第十一届“恩智浦”杯全国大学生

智能车竞赛

技术报告

学　　校：天津大学

队伍名称：天津大学摄像头二队

参赛队员：殷璞芙

张经伟

张进

带队教师：王建荣

关于技术报告和研究论文使用授权的说明

本人完全了解第十一届“恩智浦”杯全国大学生智能汽车邀请赛有关保留、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：张经伟 张进 殷璞芙

带队教师签名：

日　　期：2016.8.04

摘要

本文以第十一届恩智浦杯全国大学生智能车竞赛为背景，介绍了智能赛车控制系统的软硬件结构及开发流程。我们采用大赛统一规定的 B 型车模，以 Freescale 半导体公司生产的 MK60DN512VLL10Q 作为核心控制器，通过 ov5116 摄像头采集赛道信息并控制赛车运动轨迹。在赛车行驶过程中，用 PD 方式对舵机进行控制，通过编码器获取赛车当前运行速度，并采用 PID 控制实现速度闭环控制。在赛车调试过程中，不断优化程序算法，改进机械结构，调整电路设计使其尽可能以最快速度沿赛道行驶。

关键词：Freescale，智能车，摄像头，PID

目录

第一章　绪论

1.1 引言

恩智浦智能车大赛原名飞思卡尔智能车大赛，创办于 2006 年，如今已走过 11 个年头。自从第三届出现摄像头组以来，摄像头车的速度已经有了飞跃性的提升。经过八年的探索与磨练，赛车的控制算法、电路结构已大致统一，如何最大限度提升速度，需要队员在细节处花费更多的心思。

在制作小车的过程中，我们对小车的整体构架进行了深入的研究，分别在机械结构、硬件和软件上都进行过改进，硬件上主要是考虑并实践各种传感器的布局，改进驱动电路；软件上进行了许多探索，在控制算法上，从 PID 到 Bang-Bang，再到模糊 PID 都进行了一些研究。

本文将重点介绍本队赛车制作流程，包括程序设计、机械结构改造、电路设计以及赛车调试过程中遇到的问题及解决办法。在整个过程中，我们在许多方面遇到了较为棘手的问题，但我们依然克服困难解决了这些问题。可以说，这辆赛车是我们全队共同努力的结果。

在此感谢天津大学为我们提供的实验室以及试车场地，帮助我们有充足的时间和便利调试赛车；感谢指导教师对我们的大力支持；还要感谢车队领队在本次大赛中对我们的帮助，在他的指导下我们避免了许多弯路，同时开拓了思路。

1.2 系统设计框架介绍

系统以检测赛道边沿信息为基础， 通过单片机实现信号处理以及车体控制，使车体严格按照赛道路径循迹。电路部分主要包括单片机控制单元、电机驱动单元以及摄像头检测单元等部分，除此之外，同样需要一些外部电路如陀螺仪模块、编码器测速模块、伺服器转向控制模块以及电机等。赛车整体架构如图 1.1：

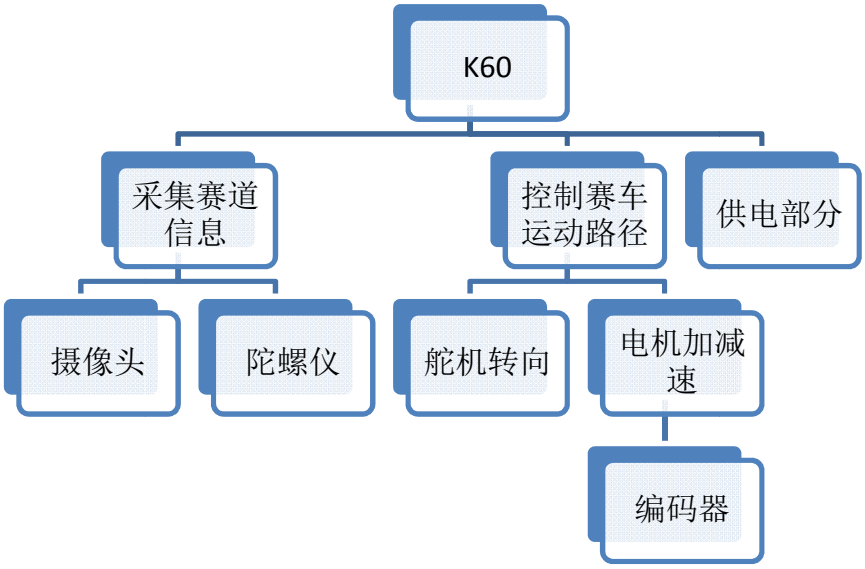


图 1.1：赛车整体架构

综上所述，本赛车系统包含的模块如下：

K60 控制模块：该部分是整个赛车的大脑与核心，赛车的所有行为都要基于单片机的决定。它汇总摄像头、陀螺仪与编码器的所有信息并作出评判，将命令发送给电机与伺服器等模块。

摄像头模块与信号处理模块是赛车的眼睛，它检测赛道边沿以及障碍物从而分辨出赛道状况，同时可以设置一定前瞻使摄像头有预见性，为单片机决策提供充足的反应时间。而陀螺仪、编码器是赛车的眼镜，帮助赛车检测到摄像头不易察觉的坡道并感知速度。

电源模块负责电路部分的供电，包括摄像头、电机驱动、单片机与众多芯片，是赛车的心脏。

电机、电机驱动部分、舵机等共同组成了赛车的腿，保证赛车以预设的速度及方向行驶。

辅助调试模块类似于赛车的跑鞋，主要用于智能车系统的功能调试、赛车状态监控等方面。

第二章 智能车机械结构调整与优化

2.1 智能车参数要求

1.车模使用 B 车模。该车模采用后轮电机驱动，电机型号 540。前轮由 SD-5 舵机控制转向。伺服电机的数量不得超过 3 个。

2.传感器允许使用面阵 CCD 或者 CMOS 摄像头以及超声传感器进行赛道检测。禁止使用激光发射管；禁止使用线阵 CCD 摄像头。传感器数量不能超过 16 个。

2.2 智能车整体参数调校

智能车的整体参数，包括车体重心、舵机电机放置的位置和高度、摄像头安装角度与高度等，这些都对整个智能车系统的稳定运行起着至关重要的作用。因此，对智能车机械系统的调节，有助于赛车更快更稳定的运行。赛车的布局以可靠、 稳定、精简为前提，通过对赛车的布局，尽量保证赛车左右平衡，并且寻找一个合适的重心，保证赛车既能够可靠地抓牢地面，又能够对前轮舵机，后轮电机有较快的响应。我们的赛车整体布局如图 2.1：

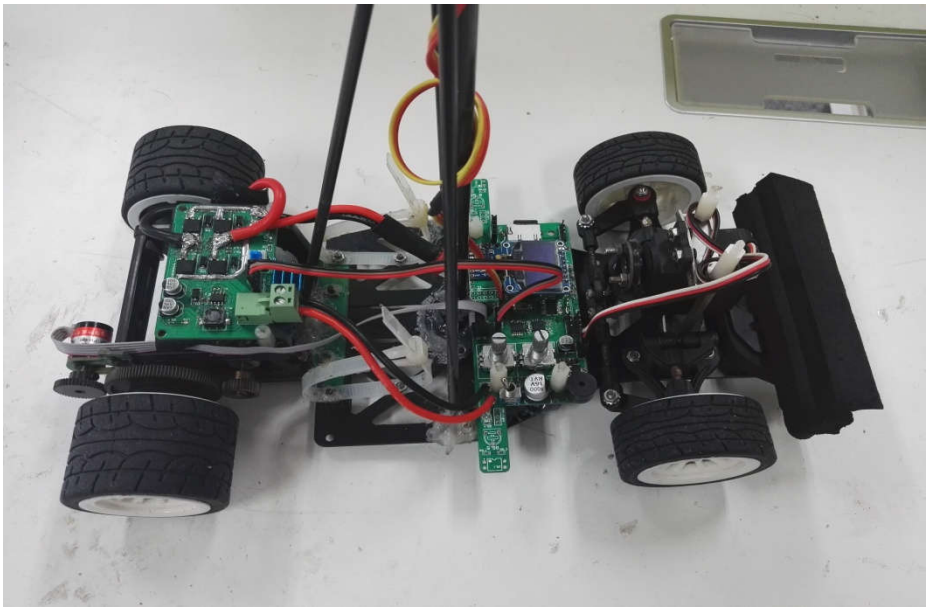


图 2.1：赛车整体布局

2.3 智能车前轮定位调整

2.3.1 主销后倾角

在汽车纵向平面内，主销轴线上端略向后倾斜，这种现象称为主销后倾。在

纵向垂直平面内，主销轴线与垂线之间的夹角叫主销后倾角，从汽车的侧面看去，主销轴线与通过前轮中心的垂线之间形成一个夹角，即主销后倾角。其值大小对汽车转向与操纵性能密切相关。

由于主销后倾，主销（即转向轴线）与地面的交点位于车轮接地点的前面。这时，车轮所受到的阻力的作用点总是在主销轴线之后，相当于主销拖着车轮前进。这样，就能保持行驶方向的稳定性。当汽车转弯时，由于车轮所受阻力作用线，不通过主销轴线，这样，车轮所受阻力在主销方向有力矩作用产生，迫使车轮自动偏转直到到车轮所受阻力作用线通过主销轴线，此时，车轮已回正，这就是转向车轮的自动回正功能。

主销后倾角越大，方向稳定性越好，自动回正作用也越强，但转向越沉重。汽车主销后倾角一般不超过 3°，由前悬架在车架上的安装位置来保证。现代轿车由于采用低压宽幅子午线轮胎，高速行驶时轮胎的变形加大，接地点后移，因此主销后倾角可以减小，甚至为负值（变成主销前倾），以避免由于回正力矩过大而造成前轮摆振。

在调试过程中，增大主销后倾角的确提高了智能车的灵敏度，转弯性能更好，最终我们将它放在了 2 到 3 度的位置。

2.3.2 主销内倾角

汽车转向节主销轴线(或独立悬架的上摆臂球销与下摆臂球销中心的连接线)与铅垂线在垂直于车辆纵向对称平面的平面上的投影锐角叫主销内倾角。具体的说，从车前后方向看轮胎时，主销轴向车身内侧倾斜，该角度称为主销内倾角。当车轮以主销为中心回转时，车轮的最低点将陷入路面以下，但实际上车轮下边缘不可能陷入路面以下，而是将转向车轮连同整个汽车前部向上抬起一个相应的高度，这样汽车本身的重力有使转向车轮回复到原来中间位置的效应，因而方向盘复位容易。

此外，主销内倾角还使得主销轴线与路面交点到车轮中心平面与地面交线的距离减小，从而减小转向时驾驶员加在方向盘上的力，使转向操纵轻便，同时也可减少从转向轮传到方向盘上的冲击力。但主销内倾角也不宜过大，否则加速了轮胎的磨损。

主销内倾和主销后倾都有使汽车转向自动回正，保持直线行驶的功能。不同之处是主销内倾的回正与车速无关，主销后倾的回正与车速有关，因此高速时主销后倾的回正作用大，低速时主销内倾的回正作用大。

尽管主销内倾增大了轮胎与地面间的摩擦力，但这导致赛车过弯时易发生侧滑。最终我们没有调整主销内倾角，使其保持平衡。

2.3.3 车轮外倾角

车轮外倾角是指车轮在安装后，其端面向外倾斜，即车轮所处平面和纵向垂直平面间的夹角。轮胎呈现"八"字形张开时称为负外倾，而呈现"V"字形张开时称正外倾。其作用是为了提高车轮工作时的安全性。由于主销与衬套之间、轮毂和轴承等处都存在着装配间隙，这些间隙在不同程度上影响着车轮正常工作。当车轮有一定外倾角时，轮胎的中心线与地面相交于 A 点，转向主销轴线与路面相交于 B 点。这两点之间的距离称为偏移。由于车轮在转向时，是绕主销轴线的中心和半径转动的，因此在转向主销周围因受到轮胎滚动阻力的作用，会产生一个大力矩，而增加转向力。偏移越大，产生的力矩也愈大。当具有外倾角时，可使偏移量减小，所以能减少转向力。此外，当车轮外倾时，在垂直载荷的作用下会产生一施加于芯轴上的分力，使车轮向内压在轴承上，以防止车轮甩脱。

当车轮外倾角不符合要求时，其产生的不良后果包括:球铰和车轮轴承的磨损加剧(负外倾角内轴承磨损加剧;正外倾角外轴承磨损加剧)。当外倾角过大，或左右不等时，还将导致车辆向正外倾角较大的一边跑偏。

2.3.4 前轮前束

前轮前束，是使汽车两前轮的前端距离小于后端距离。其距离之差叫做前束值。从汽车的上面往下看，左右两个前轮形成一个开口向后的"八"字形。采用这种结构目的是修正上述前轮外倾角引起的车轮向外侧转动。如前所述，由于有外倾，方向盘操作变得容易。另一方面，由于车轮倾斜，左右前轮分别向外侧转动，为了修正这个问题，如果左右两轮带有向内的角度，则正负为零，左右两轮可保持直线行进，减少轮胎磨损。

前轮外倾有使前轮向外转向的趋势，前轮前束有使车轮向内转向的趋势，可以抵消因前轮外倾带来的不利影响，使车轮直线滚动而无横向滑拖的现象，减少轮胎磨损。

2.4 舵机安装结构的调整

考虑到主板的安装方便以及车模转向性能，我们对舵机安装结构进行了较大的调整。比赛车模的转向是通过舵机带动左右横拉杆实现。舵机的转动速度和功率是一定的，要想加快转向机构的响应速度，唯一的办法就是优化舵机的安装位置及其力矩延长杆的长度。由于功率是速度与力矩乘积的函数，过分追求速度，必然要损失力矩，力矩太小也会造成转向迟钝，因此设计时就要综合考虑转向机构响应速度与舵机力矩之间的关系，通过优化得到一个最佳的转向效果。利用实际参数经计算，我们得出了一套可以稳定高效工作的参数及结构。

最终，我们设计了一套舵机连片(转向拉杆) ，综合考虑了速度与力矩的关系，并根据模型车底盘的具体结构，简化了安装方式，实现了预期目标。

我们的舵机安装如图 2.2 所示，在安装过程中必须保证舵机连杆可以达到预设的最大转角。与连片的啮合类似齿轮的啮合，由于连片长度以及齿宽的影响，

很难调节到合适的舵机机械中值。为了弥补这种连片的缺陷，我们采用了可调中值的舵机连片。在调试过程中，通过调整参数可以使舵机左右对称。

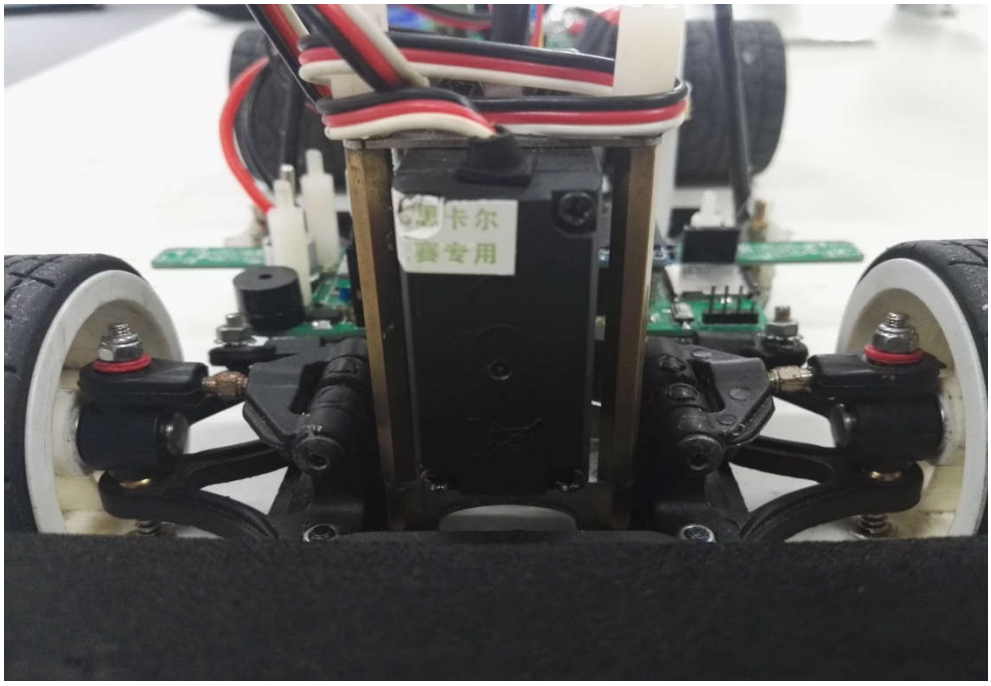


图 2.2：舵机安装图

2.5 编码器的安装

赛车选用编码器进行速度的测量。B 车模只有一个电机，实现起来相对简单，根据编码器的形状，选择合适的支架，将编码器用螺钉通过支架固定在后轮支架上，这样固定好之后，就有了较高的稳定性。然后调节编码器齿轮，使其与电机输出齿轮紧密咬合，增大测速的精确性。但是齿轮咬合过紧也增大了摩擦，减小了对电机做功的利用率，影响小车的快速行驶，因此减小摩擦同时增强齿轮间的咬合是我们主要考虑的因素。编码器安装如图 2.3：

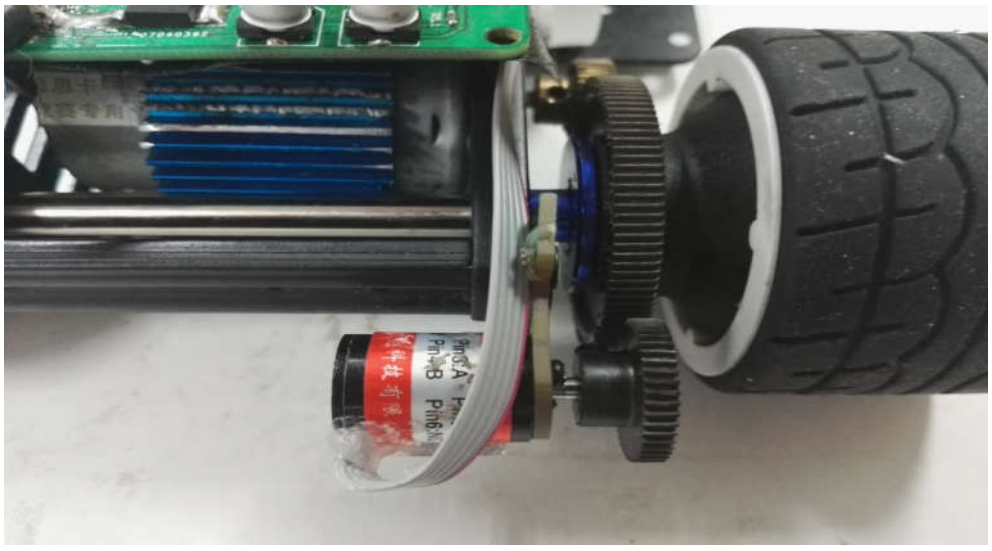


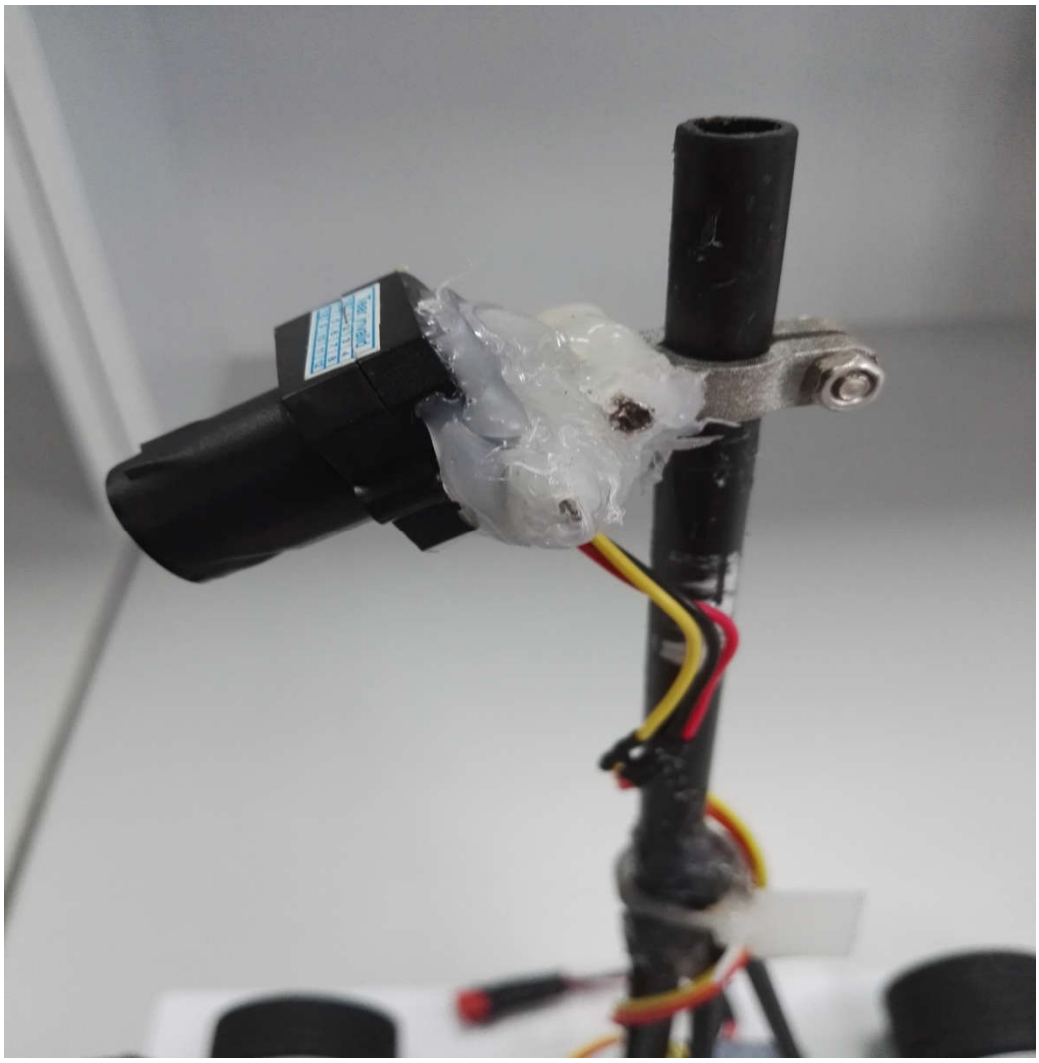
图 2.3：编码器安装图

2.6 底盘高度与赛车重心的调整

在保证顺利通过坡道的前提下，底盘尽量降低，从整体上降低模型车的重心，可使模型车转弯时更加稳定、高速。重心应尽量处于赛车中心的位置，才能全程保持平衡状态。

2.7 摄像头的安装

为了降低整车重心，需要严格控制 CMOS 摄像头的安装位置和重量，我们选择了轻巧的铝合金夹持组件并采用了碳纤维管作为安装 CMOS 的主桅，这样可以获得最大的刚度质量比，整套装置具有很高的定位精度和刚度，使摄像头便于拆卸和维修，具有赛场快速保障能力。为了保证镜头不受外部影响而松动 ，我们决定将其与摄像头芯片固定在一起。



第三章 电路设计说明

硬件电路设计应以可靠、稳定为基本原则，同时力求功能完备。可靠性与稳定性是一个系统能够完成预设功能的最大前提，为防止出现电压过载，我们将使用频率较高的5V电压分出两个部分以分担压力，并尽可能减少电路之间的干扰；在信号的采集与处理方面，我们在主控板上单独添加了硬件二值化电路与解码电路，编码器解码连续使用两个芯片，力求得到信息的准确。

本文将分别从三个方面介绍我们的电路组成，即主控板结构、驱动板结构与外部电路及传感器的选取。

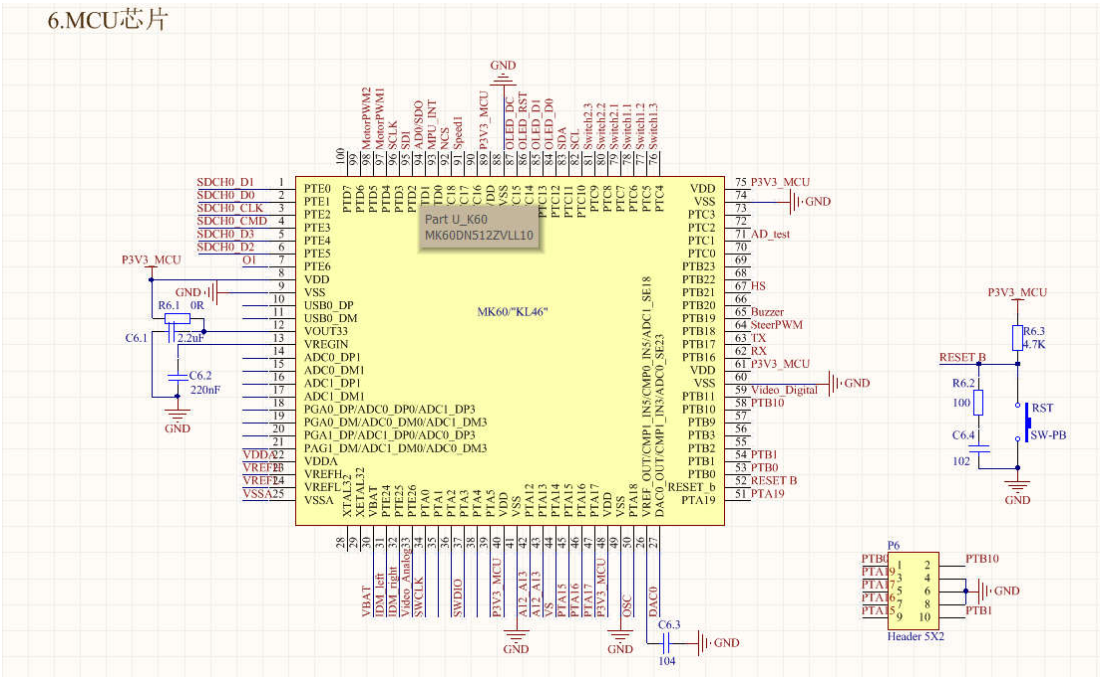
3.1 主控板电路设计

首先，本系统所采用的以 MK60DN512VLL10Q 为核心的单片机是基于 ARM® CortexTM-M4 具有超强可扩展性的低功耗、混合信号微控制器。第一阶段产品由五个微控制器系列组成，包含超过两百种器件，在引脚、外设和软件上可兼容。每个系列提供了不同的性能，存储器和外设特性。通过通用外设、存储器映射和封装的一致性来实现系列内和各系列间的便捷移植。

Kinetis 微控制器基于飞思卡尔创新的 90 纳米薄膜存储器（TFS）闪存技术，具有独特的 Flex 存储器（可配置的内嵌 EEPROM）。Kinetis 微控制器系列融合了最新的低功耗革新技术，具有高性能、高精度的混合信号能力，宽广的互连性，人机接口和安全外设。飞思卡尔公司以及其他大量的 ARM 第三方应用商提供对 Kinetis 微控制器的应用支持。其基本特性为：

- 电压范围　1.71V - 3.6V
- 温度范围　(TA) -40 to 105 °C
- 32 位 ARM Cortex-M4 内核
- 支持 DSP 指令
- 嵌套向量中断控制器　(NVIC)
- 异步唤醒中断控制器　(AWIC)

为节约主控板造价并节省空间，我们决定将 K60 最小系统板集成到主控板上，如图 3.1.1(A)所示：



下载器与晶振模块如图 3.1.1(B)：

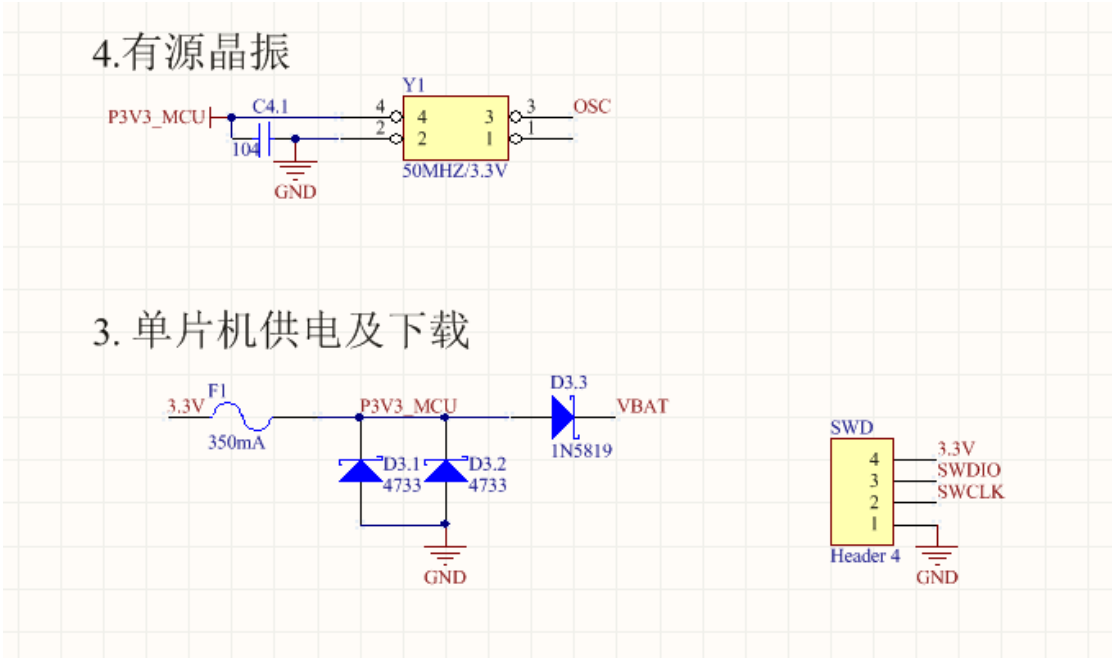


图 3.1.1 单片机最小系统板原理图

K60 最小系统板一般留有 OLED 显示屏接口与检测接口状态的 LED 灯，我们也将这一部分集成到了主控板上：

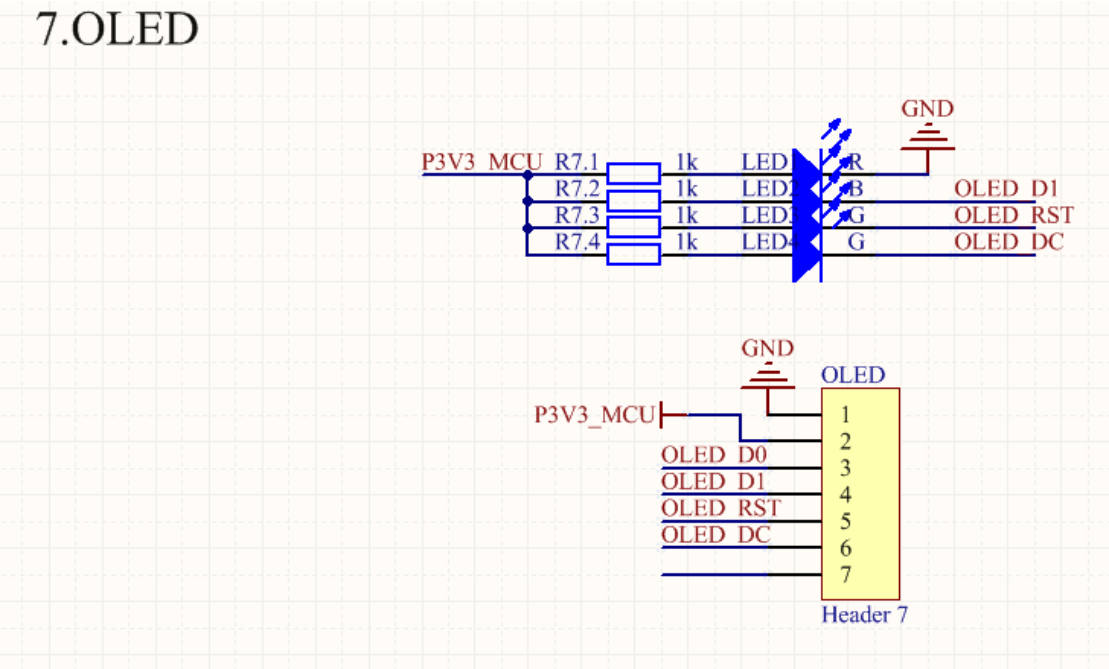


图 3.1.2　OLED 外部电路与接口

3.1.1 电源模块

为满足需求，主板上存在四种不同的供电电压，分别是舵机供电电压 5.5V，编码器及部分芯片供电电压 5V，摄像头及部分芯片供电电压 VCC（5V），OLED 显示屏与单片机供电电压 3.3V。

由于整个车模的供电电压来自 7.2V 镍铬电池，因此在电压转换时应先将其转换到 5V，才能进一步得到单片机和显示屏需要的 3.3V。主控板上大部分芯片与接口都需要用到 5V，为防止信号干扰，在这里我们使用两片 TPS7350 接到电源电压实现这一转化，并以 VCC 和 5V 区分。随后将 VCC 接到 TPS7333 的输入引脚，输出即为 3.3V 电压。为舵机供电的 5.5V 电压则由 LM1117-ADJ 完成，该芯片可以通过改变外接反馈电阻的大小自由控制输出电压，方便使用。该部分电路原理图如图 3.1.3：

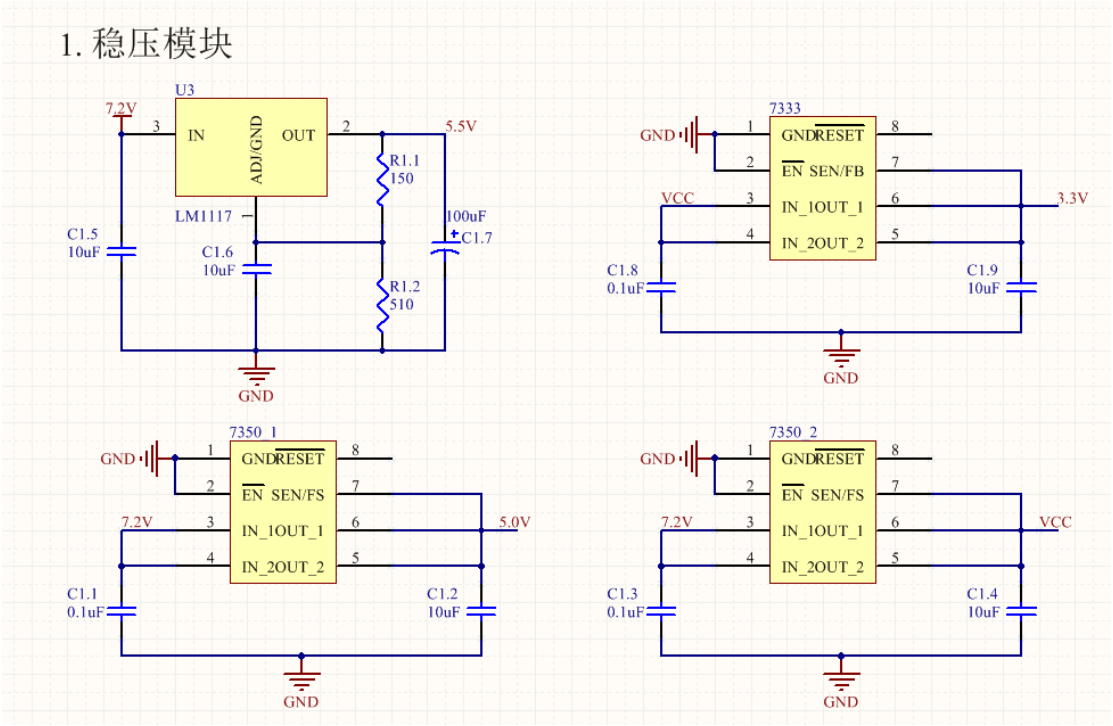


图 3.1.3：稳压模块

另外，应尽可能在单片机供电处放置电容，以起到滤波的作用，如图 3.1.4：

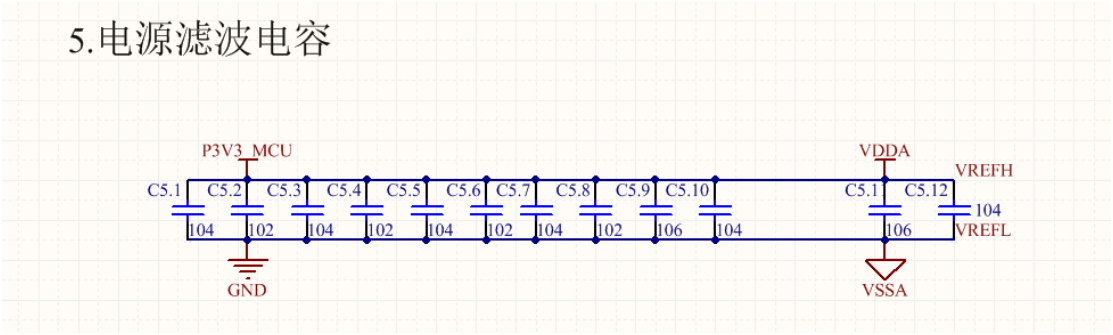


图 3.1.4：滤波电路

3.1.2 旋转按键电路

为了能够便于对小车进行实时调试及参数修改，我们采用旋钮按键电路对参数进行修改并发车。旋钮按键电路引入硬件防抖，提高准确性，其原理图如图 3.1.5：

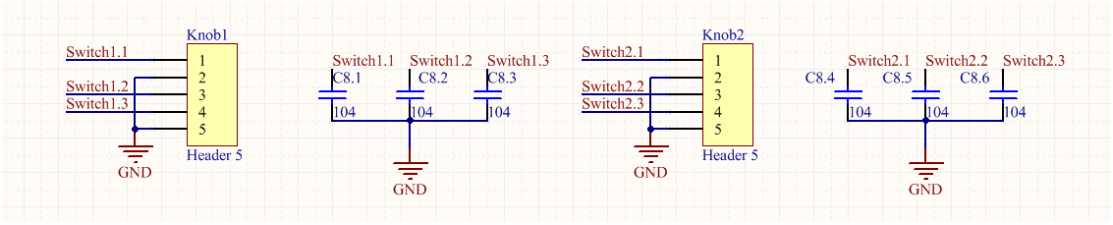


图 3.1.5：旋钮按键电路

3.1.3 电平转换芯片

为实现对舵机以及电机的 PWM 控制，我们选择 74LVC4245 输出 PWM 波，完成舵机转向以及电机加减速的控制。蜂鸣器则作为赛道特殊部分的检测标志，如起跑线、障碍物、坡道等。该部分电路如图 3.1.6：

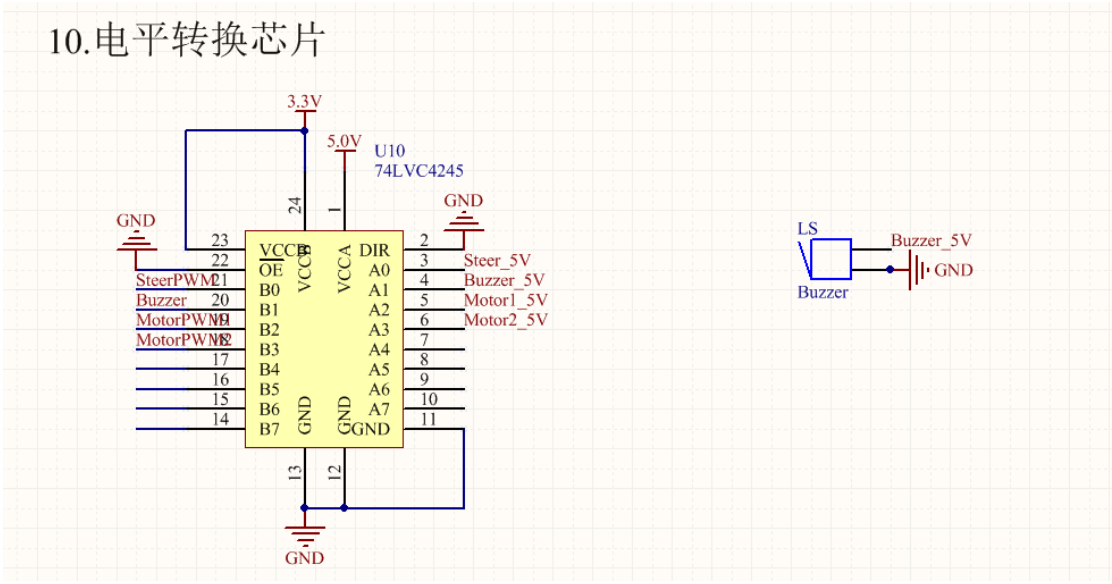


图 3.1.6：电平转换芯片

主控板实物图如图 3.1.7：

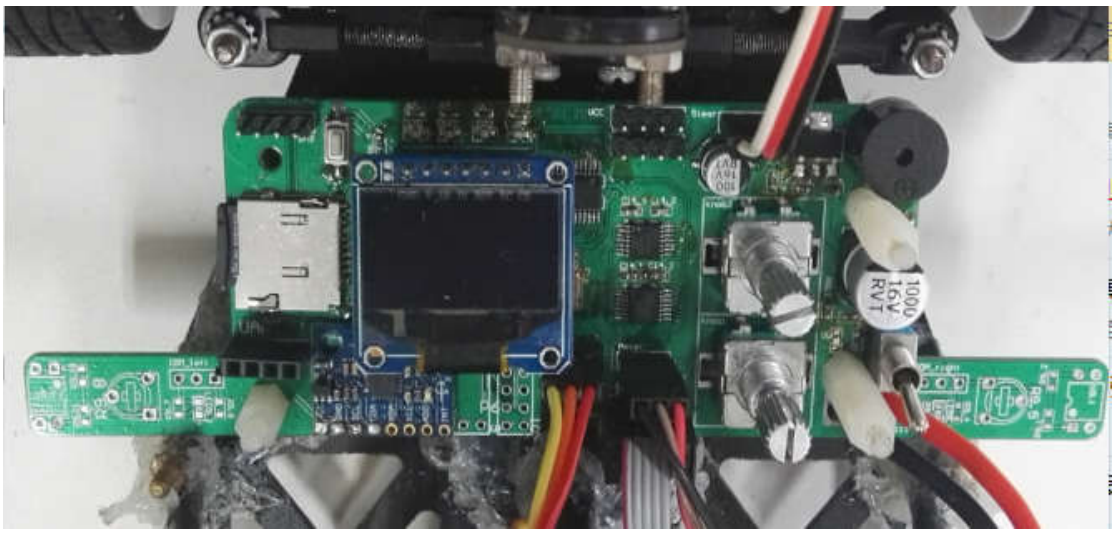


图 3.1.7：主控板实物图

3.2 驱动板电路设计

电机的速度与施加的电压成正比，输出转矩则与电流成正比。对直流电机的控制是一个挑战，因为必须在工作期间改变直流电机的速度。一般的，直流电机高效运行的最常见方法是施加一个 PWM（脉宽调制）方波，其通-断比率对应于所需速度。电机起到一个低通滤波器作用，将 PWM 信号转换为有效直流电平。PWM 驱动信号很常用，因为使用微处理器的控制器很容易产生 PWM 信号。虽然用精确的脉冲宽度可以调节电机的速度，实际应用中的 PWM 频率却是可变的，应对其进行优化，以防止电机颤抖，发出耳朵听得到的噪声。如要使直流电机反转，必须转换电机中电流的方向。

驱动板主要由两个部分组成，H 桥驱动电路与电压升压模块。下面将对这两个部分展开详细描述。

3.2.1 H 桥驱动电路

我们使用高性能 MOSFET 管和 MOSFET 驱动芯片 IR2104S 搭建了 H 桥驱动电路。这款 N 通道功率金属氧化物半导体场效应晶体管最小内部只有 0.49 毫欧姆的电阻，最大允许连续源极电流 400A ，超额符合要求，驱动能力超强，IR2014S 外接自举电路，输出驱动电压在 10V 到 20V 之间 ，具有刹车使能功能，具有较好的保护和恰当的死区，外接电路简单。原理图如图 3.2.1：

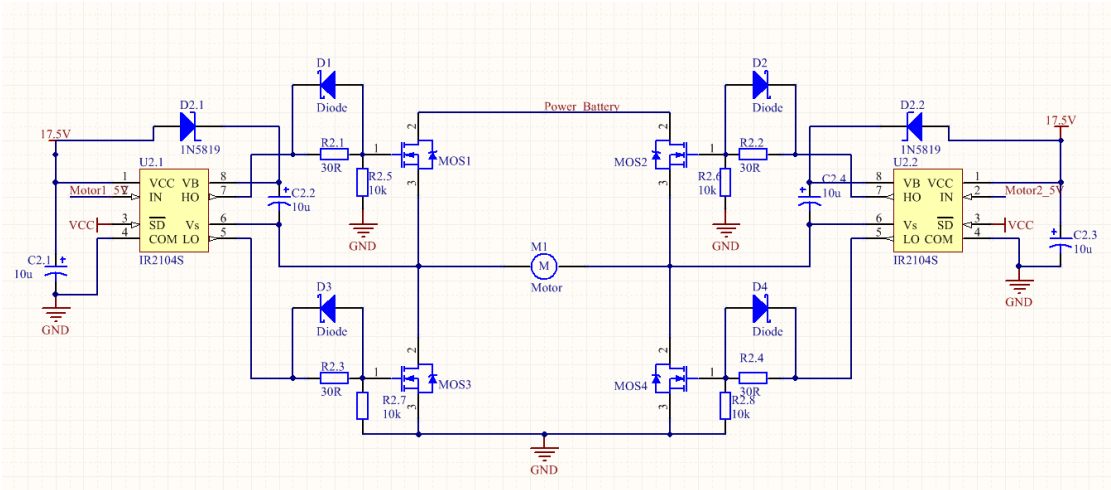


图 3.2.1：H 桥驱动电路

3.2.2 升压模块

上文提到 IR2104S 外接电压在 10V 到 20V 之间，而电源电压只有 7.2V，这就需要在驱动板上加入升压模块。我们选择了 MC34063 升压电路，该器件本身包含了 DC/DC 变换器所需要的主要功能的单片控制电路，由具有温度自动补偿功能的基准电压发生器、比较器、占空比可控的振荡器，R-S 触发器和大电流输出开关电路等组成。仅用少量的外部元件，MC34063 就可以实现升压。最终获得的电压为 17.5V，原理图如图 3.2.2：

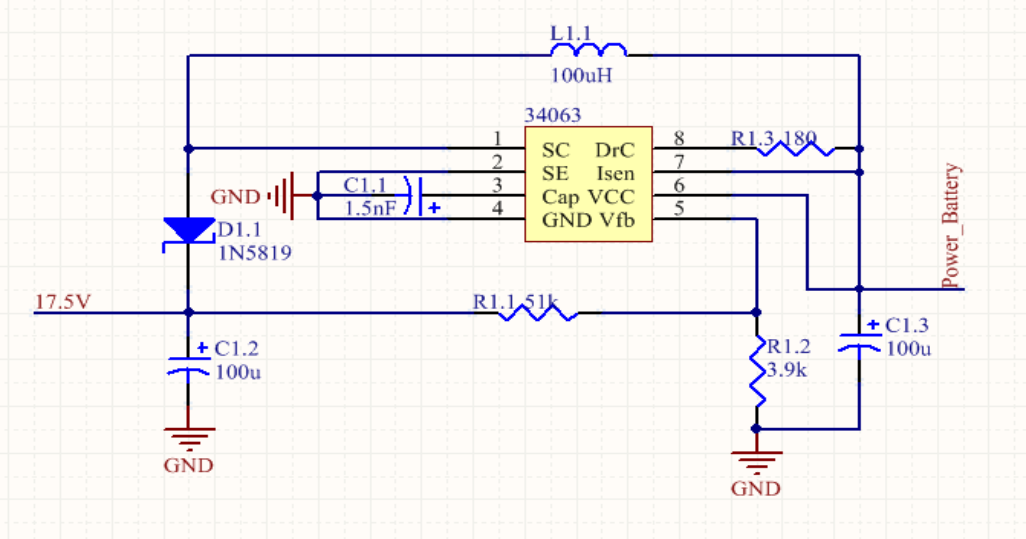


图 3.2.2：升压模块

驱动板实物图如图 3.2.3：

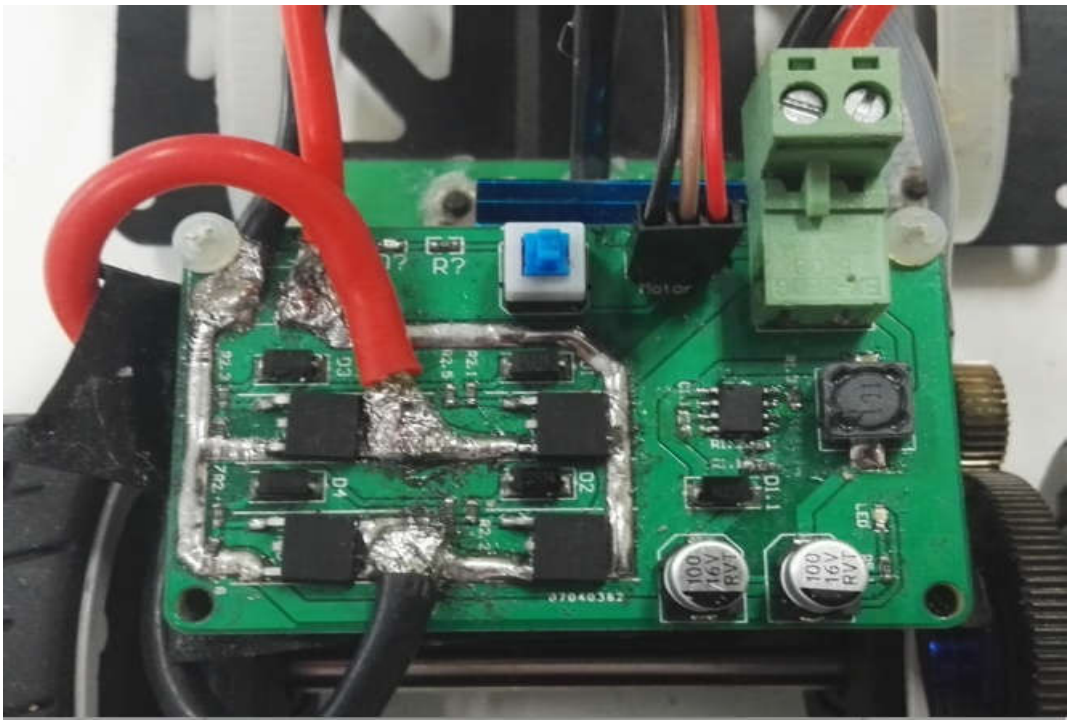


图 3.2.3:驱动板实物图

3. 3 其他外加电路与传感器

3. 3. 1 摄像头的选择

目前市场上主流的摄像头有两种，分别是 CCD 摄像头与 CMOS 摄像头。CCD 摄像头动态特性较强，对比度高，在高速运动的状态下也可以获得清晰的图像。然而 CCD 摄像头需要 12V 工作电压，且重量较大，因此必须安装在较低位置以

维持车体重心，但这样又会导致图像畸变。CMOS 摄像头的稳定性不如 CCD 摄像头，其优点在于低功耗、体积小、质量轻，而且本赛车对图像清晰度要求并不高，因此我们最终决定采用 CMOS 摄像头。

我们分别尝试了 OV5116 和 OV7620 两种不同的摄像头，其中 OV5116 为模拟摄像头，OV7620 为数字摄像头。智能车对摄像头图像分辨率要求并不高，但对动态特性要求非常高，特别是小车在高速行驶入弯或者出弯的时候，图像变化较大，这就对摄像头的自动增益有较高的要求。一般来说，在摄像头图像发生突变时，感光芯片本身会有一段适应时间，这段时间要求越小越好。在这一点上数字摄像头存在着缺陷，它的反应时间略长，这就造成了不便；虽然 OV7620 可以收到数字图像，但经过二值化后并无大量改变。

OV5116 是一款黑白 CMOS 型图像采集集成芯片，分辨率可以达到 640480，传输速率可以达到 30 帧，支持隔行扫描和逐行扫描。在实验过程中，OV5116 图像效果较好，在弯道、回环中也能得到准确的信息，总体上满足我们的要求。而 OV7620 由于解码过程较为繁琐，最终得到的图像也没有本质上的区别，因此我们最终弃用了该摄像头。

除此之外，在方案中，我们使用了 LM1881 视频分离芯片来辅助采样视频。LM1881 提取摄像 头信号的行同步脉冲、消隐脉冲和场同步脉冲，并将它们转换成数字式电平直接输给单片机的外部中断引脚，原理图如图 3.3.1：

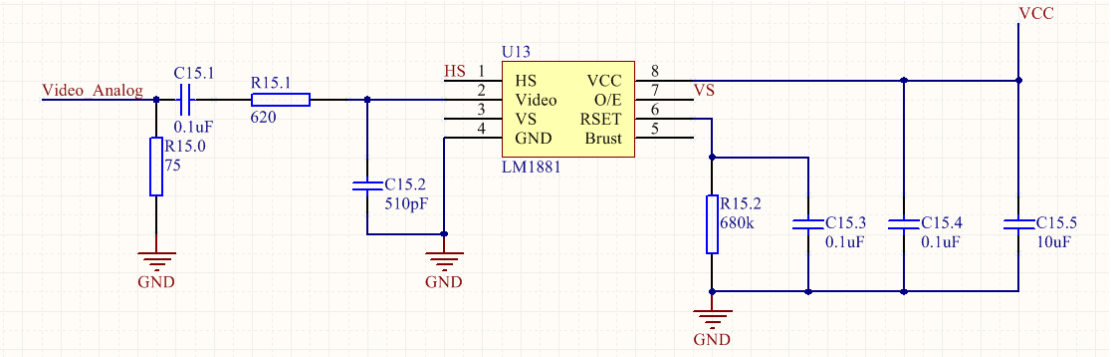


图 3.3.1：视频分离芯片

在图像采集方面使用了硬件二值化，相比于以前的 AD 采集方案，无论是在 CPU 资源利用上，采集时间上还是在图像分辨率上都大大提高了。具体实现是使用高速轨道轨运放 AD8032 先对原始信号进行放大，再将放大后的信号输入由 AD8032 搭成的迟滞比较器上，与设定的阈值进行比较，从而将摄像头输出的模拟信号转化为数字电平。并将该电平分别送入 K60 的两个中断引脚上，通过采集信号的上升/下降沿来采集图像。该部分电路如图 3.3.2 所示：

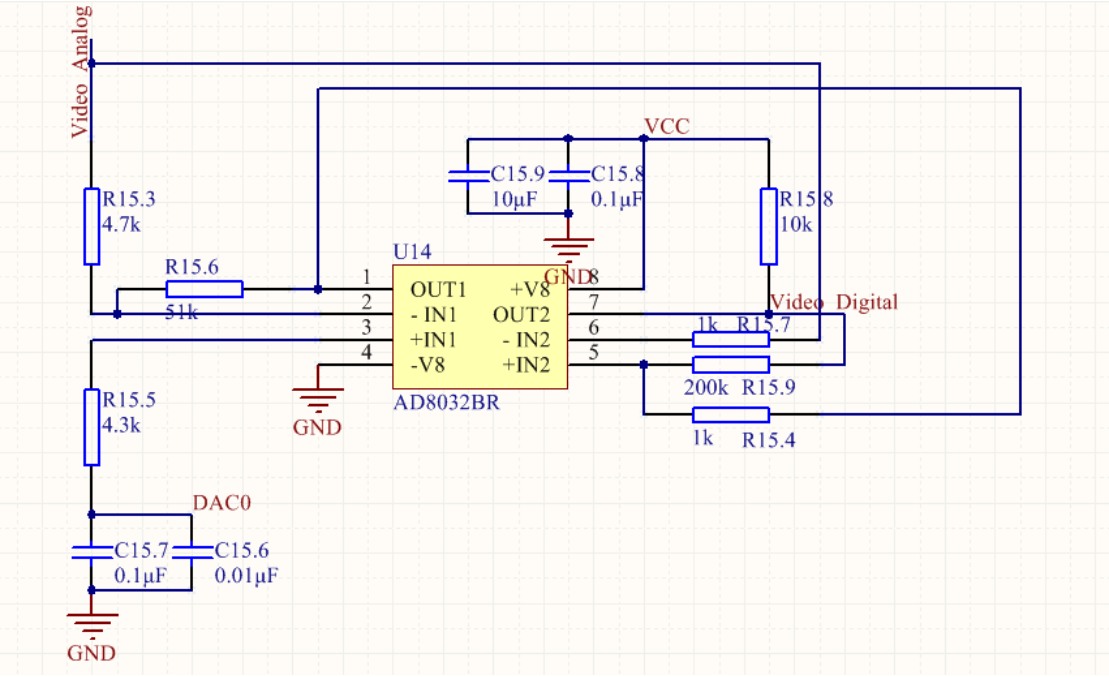


图 3.3.2：放大电路

3. 3. 2 陀螺仪的选择

由于赛道中存在斜坡，如果在通过斜坡时速度太快，就会导致赛车颠簸，严重影响赛车速度，甚至可能造成危险，因此对坡道的检测至关重要。如果用摄像头采集赛道信息判断路况，则技术难度较大且易出现误判。最终，我们选用 MPU-6050 完成这一任务。

MPU-6050 为全球首例整合性 6 轴运动处理组件，相较于多组件方案，免除了组合陀螺仪与加速器时之轴间差的问题，减少了大量的包装空间。MPU-6050 整合了 3 轴陀螺仪、3 轴加速器，并含可借由第二个 I2C 端口连接其他厂牌之加速器、磁力传感器、或其他传感器的数位运动处理(DMP: Digital Motion Processor)硬件加速引擎，由主要 I2C 端口以单一数据流的形式，向应用端输出完整的 9 轴融合演算技术。通过 MPU-6050 检测赛车抬起的角度，即可判断是否进入坡道，从而采取减速的行为。

3. 3. 3 编码器的选择

为了使用闭环控制，我们在汽车模型上附加了编码器。经过机械和电路性能的考虑和挑选，最终选用了龙邱 Mini 512 线编码器。

和其他元件相比，选用编码器可以使电路更加完善，信号更加精确。编码器功耗低，重量轻，抗冲击抗震动，精度高，寿命长，非常实用。为了保证波形稳定，我们先后采用 74HC74 与 74HC86 进行正交解码与隔离，该部分电路如图 3.3.3 所示：

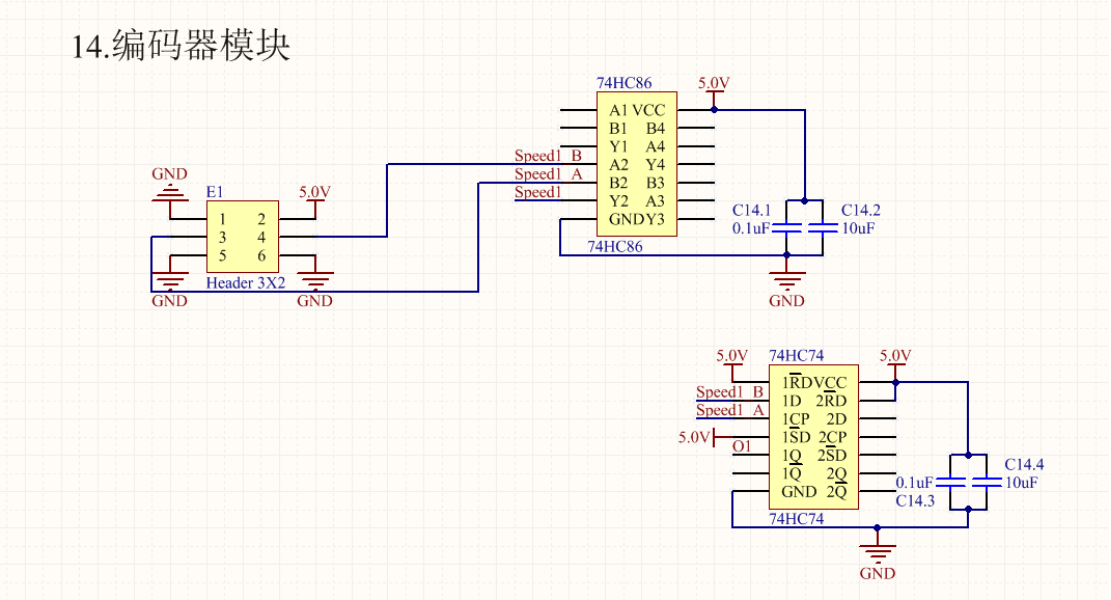


图 3.3.3：解码电路

事实上，我们曾试图使用光电管完成对起跑线的检测，主控板上有一部分集成的循迹电路；但在调试过程中发现光电管极易受到外部环境的干扰，赛车压倒路肩、产生颠簸就有可能造成误判，因此我们放弃了光电管，而使用摄像头检测起跑线。

第四章 智能车软件的设计与实现

软件部分是智能车能够正确快速行驶的关键。我们设计的智能车系统采用 CMOS 摄像头进行赛道识别，图像采集及校正处理就成了整个软件的核心内容。在智能车的转向和速度控制方面，我们使用了经典 PID 控制算法，配合使用理论计算和实际参数补偿的办法，使智能车能够稳定快速寻线。如图 4.1 为系统整体架构：

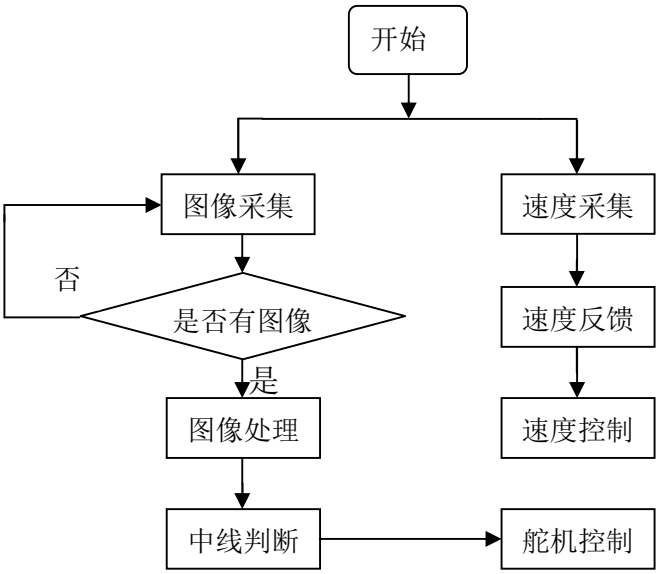


图 4.1 系统程序流程

4.1 图像采集与处理

由于我们采用的是硬件二值化的方案，因此得到的图像较为清晰，在高速运动过程中不会出现明显失真。

考虑到数据的采集主要在中断中完成，且是采用隔行扫描的方式，软件的图像数据处理中设置了两个标志位，一位为数据采集标志位，担当数据采集的数据指针，另一位为数据处理标志位，担当数据处理的数据指针。实现了图像采集与图像处理同步进行，通过软件测量表明在数据采集完成后仅需不到一行的时间就可完成数据处理任务。

我们所做的工作主要在于边沿提取，边沿提取算法的基本思想如下：

（1）直接逐行扫描原始图像，根据设定的阈值提取黑白跳变点；

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

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

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

（5）进出十字的时候，通过校正计算出边沿角度可较好的滤除十字并补线；

（6）由于权重分配的问题，如果不对障碍进行一定的处理的话，控制量对于远端的障碍相应比较小，可能在很接近障碍的时候才有响应，这样很容易造成撞到障碍，为了消除这种影响，我们利用曲线包络的形式将障碍作用的区域人为扩大，

在检测到障碍的同时人为增大偏差，这样有效避免了碰撞障碍的危险。

找到赛道边沿后，利用预计的赛道宽度对其进行进一步判定，其规则如下：如果两侧边沿均找到，则两边沿只差（既赛道宽度）与预期宽度的差值不能太大，否则判为无效；如果仅一边有效，则该边沿距离图像边缘的差值必须小于最大可能宽度。

4.2 弯道策略

在车辆进弯时，需要对三个参数进行设定：切弯路径、转向角度、入弯速度。

其中，切弯路径主要决定了车辆是选择内道过弯还是外道过弯。切内道，路径最短，但是如果地面附着系数过小会导致车辆出现侧滑的不稳定行驶状态，原因是切内道时，曲率半径过小，同时速度又很快，所以模型车需要的向心力会很大，而赛道本身是平面结构，向心力将全部由来自地面的摩擦力提供，因此赛道表面的附着系数将对赛车的运行状态有很大影响。切外道，路径会略长，但是有更多的调整机会，同时曲率半径的增加会使得模型车可以拥有更高的过弯速度。

转向角度决定了车辆过弯的稳定性。合适的转向角度会减少车辆在转弯时的调整，不仅路径可以保证最优，运动状态的稳定也会带来效率的提高，减少时间。在考虑转向角度设置时需要注意以下几个问题：对于检测赛道偏移量的传感器而言，在增量较小时的转向灵敏度；检测到较大弯道时的转向灵敏度；对于类似 S 弯的变向连续弯道的处理。

对于入弯速度的分析，应该综合考虑路径和转向角度的影响。简单而言，我们会采取的是入弯减速，出弯加速的方案，这样理论上可以减少过弯时耗费的时间。为保证赛车行驶的稳定性，我们决定让赛车过弯时的角度、速度随采集到的图像调整，尽可能保持最佳路径。

4.3 起跑与停车检测

我们采用逐列扫描的方式判断起跑线。在行驶过程中，一旦检测到图像的某一场信号上存在跳变沿，且连续的场信号均存在跳变沿，则判定为起跑线。在程序里设定一值，当跳变沿数量大于该数时定位起跑线。左右同时检测到起跑线开始计数，第二次经过时启动减速停车。

检测程序核心部分如下：

```
for(int col=75;col>60;col--)
{
if(img.Star_l>2)    break;
if(input_img[line][col]==bright&&input_img[line][col-1]==dark&&input_img[line][col-2]==dark)
{
for(int temp=col-2;temp>35;temp--)
{
if(img.Star_l>2)    break;
if(input_img[line][temp]==dark&&input_img[line][temp-1]==bright&&input_img[line][temp-2]==bright)
{
for(int i=temp-2;i>15;i--)
{
```

```
if(input_img[line][i]==bright&&input_img[line][i-1]==dark&&((input_img[line][i-2]
==dark)||(input_img[line-1][i-2]==dark)))
{
img.Star_l=1;
if(img.Star_l==1)
{
if(input_img[line+2][col-3]==bright&&input_img[line+3][col-3]==bright&&input_i
mg[line+4][col-3]==bright)
{
if(input_img[line-1][col-3]==dark||input_img[line-2][col-3]==dark)
{   img.Star_l=3;
    break;}
else   img.Star_l=0;          }
else   img.Star_l=0;          }}}}}}}
```

4.4 赛车行驶过程中的控制算法

4.4.1 PID 控制算法介绍

PID 控制是工业过程控制中历史最悠久，生命力最强的控制方式。这主要是因为这种控制方式具有直观、实现简单和鲁棒性能好等一系列的优点。PID 控制主要有三部分组成，比例、积分、微分。比例控制是一种最简单的控制方式。其控制器的输出与输入误差信号成比例关系。偏差一旦产生，调节器立即产生控制作用使被控量朝着减小偏差的方向变化，控制作用的强弱取决于K_p。当仅有比例控制时系统输出存在稳态误差。

为消除稳态误差，引入积分控制。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。为了预测预测误差变化的趋势，引入微分的控制器，这样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。原理框图如图 4.2 所示：

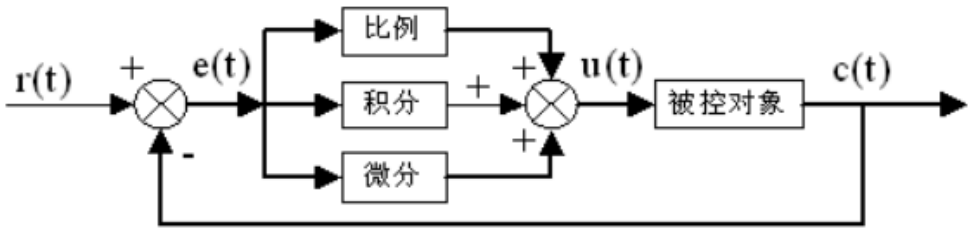


图 4.2 PID 算法原理框图

对应的误差传递函数为：

$$\frac{U(S)}{E(S)} = K_p\left(1 + \frac{1}{T_i} + T_d\right) \qquad ①$$

式中，式中，K_p为比例增益，T_i为积分时间常数，T_d为微分时间常数，U(S)为控制量，E(S)为被控量与设定值 R(S)的偏差。时域表达式为：

$$u(t) = K_p[e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_d\frac{de(t)}{dt}] \quad ②$$

在单片机中，我们仅能对数字信号处理，将上式离散化，得

$$u(k) = K_p\left\{e(k) + \frac{T}{T_i}\sum_{j=0}^{k} e(j) + \frac{T_d}{T}[e(k) - e(k-1)]\right\} \quad ③$$

简单说来，PID 控制器各校正环节的作用如下：

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

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

微分环节：能反映偏差信号的变化趋势(变化速率)，并能在该偏差信号变得太大之前，在系统中引入一个有效的早期修正信号，从而加快系统的动作速度，减小调节时间。

A. 位置式 PID 算法

直接利用上述离散化公式计算。由于积分项是将所有采集值偏差相加，在一段时间后会很浪费单片机资源。对其稍加改进，得到增量型 PID 算法。

B. 增量式 PID 算法

根据公式③得到第 个采样周期的控制量为

$$u(k-1) = K_p\left\{e(k-1) + \frac{T}{T_i}\sum_{j=0}^{k-1} e(j) + \frac{T_d}{T}[e(k-1) - e(k-2)]\right\} \quad ④$$

由③-④得

$$\Delta u(k) = K_p[e(k) - e(k-1)] + K_i \cdot e(k) = K_d[e(k) - 2e(k-1) + e(k-2)] \quad ⑤$$

由此，第 k 个采样时刻实际控制量为，为方便书写，写为

$$\Delta u(k) = q_0 e(k) + q_1 e(k-1) + q_2 e(k-2) \quad ⑥$$

其中，$q_0 = K_p(1 + \frac{T}{T_i} + \frac{T_d}{T})$，$q_1 = -K_p(1 + \frac{2T_d}{T})$，$q_2 = K_p\frac{T_d}{T}$

PID 控制流程图如图 4.3 所示：

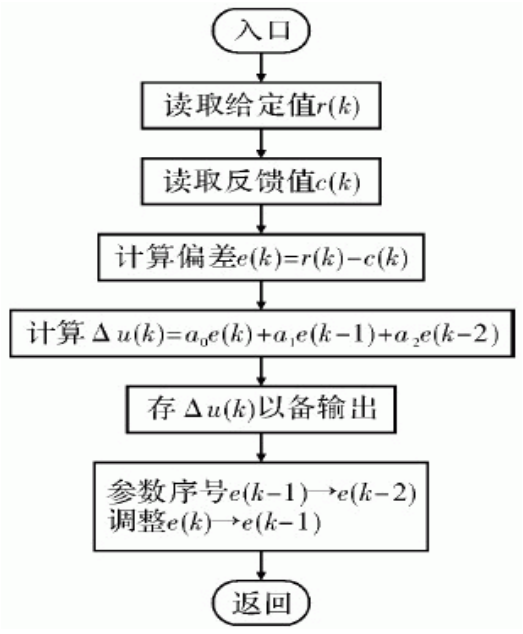


图 4.3　PID 控制流程图

由上可知，利用三个数据，递推使用，即可完成 PID 控制量。

智能车要能在赛道上快速稳定的运行，除了摄像头前瞻能精确地检测到路径信息外，转向舵机和驱动电机的准确控制和其两者之间的良好配合也是非常重要的因素。当智能车在直道上时，驱动电机要能快速提速并尽量达到最大速度；当检测到入弯时，驱动电机要立刻减速并降到一个比较低的速度，给微控制器及舵机争取足够的时间来计算、调整，来获得正确的转向角度；在弯道内适当提速，并保持角度不变，为出弯地加速节约时间；准确判断到出弯后，舵机立刻回正，驱动电机以最大的加速度加速。下面将具体介绍转向舵机和驱动电机的控制方法。

4.4.2 转向舵机控制算法

转向角度决定了车辆过弯的稳定性。合适的转向角度会减少车辆在转弯时的调整时间，不仅路径可以保证最优，运动状态的稳定也会带来效率提高，减少时间。舵机的转角直接影响到车辆转向角度的大小。根据摄像头安装的高度和角度，可以采用不同的控制算法。我们的摄像头前瞻约为 106cm。可以使赛车切赛道内道行驶，对于曲率小的弯道可以让小车跑得更直，相当于缩短了赛道长度同时提高了赛车速度，所以正确判断赛道信息尤为关键。

赛车的前瞻越长，在车子运行的过程中，越可能受到旁边赛道或者其他信号的干扰而引起较大位置的跳变，这在车子定位中是不能出现的，所以要通过软件来消除。由于舵机是一个具有较大延迟的执行机构，所以在舵机控制上，我们采用 PD 控制。将积分项系数置零，我们发现相比稳定性和精确性，舵机在这种随动系统中对动态响应性能的要求更高。更重要的是，在K_i置零的情况下，通过合理调节K_p，发现车能够在直线高速行驶时仍能保持车身非常稳定，没有震荡，基本没有必要使用K_i参数。另外，微分项系数K_d使用定值，可以使舵机在赛道中保持较好的动态响应能力。

4.4.3 电机驱动控制算法

对于速度控制，我们采用了增量式 PID 控制算法与 Bang-bang 控制相结合，基本思想是直道加速，弯道减速，通过引入速度的反馈值实现闭环控制。在最初的调试过程中，每幅图像得到的黑线位置与速度 PID 参考值成线性关系，导致加减速反应不及时，实际速度与实际速度有一定偏差。在改成二次曲线关系后，这一现象得到缓解，在过弯过程中加减速较为灵敏。具体流程图如图 4.4：

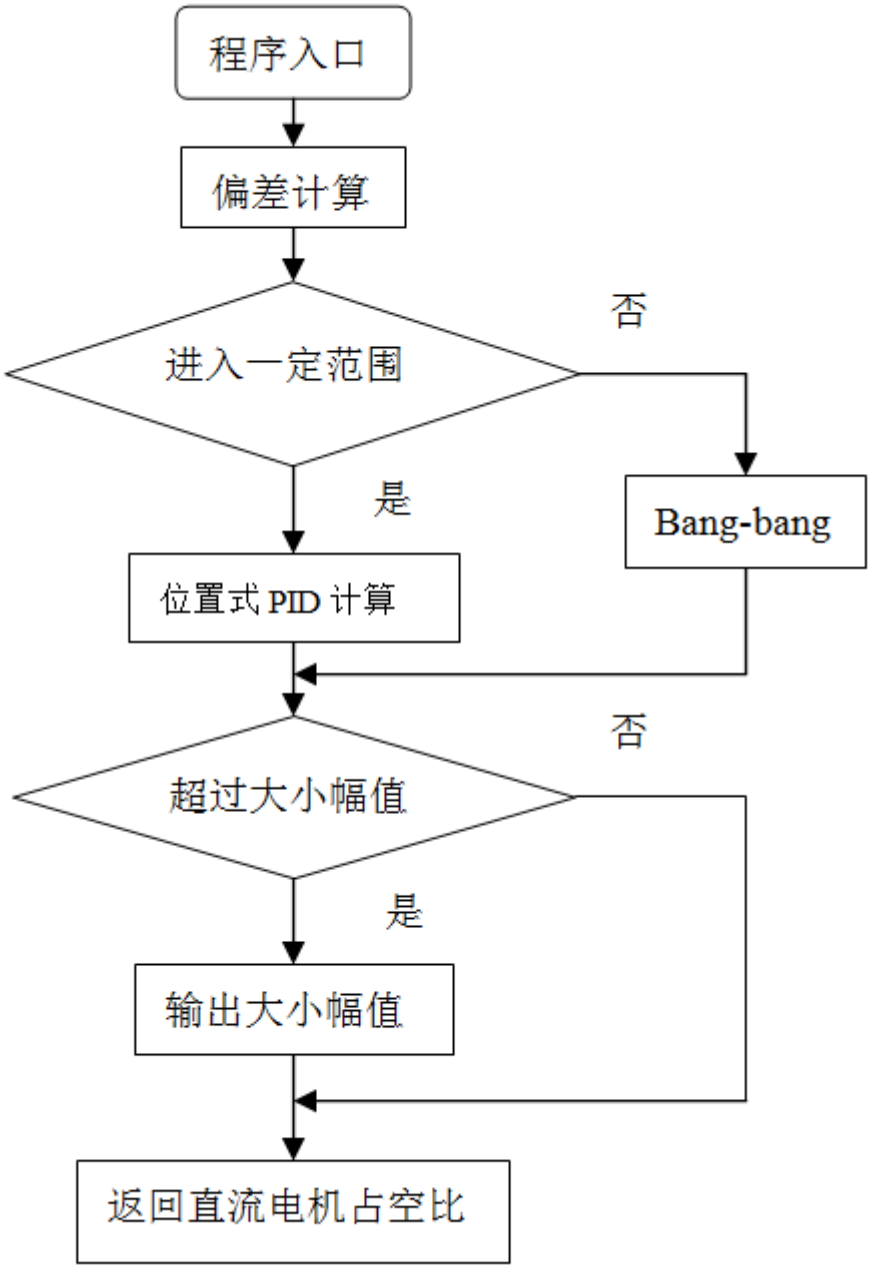


图 4.4 电机控制流程图

4.5 其他控制算法

4.5.1 Bang-bang 控制

Bang-bang 控制的思想是反馈值若比设定值小，就把控制值设置为最大，否则设置为最小，用公式表示为：

$$\text{if}(error < 0) \qquad u(k) = u_{max}$$
$$\text{else if}(error > 0) \qquad u(k) = u_{min} \qquad ⑦$$

这种控制方式会有较快的响应速度。但同时速度超调现象很严重，受赛道摩擦因数影响大，容易引起小车的不稳定，因此 Bang-bang 控制是有一定的限制范围。

4.5.2 模糊控制

一般控制系统包含了五个主要部分，即：定义变量、模糊化、知识库、逻辑

判断及反模糊化，以下将就每一部分做简单的说明：

1、定义变量：也就是决定程序被观察的状况及考虑控制的动作，例如在一般控制问题上，输入变量有输出误差 E 与输出误差之变化率 CE，而控制变量则为下一个状态之输入 U。其中 E、CE、U 统称为模糊变量。

2、模糊化：将输入值以适当的比例转换到论域的数值，利用口语化变量来描述测量物理量的过程，依适合的语言值求该值相对之隶属度，此口语化变量我们称之为模糊子集合。

3、知识库：包括数据库与规则库两部分，其中数据库是提供处理模糊数据之相关定义；而规则库则藉由一群语言控制规则描述控制目标和策略。

4、逻辑判断：模仿人类下判断时的模糊概念，运用模糊逻辑和模糊推论法进行推论，而得到模糊控制讯号。此部分是模糊控制器的精髓所在 。

5、解模糊化：将推论所得到的模糊值转换为明确的控制讯号，作为系统的输入值。

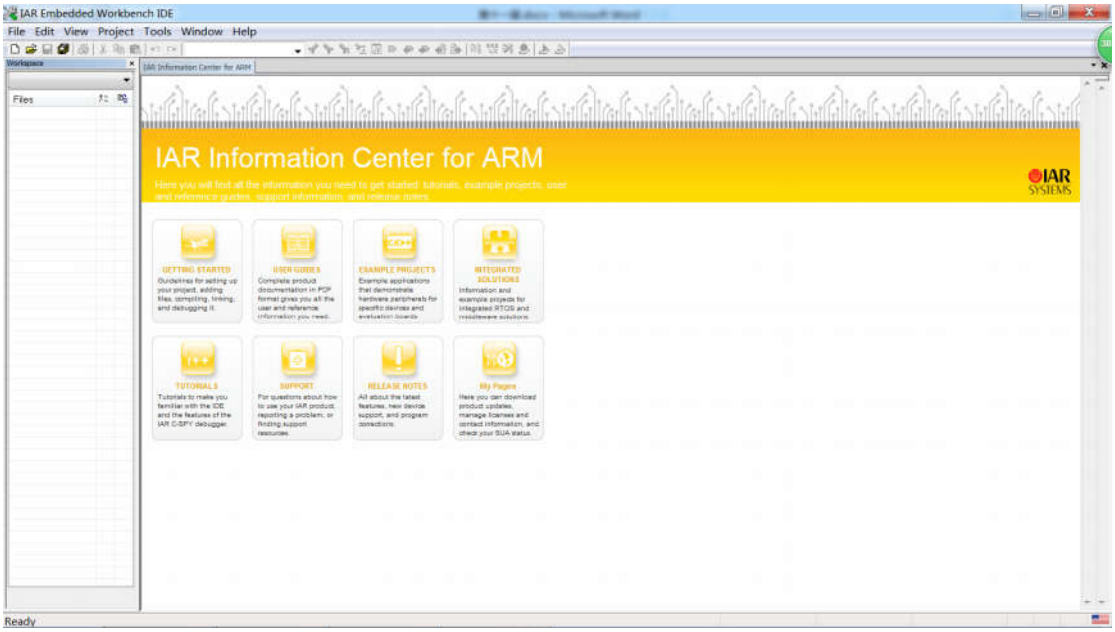
模糊算法可以解决一些非线性问题， 将赛道分为直线、入大小弯、出大小弯、蛇形弯道，对应的直线加速、入大弯减速转方向、入小弯制动转方向、出弯加速、蛇形弯道直接通过（若可以达到这种前瞻性）。要达到这种控制要通过实际检测，分析大量赛道信息，找出它们的特征。

第五章 开发工具与系统调试

5.1 开发工具

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

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



5.2 调试过程

在调试赛车参数的过程中，使用了 MK60DN512ZVLL10 的串口。行驶过程中数据被存在 SD 卡内，行驶完毕后通过蓝牙将数据发送到 PC 端的上位机。上位机我们使用的是友善串口调试助手，可以接收到赛车行驶过程中的舵机预设转角、实际转角、预设速度、实际速度等参数。将这些数据导入到 Excel 绘制成折线图，通过图像可以直接看出赛车的缺陷。

在基本程序完成之后还需要在现场对程序进行微调。所以我们自制了旋钮按键模块。上场后我们的操作选手观察赛道情况，然后根据目测结果通过按键对程序进行一下微调：

1）各种形式赛道的速度参数；

2）水平传感器和倾斜传感器在舵机控制中占得权重；

3）电机、舵机 PD 控制的K_p和K_d等参数。

此外，我们还用液晶显示各个电感的电压数值，方便临时通过主板上设置的可调电位器对信号采集后放大倍数的调整。当然所有的参数假使都已经有了较为合理的默认数值，这样可以极大的节省参数调整时间

第六章 赛车主要技术参数说明

项目	参数	
赛题组	摄像头组	
赛车基本参数（mm）	长	300
	宽	175
	高	360
车重	1100g	
功耗	空载	10W
	带载	12W 以上
电容总容量	1200uF	
传感器	CMOS 摄像头	1 个
	陀螺仪	1 个
	编码器	1 个
除车模原有驱动电机、舵机外伺服电机的个数	0 个	
赛道信息检测	前瞻（毫米）	106
	频率（次/秒）	50
	空间精度（毫米）	12.5

第七章 总结

我们小组从今年一月底开始着手准备“恩智浦”杯智能车大赛，经过半年有余的努力，如今可以做到较好完成比赛，达到了最初的目标。在这期间我们经历了大量失败，几乎所有零部件都遭到过损坏，一部分甚至多次更换。比赛的前昔，我们依然不断享受着惊魂一刻的魅力。赛车参数几经作废，又经过不断的优化，才有了今天的结果。而我们收获的不仅仅是一辆赛车，更重要的是在这个过程里积累的经验和永不言弃的精神，它将是我们一生里难忘的回忆。

报告的主体由机械结构、硬件电路和控制算法三个部分组成。机械方面的调整主要包括主销后倾角与重心的调整，以及舵机的安装。最一开始我们没有调整差速器，导致赛车提速与预想有一定偏差，而在安装舵机时又经过了多次的实验与调整才使其对称放置。电路部分尤其是主控板改动较大，不仅改变了摄像头的供电方式，还将视频分离电路整体集成到了主控板；另外，我们没有直接将 K60 最小系统板插在电路板上，而是同样将其集成到一起并加入了滤波电路；参数调节方式也略有不同。软件方面我们结合实际，利用蓝牙等获得赛车状况，加入了许多细节方面的控制，并适当调整了过障碍物、过弯的算法等。可以说调试过程对算法调整起到了至关重要的作用。

从最初报名到正式参赛接近十个月，在十个月的时间里我们取得了长足的进步，各个方面都有所提高。但我们依然有很多地方没有做到位，比如赛车的路径并不完美，速度控制始终与预想有一定差距，这些问题都需要进一步解决。

最后再次感谢学校对我们的支持，为我们准备了专门的实验室与试车场地，同时有充足的器材与元件，大大减少了我们的麻烦；感谢指导教师对我们倾囊相授，帮助我们解决了许多技术上的难题。也希望我们这一次能取得更好的成绩，在各方面有更高的突破。

参考文献

[1]卓晴，黄开胜，邵贝贝等.学做智能车：挑战“飞思卡尔”杯[M].北京:北京航空航天大学出版社，2007.
[2]谭浩强著. C 程序设计. 北京：清华大学出版社，2003.
[3]张军. AVR 单片机应用系统开发典型实例. 北京：中国电力出版社，2005.
[4]刘进，齐晓慧，李永科.基于视觉的智能车模糊 PID 控制算法[J].兵工自动化，2008,27(10).
[5]唐建文.智能小车控制系统的设计与实现[D].广东：广东工业大学硕士论文，2008.
[6]曾星星.基于摄像头的路径识别智能车控制系统设计.湖北：湖北汽车工业学院学报，2008.
[7]张晓飞，袁祥辉. 基于 DSP 成像系统的视频图像采集部分的实现. 压电与声光，第 24 卷，第三期. 2002 年 6 月.
[8]孙涵，任明武，唐振民等.基于机器视觉的智能车辆导航综述[J].公路交通科技，2005,22(5):132~135
[9] Brown M A, Blackwell K T, Khalak H G..Multi-Scale Edge Detection and Feature Binding: An Integrated Approach[J]. Pattern recognition,1998, 31(10), 1479-1490.
[10]Park K.H，Bien Z，Hwang D.H. A study on the robustness of a PID - type iterative learning controller against initial state error [J]. Int. J. Syst. Sci. 1999, 30(1)，102~135.

附录 程序源代码

```
#include "common.h"
#include "car_global.h"

#if 1
/*函数声明*/
void init_all();
void init_NVIC();
void init_battery();
void init_key();
void init_motor();
void init_servo();
void init_pit();
void init_i2c();
void init_all_pulse_counter();
void init_pulse_counter(PortPinsEnum_Type counter_input,
                        uint8 dma_chx,
                        PortPinsEnum_Type dir_input);
void init_flash();
void init_sdhc();
void init_paranum();
void init_camera();
void porta_isr();
void portb_isr();
void portc_isr();
void pit0_isr();
void pit1_isr();
uint16 battery_check();
uint16 phototube_check();
void key_check();
void show_oled();
void show_upperpage(int page);
void show_changeable();
void show_fullpage(int page);
void save_flash();
void send_uart();
void write_sd();
void save_RAM();
void save_sd(int value,char* label);
void save_u16(uint16);
void status_select();
void data_input();
void data_process();
```

```
void data_output();
void data_save();
int get_speed();
void die(FRESULT rc);
DWORD get_fattime (void);
int Track_Cal(uint8 *Edge_L,uint8 *Edge_R);
int Steer_PWM_Cal(int track_pos);
int Speed_Cal(int track_pos);
int Motor_PID_Cal(int fbspeed,int setspeed);
unsigned short CRC_CHECK(unsigned char *Buf, unsigned char CRC_CNT);//CRC
校验，确保发的数正确
void OutPut_Data(void);

/*
*   全局变量声明
*/
uint32 T = 0;                                   //PIT 计时器
uint32 starttime = 0;
STATUS_BUTTON status_button;                    //按钮状态
STATUS_CAR_STRUCT mycar = {0};                  //小车总状态
indata_STRUCT indata = {0};                     //输入变量
outdata_STRUCT outdata = {0};                   //输出变量
track_STRUCT track = {0};                       //运行参数
setpara_STRUCT setpara = {0};                   //设定参数
OLED_STRUCT oled = {0};                         //屏幕显示
IMG_STRUCT img = {0};                           //图像处理
FLAG_STRUCT flag;
uint32 save_ram_no = 0;
uint32 save_uart_no = 0;
uint8 RAM_BUFF[60][512] = {0};
uint8 SEND_BUFF[512] = {0};
uint32 PITValA,PITValB,PITVal;
uint32 temp;
#endif
#if 1 //主函数
void main (void)
{
  DisableInterrupts;
  init_all();                   //小车所有的初始化函数
  EnableInterrupts;
  PTB19_O     = 0;
  img.Star_r=0;
  img.Star_l=0;
```

```
    while(1)                    //程序主循环
    {

        if((mycar.end||img.flag.barrier)&&setpara.Buzzer)
        {
            PTB19_O     =1;
        }
        else
            PTB19_O    =0;
        key_check();            //按键查询与响应
        show_oled();            //屏幕显示
        send_uart();            //蓝牙串口发送
        write_sd();             //SD 卡写入
    }
}
#endif


#if 1 //主循环
void key_check()
{
    //记录按键时间
    uint32 pushtime=T;

    //旋钮或拨轮按下操作后屏幕初始化，以修复花屏
    if(status_button==PRESS||status_button==PRESS2)
        OLED_Init();

    switch(status_button)
    {
    case PRESS:
        while(!PTC4_I);

        if(T-pushtime<500)
        {
            oled.changepara ^= 1;       //状态取反
        }
        else //此功能常用作比赛发车按键
        {
            save_flash();
            int start_time=T;
            while(T-start_time<1000);
            mycar.status = 1;
```

```
        mycar.RunTime = 0;
        img.Star_l = 0;
        img.Star_r = 0;
        mycar.end = 0;
        img.Star_l=0;
        img.Star_r=0;
        save_ram_no = 0;
        save_uart_no = 0;
        flag.TO_SAVE_FLASH_NO = 0;
        indata.angle_slope = 0;
        flag.SLOPE = 0;

         flag.start = 0;
    }
    break;
case PRESS2:
    while(!PTC9_I);

    if(T-pushtime<500)
    {
        flag.TO_SEND_SD = 1;
    }
    else
    {
        init_setpara();             //初始化所有参数
    }
    break;
case CW2:

    if(oled.changepara)     //参数改变状态
    {
        if(oled.para_select >0)
            oled.para_select --;
        else
            oled.para_select = oled.para_num-1;
    }
    else                        //参数选择状态
    {
        if(oled.showpage > oled.showpage_min)
            oled.showpage --;
        else
            oled.showpage = oled.showpage_max;
    }
    break;
```

```
    case CCW2:

      if(oled.changepara)     //参数改变状态
      {
        if(oled.para_select <oled.para_num-1)
          oled.para_select ++;
        else
          oled.para_select = 0;
      }
      else                    //参数选择状态
      {
        if(oled.showpage < oled.showpage_max)
          oled.showpage ++;
        else
          oled.showpage = oled.showpage_min;
      }
      break;
    default:
      break;
    }
    //清除按键状态
    status_button = NONE;
}
#endif
#if 1 //数据采集
void data_input()
{
    mycar.batt_volt = battery_check();          //电池电压检查
    indata.speed = get_speed();                   //获取速度
    indata.real_speed=indata.speed;
    indata.real_speed=15000*indata.real_speed/19200;//实际速度
    indata.speed = 15000*indata.speed/19200;    //转换为实际速度,cm/s
      if(mycar.mpu6050_error==0)
    {

indata.mpu6050.gyr_y=(int16)((int8)MPU6050_ReadReg(MPU6050_GYRO_YOUT
));
    }
    while(mycar.mpu6050_error > 0)
    {
      mycar.mpu6050_error_count ++;
      I2C_InitTypeDef i2c_init_param = {0};
      i2c_init_param.I2C_I2Cx  =  MPU6050_I2CX;           //在 MPU6050.h 中修改
该值
```

```
    LPLD_I2C_Deinit(i2c_init_param);

    uint8 SELF_ID = 0;

    i2c_init_param.I2C_IntEnable = FALSE;
    i2c_init_param.I2C_ICR = MPU6050_SCL_400KHZ;  //可根据实际电路更改
SCL 频率
    i2c_init_param.I2C_SclPin = MPU6050_SCLPIN;    //在 MPU6050.h 中修改该
值
    i2c_init_param.I2C_SdaPin = MPU6050_SDAPIN;    //在 MPU6050.h 中修改
该值
    i2c_init_param.I2C_OpenDrainEnable = TRUE;
    i2c_init_param.I2C_Isr = NULL;

    LPLD_I2C_Init(i2c_init_param);

    mycar.mpu6050_error = 2;      //修理状态

    SELF_ID = MPU6050_ReadReg(MPU6050_WHO_AM_I);      //通过设备 ID
判断设备是否为 MPU6050
    if(SELF_ID == MPU6050_ID)
    {
      mycar.mpu6050_error = 0;
    }
    else
    {
      printf("MPU6050 初始化失败!\r\n");
      printf("设备 ID: 0x%X\r\n", SELF_ID);
    }

indata.mpu6050.gyr_y=(int16)((int8)MPU6050_ReadReg(MPU6050_GYRO_YOUT
));
  }

}

#endif
#if 1 //数据处理
void data_process()
{
  outdata.Track_Pos = Track_Cal(img.Edge_L,img.Edge_R);
  outdata.Steer_PWM = Steer_PWM_Cal(outdata.Track_Pos);

  outdata.Laststeer=outdata.Steer_PWM;
```

```
    outdata.Motor_Speed = Speed_Cal(outdata.Track_Pos);
    outdata.Motor_Power = Motor_PID_Cal(indata.speed,outdata.Motor_Speed);
    outdata.Laststeer=outdata.Steer_PWM;

}
#endif
#if 1 //数据输出
void data_output()
{
    LPLD_DAC_SetBufferDataN(DAC0, setpara.Threshold, 1);//二值化阈值

    if(mycar.status != 0)
    {
        if(outdata.Motor_Power>0)
        {
            LPLD_FTM_PWM_ChangeDuty(FTM0, FTM_Ch4,outdata.Motor_Power);
            LPLD_FTM_PWM_ChangeDuty(FTM0, FTM_Ch5,0);
        }
        else
        {
            LPLD_FTM_PWM_ChangeDuty(FTM0, FTM_Ch4,0);
            LPLD_FTM_PWM_ChangeDuty(FTM0, FTM_Ch5,-outdata.Motor_Power);
        }
    }
    else
    {
        LPLD_FTM_PWM_ChangeDuty(FTM0, FTM_Ch5,0);
        LPLD_FTM_PWM_ChangeDuty(FTM0, FTM_Ch4,0);
    }
    LPLD_FTM_PWM_ChangeDuty(FTM2, FTM_Ch0,outdata.Steer_PWM);

}
#endif
#if 1 //数据存储
void data_save()
{
    if(mycar.status==1&&flag.TO_SEND_SD == 0)
        save_RAM();
}
void write_sd()
{
    uint16 temp_flash_read;
```

```
    if(mycar.status)
    {
      if(flag.TO_SAVE_FLASH_NO<(2*save_ram_no)/512)
      {
        Int
i=disk_write(0,RAM_BUFF[flag.TO_SAVE_FLASH_NO%50],flag.TO_SAVE_FLA
SH_NO,1);
        if(i==0)
        {
          flag.TO_SAVE_FLASH_NO++;
        }
        else
        {
          for(int i=100;i--;);
        }
      }
    }
    if(flag.TO_SEND_SD)
    {
      DisableInterrupts;
      save_RAM();            //调用 save_RAM()函数的打印标签功能
      flag.TO_SEND_SD = 0;
      for(int i=0;i<flag.TO_SAVE_FLASH_NO;i++)
      {
        int j;
        while(disk_read(0,SEND_BUFF,i,1));
        for(j=0;j<511;j+=2)
        {
          temp_flash_read=SEND_BUFF[j]*256+SEND_BUFF[j+1];
          if(temp_flash_read==0xAAAA)
            printf("\n");
          else
            printf("%d\t",(int16)temp_flash_read);
        }
      }
      EnableInterrupts;
    }
}
```