扬 州 市 职 业 大 学

毕业设计说明书

题目：基于图像传感器的智能小车设计

学　　院：电气与汽车工程学院

专　　业：机电一体化技术

班　　级：09 级 4 班

姓　　名：蔡承镇

学　　号：0902010402

指导教师：张　平

陈　剑 桥

完成时间：2012 年 5 月

摘　要

摄像头循迹智能车专业知识涉及控制、模式识别、传感技术、汽车电子、电气、计算机、机械等诸多学科。在整个智能车控制系统中，如何准确地识别道路及实时地对智能车的速度和方向进行控制是整个控制系统的关键。

本文首先对智能车的硬件进行设计，达到了低重心、大前瞻、高稳定性的目标。其次对系统的软件部分进行设计，利用动态阈值法分割处理采集到的图像，得到跑道信息，从而得到智能车的偏航角和偏航距离。综合偏航角和偏航距离两个控制量对电机进行控制，实现了入弯走内道，S 弯直线冲过的目标，大大提高了智能车的弯道运行速度。用光电编码盘检测智能车的运行速度，再根据跑道信息给定智能车的运行速度，运用增量式 PID 算法调节驱动电机转速，实现了电机的快速响应。

整个系统涉及车体机械结构调整、传感器电路设计及信号处理、控制算法和策略优化等多个方面。经过大量测试，最终确定了现有的系统结构和各项控制参数。

关键词:智能车；图像传感器；路径识别

目 录

1 绪论

1.1 引言

智能车即轮式移动机器人，是一种集环境感知、决策规划、自动行驶等功能于一体的综合智能系统，智能车集中地运用了自动控制、模式识别、传感器技术、汽车电子、电气、计算机、机械等多个学科的知识 。随着控制技术、计算机技术和信息技术的发展，智能车在工业生产和日常生活中已经扮演了非常重要的角色。近年来，智能车在野外、道路、现代物流及柔性制造系统中都有广泛运用，已成为人工智能领域研究和发展的热点。目前，智能车领域的研究已经能够在具有一定标记的道路上为司机提供辅助驾驶系统甚至实现无人驾驶，这些智能车的设计通常依靠特定道路标记完成识别，通过推理判断模仿人工驾驶进行操作。本文所述智能车就是一种自动导引小车,能够在给定的区域内沿着轨迹自动进行行进。

1953年，美国Barrett Electric公司制造了世界上第1台采用埋线电磁感应方式跟踪路径的自动导向车，也被称作“无人驾驶牵引车”(automated guided vehicle，简称AGV)。这些自动导向车主要用于自动化仓贮系统和柔性装配系统的物料运输。20世纪60年代和70年代初，AGV仍采用这种导向方式。但是，20世纪70年代中期，具有载货功能的AGV在欧洲得到了应用并被引入到美国。这些自动导向车主要用于自动化仓储系统和柔性专配系统的物料运输。在20世纪70年代和80年代初，“智能车”的应用领域扩大而且工作条件也变得多样化，因此，新的导向方式和技术得到了更广泛的研究与开发。自动导向无轨行走车辆常用蓄电池作为动力源，它是机电一体化的典型。AGV技术在汽车工业上有着广泛的用途，欧洲和美、日等国的汽车生产工业在80年代就开始大量使用AGV技术，现已成为比较成熟的技术。

智能车有着极为广泛的应用前景。结合传感器技术和自动驾驶技术可以实现汽车的自适巡航并把车开得开得又快又稳、安全可靠；汽车夜间行驶时，如果装上红外摄像头，就能实现夜晚汽车的安全辅助驾驶；他也可以工作在仓库、码头、工厂或危险、有毒、有害的工作环境里，此外他还能担当起无人值守的巡逻监视、物料的运输、消防灭火等任务。在普通家庭轿车消费中，智能车的研发也是有价值的，比如雾天能见度差，人工驾驶经常发生碰撞，如果用上这种设备，激光雷达会自动探测前方的障碍物，电脑会控制车辆自动停下来撞车就不会发生了。

1.2 本文设计方案概述

本课题选取 CMOS 摄像头作为路径检测传感器，最终要实现智能车的自动循迹功能。本文主要对智能车的控制系统和机械机构进行设计，实现了低重心、大前瞻、运行可靠的目标。最终，智能车运行时直道不震荡、过弯走内道、S 弯直线冲过，运行效果良好。下面将对智能车的整个系统进行概述。

1.2.1 控制系统

智能车系统各模块连接如图 1.2 所示：CMOS 传感器拍摄跑道图像并以 PAL 制式信号输出到 LM1881 视频分离芯片提取出行场中断信号，同时把信号给 XS128 的 AD 模块，对视频信号进行二值化，二值化后的数据和同步信号同时输入到 S12 控制核心，进行进一步处理以获得图像信息；通过对射式光电转速传感器检测车速，并采用 S12 的输入捕捉功能进行脉冲技术计算速度和路程；车体转向采用跑道斜率和偏航距离两个变量综合控制，实现了直道不震荡、过弯走内道、S 弯直线冲过的优异性能；电机转速控制采用 PID 控制，通过 PWM 控制驱动电路调整电机的功率；而车速的目标值由限速值和对不同路径的速度控制

策略综合控制。

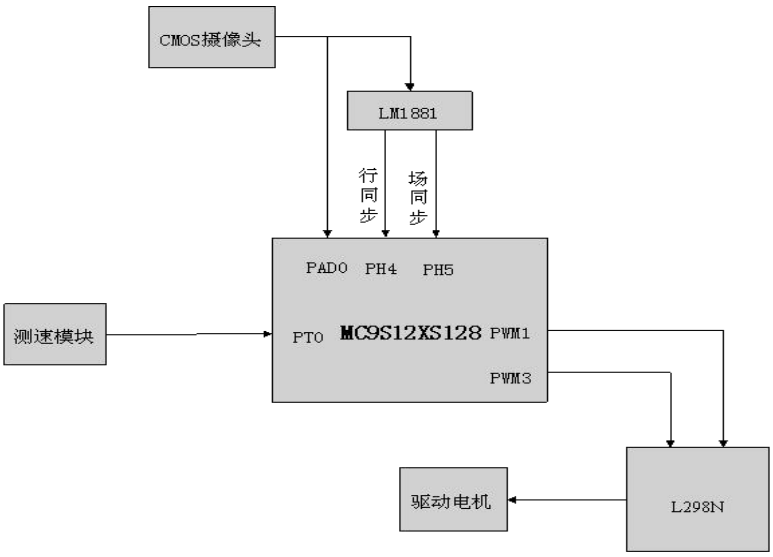


图 1.2 智能车系统模块连接图

1.2.2 电源系统

由于在跑车运行过程中阻力的变化频繁，加速制动剧烈，再加上升压 DC-DC 吸入电流高频变化，电池负载变化剧烈，输出电压也剧烈变化，且幅度很大，所以在电池端多加了滤波电容。 如图 1.3 所示

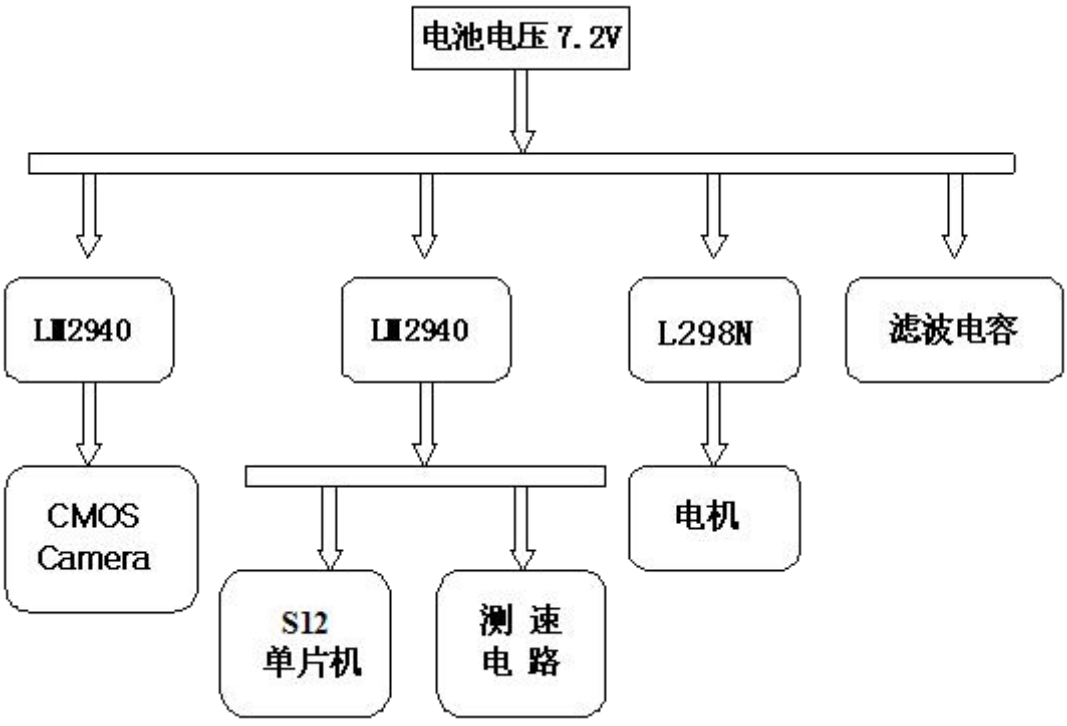


图1.3 智能车的电源系统

1.2.3 整车布局

智能车采用低重心紧凑型设计，并本着轻质量、低重心的原则，经过将近三个月的时

间，比较多套方案，采用了低位主板的布局以降低重心，同时设计了强度高质量轻的 CMOS 安装架，最终设计的车形如图 1.4 所示

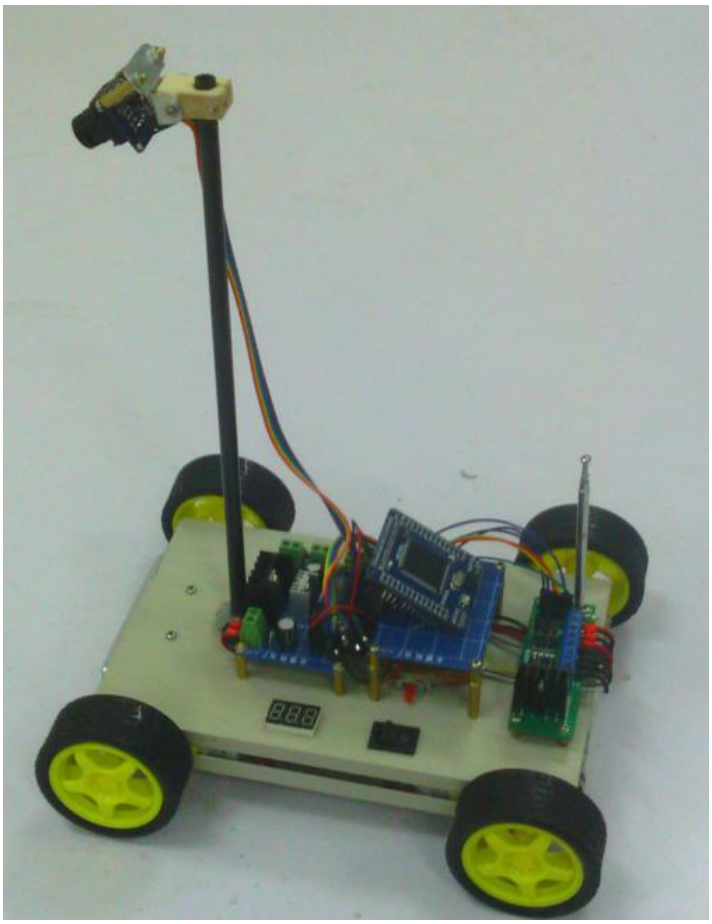


图 1.4 整车布局

1.2.4 智能车相关性能设计思想

智能车在设计过程中除了考虑动力性、操纵稳定性、响应速度等关键指标外，还应充分考虑开发、调试过程中可能遇到的种种问题。在智能车的设计、制造和配套设施及保证可靠性上做出如下要求：

a．保障智能车的可靠性，增加保护设计；
b．提高智能车的维护性，如预留备件、拆装尽可能方便、传感器重复定位性好等；
c．提高智能车和电池的续航能力，提高调试效率；
d．建立合理的调试辅助装备配套机制：维修工具、备件等；
e．建立智能车调试文档，包括硬件使用、调整说明、软硬件调试记录等。

1.3 本文主要内容

在整个智能车控制系统中，如何准确地识别道路及实时地对智能车的速度进行控制是整个控制系统的关键。路径常用的传感器检测器件主要有光电传感器和 CCD/CMOS 图像传感器。与光电传感器相比，图像传感器前瞻性更好，能检测到跑道的整体信息。光电传感器的前瞻性有了较大提高，不过在检测跑道整体信息方面摄像头还是占优势。总体而言，CCD 与 CMOS 图像传感器比其来，CMOS 传感器在功耗方面比 CCD 节省的多，故采用 CMOS 摄像头作为路径识别传感器。

本文主要研究智能车常用的 MC9S12XS128 单片机的资源配置和开发方法，设计电路图

和相应的系统软件，构建完整的智能小车摄像头导航式控制系统。目的是为了让智能车以最快的速度，沿导航线走完全部跑道。研究内容主要有以下几个方面：

首先，介绍了研究背景，阐述了控制系统的资源配置、资源需求与分配和核心处理器的寄存器,MC9S12XS128 单片机寄存器资源。相比于其它类型的单片机,16 位的 MC9S12XS128 的功能更加强大，功能引脚较多，能够很好地满足智能车控制系统的需要。

其次，设计了智能车控制系统的硬件电路，包括各个模块的电路设计方案以及相关电路。采用的方案以 MC9S12 单片机为核心，包括总体控制系统的设计，各部件需要的供电电源设计，传感器电路设计，速度检测电路的设计等。

然后，阐述了智能车机械结构调整和车体的安装。车体设计的核心思想是低重心，轻质量，提升智能车的过弯速度和加速性能。智能车机械结构调整主要是调节车的重心、前轮、后轮，使智能车在高速行走时，更加稳定。

最后，进行了软件和算法的设计。根据传感器采集的道路信息，经处理分析之后，通过增量式 PID 算法调节驱动电机转速，最后实现智能车快速的跑完全部跑道。

2 硬件设计

2.1 摄像头的选型及安装方案设计

图 2.1 CCD 和 CMOS 摄像头的比较

2.1.1 摄像头的选型

市场上常见的摄像头有CCD和CMOS两类，不管CCD或CMOS，基本上两者都是利用矽感光二极体（photodiode）进行光与电的转换。这种转换的原理与各位手上具备“太阳电能”电子计算机的“太阳能电池”效应相近，光线越强、电力越强；反之，光线越弱、电力也越弱的道理，将光影像转换为电子数字信号。比较 CCD和CMOS的结构，ADC的位置和数量是最大的不同。简单的说，CCD每曝光一次，在快门关闭后进行像素转移处理，将每一行中每一个像素（pixel）的电荷信号依序传入“缓冲器”中，由底端的线路引导输出至CCD旁的放大器进行放大，再串联ADC输出；相对地，CMOS的设计中每个像素旁就直接连着ADC（放大兼类比数字信号转换器），讯号直接放大并转换成数字信号。

关于摄像头的选择我们从以下几个方面分析：

2.1.1.1 重量和体积

图 2.2 CMOS 摄像头

图 2.3 CCD 摄像头

图2.2是CMOS摄像头，图2.3是CCD摄像头。很明显，CMOS体积可以做到非常小，而CCD，因为存在复杂的驱动电路，PCB面积至少需要4cm*4cm的空间。重量上，因为体积和远见数量的区别，CMOS摄像头也优于CCD，大约有30%-50%的重量差别。这个重量的差异，随摄像

头安装的位置的不同而对整个跑车的稳定性有不同的影响。摄像头安装的位置越高，摄像头重量对智能车重心的影响越高。好在目前的趋势看来，摄像头的安装高度越来越低，以至于这种重量的区别对智能车重心的影响也随之减小。另外 CMOS摄像头有一种数字式的摄像头，它在重量和体积上和CCD摄像头不相上下，因为其引出了三十多根信号线，实际安装的重量和体积较CCD摄像头还要臃肿，这是个特例。

2.1.1.2噪声

在图像噪声方面，CCD明显好于CMOS摄像头。这一点是这两种技术本身先天决定的，CMOS的工艺无法保证几万个感光点的一致性，而CCD因为所有感光点均通过一个电荷-电压转换器，一致性可以得到很好的保障。图像的噪声，在智能车运行中，影响不是特别大。因为拍摄图像只有全白全黑两种，来自传感器的电平轻微跳动，还不至于影响到跑道的识别。同时在一个设计不良的采集电路中，来自A/D模块的误差，也足以掩盖来自传感器的误差噪声，这一点却没有引起很多人的重视。

因此，我认为图像传感器的噪声不足以影响智能车对黑线的识别，因此可以在摄像头选型时，作为优先级较低的一个参考标准。而在其他需要高分辨率图像采集和识别的应用中，两种传感器的图像质量差别，还是需要引起一定的重视的。

2.1.1.3解析度

针对智能车使用单片机对图像型号进行采集的应用，解析度的高低直接关系到采集的难度。我们知道，标准的视频信号每秒有50或60场图像，这是固定不变的。越高的分辨率，包含了越多的行数。即在20ms的时间内，有更多的行数，以至于每行图像所占用的时间越少。一个30万像素的摄像头，每一场输出312.5行图像，行周期为64us。我们希望采集的分辨率越高，就需要在64us的时间内采样更多的点数。这就需要更快的A/D转换器，例如片外高速视频A/D。既然单片机的速度远远跟不上视频信号的速度，那么选用越低分辨率的摄像头，图像采集的效果反而越好，因为每行图像有更多的时间留给MCU进行采集。目前能买到的摄像头中，30万像素的是最佳选择，而130万像素就不适合了。

2.1.1.4感光度

感光度的差别，在智能车应用中非常重要。所谓感光度，就是对光线的敏感程度。感光度越高，其在较低光照条件下能获得越明亮的图像，也直接决定了感光器件拍摄图像的曝光时间长度。CCD的感光度大约是CMOS的3-10倍，这时在同样的光照条件下，要拍摄同样亮度的图像，CMOS的曝光时间将会是CCD的3-10倍。随着车速的不断提高，这一点越来越影响到摄像头小车的性能。我们知道，拍照时手不能抖这个道理，抖了画面就糊了。这个简单的常识在智能车中同样适用。在高速下，曝光时间越短，所拍摄的画面越清晰，曝光时间越长，拖影越严重，画面越模糊。根据这个原理，同样光照条件下，CCD在高速运动下拍摄的画面效果应该要比CMOS高一个等级。这对于目前每秒前进4 米的模型车而言，是至关重要的。

2.1.1.5电能和功耗

影响CCD在嵌入式中应用的一个重要原因，就是供电和功耗。首先，CCD需要12V供电，对智能车而言，这就意味着需要额外给CCD设置一个升压电路。这意味着成本和体积的增加。还好，越来越多的设计者在智能车电机驱动部分开始采用MOS管驱动的方式，而MOS管栅极电压很多都需要12V，因此两路电源合二为一，提高了电源的利用率。不能否认，CCD确实是耗电大户，工作时散发的热量足以烫手。对于电池供电的系统来说，每一点电能的节约都至关重要。从节能角度来说，CCD真的有点不适合，随着技术的发展，目前CCD单板机需要的电流已经从开始的150mA降到了现在的50mA，但是别忘记，CCD的供电是12V，这么看来，功耗依然很大。而 CMOS用5V供电和微功耗，在功耗方面与CCD比起来则有很大优势。

综合上述几个方面，可以看出解析度、感光度，电能和功耗是我们要着重考虑的几个

方面。选型时，考虑电机高速运行时，功耗较大，故节省电能成为要考虑的一个重要问题。虽然CMOS曝光时间较大，但是在只有黑白两种颜色的跑道上，因曝光不足造成的影响应该可以降低很多，另外我们采用的动态阈值法，一定程度上也降低了曝光不足的影响。综合考虑，我选择的型号为OV5116的CMOS动态集成摄像头。

OV5116动态集成摄像头具有以下的特点和优势：

a.摄像头电路板的尺寸 38*30mm。

b.5V 供电电压。

c. 输出五条信号线，分别是行中断信号、场中断信号、图像模拟信号、图像二值化数字信号 SU,SUO、图像动态阀值镜像信号。

d. 集成了模拟转化为数字式的电平信号 LM393/ AD8032。

e.PAL 制，每秒 25 帧，一帧两场，那么每秒就有 50 场。意味着 16.7MS 左右就有一幅图像产生。

f.集成 LM1881 视频分离芯片，直接输出 VS 场同步信号、HS 行同步信号、ODEV 奇偶场识别信号。

图 2.4 OV5116 引脚图

从左到右：1 脚接 5V 电源正极，2 脚接电源负极，3 脚接单片机 ADO，5 脚接单片机 PT2，6 脚接单片机 PT1

经过串口和调试程序得到的图像信息。

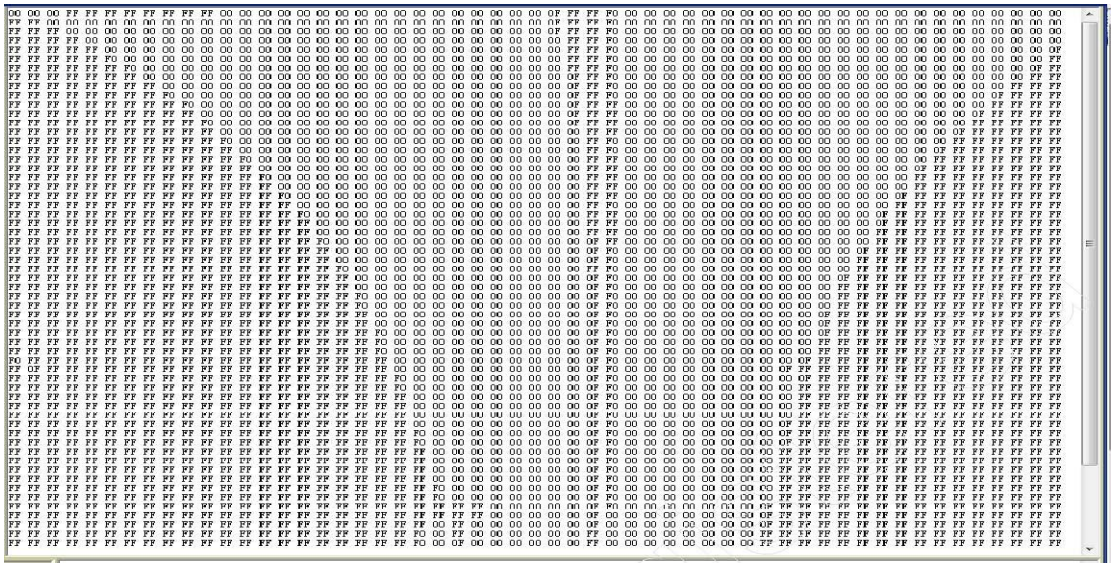


图 2.5 安装条件下车载摄像头获得跑道图像

摄像头安装在 28cm 的高度，可采集 51cm 远 50cm 宽的跑道。

对于 XS128 单片机，如果不超频的话，此时 A/D 模块最快转换速度为 8μs，而摄像头在 PAL 制式下每个扫描行约为 52μs，故每行采集的点数最多为 8 个[20]；当单片机超频到 32MHz 总线频率下的行有效采样率可达 66，根据图 2.3 可以估算其图像中间行的跑道心分辨率约为 13.5mm，此时的前瞻距离为 60cm，完全满足我们要求的分辨率及前瞻距离。

2.1.2 传感器的供电电路

型号为 OV5116 的 CMOS 摄像头需要给其提供稳定的 5V 电源，由于 LM2940 的稳压的线性度非常好，所以选用 LM2940-5 单独对其进行供电，LM2940 具有纹波小、电路结构简单的优点,另外为了降低 DC-DC 升压造成的杂波，电路中加了多个滤波电容。如图 2.4 所示。

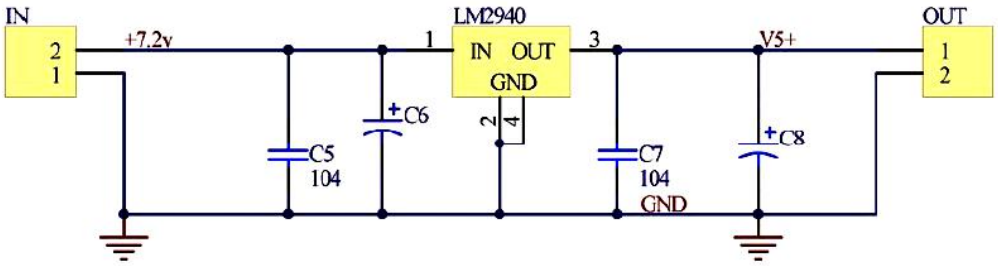


图 2.6 传感器的供电电路

2.1.3 视频处理模块

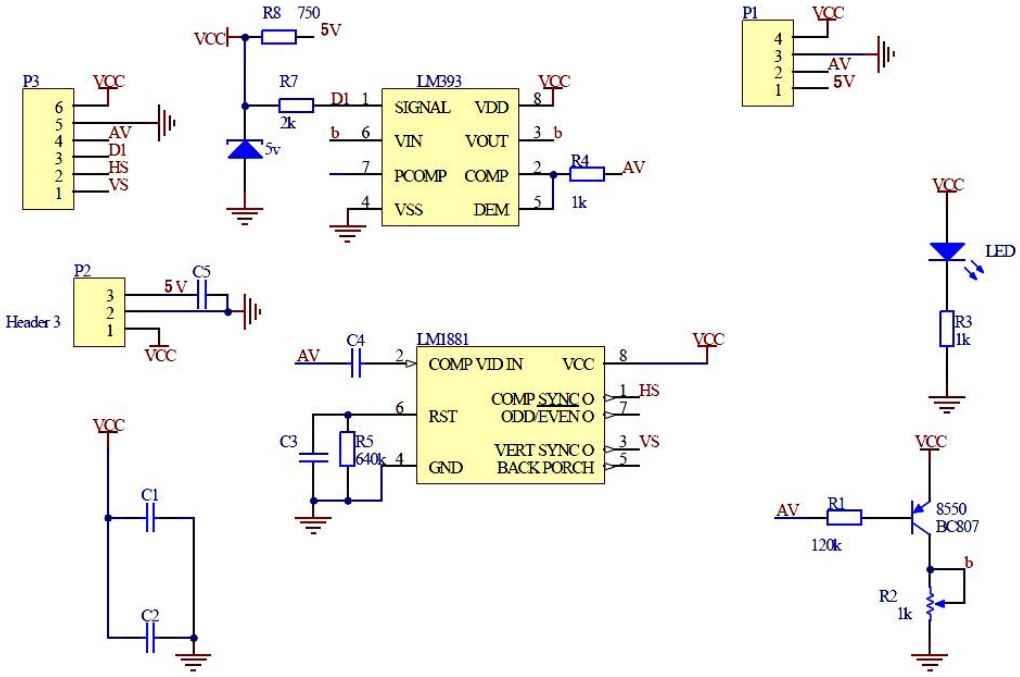


图 2.7 行场同步信号分离电路

用摄像头实时的采集道路信息，再经过单片机的内部处理，来控制智能车的转向和电机的加减速。由于比跑道路信息比较简单，只有黑白两种颜色，因此选用了一款市场上比较常见的 CMOS 摄像头 OV5116。

为了采集图像信息，CPU 需要根据行、场同步信号启动 A/D 转换器，具体过程就是在等待

到行同步信号后启动 AD 转换器，同时通过定时器设定 AD 采集的时间，然后等待下一次行中断并启动。由于视频信号的变化很快，所以需要另外设计同步分离电路。在本方案中，使用了 LM1881 视频同步分离芯片，它是针对电视信号的视频同步分离芯片，它可以直接对电视信号进行同步分离，准确地获得所需的视频图像信号。使用者可根据需要对该同步信号进行时序逻辑控来获取场同步和行同步信号。本文设计的系统，是将此同步信号连接到单片机的中断输入口，即 PH 口。

因为采集的图像无论奇场、偶场均算作一场，故我们只需采集场中断信号就可以了，这样降低了图像采集的复杂度，赢得了处理时间。对于行、场同步信号的分离电路如上图 2.5 所示。

2.1.4 传感器安装方案设计

对于 CMOS 的安装位置主要有安装高度和俯视角度两个关键参数。增加安装高度同时增大俯视角度可以在保证前瞻距离的前提下减小图像的畸变，使远端跑道图像变宽，便于单片机识别，保证识别稳定且有足够的前瞻。所以有很多代表队都把摄像头的位置安装的很高。但是，将 CMOS 安装高度增大的同时，容易使整车的重心提高，降低了侧翻极限，使智能车车在高速过弯的时候容易发生摇晃、颠簸甚至倾覆的危险。

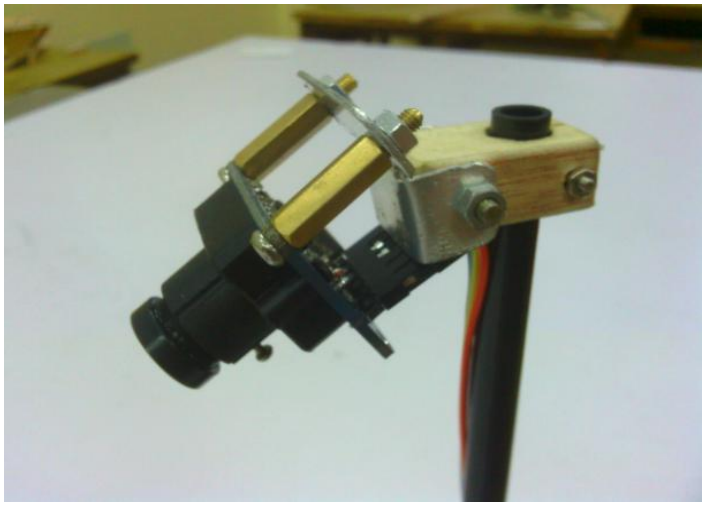


图 2.8 摄像头实际安装效果图

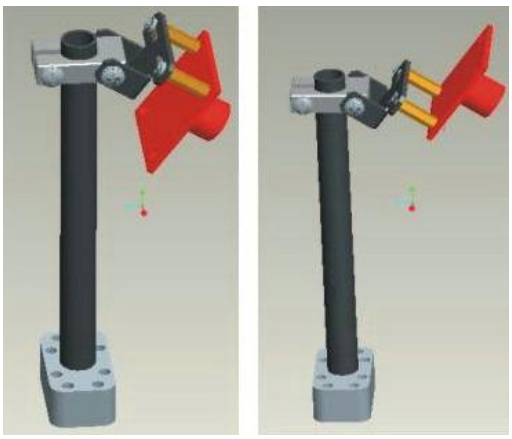


图 2.9 安装架设计图

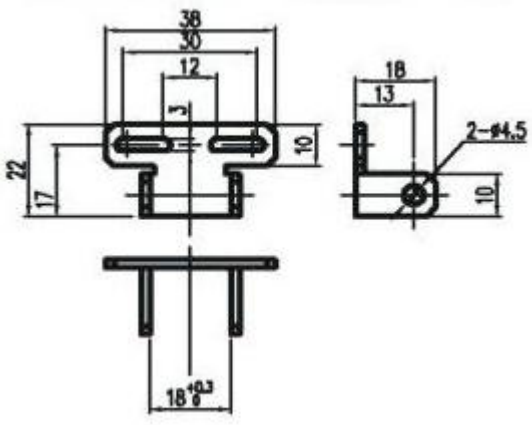


图 2.10 安装架尺寸图

综合考虑各方面的因素，最终安装高度确定为 25cm，俯视角度约为 30 度。为了降低 CMOS 高位安装对整车中心高度的影响，我专门设计了质量轻强度高的铝合金安装架，而且采用了黑色碳纤管做主桅，使高位安装的零件质量控制在 60g 以下。最终效果如

2.2 智能车车体设计

车体采用硬度较大的铝合金板材拼合制成，车体更牢固。采用轴承连接摇头电机和摄像头支撑杆，摄像头转动灵活。支撑杆选用的是质地轻、刚性大，直径小而非常坚固耐用的黑色碳纤管。自主设计了摄像头的支架，可以调节俯仰角。

2.2.1 车体布局

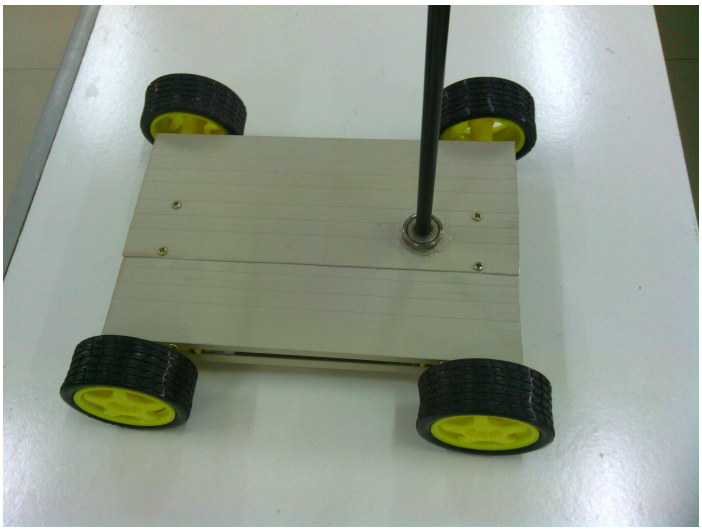


图 2.11　智能车车体

为了提高智能车高速工况下的操纵稳定性，在整车布局上主要在做了三方面工作：低位主板安装、降低摄像头安装高度和 CMOS 安装的轻量化设计、控制整车质心尽量靠近中心转向点。经过上述设计的智能车质心基本在中心转向点附近，所以车体在侧向加速度很大的极限工况下会先发生侧滑而不会发生侧翻。

2.2.2 传感器支架的设计安装

为了降低整车重心，需严格控制CMOS及其安装架的重量。首先是设计了轻巧的铝合金CMOS夹持组件，并采用了黑色碳纤管作为安装CMOS的支撑杆，这样可以获得最大的刚度质量比。在底座设计上不仅选用了铝合金来减轻重量，而且还设置了减重孔，最大限度的压缩零件重量。在CMOS安装参数的可调整性设计上，采用了平行双桅和可加紧的滑套结构。这样不仅是CMOS能上下滑动调整高度，而且能够保证在滑动过程中不发生左右横摆。而且这个定位关系是从底座安装孔获得并以较高定位精度向上传递的，从而获得良好的重复定位精度。使得能够方便的将CMOS架拆下和装上，不必顾虑摄像头的位置会发生改变。使CMOS便于拆卸和维修，具有快速保障能力。

2.2.3 主板安装

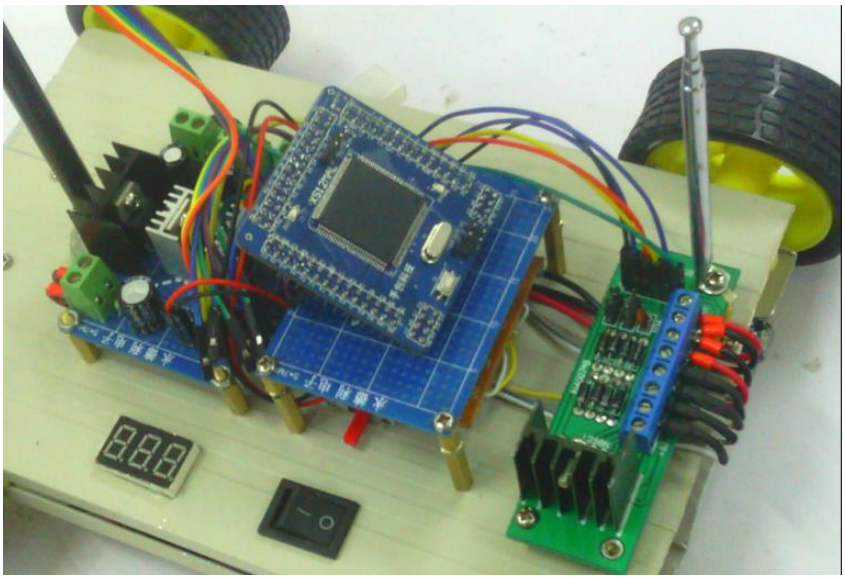


图2.12　主板安装

如图2.7所示，主板安装紧贴车体底盘，放在第一层板上方，用铜柱固定。这样做的目的就是控制车体的重心，防止过弯时发生侧翻现象。电路板的形状是60×80的长方形万能板，安装时非常方便。各电路板之间通过杜邦线连接，用扎带整齐地布在车体上。

2.3 电路设计

该智能车系统包括以下几个模块：最小系统板模块、电源模块、电池电压检测模块、串口模块、视频分离模块、测速模块、电机驱动模块。智能车的电路板主要分为上下两层，上面一层为单片机最小系统板，下面一层主要包括电源模块、电池电压检测模块，电机驱动模块及串口通信模块。测速模块安装在电机底部。各电路板之间通过杜邦线连接，安装拆卸方便，能够快速组装。

本方案将MCU 最小系统运行在单片模式下，它由以下模块组成：供电电路、时钟电路、复位电路、BDM 下载口。

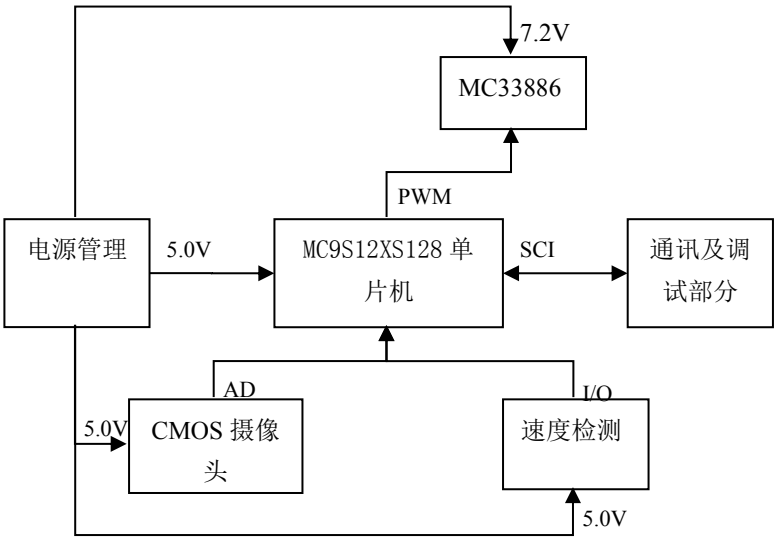


图 2.13　系统电路的硬件结构

2.3.1 电源模块

该小车由 7.2V 电池供电，但是该单片机需要的是+5V 电源。单片机 I/O 模块的供电是+5V 电压，而 S12 单片机片内使用 2.5V 电压，片外 I/O 使用 5V 电压，较低的片内电压使 CPU 运算速度快、功耗低；较高的 I/O 电平有利于抗外界干扰，所以系统使用+5V 电压供电，能够工作在复杂的环境之中。该 S12 系列的单片机内部集成了电压调整器模块，该模块产生单片机内部需要的其他电压，为了稳定＋5V 电压，克服电机运行所产生的电源电压变化，该供电电路加上一些储能电容和去耦电容。此外给电源系统接一个 LED 指示灯，是为了以后能够正确地观察电路工作情况。

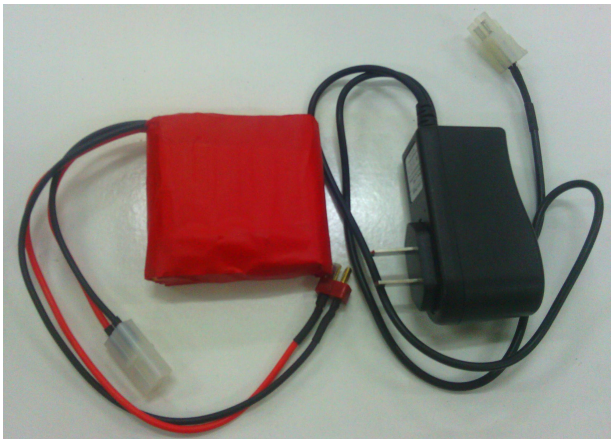


图 2.14　电源与充电器

电源模块为系统其它各个模块提供所需要的电源，主要为最小系统板模块、电源模块、电池电压检测模块、串口模块、摄像头模块、视频分离模块、测速模块、电机驱动模块供电。供电电压有 7.2V、5V 两种电压值。设计中，除了要考虑到电压范围和电流容量等基本参数之外，还要在电源转换效率、降低噪声、防止干扰和电路简易程度等方面进行优化。可靠的电源方案是整个硬件电路稳定可靠运行的基础，故在所设计的电路板中加入了电池电压检测电路，实时显示电池电压。

其原理图如下：

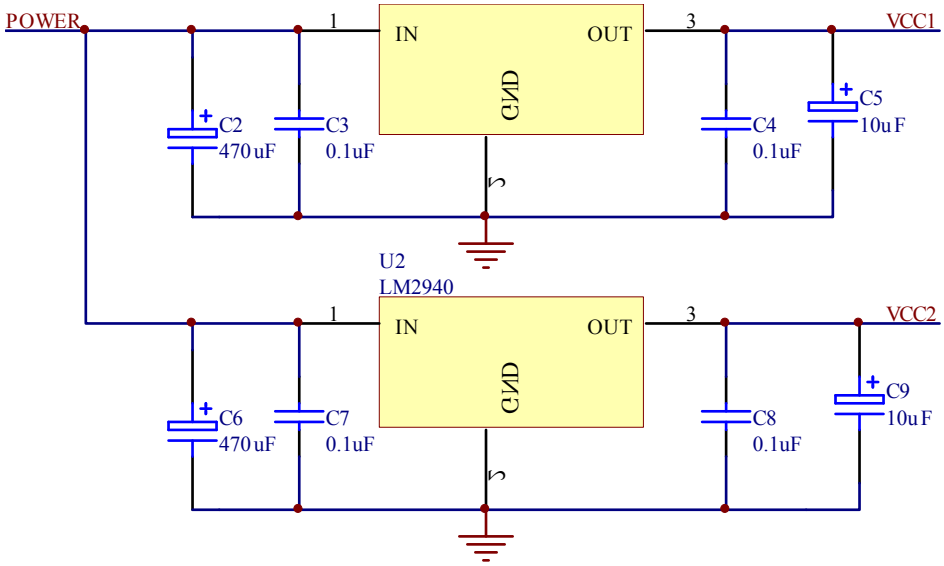


图2.15　稳压电源原理图

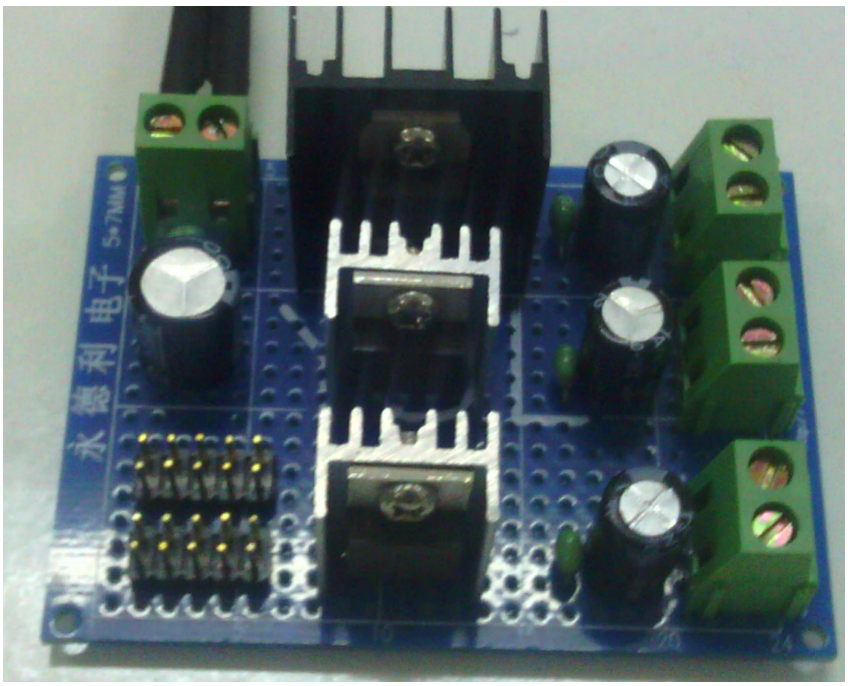


图 2.16　稳压电源实物图

其供给电压必须为 7.2V 以上，否则不能正常带动负载。

2.3.2 最小系统模块

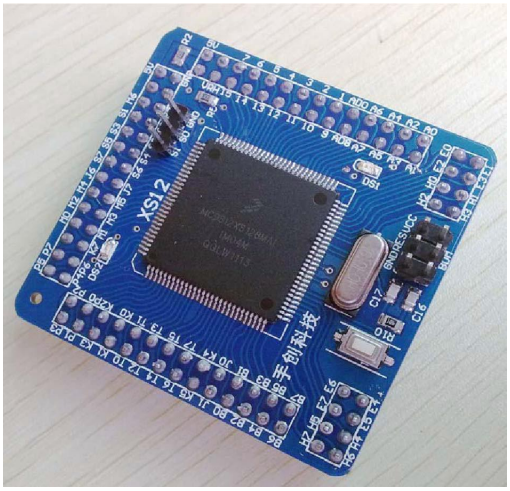


图 2.17　最小系统板

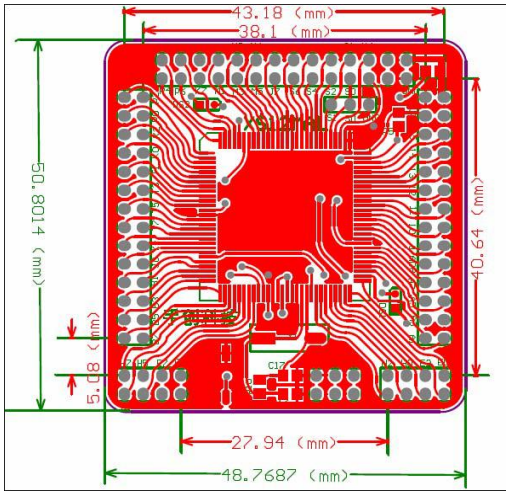


图 2.18　印刷电路板

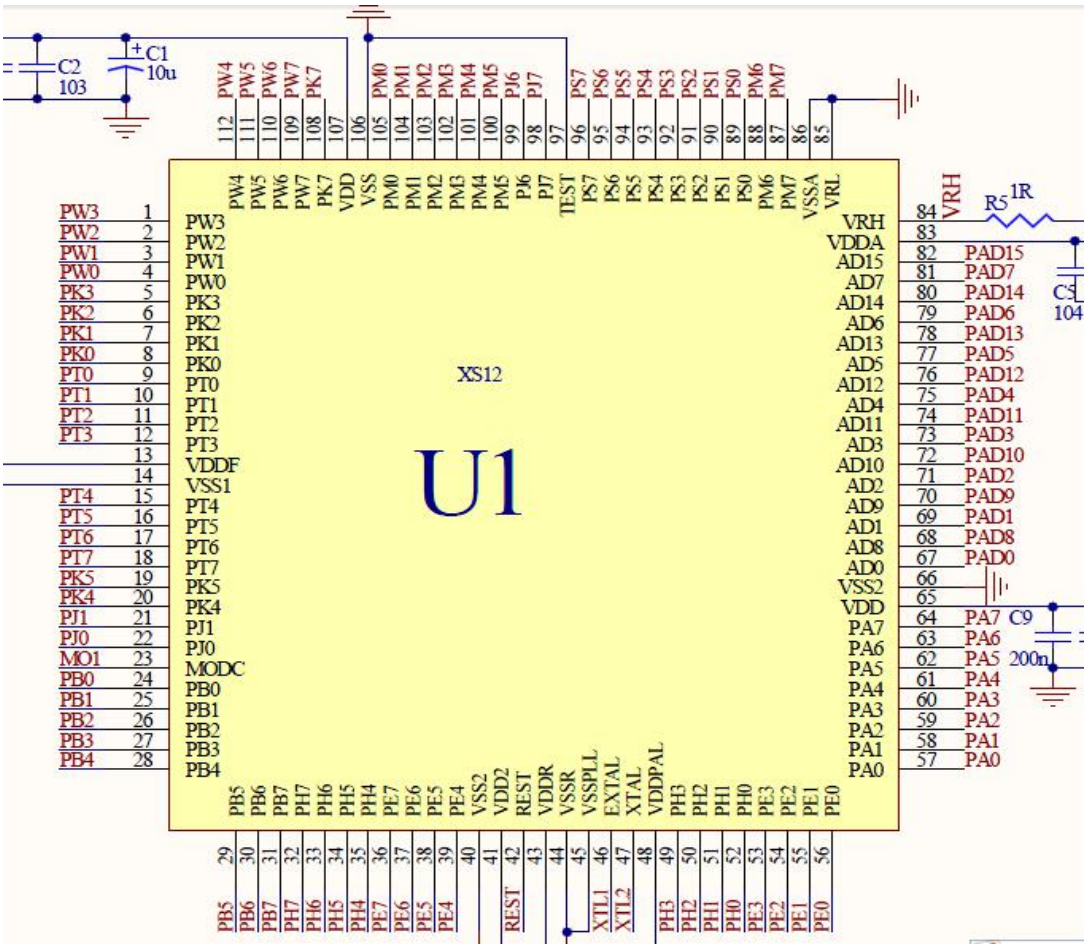


图 2.19　最小系统板引脚分布图

系列单片机是在 S12XE 系列基础上去掉 XGate 协处理器的单片机。该系列单片机采用 S12X V2 CPU 内核；可运行在 40MHz 总线频率上；
有 ECC 模块；
1 个 SPI 模块；
8 路 16 位计数器；
1 个 CAN 总线模块；
4 路外部事件触发中断输入端口；
2 个 SCI 串行通信模块支持 LIN 总线；
8 路 PWM；
16 路 8 位、10 位、12 位 AD 转换时间 3us；
112 管脚 LQFP 贴片封装

2.3.3 时钟电路

时钟电路是由石英晶体振荡器和一些电容、电阻组成，，虽然简单的单片机可以使用集成到单片机内部的 RC 振荡电路产生单片机工作所需要的时钟，但是这种简单的时钟电路频率的稳定性得不到保证，而外部晶振能提供稳定的频率为 16M 的振荡，该单片机内部的压控振荡器和锁相环（PLL）得到一个高稳定的时钟源。PLL 电路每锁定一个工作频率就需要一个特殊外部滤波电路，以消除鉴相器产生的噪声电压，从而使压控振荡器输出稳定的振荡频率。PLL 的滤波电路应设计得具有良好的抗干扰性能，它的元器件取值应经过

FREESCALE 提供的计算软件来设计，在 PCB 布局的时候用相对独立的地线将其包围，离高频线路远一点。把由外部得来的振荡提高为 24M 的总线时钟和 48M 的 CPU 工作时钟，供给单片机使用稳定，可调的时钟信号。

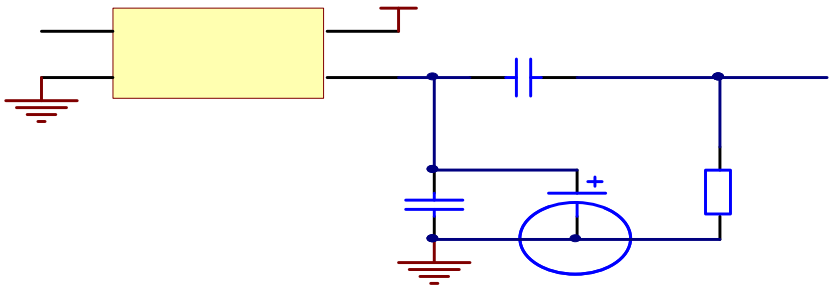


图 2.20　时钟电路

2.3.4 串口模块

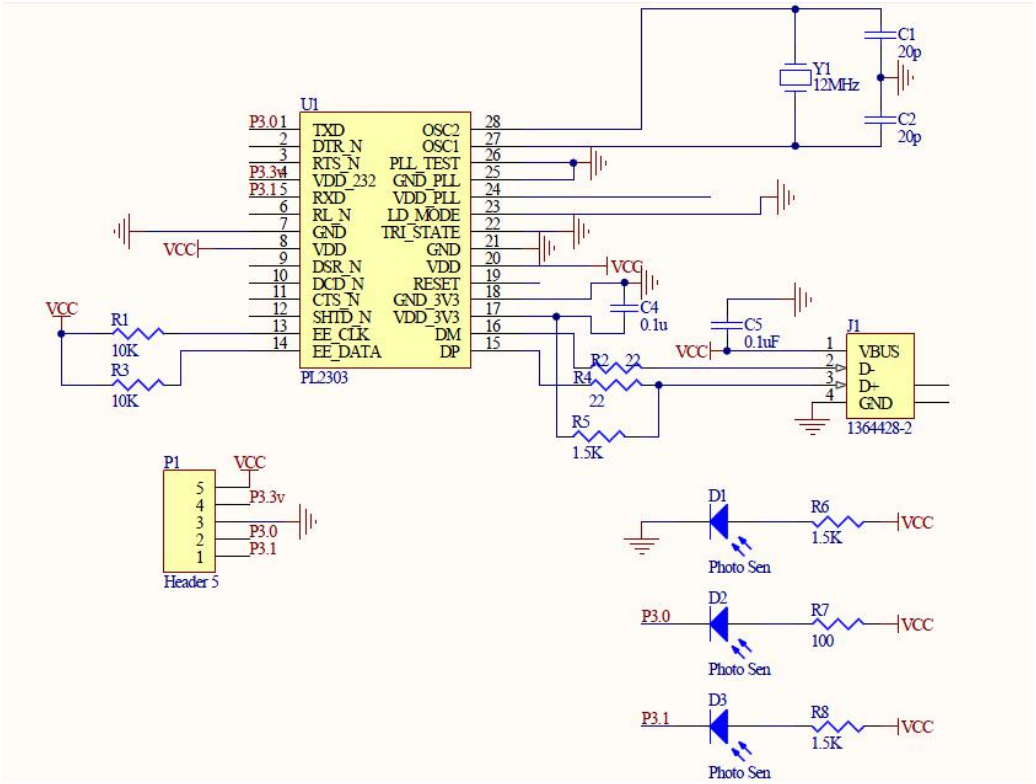


图 2.21 USB 转串口原理图

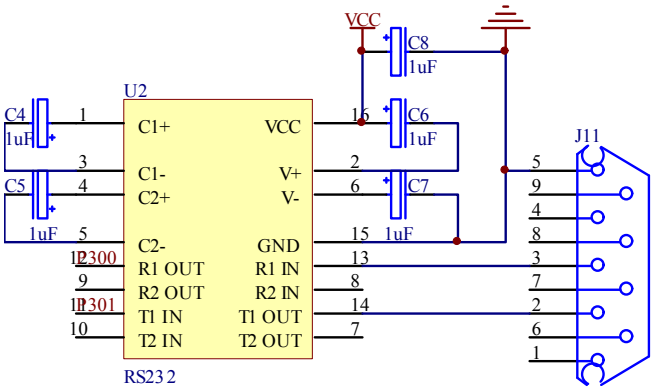


图 2.22 串口电平转换电路

图 2.23　USB 转串口实物图（PL2303）

智能车的行走路线是根据跑道上的黑线确定的，根据摄像头采集到的路况信息控制电机。而为了获取路况信息，就要求单片机能够和 PC 机通信，而比较方便有效的方式就是串行通讯。同时在进行系统调试的时候：比如 PID 参数测定，采用传感器记忆数据时也要用到串口。所以，串口电路必不可少。

2.3.5 测速模块

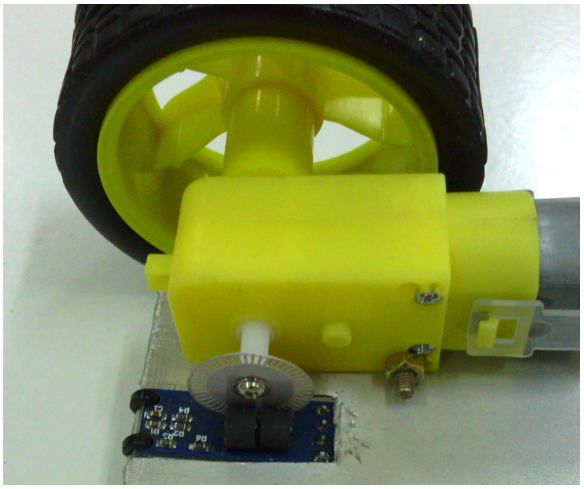


图 2.24 编码器安装示意图

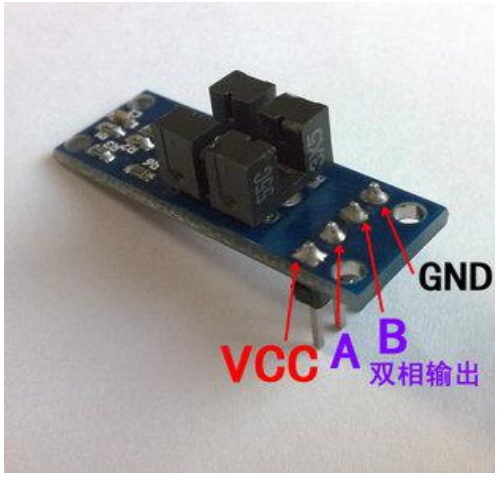


图 2.25　编码器 引脚图

此编码器由 50 线编码盘、光电一体管及信号转化电路组成

编码盘的尺寸：外径 17mm，内径 2mm，线宽 0.4mm，线长 2mm，孔个数 50，厚度 0.18mm。

组成材料为铝镁合金因而具有质量轻的优点，减轻车的重量；采用德国的激光机加工制造，检测精度更高。

光电一体管及信号转化电路输出脉冲信号，可以直接接单片机的 PT7 中断口，无需加任何外围电路，其使用方法与欧姆龙编码器相似。光电一体管及信号转化电路总的四条线，两条分别是电源线（5V 和地线），另外该编码器无需接发射驱动电路和信号转化电路。编码盘体积小，能够直接安装在驱动电机轴上。

安装时注意在固定之前，检测光电模块电路——先通电（5v），信号线接万能表红色正极端，负极接万能表黑色负极端，把万能表调至电压端，用小纸条挡住对射，检查万能表有没有电平的变化。接着安装光电一体模块电路，没有问题后就可以把光电一体模块电路固定在车体上，（注意安装位置，使得码盘转动的时候，信号线有良好的输出）。采用 AB 胶或者热熔胶来固定。

2.3.6 电机驱动模块

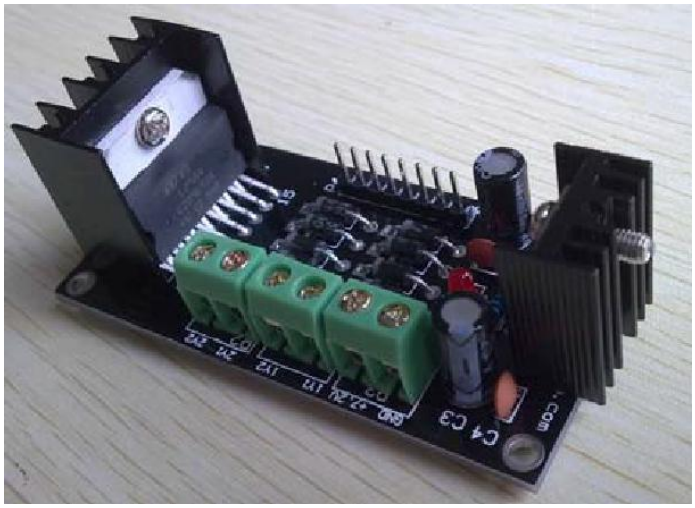


图 2.26　电机驱动实物图

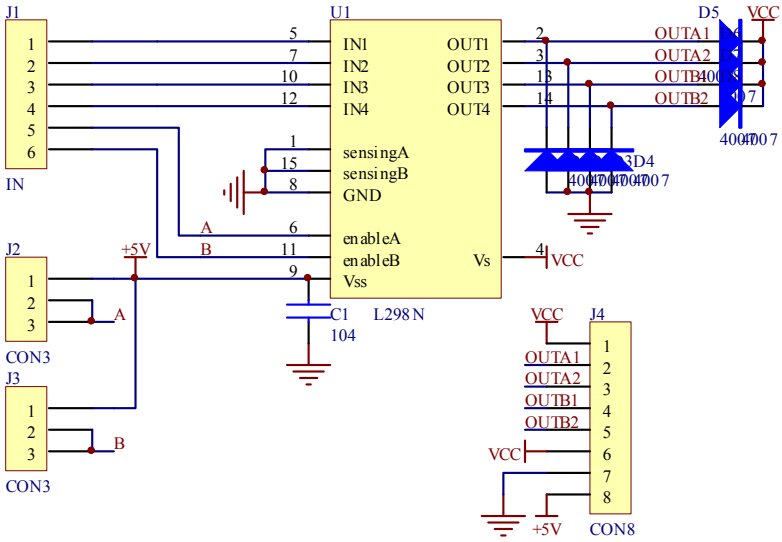


图 2.27　电机驱动原理图

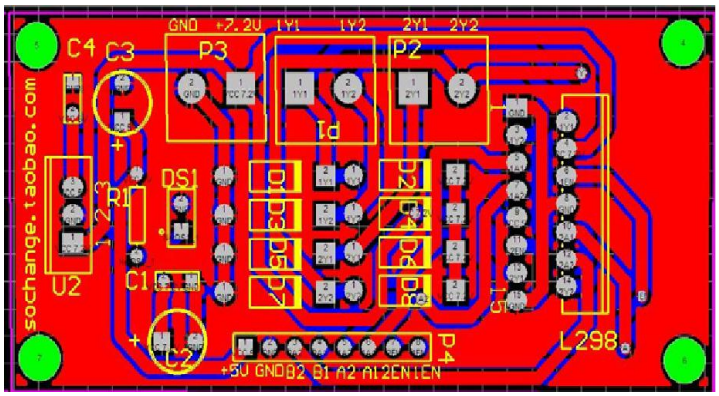


图 2.28　电机驱动印刷电路板

2.3.6.1 电动小车的电机驱动及控制

一个电动小车整体的运行性能，首先取决于它的电池系统和电机驱动系统。电动小车的驱动系统一般由控制器、功率变换器及电动机三个主要部分组成。电动小车的驱动不但要求电机驱动系统具有高转矩重量比、宽调速范围、高可靠性，而且电机的转矩-转速特性受电源功率的影响，这就要求驱动具有尽可能宽的高效率区。我们所使用的电机一般为

直流电机，主要用到永磁直流电机、伺服电机及步进电机三种。直流电机的控制很简单，性能出众，直流电源也容易实现。本文即主要介绍这种直流电机的驱动及控制。

2.3.6.2 电机驱动的选择

针对不同的电机，应该选择相对应驱动。简单地来说，功率大的电机应该选用内阻小，电流容许大的驱动，功率小的电机就可以选用较低功率的驱动。电机驱动较常规的方法是采用 PWM 控制。驱动电路既可以直接采用 MC33886 电机驱动芯片，也可以采用大功率 MOS 管来自行设计电机驱动电路，还可以用场效应管搭建 H 桥电路。

方案一：

采用大功率 MOS 管组成电机驱动电路;用这个方法电路非常简单,控制只需要一路 PWM，在管子上消耗的电能也比较少，可以有效地避免多片 MC33886 并联时由于芯片分散性导致的驱动芯片某些片发热某些不发热的现象。但是缺点是不能控制电机的电流方向，在小车的刹车的性能的提升上明显有弱势，而且电流允许值也比较小。

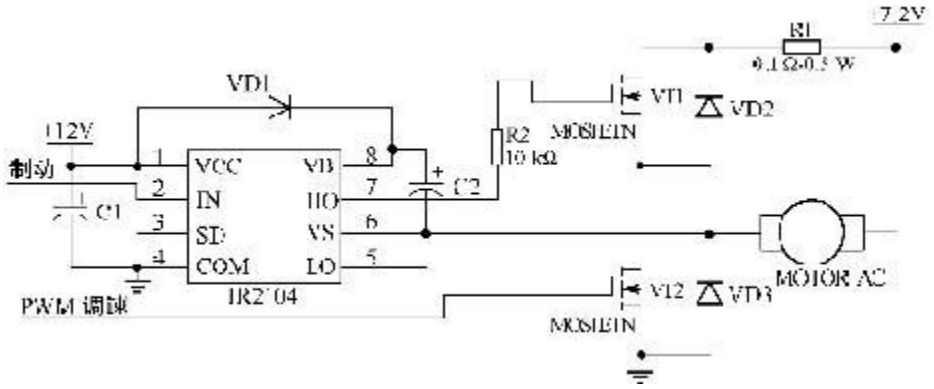


图 2.29　采用大功率 MOS 管组成电机驱动电路

方案二：

通过电机驱动模块，控制驱动电机两端电压来对模型车加速运行，或对其进行制动，采用飞思卡尔半导体公司的集成桥式驱动芯片 MC33886。MC33886 最大驱动电流为 5A，导通电阻为 140 毫欧姆，PWM 频率小于 10KHz，具有短路保护、欠压保护、过温保护等功能。体积小巧，使用简单，但由于是贴片的封装，散热面积比较小，长时间大电流工作时，温升较高，如果长时间工作必须外加散热器，而且 MC33886 的工作内阻比较大，又有高温保护回路，使用不方便。

方案三：

图 4.2 就是一种简单的 H 桥电路，它由 2 个 P 型场效应管 Q1、Q2 与 2 个 N 型场效应管 Q3、Q3 组成，P 型管在栅极低电平时导通，高电平时关闭；N 型管在栅极高电平时导通，低电平时关闭，场效应管是电压控制型元件，栅极通过的电流几乎为“零”。正因为这个特点，在连接好下图电路后，控制臂 1 置高电平（U=VCC）、控制臂 2 置低电平（U=0）时，Q1、Q4 关闭，Q2、Q3 导通，电机左端低电平，右端高电平，所以电流沿箭头方向流动。设为电机正转。

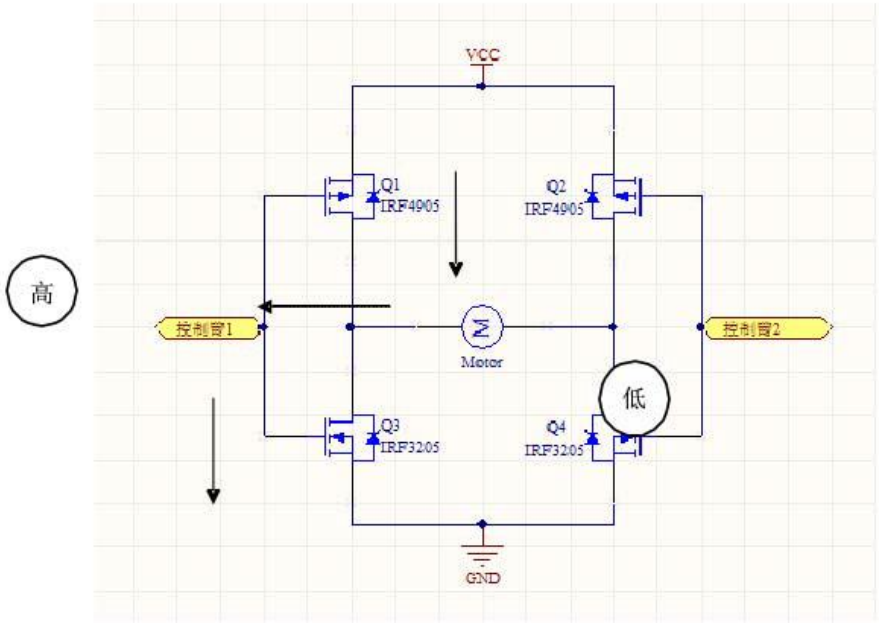


图 2.30　H 桥电路（高）

控制臂1置低电平、控制臂2置高电平时，Q2、Q3关闭，Q1、Q4导通，电机左端高电平，右端低电平，所以电流沿箭头方向流动。设为电机反转。

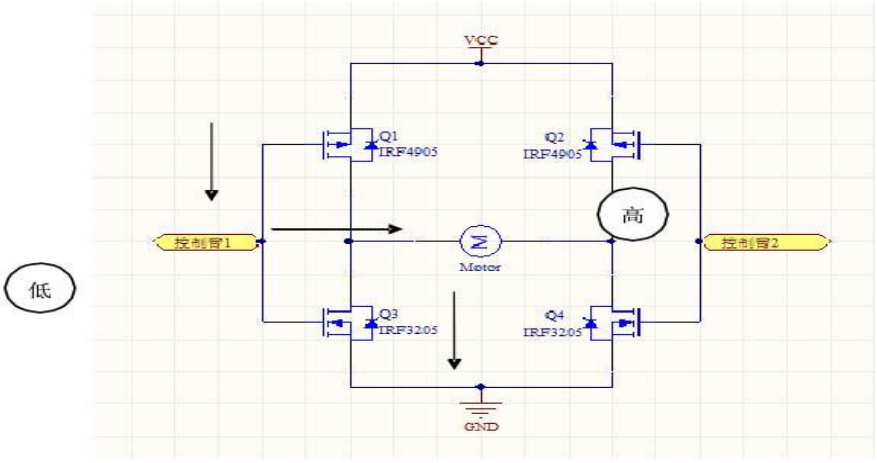


图 2.31　H 桥电路（低）

当控制臂 1、2 均为低电平时，Q1、Q2 导通，Q3、Q4 关闭，电机两端均为高电平，电机不转；当控制臂 1、2 均为高电平时，Q1、Q2 关闭，Q3、Q4 导通，电机两端均为低电平，电机也不转。所以，此电路有一个优点就是无论控制臂状态如何，H 桥都不会出现“共态导通”（短路），很适合我们使用。

方案四：

采用两片 BTS7970B，BTS7970，BTN7960，BTN7971 搭成 H 桥来驱动电机，原理跟 MOS 管搭建 H 桥相似，此种驱动方法驱动电流大，可以达到 18A 左右。

参考电路图如下：

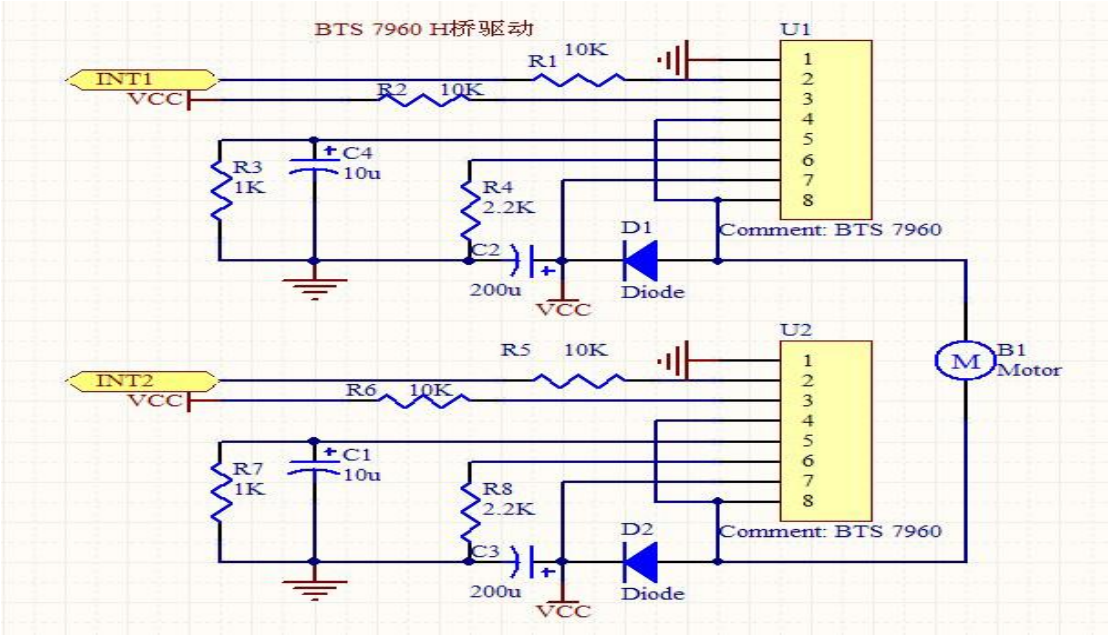


图 2.32 BTS7970B 驱动电路

目前，在智能车的应用上，方案三、方案四比较常用。

总结：以上主要分析了电机的全桥式驱动电路，这是直流电机调速使用最多的调速方法。目前市场上有很多种电机驱动的集成电路，效率高，电路简单，使用也比较广泛，但是其驱动方法大多与全桥式驱动一样。PWM 控制方法配合桥式驱动电路，是目前直流电机调速最普遍的方法。

2.3.7 抗干扰处理

在电子系统设计中，应充分考虑并满足抗干扰性的要求，避免在设计完成后再去进行抗干扰的补救措施。

2.3.7.1 抑制干扰源

抑制干扰源就是尽可能的减小干扰源的 du/dt **错误！未找到引用源。**、di/dt **错误！未找到引用源。**，这是抗干扰设计中最优先考虑和最重要的原则。 减小干扰源的 du/dt **错误！未找到引用源。**主要是通过在干扰源两端并联电容来实现。减小干扰源的 di/dt **错误！未找到引用源。**则是在干扰源回路串联电感或电阻以及增加续流二极管来实现。

本设计中所采取的措施有：电机两端增加续流二极管，消除断开线圈时产生的反电动势干扰；同时给电机加滤波电路；7.2V 电源两端并联两个电感，起到滤波的作用，其中一个大电容，大小为 470μF，一个小电容，大小为 0.01μF(常用的 103)。布线时避免 90°折线，减少高频噪声发射。

2.3.7.2 切断干扰传播路径

干扰的传播路径可分为传导干扰和辐射干扰两类，所谓传导干扰是指通过导线传播到敏感器件的干扰。高频干扰噪声和有用信号的频带不同，可以通过在导线上增加滤波器的方法切断高频干扰噪声的传播，有时也可加隔离光耦来解决。电源噪声的危害最大，要特别注意。所谓辐射干扰是指通过空间辐射传播到敏感器件的干扰。一般的解决方法是增加干扰源与敏感器件的距离，用地线把它们隔离和在敏感器件上加蔽罩。

本设计中所采取的措施有：充分考虑电源对单片机的影响，给单片机电源加滤波电路，同时对单片机的 5V 电源单独供电；单片机的 I/O 口与外部器件（控制电机、传感器、开关等）之应加光隔隔离；电路板合理分区（如强、弱信号，模拟、数字信号分开）。

2.3.7.3 提高敏感器件的抗干扰性

提高敏感器件的抗干扰性是指从敏感器件这边考虑尽量减少对干扰噪声的拾取，以及

从不正常状态尽快恢复的方法。本设计中所采取的措施有：布线时，电源线和地线尽量粗；尽量降低单片机的晶振和选用低速数字电路；IC 器件尽量直接焊在电路板上。

经过长时间的运行和改进，系统稳定性已经基本达到了我们的要求。

2.4 本章小结

硬件设计在智能车整个系统的设计中非常重要。因为车体的硬件性能直接影响到智能车运动的极限参数，如最高速度、最大响应时间、最大侧滑速度等。为了提高硬件的性能主要从三个方面进行设计，即车模调整、测速模块及传感器的安装、电机的驱动电路设计。本章主要内容就是介绍所述智能车的整体硬件设计方案，最主要的工作是对 XS128 的最小系统板的开发。

3 软件设计

基于摄像头的智能车的软件设计主要有以下几个方面：视频信号采集、视频信号处理、电机速度控制。其中视频信号的采集是最先要考虑的，如果视频信号采集不稳定的话，则直接影响到后面的控制结果，造成智能车的不稳定，严重影响智能车运行速度。

对于电机的控制则是最终体现车体速度的关键，上一章所述的硬件设计师软件设计的基础。只有硬件设计过硬才能得到一个比较好的控制实体，之后才能更容易的控制。

3.1 视频信号采集

视频信号的采集是下面各控制算法的基础，良好的视频采集程序能够提高软件运行的稳定性。

3.1.1 摄像头的工作原理

如果要对摄像头进行数据采集必须要弄清楚其具体的工作原理，CMOS 的工作原理就是：按一定的分辨率，以隔行扫描的方式采集图像上的点，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一一对应的电压值，然后将此电压值通过视频信号端输出。

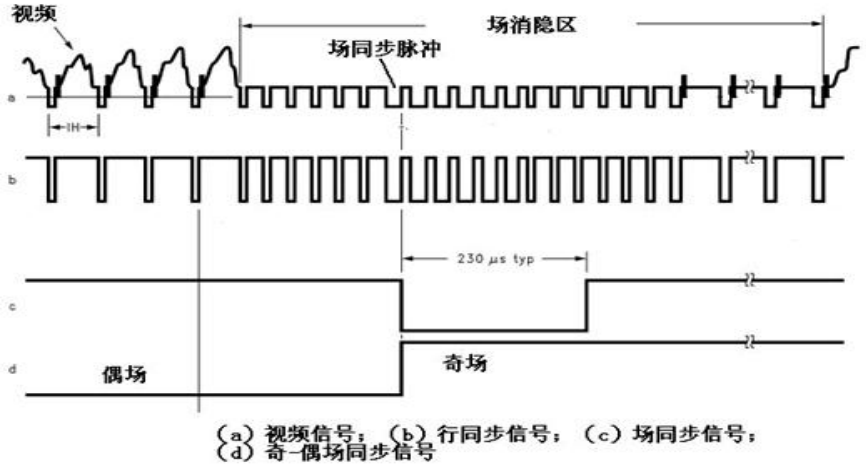


图 3.1 行场同步信号

摄像头连续地扫描图像上的一行，则输出就是一段连续的电压信号，该电压信号的高低起伏反映了该行图像的灰度变化。当扫描完一行，视频信号端就输出一个低于最低视频信号电压的电平(0.3V)，并保持一段时间。这样相当于，紧接着每行图像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），开始扫描新的一行，如此下去，直到扫描完该场的视频信号，接着会出现一段场消隐区。该区中有若干个复合消隐脉冲，其中有个远宽于（即持续时间远长于）其它的消隐脉冲，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。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。奇场时只扫描图像中的奇数行，偶场时则只扫描偶数行。

对于接收到的视频信号，需经 LM1881 进行行场同步信号分离后才能进行采集。分离得到的同步信号如图 3.1 所示。

3.1.2 采样时序

视频同步信号主要有场同步信号、奇场偶场同步信号和行同步信号。PAL 制式下场同步信号的频率为 50Hz，其中场又分为奇场和偶场，奇场和偶场扫描频率均为 25Hz。奇场扫描的是图像中的奇数行，故名思意，偶场扫描的为图像中的偶数行。为了简化视频信号的采集，只对场同步信号进行处理，忽略奇场和偶场信号的差别，实验证明该方案可行。

具体说来，采集一场视频信号的流程如下：

第一步，LM1881 分离出行、场同步信号在 DG128 的 H 口产生中断。接收到场同步信号时，根据 PAL 规范，此时意味着还有 22 行（约 1408μs，为视频信号的场消隐区）将开始第一个数据行。为确保正确的采集到数据，通过判断行中断的次数过滤掉此消隐区。

第二步，如果行中断次数达到 22 行，则初始化 ATD 准备进行数据采集。对于 5 次接收到得 ATD 数据，不存入图像数组，过滤掉行消隐区。

第三步，行消隐区过后，循环接收 ATD 转换的数据，当一行 41 个点采集完时，数据行的数据采集完成。由于一场中有 300 多个行，但 ATD 只采集其中的 28 行，故每 9 行采集一次，完成对一场数据的采集。如果一场采集完成，即图像数组已经存满，则数据采集标志置位，否则重新执行第三步。

第四步，当数据采集标志位已置位后，则图像采集已完成。为防止下一场数据又写入到数组中，将刚接收到得数据拷贝到另一个同样大小的数组中进行保存。之后，图像处理则只调用此数组。

可见，本程序的运行完全是由行、场同步信号产生的中断来驱动的。该程序的优点在于中断个数少，而且把复杂的时序用同步信号过滤掉，简化了采样程序，提高了采样稳定性。程序流程图如图 3.2 所示。

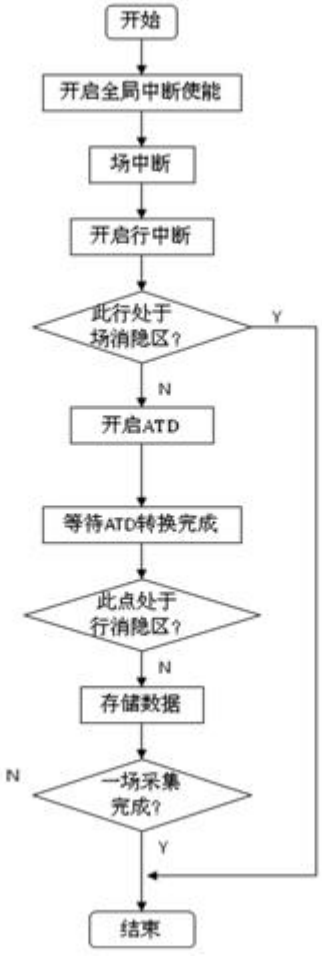


图 3.2 图像采集流程图

程序 C 代码如下：

```
#include <hidef.h>
#include <MC9S12XS128.h>
#pragma LINK_INFO DERIVATIVE "mc9s12xs128"
int ii,jj;
unsigned  int  t,t1;
unsigned  char sdata;
//-----------------------------------------------
  #define lie_end  260
  #define hang_end  91
  #define lie  52
  unsigned  char ab[lie+1];
  #define delay(num)   \
  { \
     unsigned int i,j; \
     for(i=0;i<num;i++) \
     for(j=0;j<58;j++); \
  }
  unsigned char c_lie=0,g_lie=0;
  unsigned int lie_count=0,hang_count=0,am;
  unsigned char ta[lie][hang_end];
  unsigned int get_n[]={  16,29,41,52,62,71,79,86,92,98,      //图像定距采集对
应的摄像头行数
                        103,108,112,116,120,124,128,132,135,138,
                        141,144,147,150,153,156,159,162,165,168,
                        170,172,174,176,178,180,182,184,186,188,
                        190,192,194,196,198,200,202,204,206,208,
                        210,211,212,242,243,245,246,247,248,249,250};
 unsigned char ts1,ts2, ge,shi;
 void SciTx(unsigned char text);
 void chuankou(){
      delay(1700);
      ge=37%10;
      shi=37/10;
      //sdata=shi*16+ge;
      //SciTx(sdata);
      for(ii=1;ii<40;ii++)
         {
            //第一次的调试的时候把 40 修改为 2，校对调试助手的屏幕大小，当出现
            的数据刚刚一行时，说明串口调试助手的屏幕大小适合。
           //当完成上面的操作后，把 2 修改为 40，这时候就能够看到一幅图像的数据
           了。
       for(jj=1;jj<90;jj++){
            ts1=0;ts2=0;
```

```
            if(ta[ii][jj]==1&&ta[ii][jj+1]==1) ts1=16;//&&ta[ii][jj]>20
            jj++;
            if(ta[ii][jj]==1&&ta[ii][jj+1]==1) ts2=1;//&&ta[ii][jj]>20
            sdata=ts1+ts2;
            SciTx(sdata);
        }
     }
     for(;;) {}
}
void SciInit()
{
    DDRM=0x01;
    SCI0BDH=0x00|0x01;
    SCI0BDL =0xA0;
    SCI0CR2=0X2C;
    SCI0CR1=0;
}
/*-------发射端程序----------------------*/
 void SciTx(unsigned char text)
 {
    while (!(SCI0SR1&0x80));
    SCI0DRH=0;
    SCI0DRL=text;
 }
 void chaopin(void)
 {
    CLKSEL=0X00;
    PLLCTL_PLLON=1;
    SYNR =0xc0 | 0x07;
    REFDV=0xc0 | 0x01;
    POSTDIV=0x00;
    _asm(nop);
    _asm(nop);
    _asm(nop);
    _asm(nop);
    while(!(CRGFLG_LOCK==1));
    CLKSEL_PLLSEL =1;
}
 void TIM_init(void)
 {
    PACTL=0X50;
    PACNT=0X0000;
    TIOS =0x00;
    TSCR1=0x80;
    TCTL4=0x18;
```

```
    TIE=0x06;
    TFLG1=0xFF;
}
void shijian(void) {
   PITCFLMT_PITE=0;
   PITCE_PCE0=1;
   PITLD0=9999;
   PITMTLD0=5;
   PITMUX=0X00;
   PITINTE_PINTE0=1;
   PITCFLMT_PITE=1;
}
//=========================================================================//
 void main(void)
{
    DDRA=0Xff;
    DDRJ=0X02;
    DDRK=0X30;
    DisableInterrupts;
   {
      DDRB=0X01;
      DDRM=0X00;
      DDRJ=0XBF;
      PTJ_PTJ6=1;
      DDRS=0XEE;
      PPSS=0X11;
      PPSJ=0XFF;
   }
      chaopin();
      TIM_init();
      shijian();
      SciInit();
      EnableInterrupts;
   {
      unsigned  int e,w;
       for(e=1;e<6;e++)
         for(w=0;w<10;w++);
    }
DDRS=0XFF;
PPSS=0X11;
PPSJ=0XFF;
DDRB=0XFF;
DDRJ=0XfF;
DDRA=0Xf0;
for(;;)
```

```
  {
     chuankou();
        /*开启串口调试方法首先链接好线：通过USB转串口，把芯片的S0与TXD接通，
          S1与RXD接通，共地线。接着把接通芯片的电源，和摄像头的电源，同时保证
          摄像头和芯片的线连接正确，并保证摄像头看到完整的跑道。把本程序拷入芯
          片里。打开串口调试助手，校对对应的串口端口，并选择十六进制显示。 */
  }
}

//-------------------------------------------------------------//
  #pragma CODE_SEG NON_BANKED
 void interrupt 10 IC2ISR(void)
  {
     TFLG1_C2F=1;
     if(lie_count==get_n[c_lie])
       {
         delay(1);
         for(hang_count=0;hang_count<=hang_end;hang_count++)
            {
                ta[c_lie][hang_count]=PORTA_PA0;//ATD0DR0L;
            }
         c_lie++;
       }
       lie_count++;
   }
  void interrupt 9 IC1ISR(void)
  {
      TFLG1_C1F=1;
      hang_count=0;
      lie_count=0;
      c_lie=0;
   }
void interrupt 66 PIT0Interrupt(void)
 {
   //chesudu=PACNT;
   PITTF_PTF0=1;
   PACNT=0X0000;
 }
 #pragma CODE_SEG DEFAULT        //接脚是行同步接pt2  场同步接pt1
```

3.1.3 中断分析

视频采集共涉及到2个中断：场中断、行中断。对于中断的编写我们采用了中断嵌套的方式，只需1个场中断服务程序，进入场中断程序后才开启行中断。对于中断而言，其首要任务是及时把结果寄存器中的值转移出来，因ATD在不停的在采集视频信号，如果转移不及时，新的结果会把原来的结果覆盖掉。中断程序需要根据一系列行场中断及ATD转换完成标志位来计算本次采集结果对应的图像位置，然后保存到数组中。ATD在采集间隔

为8μs，中断服务程序必须要在这段时间内完成，因此要求程序不能过于冗杂。

3.2 路径识别与自动阀值

所谓路径识别，简单的理解就是把图像中反映路径的部分提取出来。这是一个图像分割的过程。图像分割是计算机进行图像处理与分析中的一个重要环节， 是一种基本的计算机视觉技术。在图像分割中，把要提取的部分称为“物体（Object)”，把其余的部分称为“背景（Background)”。分割图像的基本依据和条件有以下4个方面：

a.分割的图像区域应具有同质性，如灰度级别相近、纹理相似等；

b.区域内部平整，不存在很小的小空洞；

c.相近区域之间对选定的某种同质判据而言，应存在显著的差异性；

d.每个分割区域边界应具有齐整性和空间位置的平整性。

现在的大多数图像分割方法只是部分满足上述判据。如果加强分割区域的同性质约束，分割区域很容易产生大量小空洞和不规整边缘；若强调不同区域间性质差异的显著性，则极易造成非同质区域的合并和有意义的边界丢失。不同的图像分割方法总是为了满足某种需要在各种约束条件之间找到适当的平衡点。

图像分割的基本方法可以分为两大类：基于边缘检测的图像分割和基于区域的图像分割。

边缘是指图像局部亮度变化最显著的地方，因此边缘检测的主要依据是图像的一阶导数和二阶导数。但是导数的计算对噪声敏感，所以在进行边缘检测前需要对图像滤波。大多数的滤波算法在滤除噪声的同时，也降低了边缘的强度。此外，几乎所有的滤波算法都避免不了卷积运算，对于智能车系统来说，这种运算的计算量是S12单片机系统所无法承受的。

阈值分割法是一种基于区域的分割技术，它对物体与背景有较强对比的景物的分割特别有用。它计算简单，而且总能用封闭且连通的边界定义不交叠的区域。阈值分割法的关键在于阈值的确定。如果阈值是不随时间和空间而变的，称为静态阈值；如果阈值随时间或空间而变化，称为而动态阈值。基于静态阈值的分割方法算法简单，计算量小，但是适应性差。基于动态阈值的分割方法其复杂程度取决于动态阈值的计算方法。

针对本智能车系统，普通的双峰法就能适合绝大部分情况，因为智能车的运行环境是跑道，背景和前景区分明显，且背景简约。但是实验环境并不理想，由于受到光线斜射的影响，有时背景和前景的对比十分不明显。结合实际需要，提出了一种新的计算阈值的方法，这种方法的思想与迭代法有些相似。

首先我们假定，智能车系统运行时，开始时采集的第一幅图像是良好的。这个“良好”含义是：第一行也就是最近处的一行，完整的包含了导航线，并且使用“双峰法”能正确提取。其次，我们对每一行都定义一个阈值Tr，每一行都用该行的阈值Tr进行分割。在此基础上按照以下规则进行阈值传递：

a.如果第r行分割出来的黑色区域（线段）是连续的，那么计算目标区域和背景区域的平均灰度值，并取其中值，做为当前行和下一行的Tr。

b.如果第r行分割出来的黑色区域是空集或者不连续，则保持当前行和下一行的阈值不变

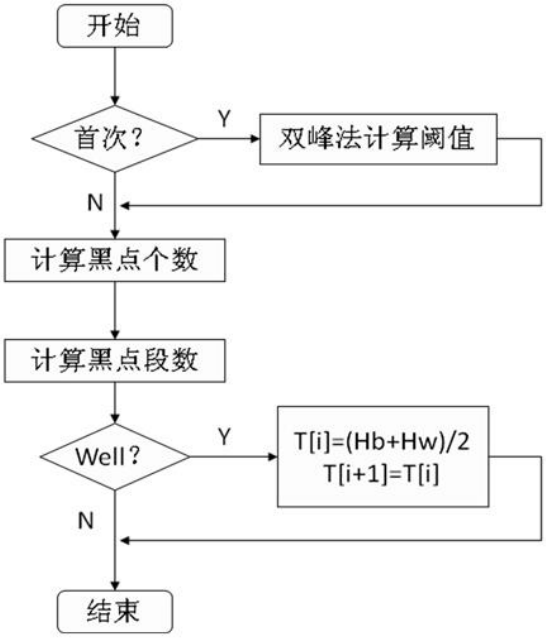


图 3.4 一行阈值计算流程图

算法整个流程，如图 3.4 所示。这种算法的特点是运算量小，但是能跟踪环境光线的逐渐变化。由于环境光线不管在时间上还是空间上都是逐渐变化的，所以这种分割算法产生严重错误的概率极小，小于 1/10000。这个数据的来源是，我们让智能车在实验跑道上运行，累计时间超过 30 分钟，没有一次智能车因为阈值分割错误而停车。

图 3.5 是智能车因错误而停车时的即时照片，在上面几行中，跑道黑线与背景间的差别已十分微弱，但图 3.6 显示，这种情况下阈值分割的结果仍然是正确的，包含了识别跑道的足够信息，减少了智能车其它的程序错误

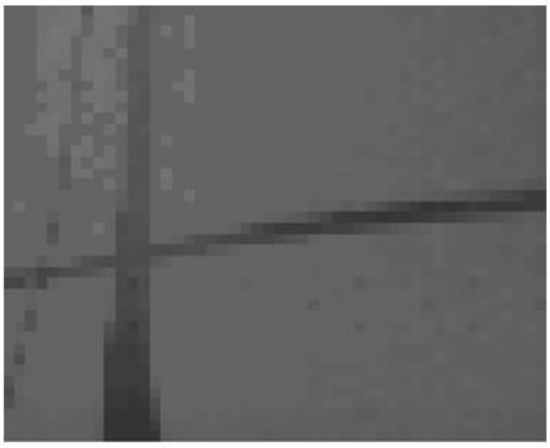


图 3.5 干扰严重时的跑道图片

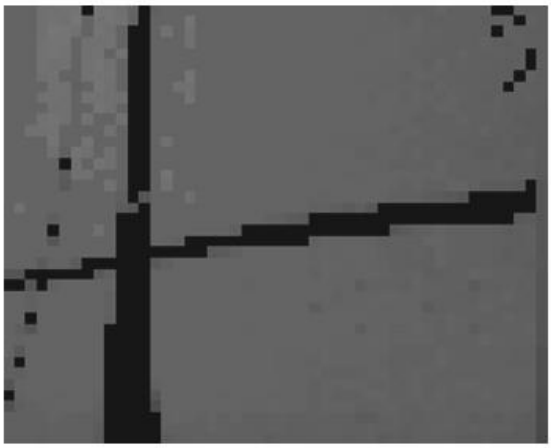


图 3.6 动态阈值分割的结果

3.3 软件抗干扰处理

抗干扰处理包括以下方面的内容：

a. 消除信号产生和传输的过程中造成的噪声；

b. 消除跑道不理想或光线不均匀形成的干扰；

c. 消除跑道中的交叉线、断续线；

d. 识别起跑线

信号产生和传输的过程中造成的噪声具有普遍性，但是只有当其非常严重时才对路径的识别造成影响。所以采取以下滤波算法：当某点的灰度值与其左边和右边点的灰度值的

差同时大于某个正阈值或同时小于某个负阈值时，认为该点是干扰点，取其左右两边灰度值的平均值作为该点的灰度值。

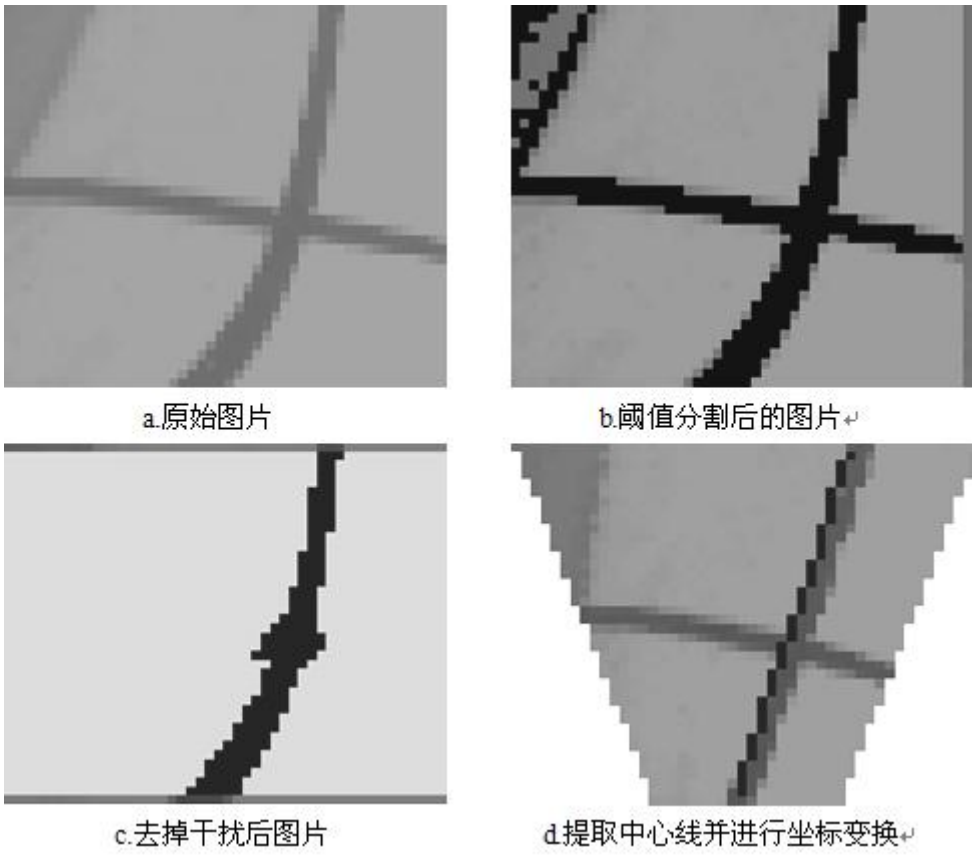


图 3.7 跑道抗干扰处理

对于跑 道不理想或光线不均匀形成的干扰，其一般特点是散布于跑道黑线的两边，所以采用以下算法处理：

步骤 1 从最近的第一行开始，寻找这样一个行：该行只有一段连续黑色线段，且线段的左右边均不是图像的左右边。标记该行为 Start 行。

步骤 2 在下一行中寻找这样一段连续黑色线段，满足这样的条件：其左边缘不在上一行黑线右边缘的右边，且其右边缘不在上一行黑线左边缘的左边。

步骤 3 如果存在步骤 2 中的黑色线段，认为该段黑线是跑道黑线，其余黑点（线段）均认为是干扰，跳步骤 2。否则，标记该行为 End 行，并计算该行与 Start 行之差，如果大于 3，则结束搜索，如果小于等于 3，则返回步骤 1。

这种算法的实质是搜索连续的黑线。因为跑道是连续的，而干扰却大部分是离散分布的。对于跑道中的交叉线，可以统计一行中黑点的个数，因为正常的黑线只有 2.5cm，在图像中不超过 6 个点。所以把 6 作为一个阈值 TN，如果一行黑点的个数超过 TN 个点，就可以认为是交叉干扰。实际上，我们做了更复杂的处理，就是根据当前跑道的倾斜程度来调节 TN，因为当跑道倾斜时其宽度将发生变化。并且，我们不只是识别出交叉干扰，而且“砍掉”它的左右两支，保留一条连续的跑道黑线。

起跑线是在跑道黑线的左右两边，对称存在两条长度各为 10cm，相距 6cm，且与跑道黑线垂直的黑色线段。正确的识别起跑线是应用更高级控制算法如记忆算法的基础。我们使用如下算法识别起跑线：在已识别出跑道黑线的行，保证该行不是交叉行（前面已经识别出），然后确定其左右两边各一个矩形区域，统计两个区域中黑点的个数，如果均大于某一阈值，则判断该行出现了起跑线。

经过以上处理，我们得到了如下信息：

首先是两个标记 Start 和 End，这两个标记标记了成功识别出的跑道的最近一行和最远一行；其次是在 Start 和 End 之间的每一行，记录了其跑道的左边缘位置和右边缘位置，并且通过坐标变换算法，计算出每行跑道中心线的实际位置。最后留下了一些其它标志，比如每行是否是交叉、起跑线或者断续线。这些标志在我们调试的过程中都用某种方式以图像的形式反映出来。处理效果如图 3.7 所示：

3.4 摄像头测量距离标定

如果要计算车体与路径偏移量的大小，必须要对摄像头的测量范围进行标定，也就是将摄像头测量的数据与实际图像的位置相对应起来，方可得到精确的测量模型。通过仔细研究各届前几名的控制算法，程序中的参数全是由实际测量得到的，只有这样写出的控制算法效果才比较好。摄像头的位置、俯仰角度决定了摄像头看到的图像范围，另外由于 CMOS 的特殊成像方式，俯仰角度的大小直接关系到图像的畸变大小，故俯仰角要合理。如果考虑摄像头的前瞻性，可适当增加摄像头安装高度，不过这样会让车体重心变高，造成过弯极限速度受限，所以摄像头的安装要综合考虑两方面的影响。

经过测验，最终确定摄像头高度为 300mm，俯仰角 60°。此时测得的最远距离 600mm，最近处距离 200mm，最远处距离宽度 400mm，最近距离宽度 200mm。如图 3.8 所示。

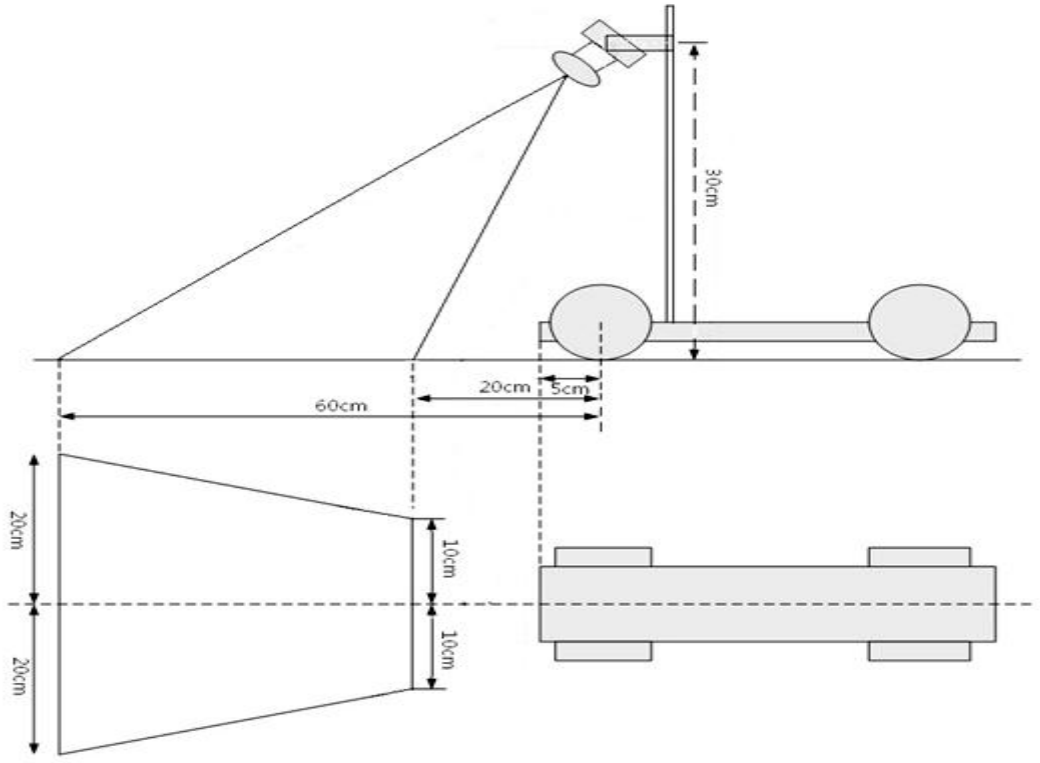


图 3.8 摄像头安装参数

3.5 电机的速度控制

电机是智能车运行的动力，为智能车的发动机。对电机的控制好坏，直接影响到智能车的运行速度。电机的控制最重要的是响应速度。

3.5.1 电机的开环响应特性

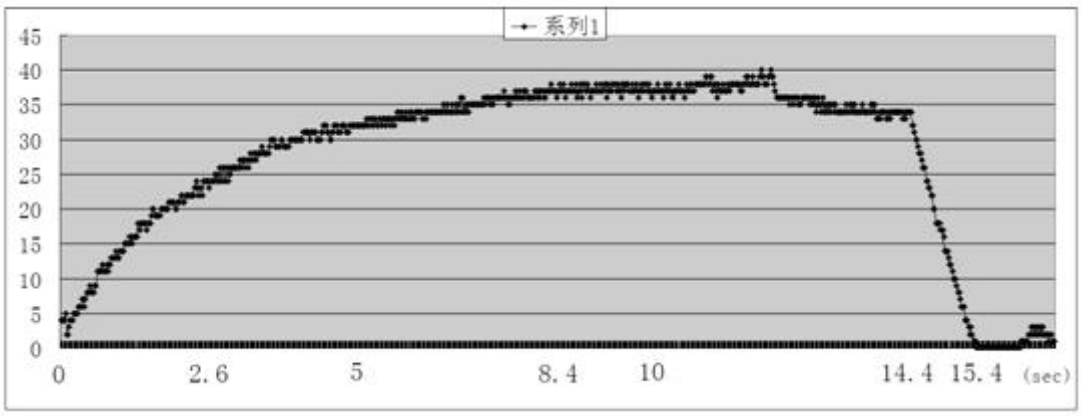


图 3.16 电机的开环响应曲线

电机的开环响应曲线如图 3.16 所示，可以看出，小车速度在增加时，速度曲线近似一递增的对数曲线。小车速度达到稳态速度的 60%需要 2.6 秒时间，而要达到稳态速度需要 8.4 秒的时间。而在速度减小时，速度成直线下降，从稳态速度下降至零只需要 1 秒时间，远小于速度上升时间。因为驱动电机在减速时（占空比设置为较低的值）采取的是电磁制动方式，所以，小车的减速快于一般的自然减速。因而，相对于速度递增时的电机响应时间，小车递减时间较短。

为了减小车模制动时间，在电机的电源端连接了一个续流二极管。另外，由于电机加速减速剧烈，故为了减小电机产生的纹波，又加了一个滤波电容。经示波器观察，加上滤波电容后，电机产生的纹波有明显的改善。

3.5.2 曲率计算

曲率问题可以归结为已知三角形的三点坐标 A(x1, y1) 、 B(x2 , y2) 、C (x3 , y3) ，求解三角形外接圆曲率。学过高数的都应该知道向量的叉乘，定义为：两个向量进行叉乘得到的是一个向量，方向垂直于这两个向量构成的平面（三个向量符合右手坐标系），大小等于这两个向量组成的平行四边形的面积。

设两个向量为$\vec{a}=(x_a,y_a,z_a)$、$\vec{b}=(x_b,y_b,z_b)$，则

$$\vec{a}\times\vec{b}=\begin{vmatrix}\vec{i} & \vec{j} & \vec{k}\\ x_a & y_a & z_a\\ x_b & y_b & z_b\end{vmatrix} \qquad (1)$$

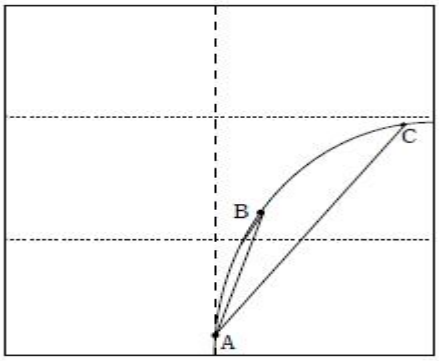
由叉乘的定义可知，求ΔABC的面积，可以通过求解$\frac{\left|\overrightarrow{AB}\times\overrightarrow{AC}\right|}{2}$来获得，将上述叉乘公式应用于二维情况，即取 z = 0，可得

$$\overrightarrow{AB}\times\overrightarrow{AC}=\begin{vmatrix}\vec{i} & \vec{j} & \vec{k}\\ x_2-x_1 & y_2-y_1 & 0\\ x_3-x_1 & y_3-y_1 & 0\end{vmatrix}=((x_2-x_1)(y_3-y_1)-(x_3-x_1)(y_2-y_1))\vec{k} \qquad (2)$$

所以

$$S_{\Delta ABC}=\frac{\left|\overrightarrow{AB}\times\overrightarrow{AC}\right|}{2}=\frac{((x_2-x_1)(y_3-y_1)-(x_3-x_1)(y_2-y_1))}{2} \qquad (3)$$

上式中，如果 ABC 三点是顺时针方向分布，则三角形面积为负值，逆时针分布为正值（如果不需要符号，取下绝对值即可）。



面积的符号对于智能车其实还是挺有用的。如上图，如果曲线向右拐，算出的面积是负的，如果曲线向左拐，算出的面积是正的，上述面积的正负反映了曲线的方向。相比于海伦公式，用上述公式计算面积既可以减轻计算量，又可以反映曲线的方向，一举两得。有了上述公式，曲率可以表现为三角形外接圆半径的倒数，从而可得曲率计算公式：

$$K=\frac{4S_{\Delta ABC}}{AB\times BC\times AC} \qquad (4)$$

上式中要求出三边长，会用到求根公式，在单片机中开根号，那是很要命的。如果精度要求不高，可以自己写一个简单的求根函数，东北大学给出的函数是

```
unsigned int m_sqrt(unsigned int x)
{
     uchar ans=0,p=0x80;
     while(p!=0)
     {
           ans+=p;
           if(ans*ans>x)
          {
           ans-=p;
           }
           p=(uchar)(p/2);
      }
      return(ans);
 }
```

3.5.3 速度 PID 算法

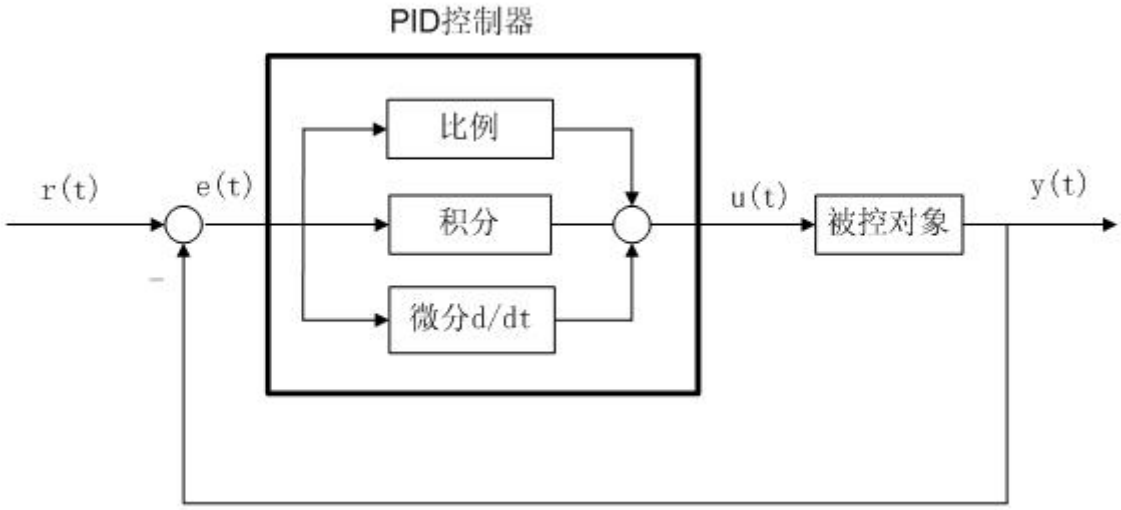


图 3.17 PID 控制系统原理框图

根据偏差的比例（Proportional）、积分（Integral）、微分（Derivative）的线性组合进行反馈控制（简称 PID 控制），数字 PID 控制算法是电机微机控制中常用的一种基本控制算法[32]。

在连续系统中，模拟 PID 调节器是一种线性调节器，控制系统原理框图如图 3.18。控制规律如式 3.4 所示：

$$u(t) = K_p[e(t) + \frac{1}{T_I}\int_0^t e(t)dt + T_d\frac{de(t)}{dt}] \qquad (3.4)$$

式中：Kp：比例增益，Kp 的倒数称为比例带；TI：积分时间常数；TD：微分时间常数；u(t)：控制量；e(t)：偏差，等于给定量与反馈量的差。

在计算机控制系统中，数字 PID 控制算法通常又分为位置式 PID 和增量式 PID。本次设计中，我们采用增量式 PID。

增量型算法与位置型算法相比，具有以下优点：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；增量型算法得出的是控制量的增量，而位置型算法的输出是控制量的全量输出，误动作影响大；采用增量型算法，易于实现手动到自动的无冲击切换。

在实际应用中，采样的反馈值 y(k)即为脉冲累加器中的 PACNO 中的脉冲数，预设门限值 A 在参数整定时根据实际情况调节，输出 u(k)并不能直接用来控制电机，需要将其转换为控制 PWM 占空比，然后用增大或减小 PWM 占空比的方法来实现对电机的加减速的控制。换句话说，在求偏差量时，实际上用的是每 20ms 电机转过的齿轮数和实际期望电机转过的齿轮数，通过二者的差值，再乘以相应的系数，即 KP、KI、KD 的协调控制，计算出相应的 PWM 占空比，实际上用的是 PWMDTY 的值[33]。

本设计中综合考虑各种因素，最后选用的采样周期为 20ms，即每 20ms 对电机进行一次 PID 调节。由于在程序中，对图像的采集使用的是 PH 口的中断程序，因此，PID 采样周期的选择实际上是受限制与图像采集，因为每行的扫描周期为 64μs，有效扫描时间为 52μs，采用的是隔行扫描的方式，即每隔 9 行采集一行图像的信息，如果在每行之间加入 PID 调节的话，那么处理 PID 子程序的时间必须控制在 64*5＝320μs 之内，因此，经过分析，最终决定将 PID 的采样周期定为 20ms，即当进行一次场采集进行一次 PID 调节。而且经过最终的检验，这样能够满足对速度控制的需要。

程序代码：

```
include <hidef.h>
#include <MC9S12XS128.h>
#pragma LINK_INFO DERIVATIVE "mc9s12xs128"
//变量 chesudu 是采集到的速度数据
//接口定义: 编码器脉冲中断信号接 PT7 （需要上拉 5-10K 电阻）
void shijian(void);
void chaopin(void);  //超频程序
void TIM_init(void);//定时程序
int chesudu;
void main(void)//主函数
{
   chaopin(); //超频函数初始化
    shijian() ;
   TIM_init();//计数器函数初始化
   DisableInterrupts;
   for(;;)
   {
      if(speedmax<=pulse_speed)
         speedmax=pulse_speed;
      if(speedmax>=40)
         speedmax=40;
         ideal_speed=speed_table2[absolute(car_positn)];
         speed_error=ideal_speed-pulse_speed;
         dri_flag=1;
         pid();
      if(pulse_speed>=9)
       {
       if(car_positn==10)
          {
            angle_data=1265;
            dri_flag=0;
            car_driver=600;
           }
       if(car_positn==-10)
          {
            angle_data=1573;
            dri_flag=0;
            car_driver=600;}
          }
       }
        if(car_driver>=1000)
           car_driver=1000;
        else if(car_driver<=0)
           car_driver=0;
```

```
    }
}
 void chaopin(void)//64M
{
    CLKSEL=0X00;
    PLLCTL_PLLON=1;
    SYNR =0xc0 | 0x07;
    REFDV=0xc0 | 0x01;
    POSTDIV=0x00;
    _asm(nop);
    _asm(nop);
    _asm(nop);
    _asm(nop);
    while(!(CRGFLG_LOCK==1));
    CLKSEL_PLLSEL =1;
}
void TIM_init(void)
{
  PACTL=0X50;    //端口初始化
  PACNT=0X0000;
}
void shijian(void) //中断时间设计
{
   PITCFLMT_PITE=0;
   PITCE_PCE0=1;
   PITLD0=9999;
   PITMTLD0=5;
   PITMUX=0X00;
   PITINTE_PINTE0=1;
   PITCFLMT_PITE=1;
}
//=======速度采集中断函数，已经完全可以采集速度的=========
 #pragma CODE_SEG NON_BANKED
void interrupt 66 PIT0Interrupt(void)//测速的中断程序
{
   chesudu=PACNT;//采集小车的当前速度
   PITTF_PTF0=1;
   PACNT=0X0000;
}
#pragma CODE_SEG DEFAULT
```

3.6 本章小结

本章针对智能车的软件系统设计展开叙述，首先介绍了视频信号的采集，用到了动态阈值分割法，增加了智能车对场地的适应性，减少了比较严重的控制误差发生几率。之后对摄像头的安装参数进行标定，通过对得到的图像进行校正，运用测路径曲率的方法，实

现了智能车的完美过弯。

对于电机的驱动程序则采用了速度 PID 控制，通过对跑道特征的判断，确定速度期望值，由捕捉到的编码盘产生的脉冲数测得电机实际速度，计算二者之差得到偏差量，再通过 KP、KI、KD 协调控制，计算得到控制电机的 PWM 值，实现电机的 PID 控制。总程序详见附录 A。

4 智能车的开发与调试

4.1 编译环境

4.2 下载调试

4.2.1 安装 BDM 驱动

4.2.2 调试程序

总结与展望

本课题对智能车的硬件和软件进行了设计，最终达到让智能车自动循线的目的。本文研究工作总结如下：

1.设计开发了 MC9S12XS128 单片机的最小系统板。由于其特殊的外形，更适合在智能车上安装，大大降低了智能车的重心。

2.使用动态阈值法从采集的图像数据中分割出跑道信息，提高了算法对场地的适应性以及抗干扰性。

3.提出了一种基于跑道斜率算法，实现了入弯走内道、S 弯直线通过的目标，提高了智能车的运行速度。

4.用光电编码盘检测智能车运行速度，形成速度闭环，应用增量式 PID 对电机进行控制，提高了电机的响应速度。

对于我们而言，准确的寻线并不是最终目的，在不冲出智能道的情况下，尽可能缩短智能车单圈行驶时间才是我们所追求的。从这个意义上说，路径识别和寻线算法只是我们为了保证完成单圈行驶而采取的辅助手段。

但是由于水平有限，系统中尚存在许多问题有待改进：

1.抗干扰处理。智能车在跑道上会遇到各种情况，如交叉线、断续线，以及还有实验跑道上的干扰点，以及由环境光线不均匀造成的干扰。如何确保在任何情况下都能识别出跑道，或者判断出智能车在跑道上的位置，需要做进一步的努力。

2.参数计算的准确性。虽然实现了斜率和曲率的计算，但是其精度仍然不够理想，特别是在能够识别的路径比较段的情况下，计算出的数据与实际有很大的偏差。笔者运用基于三个坐标点测量跑道曲率的方法，发现其与本摄像头不能完美匹配。因为 CMOS 摄像头测得范围比较窄，在摄像头看到区域黑线一般呈直线，测得的曲率几乎一直为零值附近，故广角 CCD 摄像头可以考虑作为路径检测传感器。为了提高车速，对广角 CCD 摄像头的研究值得继续探索。

3.控制算法的改进。本系统采用了单闭环加前馈的控制方法，这在控制算法中是比较复杂的，实际应用也验证了效果的良好，但其参数调节太过复杂，需要太长的时间调节。

4.电机驱动的改进，智能车高速运行时，快速刹车达不到要求，限制了车速，故对其他电机驱动的开发亟待解决。

参考文献

[1] 飞思卡尔半导体公司.全国大学生智能车竞赛与飞思卡尔 S12 单片机.单片机与入式系统应用[J].2007，(8)：78-79

[2] 曾星星，基于摄像头的路径识别智能车控制系统设计[J]，湖北汽车工业学院学报，2008，22(2)：73-78

[3] 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)：19-22

[4] 金宝智.图像传感器 CCD 与 CMOS 的对比[J].现代电子技术.2005，5

[5] 冯津.基于 MC9S12 单片机的摄像头参数测量及模拟视频信号离散化[A].智能检测控制技术及仪表装置发展研讨会论文集，2007：154-160

[6] 马忠梅、籍顺心、张凯等编著，单片机的 C 语言应用程序设计，北京：北京航空航天大学出版社，1998.

[7] National Semiconductor LM1881 Video Sync Separator 2003.6

[8] 谭浩强，C 程序设计，北京：清华大学出版社，1999

[9] Steven F.Barrett，Daniel J.Pack 著，嵌入式系统——使用 68HC12 和 HCS12 的设计与应用，北京，电子工业出版社，2006

[10]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

[11]史毅俊.卫星通信基带解调中的关键技术研究和实现[J].上海交通大学,2007

[12]林辛凡.李红志.黄颖.第二届“飞思卡尔”全国大学生智能汽车邀请赛三角洲队技术报告[R].清华大学，2006

[13]Wang Xiuquan.Route Identification and Direction Control of Smart Car Based on CMOS Image Sensor.ISECS International Colloquium on Computing, Communica

[14]手创科技 BDM 使用手册 2011

[15]百度文库 MC9S12 超详细中文资料 2012

[16]卓晴 ， 黄开胜 ， 邵贝贝 .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.2006

[17]孙肖子，张企民 . 模拟电子技术基础.西安:西安电子科技大学出版社,2000

[18]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].机械工业出版社，2008

[19]王俭,轮式小车的轨迹跟踪控制[J].控制与决策，2005，15(5):626-628

附件

A：智能车程序代码

```
#include <hidef.h>
#include <MC9S12XS128.h>
#pragma LINK_INFO DERIVATIVE "mc9s12xs128"
//接口定义: 编码器脉冲中断信号接 PT7  模拟信号接 AD0-AD12
// PWMDTY45 和 PWMDTY67 接电机
//-------------转角变量----------------
unsigned char sam_atd_g[13]; //传感器坐标标志位
         int temp_positn[5];
unsigned int adc_limit; //ATD 电压比较阀
unsigned int car_positn;  //车的当前位置
unsigned int black_sensor_number; //检测到黑线的传感器个数
unsigned int times,time;    //丢失黑线的次数
 int wigwt0,wigwt1,wigwt2,pk1=0;
//-------------速度变量------------------
unsigned int car_driver;  //驱动力参数
unsigned int pulse_speed;//电机当前速度
unsigned int ideal_speed; //理想状态下的速度
         int speed_error; //理想速度与当前速度的差值
         int pre_error=0; //PID 控制的速度差值
         int pre_d_error=0;  //PID 控制的速度上一次差值
         int pk=0,error=0;  //速度的 PID 值
         int pre_car_positn;
         int speedmax;
//---------------标志变量-------------
unsigned char finish_flag;  //完成起跑线检测标志位
unsigned int dri_flag; //驱动标志
unsigned int ten_line; //统计时间
unsigned int ten_flag;
unsigned int ten_2_line;
unsigned int checkstart;
unsigned int start;
//------------表格值-------------------
unsigned int speed_table2[11];
#define kp 213     //宏定义
#define ki 7        //宏定义
#define kd 15        //宏定义
#define kp1 14      //宏定义
#define kd1 5              //宏定义
#define Angle_Center  1419      //宏定义
//----------函数定义----------------
```

```
void data_init(void);        //关键数据初始化
void SetBusCLK_40M(void);    //锁相环初始化
void atd_init(void);    //模/数转化初始化
void pwm_init(void);        //脉冲 PWM 初始化
void TIM_init(void);         //测速模块 TIM 初始化
void sam_position(void);    //模/数转化模块函数
void check_start(void);    //检测起跑线函数
void car_position(void);  //计算当前模型车的位置函数
void speed(void);     //计算车的速度控制函数
void driver(void);      //车的电机驱动函数
void delay(unsigned int X);    //延时函数
void stop(void);    //停车控制函数
void pid(void);     //速度的 PID 控制函数
unsigned int absolute(int);
void Timer(void);
void getATDvlaue(void);
//=========********主程序*****============
void main(void)
{    unsigned int n;  //先初始化各个初始化函数
     data_init();        //设置基本参数
     SetBusCLK_40M();  //锁相环初始化
     TIM_init();   //TIM 初始化
     atd_init();    //ATD 初始化
     pwm_init();
     Timer();    //PWM 初始化
     getATDvlaue();
     for(n=0;n<=65;n++) delay(40000);
      EnableInterrupts;
for(;;)          //控制车行驶的函数
  {
     car_position(); //计算车的位置
     speed();      //计算车的速度
     driver(); //驱动控制
  }
}
//===========子程序===============
//*************初始化函数***************
//----------数据初始化-------------
void data_init(void)
 {
      finish_flag=0;
      pulse_speed=0;
      pre_car_positn=0;
      ten_flag=0;ten_2_line=0;
      checkstart=0;
```

```
      speedmax=0;
      time=0;
      wigwt0=0;wigwt1=0;wigwt2=0;pk1=Angle_Center;
      pk=0;
      speed_table2[0]=13;
      speed_table2[1]=13;
      speed_table2[2]=12;
      speed_table2[3]=12;
      speed_table2[4]=12;
      speed_table2[5]=12;
      speed_table2[6]=12;
      speed_table2[7]=12;
      speed_table2[8]=12;
      speed_table2[9]=11;
      speed_table2[10]=10;
 }
 //------------锁相环初始化-----------
 void SetBusCLK_40M(void)
{
    CLKSEL=0X00;//disengage PLL to system
    PLLCTL_PLLON=1;//turn on PLL
    SYNR=4;
    REFDV=1;//pllclock=2*osc*(1+SYNR)/(1+REFDV)=80MHz;
    _asm(nop); //BUS CLOCK=40M
    _asm(nop);
    while(!(CRGFLG_LOCK==1));
    CLKSEL_PLLSEL=1;//engage PLL to system;
}
//------------ATD 初始化---------------
void atd_init(void)
{
    ATD0CTL1=0x00;     //8 位精度，
   ATD0CTL2=0X60;   //快速清除标志位,禁止外部触发，不使能中断
   ATD0CTL3=0XD8;   //右对齐方式 完成转换后冻结 转换序列长度为 15
   ATD0CTL4=0X29;   //采样时间为 2MHZ
   ATD0CTL5=0X30;  //多通道转换和连续转换
    ATD0DIEN=0x00;//禁止数字输入
 }
//---------TIM 初始化 -----------
void TIM_init(void)
{
  PACTL=0X50;
  PACNT=0X0000;
}
//------------------
```

```
void Timer(void) {
   PITCFLMT_PITE=0; //disable PIT
   PITCE_PCE0=1;   //enable timer channel 0
   PITMTLD0=40-1;  //time base  240 clock cycles ,it's 0.1M Hz
   PITMUX=0X00; // ch0 connected to micro timer 0
   PITLD0=40000-1;  //INTVERAL micro time bases
   PITINTE_PINTE0=1;  //enable interupt channel 0
   PITCFLMT_PITE=1;  //enable PIT
}
//--------------PWM 初始化-------------
void pwm_init(void)
{
     PWMCTL=0XA0;     //设置 6,7  2,3 通道连级
     PWMCTL_CON45=1;
     PWMPRCLK=0X22;     //SA_COLK=10MHZ  SB_COLK=10MHZ
     PWMSCLA=0X05;         //COLKSA=1MHZ
     PWMSCLB=0X05;         //COLKSB=1MHZ
     PWMCLK=0X88;          //PWM23 选择 SB_CLK  6.7=SB
     PWMCLK_PCLK5=1;
     PWMPOL=0X08;       //极性选择起始为高电平
     PWMCAE=0X00;          //左对齐方式
     PWMCNT23=0X0000;
     PWMCNT67=0X0000;
     PWMCNT45=0X0000;
     PWMPER23=20000;     //设置周期为 20ms
     PWMPER67=1000;    //F=1kHZ  PWM67 驱动电机正转
     PWMPER45=1000;
     PWMDTY23=1419;
     PWMDTY67=0;
     PWMDTY45=0;
     PWME_PWME3=1;
     PWME_PWME7=0;
     PWME_PWME5=0;
}
//*************循环控制函数*************
//--------------读 ATD 转换值---------------
void sam_position(void)
{
        unsigned char i;
        ten_line=0;
       // delay();
   while(!ATD0STAT2_CCF0)  ;
        sam_atd_g[11]=ATD0DR0L;
   while(!ATD0STAT2_CCF1)  ;
        sam_atd_g[10]=ATD0DR1L;
```

```
   while(!ATD0STAT2_CCF2)  ;
        sam_atd_g[9]=ATD0DR2L;
   while(!ATD0STAT2_CCF3)  ;
        sam_atd_g[8]=ATD0DR3L;
   while(!ATD0STAT2_CCF4)  ;
        sam_atd_g[7]=ATD0DR4L;
   while(!ATD0STAT2_CCF5)  ;
        sam_atd_g[6]=ATD0DR5L;
   while(!ATD0STAT2_CCF6)  ;
        sam_atd_g[5]=ATD0DR6L;
   while(!ATD0STAT2_CCF7)  ;
        sam_atd_g[4]=ATD0DR7L;
   while(!ATD0STAT2_CCF8)  ;
        sam_atd_g[3]=ATD0DR8L;
   while(!ATD0STAT2_CCF9)  ;
        sam_atd_g[2]=ATD0DR9L;
   while(!ATD0STAT2_CCF10)  ;
        sam_atd_g[1]=ATD0DR10L;
    for(i=1;i<12;i++)
     {
      if(sam_atd_g[i]>adc_limit) sam_atd_g[i]=0;
        else    {
        sam_atd_g[i]=1;ten_line++;
       }
     }
       check_start();
       if(ten_line>=8) {
       ten_flag=1;
       for(i=1;i<6;i++)
       if(sam_atd_g[i]&&sam_atd_g[i+1]&&
sam_atd_g[i+2]&&sam_atd_g[i+3]&&
sam_atd_g[i+4]&&sam_atd_g[i+5]&&
sam_atd_g[i+6])   ten_flag=1;
     }
 if(ten_flag==0) ten_flag=0;
   else if (ten_flag==1&&(ten_line==1||
ten_line==2||ten_line==3))  {
ten_flag=0; ten_2_line++;
                  }  //ten_2_line为1减
}
//------------检测起跑线--------
void check_start(void)
{
      unsigned char i,j=0;
 start=0;
```

```
 for(i=1;i<12;i++)
  if(sam_atd_g[i]^sam_atd_g[i+1])  j++;
  if(j>=4)
        {
  if(sam_atd_g[3]&&((!sam_atd_g[2])&&
(!sam_atd_g[4]))&&((!sam_atd_g[1])&&
(!sam_atd_g[5]))&&sam_atd_g[7]) start=1;
  else if(sam_atd_g[3]&&((!sam_atd_g[2])&&
(!sam_atd_g[4]))&&(sam_atd_g[1]&&
sam_atd_g[5])&&sam_atd_g[7])     start=1;
 else if((sam_atd_g[3]&&sam_atd_g[4])&&
((!sam_atd_g[2])&&(!sam_atd_g[5]))&&
(sam_atd_g[1]&&sam_atd_g[6])&&sam_atd_g[7]) start=1;
 else if(sam_atd_g[4]&&((!sam_atd_g[3])&&
(!sam_atd_g[5]))&&((!sam_atd_g[2])&&
(!sam_atd_g[6]))&&sam_atd_g[8])  start=1;
 else if(sam_atd_g[4]&&((!sam_atd_g[3])&&
(!sam_atd_g[5]))&&(sam_atd_g[2]&&
sam_atd_g[6])&&sam_atd_g[8])     start=1;
 else if((sam_atd_g[4]&&sam_atd_g[5])&&
((!sam_atd_g[3])&&(!sam_atd_g[6]))&&
(sam_atd_g[2]&&sam_atd_g[7])&&sam_atd_g[8])     start=1;
 else if(sam_atd_g[5]&&((!sam_atd_g[4])&&
(!sam_atd_g[6]))&&((!sam_atd_g[3])&&
(!sam_atd_g[7]))&&sam_atd_g[9])               start=1;
 else if(sam_atd_g[5]&&((!sam_atd_g[4])&&
(!sam_atd_g[6]))&&(sam_atd_g[3]&&
sam_atd_g[7])&&sam_atd_g[9])                    start=1;
 else if((sam_atd_g[5]&&sam_atd_g[6])&&
((!sam_atd_g[4])&&(!sam_atd_g[7]))&&
(sam_atd_g[3]&&sam_atd_g[8])&&sam_atd_g[9])     start=1;
 else if(sam_atd_g[6]&&((!sam_atd_g[5])&&
(!sam_atd_g[7]))&&((!sam_atd_g[4])&&
(!sam_atd_g[8]))&&sam_atd_g[10])              start=1;
else if(sam_atd_g[6]&&((!sam_atd_g[5])&&
(!sam_atd_g[7]))&&(sam_atd_g[4]&&
sam_atd_g[8])&&sam_atd_g[10])                   start=1;
 else if((sam_atd_g[6]&&sam_atd_g[7])&&
((!sam_atd_g[5])&&(!sam_atd_g[8]))&&
(sam_atd_g[4]&&sam_atd_g[9])&&sam_atd_g[10])    start=1;
 else if(sam_atd_g[7]&&((!sam_atd_g[6])&&
(!sam_atd_g[8]))&&((!sam_atd_g[5])&&
(!sam_atd_g[9]))&&sam_atd_g[11])              start=1;
 else if(sam_atd_g[7]&&((!sam_atd_g[6])&&
(!sam_atd_g[8]))&&(sam_atd_g[5]&&
```

```
sam_atd_g[9])&&sam_atd_g[11])    start=1;
 else if((sam_atd_g[7]&&sam_atd_g[8])&&
((!sam_atd_g[6])&&(!sam_atd_g[9]))&&
(sam_atd_g[5]&&sam_atd_g[10])&&sam_atd_g[11])   start=1;
else if(sam_atd_g[8]&&((!sam_atd_g[7])&&
(!sam_atd_g[9]))&&((!sam_atd_g[6])&&
(!sam_atd_g[10]))&&sam_atd_g[4])   start=1;
else if(sam_atd_g[8]&&((!sam_atd_g[7])&&
(!sam_atd_g[9]))&&(sam_atd_g[6]&&
sam_atd_g[10])&&sam_atd_g[4])      start=1;
 else if((sam_atd_g[8]&&sam_atd_g[9])&&
((!sam_atd_g[7])&&(!sam_atd_g[10]))&&
(sam_atd_g[6]&&sam_atd_g[11])&&sam_atd_g[4])   start=1;
else if(sam_atd_g[9]&&((!sam_atd_g[8])&&
(!sam_atd_g[10]))&&((!sam_atd_g[7])&&
(!sam_atd_g[11]))&&sam_atd_g[5])   start=1;
else if(sam_atd_g[9]&&((!sam_atd_g[8])&&
(!sam_atd_g[10]))&&(sam_atd_g[7]&&
sam_atd_g[11])&&sam_atd_g[5])  start=1;
        }
        if(start==1){
        if( ten_2_line==0 )       checkstart=1 ;
        else if(ten_2_line>=2)    checkstart=2;
        }
}
//--------------------------计算车的当前位置-----------------
void car_position(void)
{
  unsigned int i,j,m,n;
  int temp,k,positn_error;
  times=0;
  for(i=1;i<5;i++)
 {
    sam_position();
    k=0;
    black_sensor_number=0;
    for(j=1;j<12;j++)
    {
      if(sam_atd_g[j])
       {
           black_sensor_number++;k=k+j;
       }
     }
      if(k>0)
        {
```

```
            k*=2;k=k/black_sensor_number;k=k-12;
            temp_positn[i]=k; //计算出来的车位置
          }
       else  times++;
  }
   if(times<4)
  {
     for(m=1;m<5;m++)
        {
         for(n=m+1;n<5;n++)
            {
               if(temp_positn[m]>temp_positn[n])
                  {
                     temp=temp_positn[m];
                    temp_positn[m]=temp_positn[n];
                   temp_positn[n]=temp;
                  }
            }
          }
            car_positn=(temp_positn[2]+temp_positn[3])/2;
   }
  positn_error=car_positn-pre_car_positn;
  positn_error=absolute(positn_error);
  if(positn_error>=10)   car_positn=pre_car_positn;
  pre_car_positn=car_positn;
}
//-----------------计算车的速度-----------------
void speed(void)
{  if(speedmax<=pulse_speed)   speedmax=pulse_speed;
    if(speedmax>=40)
       speedmax=40;
        ideal_speed=speed_table2[absolute(car_positn)];
        speed_error=ideal_speed-pulse_speed;
        dri_flag=1;
         pid();
   if(pulse_speed>=9)
     {
        if(car_positn==10)
          {
             angle_data=1265;
             dri_flag=0;
             car_driver=600;
           }
        if(car_positn==-10)
           {
```

```
                angle_data=1573;
                dri_flag=0;
                car_driver=600;}
        }
        if(car_driver>=1000)  car_driver=1000;
        else if(car_driver<=0)  car_driver=0;
 }
//--------------负值转为正值------------------
unsigned int absolute(int x)
{
     if(x<0)   x=-x;
     return(x);
}
//----------------驱动电机-------------
void driver(void)
{
    if(checkstart==2) stop();
     if(dri_flag==1)
       {
          PWME_PWME5=1;
          PWME_PWME7=0;
          PWMDTY45=car_driver;
       }
else if(dri_flag==0)
   {
       PWME_PWME5=0; PWME_PWME7=1;
       PWMDTY67=car_driver;PWMDTY23=angle_data;
   }
}
//------------PID 控制----------------

void pid(void)
{
    signed int  d_error, dd_error;  //error=speed_error
   error=ideal_speed-pulse_speed;
    d_error=error-pre_error;  //d_error 当前速度差与上一次速度差之差
    dd_error=d_error-pre_d_error;
    pre_error=error;    //存储当前偏差
    pre_d_error=d_error;
    pk+=kp*d_error+ki*error+kd*dd_error;
    if(pk<=0)    pk=0;
    else if(pk>=1000)  pk=1000;
    car_driver=pk;
}
```

```
//-------------停车控制--------------
void stop()
{
     PWMDTY23=Angle_Center;
     PWME_PWME7=0;
     PWME_PWME5=0;
     for(;;) ;
}
//--------------延时函数-------------------
void delay(unsigned int X)
{
      unsigned int i,j;
      for(i=0;i<X;i++)
      for(j=0;j<3;j++);
}
//--------测速函数(TIM_ISR)------------------
#pragma CODE_SEG NON_BANKED
void interrupt 66 PIT0Interrupt(void)
{
     pulse_speed=PACNT;
     // _asm  LDAA  PITTF;
     //_asm  MOVB  #$01,PITTF;
      PITTF_PTF0=1;
      PACNT=0X0000;
}
#pragma CODE_SEG DEFAULT
//==========*****THE CODE END*****==================
void getATDvlaue(void)
{
  int i,k,j,min[10];
  i=0;
    for(k=0;k<10;k++)    min[k]=255;
    for(i;i<10;i++) {
         while(!ATD0STAT2_CCF0)  ;
            sam_atd_g[11]=ATD0DR0L;
         while(!ATD0STAT2_CCF1)  ;
             sam_atd_g[10]=ATD0DR1L;
         while(!ATD0STAT2_CCF2)  ;
             sam_atd_g[9]=ATD0DR2L;
         while(!ATD0STAT2_CCF3)  ;
            sam_atd_g[8]=ATD0DR3L;
         while(!ATD0STAT2_CCF4)  ;
            sam_atd_g[7]=ATD0DR4L;
         while(!ATD0STAT2_CCF5)  ;
            sam_atd_g[6]=ATD0DR5L;
```

```
        while(!ATD0STAT2_CCF6)  ;
           sam_atd_g[5]=ATD0DR6L;
        while(!ATD0STAT2_CCF7)  ;
           sam_atd_g[4]=ATD0DR7L;
        while(!ATD0STAT2_CCF8)  ;
           sam_atd_g[3]=ATD0DR8L;
        while(!ATD0STAT2_CCF9)  ;
           sam_atd_g[2]=ATD0DR9L;
        while(!ATD0STAT2_CCF10)  ;
           sam_atd_g[1]=ATD0DR10L;
  for(j=1;j<=11;j++)
       if(min[i]>=sam_atd_g[j])
         min[i]=sam_atd_g[j];
  }
  adc_limit=(min[0]+min[1]+min[2]+min[3]+
  min[4]+min[5]+min[6]+min[7]+min[8]+min[9])/10+35;
}
```

致　谢

首先，感谢张平老师和陈剑桥老师的细心指导，由于两位老师的认真负责，我才能顺利的完成毕业设计。本文的研究、撰写直至论文的定稿，每个环节和步骤，老师都严格把关，精心指导，论文的字里行间都浸透了老师辛勤的汗水，在此向指导老师们表示深深的谢意！

在智能车电路的开发过程中宋正强老师，曾给了我很多帮助，对整个智能车系统的设计提供了不少的建议，指引着我克服了很多技术问题，在此向这位良师益友表示诚挚的感谢！

另外，对于一直陪同我制作智能车的伙伴，想发自肺腑地说一声谢谢，因为有他们的默默支持才能让我顺利完成毕业设计。

最后，感谢各位评阅老师，想跟你们说声：老师，您辛苦了！