦察关键技术是为微型飞行器的应用领域而开展研究的，微型飞行器的主要作用是侦察、 电磁干扰和攻击，但这些应用的基础是侦察，没有前期的侦察结果就没有后期的电磁干扰和直接攻击。因此，空中侦察是微型飞行器的主要应用成果。空中侦查常用的是 CCD 相机拍摄。目前 CCD 电荷耦合器件是主要的实用化固态图像传感器件，它具有读取噪声低,动态范围大，响应灵敏度高等优点[4]。

CCD 传感器输出的信号为视频混合信号即全电视信号，本系统采用 TVP5150AM1 芯片对混合信号进行解码，分离出行同步信号、场同步信号等，并通过外置 FIFO 缓存数据。

2.3.1　CCD 传感器输出信号介绍

CCD 传感器直接输出的信号为彩色全电视信号，除与黑白全电视信号相同含有亮度、 复合同步、 复合消隐、 均衡等脉冲信号外，还含有彩色信息的色度信号和保证彩色稳定的色同步信号。因此，CCD 传感器输出信号具有如下特点：

1、参于混合的各种信号均保持着独立性，也就是说，可用各种方法将它们一一分离。 例如色度与亮度信号在时域重叠而在频域交错，色度与色同步在频域重叠而在时域交错，扫描用的同步与消隐信号在频域、时域均重叠，但在电平高低上有区别，它们与图像信号在时域交错，互不干扰。

2、它是视频单极性信号，既有直流成分，又含有交流成分，且是上下不对称的信号，占有 0~6MHz 的频带宽度。

3、对静止的图像而言，其电视信号以帧为周期重复，其场间、行间相关性也较大。对活动图像而言，则可说是帧间、 行间相关性较大的非周期信号，但其同步与消隐信号仍是周期性的。

传统的彩色全电视信号分离框图如图 2.6 所示。

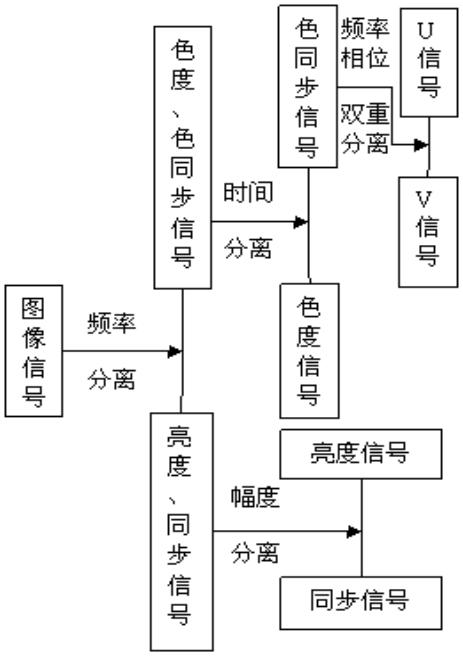

图 2.6　彩色全电视信号分离框图

2.3.2　CCD 信号分离电路

由于 CCD 传感器输出信号为混合信号，因此需要对其信号进行分离，将图像进行还原。本智能车系统只需要采集道路上的黑色引导线信息，因此，将信号分离后，只需利用灰度值即可确定跑道信息。

TVP5150AM1 可以从 0.5～2V 的标准负极性 NTSC 制、PAL 制、SECAM 制视频信号中提取复合同步、场同步、奇偶场识别等信号，也能对非标准的视频信号进行同步分离，通过固定的时间延迟产生默认的输出作为场同步输出。电路原理图如图 2.7 所示。

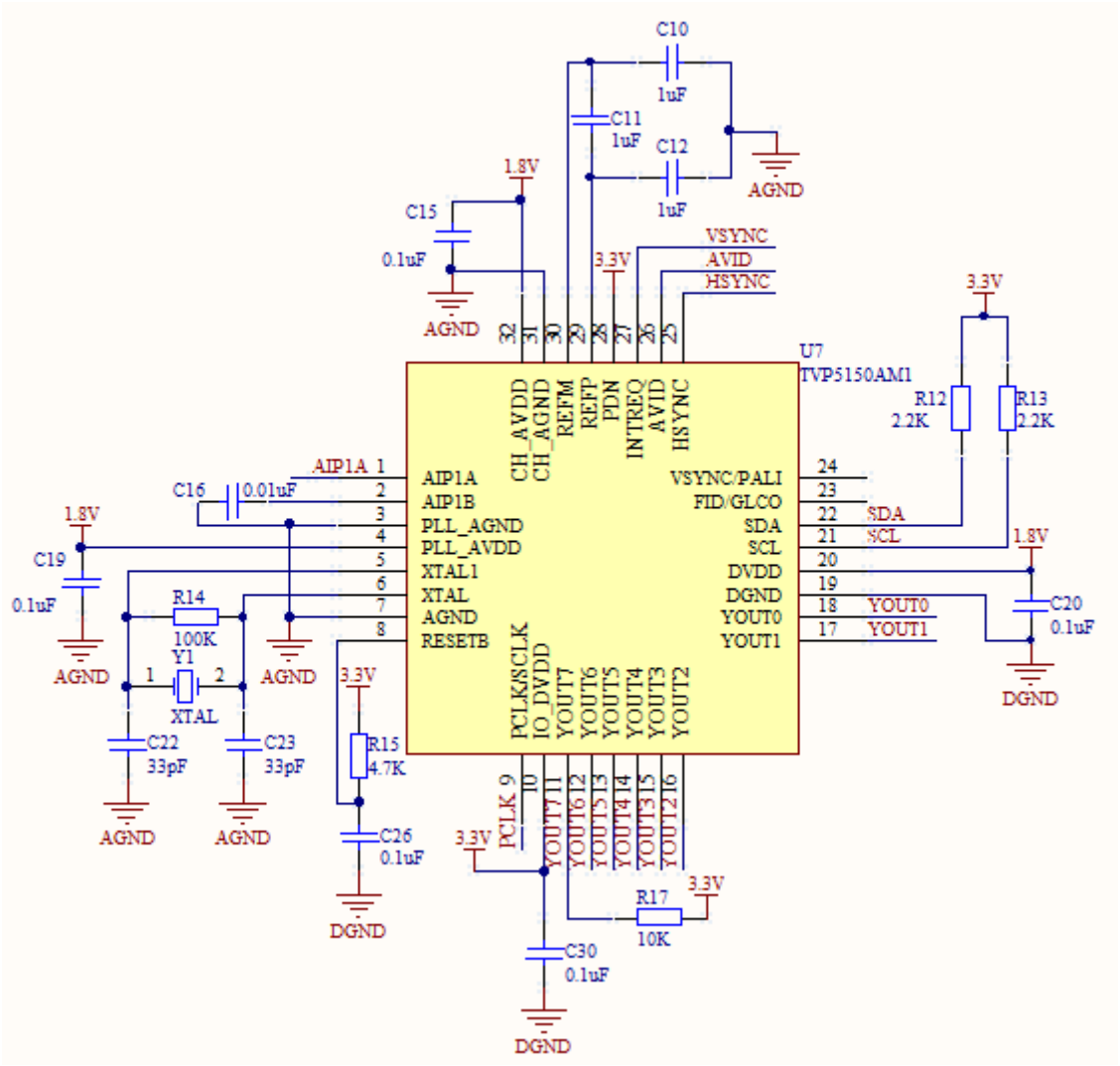

图 2.7　CCD 信号分离电路原理图

2.3.3　FIFO 电路

本设计采用 AL422B 对解码后的数字信号进行缓存，电路图如图 2.8.

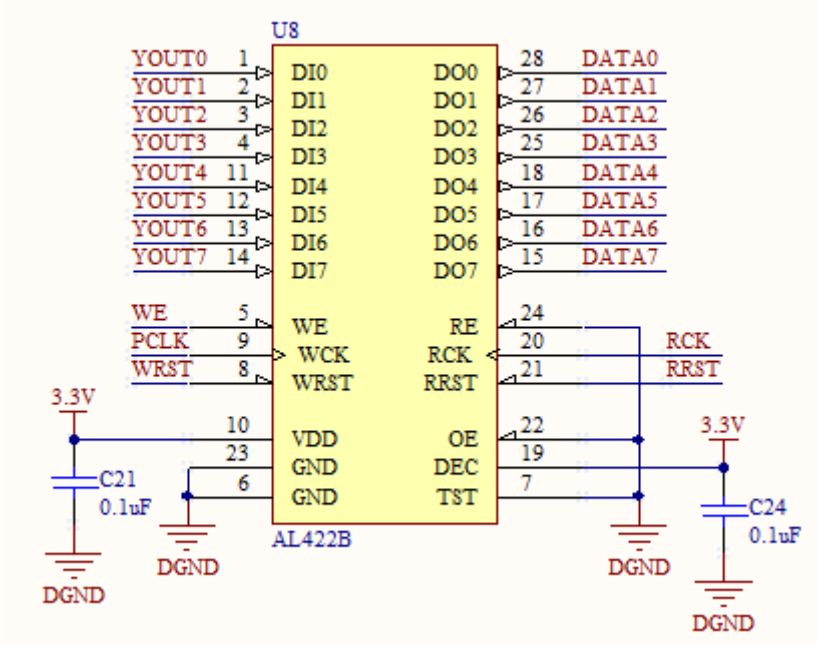

图 2.8　TLC5540 电路原理图

2.4　电机驱动模块

2.4.1　H 桥电路模块

本智能车系统车模的电机型号较小，对电机驱动的输出电流的要求并不苛刻，因此本设计的驱动电路由 2 片 BTS7960 构成 H 桥。通过控制 4 个 MOS 管的导通和关断来实现正反转，并通过控制输入的 PWM 波的占空比来调节电机两端的平均电压，达到控制电机的转速的目的，具体电路图如图 2.10 所示。

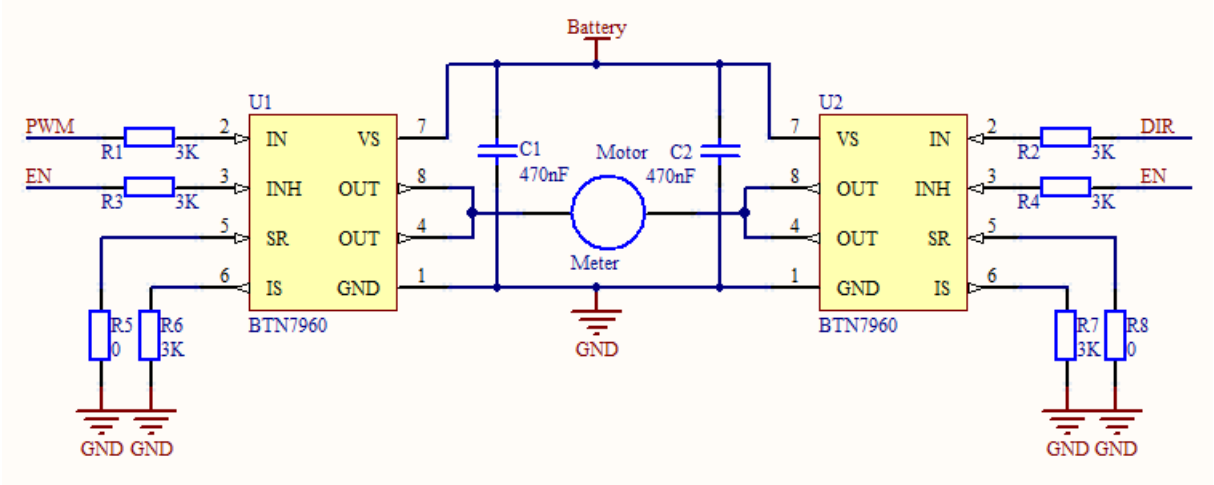

图 2.10　电机驱动原理图

2.5 无线通讯模块

2.5.1 无线通讯模块简介

nRF24L01 是一款工作在 2.400 - 2.4835GHz 世界通用 ISM 频段的单片无线收发器芯片。无线收发器包括：频率发生器、增强型 SchockBurst™ 模式控制器、功率放大器、晶体振荡器、调制器、解调器[5]。

nRF24L01 的无线电前端使用 GFSK 调制。它具有用户可配置的参数，如频率通道，输出功率和空中数据速率。nRF24L01 可配置实现高达 2Mbpsde 空中数据传输速率。其内部稳压器可确保高电源抑制比（PSRR）并支持宽电源电压范围。其内部结构框图如图 2.11 所示。

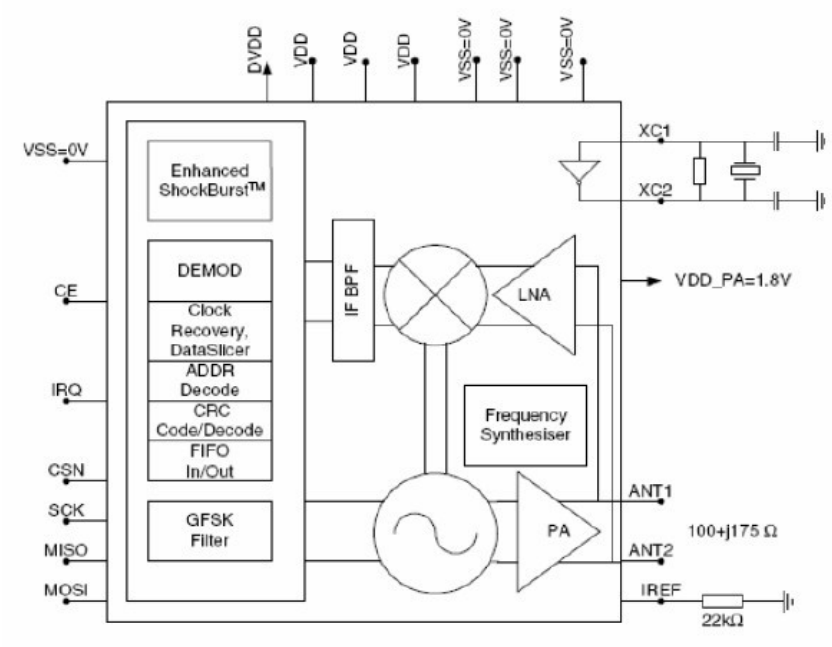

图 2.11　nRF24L01 内部结构框图

通过配置寄存器可将 nRF241L01 配置为发射、接收、空闲及掉电四种工作模式。工作模式由 PWR_UP register、PRIM_RX register 和 CE 决定，具体配置方式见表 2.1。

表 2.1　nRF241L01 工作模式配置

MODE	PWR_UP register	PRIM_RX register	CE	FIFO state

RXmode	1	1	1	-
TXmode	1	0	1	Data in TX FIFO
TXmode	1	0	1—>0	Stays in TX mode until packet Transmission is finished
Standby_2	1	0	1	TX FIFO emrty
Standby_1	1	-	0	No ongoing packet transmission
Power Down	0	-	-	-

（1）发射、接收模式

收发模式有三种：ShockBurst™ 收发模式、增强型 ShockBurst™ 收发模式和直接收发模式。本设计主要采用的是增强型 ShockBurst™ 收发模式。在此模式下，系统工作更加稳定并且具有自动应答功能。

增强型 SchockBurst™ 模式可以使得双向链接协议执行起来更为容易、有效。典型的双向链接为：发送方要求终端设备在接收到数据后有应答信号，以便发送方检测有无数据丢失。一旦数据丢失，则通过重新发送功能将丢失的数据恢复。增强型 SchockBurst™ 模式可以同时控制应答及重发功能而无需增加微控制器的工作量。

在增强型 ShockBurst™ 收发模式下，数据是使用节能方式进行发射，数据低速进入微处理器，高速发射，发射时使用片内的先入先出堆栈区，如此，该模块虽然只使用了低速的微控制器，但是却可以得到非常高的射频数据发射速率。此外，为了尽量节能，并降低系统费用，该模块使数据在空中停留时间短，抗干扰性高与射频协议相关的所有高速信号处理都在片内进行。增强型 ShockBurst™ 技术同时也减小了整个系统的平均工作电流。nRF24L01 功耗极低，当发射功率为-6dbm 时，工作电流只有 9mA,接收时工作电流亦只有 12.3mA。

当 nRF24L01 进入增强型 ShockBurst™ 收发模式时,nRF24L01 将自动处理字头和 CRC 校验码，与此同时，在数据接收端，nRF24L01 自动把字头和 CRC 校验码移去。当发送端发送数据时，单片机通过 SPI 将数据写入 nRF24L01 的 MOSI 引脚，nRF24L01 自动加上字头和 CRC 校验码，置 CE 为高，等待一段延时（约大于 10us），至此发送过程完成。

增强型 ShockBurst™ 发射流程：

A. 初始化 nRF24L01，将接收机的地址和要发送的数据送入 nRF24L01；

B. 初始化 CONFIG 寄存器，使 nRF24L01 进入增强型 ShockBurst™ 模式。

C. 把 CE 置高，激发 nRF24L01 进行高速数据发射，该过程有模块内部自动完成；

D. nRF24L01 的增强型 ShockBurst™ 发射具体流程: a.给射频前端供电；b.射频数据打包(加字头、CRC 校验码)；c.高速发射数据包；d.发射完成，nRF24L01 进入空闲状态。

增强型 ShockBurst™ 接收流程：

A. 初始化，设置本机地址和要接收的数据包大小；

B. 初始化 CONFIG，使之进入接收模式，置高 CE。

C. 130us 后，nRF24L01 监视数据包的到来；

D. 当接收到正确的地址和 CRC 校验码时，nRF24L01 将自动把字头、地址和 CRC 校验位移去；

E. nRF24L01 通过把 STATUS 寄存器的 RX_DR 置位(STATUS 一般引起微控制器中断)通知微控制器。微控制器把数据从 nRF24L01 读出；

F.当将数据取出后，清除 STATUS 寄存器。

（2）空闲模式

nRF24L01 的空闲模式是为了减小平均工作电流而设计，其最大的优点是实现节能的同时，缩短芯片的起动时间。在空闲模式下，部分片内晶振仍在工作，此时的工作电流跟外部晶振的频率有关。

（3）掉电模式

在掉电模式下，为了得到最小的工作电流，一般此时的工作电流为 900nA 左右。关机模式下，配置字的内容也会被保持在 nRF24L01 片内，这是该模式与断电状态最大的区别。

综上，nRF24L01 是一款性价比很高的单片无线收发芯片，其内置了 CRC 纠检错硬件电路和协议，数据传输更为可靠。由此选择 nRF24L01 芯片作为无线模块的收发芯片。

2.5.2 无线通讯电路设计

无线通讯电路主要有单片机和 nRF24L01 的集成模块组成，nRF24L01 片内具有 SPI 模块，可以与 S12 系列单片机的 SPI 直接通信。由于 S12 系列单片机的 I/O 输出电压为 5V，因此需要在 I/O 与无线模块之间串接一限流电阻，具体电路原理如图 2.12 所示。

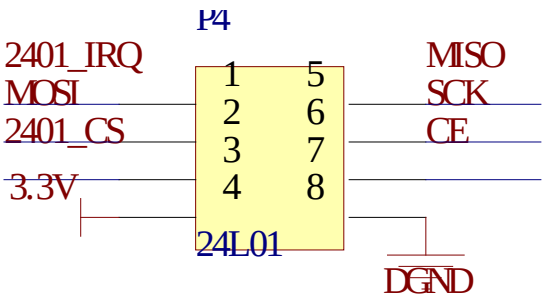

图 2.12　无线通讯模块电路设计

上位机电路通过 MAX232 与 PC 的 USB 口相连，实现智能车与上位机的数据传输。MAX232 芯片是美信公司专门为电脑的 RS-232 标准串口设计的单电源电平转换芯片,使用+5V 单电源供电，因此，本文并未对无线模块单独设计电源模块。

在硬件电路设计中，电源管理电路是各个模块稳定工作的基础，在电源电压由于负载变化而发生剧烈变化时，该电源管理模块能够稳定输出电压，保证各模块的正常工作。电机驱动模块通过 6N137 进行光电隔离，成功隔离了驱动部分对控制部分的影响，保证了驱动电机的精确且稳定控制。

3 智能车图像信息处理

智能车采集图像信息的底层处理算法是整个上层控制策略的基础，图像采集的稳定性、图像处理方法、路径识别的准确性都决定着上层控制策略能否发挥作用，只有准确的识别出路径信息，智能车才能实现高速稳定行驶。本部分采用 C 语言编写程序，在文献[6]和文献[7]的指导下成功实现了智能车的图像信息处理。

3.1 图像采集

3.1.1 图像数据简介

摄像头的主要工作原理是[8]：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输出。当摄像头扫描完一行，视频信号端的输出有一个电压 “凹槽”，并保持一段时间，此 “凹槽” 叫做行同步脉冲，它是扫描换行的标志。跳过一行后，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。在这若干个消隐脉冲中，有个脉冲远宽于其他的消隐脉冲，该消隐脉冲即称为场同步脉冲，它是扫描换场的标志，如图 3.1 所示。

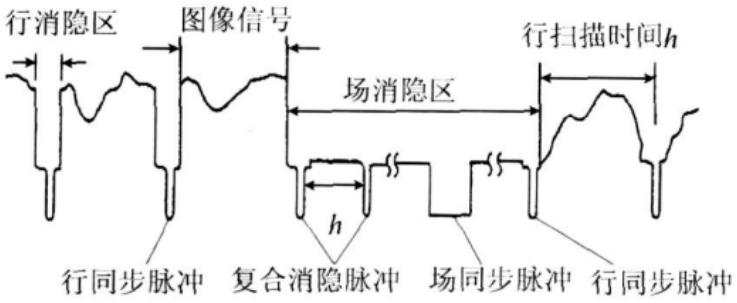

图 3.1 摄像头视频信号

CCD 采集的彩色全电视信号经过 LM1881 芯片信号分离后，可以得到场同步信号、行同步信号和图像灰度信息，其中图像灰度信息通过模数转换后可以直接输入到单片机 I/O 接口，这样单片机就可以读取图像信息进行处理了。

3.1.2 单片机读取图像信息方法

本智能车系统采用的 CCD 传感器输出为 PAL 制信号，分奇偶场输出，由于奇场和偶场所呈现的灰度值图像的效果是相同的，因此本设计只读取其中一场信息即可得到路径信息。经过测试发现摄像头每秒可以扫描 25 幅图像，每幅图像又分为奇场和偶场，

先奇场后偶场，这样相当于每秒扫描 50 场图像，由此可知扫描周期为 20ms[9]。直读取其中一场数据时，单片机就可以将程序周期设定为40ms，前20ms用于图像信息的读取，后 20ms 用于处理数据和控制策略的实现。

由于每场图像的数据量非常庞大，单片机总线的速度远远慢于 CCD 传感器发送速度，即使采取只读取奇偶场中的一场的方法，将待读取数据量减少为原来的 1/2，单片机同样无法全部将数据读取进来。基于上述原因，本系统设计了间隔读取的方法。

经过测试，当单片机通过锁相环将 CPU 时钟倍频到 60MHz 时，单片机每行最多可以读取 296 个像素点，每幅图像读取 270 行时，获得分辨率为 296*270 的图像。

单行像素点读取方法如下：

由于像素同步信号频率高于单片机倍频后的频率，因此本系统对每行像素点采取等时间间隔读取的方法，即当行中断到来后每隔一个时间就读取一个像素点，直到下一个行中断，这样，每行的数据就相当于按比例放入单片机存储区中。

行读取方法如下：

行同步信号的频率是低于单片机倍频后的频率的，因此理论上我们可以收到每一行的同步信号，而不像读取每行像素点那样必须要丢弃一部分点，但是由于单片机的存储空间有限，无法存储整幅图像，因此行读取也应采用间隔读取方法。

经过试验发现，当行数据按照读取像素点的方法读取时，即每隔 4 个行同步信号读取一行时，由于摄像头畸变，会发生如下现象：在离摄像头近处的区域，读取的数据所占比例很大，而远处区域所占比例很小。而实际在智能车行驶过程中，车近处的道路信息是比较稳定的，而变化大的是远处区域，这与采集情况恰恰相反，这样在远处区域要用较少的点表示较大的变化，使得远处每个的代表跑道信息的点间隔都很大，与噪点的分布类似，造成远处信息无法利用的情况，相当于减少了智能车的前瞻距离，而前瞻少对于高速度行驶的摄像头智能车来说是致命的缺点，会严重影响车的速度和行驶稳定性。

本设计系统创新性采用等距离采行的方法。每隔 3 厘米采一行数据，这样既可以保证远处区域信息的数量，又消除了摄像头纵向压缩畸变的影响，经过实际测验后，我们发现很难将起跑线信息检测出来，原因是起跑线宽度为 2.5 厘米，小于每行的间隔，这样，起跑线很可能处于两行数据的中间，导致无法采集到起跑线信息。即使采集到其信息，也只是位于一行之中，造成数据的不稳定。经过研究后，我们在智能车近处增加了采集信息的行数，成功解决了这一问题，使得等距离间隔采行法得以成功应用，并通过路径对比体现出其优势。

等距离间隔采行法的具体实现方法如下：以间隔 3 厘米为例，将智能车放一条长直标准跑道上，开启采集图像模块，从第一行开始，将距离第一行 3 厘米后面的跑道部分

用白板遮住，看采集到的黑线最终一行在图像中的行数，并记录下来。再将白板后撤 3 厘米，找到黑线最后一行在图像中的行数，记录下来。以此类推，直到白板达到要求的前瞻位置，这样所有记录的行加上判断起跑线需要读的行即为单片机需要读取的行。

等距离间隔采行法的程序流程图如图 3.2：

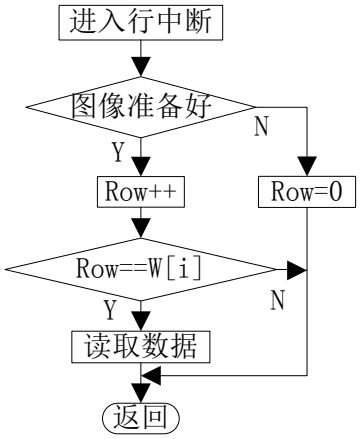

图 3.2　等间隔采行法

图中 Row 为行同步信号计数器，W[i]数组存放需要读取的行数，包括记录的隔 3 厘米采行的行数和辅助判断起跑线行数，并在其中按升序排列。

3.2　图像处理

由于图像处理是路径识别的基础，也是控制策略能否发挥作用的重要影响因素，因此，本文将对此部分进行详细介绍。

3.2.1　阈值的确定方法

在数字化后的图像数组中，较暗的部分数值较小，亮的部分数值较大，而智能车跑道上的黑色引导线和白色的跑道对比是非常明显的，数值差也较大。根据这一点，我们可以将图片的每一行与适当的数即阈值比较，当低于阈值时，便可以认为该点为黑色点，反之为白色点。

阈值分为静态阈值和动态阈值两种。静态阈值是指将阈值设定为固定值，在车行驶过程中，阈值不发生变化。这种方法的优点是计算简单，只需作比较即可，节省运算时间。缺点是环境适应性差，一旦换一种环境或者车行驶在光线变化较大的场地中，固定的阈值就可能不再合适，需要手动调整。动态阈值是指阈值可以随车的行驶环境进行动态调整，以适应环境的变化。这种方法的优点很明显，环境适应性很强。缺点是需要实

时计算阈值，增加了计算量。由于本实验室光线变化较明显，而且经过测试，整个算法一个周期占用时间大约为 4ms，可以加入计算阈值算法，因此本设计采用动态阈值。

由于摄像头的前瞻一般大于 1.5 米，因此，整幅图像包含的区域较大，在一幅图像中，明暗变化可能会比较明显，此时如果整幅图像用一个阈值就会出现不合适的情况。本设计每行都进行阈值调整，根据本行信息确定阈值。

计算阈值的方法为遍历整行，找出最大值和最小值，需要注意的是最小值不能低于正常数值范围，否则是噪点或同步信号，应跳过，然后对最大值和最小值求平均值，将此平均值作为该行的阈值。

3.2.2 二值化

为了减少计算量，降低数据处理量，节省预算时间，更加直观的提取出路径信息，本设计对图像数据每一行按照相应的阈值进行二值化，小于阈值时置 1，表示其为黑色，否则置 0，表示为白色。程序流程图如图 3.3：

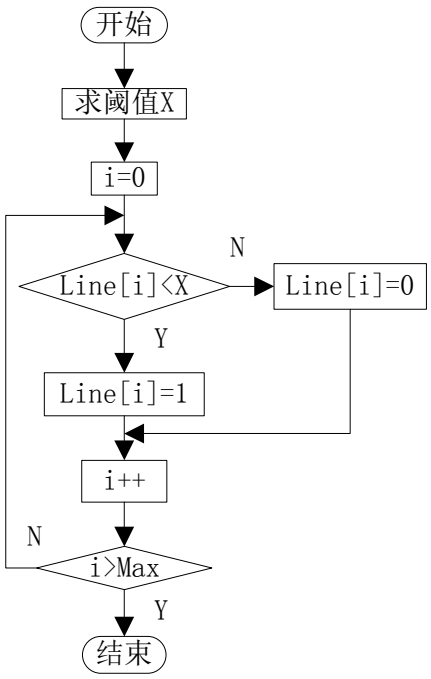

图 3.3 单行二值化程序流程图

上图中 i 为计数作用，表示当前处理的数值在本行数组中的位置。Line[i]数组存放本行数据信息。X 为本行阈值。Max 等于本行数据数量减 1 。

3.2.3 边沿化

处理图像的最终目的是识别出路径信息，这就要求我们找到黑线的左右边沿，这样才能计算出黑线的中心位置。为了更快的找到黑色引导线的边沿，本文设计了路径的边沿化方案。

一般的边沿化方法为从第一个数开始与后面相邻的数进行比较。如果相等，则下一个数与后面相邻的数比较。如果不相等，则将此数设置为 100，然后再下一个数与后面相邻的数比较，如此循环，直至此行遍历结束。流程图如图 3.4：

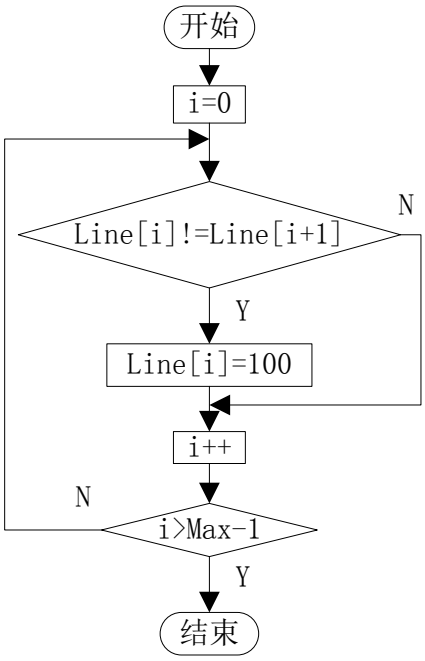

图 3.4　普通单行边沿化程序流程图

上图中 i 为计数作用，表示当前处理的数值在本行数组中的位置。Line[i]数组存放本行数据信息。Max 等于本行数据数量减 1 。

在实际应用中，我们发现当远处黑线在数组中只占一列时，上述边沿化方法只能处理出两个沿，而丢失了黑线信息，致使前瞻减小。当前瞻大于 1.3m 时，后面采集的黑线数据均只占一列，这样导致了大量的赛道信息丢失。

本文改进了边沿化算法，在进行边沿化时，只将与黑色相邻的白色点设置为边沿，而黑色点不变。这样，当只有一个黑色点时，其两边是沿，中间还是黑色点，有效解决了上述问题，经过对运行结果的观测，该算法可以将一个黑色点的赛道信息准确识别出来，很大程度上提高了前瞻距离，为路径信息的优化提供了较大助力，算法流程图如图

3.5 所示。图中 i 为计数作用，表示当前处理的数值在本行数组中的位置。Line[i]数组存放本行数据信息。Max 等于本行数据数量减 1 。

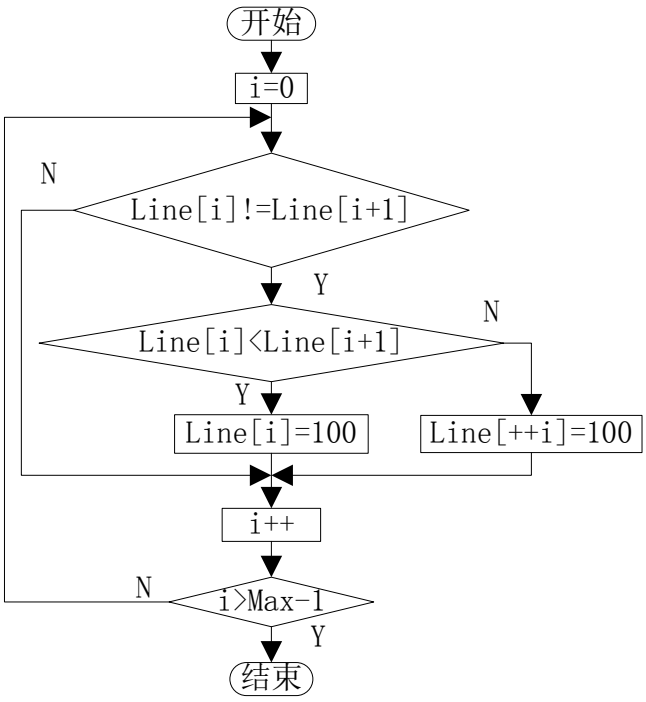

图 3.5　优化单行边沿化程序流程图

3.3　路径识别

路径信息是整个智能车工作的软件基础，是上层决策系统的最重要依据，路径识别的准确性关系到整个系统的控制策略的验证，本文对此部分非常重视，将进行重点论述。

3.3.1　路径识别算法总体设计思想

一般在图片中寻找颜色信息可以遍历整张图片，直到找到全部信息，这种方法适用于待找信息几乎无规律分布的情况。当图像数据像素较大时，每个像素点都去遍历会耗费大量的时间，本智能车系统每幅图像只有 20ms 的处理时间，这样大规模的挥霍时间是本系统无法承受的。仔细分析图像信息以及我们所要提取的路径信息就可以发现，在一幅图像中，每条黑色引导线都是连续的，而黑线两边的点对我们毫无用处，我们只需知道黑线的中心位置即可。因此，本系统根据这一点，设计出专用于寻找黑线中心位置的算法方案。

由于黑线的连续性，我们只需要找到起始行的黑线位置，确定出第一行的黑线中心点，而下一行的黑线必位于上一行黑线中心附近。这样我们处理一行数据时，只要在上一行的黑线中心附近找就可以了，而不必再去遍历整行，不断的按上述方法寻找下去，直到找到本图像中的所有黑线中心。这样不但提高了算法的效率，还提高了路径识别的准确性，排除了赛道外可能存在的黑色区域的干扰。

3.3.2 基准行的确定

确定出总体的算法思想后，通过实际测试我们发现此算法还有较大缺点，当找不到甚至找错起始行时，整个算法就无法执行下去了，有时甚至会误导智能车冲出跑道。

为了确保起始行的可靠性，本文确定了以下解决方案：当一幅图像读取完毕时，对第一行进行求阈值、二值化、边沿化处理，然后判断边沿的个数，如果边沿过多说明此行的干扰过多，由于基准行的准确性极其重要，我们将这一行舍弃，继续对第二行进行上述处理。以此类推，直到找到边沿符合要求的行，此时该行的干扰是比较少的。接下来寻找间隔在黑线范围内的两个沿，这两个沿很可能就是黑线的边沿，通过检测两个边沿的中心是否为黑色点来最终确定黑线位置。如果是，则此行的黑线中心为两个沿的中心，否则继续向后寻找，直到找到符合要求的沿。

这样寻找出的黑线中心在一般情况下是正确的，但在实际运行中，赛道上可能会有黑斑或者光线会造成干扰，使得某些行恰好出现符合要求的边沿，这样，我们所找的起始行就不准确了。

考虑到环境的这些干扰因素，本智能车系统加入了起始行的稳定性判断，在找到符合要求的起始行后，从起始行的中心附近向下一行寻找，如果有连续 4 行能找到黑色引导线，那么就认为该起始行是正确的，否则放弃该起始行，继续寻找新的起始行，直到找到符合要求的行。至此，起始行的寻找结束，将此起始行作为图片的基准行进行路径识别。程序流程图如图 3.6 所示。

图中 i 表示图像中当前处理的行是为第 i 行，S 为黑线搜索范围，F 为找到第一个符合要求的行的标志，F=1 表示已经找到起始行，C 为起始行稳定性判断计数器，P[i]数组中存放路径信息，即 P[i]中存放的是图像中第 i 行的黑线中心位置，Base 为基准行的行数。“对 i 行中列值在 S 范围的数进行处理”语句是指对图像数据矩阵的 i 行数据中，列值处于 S 的数值范围的数进行求阈值、二值化和边沿化处理。

经过实际试验，上述基准行的确定方案是准确可行的，该算法成功将干扰信息过滤掉，确定出基准行的准确位置，为后面的路径信息提取打下了可靠的基础。

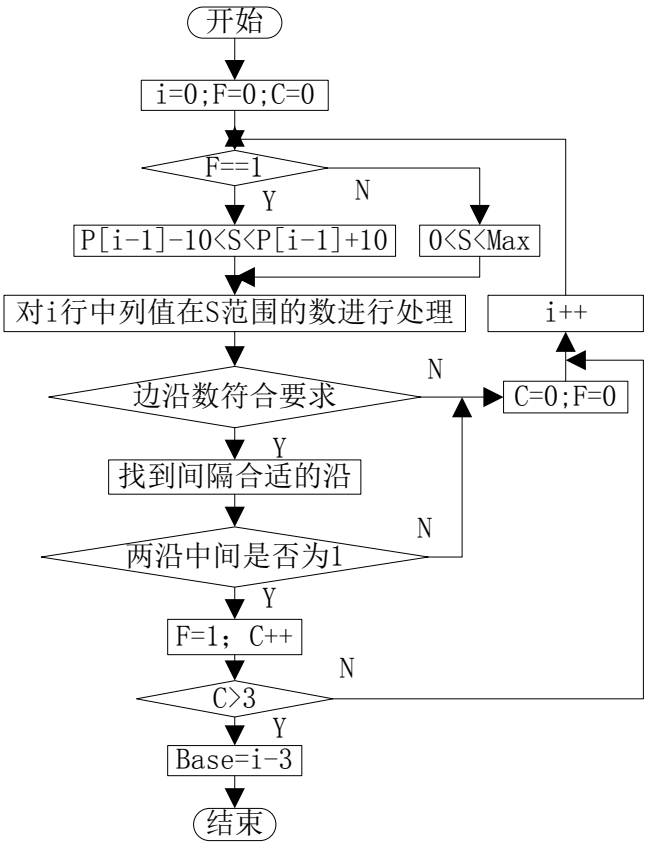

图 3.6　基准行识别算法流程图

3.3.3　路径识别算法

有了准确可靠的基准行的位置，后面识别算法就简单可行了，从基准行不断地向后寻找，那么路径识别算法的主要问题就在于如何确定找到黑线终点了。

在此过程中，需要注意的是，黑线并不一定是越靠后越窄的。在直道上时，黑线越靠后越窄，而当前面有弯路时，用图像采集卡可以看出后面的弯路集中分布在几行当中，这样黑线就会较宽，甚至宽于基准行黑线，这种现象是由于摄像头图像畸变造成的，因此寻找黑线的搜索范围应该适当定义。

黑线终点的确定一般可以是当某行开始找不到黑线时就将此行作为黑线的终点行，但是在实际的图像采集过程中，由于 CCD 外围电路的精确性原因，偶尔会发生将同步信号作为数据的情况，这样，该行整体就会偏移一定的列数，导致在该行无法找到黑线，但同时此行后面的行的数据是正确的。因为某一行的采集出现失误而丢弃后面的路径信息是不科学的，也是不合理的，为了充分利用这些数据，本智能车系统参照上面寻找基

准行的思想，设计了终止行的稳定性判断。，即只有在连续 3 行找不到黑线时才认为搜索结束，找到黑线终点所在的行。算法流程图如图 3.7 所示。

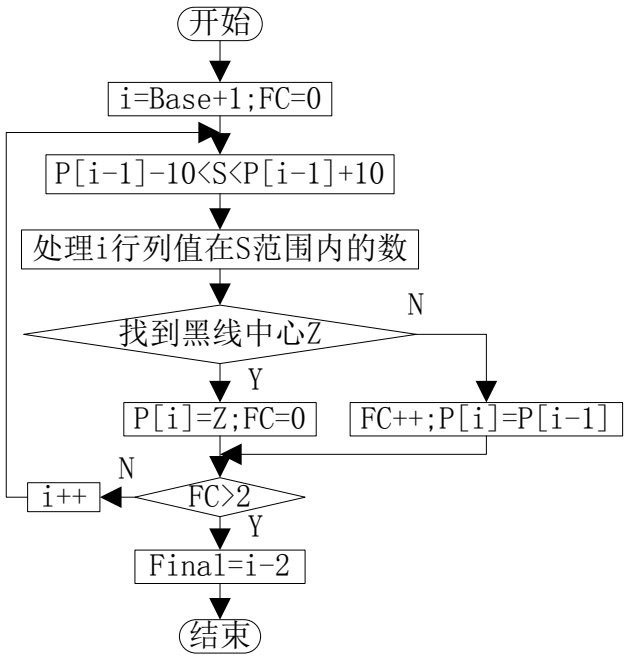

图 3.7　路径识别算法流程图

图中 Base 为黑线基准行的行数，FC 为终止行稳定性计数器，P[i]存放的为路径中心位置信息，i 表示图像中当前处理的行是为第 i 行，Final 为黑线的终止行。

通过上述算法，我们成功避过了干扰行，并且发现这种算法可以用来处理赛道的虚线部分，同时还可以解决光线、赛道整洁度等环境因素对黑线终点的判断，通过上述识别算法，经过实际测验，本智能车系统能够在光线变化较大、路面杂物较多等环境干扰较大的情况准确识别出路径信息，并能够在虚线部分识别出完整的路径信息。达到系统要求。

3.4　图像消畸变

3.4.1　消畸变方案的确定

摄像头的采集范围是一个近似扇形的区域，把这个区域的信息以数据矩阵的形式传输给单片机，远处较大的区域信息需要进行压缩，而近处的区域信息则进行扩伸。由此可知单片机收到的数据坐标实际上是经过了变换后的，要得到实际坐标，则需将图像坐标还原到世界坐标。

坐标还原的方法有很多种，最常见的是将图像坐标乘以一个简单的经验系数，这种方法简单易行，但是很明显数据还原的很不准确，有可能造成较大的偏差。第二种就是公式法，根据实际坐标与图像坐标的关系，通过某种途径计算出映射函数，然后处理数据时按此函数进行还原。这种方法比较科学，还原的坐标偏差较小，但是还原函数一般较为复杂，往往需要进行乘除法运算，增加了单片机运算时间。第三种为查表法，即将图像中每个像素点代表的世界坐标都实际测出并放于数组中，通过查询图像坐标在数组中对应的世界坐标来还原数据，这种方法算法简单，单片机处理速度快，但是实际操作困难，尤其是当图像像素点较多时，制作表格会浪费大量的时间。

经过多重比较，本设计系统将第二种与第三种方法结合使用，先将还原函数计算出来，然后根据还原函数利用上位机生成还原表，并存储在单片机的 FLASH 存储器中，这样既节省了制作表格时间，又提高了单片机算法执行速度。

3.4.2 消畸变的实现

本设计系统通过单片机与上位机的综合使用来实现图像坐标的还原，具体实现方法如下：

1、制作一个等间隔的黑线板来确定黑线的世界坐标。本系统设定的黑线间隔为 10cm，长度为 1.5m，如图 3.8 所示。

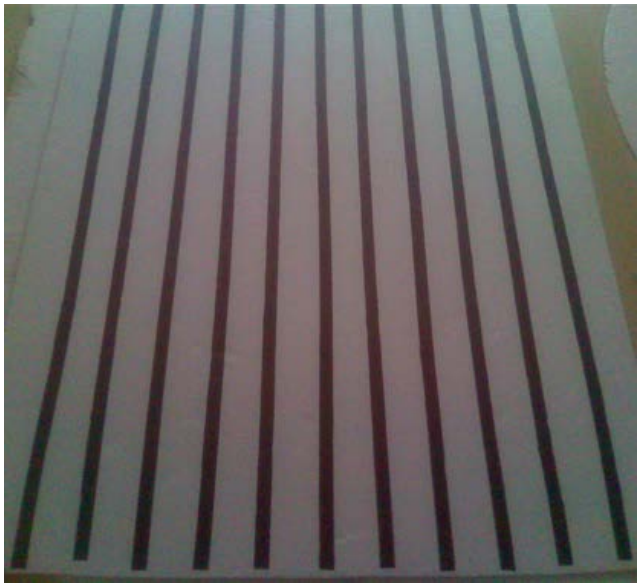

图 3.8　消畸变板

2、利用智能车数据采集系统采集消畸变板信息，并通过无线发送至上位机，采集的数据为 296*270 的数据矩阵，其灰度值用上位机显示如图 3.9 所示。

图 3.9　系统采集图像

3、上位机利用图像数据将每条黑线的中心位置确定出来，并滤掉干扰线，画出图像，与原图像比较，验证其正确性，如图 3.10 所示。

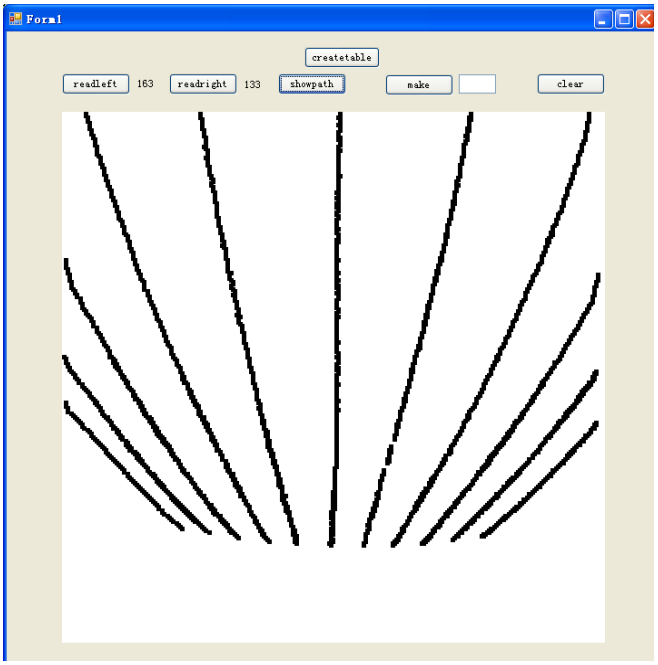

图 3.10　黑线中心线绘制

由于智能车系统发送的数据放于 Excel 表格中，而本上微机系统读取 Excel 表格默认最大读取到 255 列，因此将图像数据分为左右两部分，分两次读入。

通过与真实图像比较可以得出中心位置的提取是否正确，很明显，上图提取出的所有黑线的中心位置是正确的。

4、将每条黑线提取出来，分别研究其图像坐标与世界坐标的函数关系，如图 3.11、3.12、3.13、3.14 所示，其中 make 按键功能为提取图像黑线，通过设置其后面的文本框来设定提取的黑线数量。

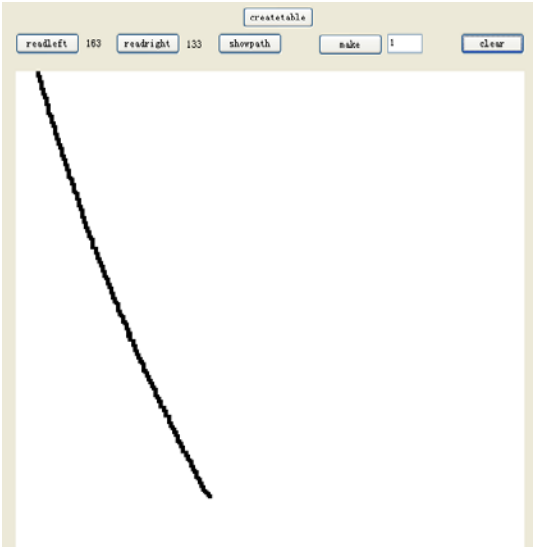

图 3.11　提取 1 条

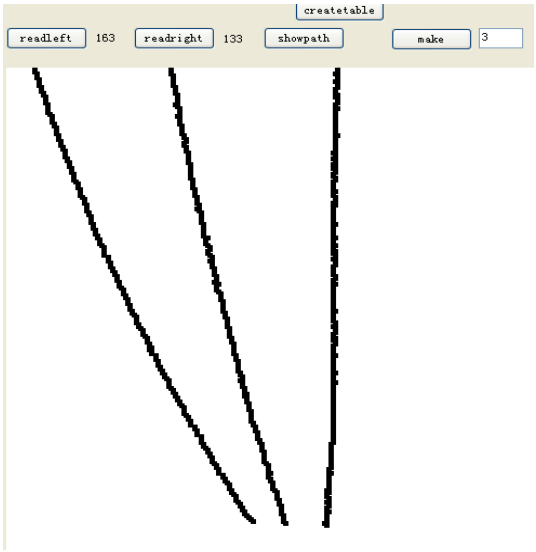

图 3.12　提取 3 条

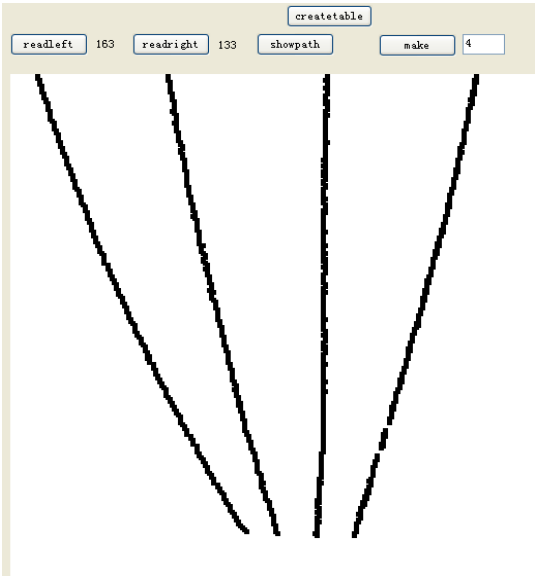

图 3.13　提取 4 条

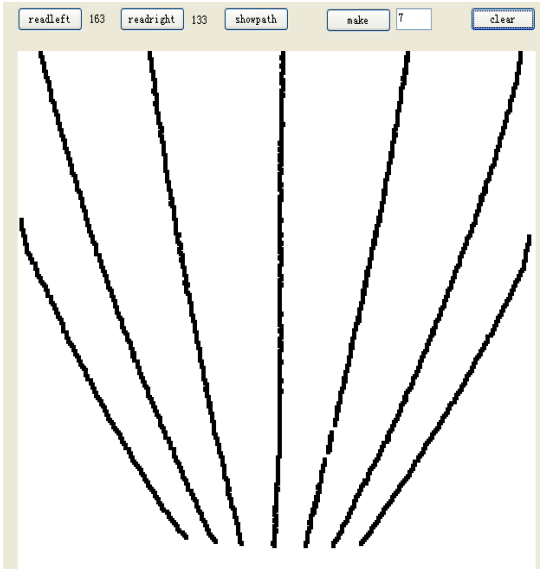

图 3.14　提取 7 条

5、根据每条黑线的图像坐标与世界坐标的函数关系，找寻各函数关系和对应黑线的位置之间的联系规律，确定统一的消极变函数。本设计系统确定出的函数如下：

$$Col = (350 * j - 150 * i) / (350 - i) \quad (3.1)$$

$$Row=23620 / (460 - i) - 50 \quad (3.2)$$

由式 3.1 和式 3.2 可知：图像坐标（i , j）变换后的世界坐标为（Row , Col）。

6、根据确定出的函数关系将黑线进行变换，并绘制出来，验证函数正确性，还原效果如图 3.15 所示。

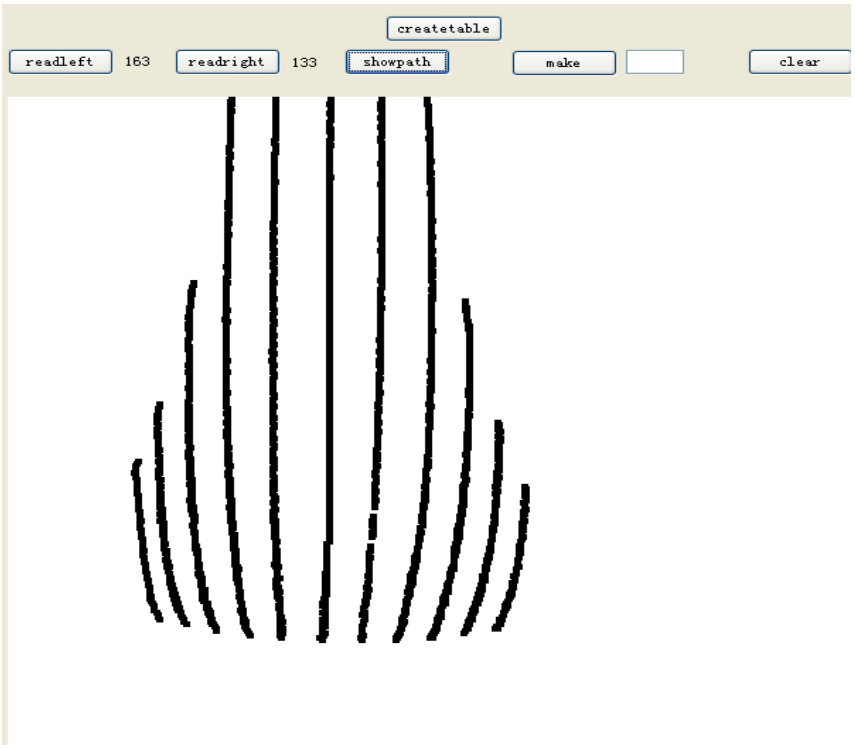

图 3.15　消畸变效果图

7、根据上述公式，将每个图像坐标都计算出相应的世界坐标（createtable 按键），并存储于单片机的 FLASH 存储区，至此消畸变完成。

由上图的消畸变效果可以看出，经过消畸变处理后，图像的畸变程度明显变小，图像变换后的坐标与世界坐标差别较小，完全可以满足智能车行驶的要求，达到了消畸变的目标。经过实验证明，该消畸变方法成功的矫正了图像信息，为智能车的控制算法的参数的提取奠定了基础。由于图像信息处理的稳定性和准确性，智能车的控制策略得以很好的实现。

4 智能车的控制策略研究

上层控制策略对智能车的速度和路径起决定性作用，它是智能车的控制系统，决定着智能车的极限速度。本智能车系统的智能控制策略主要包括路径优化策略、伺服器控制策略、驱动电机控制策略、行驶过程控制策略和停车保护策略。

4.1 路径优化策略

本文的路径优化是指将智能车识别出的路径进行处理变换，来方便智能车行驶，使智能车能够在不冲出跑道的前提下以最快的速度按优化后的路线通过相应的区域。

4.1.1 路径优化思想的确定

由于本智能车后置双电机驱动，将两个后轮分开驱动，这样智能车在后轮主动差速的作用下理论上可以以一个很小的转弯半径快速转弯和漂移。基于上述情况，我们将车的路径设计为内侧过弯，缩短实际行驶路程。

对于“S”型弯路，本设计系统采用曲线拟合的方法，将弯路拟合成直线，从而使智能车以较小的转角幅度甚至不转角快速通过“S”弯。

总之，在主动差速下，转弯几乎不再是智能车速度的主要限制因素，因此本设计的路径优化总思想是尽量将实际路程优化到最短。

4.1.2 路径优化的实现

道路中心线是由一系列线路转折点组成的，路线中线的平面集合线形由直线和曲线组成，一般对曲线的处理方法有插值和拟合两种。插值是目前常用的一种方法，利用三次样条插值法、抛物线加权平均法和张力样条函数插值法都可以很好的解决这一问题，但是计算量往往非常复杂，而单片机对效率要求非常高，所以可以考虑用最小二乘拟合代替[10]。两点之间最短的路程为其直线距离，本文采取将路径拟合为直线的方法来取得最短的优化路径。

最小二乘法原理如下：

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

设直线方程的表达式为：

$$y = a + bx \tag{4.1}$$

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

$$d_1 = y_1 - a - bx_1 \tag{4.2}$$

$$d_2 = y_2 - a - bx_2 \tag{4.3}$$

$$\vdots$$

$$d_n = y_n - a - bx_n \tag{4.4}$$

显然最好测量点都在直线上（即 $d_1=d_2=\cdots\cdots=d_n=0$），求出的 a 和 b 是最理想的，但测量点不可能都在直线上，这样只有考虑 d_1、d_2、……、d_n 为最小，也就是考虑 $d_1+d_2+\cdots\cdots+d_n$ 为最小，但因 d_1、d_2、……、d_n 有正有负，加起来可能相互抵消，因此不可取；而$|d_1|+ |d_2|+\cdots\cdots+ |d_n|$又不好解方程，因而不可行。现在采取一种等效方法：当 $d_1^2+d_2^2+\cdots\cdots+d_n^2$ 对 a 和 b 为最小时，d_1、d_2、……、d_n 也为最小。取（$d_1^2+d_2^2+\cdots\cdots+d_n^2$）为最小值，求 a 和 b 的方法叫最小二乘法。

令

$$D = \sum_{i=1}^{n} d_i^2 = \sum_{i=1}^{n} [y_i - a - b_i]^2 \tag{4.5}$$

D 对 a 和 b 分别求一阶偏导数为：

$$\frac{\partial D}{\partial a} = -2[\sum_{i=1}^{n} y_i - na - b\sum_{i=1}^{n} x_i] \tag{4.6}$$

$$\frac{\partial D}{\partial b} = -2[\sum_{i=1}^{n} x_i y_i - a\sum_{i=1}^{n} x_i - b\sum_{i=1}^{n} x_i^2] \tag{4.7}$$

再求二阶偏导数为：

$$\frac{\partial^2 D}{\partial a^2} = 2n \tag{4.8}$$

$$\frac{\partial^2 D}{\partial b^2} = 2\sum_{i=1}^{n} x_i^2 \tag{4.9}$$

显然：

$$\frac{\partial^2 \mathrm{D}}{\partial \mathrm{a}^2} = 2\mathrm{n} \geq 0 \tag{4.10}$$

$$\frac{\partial^2 D}{\partial b^2} = 2\sum_{i=1}^{n} x_i^2 \geq 0 \tag{4.11}$$

满足最小值条件，令一阶偏导数为零：

$$\sum_{i=1}^{n} y_i - na - b\sum_{i=1}^{n} x_i = 0 \tag{4.12}$$

$$\sum_{i=1}^{n} x_i y_i - a\sum_{i=1}^{n} x_i - b\sum_{i=1}^{n} x_i^2 = 0 \tag{4.13}$$

引入平均值：

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{4.14}$$

$$\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{4.15}$$

$$\overline{x^2} = \frac{1}{n}\sum_{i=1}^{n} x_i^2 \tag{4.16}$$

$$\overline{xy} = \frac{1}{n}\sum_{i=1}^{n} x_i y_i \tag{4.17}$$

则：

$$\overline{y} - a - b\overline{x} = 0 \tag{4.18}$$

$$\overline{xy} - a\overline{x} - b\overline{x^2} = 0 \tag{4.19}$$

解得：

$$a = \overline{y} - b\overline{x} \tag{4.20}$$

$$b = \frac{\overline{xy} - \overline{x}\,\overline{y}}{\overline{x^2} - \overline{x}^2} \tag{4.21}$$

将 a、b 值带入线性方程 $y = a + bx$，即得到回归直线方程。

将计算公式离散化后为：

$$B = \left[\sum (y-\overline{y})(x-\overline{x})\right] / \left[\sum (x-\overline{x})^2\right] \tag{4.22}$$

$$A = \overline{y} - B\overline{x} \tag{4.23}$$

相应程序如下：

```
AvaliableLines=FinalLine-BasedLine;     // FinalLine 为终止行，
                                        //BasedLine 为基准行
MiddleLine=(BasedLine+FinalLine)/2;
SumX=0;
SumY=0;
for(i=BasedLine;i<FinalLine;i++)
{
```

```
    SumX+=i;
    SumY+=Path[i];
  }

  AverageX=SumX/AvaliableLines;
  AverageY=SumY/AvaliableLines;

  SumUp=0;
  SumDown=0;
  for(i=BasedLine;i<FinalLine;i++)
  {
    SumUp+=(Path[i]-AverageY)*(i-AverageX);
    SumDown+=(i-AverageX)*(i-AverageX);
  }
  if(SumDown==0) B=0;
  else B=(int)(SumUp/SumDown);
  A=(SumY-B*SumX)/AvaliableLines;
```

根据上述方法，成功计算出拟合直线的斜率和偏移量。

4.2 伺服器控制策略

4.2.1 伺服器控制方法选择

本系统伺服器控制目标是对于不同曲率半径的跑道，伺服器可以快速而连续的转动合适的角度。伺服器控制常见的算法有三种：一是模糊自适应控制[11]；二是改进 PD 算法或模糊 PD 控制算法[12]；三是直接 PWM 控制法[13]。

模糊控制的优势是其调节响应速度快，但是却会导致伺服器出现打角不连续且参数不直观，调试困难。PD 算法打角比较连续，但是响应速度较慢，当车速度较快时打角迟缓现象会很明显。直接 PWM 控制是指根据检测的不同路径，判断出小车所在位置，按不同的区间给出不同的舵机 PWM 控制信号。这种方法的优点很明显，由于伺服器内部电机为闭环控制，因此可以准确快速的响应 PWM 占空比的变化，因此这种控制是最迅速、最准确的。但同时由于完全依赖 PWM 占空比，如果输入占空比不连续，就会导致打角不连续的情况，因此这种方法对于输入占空比的连续性要求较高。虽然闭环控制 PD 算法在实现所需的角度值方面优于开环控制算法，但是 PD 参数选择不当会很容易导致角度过冲，这使得智能车剧烈摇晃并且转向齿轮转动不顺畅[14]。

本智能车系统是由摄像头进行数据采集的，因此可以采集到较长的跑道信息，同时每 40ms 采集一幅图片，即使车的速度达到 3m/s，每幅图像相差的距离也就是 0.12m，

这样，在摄像头前瞻为 1.5m 时，每幅图像的重复部分会很大，也就是说由相邻图像分析出的转向角度变化会很小。基于上述考虑，我们可以认为本智能车系统几乎不会出现打角不连续的情况。这样，在排除一个限制条件后，舵机控制策略的目标就变为使伺服器以最快的速度转动相应的角度。经过比较，本设计系统选择第三种控制方法，即直接 PWM 控制。

4.2.2 转向角度的计算

进行路径优化后，路径变为一条直线，单片机应控制伺服器打角，使车准确地沿优化后的路线行驶，角度的计算有如下几种方法。

1、面积法。即利用积分的思想，将路径在数据矩阵中的位置进行积分，所得结果即为路径与坐标轴 y 轴包围的面积，根据面积的大小判断出车与黑线的偏移量，从而确定车的转向角度。

2、斜率法。即计算根据优化路径计算路径的斜率和截距，根据斜率和截距确定车转向角度。

3、模式识别法。赛道曲线大致分为直线、90 度曲线、大 S 曲线、小 S 曲线和环路。根据识别出的赛道曲线类型来确定转向角度方案。

由于本系统采用最小二乘法进行路径优化，已经计算出优化后路径的斜率和截距，因此本系统采用斜率法计算转向角度。

4.3 驱动电机控制策略

本系统驱动电机控制采用闭环 PID 控制，为了避免误差累加，采取增量 PID 算法。

4.3.1 PID 算法简介

PID 控制即比例、积分、微分控制，该方法在工程实际中应用相当广泛。PID 控制算法具有结构简单、工作可靠、便于调整、性能稳定等优点。当被控对象的性能、结构等一系列参数无法得知，而且建模等控制方法亦无从下手时，我们便可以应用 PID 控制技术，但是，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。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 控制器就是根据系统的误差，利用比例、积分、微分计算出控制量进行控制的。图 4.1 为 PID 控制系统原理图。

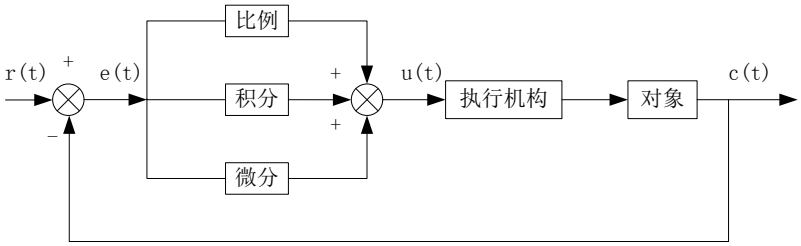

图 4.1　PID 控制系统原理图

PID 函数的一般形式为：U(s)=kp(1+1/(TI*s)+TD*s)；其中 kp 为比例系数，TI 为积分时间，TD 为微分时间常数。

比例控制是一种简单实用的控制方式。该方法的输出与输入误差成比例关系。比例系数越大，调节作用亦越大，减少误差的过程则越迅速，但是比例系数如果过大，则系统的稳定性会大大下降。

对于积分控制，控制器的输出与系统的输入误差信号的积分成正比关系。积分项的引入是为了消除稳态误差，积分项对误差取决于时间的积分，随着时间的增加，积分项会增大。这样，即便误差很小，积分项也会随着时间的增加而加大，它推动控制器的输出增大使稳态误差进一步减小，直到等于零

微分控制中，控制方法的输出与输入误差的变化率成正比关系。智能车控制系统由于存在有较大惯性，其变化总是落后于误差的变化自动控制系统，在克服误差的调节过程中可能会出现振荡，严重时会导致不稳定。为了充分解决此问题，我们需要提前抑制误差的变化作用，即在误差接近零时，抑制误差的作用就应该是零。这就是说，在控制器中仅引入“比例”项往往是不够的，比例项的作用仅是放大误差的幅值，而目前需要增加的是“微分项”，它能预测误差变化的趋势，这样，具有比例+微分的控制器，就能够提前使抑制误差的控制作用等于零，甚至为负值，从而避免了被控量的严重超调。

4.3.2　PID 参数整定

PID 控制算法中 PID 参数的整定是控制系统设计的核心内容。该过程是根据被控过程的特性确定 PID 控制器的比例系数、积分时间和微分时间的大小。

PID 控制器参数整定的方法很多，概括起来有两大类：一是理论计算整定。该方法需要依靠一定的数学模型，即通过一定的模型进行理论计算，最后确定控制器参数。但是，理论计算的方法所得到的数据常常是不能够直接运用到实际中的，其必须通过实际工程进行验证，并进行调整和进一步修改。二是工程经验整定。该方法主要依赖工程经验，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。工程整定方法中的 PID 参数整定主要有临界比例法、衰减法、反应曲线法。三种方法各

有其特点，每一种方法所得到的控制器参数，都需要在实际运行中进行最后调整与完善。此外，其共同点都是通过试验，然后按照工程经验公式对控制器参数进行整定。

进行 PID 控制器参数的整定步骤一般如下：

1、预选择一个足够短的采样周期让系统工作；

2、仅加入比例控制环节，直到系统对输入的阶跃响应出现临界振荡，记下这时的比例放大系数和临界振荡周期；

3、在一定的控制度下通过公式计算得到 PID 控制器的参数；

4、根据实际运行情况对计算出的 PID 控制器的参数进行调整。

本系统的 PID 参数为通过上位机调试得来，具体调试方法为：先将 PID 参数设置为经典参数，然后通过上位机观察速度曲线，不断改变 PID 参数，直至观察速度曲线发现其加减速时间很短，超调量很少，则说明此时的 PID 参数已经基本比较合适，这样就确定出适合本系统的一组 PID 参数。

4.3.3 增量型 PID

数字 PID 控制算法包括位置式和增量式 PID，为了避免历史数据中误差的累加，本设计采用增量式 PID。

增量型算法具有很多优点：增量型算法每次计算的是输出控制量的增量，而控制量是需要加上上次的输出量，这样产生误动作的几率大大降低；增量型算法不需要做累加，增量的确定仅与最近几次偏差采样值有关，计算精度对控制量的计算影响较小，而位置型算法要用到过去偏差的累加值，容易产生大的累加误差；采用增量型算法，易于实现手动到自动的无冲击切换。

控制编程依据：

$$\begin{aligned} u(n) = u(n-1) + Kp*[e(n)-e(n-1)] + Ki*e(n) \\ + Kd*[e(n-2)-2e(n-1)+e(n)] \end{aligned} \tag{4.24}$$

其中： $u(n)$为第 n 次输出控制量；$u(n-1)$为第 n-1 次输出控制量；$e(n)$为第 n 次偏差；$e(n-1)$为第 n-1 次偏差；$e(n-2)$为第 n-2 次偏差；Kp 为比例增益系数；Ki 为积分增益系数；Kd 为微分增益系数。

电机 PID 控制子程序流程如图 4.2 所示。

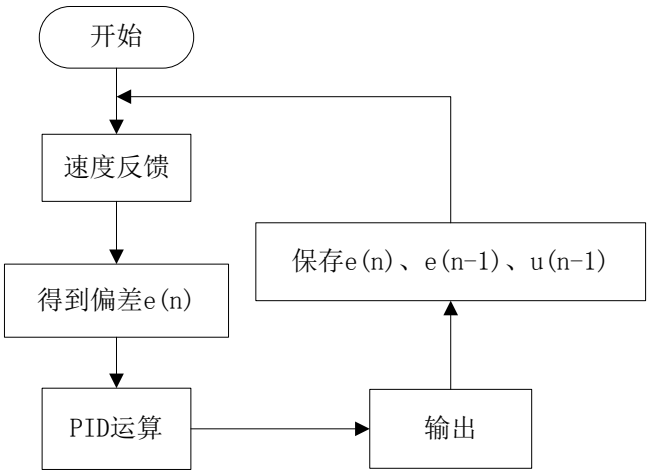

图 4.2　PID 控制流程图

为了实现电机的连续闭环控制，单片机需要实时对测速传感器输出的信号进行采样，通过对电机最大转速、霍尔传感器分辨率、单片机指令执行速度等参数进行分析，结合以往电机控制中的经验，每次采样的周期定为 1.5 毫秒，即电机控制周期为 3 毫秒。

本系统增量型 PID 参数确定是根据 Z-N 条件先确定出一组 PID 参数，然后根据智能车实际行驶状况进行调整，最终变为适合本智能车的经验参数。

4.3.4　主动差速控制策略

本设计系统的差速调节主要思想是当智能车向左转弯时，令左侧驱动电机速度小于右侧驱动电机速度，当智能车向右转弯时，则令右侧驱动电机速度小于左侧驱动电机速度。

在实际测试中，我们发现很难将两个驱动电机的速度设置合适，因为不同的转弯半径下差速是不同的，而差速设置不合适则会影响车整体的行驶路线，只靠伺服器已经无法完全校正车的行驶方向。

为了确定驱动电机的速度控制方案，我们创新性引入了 Ackermann 转向模型的思想来计算转弯时理想状态下每个车轮的速度。

使用 Ackermann 转向模型进行转向时，分析四轮速度关系的假设前提条件为：

1、刚性车体；

2、车轮纯滚动，即不考虑已发生滑移、滑转；

3、行驶时所有轮胎都未离开地面；

4、轮胎侧向变形与侧向力成正比。

该转向模型如图 4.3 所示。

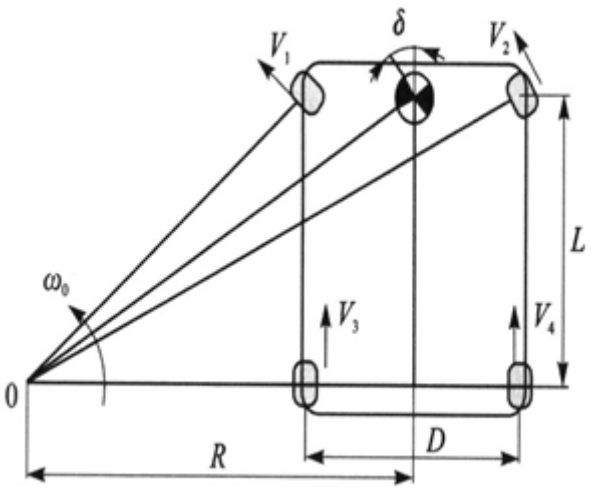

图 4.3　Ackermann 转向模型

其中，轴距 L 和两侧轴线距离 D 是常数值，δ 是伺服器的转角，ω_0 为车绕转向瞬心的角速度，V_1、V_2、V_3、V_4 是 4 个转动轮的速度。由图 4.3 可得：

$$R = L / \tan\delta \tag{4.25}$$

$$V_1 = \omega_0 \times \sqrt{(R - D/2)^2 + L^2} \tag{4.26}$$

$$V_2 = \omega_0 \times \sqrt{(R + D/2)^2 + L^2} \tag{4.27}$$

$$V_3 = \omega_0 \times (R - D/2) \tag{4.28}$$

$$V_4 = \omega_0 \times (R + D/2) \tag{4.29}$$

对于本智能车系统来说，只需对后两轮的速度即 V_3 和 V_4 进行计算即可，而实际上智能车是无法测出 ω_0 的，需要对公式进行变换，变换过程如下：

$$\omega_0 = V / R \tag{4.30}$$

其中 V 为设定的智能车整体速度。将式 4.30 和式 4.25 代入 4.28，消去 ω_0 和 R 后可以得到：

$$V_3 = V - V \times D \times \tan\delta / (2L) \tag{4.31}$$

同理

$$V_4 = V + V \times D \times \tan\delta / (2L) \tag{4.32}$$

由于在单片机中运算三角函数的方法为查表法，运算速度很慢，因此暂时用角度的系数来代替，这样公式就变为：

$$V_3 = V - V \times (\delta / m_3) / n_3 \tag{4.33}$$

$$V_4 = V + V \times (\delta / m_4) / n_4 \tag{4.34}$$

式中 m_3、n_3、m_4、n_4 为固定系数，V 为智能车整体速度，δ 为智能车转向角度。

这样当确定了智能车速度和转向角度后，我们就可以计算出左右驱动电机的速度了。

经过多次测验，智能车在此差速控制策略下能够平稳圆滑过弯，效果较好。

4.3.5 速度控制策略

速度控制策略主要控制的是加速和减速。主要思想为在入弯时迅速减速，然后过弯，出弯时加速，以便安全快速通过弯道。

在早期电机速度控制的方案中，我们将转角的大小作为速度的控制依据，转角越大，速度越小，转角越小，速度越大，这样理论上是可以实现速度控制思想的。但在实际运行中，由于电机速度的调节需要一定时间，因此速度的控制总是慢于转向角度的控制。当车速小于 2m/s 时，智能车可以按这种方法安全沿线行驶，当车速高于 2m/s 时，电机速度调节时间就明显长于转向角度了，导致加减速不及时，从而使智能车冲出赛道。

根据上述情况，本设计系统对速度控制策略进行了调整，以路经判断为主，以转向角度为辅来决定速度的调节。

4.4 行驶过程控制策略

在智能车的行驶过程中，需要实时地对智能车的位置及车体姿态进行调整，否则偏差太大就会影响到车的速度。本系统的控制策略为在转角很小时采用较小的调整幅度，防止智能车在直道上左右乱晃，当转角较大时调整幅度也较大，保证智能车足够的转向角度。当利用中值定理判断出智能车将要入弯时，使用模糊加减速控制方法[15]减速，同时由于本系统设计智能车内侧过弯，因此提前将车向弯路的内侧小幅度靠拢，辅助转弯。

5 上位机系统设计

一般的上位机系统应该主要包括以下几个基本功能：（1）串口通讯功能，该功能是整个上位机系统的基础，此环节是单片机与 PC 机通讯的保证；（2）存储数据并进行回放显示，即把采集的数据保存起来，以便于分析；（3）数据分析，把采集到的数据进行适当的处理分析，此处的数据分析需要进行适当的变成测试，并加入一些算法进行验证。上位机系统流程图如图 5.1 所示。

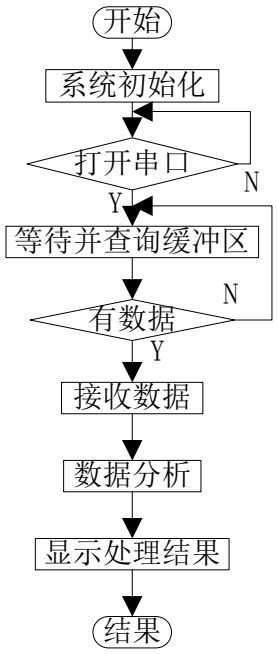

图 5.1 上位机系统流程图

5.1 软件语言的选择

上位机的设计语言有很多种，常见的如: VB、VC++、C#等等 。大家都了解 VB 的不足之处，其早已不符合时代的需要了，微软已停止支持。C++是一门比较不错的程序设计语言，完全面向对象，也支持底层，但 VC++比较难学，对于时间不是很充裕的项目来说，一般应该避免使用。

作为一种面向对象的语言，C#支持面向对象的三大特性，即封装、继承和多态。C#由类构成，所有的变量和方法都封装在类中。类只能直接继承于一个父类，但是可以实现任意数量的接口。而且，C#只允许单继承，即一个类不会有多个基类，从而可以有效避免基类定义导致的混乱。

C#拥有比 C，C++或者 Java 更广泛的数据类型.这些类型是 byte、 ubyte、 short、 ushort、 int、 uint、 long、 ulong、 float、 double 和 decimal。

C#语法表现力强，简单易学，其语法与 C\C++非常相似，因此便于只有普通 C 语言基础的用户学习。与 C++相比，C#的语法简化了许多的复杂特性，如指针访问；而且还提供了很多比较强大的功能，如枚举、委托等。

C#拥有像 VB 一样的快速开发能力；C#的效率接近 VC++，高于 VB；他完全面向对象，符合时代的需要；C#为托管型语言，开发效率更高，出错率更低，并且有很好的异常处理能力；现在 C#的资料越来越多，很方便学习。

C#总体来说更像 Java 语言，而本文作者曾深入学习过 Java 语言，鉴于以上原因及开发周期的考虑，本设计采用了 C#。

5.2 开发软件介绍

本设计使用的开发软件为 Visual Studio 2010。Visual Studio 是微软公司推出的开发环境，是目前流行的 Windows 平台应用程序开发环境。Visual Studio 可以用来创建 Windows 平台下的 Windows 应用程序和网络应用程序，也可以用来创建网络服务、智能设备应用程序和 Office 插件。Visual Studio 包含了许多强大的工具，支持多种编程语言（C#、VB、C++、HTML 与 JavaScript 等），所有语言开发出来的.NET 应用程序效果一样。

Visual Studio 提供了窗体应用程序开发功能，窗体应用程序由于具有图形化窗体接口，因此能够提供给用户较为友善的使用界面。制作窗口界面的应用程序，所需要的功能均封装与一组可视化组件的类里，这些类位于命名空间 System.Windows.Forms，提供大量支持开发图形界面所需的可视化组件，其中内置了支持复杂的用户交互的功能。

此外，Visual Studio 提供了使用鼠标拖拽可视化组件的设计环境，让程序开发人员可以快速的完成上位机界面的创建，而将大部分时间用于编写算法。

Visual Studio 2010 具有如下特点：

1、支持 Windows Azure 。

2、助力移动与嵌入式装置开发。

3、实践当前最热门的 Agile/Scrum 开发方法，强化团队竞争力。

4、升级的软件测试功能及工具，为软件质量严格把关。

5、搭配 Windows 7，Silverlight 4 与 Office，发挥多核并行运算威力，美感与效能并重。

6、支持最新 C++标准，增强 IDE，切实提高程序员开发效率。

5.3 上位机界面设计

在应用 VS2010 的程序设计时，运行在个人电脑上的应用程序由“Windows Form”类型的项目生成。Windows Form 类似于使用 VB6、Dwlphi、VC6 开发出来的窗体，其用户交互功能非常强大，窗体可以拖拽、最大最小化、弹出系统菜单等等。此外，Windows Form 通过调用.NET Framework 和 Windows 操作系统的 API（Application Programming Interface,应用程序编程接口），还可以方便的使用操作系统提供的任意功能。

此外，窗体应用程序的创建需要用到 Form 类，Form 类代表一个应用程序与用户沟通的可视界面或者是对话框，通常创建一个 Windows 应用程序会利用产生 Form 类的实例对象作为人机交互上位机界面，Form 类封装了大量的属性、事件、方法成员，允许程序开发人员对窗体外观进行设置。

本系统的设计用到了大量的控件。控件是一种应用于 Windows 应用程序的可视化组件，如，文本框、按钮以及下拉式菜单等等。这些控件可以用于创建具有图形化接口的应用程序，这些控件类各自具有其特定的功能，统一位于命名空间 System.Windowns.Forms。窗体应用程序的创建界面见图 5.2。

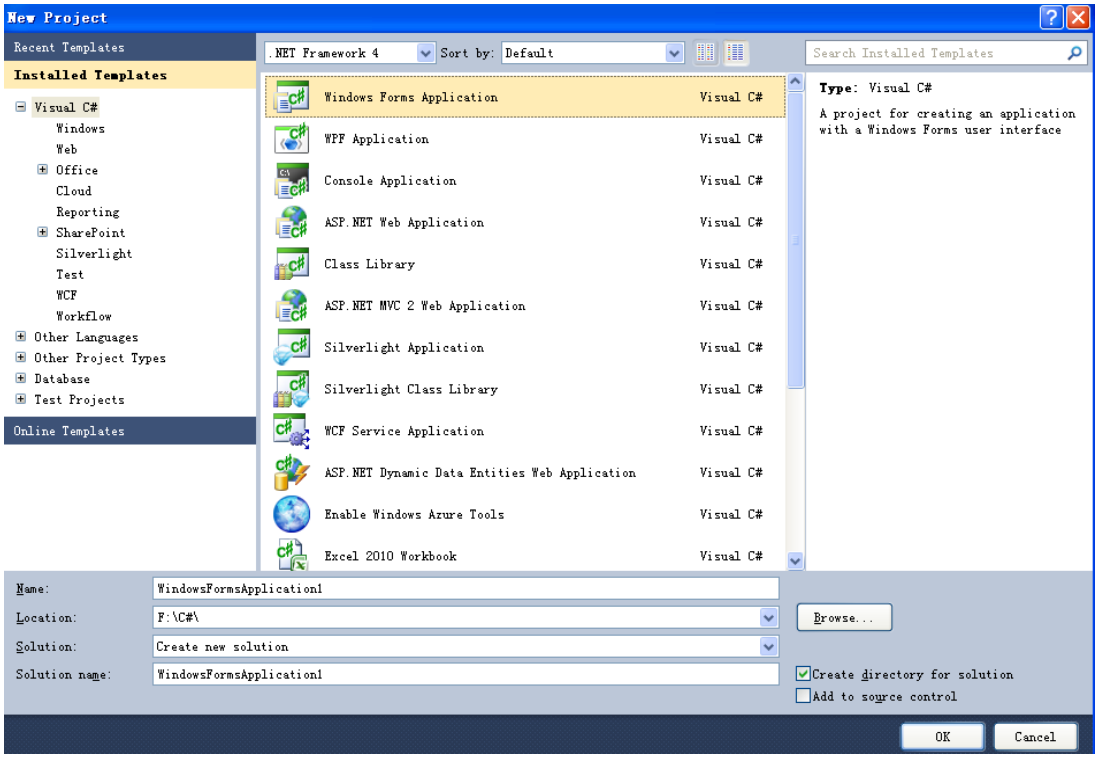

图 5.2 窗体应用程序创建窗口

在这里，我们需要介绍一下 Control 类，该类同样位于命名空间 System.Windowns.Forms，虽然此类不是可见的基本控件，但是其作用却很大。Control 类是所有控件的基类，它定义了多种控件类的共同属性、方法与事件，控件类均是直接或间接继承于此类。因此该类定义了所有控件的一些共同属性，如 Text 属性，该属性为所有控件必有的属性，对于 Button 控件类来说就是按钮上面显示的文字；对于 Form 控件类来说则为 Form 对象的标题。

本设计上位机系统主要利用 VS2010 的串口类（serialport）实现串口通讯，应用窗口应用程序编写上位机界面。本设计的上位机界面主要包括一些关于串口操作的按钮、赛道模拟显示、舵机转角、左右电机速度以及一些操作图形的按钮，如图 5.3 所示。

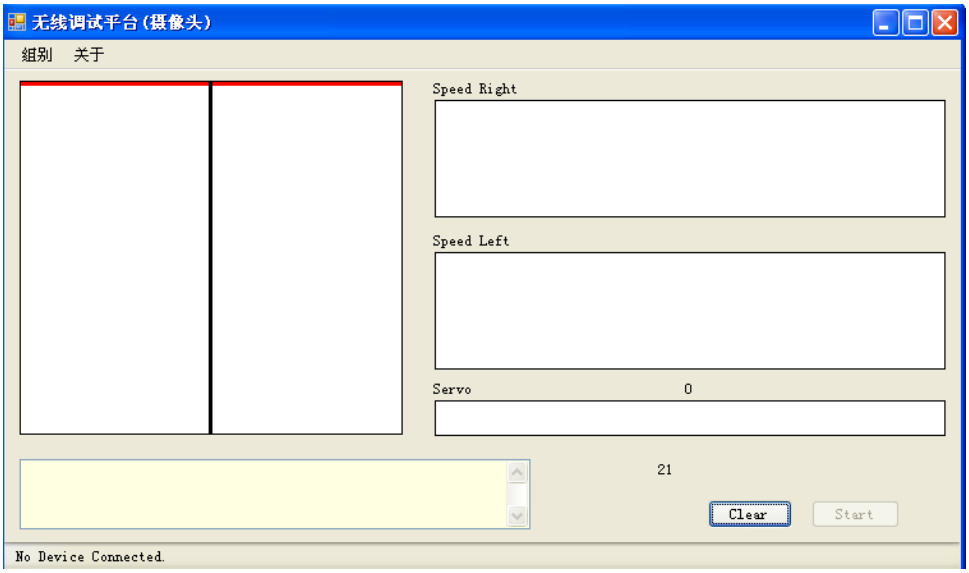

图 5.3　上位机界面

上图中左侧白色框为赛道模拟窗口，Speed Right 窗口用来绘制右轮速度曲线，Speed Left 窗口绘制左轮速度曲线。Servo 窗口显示伺服器的转向角度。左下方信息框显示串口连接信息。当系统检测到串口数据时，Start 按键自动转为活跃状态，点击即可开始接收数据并显示。

5.4　数据处理

由于智能车路径信息数据较为庞大，为了保证传输过程中不会丢字，我们采用分批次传输方法，本设计中将图像信息分为两部分分别传送，再由上位机进行数据整合，将图像信息进行还原。

在数据传输中，本上位机系统定义传输协议如下：第一字节为校验字，设定为 0xFF；第二字节为状态字，0x01 表示传送的是图像前 30 行数据，0x02 表示传送的是图像的后 30 行数据，0x03 表示传送的是基准行、终止行、左右驱动电机速度、转向角度等信息。

上位机接收数据时首先校验第一字节是否正确，如果正确则根据数据协议进行数据处理，否则认为数据有误，丢弃此次接收的数据。

5.5 运行结果分析

经过测试，上位机系统能够正确显示路径、速度、转向角度等信息，达到设计目标。通过观察上位机显示信息，可以直观方便的获知智能车的行驶状态，从而进行分析。上位机运行界面如图 5.4 所示。

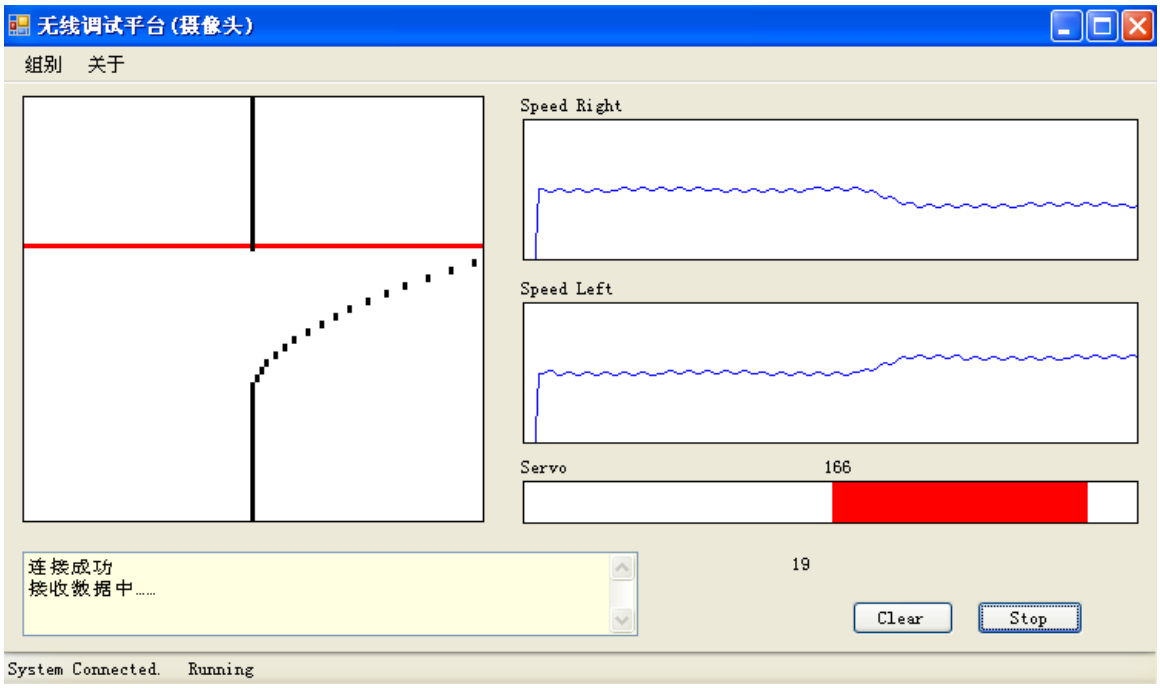

图 5.4 上位机运行界面

上图显示的是智能车右转时的各方面信息。赛道模拟窗口显示的是智能车识别出的路径信息，Speed Right 窗口显示的是右侧电机的速度曲线，Speed Left 窗口显示的是左侧电机的速度曲线。当智能车向右转弯时，根据 Ackermann 转向模型，右侧轮进行减速，左侧轮加速，上图正确显示出速度调节结果。

在智能车调试过程中，上位机系统提供了准确的智能车运行参数，为调试及分析数据提供了重要依据，极大地加快了调试进程。

结　论

本智能车系统的控制策略在历届智能车竞赛的基础上进行了较多的改进，经过大量测试，本智能车系统能够在标准赛道上行驶。在保证不冲出赛道的前提下，智能车的最高速度可达 3m/s，成功达到各项技术指标。本智能车系统最重要的改进如下：

1、路径优化中引入最小二乘法曲线拟合方法。该方法的使用成功解决了智能车系统由于存在个别错误的路径点而对整个路径的判断偏差较大的问题，保证了智能车路径的准确性和稳定性。

2、本智能车系统的重大创新在于在差速控制策略中引入 Ackermann 转向模型。根据此模型确定出的左右电机速度与智能车速度的关系，根据此关系设定左右驱动电机的速度取得的效果明显优于线性差速效果，成功实现圆滑顺利过弯。

由于时间等原因，本系统的控制策略还不够完美。如果时间和财力允许的情况下，可以做如下改进：

1、坡道的识别

由于时间原因，本智能车系统的路径识别中并未加入坡道识别算法，实际上在上坡下坡时，黑色引导线的宽度会有一个反常的变化。正常时距离车身越远的地方采集到的黑线越窄，当智能车前面有坡道时，会出现一段区域，在此区域内黑线变窄的很缓慢甚是会变宽，根据这一原理，可以识别出前方的坡道。另一种方法为安装陀螺仪等车体姿态检测装置，但此类装置价格一般较为昂贵，不推荐使用此方法。

2、主动差速的调节

在主动差速的条件下，理论上智能车可以实现不减速过弯，即转弯时外侧车轮加速，内侧车轮不减速，这样可以进一步提高智能车的平均速度。

3、上位机系统改进

由于时间原因，本智能车系统的上位机软件只能接收数据，并不能向智能车发送命令或者修改参数，调试不太方便。因此可以考虑编写具有双向通信功能的上位机系统，方便智能车的调试。

参 考 文 献

[1] 康华光，陈大钦，张林.电子技术基础[M].北京:高等教育出版社,2006.

[2] 阎石.数字电子技术基础[M].北京:高等教育出版社,1998.

[3] 孙同景.陈桂友.Freescale 9S12 十六位单片机原理及嵌入式开发技术[M].北京:机械工业出版社,2008.

[4] 雷霏霖,梁志毅.基于 CMOS 传感器 OV7620 采集系统设计[J].电子测量技术,2008,12(31):110-112.

[5] 王小红.基于 nRF2401 的无线数据传输系统[J].太原师范学院学报,2006,1(5): 64-66.

[6] 谭浩强.C 程序设计[M].北京:清华大学出版社,2005.

[7] 郭芳,曹桂琴.数据结构基础[M].大连：大连理工大学出版社，1994.

[8] 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.

[9]葛亚明,刘涛,王宗义. 视频同步分离芯片 LM1881 及其应用[J]. 应用科技,2004,31(9):20-22.

[10] 高盼,郭广礼.基于最小二乘法道路平面曲线拟合[J].测绘信息与工程,2011,02(36):19-21.

[11] YUAN Quan,ZHANG YunZhou,WU Hao,et al. Fuzzy Control Research In The Courses Of Smart Car[C]. Machine Vision and Human-Machine Interface (MVHI), Kaifeng,China, 2010:764-767.

[12] 侯虹.采用模糊 PID 控制律的舵机系统设计[J].航空兵器，2006,2(1):7-9.

[13] 孙浩,程磊,黄卫华,等.基于 HCS12 的小车智能控制系统设计[J].单片机与嵌入式系统应用,2007,03(16):51-54.

[14] LU Zhenlin,LI Jingjiao,Zhang Minghui.Intelligent Control Research based on the Smart Car[C]. Advanced Computer Control (ICACC),Shenyang,2010:292-297.

[15] Hyun Mun Kim,Julie Dickerson,Bart Kosko.Fuzzy throttle and brake control for platoons of smart cars[J]. Fuzzy Sets and Systems,1996,86:209-234.

附录

附录 A：程序源代码

```
#include "Cpu.h"
#include "Events.h"
#include "Servo.h"
#include "MotorRight.h"
#include "MotorLeft.h"
#include "MotorDirRight.h"
#include "MotorDirLeft.h"
#include "SpeedTimer.h"
#include "SpeedCounter.h"
#include "SpeedLeft.h"
#include "SpeedRight.h"
#include "Vertical_Int.h"
#include "Field_Int.h"
#include "Video.h"
#include "stopCar.h"
#include "TI1.h"
#include "speed.h"
/* Include shared modules, which are used for whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
#include "IO_Map.h"

/* User includes (#include below this line is not maintained by Processor Expert) */
#include "PID.h"
#include "IMAQ.h"
#include "PWM.h"
#include "Process.h"
#include "nRF_API.h"

//**********************IMAQ**************************//
const word WantedVerticalNum[]=
{
   28, 55, 77, 92,109,
  124,137,150,160,169,
  178,186,193,200,206,
  212,217,222,226,231,
  235,240,243,246,250,
  253,256,258,261,264,
  266,268,270,272,274,
  275,276,278,280,282,
  284,286,287,288,289,
  290,291,292,293,294,
```

```
  295,296,297,298,299,
  300,301,302
};

volatile byte Image[MaxVerticalNum][MaxPointNum];
volatile byte staus = CaptureComplete;
volatile word VerticalNum =0;
volatile word ValidVerticalNum =0;
volatile word pointsNum = 0;

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

//*********************Process************************//
const int PathTable[]={
0,0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,3
1,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,48,50,51,52,52,54,55,55,57,58,58,
59,61,61,62,64,65,65,66,68,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87
,88,89,90,91,92,93,94,95,96,97,98,99,100,101,102,103,104,105,106,107,108,109,110,111,1
12,113,114,115,116,117,118,119,120,121,122,123,123,125,126,126,
-5,-4,-3,-2,-1,0,0,1,3,3,4,6,7,8,9,10,11,12,13,14,15,16,18,18,19,20,22,23,24,25,26,27,
28,29,30,31,32,34,34,35,37,38,39,40,41,42,43,44,45,46,47,49,49,50,51,53,54,55,56,57,58
,59,60,61,62,64,65,65,66,68,69,70,71,72,73,74,75,76,77,78,80,80,81,82,84,85,86,87,88,8
9,90,91,92,93,95,95,96,97,99,100,101,102,103,104,105,106,107,108,109,111,111,112,114,1
15,116,117,118,119,120,121,122,123,124,126,126,127,128,130,131,132,
-10,-9,-8,-7,-6,-5,-4,-3,-1,0,0,2,3,4,5,6,7,8,9,11,12,13,14,15,16,17,19,20,21,22,23,24
,25,27,28,29,30,31,32,33,35,36,37,38,39,40,41,43,44,45,46,47,48,49,50,52,53,54,55,56,5
7,58,60,61,62,63,65,66,67,68,69,70,71,73,74,75,76,77,78,79,81,82,83,84,85,86,87,89,90,
91,92,93,94,95,97,98,99,100,101,102,103,104,106,107,108,109,110,111,112,114,115,116,11
7,118,119,120,122,123,124,125,126,127,128,130,131,132,133,135,136,137,
-14,-13,-12,-11,-9,-8,-7,-6,-5,-3,-2,-1,0,0,1,3,4,5,6,8,9,10,11,13,14,15,16,17,18,20,2
1,22,23,25,26,27,28,30,31,32,33,34,35,36,38,39,40,42,43,44,45,46,48,49,50,51,52,53,55,
56,57,58,60,61,62,63,65,66,67,68,69,70,71,73,74,75,77,78,79,80,81,83,84,85,86,87,88,90
,91,92,93,95,96,97,98,100,101,102,103,104,105,106,108,109,110,112,113,114,115,116,118,
119,120,121,122,123,125,126,127,128,130,131,132,133,135,136,137,138,139,140,
-19,-18,-17,-16,-14,-13,-12,-11,-9,-8,-7,-5,-4,-3,-2,0,0,1,2,4,5,6,8,9,10,12,13,14,15,
17,18,19,21,22,23,24,26,27,28,30,31,32,33,35,36,37,39,40,41,43,44,45,46,48,49,50,52,53
,54,55,57,58,59,61,62,63,65,66,67,68,70,71,72,74,75,76,77,79,80,81,83,84,85,86,88,89,9
0,92,93,94,95,97,98,99,101,102,103,104,106,107,108,109,111,112,113,115,116,117,118,120
,121,122,124,125,126,127,129,130,131,132,134,135,136,138,139,140,141,143,144,145,
-24,-23,-22,-20,-19,-17,-16,-15,-13,-12,-11,-9,-8,-7,-6,-4,-3,-1,0,1,2,3,5,6,7,8,10,11
,13,14,16,17,18,20,21,22,23,25,26,27,29,30,32,33,35,36,37,39,40,41,42,44,45,47,48,50,5
1,52,54,55,56,57,59,60,62,63,65,66,67,69,70,71,72,74,75,76,78,79,81,82,84,85,86,87,89,
90,91,93,94,96,97,99,100,101,103,104,105,106,108,109,110,112,113,115,116,118,119,120,1
21,123,124,125,127,128,130,131,133,134,135,136,138,139,140,142,143,145,146,147,149,150
,
```

−29, −28, −26, −25, −23, −22, −21, −19, −18, −16, −15, −13, −12, −11, −9, −8, −6, −5, −4, −2, −1, 0, 2, 3, 4, 5, 7, 8, 10, 12, 13, 14, 15, 17, 19, 20, 21, 23, 24, 25, 27, 29, 30, 31, 33, 34, 36, 37, 39, 40, 41, 43, 44, 46, 47, 49, 50, 51, 53, 54, 56, 57, 59, 60, 61, 63, 65, 66, 67, 69, 70, 71, 73, 75, 76, 77, 79, 80, 81, 83, 85, 86, 87, 88, 90, 92, 93, 95, 96, 97, 98, 100, 102, 103, 105, 106, 107, 109, 110, 112, 113, 114, 116, 117, 119, 121, 122, 123, 124, 126, 127, 129, 131, 132, 133, 134, 136, 138, 139, 140, 142, 143, 144, 146, 148, 149, 150, 152, 153, 154,
−34, −33, −32, −30, −28, −27, −26, −24, −22, −21, −20, −18, −16, −15, −14, −12, −10, −9, −8, −6, −4, −3, −1, 0, 1, 2, 4, 5, 7, 9, 10, 11, 13, 15, 16, 17, 19, 21, 22, 23, 25, 27, 28, 29, 31, 33, 34, 36, 37, 39, 40, 42, 43, 45, 46, 48, 49, 51, 53, 54, 55, 57, 59, 60, 61, 63, 65, 66, 67, 69, 70, 72, 73, 75, 76, 78, 80, 81, 82, 84, 86, 87, 88, 90, 92, 93, 94, 96, 98, 99, 100, 102, 104, 105, 107, 108, 110, 111, 113, 114, 116, 117, 119, 120, 122, 124, 125, 126, 128, 130, 131, 132, 134, 136, 137, 138, 140, 142, 143, 144, 146, 148, 149, 151, 152, 154, 155, 157, 158, 160,
−39, −37, −36, −35, −33, −31, −30, −28, −26, −25, −24, −21, −20, −19, −17, −15, −14, −12, −11, −9, −8, −6, −4, −3, −1, 0, 1, 3, 4, 6, 7, 9, 10, 12, 14, 15, 16, 19, 20, 21, 23, 25, 26, 28, 30, 31, 32, 35, 36, 37, 39, 41, 42, 44, 45, 47, 48, 50, 52, 53, 55, 56, 58, 60, 61, 63, 65, 66, 67, 69, 71, 72, 74, 76, 77, 78, 81, 82, 83, 85, 87, 88, 90, 91, 93, 94, 96, 98, 99, 101, 102, 104, 106, 107, 109, 110, 112, 113, 115, 117, 118, 119, 122, 123, 124, 126, 128, 129, 131, 133, 134, 135, 138, 139, 140, 142, 144, 145, 147, 148, 150, 151, 153, 155, 156, 158, 159, 161, 163, 164,
−43, −42, −40, −39, −37, −35, −34, −32, −30, −29, −27, −25, −24, −22, −21, −19, −17, −16, −14, −12, −11, −9, −7, −6, −4, −3, −1, 0, 1, 4, 5, 6, 8, 10, 12, 13, 14, 17, 18, 20, 22, 23, 25, 26, 28, 30, 31, 33, 35, 36, 38, 40, 41, 43, 44, 46, 48, 49, 51, 53, 54, 56, 58, 59, 61, 63, 65, 66, 67, 70, 71, 72, 74, 76, 78, 79, 81, 83, 84, 86, 88, 89, 91, 92, 94, 96, 97, 99, 101, 102, 104, 106, 107, 109, 111, 112, 114, 115, 117, 119, 120, 122, 124, 125, 127, 129, 131, 132, 133, 136, 137, 139, 141, 142, 144, 145, 147, 149, 150, 152, 154, 155, 157, 159, 160, 162, 163, 165, 167, 168,
−48, −47, −45, −44, −41, −40, −38, −37, −35, −33, −32, −29, −28, −26, −25, −22, −21, −19, −18, −16, −14, −13, −10, −9, −7, −6, −4, −2, 0, 1, 2, 4, 5, 8, 9, 11, 12, 14, 16, 17, 20, 21, 23, 24, 27, 28, 30, 32, 33, 35, 36, 39, 40, 42, 43, 46, 47, 49, 51, 52, 54, 55, 58, 59, 61, 63, 65, 66, 68, 70, 71, 73, 74, 77, 78, 80, 82, 83, 85, 86, 89, 90, 92, 93, 96, 97, 99, 101, 102, 104, 105, 108, 109, 111, 113, 115, 116, 118, 120, 121, 123, 124, 127, 128, 130, 132, 134, 135, 137, 139, 140, 142, 144, 146, 147, 149, 151, 152, 154, 156, 158, 159, 161, 163, 165, 166, 168, 170, 171, 173,
−53, −51, −50, −48, −46, −44, −43, −41, −39, −37, −36, −33, −32, −30, −29, −26, −25, −23, −21, −19, −17, −16, −13, −12, −10, −9, −6, −5, −3, −1, 0, 1, 3, 5, 7, 8, 10, 12, 14, 16, 18, 19, 21, 23, 25, 27, 28, 31, 32, 34, 35, 38, 39, 41, 42, 45, 46, 48, 50, 52, 53, 55, 57, 59, 61, 63, 65, 66, 68, 70, 72, 73, 75, 77, 79, 80, 83, 84, 86, 87, 90, 91, 93, 95, 97, 98, 100, 102, 104, 106, 107, 110, 111, 113, 115, 117, 118, 120, 122, 124, 125, 127, 129, 131, 132, 135, 136, 138, 140, 142, 143, 145, 147, 149, 151, 152, 155, 156, 158, 159, 162, 163, 165, 167, 169, 170, 172, 174, 176, 177,
−57, −56, −54, −53, −50, −48, −47, −45, −43, −41, −39, −37, −35, −34, −32, −30, −28, −26, −25, −22, −21, −19, −16, −15, −13, −12, −9, −7, −6, −3, −2, 0, 1, 3, 5, 6, 8, 10, 12, 14, 16, 18, 19, 21, 24, 25, 27, 29, 31, 33, 34, 37, 38, 40, 42, 44, 46, 47, 50, 51, 53, 55, 57, 59, 60, 63, 65, 66, 68, 70, 72, 74, 75, 78, 79, 81, 83, 85, 87, 88, 91, 92, 94, 96, 98, 100, 101, 104, 105, 107, 109, 111, 113, 115, 117, 119, 120, 122, 124, 126, 128, 129, 132, 133, 135, 137, 139, 141, 142, 145, 146, 148, 151, 152, 154, 156, 158, 160, 161, 163, 165, 167, 169, 171, 173, 174, 176, 178, 180, 182,
−62, −61, −59, −57, −55, −53, −51, −50, −47, −45, −44, −41, −39, −38, −36, −33, −32, −30, −28, −26, −24, −22, −20, −18, −16, −15, −12, −10, −9, −6, −4, −3, −1, 1, 2, 4, 6, 8, 10, 12, 14, 16, 18, 19, 22, 24, 25, 28, 30, 31, 33, 36, 37, 39, 41, 43, 45, 47, 49, 51, 53, 54, 57, 59, 60, 63, 65, 66, 68, 70, 72, 74, 76, 78, 80, 82, 84, 86, 88, 89, 92, 93, 95, 97, 99, 101, 103, 105, 107, 109, 111, 113, 115, 116, 119, 121, 122, 124, 127, 128, 130, 13

2, 134, 136, 138, 140, 142, 144, 145, 148, 150, 151, 154, 156, 157, 159, 162, 163, 165, 167, 169, 171, 173, 175, 177, 179, 180, 183, 185, 186,
−67, −65, −63, −61, −59, −57, −55, −54, −51, −49, −47, −45, −43, −41, −39, −37, −35, −33, −31, −29, −27, −25, −23, −21, −19, −17, −15, −13, −11, −9, −7, −5, −3, −1, 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 27, 28, 30, 32, 35, 36, 38, 40, 42, 44, 46, 49, 50, 52, 54, 57, 58, 60, 63, 65, 66, 68, 71, 72, 74, 76, 79, 80, 82, 85, 87, 88, 90, 93, 94, 96, 98, 101, 102, 104, 107, 109, 110, 112, 115, 117, 118, 121, 123, 124, 126, 129, 131, 132, 134, 137, 139, 140, 143, 145, 147, 148, 151, 153, 154, 157, 159, 161, 162, 165, 167, 169, 170, 173, 175, 176, 179, 181, 183, 184, 187, 189, 191,
−72, −70, −68, −66, −63, −61, −60, −58, −55, −53, −51, −49, −47, −45, −43, −40, −39, −37, −35, −32, −30, −29, −26, −24, −22, −20, −18, −16, −14, −11, −9, −8, −6, −3, −1, 0, 1, 4, 6, 8, 11, 12, 14, 16, 19, 21, 22, 25, 27, 29, 31, 33, 35, 37, 39, 42, 43, 45, 48, 50, 52, 54, 56, 58, 60, 63, 65, 66, 68, 71, 73, 75, 76, 79, 81, 83, 86, 87, 89, 91, 94, 96, 97, 99, 102, 104, 106, 108, 110, 112, 114, 117, 118, 120, 123, 125, 127, 128, 131, 133, 135, 137, 139, 141, 143, 146, 148, 149, 151, 154, 156, 158, 160, 162, 164, 166, 169, 170, 172, 174, 177, 179, 181, 183, 185, 187, 189, 191, 193, 195,
−76, −74, −72, −70, −67, −65, −64, −62, −59, −57, −55, −52, −50, −48, −47, −44, −42, −40, −38, −35, −33, −32, −29, −27, −25, −23, −20, −18, −16, −14, −12, −10, −8, −5, −3, −1, 0, 2, 4, 6, 9, 11, 13, 15, 17, 19, 21, 24, 26, 28, 30, 32, 34, 36, 38, 41, 43, 45, 48, 49, 51, 53, 56, 58, 60, 63, 65, 66, 68, 71, 73, 75, 77, 80, 81, 83, 86, 88, 90, 92, 95, 97, 98, 100, 103, 105, 107, 110, 112, 113, 115, 118, 120, 122, 125, 127, 129, 130, 133, 135, 137, 139, 142, 144, 146, 148, 150, 152, 154, 157, 159, 161, 163, 165, 167, 169, 172, 174, 176, 178, 180, 182, 184, 187, 189, 191, 193, 195, 197, 199,
−80, −78, −76, −75, −72, −70, −68, −66, −63, −61, −59, −56, −54, −52, −50, −47, −45, −43, −41, −39, −37, −35, −32, −30, −28, −26, −23, −21, −19, −16, −14, −12, −10, −7, −5, −4, −2, 0, 2, 4, 7, 9, 11, 13, 16, 18, 20, 23, 25, 27, 29, 31, 33, 35, 37, 40, 42, 44, 47, 49, 51, 53, 56, 58, 60, 63, 65, 66, 68, 71, 73, 75, 77, 80, 82, 84, 87, 89, 91, 93, 96, 98, 100, 101, 104, 106, 108, 111, 113, 115, 117, 120, 122, 124, 127, 129, 131, 133, 135, 137, 139, 141, 144, 146, 148, 151, 153, 155, 157, 160, 162, 164, 167, 169, 170, 172, 175, 177, 179, 181, 184, 186, 188, 191, 193, 195, 197, 200, 202, 204,
−84, −82, −80, −78, −75, −73, −71, −69, −66, −64, −62, −59, −57, −55, −53, −50, −48, −46, −44, −41, −39, −37, −34, −32, −30, −28, −25, −23, −21, −18, −16, −14, −12, −9, −7, −5, −3, 0, 1, 3, 6, 8, 10, 12, 15, 17, 19, 22, 24, 26, 28, 31, 33, 35, 37, 40, 42, 44, 47, 49, 51, 53, 56, 58, 60, 63, 65, 66, 68, 71, 73, 75, 77, 80, 82, 84, 87, 89, 91, 93, 96, 98, 100, 102, 105, 107, 109, 112, 114, 116, 118, 121, 123, 125, 128, 130, 132, 134, 137, 139, 141, 143, 146, 148, 150, 153, 155, 157, 159, 162, 164, 166, 169, 171, 173, 175, 178, 180, 182, 184, 187, 189, 191, 194, 196, 198, 200, 203, 205, 207,
−89, −87, −85, −83, −80, −78, −76, −74, −71, −69, −67, −63, −61, −59, −57, −54, −52, −50, −48, −45, −43, −41, −38, −36, −34, −31, −28, −26, −24, −21, −19, −17, −15, −12, −10, −8, −6, −3, −1, 1, 4, 6, 8, 10, 13, 15, 17, 20, 22, 24, 26, 29, 31, 34, 36, 39, 41, 43, 46, 48, 50, 52, 55, 57, 59, 62, 65, 67, 69, 72, 74, 76, 78, 81, 83, 85, 88, 90, 92, 94, 98, 100, 102, 104, 107, 109, 111, 114, 116, 118, 120, 123, 125, 127, 131, 133, 135, 137, 140, 142, 144, 146, 149, 151, 153, 156, 158, 160, 163, 166, 168, 170, 173, 175, 177, 179, 182, 184, 186, 188, 191, 193, 196, 199, 201, 203, 205, 208, 210, 212,
−94, −91, −89, −87, −84, −82, −80, −78, −75, −72, −70, −67, −65, −63, −61, −58, −55, −53, −51, −48, −46, −44, −41, −38, −36, −34, −31, −29, −27, −24, −21, −19, −17, −14, −12, −10, −8, −5, −2, 0, 2, 4, 6, 8, 11, 14, 16, 19, 21, 23, 25, 28, 31, 33, 35, 38, 40, 42, 45, 48, 50, 52, 55, 57, 59, 62, 65, 67, 69, 72, 74, 76, 78, 81, 84, 86, 89, 91, 93, 95, 98, 101, 103, 105, 108, 110, 112, 115, 118, 120, 122, 125, 127, 129, 132, 135, 137, 139, 142, 144, 146, 148, 151, 154, 156, 159, 161, 163, 165, 168, 171, 173, 176, 178, 180, 182, 185, 188, 190, 192, 195, 197, 199, 202, 205, 207, 209, 212, 214, 216,
−99, −97, −95, −93, −89, −87, −85, −83, −80, −77, −75, −72, −70, −67, −65, −62, −60, −58, −55, −52, −50, −48, −44, −42, −40, −38, −35, −32, −30, −27, −25, −22, −20, −17, −15, −13, −10, −7, −5, −3, 0, 2, 4, 6, 10, 12, 1

4, 17, 19, 22, 24, 27, 29, 32, 34, 37, 39, 41, 45, 47, 49, 51, 55, 57, 59, 62, 65, 67, 69, 72, 74, 77, 79, 82, 84, 86, 90, 92, 94, 96, 100, 102, 104, 106, 110, 112, 114, 117, 119, 122, 124, 127, 129, 132, 135, 137, 139, 141, 145, 147, 149, 151, 155, 157, 159, 162, 165, 167, 169, 172, 174, 177, 180, 182, 184, 186, 190, 192, 194, 196, 200, 202, 204, 207, 210, 212, 214, 217, 219, 222,
−103, −101, −99, −96, −93, −91, −88, −86, −83, −81, −78, −75, −73, −70, −68, −65, −63, −60, −58, −55, −52, −50, −47, −45, −42, −40, −37, −34, −32, −29, −27, −24, −22, −19, −17, −14, −12, −9, −6, −4, −1, 0, 3, 5, 8, 11, 13, 16, 18, 21, 23, 26, 29, 31, 33, 36, 39, 41, 44, 47, 49, 51, 54, 57, 59, 62, 65, 67, 69, 72, 75, 77, 79, 82, 85, 87, 90, 93, 95, 97, 100, 103, 105, 107, 111, 113, 115, 118, 121, 123, 125, 129, 131, 133, 136, 139, 141, 143, 147, 149, 151, 153, 157, 159, 161, 164, 167, 169, 171, 175, 177, 179, 182, 185, 187, 189, 193, 195, 197, 199, 203, 205, 207, 211, 213, 215, 217, 221, 223, 225,
−107, −105, −102, −100, −97, −94, −92, −90, −86, −84, −82, −78, −76, −74, −71, −68, −65, −63, −61, −57, −55, −53, −49, −47, −45, −43, −39, −37, −34, −31, −29, −26, −24, −21, −18, −16, −14, −10, −8, −6, −2, 0, 1, 4, 7, 9, 12, 15, 17, 20, 22, 25, 28, 30, 32, 36, 38, 40, 44, 46, 48, 51, 54, 56, 59, 62, 65, 67, 69, 73, 75, 77, 79, 83, 85, 87, 91, 93, 96, 98, 101, 104, 106, 108, 112, 114, 116, 120, 122, 124, 127, 130, 132, 135, 138, 140, 143, 145, 148, 151, 153, 155, 159, 161, 163, 167, 169, 171, 174, 177, 179, 182, 185, 187, 190, 192, 195, 198, 200, 202, 206, 208, 210, 214, 216, 218, 221, 224, 227, 229,
−112, −110, −107, −105, −102, −99, −97, −94, −91, −89, −86, −83, −80, −78, −76, −72, −70, −67, −65, −61, −59, −57, −53, −51, −48, −46, −42, −40, −38, −34, −32, −29, −27, −23, −21, −19, −16, −13, −10, −8, −4, −2, 0, 2, 5, 8, 10, 14, 16, 18, 21, 24, 27, 29, 31, 35, 37, 40, 43, 46, 48, 50, 54, 56, 59, 62, 65, 67, 69, 73, 75, 78, 80, 83, 86, 88, 92, 94, 96, 99, 102, 105, 107, 110, 113, 115, 118, 121, 124, 126, 128, 132, 134, 137, 140, 143, 145, 147, 151, 153, 156, 158, 162, 164, 166, 170, 172, 175, 177, 181, 183, 185, 189, 191, 194, 196, 200, 202, 204, 207, 210, 213, 215, 219, 221, 223, 226, 229, 232, 234,
−117, −114, −112, −109, −106, −103, −101, −98, −95, −92, −90, −86, −84, −81, −79, −75, −73, −70, −68, −64, −62, −59, −56, −53, −51, −49, −45, −42, −40, −36, −34, −32, −29, −26, −23, −21, −18, −15, −12, −10, −6, −4, −1, 0, 4, 6, 9, 12, 15, 17, 20, 23, 26, 28, 31, 34, 37, 39, 43, 45, 48, 50, 54, 56, 58, 62, 65, 67, 69, 73, 75, 78, 80, 84, 86, 89, 92, 95, 97, 100, 103, 106, 108, 111, 114, 117, 119, 123, 125, 128, 130, 134, 136, 139, 142, 145, 147, 149, 153, 156, 158, 160, 164, 166, 169, 172, 175, 177, 180, 183, 186, 188, 192, 194, 197, 199, 203, 205, 208, 210, 214, 216, 219, 222, 225, 227, 230, 233, 236, 238,
−121, −118, −116, −114, −110, −107, −105, −102, −99, −96, −94, −90, −87, −85, −82, −79, −76, −74, −71, −68, −65, −63, −59, −56, −54, −51, −48, −45, −43, −39, −36, −34, −31, −28, −25, −23, −20, −17, −14, −12, −8, −5, −3, 0, 2, 5, 7, 11, 14, 16, 19, 22, 25, 27, 30, 33, 36, 38, 42, 45, 47, 50, 53, 56, 58, 62, 65, 67, 69, 73, 76, 78, 81, 84, 87, 89, 93, 96, 98, 101, 104, 107, 109, 112, 115, 118, 120, 124, 127, 129, 132, 135, 138, 140, 144, 147, 149, 152, 155, 158, 160, 163, 166, 169, 171, 175, 178, 180, 183, 186, 189, 191, 195, 198, 200, 202, 206, 209, 211, 214, 217, 220, 222, 226, 229, 231, 234, 237, 240, 242,
−124, −122, −119, −117, −113, −110, −108, −105, −101, −99, −96, −92, −90, −87, −85, −81, −79, −76, −74, −70, −67, −65, −61, −58, −56, −53, −50, −47, −44, −41, −38, −36, −33, −29, −27, −24, −22, −18, −15, −13, −9, −7, −4, −1, 1, 4, 6, 10, 13, 15, 18, 22, 24, 27, 29, 33, 35, 38, 42, 44, 47, 49, 53, 56, 58, 62, 65, 67, 70, 73, 76, 78, 81, 85, 87, 90, 94, 96, 99, 101, 105, 107, 110, 113, 116, 119, 121, 125, 128, 130, 133, 137, 139, 142, 145, 148, 150, 153, 157, 159, 162, 164, 168, 171, 173, 177, 180, 182, 185, 188, 191, 193, 197, 200, 202, 205, 209, 211, 214, 216, 220, 222, 225, 229, 231, 234, 236, 240, 243, 245,
−129, −126, −124, −121, −117, −115, −112, −110, −106, −103, −100, −97, −94, −91, −89, −85, −82, −80, −77, −73, −71, −68, −64, −62, −59, −56, −52, −50, −47, −43, −41, −38, −36, −32, −29, −27, −24, −20, −17, −15, −11, −8, −6, −3, 0, 2, 5, 9, 11, 14, 17, 20, 23, 26, 28, 32, 35, 37, 41, 44, 46, 49, 53, 55, 58, 62, 65, 67, 70, 74, 76, 79, 81, 85, 88, 90, 94, 97, 100, 102, 106, 109, 111, 114, 118, 120, 123, 127, 129, 132, 135, 138, 141, 144, 147, 150, 153, 155, 159, 162, 164, 167, 171, 173, 176, 180, 182, 185, 188, 192, 194, 197, 201, 203, 206, 208, 212, 215, 217, 220, 224, 227, 229, 233, 236, 238, 241, 245, 247, 250,

−134, −131, −129, −126, −122, −119, −117, −114, −110, −107, −105, −101, −98, −95, −93, −89, −86, −84, −81, −77, −74, −72, −68, −65, −62, −60, −56, −53, −50, −46, −44, −41, −38, −34, −32, −29, −26, −22, −20, −17, −13, −10, −8, −5, −1, 1, 3, 7, 10, 13, 15, 19, 22, 25, 27, 31, 34, 37, 41, 43, 46, 49, 53, 55, 58, 62, 65, 67, 70, 74, 76, 79, 82, 86, 88, 91, 95, 98, 100, 103, 107, 110, 112, 115, 119, 122, 124, 128, 131, 134, 136, 140, 143, 146, 150, 152, 155, 158, 162, 164, 167, 170, 174, 176, 179, 183, 186, 188, 191, 195, 198, 200, 204, 207, 210, 212, 216, 219, 221, 224, 228, 231, 233, 237, 240, 243, 245, 249, 252, 255,
−138, −135, −132, −130, −125, −123, −120, −117, −113, −111, −108, −104, −101, −98, −96, −92, −89, −86, −83, −79, −77, −74, −70, −67, −65, −62, −58, −55, −52, −48, −46, −43, −40, −36, −33, −31, −28, −24, −21, −18, −14, −12, −9, −6, −2, 0, 2, 6, 9, 12, 14, 18, 21, 24, 27, 31, 33, 36, 40, 43, 46, 48, 52, 55, 58, 62, 65, 67, 70, 74, 77, 79, 82, 86, 89, 92, 96, 98, 101, 104, 108, 111, 113, 116, 120, 123, 125, 130, 132, 135, 138, 142, 144, 147, 151, 154, 157, 159, 163, 166, 169, 171, 176, 178, 181, 185, 188, 190, 193, 197, 200, 203, 207, 209, 212, 215, 219, 222, 224, 227, 231, 234, 236, 241, 243, 246, 249, 253, 255, 258,
−141, −139, −136, −133, −129, −126, −123, −121, −117, −114, −111, −107, −104, −101, −99, −94, −92, −89, −86, −82, −79, −77, −72, −70, −67, −64, −60, −57, −54, −50, −48, −45, −42, −38, −35, −32, −30, −26, −23, −20, −16, −13, −10, −8, −3, −1, 1, 5, 8, 11, 13, 18, 20, 23, 26, 30, 33, 36, 40, 42, 45, 48, 52, 55, 58, 62, 65, 67, 70, 74, 77, 80, 82, 87, 89, 92, 96, 99, 102, 104, 109, 111, 114, 117, 121, 124, 127, 131, 133, 136, 139, 143, 146, 149, 153, 156, 158, 161, 165, 168, 171, 173, 178, 180, 183, 187, 190, 193, 195, 200, 202, 205, 209, 212, 215, 218, 222, 224, 227, 230, 234, 237, 240, 244, 247, 249, 252, 256, 259, 262,
−145, −142, −140, −137, −133, −130, −127, −124, −120, −117, −114, −110, −107, −104, −102, −97, −95, −92, −89, −85, −82, −79, −75, −72, −69, −67, −62, −59, −57, −52, −50, −47, −44, −40, −37, −34, −31, −27, −24, −22, −17, −15, −12, −9, −5, −2, 0, 4, 7, 10, 13, 17, 20, 22, 25, 29, 32, 35, 39, 42, 45, 48, 52, 55, 57, 62, 65, 67, 70, 74, 77, 80, 83, 87, 90, 93, 97, 100, 102, 105, 109, 112, 115, 118, 122, 125, 128, 132, 135, 138, 140, 145, 147, 150, 154, 157, 160, 163, 167, 170, 173, 175, 180, 182, 185, 189, 192, 195, 198, 202, 205, 208, 212, 215, 218, 220, 225, 227, 230, 233, 237, 240, 243, 247, 250, 253, 255, 260, 263, 265,
−149, −146, −143, −141, −136, −133, −131, −128, −123, −121, −118, −113, −110, −108, −105, −100, −98, −95, −92, −88, −85, −82, −78, −75, −72, −69, −65, −62, −59, −55, −52, −49, −46, −42, −39, −36, −33, −29, −26, −23, −19, −16, −13, −10, −6, −3, 0, 3, 6, 9, 12, 16, 19, 22, 24, 29, 32, 34, 39, 42, 44, 47, 52, 54, 57, 62, 65, 67, 70, 75, 77, 80, 83, 87, 90, 93, 97, 100, 103, 106, 110, 113, 116, 119, 123, 126, 129, 133, 136, 139, 142, 146, 149, 152, 156, 159, 162, 165, 169, 172, 175, 178, 182, 185, 188, 192, 195, 198, 200, 205, 208, 210, 215, 218, 220, 223, 228, 230, 233, 236, 240, 243, 246, 251, 253, 256, 259, 263, 266, 269,
−153, −150, −147, −145, −140, −137, −134, −131, −127, −124, −121, −117, −114, −111, −108, −104, −101, −98, −95, −91, −88, −85, −80, −77, −75, −72, −67, −64, −61, −57, −54, −51, −48, −44, −41, −38, −35, −31, −28, −25, −21, −18, −15, −12, −7, −5, −2, 2, 5, 8, 11, 15, 18, 21, 24, 28, 31, 34, 38, 41, 44, 47, 51, 54, 57, 62, 65, 67, 70, 75, 78, 81, 83, 88, 91, 94, 98, 101, 104, 107, 111, 114, 117, 120, 124, 127, 130, 135, 137, 140, 143, 148, 151, 153, 158, 161, 164, 167, 171, 174, 177, 180, 184, 187, 190, 194, 197, 200, 203, 207, 210, 213, 218, 221, 223, 226, 231, 234, 237, 240, 244, 247, 250, 254, 257, 260, 263, 267, 270, 273,
−155, −152, −149, −147, −142, −139, −136, −133, −129, −126, −123, −119, −116, −113, −110, −105, −102, −99, −96, −92, −89, −86, −82, −79, −76, −73, −68, −66, −63, −58, −55, −52, −49, −45, −42, −39, −36, −32, −29, −26, −21, −18, −15, −13, −8, −5, −2, 1, 4, 7, 10, 14, 17, 20, 23, 28, 31, 34, 38, 41, 44, 47, 51, 54, 57, 62, 65, 67, 70, 75, 78, 81, 84, 88, 91, 94, 98, 101, 104, 107, 112, 115, 118, 120, 125, 128, 131, 135, 138, 141, 144, 148, 151, 154, 159, 162, 165, 168, 172, 175, 178, 181, 185, 188, 191, 196, 198, 201, 204, 209, 212, 215, 219, 222, 225, 228, 232, 235, 238, 241, 246, 249, 252, 256, 259, 262, 265, 269, 272, 275,
−158, −155, −152, −149, −144, −141, −138, −135, −131, −128, −125, −120, −117, −114, −111, −107, −104, −101, −98, −94, −91, −88, −83, −80, −77, −74, −70, −67, −64, −59, −56, −53, −50, −46, −43, −40, −37, −33, −30, −27, −22, −19, −16, −13, −9, −6, −3, 1, 4, 7, 9, 14, 17, 20, 23, 27, 30, 33, 38, 41, 44, 47, 51, 54, 57, 62, 65, 67, 70, 75, 78, 81, 84, 88, 91, 94, 99, 102, 105, 108, 112, 115, 118, 121, 125, 128, 131, 136, 139, 142, 145

, 149, 152, 155, 160, 163, 166, 169, 173, 176, 179, 182, 186, 189, 192, 197, 200, 203, 206, 210, 213, 216, 221, 224, 227, 230, 234, 237, 240, 243, 247, 250, 253, 258, 261, 264, 267, 271, 274, 277,
−162, −159, −156, −153, −148, −145, −142, −139, −135, −132, −129, −124, −121, −118, −115, −110, −107, −104, −101, −97, −94, −91, −86, −83, −80, −77, −73, −69, −66, −62, −59, −56, −53, −48, −45, −42, −39, −35, −32, −29, −24, −21, −18, −15, −10, −7, −4, 0, 2, 5, 8, 13, 16, 19, 22, 27, 30, 33, 37, 40, 43, 46, 51, 54, 57, 61, 65, 68, 71, 75, 78, 81, 84, 89, 92, 95, 99, 102, 105, 108, 113, 116, 119, 122, 127, 130, 133, 137, 140, 143, 146, 151, 154, 157, 162, 165, 168, 171, 175, 178, 181, 184, 189, 192, 195, 199, 203, 206, 209, 213, 216, 219, 224, 227, 230, 233, 237, 240, 243, 247, 251, 254, 257, 262, 265, 268, 271, 275, 278, 281,
−167, −164, −160, −157, −153, −150, −147, −143, −139, −136, −133, −128, −125, −122, −119, −114, −111, −108, −105, −100, −97, −94, −89, −86, −83, −80, −75, −72, −69, −65, −61, −58, −55, −51, −47, −44, −41, −37, −34, −30, −26, −23, −20, −17, −12, −9, −6, −1, 1, 4, 7, 12, 15, 18, 21, 26, 29, 32, 37, 40, 43, 46, 51, 54, 57, 61, 65, 68, 71, 75, 78, 82, 85, 89, 92, 95, 100, 103, 106, 109, 114, 117, 120, 123, 128, 131, 134, 139, 142, 145, 148, 153, 156, 159, 164, 167, 170, 173, 177, 181, 184, 187, 191, 195, 198, 202, 205, 208, 212, 216, 219, 222, 227, 230, 233, 236, 241, 244, 247, 250, 255, 258, 261, 266, 269, 272, 275, 280, 283, 286,
−171, −168, −165, −162, −157, −154, −151, −148, −143, −140, −137, −132, −129, −126, −123, −118, −115, −111, −108, −104, −100, −97, −92, −89, −86, −83, −78, −75, −72, −67, −64, −61, −58, −53, −50, −47, −44, −39, −36, −32, −28, −25, −21, −18, −13, −10, −7, −2, 0, 3, 6, 11, 14, 17, 20, 25, 28, 31, 36, 39, 42, 46, 50, 53, 57, 61, 65, 68, 71, 76, 79, 82, 85, 90, 93, 96, 101, 104, 107, 110, 115, 118, 121, 125, 129, 132, 136, 140, 143, 147, 150, 155, 158, 161, 166, 169, 172, 175, 180, 183, 186, 189, 194, 197, 200, 205, 208, 211, 215, 219, 222, 226, 230, 234, 237, 240, 245, 248, 251, 254, 259, 262, 265, 270, 273, 276, 279, 284, 287, 290,
−177, −173, −170, −167, −162, −159, −156, −152, −147, −144, −141, −136, −133, −130, −127, −122, −118, −115, −112, −107, −104, −101, −96, −93, −89, −86, −81, −78, −75, −70, −67, −64, −60, −56, −52, −49, −46, −41, −38, −35, −30, −26, −23, −20, −15, −12, −9, −4, −1, 2, 5, 10, 13, 16, 19, 24, 27, 31, 35, 39, 42, 45, 50, 53, 56, 61, 65, 68, 71, 76, 79, 82, 85, 90, 94, 97, 102, 105, 108, 111, 116, 119, 123, 126, 131, 134, 137, 142, 145, 148, 152, 156, 160, 163, 168, 171, 174, 177, 182, 186, 189, 192, 197, 200, 203, 208, 211, 215, 218, 223, 226, 229, 234, 237, 240, 244, 248, 252, 255, 258, 263, 266, 269, 274, 277, 281, 284, 289, 292, 295,
−182, −178, −175, −172, −167, −164, −160, −157, −152, −149, −146, −141, −137, −134, −131, −126, −122, −119, −116, −111, −108, −104, −99, −96, −93, −89, −85, −81, −78, −73, −70, −66, −63, −58, −55, −52, −48, −43, −40, −37, −32, −28, −25, −22, −17, −14, −10, −5, −2, 0, 4, 8, 12, 15, 18, 23, 27, 30, 35, 38, 41, 45, 50, 53, 56, 61, 65, 68, 71, 76, 79, 83, 86, 91, 94, 97, 102, 106, 109, 112, 117, 121, 124, 127, 132, 135, 139, 144, 147, 150, 154, 158, 162, 165, 170, 173, 177, 180, 185, 188, 191, 195, 200, 203, 206, 211, 215, 218, 221, 226, 229, 233, 238, 241, 244, 247, 252, 256, 259, 262, 267, 271, 274, 279, 282, 285, 289, 294, 297, 300,
−185, −181, −178, −175, −170, −166, −163, −160, −155, −151, −148, −143, −140, −136, −133, −128, −125, −121, −118, −113, −110, −106, −101, −98, −95, −91, −86, −83, −80, −75, −71, −68, −65, −60, −56, −53, −50, −45, −41, −38, −33, −30, −26, −23, −18, −15, −11, −6, −3, 0, 3, 8, 11, 15, 18, 23, 26, 30, 35, 38, 41, 45, 50, 53, 56, 61, 65, 68, 71, 76, 80, 83, 86, 91, 95, 98, 103, 106, 110, 113, 118, 121, 125, 128, 133, 136, 140, 145, 148, 151, 155, 160, 163, 166, 171, 175, 178, 181, 186, 190, 193, 196, 201, 205, 208, 213, 216, 220, 223, 228, 231, 235, 240, 243, 246, 250, 255, 258, 261, 265, 270, 273, 276, 281, 285, 288, 291, 296, 300, 303,

−187, −184, −181, −177, −172, −169, −165, −162, −157, −154, −150, −145, −142, −138, −135, −130, −127, −123, −120, −115, −111, −108, −103, −100, −96, −93, −88, −84, −81, −76, −73, −69, −66, −61, −58, −54, −51, −46, −42, −39, −34, −31, −27, −24, −19, −15, −12, −7, −4, 0, 2, 7, 11, 14, 17, 22, 26, 29, 34, 38, 41, 44, 49, 53, 56, 61, 65, 68, 71, 76, 80, 83, 86, 91, 95, 98, 103, 107, 110, 113, 118, 122, 125, 129, 134, 137, 140, 145, 149, 152, 156, 161, 164, 167, 172, 176, 179, 182, 188, 191, 194, 198, 203, 206, 209, 214, 218, 221, 225, 230, 233, 236, 241, 245, 248, 252, 257, 260, 263, 267, 272, 275, 279, 284, 287, 290, 294, 299, 302, 305,

−190, −187, −183, −180, −175, −171, −168, −165, −159, −156, −153, −148, −144, −141, −137, −132, −129, −125, −122, −117, −113, −110, −105, −102, −98, −95, −90, −86, −83, −78, −74, −71, −67, −62, −59, −55, −52, −47, −44, −40, −35, −32, −28, −25, −20, −16, −13, −8, −4, −1, 1, 7, 10, 13, 17, 22, 25, 29, 34, 37, 41, 44, 49, 53, 56, 61, 65, 68, 71, 76, 80, 83, 87, 92, 95, 99, 104, 107, 111, 114, 119, 122, 126, 129, 134, 138, 141, 146, 150, 153, 157, 162, 165, 168, 174, 177, 180, 184, 189, 192, 196, 199, 204, 208, 211, 216, 220, 223, 226, 232, 235, 238, 243, 247, 250, 254, 259, 262, 266, 269, 274, 278, 281, 286, 289, 293, 296, 301, 305, 308,
−193, −190, −186, −183, −178, −174, −171, −167, −162, −159, −155, −150, −146, −143, −140, −134, −131, −128, −124, −119, −115, −112, −107, −103, −100, −97, −91, −88, −84, −79, −76, −72, −69, −64, −60, −57, −53, −48, −45, −41, −36, −33, −29, −26, −21, −17, −14, −9, −5, −2, 1, 6, 9, 13, 16, 21, 25, 28, 33, 37, 40, 44, 49, 52, 56, 61, 65, 68, 71, 77, 80, 83, 87, 92, 96, 99, 104, 108, 111, 114, 120, 123, 127, 130, 135, 139, 142, 147, 151, 154, 158, 163, 166, 170, 175, 178, 182, 185, 190, 194, 197, 201, 206, 209, 213, 218, 221, 225, 228, 233, 237, 240, 245, 249, 252, 256, 261, 264, 268, 271, 276, 280, 283, 289, 292, 295, 299, 304, 308, 311,
−196, −193, −189, −186, −180, −177, −173, −170, −165, −161, −158, −152, −149, −145, −142, −137, −133, −130, −126, −121, −118, −114, −109, −105, −102, −98, −93, −90, −86, −81, −77, −74, −70, −65, −62, −58, −55, −50, −46, −43, −37, −34, −30, −27, −22, −18, −15, −9, −6, −2, 0, 5, 9, 12, 16, 21, 24, 28, 33, 37, 40, 44, 49, 52, 56, 61, 65, 68, 71, 77, 80, 84, 87, 92, 96, 99, 105, 108, 112, 115, 120, 124, 127, 131, 136, 139, 143, 148, 152, 155, 159, 164, 167, 171, 176, 180, 183, 187, 192, 195, 199, 202, 207, 211, 214, 220, 223, 227, 230, 235, 239, 242, 248, 251, 255, 258, 263, 267, 270, 274, 279, 282, 286, 291, 295, 298, 302, 307, 310, 314,
−199, −196, −192, −188, −183, −180, −176, −173, −167, −164, −160, −155, −151, −148, −144, −139, −136, −132, −128, −123, −120, −116, −111, −107, −104, −100, −95, −91, −88, −83, −79, −76, −72, −67, −63, −60, −56, −51, −47, −44, −39, −35, −31, −28, −23, −19, −16, −10, −7, −3, 0, 5, 8, 12, 15, 20, 24, 27, 33, 36, 40, 43, 49, 52, 56, 61, 65, 68, 72, 77, 80, 84, 87, 93, 96, 100, 105, 109, 112, 116, 121, 124, 128, 132, 137, 140, 144, 149, 153, 156, 160, 165, 169, 172, 177, 181, 184, 188, 193, 197, 200, 204, 209, 213, 216, 221, 225, 229, 232, 237, 241, 244, 250, 253, 257, 260, 266, 269, 273, 276, 281, 285, 288, 294, 297, 301, 304, 310, 313, 317,
−202, −199, −195, −191, −186, −183, −179, −175, −170, −166, −163, −158, −154, −150, −147, −141, −138, −134, −131, −125, −122, −118, −113, −109, −106, −102, −97, −93, −90, −84, −81, −77, −74, −68, −65, −61, −58, −52, −49, −45, −40, −36, −33, −29, −24, −20, −17, −11, −8, −4, −1, 4, 7, 11, 15, 20, 23, 27, 32, 36, 40, 43, 48, 52, 56, 61, 65, 68, 72, 77, 81, 84, 88, 93, 97, 100, 106, 109, 113, 116, 122, 125, 129, 132, 138, 141, 145, 150, 154, 157, 161, 166, 170, 173, 179, 182, 186, 189, 195, 198, 202, 205, 211, 214, 218, 223, 227, 230, 234, 239, 243, 247, 252, 255, 259, 263, 268, 271, 275, 279, 284, 288, 291, 296, 300, 304, 307, 313, 316, 320,
−205, −202, −198, −195, −189, −185, −182, −178, −173, −169, −166, −160, −157, −153, −149, −144, −140, −137, −133, −128, −124, −120, −115, −111, −108, −104, −99, −95, −92, −86, −83, −79, −75, −70, −66, −63, −59, −54, −50, −46, −41, −37, −34, −30, −25, −21, −18, −12, −9, −5, −1, 3, 7, 10, 14, 19, 23, 27, 32, 36, 39, 43, 48, 52, 55, 61, 65, 68, 72, 77, 81, 84, 88, 93, 97, 101, 106, 110, 113, 117, 122, 126, 130, 133, 139, 142, 146, 151, 155, 158, 162, 167, 171, 175, 180, 184, 187, 191, 196, 200, 204, 207, 213, 216, 220, 225, 229, 232, 236, 241, 245, 249, 254, 258, 261, 265, 270, 274, 278, 281, 287, 290, 294, 299, 303, 306, 310, 315, 319, 323,
−209, −205, −201, −198, −192, −188, −185, −181, −176, −172, −168, −163, −159, −156, −152, −146, −143, −139, −136, −130, −126, −123, −117, −114, −110, −106, −101, −97, −93, −88, −84, −81, −77, −72, −68, −64, −61, −55, −51, −48, −42, −39, −35, −31, −26, −22, −19, −13, −9, −6, −2, 2, 6, 10, 13, 19, 22, 26, 32, 35, 39, 43, 48, 52, 55, 61, 65, 68, 72, 77, 81, 85, 88, 94, 97, 101, 107, 110, 114, 117, 123, 127, 130, 134, 139, 143, 147, 152, 156, 160, 163, 169, 172, 176, 181, 185, 189, 192, 198, 202, 205, 209, 214, 218, 222, 227, 231, 234, 238, 244, 247, 251, 256, 260, 264, 267, 273, 276, 280, 284, 289, 293, 297, 302, 306, 309, 313, 318, 322, 326,

−212, −208, −205, −201, −195, −192, −188, −184, −179, −175, −171, −166, −162, −158, −155, −149, −145, −142, −138, −132, −129, −125, −119, −116, −112, −108, −103, −99, −95, −90, −86, −82, −79, −73, −70, −66, −62, −57, −53, −49, −44, −40, −36, −33, −27, −23, −20, −14, −10, −7, −3, 2, 5, 9, 13, 18, 22, 26, 31, 35, 39, 42, 48, 52, 55, 61, 65, 68, 72, 77, 81, 85, 89, 94, 98, 101, 107, 111, 114, 118, 124, 127, 131, 135, 140, 144, 148, 153, 157, 161, 164, 170, 174, 177, 183, 187, 190, 194, 200, 203, 207, 211, 216, 220, 224, 229, 233, 237, 240, 246, 249, 253, 259, 262, 266, 270, 275, 279, 283, 286, 292, 296, 299, 305, 309, 312, 316, 322, 325, 329,
−215, −212, −208, −204, −199, −195, −191, −187, −182, −178, −174, −169, −165, −161, −157, −152, −148, −144, −140, −135, −131, −127, −122, −118, −114, −111, −105, −101, −97, −92, −88, −84, −81, −75, −71, −67, −64, −58, −54, −51, −45, −41, −37, −34, −28, −24, −21, −15, −11, −8, −4, 1, 5, 8, 12, 18, 21, 25, 31, 35, 38, 42, 48, 51, 55, 61, 65, 68, 72, 78, 81, 85, 89, 94, 98, 102, 108, 111, 115, 119, 124, 128, 132, 136, 141, 145, 149, 154, 158, 162, 166, 171, 175, 179, 184, 188, 192, 196, 201, 205, 209, 212, 218, 222, 226, 231, 235, 239, 242, 248, 252, 255, 261, 265, 269, 272, 278, 282, 285, 289, 295, 299, 302, 308, 312, 315, 319, 325, 329, 332,
−219, −215, −211, −208, −202, −198, −194, −190, −185, −181, −177, −171, −168, −164, −160, −154, −151, −147, −143, −137, −134, −130, −124, −120, −117, −113, −107, −103, −99, −94, −90, −86, −82, −77, −73, −69, −65, −60, −56, −52, −46, −43, −39, −35, −29, −26, −22, −16, −12, −8, −5, 0, 4, 8, 11, 17, 21, 25, 30, 34, 38, 42, 47, 51, 55, 61, 65, 68, 72, 78, 82, 85, 89, 95, 99, 102, 108, 112, 116, 119, 125, 129, 133, 137, 142, 146, 150, 156, 159, 163, 167, 173, 176, 180, 186, 190, 193, 197, 203, 207, 210, 214, 220, 224, 228, 233, 237, 241, 245, 250, 254, 258, 264, 267, 271, 275, 281, 284, 288, 292, 298, 301, 305, 311, 315, 319, 322, 328, 332, 336,
−222, −219, −215, −211, −205, −201, −198, −194, −188, −184, −180, −174, −171, −167, −163, −157, −153, −150, −146, −140, −136, −132, −126, −123, −119, −115, −109, −105, −102, −96, −92, −88, −84, −78, −75, −71, −67, −61, −57, −54, −48, −44, −40, −36, −30, −27, −23, −17, −13, −9, −6, 0, 3, 7, 11, 17, 20, 24, 30, 34, 38, 41, 47, 51, 55, 61, 65, 68, 72, 78, 82, 86, 89, 95, 99, 103, 109, 112, 116, 120, 126, 130, 134, 137, 143, 147, 151, 157, 160, 164, 168, 174, 178, 182, 187, 191, 195, 199, 205, 208, 212, 216, 222, 226, 230, 235, 239, 243, 247, 253, 256, 260, 266, 270, 274, 278, 283, 287, 291, 295, 301, 304, 308, 314, 318, 322, 326, 331, 335, 339,
−226, −222, −218, −215, −209, −205, −201, −197, −191, −187, −183, −178, −174, −170, −166, −160, −156, −152, −148, −143, −139, −135, −129, −125, −121, −117, −111, −108, −104, −98, −94, −90, −86, −80, −76, −73, −69, −63, −59, −55, −49, −45, −41, −38, −32, −28, −24, −18, −14, −10, −6, −1, 2, 6, 10, 16, 20, 24, 30, 33, 37, 41, 47, 51, 55, 61, 65, 68, 72, 78, 82, 86, 90, 96, 100, 103, 109, 113, 117, 121, 127, 131, 135, 138, 144, 148, 152, 158, 162, 166, 170, 175, 179, 183, 189, 193, 197, 201, 206, 210, 214, 218, 224, 228, 232, 238, 241, 245, 249, 255, 259, 263, 269, 273, 276, 280, 286, 290, 294, 298, 304, 308, 311, 317, 321, 325, 329, 335, 339, 343,
−230, −226, −222, −218, −212, −208, −204, −200, −195, −191, −187, −181, −177, −173, −169, −163, −159, −155, −151, −145, −141, −137, −131, −128, −124, −120, −114, −110, −106, −100, −96, −92, −88, −82, −78, −74, −70, −65, −61, −57, −51, −47, −43, −39, −33, −29, −25, −19, −15, −11, −7, −1, 1, 5, 9, 15, 19, 23, 29, 33, 37, 41, 47, 51, 55, 61, 65, 68, 72, 78, 82, 86, 90, 96, 100, 104, 110, 114, 118, 122, 128, 131, 135, 139, 145, 149, 153, 159, 163, 167, 171, 177, 181, 185, 191, 195, 198, 202, 208, 212, 216, 220, 226, 230, 234, 240, 244, 248, 252, 258, 261, 265, 271, 275, 279, 283, 289, 293, 297, 301, 307, 311, 315, 321, 325, 328, 332, 338, 342, 346,
−234, −230, −226, −222, −216, −212, −208, −204, −198, −194, −190, −184, −180, −176, −172, −166, −162, −158, −154, −148, −144, −140, −134, −130, −126, −122, −116, −112, −108, −102, −98, −94, −90, −84, −80, −76, −72, −66, −62, −58, −52, −48, −44, −40, −34, −30, −26, −20, −16, −12, −8, −2, 1, 5, 9, 15, 19, 23, 29, 33, 37, 41, 47, 51, 55, 61, 65, 68, 72, 78, 82, 86, 90, 96, 100, 104, 110, 114, 118, 122, 128, 132, 136, 140, 146, 1

```
50,154,160,164,168,172,178,182,186,192,196,200,204,210,214,218,222,228,232,236,242,246
,250,254,260,264,268,274,278,282,286,292,296,300,304,310,314,318,324,328,332,336,342,3
46,350
};
int Path[MaxVerticalNum];
int BasedLine;
int FinalLine;
int MiddleLine;
int MiddlePointPosition;
byte OneLine[MaxPointNum];

int AverageX;
int AverageY;
int SumX;
int SumY;
long int SumUp;
long int SumDown;
int B;
int A;
int Offset;
int OffsetFinal;

int TurnSet;
int TurnSize;
#define BaseSpeed 45
int TurnSetLast;

int speed1 =0;
int speed2 =0;

int sssssssss=0;
//*****************************************************//

void wirelessSend(void);

void main(void)
{
  int i=0;
  int j=0;
  int line;

  static int BaseStableCounter=0;
  static int FinalStableCounter=0;
  static int ImageStableCounter=0;
  static int SearchPoint;
  static int SearchPointL;
```

```
static int SearchPointR;

int temp1;
int temp2;
int temp3;

word MinPoint;
word MaxPoint;

static word TresholdValue;

unsigned char EdgeL;
unsigned char EdgeR;
unsigned char EdgeCenter;
static unsigned char Edge_Counter;
static bool Edge_OK=FALSE;
static bool Image_OK=FALSE;

int AvaliableLines;

/*** Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! ***/
PE_low_level_init();
/*** End of Processor Expert internal initialization.                    ***/
//****************无线初始化****************//
//SPI_Init();
//nRF24L01_Init();
//******************************************//

//****************状态初始化****************//
(void)Servo_SetRatio16(ServoMiddlePosition);
speedSetRight = 35;
speedSetLeft = 35;
sssssssssss = speed_GetVal();
switch(sssssssssss)
{
  case 0:speed1=80;speed2=56;break;
  case 1:speed1=80;speed2=55;break;
  case 2:speed1=80;speed2=54;break;
  case 3:speed1=80;speed2=53;break;
  case 4:speed1=54;speed2=54;break;
  case 5:speed1=53;speed2=53;break;
  case 6:speed1=52;speed2=52;break;
  case 7:speed1=50;speed2=50;break;
  case 8:speed1=40;speed2=40;break;
  case 9:speed1=35;speed2=35;break;
  case 10:speed1=30;speed2=30;break;
```

```
  case 11:speed1=50;speed2=30;break;
}

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

//****************延时****************//

Cpu_Delay100US(10000);
Cpu_Delay100US(10000);

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

//****************开启测速****************//
SpeedLeft_SetVal();
SpeedRight_ClrVal();
SpeedCounter_ResetCounter();
(void)SpeedTimer_Enable();
motor = RIGHT;
//*****************************************//
for(;;)
{
//****************图像采集****************//
  ECLKCTL = 0x4B;//5MHz
  staus = Capturing;
  Field_Int_Enable();
  while(staus == Capturing);
  ECLKCTL = 0xC0;
  Field_Int_Disable();
  Vertical_Int_Disable();
//*****************************************//

//***********寻找基础行(BaseLine)***********//
  BasedLine = 0;
  BaseStableCounter = 0;
  //ImageStableCounter=0;
  Image_OK=FALSE;
  for(line=0;line<10;line++)
  {
    Edge_OK = FALSE;
    //求当前行阈值
    MaxPoint = MinPoint = 40;
    for(i=0;i<MaxPointNum;i++)
    {
      if(Image[line][i] > Image[line][MaxPoint])
        MaxPoint = i;
      else if(Image[line][i] < Image[line][MinPoint]&&Image[line][i]>30)
```

```
    MinPoint = i;
}
TresholdValue = (Image[line][MinPoint] + Image[line][MaxPoint]) / 2;


//二值化
for(i=0;i<MaxPointNum;i++)
{
  if(Image[line][i] > TresholdValue)
    OneLine[i] = White;
  else
    OneLine[i] = Black;
}

//边沿化
Edge_Counter = 0;
for(i=0;i<MaxPointNum-1;i++)
{
  if(OneLine[i] != OneLine[i+1])
  {
    OneLine[i++] = Edge;
    Edge_Counter++;
  }
}
if(Edge_Counter < MaxEdgeNum)
{
  //寻找类赛道边沿信息
  if(SearchPoint == 0)
  {
    SearchPointL=0;
    SearchPointR=MaxPointNum-1;
  }
  else
  {
    if(SearchPoint < SearchRange)
      SearchPointL = 0;
    else
      SearchPointL = SearchPoint - SearchRange;

    if(SearchPoint > MaxPointNum - SearchRange - 1)
      SearchPointR = MaxPointNum - 1;
    else
      SearchPointR = SearchPoint + SearchRange;
  }
```

```
  for(i=SearchPointL;i<SearchPointR-1;i++)
  {
    if(OneLine[i] == Edge)
    {
      for(j=i+1;j<SearchPointR;j++)
      {
        if(OneLine[j] == Edge)
        {
          if(j-i>15) break;
          if(j-i>3&&j-i<=15)
          {
            EdgeCenter=(byte)(i+j)/2;
            if(OneLine[EdgeCenter]==Black)
            {
              Edge_OK=TRUE;
              EdgeL=(byte)i;
              EdgeR=(byte)j;
            }
            i=j-1;
            break;
          }
        }
      }
      if(Edge_OK == TRUE)
        break;
    }
  }
}
//判断基础黑线稳定性
if(Edge_OK == TRUE)
{
  BaseStableCounter++;
  Path[line] = (EdgeR + EdgeL) / 2;
  SearchPoint = Path[line];
}
else
{
  BaseStableCounter = 0;
  SearchPoint = 0;
}

if(BaseStableCounter > BaseLineStableNum)
{
  BasedLine = line - BaseLineStableNum;
  Image_OK=TRUE;
  break;
```

```
    }
  }
//****************************************//

//***********寻找终止行(FinalLine)***********//
  if(Image_OK)
  {
    ImageStableCounter=0;
    FinalStableCounter=0;
    for(line=BasedLine+BaseLineStableNum+1;line<MaxVerticalNum;line++)
    {
      //得到搜索黑线起点
      SearchPoint = Path[line-1];
      if(SearchPoint < SearchRange)
        SearchPointL = 0;
      else
        SearchPointL = SearchPoint - SearchRange;

      if(SearchPoint > MaxPointNum - SearchRange - 1)
        SearchPointR = MaxPointNum - 1;
      else
        SearchPointR = SearchPoint + SearchRange;

      //搜索黑线边沿
      //求阈值
      MaxPoint = MinPoint = SearchPointL;
      for(i=SearchPointL;i<SearchPointR;i++)
      {
        if(Image[line][i] > Image[line][MaxPoint])
          MaxPoint = i;
        else if((Image[line][i] > 30) && (Image[line][i] < Image[line][MinPoint]))
          MinPoint=i;  //含防噪点判断
      }
      TresholdValue = (Image[line][MinPoint] + Image[line][MaxPoint]) / 2;

      //二值化
      for(i=SearchPointL;i<SearchPointR;i++)
      {
        if(Image[line][i] > TresholdValue)
          OneLine[i] = White;
        else
          OneLine[i] = Black;
      }

      //边沿化
      Edge_Counter = 0;
```

```
for(i=SearchPointL;i<SearchPointR-1;i++)
{
  if(OneLine[i] < OneLine[i+1])
  {

    OneLine[i] = Edge;
    Edge_Counter++;
  }
   else if(OneLine[i] > OneLine[i+1])
  {

    OneLine[++i] = Edge;
    Edge_Counter++;
  }
}

//判断黑线边沿
Edge_OK = FALSE;
if(Edge_Counter<5)
{
  for(i=SearchPointL;i<SearchPointR-1;i++)
  {
    if(OneLine[i] == Edge)
    {
      for(j=i+1;j<SearchPointR;j++)
      {
        if(OneLine[j] == Edge)
        {
          if(j-i>15) break;
          if(j-i<=15&&j-i>1)
          {
            EdgeCenter=(byte)(i+j)/2;
            if(OneLine[EdgeCenter]==Black)
            {
              Edge_OK=TRUE;
              EdgeL=(byte)i;
              EdgeR=(byte)j;
            }
            i=j-1;
            break;
          }
        }
      }
      if(Edge_OK == TRUE)
        break;
    }
```

```
        }
      }

      if(Edge_OK&&Image[line][MaxPoint]-Image[line][MinPoint]>30 )
      {
        Path[line] = (EdgeR+EdgeL) / 2;
        FinalStableCounter=0;
      }
      else
      {
        FinalStableCounter++;
        temp1=Path[line-1];
        Path[line]=temp1;
      }
      if(FinalStableCounter>0||line>MaxVerticalNum-1)
      {
        FinalLine = line-FinalStableCounter+1;
        break;
      }
    }

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

//*****************消畸变*****************//
    for(i=BasedLine;i<FinalLine;i++)
      Path[i]=PathTable[i*MaxPointNum+Path[i]];        //公式法与查表法相结合
//*****************************************//

//****************最小二乘法****************//
      AvaliableLines=FinalLine-BasedLine;
      MiddleLine=(BasedLine+FinalLine)/2;

      SumX=0;
      SumY=0;
      for(i=BasedLine;i<FinalLine;i++)
      {
        SumX+=i;
        SumY+=Path[i];
      }

      AverageX=SumX/AvaliableLines;
      AverageY=SumY/AvaliableLines;

      SumUp=0;
      SumDown=0;
      for(i=BasedLine;i<FinalLine;i++)
```

```
        {
          SumUp+=(Path[i]-AverageY)*(i-AverageX);
          SumDown+=(i-AverageX)*(i-AverageX);
        }
        if(SumDown==0) B=0;
        else B=(int)(SumUp/SumDown);

        /*4 月 11 日更改部分*/
        A=(SumY-B*SumX)/AvaliableLines;//使用二值法拟合出的直线位置计算偏移量


        if(AvaliableLines<5)
        {
          if(TurnSetLast>100)
             TurnSet=TurnSetLast+10;
          else if (TurnSetLast<-100)
             TurnSet=TurnSetLast-10;
          else
             TurnSet=TurnSetLast;
        }
        else
        {
          Offset=A-ImageCenter;

          if(B>0)               //左
              TurnSet=B*60+Offset*4;
          else                  //右
              TurnSet=B*60+Offset*4;
        }
        /*4 月 11 日更改部分结束*/



        if(TurnSet>500) TurnSet=500;            //左
        else if(TurnSet<-500) TurnSet=-500;     //右

        (void)Servo_SetRatio16(ServoMiddlePosition+TurnSet);

        TurnSetLast=TurnSet;
    //*****************************************//
        for(i=BasedLine;i<FinalLine-1;i++)//积分中值定理算是否出弯入弯
        {

if((Path[i]>=AverageY&&Path[i+1]<=AverageY)||(Path[i]<=AverageY&&Path[i+1]>=AverageY))
          {
            MiddlePointPosition=i;
```

```
          break;
        }
      }

  }
//*****************************************//
  else
  {
    ImageStableCounter++;
  }

  if(ImageStableCounter>1&&ImageStableCounter<100)
  {
    ImageStableCounter=1;
    if(TurnSetLast>100)
    {
      TurnSet=TurnSetLast+50;
      temp1=temp2=45;
    }
    else if (TurnSetLast<-100)
    {
      TurnSet=TurnSetLast-50;
      temp1=temp2=45;
    }
    else TurnSet=TurnSetLast;
    if(TurnSet>500) TurnSet=500;            //左
    else if(TurnSet<-500) TurnSet=-500;     //右

    (void)Servo_SetRatio16(ServoMiddlePosition+TurnSet);

    TurnSetLast=TurnSet;
  }
  else if(ImageStableCounter>=100)
    temp1=temp2=3;


    if(MiddlePointPosition<30&&TurnSet<200&&TurnSet>-200&&AvaliableLines>20)
       temp1=temp2=speed1;
    else temp1=temp2=speed2;
    speedSetLeft=temp1-(temp1*(TurnSet/100))/48;
    speedSetRight=temp2+(temp2*(TurnSet/100))/48;

}

/*** Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! ***/
for(;;){}
```

```
  /*** Processor Expert end of main routine. DON'T WRITE CODE BELOW!!! ***/
}
```