第二届全国大学生“飞思卡尔”杯
智能汽车竞赛

技 术 报 告

附件 B 基于路径记忆的最佳赛车控制算法研究[注]

学　　校:　清华大学

队伍名称:　三角洲队（清华一队）

参赛队员:　林辛凡

李红志

黄　颖

带队教师:　李立国

关于技术报告和研究论文使用授权的说明

本人完全了解第二届全国大学生“飞思卡尔”杯智能汽车竞赛关保留、使用技术报告和研究论文的规定，即：参赛作品著作权归参赛者本人，比赛组委会和飞思卡尔半导体公司可以在相关主页上收录并公开参赛作品的设计方案、技术报告以及参赛模型车的视频、图像资料，并将相关内容编纂收录在组委会出版论文集中。

参赛队员签名：_____

带队教师签名：_____

日　　期：_____

目录

第一章　引言

1.1　概述

为响应教育部关于加强大学生的创新意识、合作精神和创新能力的培养的号召，我们组队积极参加了第二届“飞思卡尔”杯全国大学生智能汽车邀请赛。从 2007 年 3 月开始着手进行准备，历时近 6 个月，经过设计理念的不断进步共制作出 3 代赛车硬件平台及相关算法。鉴于 CCD 传感器视距远、信息丰富的特点和新技术路线带来的挑战，我们在今年的方案设计中选择了 CCD 传感器。但与去年传感器的创新点不同：去年采用连续化算法处理光电传感器信号以获得更多更精确的信息；今年的创新点是采用硬件二值化的方法降低图像信息量，以在有限计算能力下获得最高的分辨率和赛道中心检测精度。与之相配合，电路设计上增加了 DCDC 升压模块和 CCD 信号处理模块，并继续加强在电源管理、噪声抑制、驱动优化、整车布局等方面的研究工作，使智能车能够满足高速运行下的动力性和稳定性需求，获得了良好的综合性能和赛场表现。

本技术报告将针对我们的传感器信号处理设计安装、底盘参数选择、电路设计、HCS12 控制软件主要理论、控制算法等方面进行阐述，并列出了模型车的主要技术参数。

1.2　整车设计思路

1.2.1　控制系统

智能车的工作模式如图 1.1 所示：CCD 传感器拍摄赛道图像并以 PAL 制式信号输出到 CCD 信号处理模块进行二值化并进行视频同步信号分离，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；通过片上 AD 检测电池电压；舵机转向采用分段 PID 控制；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；而车速的目标值由默认值、运行安全监控和基于路径记忆的优化策略综合控制。

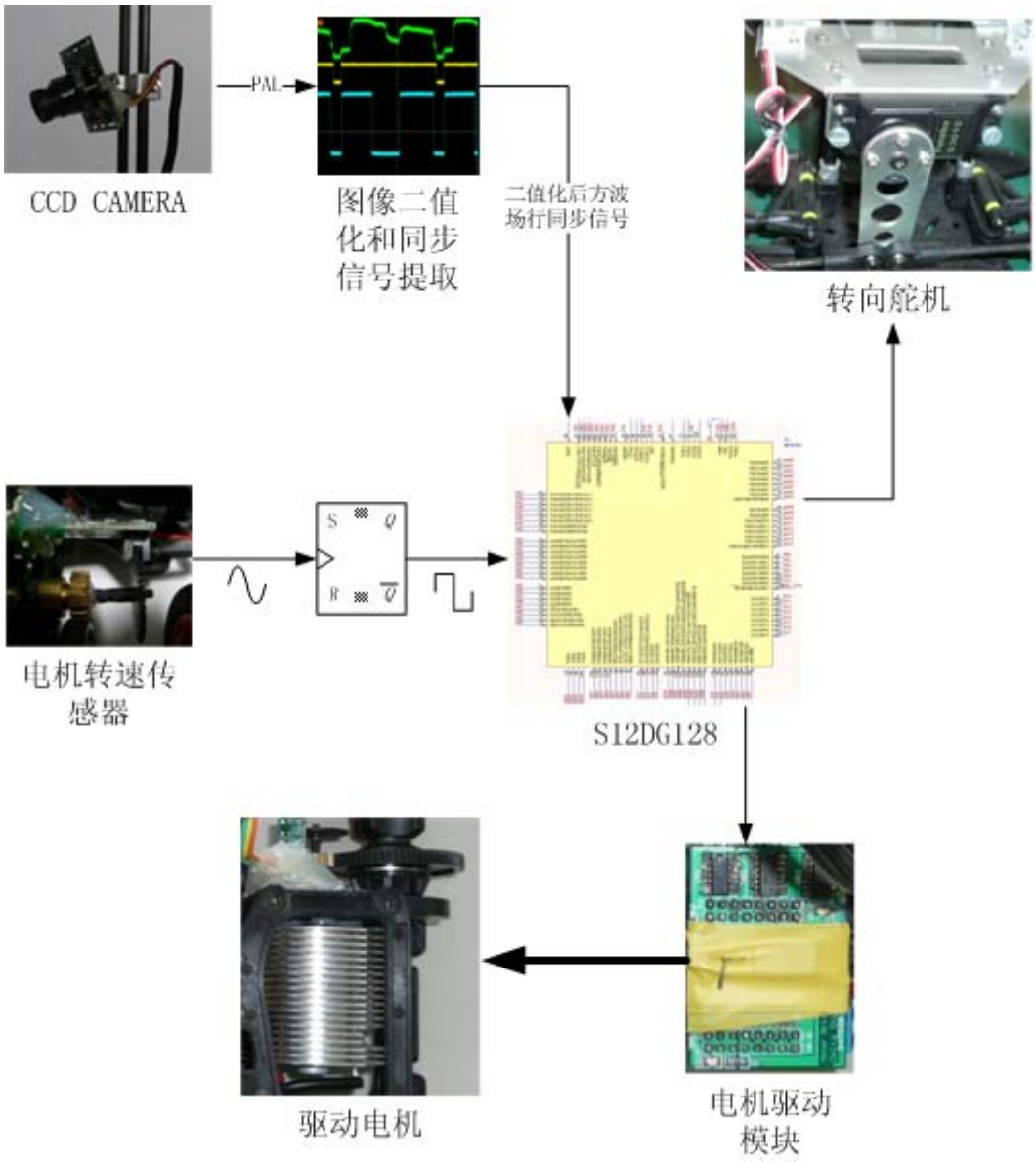


图 1.1　智能车的工作模式

1.2.2　电源系统

智能车电源系统如图 1.2 所示，由于在赛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，所以加强了电池端的滤波并在 DC-DC 后增加了 LDO。

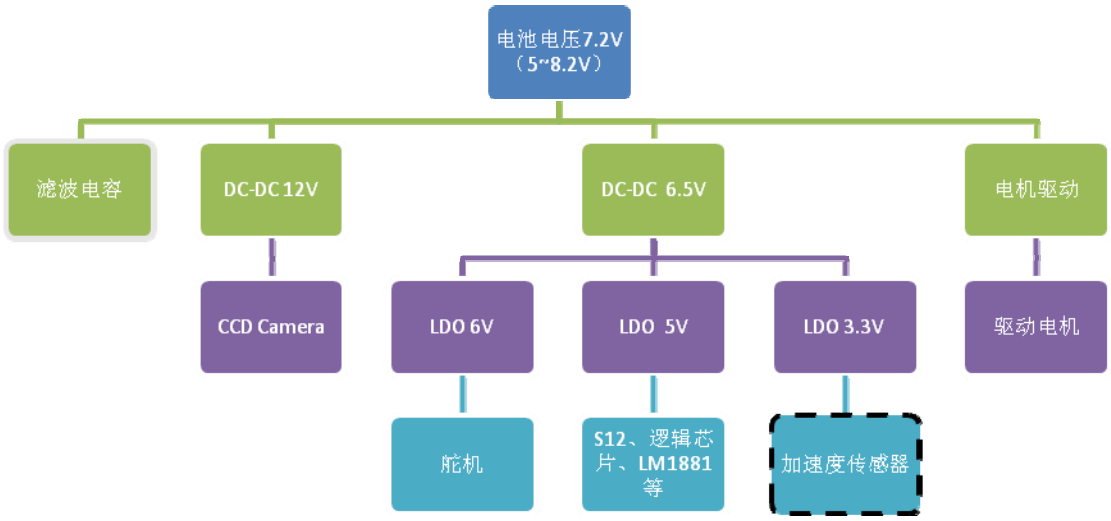


图 1.2　智能车电源系统

1.2.3　整车布局

鉴于赛车和赛道的特点，今年在整车布局上仍延续 2006 年赛车的思路，采用低重心紧凑型设计，并架高舵机以提高响应速度。而且由于驱动电路的改进，电机动力性提高的同时热负荷也显著增加了，所以为电机专门设计了散热片，提高了赛车持续调试和比赛的能力。在降低整车重心方面采用了低位主板的布局，同时设计了强度高质量轻的 CCD 安装架，并采用了裸板 CCD。

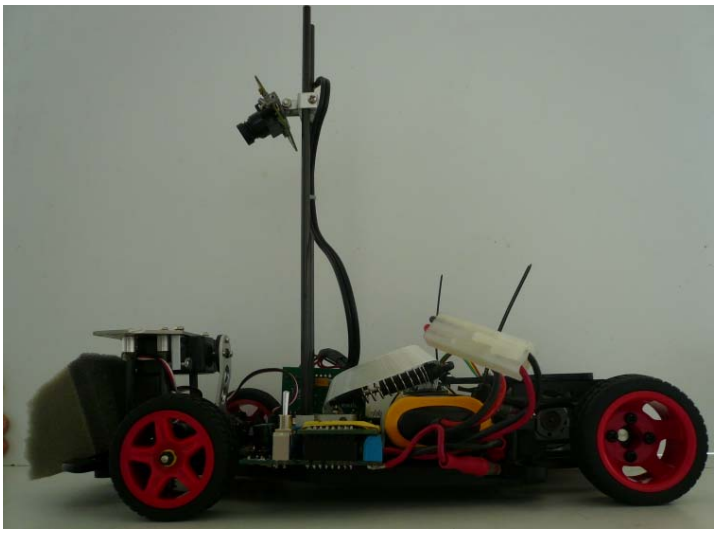


图 1.3 整车布局

图 1.4 电机散热片

1.2.4 赛车相关性能设计思想

智能车作为一个竞技作品，在设计过程中除了考虑赛场上的动力性、操纵稳定性、响应速度等关键指标外，还应充分考虑开发、调试和赛前准备过程可能遇到的种种问题。在赛车的设计、制造和配套设施上，我们都秉持赛场如战场的准则，在可靠性、后勤保障和换代步骤上都参考军备的设计思想，并在如下几个方面作了细致工作。

1. 保障赛车的可靠性：杜绝功败垂成的悲剧；
2. 提高赛车的维护性：预留备件、拆装方便、传感器重复定位性好；
3. 提高赛车和电池的续航能力：提高调试效率；
4. 建立合理的调试和出赛的辅助装备配套机制：运输、维修工具、备件等；
5. 建立赛车调试随车文档：硬件使用和调整说明、软硬件调试记录、设备负责人；
6. 加强设备管理，维护试验环境；
7. 建立人才培养机制：是绝大部分研究成果得到继承，减小研究进度的锯齿效应。

第二章　硬件设计

2.1　传感器选型及安装方案设计

2.1.1　传感器的选型

摄像头和光电传感器阵列是最常用的两种赛道检测方式，在第一届的智能车比赛中我们采用了光电传感器，而在今年的比赛中我们采用了 CCD 图像传感器。对于传感器的选择我们进行了如下分析:

我们设计的大前瞻光电传感器阵列（7 个传感器一字均布成 24cm 宽）可以达到 30~40cm 的前瞻距离，探测宽度可达 32cm，分辨率约 1mm，探测精度约为 1~5mm（依赖于模型标定精度和器件均一性），最高探测频率 10KHz，最大单次探测功耗 1.05mJ，在 50Hz 探测频率下的功耗为 52.5mW，对低频变化的环境光不敏感，对阴影不敏感，传感器总重（含安装架）不大于 30g，供电电压大于 3.3V，工作电压 9~12V。传感器辅助系统包括额定电流 1A 最大输出电压 12V 的低波纹 DC-DC 供电电路和底边驱动电路。占用 7 个 AD 端口和一个通用 IO。数据处理时间小于 1ms（总线频率 24MHz）。

市场上常见的摄像头有 CCD 和 CMOS 两类，CCD 多数采用 PAL 制式输出，CMOS 则有 PAL 模拟输出和 USB 数字信号输出两种，还有一些摄像头可以提供并行数字量输出。由于 CCD 的对比度高、动态特性好的优点，我们选用了市场上常见的安防用 CCD 摄像头，并拆除外壳使用其裸板。我们采购到的 CCD 型号为 DST-302，1Vp-p PAL 信号输出，工作电压 12V，额定电流为 150mA，即功率为 1.8W。PAL 信号的帧速为 25 帧/s，即有 50 场/s 的图像。其辅助系统包括一个额定电流 200mA 输出电压 12V 的低波纹 DC-DC 供电电路和一个基于 LM1881 的同步信号分离电路。参考上海交大 2006 年的技术报告，可知其在 32MHz 总线频率下的行有效采样率为 66，根据图 2.1 可以估算其图像中间行的赛道心分辨率约为 13.5mm，此时的前瞻距离为 32cm，远低于光电传感器的分辨率。而对于图 2.1 的图像而言，如果单片机不超频，最大前瞻距离约为 60cm，所以在 AD 采集 PAL 制式信号的方案下，由于 CCD 采样频率低的缺点，如果仅采用转向 PID 控制策略的话，潜力远没有光电传感器大。

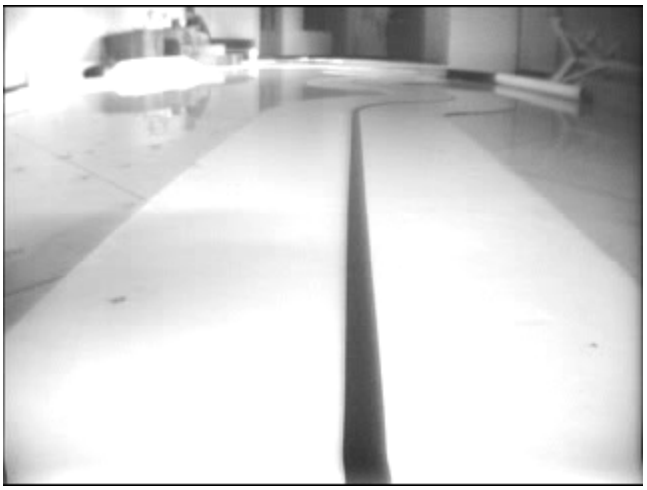


图 2.1 安装条件下车载摄像头获得赛道图像

但我们并没有因此放弃用摄像头作为赛道识别传感器的计划，在信号处理上下文章，以提高对图像的处理能力和识别精度。在最初我们主要向以下几个方向进行了试探性研究：

1. 采用数字信号输出的摄像头

通过初步市场调研发现，容易买到的多是 USB PC Camera，能够订购的有数字输出的 CMOS 芯片。前者需要在增加 USB 接收辅助电路，并且要破解其驱动程序，我们当时不具有这个技术能力，所以放弃了这个方案。后者需要设计整个 CMOS Camera，这个工作量和技术难度都不小，开发风险也比较大，所以也放弃了。

2. 采用专用视频处理芯片对信号进行数字化：

通过调研发现，有专用的视频信号处理芯片可以对 PAL 信号进行处理，并输出并行数据，如 XRD4460A。XRD4460A 具有一个 10 位 A / D 转换器，最高采样速率高达 16MHz，内置高带宽的差分相关双采样器(CDS)和 8 位的数字可编程增益放大器(PGA)。模拟偏移量可控制，差分信号输入，差分外部时钟，片内带有输入缓存和采样 / 保持器，10 位并行数据输出。

3. 采用外部 AD 采集 PAL 信号

由于可以买到采样速率很高的 AD 转换芯片，曾考虑使用外部 AD 进行 PAL 信号采样。

4. 采用模拟电路对 PAL 信号进行二值化

在 2006 年采用光电传感器的时候，为了充分利用光电传感器的信息，采用

了 AD 采样和传感器信息融合的思路开发了连续算法。而今年则反其道行之，为了降低单片机的计算负荷而对 PAL 信号进行硬件二值化，将灰度图象转换成黑白图像，这样就不需要用 AD 就可以采集图象了。而且普通 IO 的操作速度要比 AD 快，使提高分辨率成为可能。由于通过模拟电路实现二值化的硬件比较容易实现，所以这个方案最快进入了测试阶段并取得来满意效果，所以最后采用了此技术路线而放弃了其它方案的研究。

在传感器类型的选择上，我们并不是因为 CCD 的优点而放弃了光电方案，而是出于对探求新问题的渴望。而且我们发现了大有文章的创新点，使基于硬件二值化的 CCD 信号处理成为我们研究的焦点。

2.1.2 传感器的供电电路

由于 CCD 传感器需要 12V 供电，所以需要通过 DC-DC 为其提供一个 12V 的电源。我们采用了专用的 DC-DC 控制芯片搭建此电路，并且用一个 LDO 来获得低波纹的电源。如图 2.2 所示，此电路的最低工作电压可达 3V，输出电流可达 500mA。

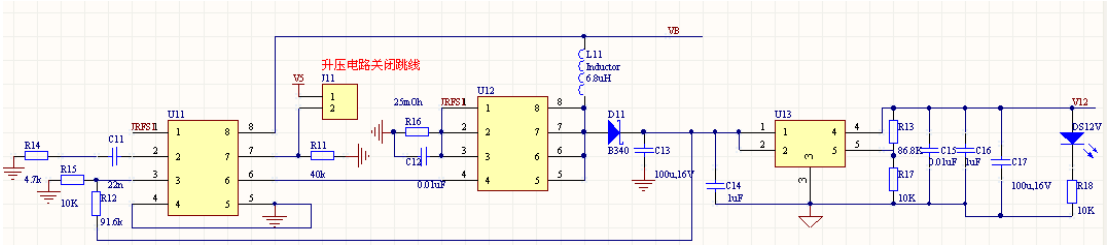


图 2.2 DC-DC 升压电路

2.1.3 传感器的信号处理

对于 CCD 输出的 PAL 信号，首先需要从中分离出场同步信号和行同步信号，以使单片机能够分辨信号输出的进程，从而正确的选择采集信号的时刻和所采集的信号在图像中的坐标。同时要采用图像二值化电路将 PAL 信号处理成黑白信号。如图 2.3 所示，图 2.3 为最初的采用固定参考电平进行二值化的零代二值化电路，图 2.4 为 1.0 版的通过移相边沿检测识别的二值化电路，后来又开发 2.0 版的微分边缘检测二值化电路和 3.0 版的黑白双限微分边缘检测二值化电路。

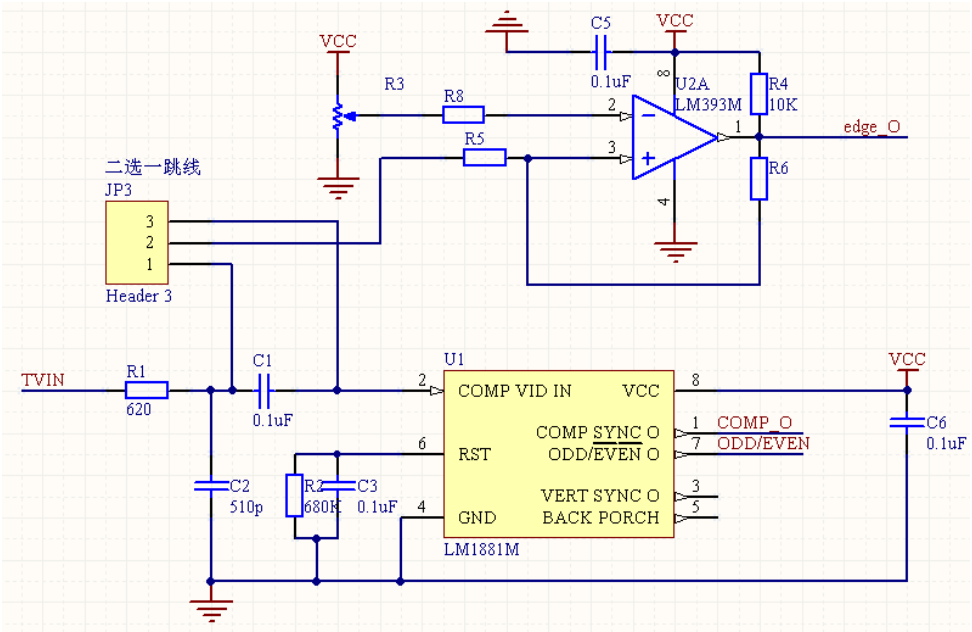


图 2.3 采用固定参考电压的二值化电路

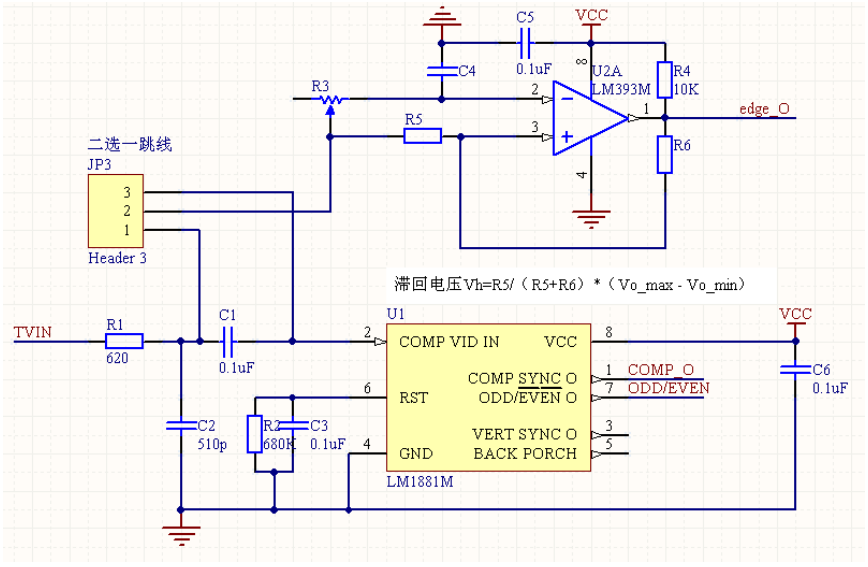


图 2.4 采用边沿检测的二值化电路

如图 2.5 所示，是 1.0 版二值化电路的处理效果。其中，2 通道为原始的 PAL 信号，1 通道为硬件二值化处理之后的信号。

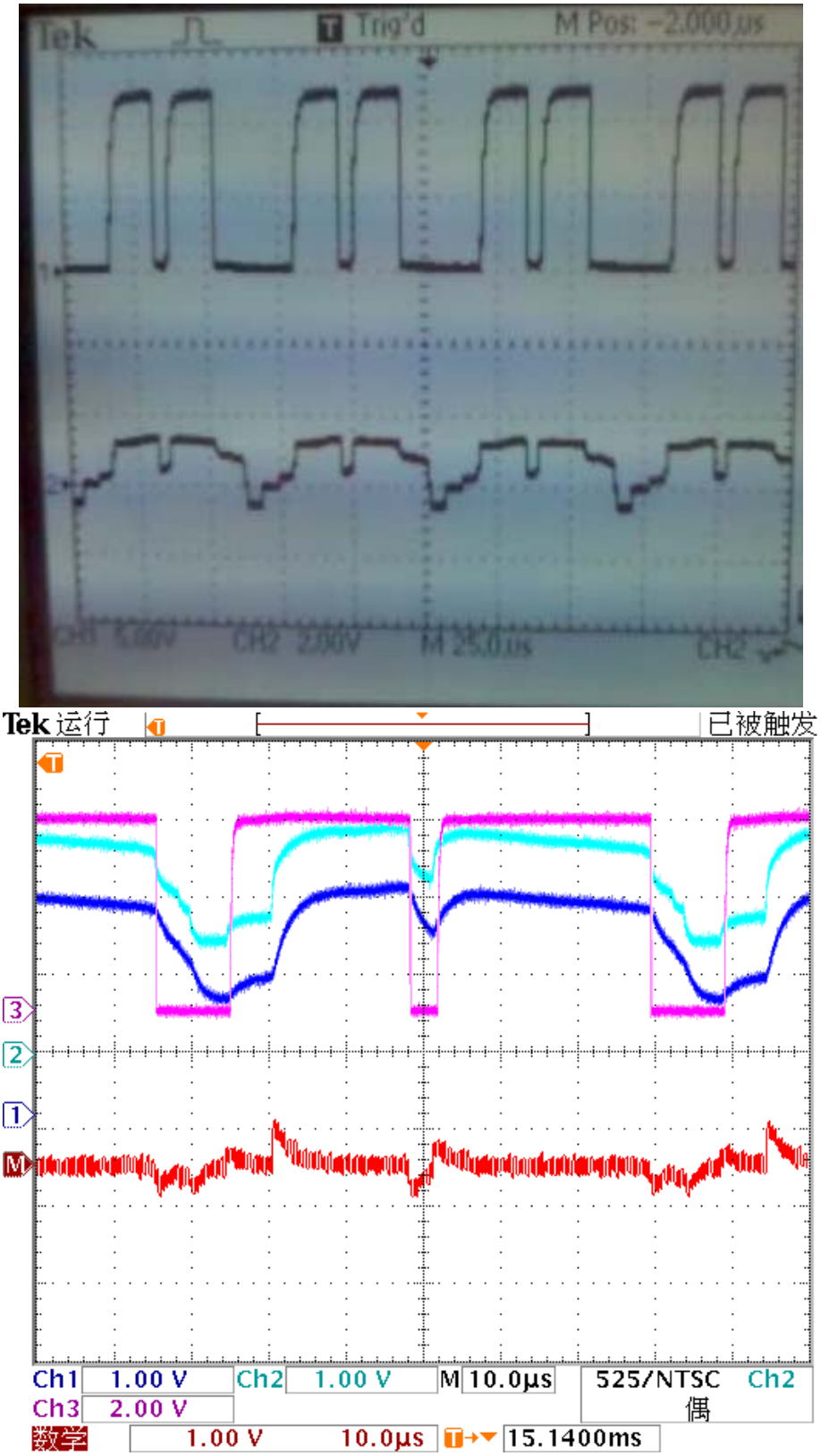


图 2.5 PAL 信号硬件二值化处理效果

（上图为零代硬件处理结果，下图为 1.0 版硬件处理效果）

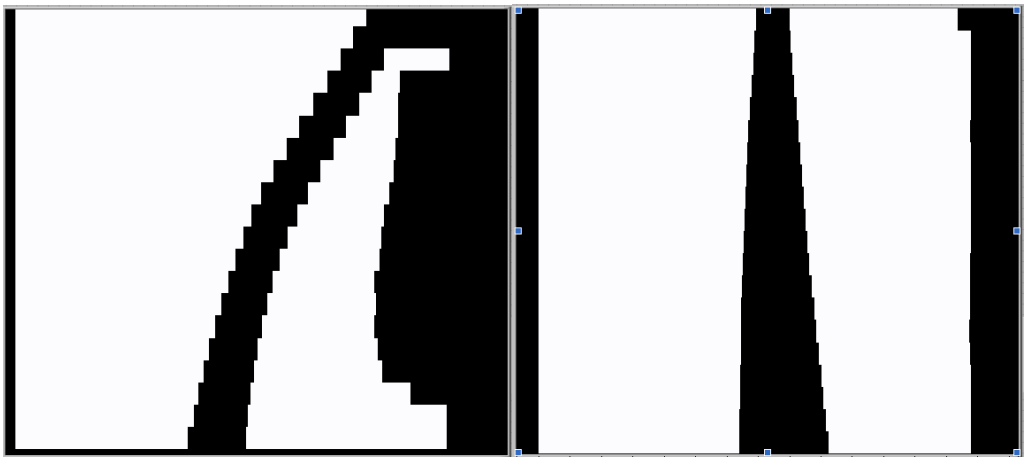


图 2.6 硬件二值化后单片机采集的图像（左：弯道 右：直道）

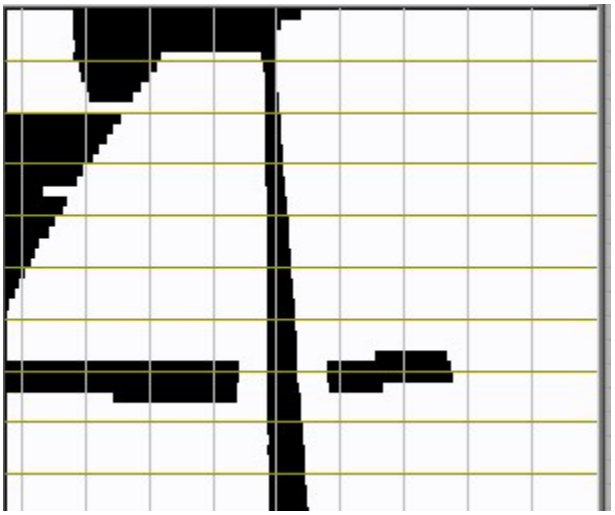


图 2.7 图 1.3CCD 安装条件下对起点图像的采集结果

通过数据采集程序效率的优化，在总线频率 24MHz 条件下可以准确捕获宽度 1.5us 的黑线信号。并可以达到 1200 多的行分辨率和很高的中心解析精度。

2.1.4 传感器的安装方案设计

对于 CCD 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，使远端赛道图像变宽，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 CCD 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，使赛车在高速过弯的时候容易发生摇

晃、颠簸甚至倾覆的危险。由于图像信号处理方式上的改进极大的提高了对图像的分辨能力，降低了对高度的要求，所以我们的安装高度确定为 25cm，俯视角度约为 30 度。为了降低 CCD 高位安装对整车中心高度的影响，采用了裸板 CCD Camera 并专门设计了质量轻强度高的铝合金安装架，而且采用了碳纤维的主桅，使高位安装的零件质量控制在 30g 以下。

图 2.8 CCD camare 和安装架

2.1.5 传感器技术方案综述

首先采用对比度高动态性能优越的 CCD 摄像头，然后创新性的采用了硬件二值化电路对图像进行预处理，使单片机可以采用普通 IO 口对图像信号进行采集，从而大大提高了分辨率，为图像处理提供了良好的原始数据。并且为了进一步提高有效视野，最大限度的优化了图像采集程序，从而使分辨率进一步提高。

2.2 模型车机械设计

由于赛车和电机、舵机等都是不能做改动的，模型车机械设计较少，我们主要进行了传感器支架的设计安装、底盘钻孔、舵机安装设计和电机散热片设计。

2.2.1. 整车布局

为了提高赛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 CCD 安装的轻量化设计、控制整车质心尽量靠近中性转向点。经过上述设计的赛车质心基本在中心转向点附近，前轮的静态侧翻极限是 54 度，后轮的侧翻极限是 65 度，而在赛道上侧滑极限是 40 度，约为前轮静态侧翻极限的 74%和后轮侧翻极限的 62%，所以赛车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 CCD 及其安装架的重量。首先，在 CCD 摄像头的选择上采用了镜头固定在电路板上的裸板 CCD，去掉了 CCD 外壳的重量。然后是设计了轻巧的铝合金 CCD 夹持组件，并采用了碳纤维管作为安装 CCD 的主桅，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。同时为了使整车重心维持在中性转向点附近，而把安装架底座设计在原车减振器和天线座的位置上，兼有 CCD 架底座的作用和减振器底座的作用，而且具有很高的重复定位精度和刚度。使 CCD 便于拆卸和维修，具有赛场快速保障能力。

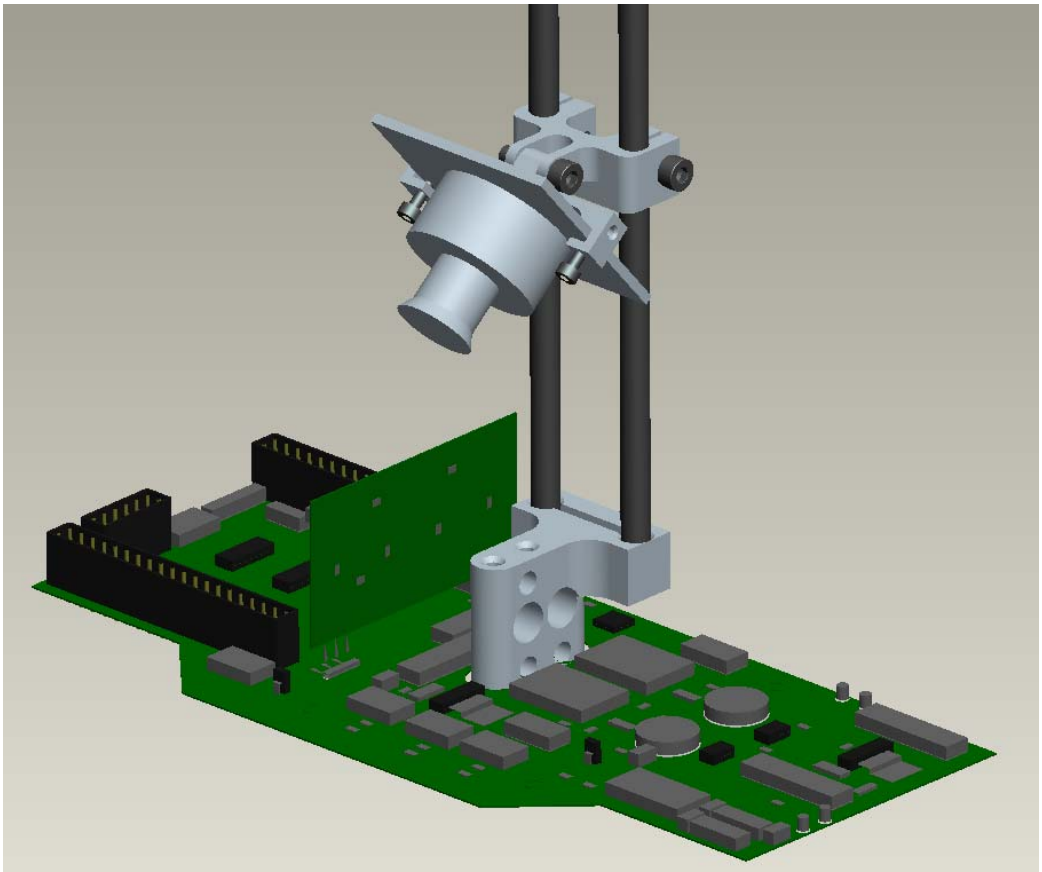


图 2.9　CCD 及安装架安装效果图

在 CCD 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 CCD 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 CCD 架拆下和装上，不必顾虑摄像头的位置会发生改变。

2.2.3. 主板安装

如图 1.3 和图 2.9 所示，主板安装紧贴赛车底盘，放在电池和舵机之间的空间里，用螺钉固定。同时主板上留出减振器座的安装孔，保留原车的减振器设计。

2.2.4. 电机散热设计

为了获得最大的动力性，电机在调速过程中多工作在正负外特性上，工作电流很大，电机发热严重。为了防止电机过热降低性能或烧毁线圈，专门设计了电机散热片（如图 2.10），并用导热硅胶粘接在电机外壳上。安装散热片后可以有效降低电机的工作温度，使赛车能够连续运行 10 分钟以上而不发生电机过热。省去了调试间隙为电机降温的时间，大大提高了调试效率。

图 2.10　电机散热片

2.3　电路设计

为了提高电路硬件的可维护性并减低升级成本，采用了模块化设计。如图在最初的电路选型和验证阶段将硬件分为了：主板、电源和光电管驱动子板、电机驱动子板、CCD 信号处理子板、S12 子板（组委会提供）5 个模块。

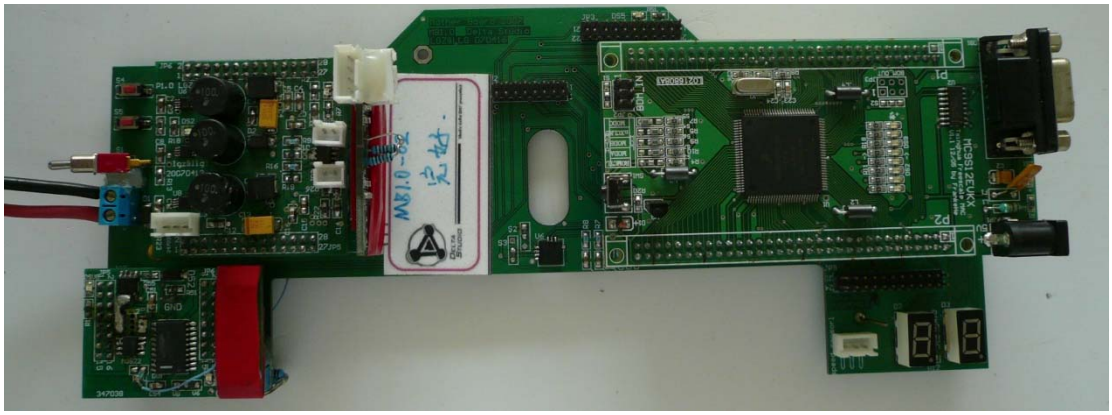


图 2.11　1.0 版智能车电路硬件平台

本届大赛备赛过程中共设计制造了 3 代电路硬件平台，其中，母板升级了 3 代（1.0 /2.0 /3.0）、电机驱动升级了 1 代半（1.0/1.1）、CCD 信号处理板升级了 4 代半（0/1.0/2.0/2.1/3.0）、电源子板升级了 3 代（1.0/2.0/3.0）、S12 最小系统升级了 2 代（1.0/2.0）、数码显示子板升级了 2 代（2.0/3.0），而在决赛作品上采用的硬件平台是由 3.0 版的母板模块、1.1 版的电机驱动模块、2.1 版的 CCD 信号处理模块、2.0 版的 S12 最小系统模块和 3.0 版的数码管显示模块组成的，而且兼容电源子板 3.0，具有支持 13 个光电传感器的能力。从上述数据可以看出，不仅可以通过硬件兼容降低升级成本，而且还可以使升级换代进度错开，均衡工作压力，保证软件调试工作连续进行，降低了换代过程的风险。

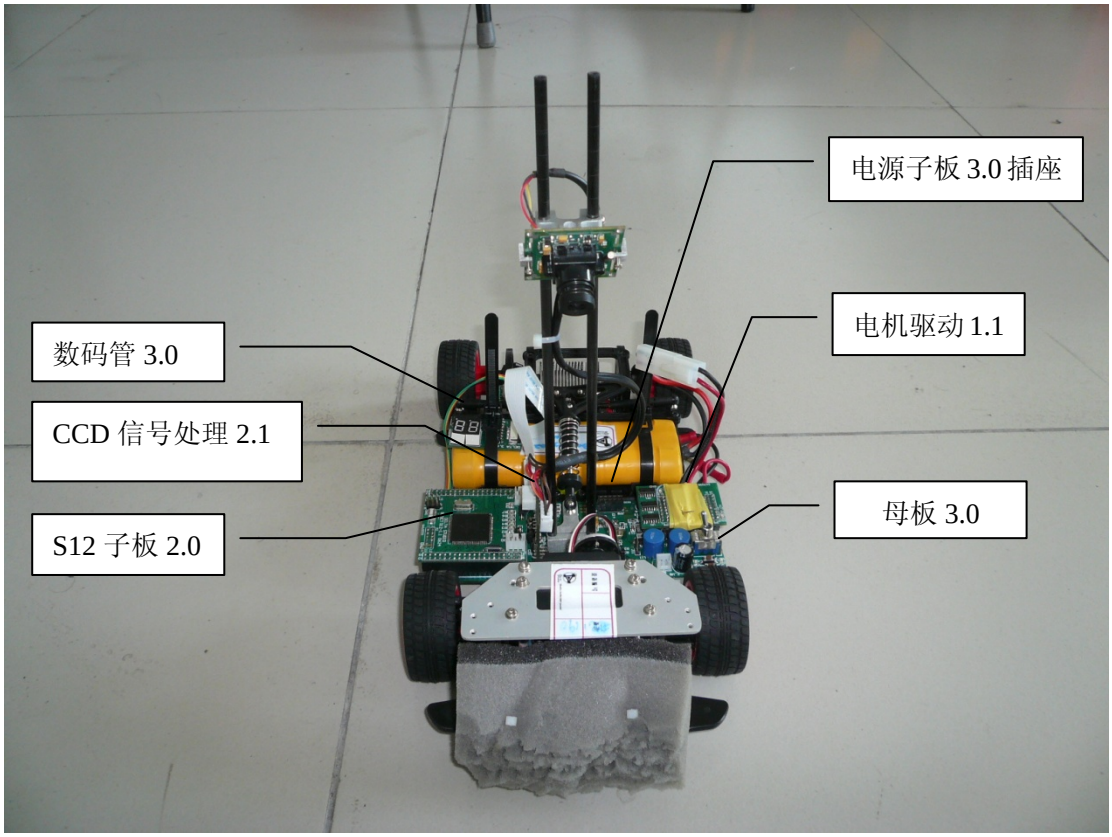


图 2.12 整车电路布局

2.3.1. 电源电路

整车电源构架和 DC-DC 的电路在图 1.2、图 2.2 和图 2.13 中已经有详细信息，这里不再赘述。

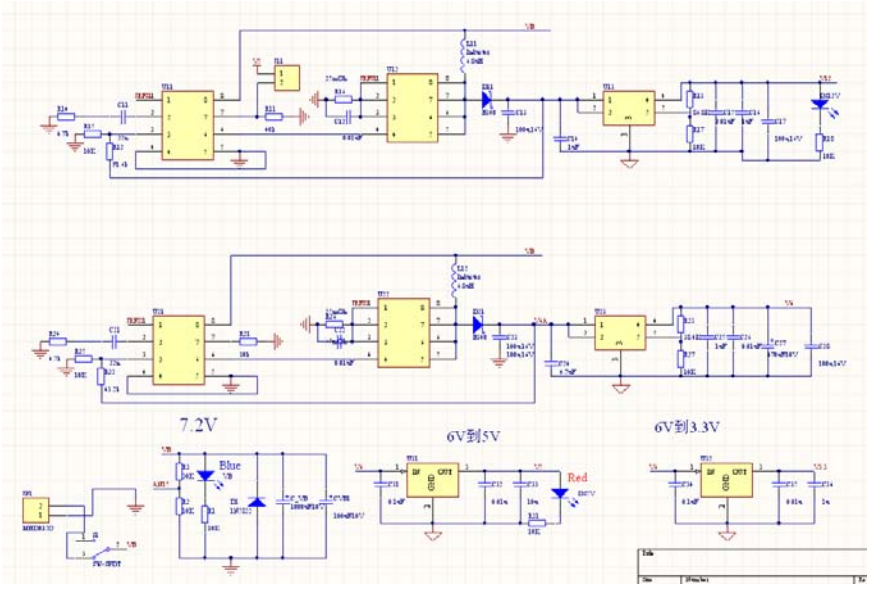


图 2.13 整车电源系统

2.3.2. 电机驱动电路

在 2006 年的第一届智能车邀请赛中我们采用的驱动电路（如图 2.14）是基于 MC33886 驱动芯片的。MC33886 芯片的额定输出电流是 5.2A，而赛车电机在起步和制动过程中的最大工作电流较高，长期工作在这种条件下，容易造成驱动芯片性能衰退甚至烧毁。所以在去年的设计方案中我们采用了多片 MC33886 并联的方式，而且采用制作成电机驱动子板的方式使其便于更换。但这都不是解决问题的根本办法。

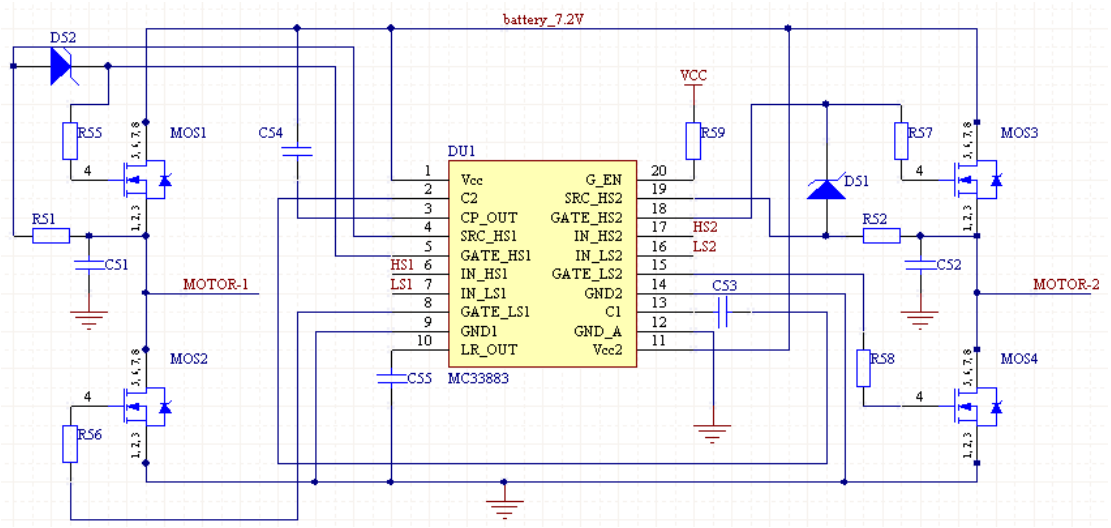


图 2.14 MC3886 电机驱动电路

在本届大赛中我们采用了如图 2.15 所示的电机驱动电路，这个电路仍是一个 PWM 调速的 H 桥全桥驱动电路，采用大电流低内阻的 N 型 MOSFET 构成桥路开关，使驱动电路获得足够的带载能力和高效率。为了兼容 2006 年开发的 MC33886 子板，1.0 和 2.0 两代驱动子板都采用了和 MC33886 子板相同的控制方式和工作模式，并且在接插件上完全兼容 MC33886 的子板。为了达到控制方式相同的目的，用了一套逻辑电路将两位控制字和 PWM 信号翻译成 MC33883 的驱动指令，再通过 MC33883 驱动 MOSFET，最后由 MOSFET 实现桥路的开关。

驱动电路逻辑真值表如表 2.1 所示。

表 2.1 逻辑控制电路真值表

IN1	1	0	1	0
IN2	0	1	1	0
PWM	1	1	X	1

状态	前进	后退	FreeRun	FastStop

其中前进、后退、FastStop 三种状态都可以被 PWM 信号调制，但 FreeRun 状态不受 PWM 信号控制。

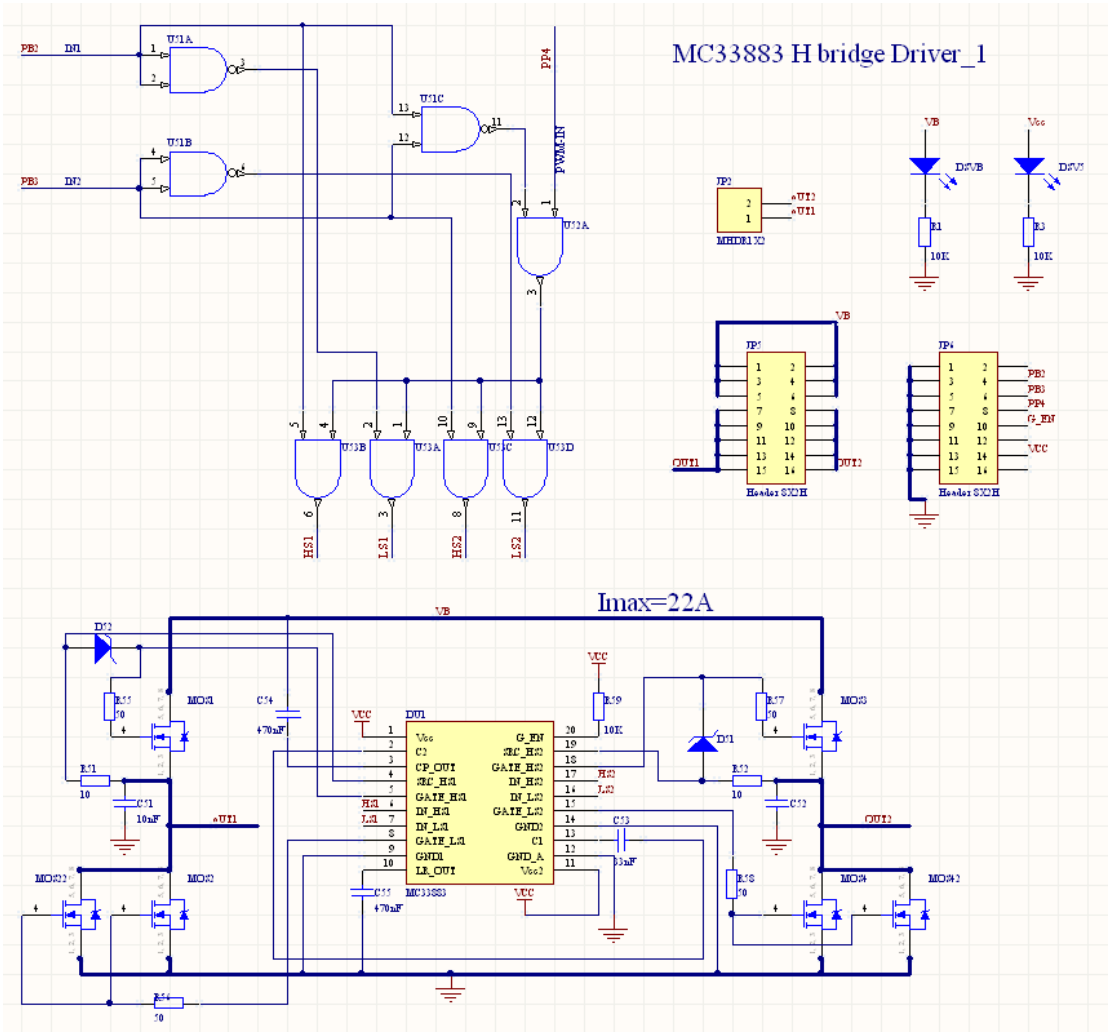


图 2.15 电机驱动子板电路

第三章　软件设计

3.1　HCS12 控制软件主要理论、算法说明

3.1.1　基于 CCD 的路径识别算法

3.1.1.1　CCD 信号采集

硬件上采用了 CCD 摄像头传感器，经过硬件二值化电路处理后输出与实际路径信息对应的高低逻辑信号，利用 S12 单片对此信号以及经视频分离电路处理后的同步信号进行采集，得到完整的图像信息。

具体采集过程主要如下。效果如图 3.1、3.2 所示。

（1）　利用普通 IO 口对视频分离之后的场同步信号进行采集，以进行场信号同步；

（2）　对行同步信号采用中断采集的方式，当行同步信号来临时，产生中断，进入中断子程进行图像信号的采集；

（3）　在行中断子程中，利用 IO 端口对每行二值化处理之后的逻辑信号进行采集，得到赛道图像信息。

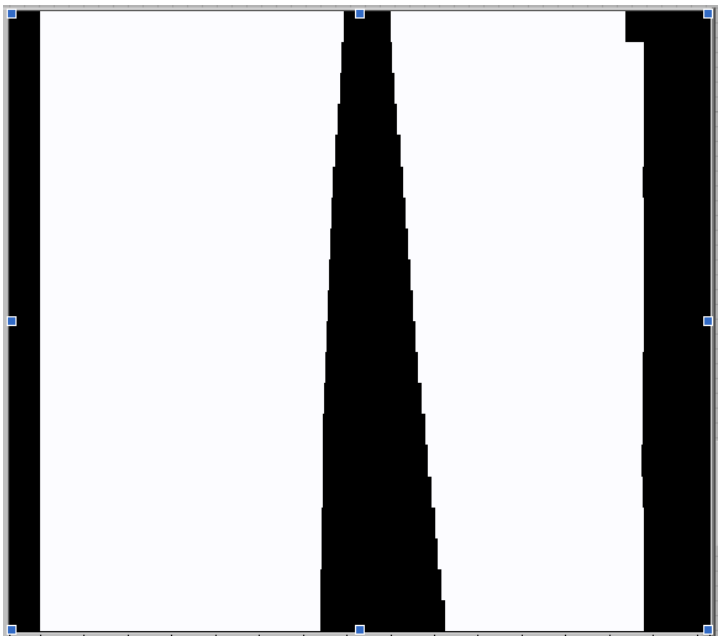


图 3.1　软件采集的实际赛道图像（直道）

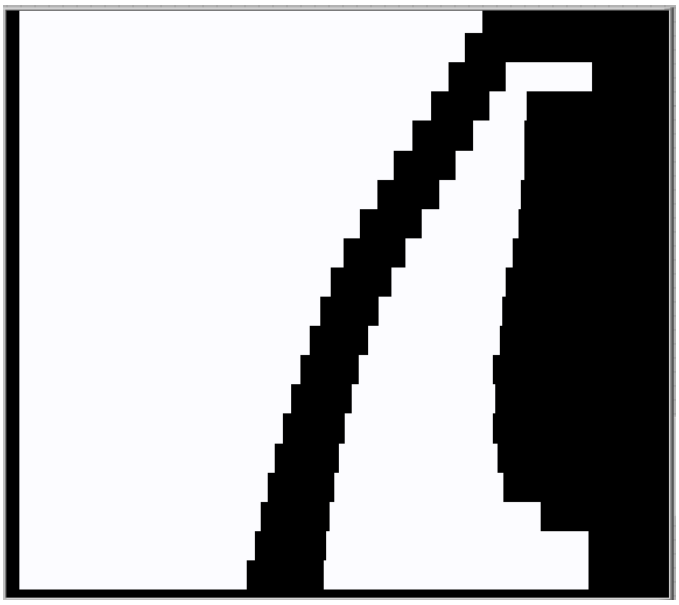


图 3.2　软件采集的实际赛道图像（弯道）

3.1.1.2　图像信号处理

在单片机采集出图像信号后需要对其进行处理以提取赛道信息，同时，由于杂点、交叉线、起始线的影响、光线、赛道连接处以及赛道以外图像的干扰，都会影响图像的效果，因此，软件上需要排除干扰因素，对有效赛道进行识别，并提供尽可能多的信息供决策使用。

图像信号处理中提取的赛道信息主要包括：

（1） 每一行的赛道中心位置

（2） 每一行的赛道宽度

（3） 赛道的曲率

（4） 赛道的变化幅度

（5） 赛道任一点的导数值

（6） 赛道形式，包括直道、左弯、右弯、大 S 弯、小 S 弯等等

由于摄像头倾斜放置导致的梯形畸变，以及无法避免的桶形畸变，这使得同一段赛道位于摄像头视域的不同区域时(近端、远端，边缘、中间)的表征会有所差别。为了还原出真实的赛道信息，我们根据摄像头的放置位置、角度等参数，对图像进行了梯形畸变和桶形畸变的校正，使得图像的处理更接近真实空间。

图像信号处理中实现鲁棒性的算法主要包括以下有效性的验证：

（1） 图像连续性判断

（2） 赛道一阶、二阶连续性判断

（3） 赛道宽度有效性验证

（4） 无效区域的判断

大 S 弯实际赛道和软件识别的赛道（包括单片机采集原始图像和处理后图像）如图 3.3 所示。

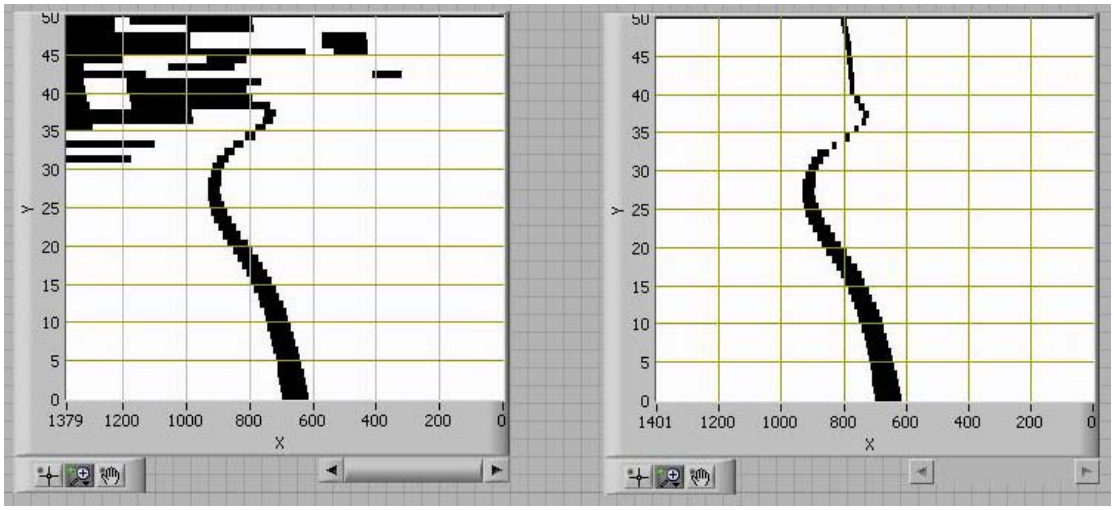


图 3.3　大 S 弯实际赛道和软件识别的赛道

图 3.4 显示了对赛道杂点、交叉线等因素干扰的处理。

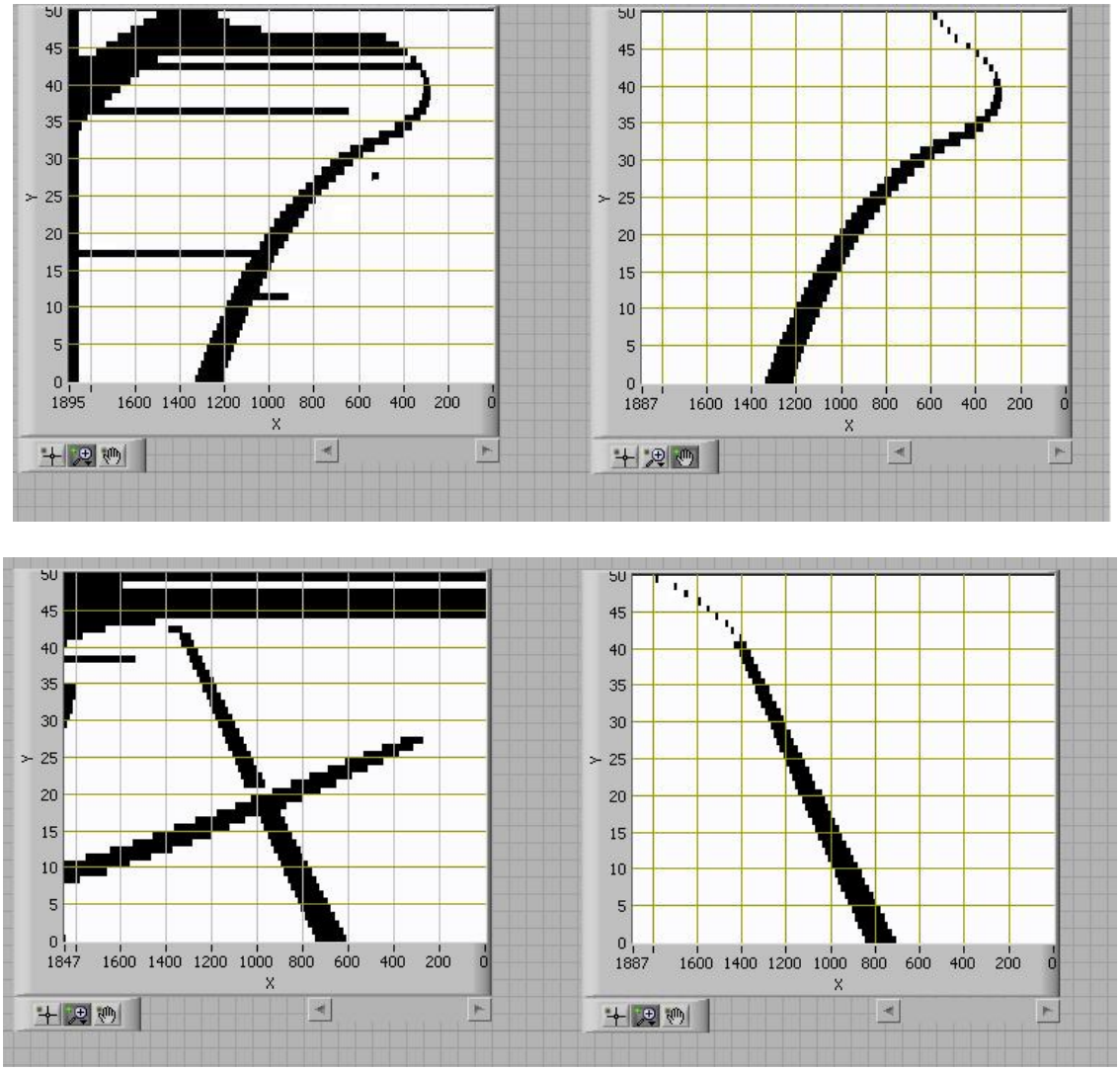


图 3.4　软件对赛道杂点、交叉线等因素干扰的处理效果

3.1.2　转向控制策略

3.1.2.1　转向 PID 调节

为了使舵机能更好的对给定的转角值做出响应，采用 PID 调节，通过不断改变 P、I、D 三个参数进行实验，得到最理想的转向响应速度的一组参数。选择合适的参数，可以使得在高速时车保持很高的稳定性，从而大大消除由于传感器带来的误差。

具体的调整方式大致为：

（1）P 环节与转向的准确性和快速性相关；

（2）I 环节与转向的准确性相关；

（3）D 环节与转向的快速型相关。

另外，由于被控对象（智能车）系统较为复杂，为非线性系统，因此单一的 PID 参数很可能难以满足实际控制效果需要，因此也可以考虑采用多 PID 参数或分段 PID 参数的方式来改善控制效果。

3.1.2.2　基于路径记忆的转向策略

此算法基于路径记忆算法。大致思想如下所示。

（1）在第一圈的记忆圈中，通过采集路径信息对赛道进行标识，提取各弯道的半径，长度以及弧度等信息；

（2）在第二圈行至相应路段时，根据第一圈预知的赛道信息采用最优的转角以及速度完成过弯。

采用如上方法后，就可以避免随动转向带来的一系列问题。

此法需要事先对赛车过各种弯道的性能进行标定试验，以确定赛车过各种弯道的最佳速度和转向角等信息。

3.1.3　车速控制策略

3.1.3.1　转速 PID 调节

由于硬件上加装了车速传感器，这样利用 HCS12 单片机上的脉冲捕捉端口 PT，通过计算由光码盘接受到的脉冲数转换得到当前车速，采用 PID 闭环控制，可以及时快速调节车速达到预定值。通过 PWM 波调制给驱动电机输入一定的占空比，使电机工作在一定转矩，由于车在赛道上行驶是，负载不断变化带来了电机工作时的波动，影响了车的实际速度，采用定速策略，可以就是根据实际中的不同负载状况及时快速调整 PWM 波，使车稳定于某一车速。图 3.5、3.6 是在调节过程中得到的中间曲线。

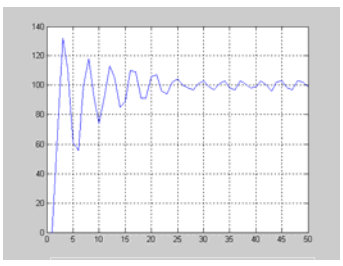


图 3.5 中间曲线 1

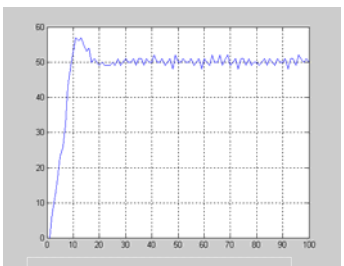
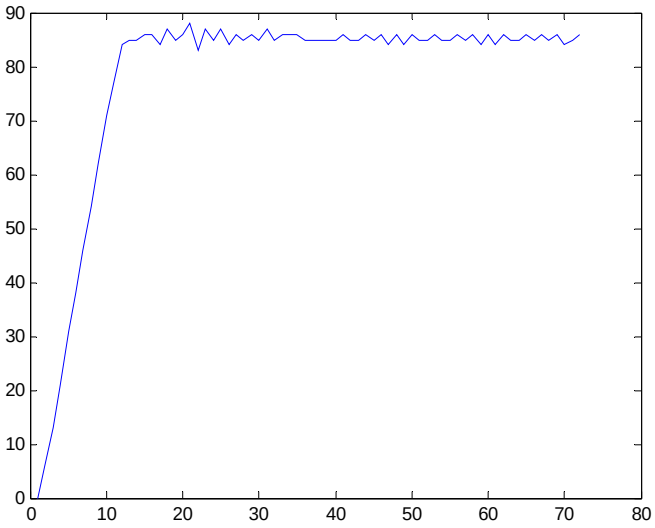


图 3.6 中间曲线 2

PID 调节最理想的状态就是能够让上升段快速升至目标点，并且不出现超

调，又能稳定在这一目标点，就如图 3.7 所示。



3.1.3.2　基于路径记忆的车速控制策略

大致与基于路径记忆的转向策略相同。为了尽量提高直道车速，同时保证在弯道上采用最优化的速度，可根据第一圈记忆圈对赛道信息进行采集，提取各赛段信息，从而实现对各赛段预先设定最理想的速度，达到最优速度控制效果。

3.1.4　道路记忆——M 算法 II

3.1.4.1　M 算法的实现

首先该算法要在保证赛车行驶两圈的前提下才可以实现，　赛车第一圈时通过记录转速传感器采集到的脉冲数来实现对未知道路信息的记忆，根据脉冲的正负、宽度、个数来判断区分直道、弯道、S 弯道以及转弯的方向与转弯半径等等信息，这样在进入第二圈后，就可以直接利用这些信息进行控制策略的制定。首圈实现的路径记忆效果图如图 3.8 所示。

3.1.4.2　基于 M 算法的转向及转速策略

根据第一圈记录的数据信息，对第二圈的各个道路点进行分段处理。直道上采用最高速加速，在进入弯道之前提前进行减速，减至过弯的极限最高车速，对于不同半径的弯道，选择不同的车速。M 算法的优势在于对于复杂的 S 弯道，它可以实现选用小的转向角度通过，这样可以大大缩短时间。

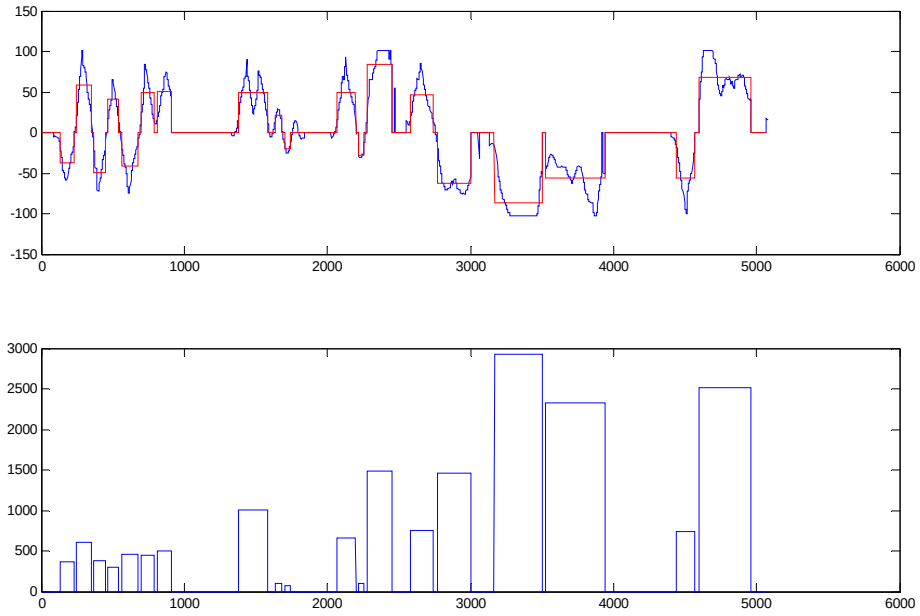


图 3.8 首圈实现的路径记忆效果图

3.2 开发工具及安装调试过程

软件开发工具为 Codewarrior3.1，调试软件为 Codewarrior4.1，调试器为清华大学工程物理系开发的 BDM。软件编写初期进行调试，只要让车在静态状态下，边采集传感器信息，边通过 hiwave.exe 程序的窗口中读取通过软件处理的数据，通过与理论值进行对比来判断程序的正误。

利用 BDM 可以即时的对软件和硬件进行监控，在调试过程中可以方便的设置断点、选用单步运行的方式，同时查看各个寄存器的值，EEPROM 中的数据掉电之后不会丢失，可以用来记录数据再进行后处理。这些强大的功能，都为我们的软件调试提供了极大的便利。

下面介绍我们具体的调试过程。

（1）在线调试：

如上面介绍的开发工具，使车处于静止状态，利用 BDM 将数据从单片机传入电脑，可以直接在线分析数据，对错误和故障进行诊断。这个调试方法适用于初期软件编写。

（3）离线调试：

在经过编译调试正确之后，就可以进行离线调试，由于传感器存在动静态的差异，所以离线调试主要是测量车在动态过程中采集到的信息并对其进行处

理。常用的手段是：

A．编程采集道路或者车速传感器的信息记录到 EEPROM 中，在单片机重起之后可以仍然可以利用串行通信端口将数据读出，再进行分析。

我们在调节转速 PID、进行转向阶越匹配标定等等过程的时候，就是利用这种离线测试的办法，通过不断的试验、分析数据曲线，最终确定一组最合适的参数。

B．不用 EEPROM 记录数据，直接通过观察来分析结果。在一些现象比较明显的调试阶段使用。

总体上，我们是可以充分利用了现有的调试工具和通信设备，得到了我们所需要的数据和结果，以便更好的为我们的决策进行优化。

（4）其他辅助调试工具：

在算法最初选定阶段，我们使用由我们自己开发的 Plastid 仿真工具进行预仿真，以观察在实际应用中的效果。MATLAB 目前在我们后期数据处理，算法验证及离线仿真的过程中起到了关键作用。

3.3 代码设计简介

3.3.1. 代码分类

整体的软件代码设计分为两种类型，一类是与硬件相关，作为驱动程序的代码，另一类是纯算法设计的代码，下面就这两种类型进行说明。

表 3.1 代码类型说明

原程序代码文件		实现功能
纯算法设计	course.c	编写道路记忆的相关函数
	PID.c	编写 PID 闭环调节函数（用于转速与转向）
与硬件相关的设计	ADConv.c	对 ATD 口进行初始参数设置，控制 AD 转换
	PAC.c	脉冲捕捉相关函数
	EEPROM.c	对于 EEPROM 的写入，传出及起始地址设置
	LEDTube.c	有关数码管显示设置
	motor.c	驱动电机相关参数初始化设置，输入 PWM
	operation.c	母板上按钮开关的与控制函数
	rudder.c	控制舵机的函数

	Timer.c	利用单片机时钟中断的函数
主函数	main.c	包括初始化、函数调用、中断等

3.3.2 接口标准

与硬件相关软件需要预先设定与硬件接口的标准，也就是参数的初始化设置，软件中就表现为各个寄存器标志位的不同设置，这里不再赘述。

第四章　赛车主要技术参数

表 4.1 模型车主要参数

项目	参数
车模几何尺寸（长、宽、高）(毫米)	290\170\290
车模轴距/轮距（毫米）	198\140
车模平均电流（匀速行驶）(毫安)	2A
电路电容总量（微法）	1900
传感器种类及个数	CCD 一个，光电一个
新增加伺服电机个数	0
赛道信息检测空间精度（毫米）	0.5
赛道信息检测频率（次/秒）	50
除 MC9S12DG128 之外其它主要芯片	LM1881 MC33883
车模重量（带有电池）(千克)	2

第五章　结论

从小前瞻单排离散传感器到脉冲发光大前瞻双排连续传感器，从元件海选到芯片订购，从 PID 调速到 M 算法，从软件模拟到实车调试……伴随着设计理念的进步和任务目标的明确，我们完成了智能车从里到外、从硬件到软件的全部设计制作。在设计制作过程中，我们采用了模块式的硬件安装方法，提高了维修检测的效率；创新了传感器的安装方式，增大了探测精度，技术的进步使我们的模型车在自制赛道上取得了令人满意的成绩。但我们的车依然有改进的余地，增大探测距离，提高车身定位精度，改进算法模糊寻迹切内弯，这些都是我们下一代智能车的改进方向。

技术进步永无止境！

参考文献

[1] 周斌，蒋荻南，黄开胜．基于虚拟仪器技术的智能车仿真系统．电子产品世界, 2006(3)

[2] 黄开胜，金华民，蒋荻南．韩国智能模型车技术方案分析．电子产品世界, 2006(5)

[3] 周斌，李立国，黄开胜．智能车光电传感器布局对路径识别的影响研究．电子产品世界, 2006(9)

[4] 陈宋，李立国，黄开胜．智能模型车底盘浅析．电子产品世界, 2006(11)

[5] 周斌，刘旺，林辛凡，等．智能车赛道记忆算法的研究．电子产品世界, 2006(15)

[6] 黄开胜，李立国，刘　旺，等．基于光电传感和路径记忆的车辆导航系统．电子产品世界, 2007(2)

[7] 林辛凡，刘旺，周斌，等．基于离散布置光电传感器的连续路径识别算法．电子产品世界, 2007(3)

[8] 清华大学 2 队技术报告．//卓晴，黄开胜，邵贝贝．学做智能车．北京: 北航出版社, 2007.3

附录

附录 A　　程序源代码

```
#include "stdinc.h"
#include "utility.h"
#include "EEPROM.h"
#include "ADConv.h"
#include "PAC.h"
#include "adapter_C_CPP.h"
#include "PID.h"
#include "Sci.h"
#include "rule_table.h"
#include "trash.h"
#include "operation.h"
#include "course.h"
#include "widget.h"

#define Pixel_Num 6
#define Line_Num 50
xuint16 Line_Sample_Interval=5;
xuint16 Start_Of_Frame=30;
xuint16 End_Of_Frame=0;

#define PULSE_CNT 750
//xuint16 speedPulse[PULSE_CNT];

xuint16 Frame_Data[Line_Num][Pixel_Num];
xuint8 Frame_OddEven=0;
xint16 Frame_Data_Processed[Line_Num][2];
//xuint8 f[6];
xuint16 frame_num=PULSE_CNT;
xint16 position_temp[PULSE_CNT];
//xint16 rudder_temp[PULSE_CNT];
xint16 curve_temp[PULSE_CNT];


xuint8 line_num=0,pixel_num=0;

void system_init(void);
void auto_ctrl(void);
void frame_data_process(void);
void route_opti(xint16);
```

```
void Set_Rudder(xint16 Position);
xint16 get_curve(void);
void Set_Speed(xuint16 curve);

//PID
PID g_speedPID;
PID g_rudderPID;
//course

extern void timer_interrupt(void);

void system_init(void);

  route_status Route_Status;
//--Program entry----------------------------------------------------------
void main(void)
{
  xuint16 delay;
  xuint8 i,j;



system_init();


wait_until_button_up(0);

Frame_OddEven=PORTA_BIT0;
while(Frame_OddEven==PORTA_BIT0);
Frame_OddEven=PORTA_BIT0;


EnableInterrupts;

//wait_until_button_up(1);

while(frame_num||1)
```

```
{

}



DisableInterrupts;
g_motor_set_status(FAST_STOP);
//g_motor_set_status(REVERSE);
g_motor_set_speed(255);


  wait_until_button_up(0);



while(1)
{
}


}
//--Function Implementation----------------------------------------------------

void actuate_speed_PID(xint16 speed)
{
	//xint16 disableThresh = 70;
	//xint16 highThresh = 25;
	//xint16 lowThresh = 5;
	xint16 PWMout = PIDCalc(&g_speedPID, speed);

	if (PWMout>0)
	{
		g_motor_set_status(FORWARD);
			g_motor_set_speed(PWMout);
	}
	else if (PWMout<=0) {
	  // g_motor_set_status(REVERSE);
	    if (PWMout>-50)
	    {
```

```
            g_motor_set_speed(-PWMout);
            g_motor_set_status(REVERSE);
          }
          else if(PWMout>-255)
          {
            g_motor_set_speed(-PWMout);
            g_motor_set_status(FAST_STOP);
          }
          else
          {
           g_motor_set_speed(255);
           g_motor_set_status(REVERSE);

        //    else g_motor_set_status(FREE_RUN);
          }
       }
       else
          g_motor_set_status(FAST_STOP);
}
//---------------------------------------------------------------------------




#define PWMCNT 7
#define ANGLECNT 6
xuint8 PWMSerial[PWMCNT] = {100, 130, 160, 190, 220, 255};
xint8 AngleSerial[ANGLECNT] = {10, 16, 22, 28, 34, 40};
xint8 PWMSelected = 0;
xint8 PWMIndex = 0;
xint8 AngleSelected = 0;
xint8 AngleIndex = 0;

xint8 stepResponseCnt = 0;



//--Timer interrupt service function-----------------------------------------
//VECTOR ADDRESS 0xFFCA timer_interrupt
#pragma TRAP_PROC
```

```
void timer_interrupt(void) {}
//------------------------------------------------------------------------------

void system_init(void)
{
   xuint16 delay;
   //PLL settings
   PLLCTL_ACQ = 1;
   PLLCTL_PLLON = 1;
   SYNR = 0x02;
  // SYNR=0x01;
   REFDV = 0x01;
   //wait for a given time
   for (delay=0;delay<0xffff;++delay){}
   CLKSEL_PLLSEL = 1;
   DDRT=0x00;
   //PACA
   init_PACA();
   set_PACA(0);
   init_PACB();
   //motor

  // timer_init();
   //Timer's settings
   //Sensor IO direction
   DDRA=0x00;
   DDRB=0xFF;
   //
   DDRH=0xFF;

   init_SCI(BAUD_9600);



  // PIDInit(&g_speedPID, 2.5, 0.1, 0, 255, 0);
       PIDInit(&g_speedPID, 30, 0.5, 0, 255, 0);
       //     PIDInit(&g_speedPID,5, 0.1, 0, 255, 0);
```

```
   g_speedPID.SetPoint = 50;                                    //35


}
//------------------------------------------------------------------------------

void Set_Rudder(xint16 Position);
xint16 get_curve(void);
void Set_Speed(xuint16 curve);
xint16 Get_position(void);

void Set_Param(void);
xuint8 Get_Line_Selected(xuint16);

xuint8 Line_Selected = 25;              //32 = 40cm

#define ROUTE_CENTER 670
xint16 route_center = ROUTE_CENTER ;        // for adj

//#define Predict_Line_Num 6
#define Predict_Line_Begin        5               //5
#define Predict_Line_End           39             //40

xuint8 Predict_invalid_cnt = 0;



void auto_ctrl(void)
{
  volatile xint16 position=0;
  xuint8 i=0;
  static xuint8 cnt = 0;
  xuint8 speed = 0;

  volatile xint16 curve=0;



  curve = get_curve();
  curve_temp[PULSE_CNT-frame_num]=curve;
// curve_tmp = curve;
```

```
  // g_led_dec(abs(position)/10);

  g_led_dec(abs(curve)/20);

//    g_led_dec(Line_Selected);
// route_opti(curve);

  //g_rudder_set_status(Route_Status);

//        g_rudder_set_angle(30);
// Set_Param();
// g_rudder_set_status(Route_Status);


// g_speedPID.SetPoint=55;
// Line_Selected = 27;

// g_led_dec(g_rudder_get_d()/10);

// g_led_dec(Line_Selected);
// g_led_dec(Route_Status);
  Set_Speed(curve);
/*
      cnt++;
      if(cnt==10)
      {
         g_led_dec(abs(curve)/10);
         frame_num--;
         position_temp[50-frame_num]=position;
         rudder_temp[50-frame_num]=g_rudder_set_angle_p(position);

         cnt = 0;
      }

*/
}


#define Speed_Straight 90
```

```
#define Speed_Curve 60
#define Speed_S_Curve 65

#define Position_Dangerous 400
//#define Position_Devia_Trend

void Set_Param(void)
{
   static xint16 last_position=0;
   xint16 position_trend=0;
   xint16 position;
   xint16 angle;

   switch (Route_Status)
   {
      case STRAIGHT:
            if(g_speedPID.SetPoint<Speed_Straight)
            g_speedPID.SetPoint+=10;
            Line_Selected = Get_Line_Selected(get_PACA());
          //     dead1 = 10, dead2 =150, slope1 = 8, slope2 = 5, saturation = 100;
          break;

      case CURVE_L:
            g_speedPID.SetPoint= Speed_Curve;
            Line_Selected = Get_Line_Selected(g_speedPID.SetPoint);
          break;

      case CURVE_R:
                g_speedPID.SetPoint= Speed_Curve;
                Line_Selected = Get_Line_Selected(g_speedPID.SetPoint);
          break;

      case S_CURVE:
                g_speedPID.SetPoint= Speed_S_Curve;
                Line_Selected = 32;//Get_Line_Selected(g_speedPID.SetPoint);

          break;


      case DANGEROUS:
```

```
            break;
}
position=Get_position();
angle=g_rudder_get_angle();

if(abs(position)>Position_Dangerous)
{
position_trend=position-last_position;
if((position>0&&position_trend>0)||(position<0&&position_trend<0))
g_speedPID.SetPoint-=0;
}
last_position=position;

/*
if(abs(angle)>=100)
{
position_trend=position-last_position;
if((angle>0&&position_trend>0)||(angle<0&&position_trend<0))
g_speedPID.SetPoint-=5;
}
*/

last_position=position;

}

#define Speed_Norm 55
#define Line_Selected_Norm 27
#define Line_Selected_Max 30


xuint8 Get_Line_Selected(xuint16 speed)
{
xuint16 line=0;
line=speed*Line_Selected_Norm/Speed_Norm;

if(line>Line_Selected_Max)
    line=Line_Selected_Max;
```

```
return (xuint8)(line);
}


void Set_Rudder(xint16 Position)
{


}


//====================== realtime control area ======================
#define Line_Width_Norm 627       //30cm line19
#define Position_Comp_Slope 18
#define Position_Comp_Intercept 968

xint16 Get_position(void)
{
   long int pos,pos_ori;
   static xint16 pos_last = 4000;
   #define DIFF_VAL 800
   xint16 line_width=0;

   pos_ori = Frame_Data_Processed[Line_Selected][0];

   //根据前瞻距离补偿 pos
      line_width = - Line_Selected*Position_Comp_Slope + Position_Comp_Intercept;
      pos = pos_ori * line_width / Line_Width_Norm;


   //跳变处理，跳变过大则滤去
   if (abs(pos - pos_last) < DIFF_VAL || pos_last == 4000)
      pos_last = pos;
   else
      pos = pos_last;

   return (xint16)(pos);

}
```

```
//====================== speed estimation ============================

xint16 get_curve(void)
{
static xuint8 idx=0;
/*
xint16 average=0;
xuint8 i=0;
xuint16 curve=0;
  for(i=0;i<Predict_Line_Num;i++)
  average+=Frame_Data_Processed[Line_Selected-i-1][0];

  for(i=0;i<Predict_Line_Num;i++)
  curve+=abs(Frame_Data_Processed[Line_Selected-i-1][0]-average);

  return curve;

  */


//   斜率的方差 作为 曲率的估计
//   Var[x]=E[x^2]-[E(x)]^2

    volatile xint16 curve = 0;
    long int curve_part1 = 0;
    long int curve_part2 = 0;
    xuint8 i, num = Predict_Line_End - Predict_Line_Begin;

              /*
    if (Predict_invalid_cnt > 4)
    {
       Predict_invalid_cnt=0;
       return(0x0FF);
    }
              */



    for (i = Predict_Line_Begin + 1 ; i < Predict_Line_End+1 ; i++)
```

```
    {
        curve_part1 += (Frame_Data_Processed[i][0] - Frame_Data_Processed[i-1][0]) *
(Frame_Data_Processed[i][0] - Frame_Data_Processed[i-1][0]) ;
        curve_part2 += (Frame_Data_Processed[i][0] - Frame_Data_Processed[i-1][0]) ;
    }

    curve = ( curve_part1 * num  - curve_part2 * curve_part2) / (num*num) ;

    return(curve);
}



#define PID_OR_NOT 0        // 0: PWM CONTROL      0 default
                            // 1: PID CONTROL


void Set_Speed(xuint16 curve)
{
volatile xuint16 speed=0;
static xuint8 cnt=0;
speed=get_PACA();
//speedPulse[PULSE_CNT-frame_num]=speed;
frame_num--;
set_PACA(0);



/*
    if (curve < 100)
        speed = 100;
    else if (curve < 200)
        speed = 60;
    else if (curve < 300)
        speed = 40;
*/



cnt++;
```

```
if(cnt<20)
{
//g_led_dec(speed/2);
cnt=0;
}

#if    PID_OR_NOT
  // if(frame_num<200)
    actuate_speed_PID(speed);
  // else
  // g_motor_set_status(FAST_STOP);

#else

    g_motor_set_status(FORWARD);
    g_motor_set_speed(10);

#endif


}



//============================   数据处理   每行处理一次 ================
void frame_data_process(void)
{
xuint8 p=0;
xint16 delta[Pixel_Num], pos = - route_center;
static xint16 pos_last = 0;
xuint8 j=0;
static xuint8 begin_valid = 0;   //起始若干行有效性判断
static xint16 begin_pos_last = 0;
static xint16 slope_last = 0     ,   slope = 0;   //斜率
static xuint8 slope_invalid = 2;            //斜率有效性，0 时为有效
static xint16 width_last;

//==========s-curve identification use============

static xuint8 s_cnt = 0 ;
```

```
#define S_CNT_VALVE 10

static xuint8 s_iden1 = 0 , s_iden2 = 0;
static xint8 s_slope_now , s_slope_last;
static xint8 s_slope1;              // slope of the first part of S-Curve


//================================================


  volatile xint16 position=0;




/*
if(line_num==7)
PORTB_BIT0=1;
   */
if (line_num == 0)
{
   begin_valid = 0;
   slope_invalid = 2;
   slope_last = 0;
}



   for(p=0;p < Pixel_Num-1;p++)
   {

      if ( (Frame_Data[line_num][p+1] == 0) && (begin_valid == 0) )       //起始无效行检测
      {
         begin_valid = 0;
         break;
      }

      if(Frame_Data[line_num][p]<Frame_Data[line_num][p+1])   //计算脉冲宽度
         delta[p]=Frame_Data[line_num][p+1]-Frame_Data[line_num][p];
      else
```

```
    delta[p]=65536-Frame_Data[line_num][p]+Frame_Data[line_num][p+1];



slope = pos - pos_last + delta[p]/2;

if ( p >= 1                         &&
          (delta[p] > 20) && (delta[p] < 200)      &&
       //线宽检验
        ((abs(slope) < 100) || !line_num || !begin_valid ) && //位置跳变检验
        ((abs(delta[p] - width_last) < 30) || !line_num || !begin_valid    )  &&
//宽度不能跳变        30
        (( abs(pos+delta[p]/2 - begin_pos_last) < 200 ) || begin_valid ||1)
 //起始端位置不能跳变
     )
{
   if((abs(slope-slope_last) < 30) ||    (slope_invalid) )    //斜率不能调变                    50
   {

      Frame_Data_Processed[line_num][0] = pos + delta[p]/2 ;
      Frame_Data_Processed[line_num][1] = delta[p];


      if(begin_valid == 0)
         begin_pos_last =    Frame_Data_Processed[line_num][0];

      begin_valid = 1;

      if(slope_invalid >0)
            slope_invalid-- ;

         slope_last = slope ;

               break;
   }

}
   else if (p == (Pixel_Num - 2) && (line_num > 0))                                        //
无效处理
   {
```

```
            Frame_Data_Processed[line_num][0]    =    Frame_Data_Processed[line_num-1][0]    +
sign(slope_last) * min(20,abs(slope_last));                    //20
          Frame_Data_Processed[line_num][1] = Frame_Data_Processed[line_num-1][1];


      //    if (line_num>Predict_Line_Begin && line_num<Predict_Line_End)
             Predict_invalid_cnt++;

      }


      pos += delta[p];
   }
 /*
       */


       //饱和区
         if (       Frame_Data_Processed[line_num][0] > 700)
                    Frame_Data_Processed[line_num][0] = 700;
         if (       Frame_Data_Processed[line_num][0] < -700)
                    Frame_Data_Processed[line_num][0] = -700;



         pos_last = Frame_Data_Processed[line_num][0];
         width_last = Frame_Data_Processed[line_num][1];


   // clear


   for(j=4;j<Pixel_Num;j++)
   {
   Frame_Data[line_num][j]=0;
   }



      //======================    rudder control ========================
```

```
      if(line_num == 30)
      {

         position = Get_position();         //    Frame_Data_Processed[Line_Selected][0];
               position_temp[PULSE_CNT-frame_num]=position;
               g_rudder_set_angle_p(position);
           //    g_led_dec(abs(position)/10);

      }



    //======================                    s-curve                identification
==========================

    if(line_num == 0)
    {
       s_slope_last = 0;

       s_cnt = 0;
       s_iden1 = 0;
       s_iden2 = 0;
       Route_Status = STRAIGHT;

    }

    else
    {
       s_slope_now              =              (Frame_Data_Processed[line_num][0]              -
Frame_Data_Processed[line_num-1][0]);

       if (abs(s_slope_now)<5)
          s_slope_now = 0;
       else
          s_slope_now = sign(s_slope_now);



    if( s_slope_now != 0)
```

```
    {

       if (s_slope_now == s_slope_last)
            s_cnt++;
       else
            s_cnt = 0;




       if ( s_cnt > S_CNT_VALVE )
       {
          if(s_iden1 == 0)
          {
           s_iden1 = 1;
           s_slope1 = s_slope_now;

          }
          else if (s_slope_now != s_slope1)
               s_iden2 = 1;
       }


          s_slope_last = s_slope_now;

    }


     if (s_iden2 == 1)
             Route_Status = S_CURVE;



    }

//=================================================================
=======
```

```
}

//=========================   route optimization ====================


#define Turn_Curve_H 1       //定义的认为前方为弯道的 Curve 值
#define Turn_Curve_L 1
#define Route_Delta_Step 20       //入弯过程每次调整的偏移量
#define Route_Delta_Max 500           //最大偏移量，需标定与 10cm 左右对应
//#define S_Turn_Curve 50
//#define S_Curve_Bound 300
#define Slope_Delta 50
#define S_Curve_ID_Num 5

void route_opti(xint16 curve)
{
   static route_status Route_Status_Last=STRAIGHT;
//static xuint8 turn_flag=0;
//static xuint8 turn_exit_flag=0;
static xint16 route_delta=0;
   xuint8 i=0;
   xuint8 slope_increase_num=0;
   xuint8 slope_decrease_num=0;
   static xuint8 straight_cnt=0;
  // status identification


   /*
   for(i=Predict_Line_Begin+1;i<Predict_Line_End;i++)
   {
   if(Frame_Data_Processed[i][2]>Slope_Delta)
   slope_increase_num++;
   else if(Frame_Data_Processed[i][2]<-Slope_Delta)
   slope_decrease_num--;
   }
   if(slope_increase_num>S_Curve_ID_Num&&slope_decrase_num>S_Curve_ID_Num)
```

```
    Route_Status = S_CURVE;
    */
    if(Route_Status==S_CURVE)
    {
        if(curve<=Turn_Curve_L || curve > 140)
            Route_Status==STRAIGHT;
    }

    if(Route_Status!=S_CURVE)
    {
    if(curve>Turn_Curve_H)
    {
    curve *= sign(Frame_Data_Processed[35][0]-Frame_Data_Processed[5][0]) ;                    //
给 curve 符号
    if(curve>0)
    Route_Status = CURVE_L;
    else
    Route_Status = CURVE_R;
    }
    else
    {
     if(Route_Status_Last==CURVE_L||Route_Status_Last==CURVE_R)
     {
      if(curve<Turn_Curve_L)
      straight_cnt++;
      if(straight_cnt>=3)
      {
      Route_Status=STRAIGHT;
      straight_cnt=0;
      }
     }
     else
     {
     Route_Status = STRAIGHT;
     straight_cnt=0;
     }
    }
    }
    Route_Status_Last=Route_Status;
}
```