第十一届“恩智浦”杯全国大学生智能汽车竞赛

技术报告

杭州电子科技大学信息工程学院

HANGZHOU DIANZI UNIVERSITY INFORMATION ENGINEERING SCHOOL

学　　校：　杭州电子科技大学信息工程学院

队伍名称：　杭电信工摄像头9队

参赛队员：　章琦杰

林　涛

张道龙

指导老师：　刘建岚　余皓珉

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，比赛组委会和恩智浦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：

指导老师签名：

日　　期：2016/08/12

摘要

本文介绍了杭州电子科技大学信息工程学院的队员们在准备此次比赛中的成果。本次比赛采用大赛组委会提供的1:16仿真车模，硬件平台采用带MK60DN512ZVLQ10单片机的K60环境，软件平台为Keil开发环境。

文中介绍了本次我们的智能车控制系统软硬件结构和开发流程，整个智能车涉及车模机械调整，传感器选择，信号处理电路设计，控制算法优化等许多方面。整辆车的工作原理是先将小车的控制周期中提取出相应的时间片，相应的时间片用来控制车体的平衡，留下的时间片用来控制速度和转向，由ov7225鹰眼摄像头采集到赛道的信息到单片机，再由单片机读取信号进行分析处理，运用我们自己的软件程序对赛道信息进行提取并选择最佳路径，通过对电机的精确控制从而实现小车在赛道上精彩漂亮的飞驰！

为了进一步提高小车在运行时的稳定性和速度，我们组在软件方面使用了多套方案进行比较。更新了SD卡技术实时存储赛道信息。硬件上为了稳定的考虑，采用了以前比较稳定的方案，但是在电源部分做了调整，使得整车的电源裕度更大，硬件鲁棒性更强。为更好的分析调车数据，我们继承并且改进上届的上位机，用C#软件编写了新的上位机程序来进行车模调试，很大程度上提高了调车效率。在进行大量的实践之后，表明我们的系统设计方案完全是可行的。

关键字:恩智浦智能车，MK60DN512ZVLQ10,ov7225摄像头,PID控制，C#上位机

Abstract

This paper introduces the hangzhou university of electronic science and technology information engineering college players in preparation for the achievements in this match.The game USES the competition committee provided for simulation models, MK60DN512ZVLQ10 microcontroller hardware platform using belt K60 environment, software platform for the Keil development environment.

The whole car works is the first control cycle of the car to extract the corresponding time slice, the time slice corresponding to control the balance of the body, leaving time slices used to control the speed and steering, collected by the Hawkeye camera to track information into the device, and then read by the microcontroller signal analysis and processing, using our own software program to track information is extracted and select the best path through the precise control of the motor in order to achieve pretty exciting car on the track speeding!

In order to further improve the stability and speed of the car at run time, our group in the aspect of software USES the multiple sets of scheme comparison.Update the SD card technology real-time information storage track.Hardware to stabilize, the relatively stable solution before, but in the power part of the adjustment, makes the vehicle power supply margin larger, stronger robustness hardware.For a better analysis of the shunting data, we inherit and improve the previous PC, a new PC software written in c # program to adjustment of the models, greatly improve the efficiency of shunting.After a lot of practice, it shows that our system design scheme is feasible completely.

Key words: car NXP intelligence, MK60DN512ZVLQ10, Hawkeye , PID control, the c# upper machine,

目录

第一章. 引言

在以往的智能车比赛中，我校凭借可靠的机械架构，出色的软件控制，稳定的硬件电路从而取得了优异的成绩，但是我们觉得学习是无止境的，本着对更高目标的追求和对智能车比赛的热爱，我们在前辈的基础上又对机械架构，硬件设计和软件控制算法上进行了大胆的尝试和创新。

前期我们花了大量的时间去选择车模的架法，摄像头的选择，硬件方案的实验，大体上确定了车模的机械架法和硬件上的芯片选型，PCB布局等。中期是以软件调试为主，当然这段时期硬件上也做了合理的调整，特别是在机械上我们花了很多心思，实现了各种配件的专业设计和制作，这是往届没做过的，实践证明在机械上的优化对软件的调车有很大的推动作用。后期软件算法已经确定，硬件上也很稳定，所以我们在原先的机械思路上更近一步，实践证明低重心的智能车确实过弯比以前很多，但是由于今年多加了路肩的缘故，将车身压平使得车子底板与地面更近，路径不佳会导致卡路肩使得比赛失败。权衡利弊之后我们选择不降低车身底盘。正是这种对智能车精益求精的追求，使得我们的智能车有了明显的提速。

在本文报告内容框架安排上，第一章是引言，主要对车模设计和报告的架构进行了概述；第二章是系统硬件电路的设计，详细介绍了电源部分、电机驱动等电路模块的设计思路及方案。第三章是机械架构调整，着重分析了摄像头的新式架法和电池的驾法，同时也对小车机械的整体布局和架车理念做了细致的解释和说明。第四章是软件系统设计，主要介绍了图像处理，车身打角控制和速度控制的模块编写，整体的模块调试和上位机的开发。第五章是开发智能车调试过程的说明，讲述了我们自己在调车训练时的方法和训练工具。第六章为车模的技术参数说明。第七章为总结部分。

第二章. 系统硬件电路设计

2.1. 硬件电路整体架构框图

我们的智能车硬件系统主要包括以下模块：电源管理模块、摄像头模块、、电机驱动模块及测速模块、起跑线检测模块，调试模块、电源管理模块、MK60DN512ZVLQ10电路。

智能车硬件电路整体结构框图，如图2.1所示。

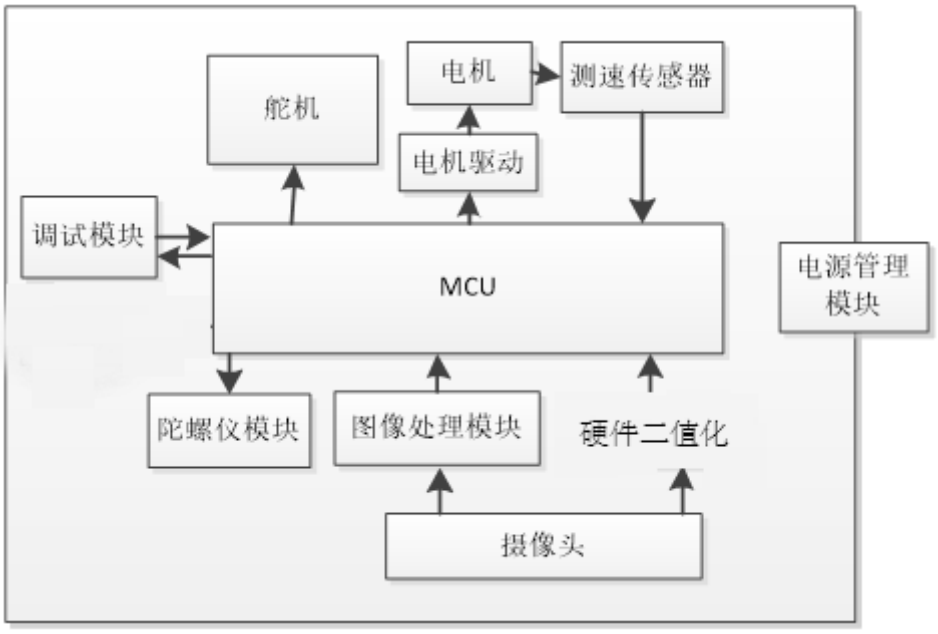


图 2.1　　硬件电路整体架构框图

2.2. 单片机外围电路设计

为了更好的配合程序，结合自己智能车的使用情况，我们自己设计了单片机外围，外围电路包括单片机时钟与复位电路，串口及供电电路，并且集成到主板上，以下是MCU部分的PCB图：

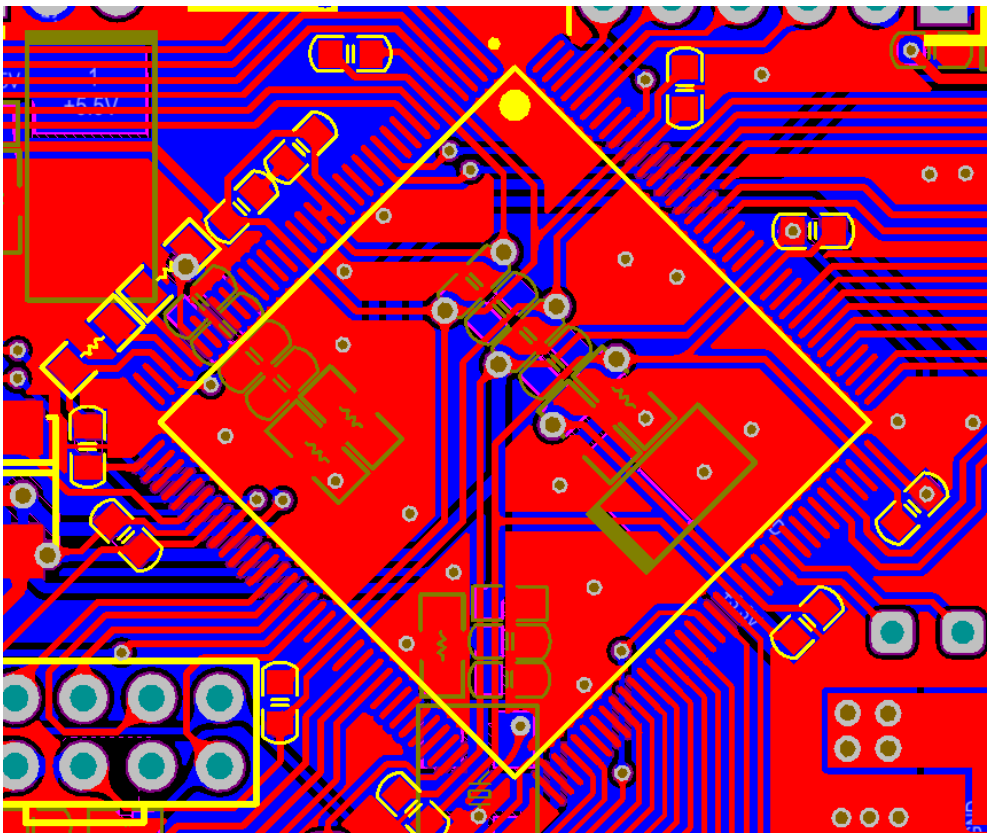


图 2.2 MCU PCB 图

2.3. 摄像头信号处理电路设计

本届我们采用ov7225鹰眼摄像头。经过硬件二值化，摄像头输出的数字信号，只需进行简单的信号处理相比模拟摄像头，外围电路更为简单可靠，如图2.3所示。

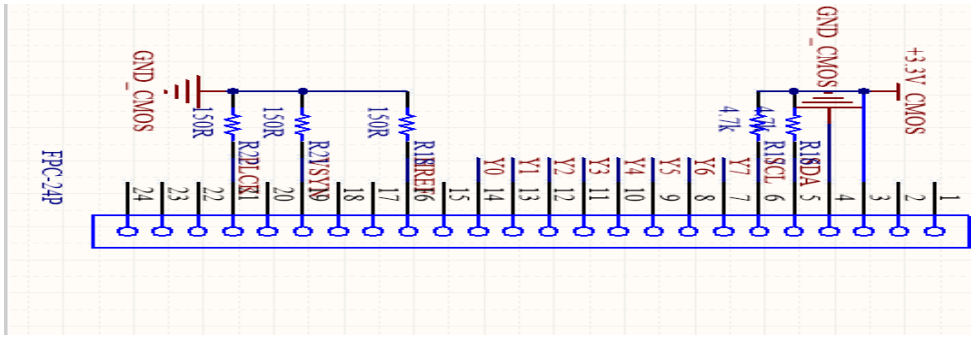


图 2.3 视频信号处理

2.4. 电源部分电路设计

电源分配 框图：

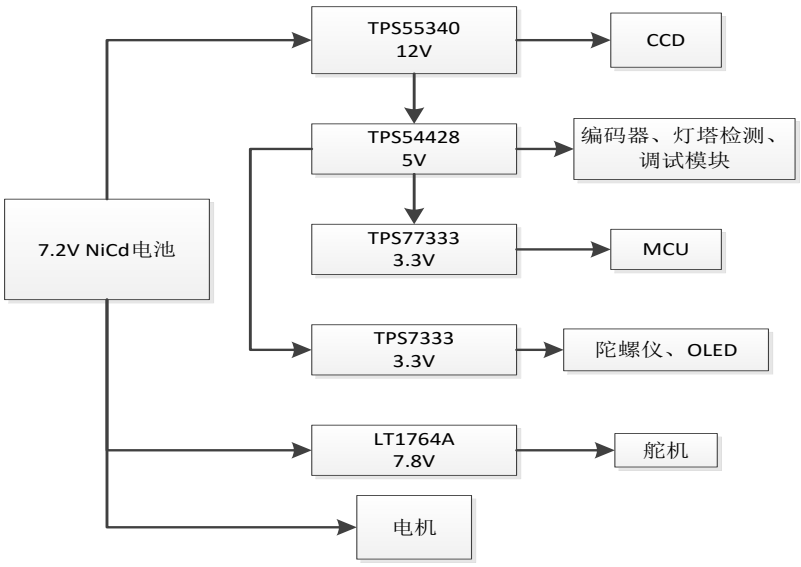


图 2.4.1 电源分配框图

整辆小车都是由 2000mAh 的镍镉电池提供，但是各种元器件所需的电压不同，这就需要我们用不同的电源芯片来生成不同的电压。

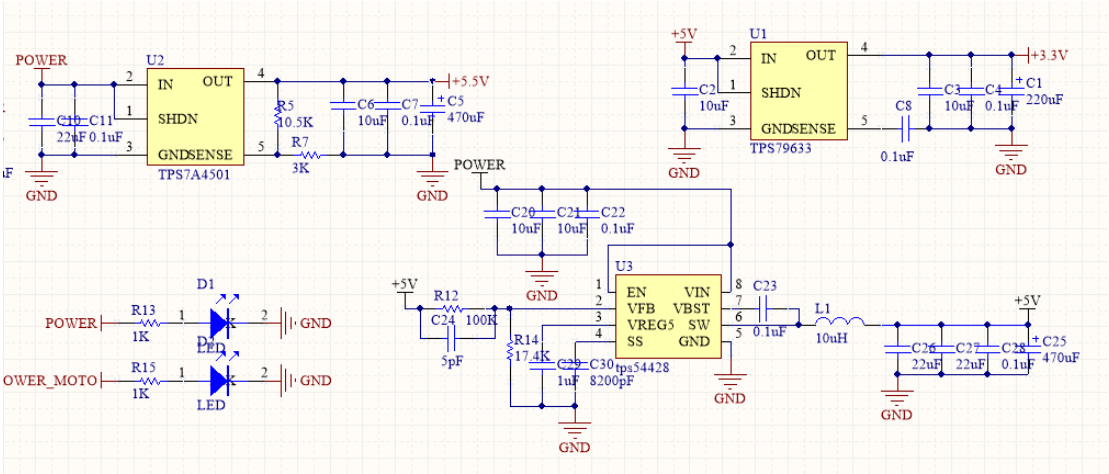


图 2.4.2 电源电路原理图

2.5. 电机驱动电路设计

电机驱动电路为一个由分立元件制作的直流电动机可逆双极型桥式驱动器，其功率元件由四支 N 沟道功率 MOSFET 管组成，额定工作电流可以轻易达到 100A 以上，大大提高了电动机的工作转矩和转速。该驱动器主要由以下部分组成：PWM 信号输入接口、逻辑换向电路、死区控制电路、电源电路、上桥臂功率 MOSFET 管栅极驱动电压泵升电路、功率 MOSFET 管栅极驱动电路、桥式功率驱动电路、缓冲保护电路等。

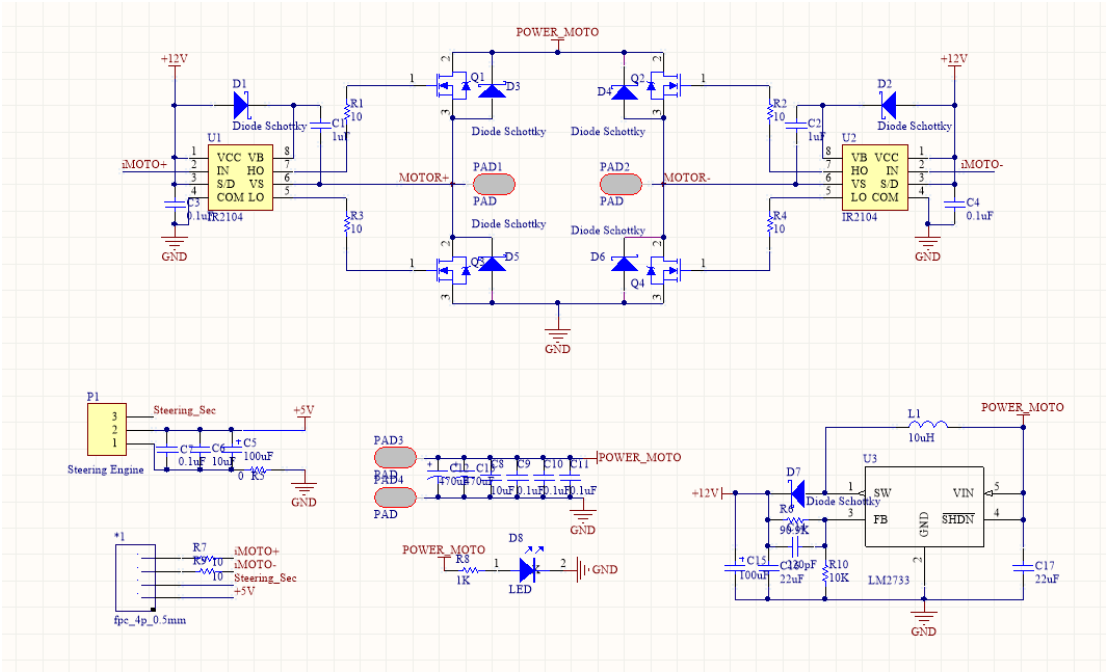


图 2.5 驱动原理图

2.6. 陀螺仪

陀螺仪在车上的作用是进行打角修正，根据陀螺仪传回的数据来进行打角值的补偿修正。

2.7. 小车调试模块

为使软件上参数调节的方便，且兼顾硬件上的简洁与美观，我们特意将这部分设计在主板上，而且使用最新的OLED显示模块来代替之前的NOKIA5110。OLED比NOKIA5110体积上更小，分辨率更高，功耗低。调试中，这个模块对参数的调试带来了极大的方便，可以减少烧录程序的次数提高调试效率。

2.8. ESD 防护

调试过程中发现小车数据口易受静电干扰复位、死机，往往需要用铝箔接地，加湿器加湿赛道环境。所以硬件设计上首先将主板固定螺丝接电池负端，然后将编码器等数据口加 TVS 管限幅，电机驱动数据口进行隔离。为了减小体积，设计上采用集成的 USB 口防护芯片保护数据口，隔离芯片采用高速率低延时的磁耦隔离芯片。

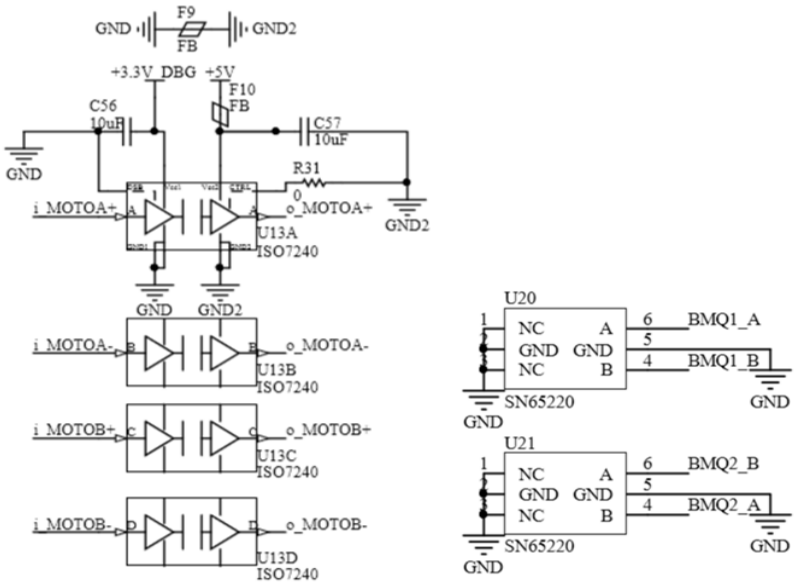


图 2.5　ESD 防护原理图

2.9. 主板板级设计

由于需要高速处理 CCD 的模拟信号，PCB 的设计影响信号的完整性，所以需要合理的设计元器件的布局及走线，阻抗匹配。特别注意模拟信号与高速数字信号、电感，晶振之间的布局。

设计的 PCB 如图 2.9 所示

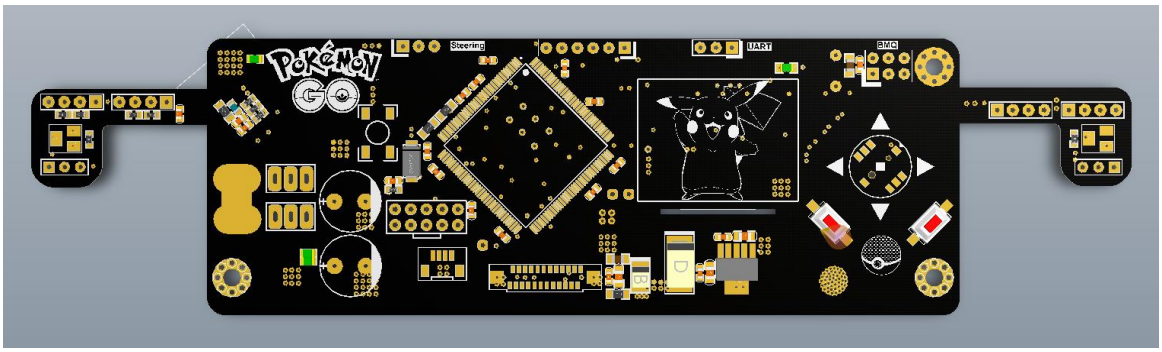


图 2.9　主板电路 PCB 图

2.10.硬件电路部分总结

对于硬件电路部分，一定要做的足够稳定，这对于整个系统的稳定性占着至关重要的位置。我们在实验过程中，先分模块根据经典应用设计制作电路，确保实现功能和稳定性后，再将所有模块组合起来绘制PCB，再次制作电路板验证电路系统的可靠性，最后打样制作电路板。

第三章. 机械架构调整

3.1. 摄像头的固定和安装

摄像头作为我们摄像头组最重要的传感器，它的固定和安装对小车的影响是十分巨大的，摄像头的布局和安装取决于系统方案，反过来又会影响系统的稳定性与可靠性以及软件的编写。

3.2. 摄像头固定与底座改装

由于本届车模为四轮车，所以摄像头架在车子的中间部分，介于电池和电路板之间，这样节省空间而且也不会让重心偏移太大，而摄像头的角度也很有讲究，角度低的时候能看到很远的赛道信息，但是图像较为模糊，不适合图像处理的编写，角度较高是，能看到的图像信息较少，但是分辨率明显更好，但因为某些特殊赛道的需要(如十字，障碍等)，我们还是将摄像头的前瞻保持在2.5米以上以求最有效的信息，在程序的编写中，我们发现摄像头视野的宽广往往直接影响赛道信息提取的精准度，在浙江省赛上的成功尝试，也证明了这种架法的优越性。

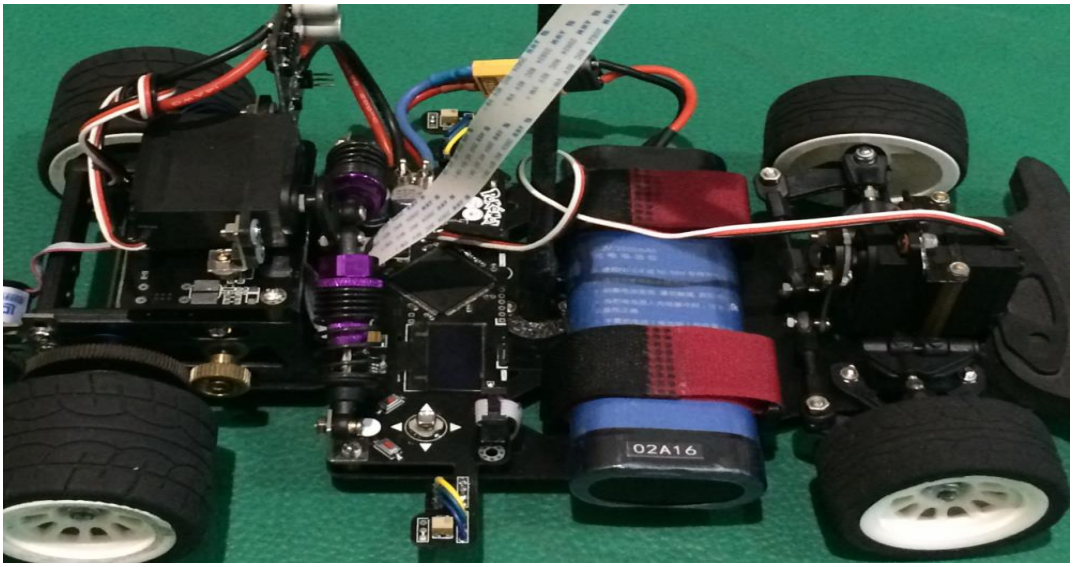


图 3.1 摄像头架法

3.3. 摄像头与支撑杆之间的安装

摄像头与支撑杆之间的安装能否稳定是整个采集图像是否可靠地重要原之一。为此我们也想了很多，最后我们采用了一种比较轻便的安装方法，其中碳棒为 6.0mm 的，但是直接用 6.0mm 的钻头的，打孔时加上机械抖动会放大孔径，所以我们采用了 5.9mm 的钻头进行打孔。这样碳棒卡的比较紧。另外固定螺丝的孔径为 3.0mm，但是这个孔径可以自己任意调节，只要便于固定，有适合的螺丝就可以。具体的固定如图 3.2 所示：

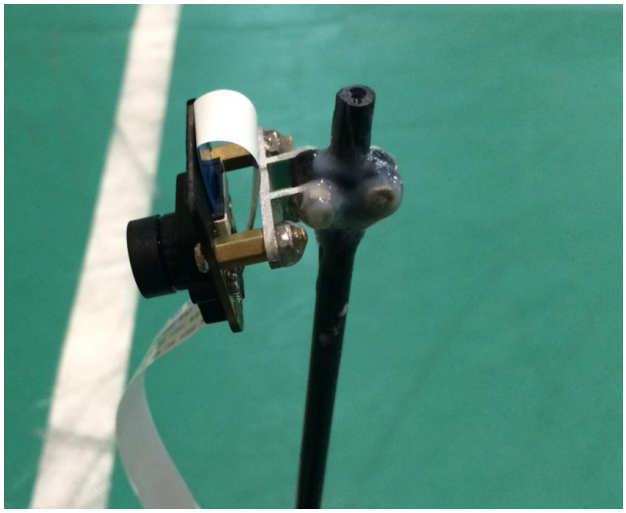


图 3.2 摄像头与碳棒之间固定

3.1.3 支撑杆与底盘之间的安装

除了摄像头与支撑杆之间的连接会影响摄像头采图之外，支撑杆与底盘之间的固定也同样重要，本届车模为四轮车，在车子跑动时，受到速度变化或者打脚影响，车子会有左右倾的状态，因此底盘需要特别的结实才可以的。具体的固定如图 3.3 所示。

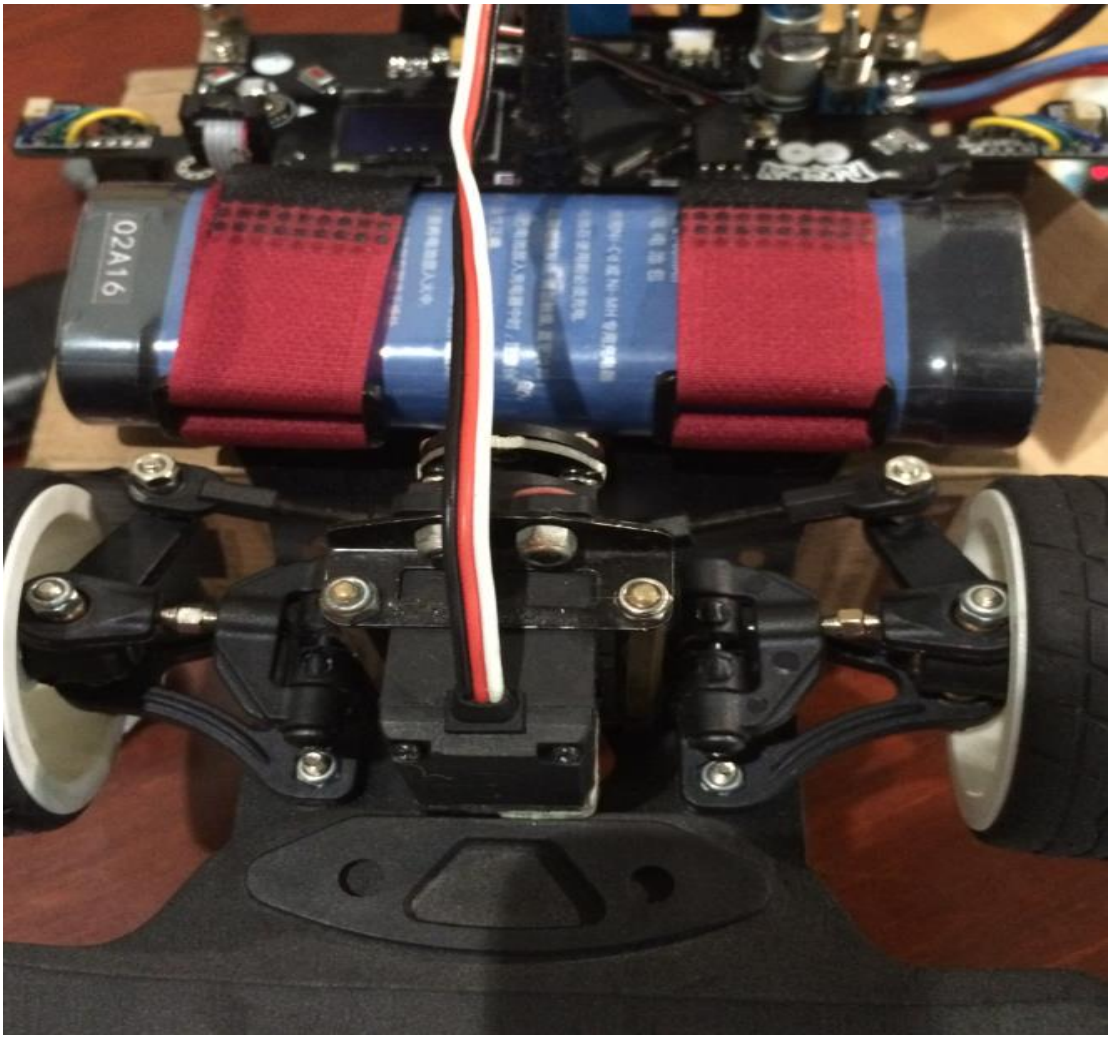


图 3.3 支撑杆与底盘

3.4. 编码器的安装

为使整个控制形成闭环，当前速度反馈是必须的。为了测出当前速度，我们用了两个个光电编码器。编码器的固定用的是钢质结构，足能稳定固定。在安装编码器的时候要保证有合适的齿轮咬合。咬合完美的原则是：两个传动齿轮轴保持平行，齿轮间的配合间隙要合适，过松容易打坏齿轮，过紧又会增加传动阻力；传动部分要轻松、顺畅，容易转动。判断齿轮传动是否调整好的一个依据是，听一下电机带动后轮空转时的声音。声音刺耳响亮，说明齿轮间的配合间隙过大，传动中有撞齿现象；声音闷而且有迟滞，则说明齿轮间的配合

间隙过小，咬合过紧，或者两齿轮轴不平行，电机负载加大。调整好的齿轮传动噪音小，并且不会有碰撞类的杂音。安装好的编码器如图 3.6 所示。固定钢片尺寸如图 3.7 所示

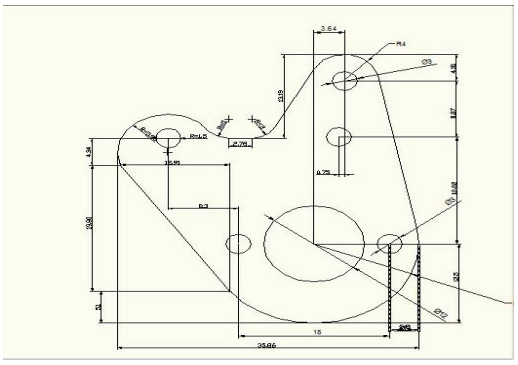


图 3.4 光电编码器固定钢片

3.5. 机械部分调整

3.5.1 前轮定位

为了使模型车在直线行驶时更加稳定，转向轻便，转向后能自动回正，并减少轮胎和转向系零件的磨损等，在转向轮、转向节和前轴之间形成一定的相对安装位置，叫车轮定位。其中包括主销后倾、主销内倾、车轮外倾和前束。接下来主要介绍主销后倾、主销内倾、前轮外倾和前束对车子行驶的影响及调教。

3.5.2 主销后倾

主销在车模的纵向平面内（汽车的侧面）有一个向后的倾角 γ，即主销线与地面垂直线在车模在纵向平面的夹角，称为“主销后倾角”，车模采用主销后倾的原因是由于汽车在车轮偏转后会产生一回正力矩，纠正车轮偏转。车模的主销后倾不可直接调教，在此我们采用预设的主销后倾角，大概 3 度左右。如图 3.5.2 所示

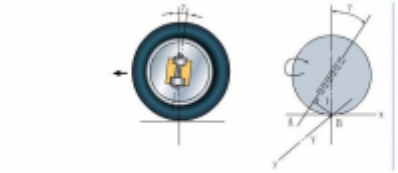


图 3.5.2 主销后倾

3.6. 小车轮胎的定位调整

3.6.1 前轮外倾角

前轮外倾角：转向轮上端略向外倾斜叫前轮外倾角。我们今年所使用的 B 车模不可调整，不多赘述。

3.6.2前轮前束

前束是转向灵敏度与稳定性的权衡。前束不可以无限度增大（向左侧调整）， 太大了的话，直道行驶时车轮与地面发生的就不是滚动摩擦，而是滑动摩擦， 轮胎磨损将急剧增大，且会导致阻力加大，降低直道速度。过度减小（向右侧调整）会导致稳定性降低，车辆抖动，难以操控。如图3.6.2示例所示。

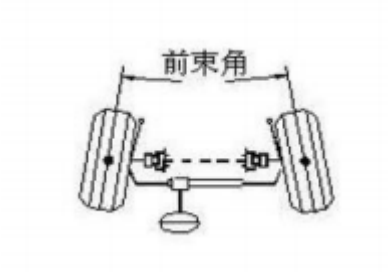


图3.6.2 前轮前束

3.3.2 小车差速的调整

小车采用的差速器为滚珠式差速器。合适的差速器调整能够提高小车入弯速度，提高弯道性能。差速器调整可以通过右后轮轮轴上面的螺丝。注意调整 过松，会严重影响直道加速性能；调整过紧则会使差速器处于无效状态。差速器滚珠处可以适当添加润滑剂，保证差速器平滑。

3.7. 舵机的安装

舵机安装直接关系到转向问题。如果舵机调整不到位，将很大程度上限制转向角度和转向响应速度。舵机安装有两种方式，一种是卧式安装，另外一种为立式安装。卧式安装为车模默认安装方式，但这样安装会使左右两边轮子连杆不等长，根据杠杆原 理可知舵机对长连杆轮子用的力要大些，因此
造成了舵机对左右两边转向响应 时间不一样。另外由于卧式安装会使连杆与水平面呈现一定角度，从力学知识 可以知道在轮子转向获得的力只是舵机施加在连杆上力的一个水平方向上的分 力。综合考虑，我们选择了舵机立式安装方式。舵机立式安装能够解决上述卧式安装的缺点，即连杆等长和连杆与水平面夹角小的问题。立式安装需要将自制或者购买舵机与连杆之间的连接器，我们从网上购买舵机摆摆臂。舵机安装高度则需要多次实验确定。舵机安装效果图如图3.6 所示。

图3.6 舵机的安装

上面我们已经介绍了车模的基本几个重要的机械架构部分。对于一辆架构优良的车来说，除了考虑以上几个部分还需要考虑一些细节，同时也要对车模的整体架构进行微小调整和重量的匹配。车模的重量均匀的分布在车模的四个轮子上对车模整体的平衡稳定都是有非常大的意义的。同时这也在一定程度上影响了左右的打角。另外要注意的是车模的底盘高度和重心问题，实践证明降低底盘高度，降低整个车模的整体的重心对转弯是有很大帮助的。但是同时也要考虑到车模是否过坡是檫到赛道。总之，具体的情况依车模而定。下图是我

们车模的整体结构和布局。

图3.7 车模侧面

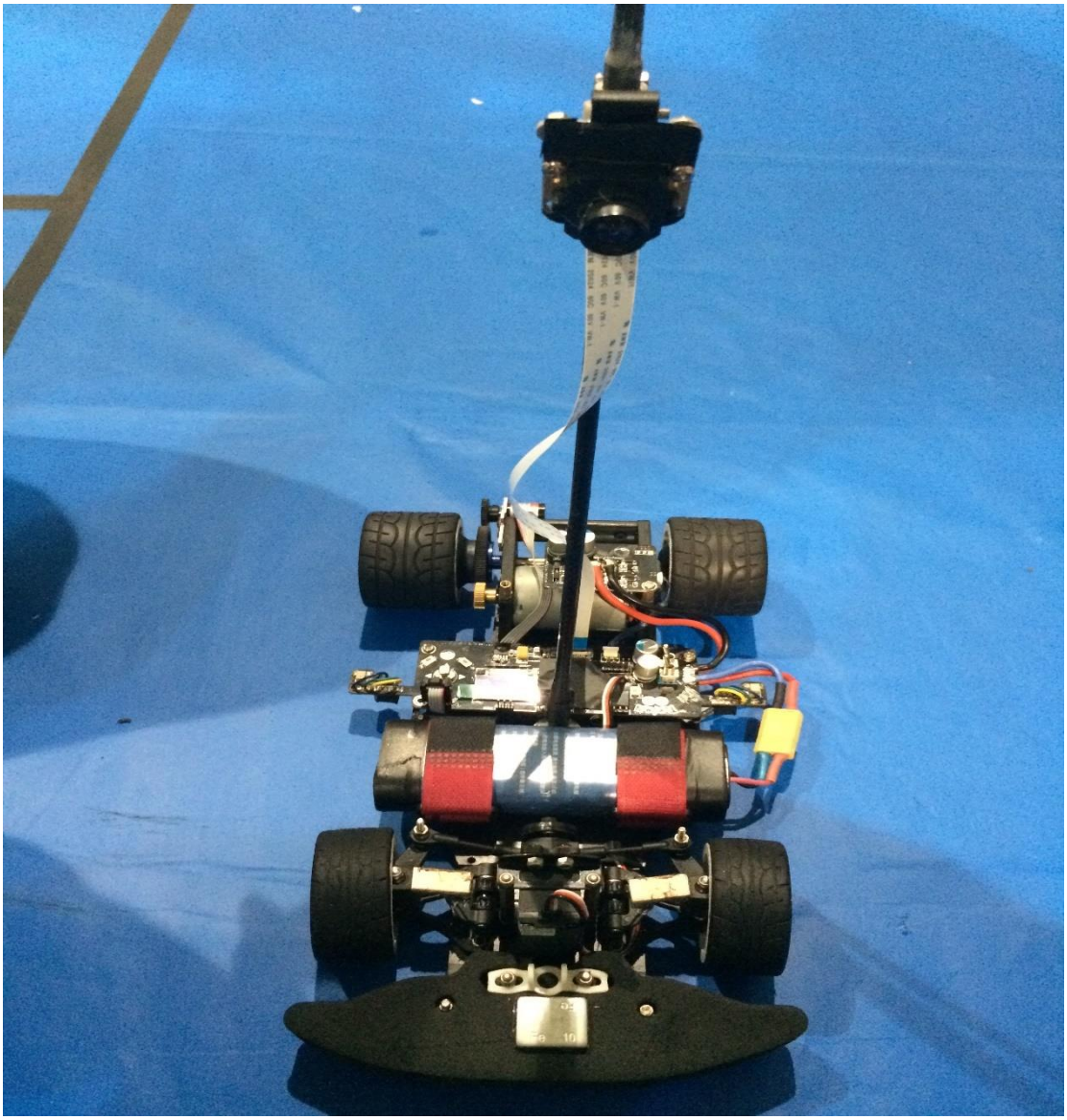


图3.7 车模正面

我们现在的车模结构总体均衡，构架轻巧，结构牢固。每一小块机械结构都是经过自己亲手设计打造。为这次比赛，我们不知换过多少方案，架构过多少次车模，机械结构是改过又改过，车模在我们手上是架了拆，拆了又架。

第四章. 软件系统设计

4.1. 软件控制程序的整体思路

软件的控制，就是让控制车模在符合大赛规则前提下，以最快的速度跑完整个赛道。不论上哪种方案，软件的总体框架总是相似的，我们追求的就是稳定至上，兼顾速度，以我们的方案，软件上就是图像采集、图像处理识别黑线、平衡控制，速度控制以及速度反馈。

4.2. 图像采集

图像采集部分主要有鹰眼摄像头。摄像头主要由镜头、CCD 图像传感器、图像信号形成电路、同步信号发生器，硬件二值化等外围电路组成。

4.3. 摄像头的工作原理

我们比赛用的摄像头主要分为黑白和彩色两种。彩色摄像头对比度比黑白的要好，对赛道不同背景色有较好的抗干扰能力，但考虑到比赛时我们只关注黑线的提取而不关心图像彩色信息，所以我们只选用硬件黑白摄像头。接下来，我们用眼睛来打比方以说明成像原理：当光线照射景物，景物上的光线反射通过人的晶状体聚焦，在视网膜上形成图像，之后视网膜的神经将信息传导到大脑，我们就能看见东西了。摄像头成像的原理和眼睛非常相似，光线照射景物，景物上的光线反射通过镜头聚焦， CCD 图像传感器就会感知到图像。而感知图像的具体过程是这样的：摄像头按一定的分辨率，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之一一对应的电压值，然后将此电压值通过二值化芯片比较器输出。摄像头使用的是山外的鹰眼，摄像头信号波形如图4.1和4.2所示。

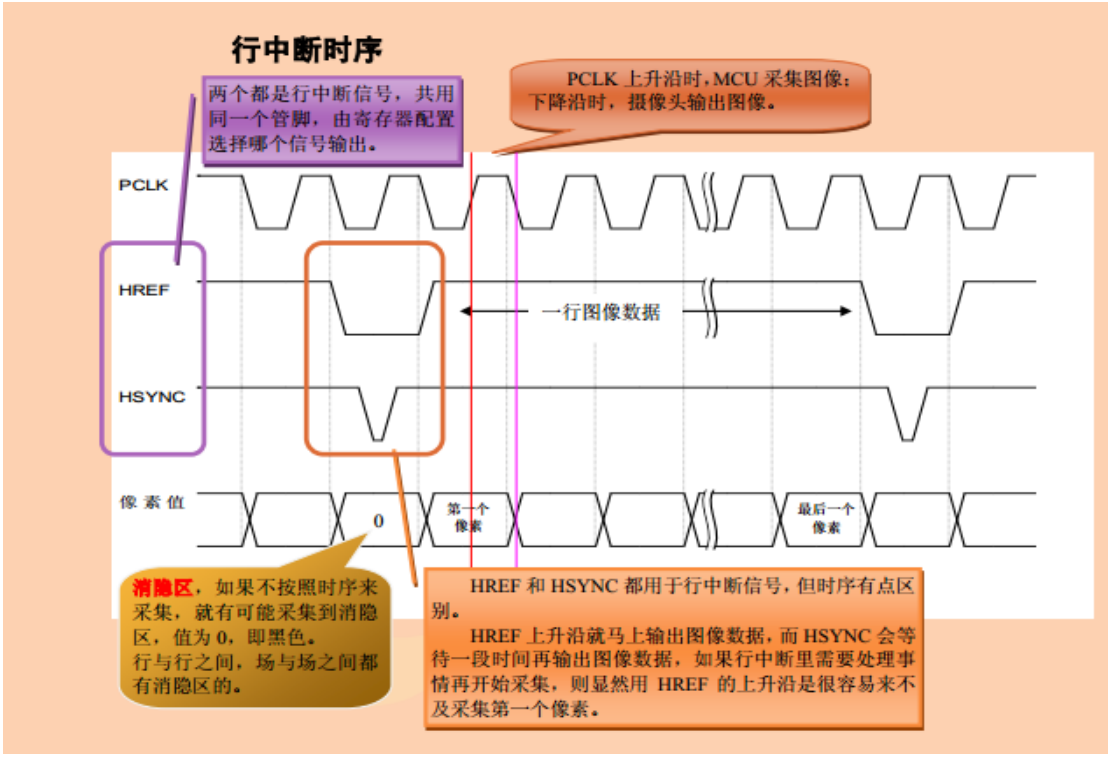


图4.1 摄像头行信号波形

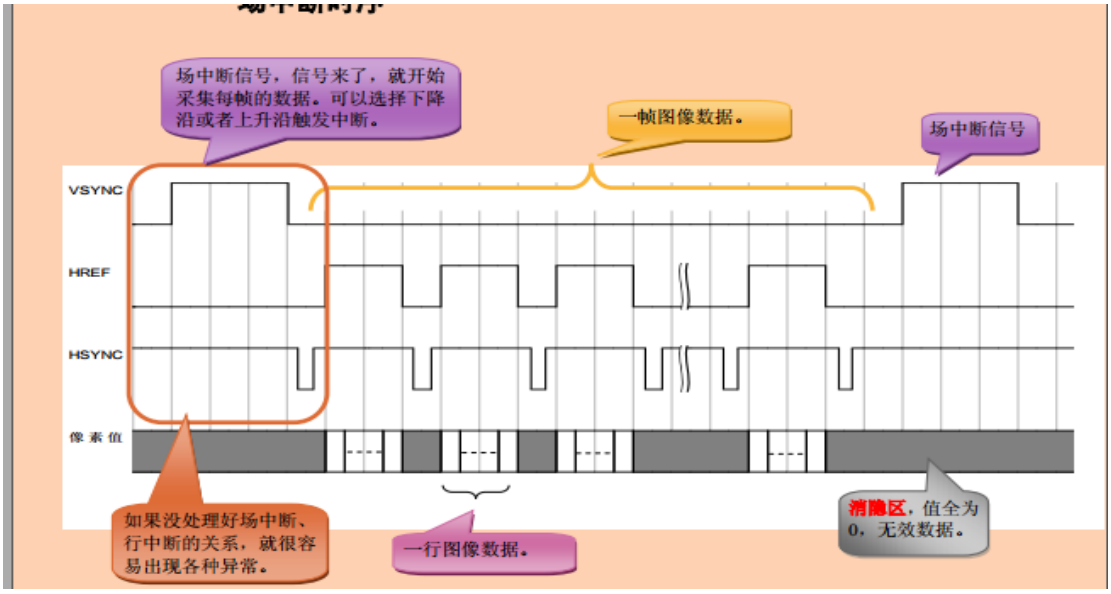


图4.1 摄像头场信号波形

4.3.2 选用0V7725理由

ov7725 信噪比更高、速度更快、稳定性更好和微光灵敏度更高、 绝非 ov7620、 ov7670、ov6620 这类可比， ov 系列三十万像素当中成像质量最棒的摄像头，是智能车比赛的最佳选I2C 协议 + 读操作时停止条件 = SCCB 协议兼容 I2C 协议，可直接用 I2C 模块来控制……

低照度好（即微光灵敏度高） 有什么好处？

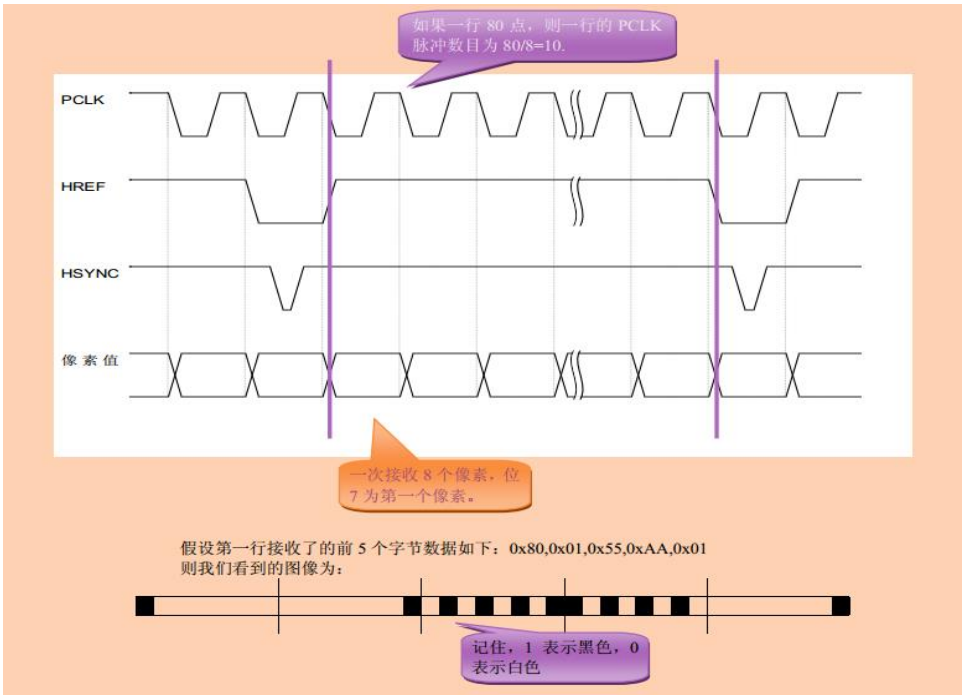
正如我们平时拍摄高速运动中物体的图片一样，高速奔跑中的小车拍摄的图像非常容易变模糊的，目前智能车比赛所用的摄像头中，没一个卖家的摄像头集效果图片是晃动中拍摄的， 原因你懂的。

如何克服拍摄运动图像变模糊问题呢？

没错，减少曝光时间……

4.2.3 野火鹰眼摄像头时序

野火鹰眼，时序与 ov7725 完全一样的，唯一不同的是数据格式



4.2.4 黑线提取

图像处理简单的来说就是根据摄像头传回来的视频信号中提取出黑线的置。常用的黑线提取算法划分为二值化算法、直接边缘检测算法和跟踪边缘检算法二值化由于是硬件二值化，不做概述。直接边缘检测算法：采用逐行搜索的算法，首先找到从白色像素到黑色像素的下降沿和从黑色像素到白色像素的上升沿， 然后计算上升沿和下降沿的位置差，如果大于一定的标准值，即认为找到了黑线， 并可求平均值算出黑线的中心点。至于上升沿、下降沿的检测，可以通过上上

次采样数与这次采样数的差值的绝对值是否大于一个阈值来判断，如果“是 ”且差值为负，则为上升沿；如果“是”且差值为正，则为下降沿。跟踪边缘检测算法：由于黑色的目标引导线是连续曲线，所以相邻两行的左边缘点比较靠近。跟踪边缘检测正是利用了这一特性，对直接边缘检测进行了简化。其思路是若已寻找到某行的左边缘，则下一次就在上一个左边缘附近进行搜寻。这种方法的特点是始终跟踪每行左边缘的附近，去寻找下一列的左边缘，所以称为“跟踪”边缘检测算法。

我们采用的是直接边缘检测算法，因为该方法抗环境光强变化干扰的能力更强，同时还能消除垂直交叉黑色引导线的干扰。由于智能车上安装的摄像头相对于赛道存在一定的倾斜角度，因此会造成采集到的赛道图像具有一定的梯形失真，即图像中的赛道远端窄、近端宽，远端图像不清晰而近端图像清晰可靠。所以就将一场图像分为两部分，近端部分和远端部分

4.2.5 赛道中心拟合

因为是双边的黑线赛道，所以我们在正确提取黑线信息的基础上，外加赛道中心拟合函数模块，其具体思想为“两点求中线”。下图为上位机上的处理

效果图

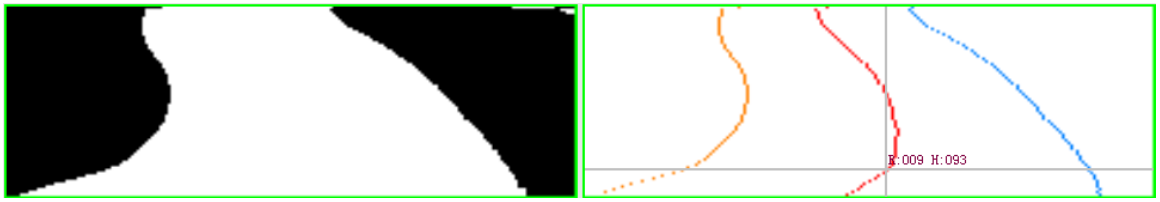


图.4 赛道中心拟合图

4.4. 舵机打角控制

我们采用的曲率打角，只根据光电式编码器反馈来的脉冲数转化为速度再叠加电机的相对反应时间，然后根据这打角行位置与图像中心位置的偏差通过一个二次函数来计算出所需打脚力度。图 4.5 为电机打角曲线图。

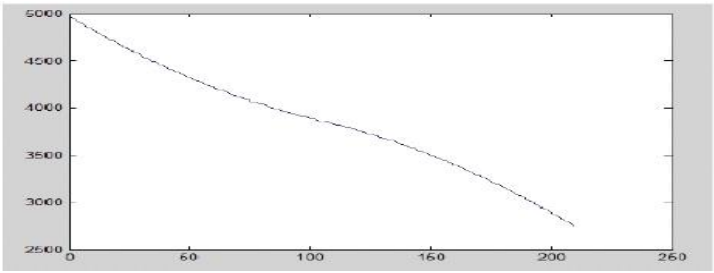


图 4.5 赛道中心拟合图

但是在后期调试中我们发现取车身前一段范围内黑线平均位置来确定打角值这样赛车走的路径会得到很大程度上的优化。

4.5. 速度控制

速度控制策略的好坏就决定小车能否取得优秀成绩，如何给速度是关键，

什么样的赛道配合什么样的速度。比如过弯时要提前减速使赛车在入弯前达到

一个安全速度以免冲出赛道，过 S 弯时要同一般弯道和直道区分开，加减速控制就是让车模以最快的反应速度达到标定的速度，反应越快越好。程序每进入一次速度控制函数，首先读出于编码器连接的脉冲累加器计数的值，计算出当前速度，再与标定的速度比较，如果当前速度大于标定的速度再根据两速

度的差值决定减速的力度；如果当前速度小于标定的速度，就根据两速度差值决定加速的急缓。减速就是递减电机正转的 PWM 值，如果速度差很大，便要启动电机反转刹车以迅速达到减速的目的。加速的过程是对电机正转的过程，这个过程对电机的驱动的承受能力是个很大的考验，而且它是 PWM 累加的过程，但是，要对 PWM 限定范围，加速的急缓只是每次累加的量大小不同。

经过实际测试我们将速度给定处理成二次曲线，这样达到的实际效果还是比较理想的。

代码如下：

Speed_give=Speed_min+(Effective_Rows-40)*(Effective_Rows-40)*(Speed_max-Speed_min)/((57-40)*(57-40));

为了使赛车能快速达到给定速度，我们一开始采用的是 PID 控制，代码如下：

actual_given_PID=Kp*iError*7/10-Ki*LastError*7/10+Kd*PrevError;

在经典 PID 控制中，给定值与测量值进行比较，得出偏差 e（t），并依据偏差情况，给出控制作用 u（t）。对连续时间类型，PID 控制方程的标准形式为，u（t）= Kp ［ e（t）+ ∫e（t）dtTI+ TDde（t）dt］。式中，u（t）为 PID 控制器的输出，与执行器的位置相对应；t 为采样时间；KP 为控制器的比例增益；e（t）为 PID 控制器的偏差输入，即给定值与测量值之差；TI 为控制器的积分时间常数；TD 为控制器的微分时间常数离散 PID 控制的形式为 u（k）= KP｛e（k）+Σe（k）T/TI+［e（k）-e（k- 1）］TD/T｝。式中，u（k）为第 k 次

采样时控制器的输出；k 为采样序号，k =0，1，2 ；e（k）为第 k 次采样时的偏差值；T 为采样周期；e（k - 1）为第（ k- 1）次采样时的偏差值。

从系统的稳定性、响应速度、超调量和稳态精度等方面来考虑， KP、KI、

KD 对系统的作用如下。

①系数 KP 的作用是加快系统的响应速度，提高系统的调节精度。KP 越大，系统的响应速度越快，系统的调节精度越高，但易产生超调，甚至导致系统不稳定；KP 过小，则会降低调节精度，使响应速度缓慢，从而延长调节时间，使系统静态、动态特性变坏。

②积分系数 KI 的作用是消除系统的稳态误差。KI 越大，系统的稳态误差消除越快，但 KI 过大，在响应过程的初期会产生积分饱和现象，从而引起应过程的较大超调若 KI 过小将使系统稳态误差难以消除影响系统的调③微分作用系数 KD 的作用是改善系统的动态特性。其作用主要是能反应偏差信号的变化趋势，并能在偏差信号值变的太大之前，在系统引入一个有效的早期修正信号，从而加快系统的动作速度，减少调节时间。图 4.6 是 PID 控制系统仿真结果。

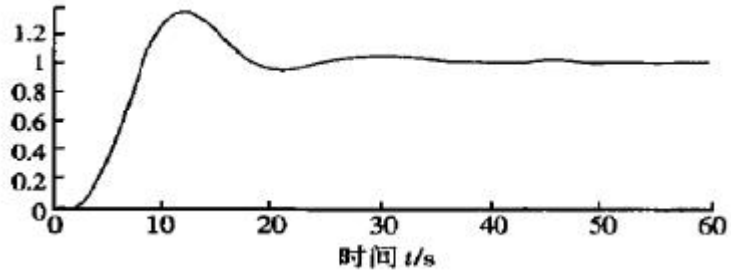


图 4.7 系统仿真结果

经过很长时间的测试，我们最终采用了增量式 PID，因为通过实验证明这样能让电机有较好的加减速能力。

4.6. 软件程序总结

软件部分是整个控制系统的核心。软件上主要有以下几个难点：

1) 怎样判断采样回来的赛道信息的有效性；

2) 怎样使速度控制配合好打角。

对于本车模的软件系统，我们充分发挥后架车的优势，超远的大前瞻和广

阔的视角带给我们不少创作灵感。对于车模来说，软件控制是核心，而对于软件来说，打角和速度控制都不算是核心，真正的核心应该是打角和速度控制的相互配合！本车模软件系统对速度的控制均以 PID 控制技术为原理，适当发挥再加以利用。

第五章. 智能车调试说明

前面几章一直是在为系统制定方案以及方案的细化。但整个系统的完善主要还是在系统的现场调试。在细分的每个模块中，大部分都涉及有众多参数，对这些参数的确定就需要软硬件联合调试。而这过程就需要一整套开发调试环境已工具。包括程序源代码的编辑以编译环境，参数调节与设定工具。

5.1. 上位机调试软件的设计

我们的上位机是本届完全自主开发的，图 5.1 是上位机软件界面。

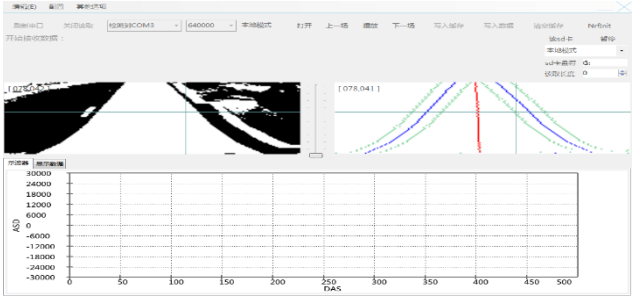


图5.1 上位机1

仿真模式：由于图像处理越来越复杂，对于图像仿真越来越重要，PC 仿真模式可以将单片机代码导入，更加直观看到图像处理结果。

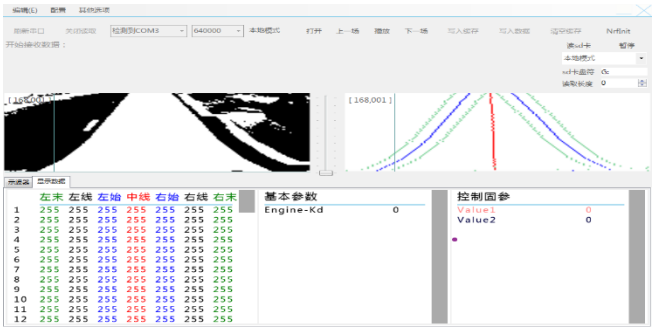


图5.2 上位机2

本地模式：小车的实际图像反馈对于调车非常重要，利用 SD 卡将小车行进中的图像以及处理信息保存，上传到上位机，以便常看。

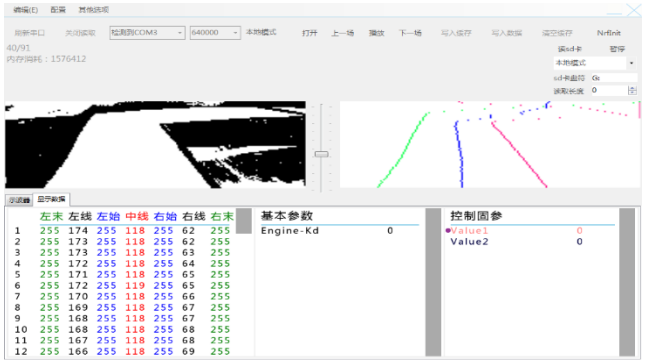


图 5.3 上位机 3

调试模式：为了提高调试效率，特别为上位机做了在线调试系统，信息一目了然。

5.2. Flash 存储模块

因为摄像头车模所需处理的信息量非常大，在调试过程中往往需要存储这些信息以供研究分析，而 K60 的存储空间远不能满足我们的要求。所以我们设计了外部 SD 卡来存储海量信息，这让我们在图像处理上有很多的发挥空间。通过 SD 卡，我们存下了我们需要的图像信息和处理之后的信息，然后通过串口发送给上位机，这样我们就可以方便地分析出图像上的各种问题，以便在程序上作出修改。

5.3. 现场调试

在完成基本程序之后，做的最多的工作就是调试，调试是一持续性的工作。之前说过，我们为调试的方便，特别制作了按键加液晶模块。写程序的时候，

自己想好的一些参数在静态测试的时候可能很好，但是当车在赛道上跑的时候，情况就不一样了。所以，车模必须在赛道上多跑以发现问题进而解决问题。车模主板上的OLED和按键就为这种现场调试提供了很好地硬件平台，软件的同学在修改某个参数的时候就无需跑去电脑边上下载了，非常的方便。

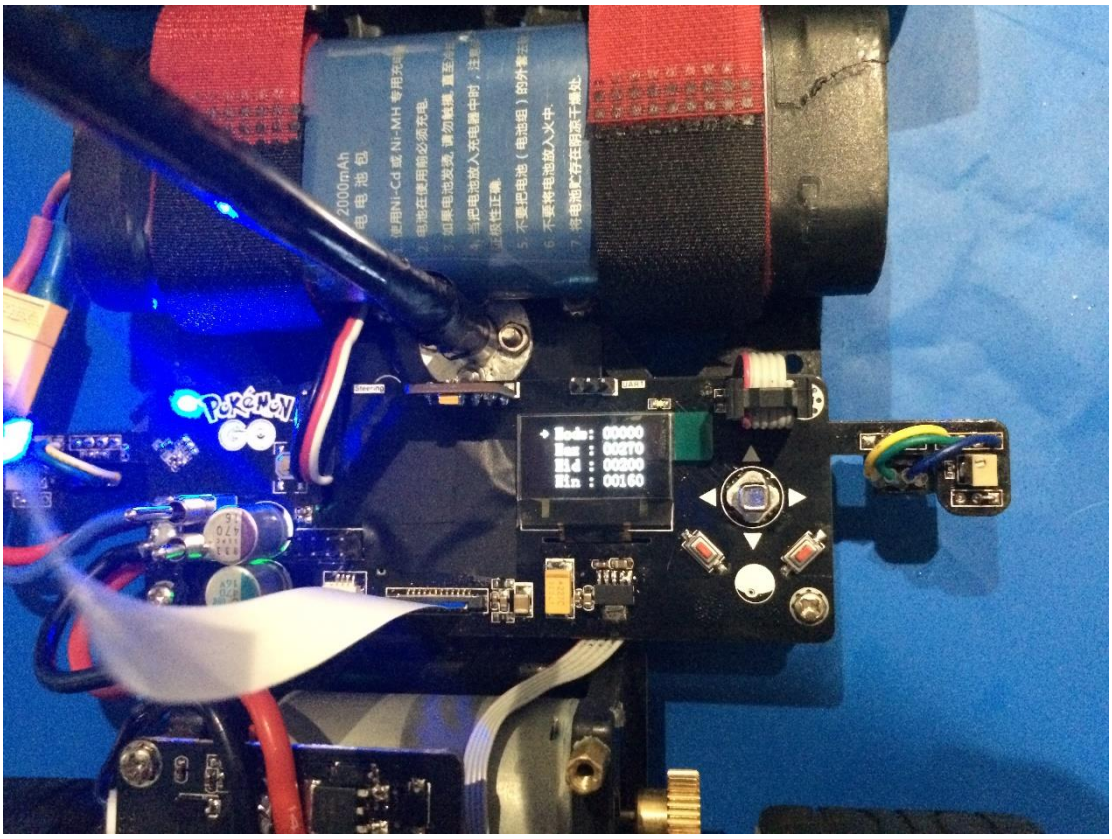


图 5.4 OLED

第六章. 车模技术参数说明

6.1. 车模主要技术参数

名称	参数
车模总质量	950g
车模长度	309mm
车模宽度	200mm
车模高度	40mm

电路电容器总容量	1683.71uF
传感器种类及个数	编码器1个， 鹰眼摄像头1 个，陀螺仪1个，光电对管2个
外加伺服电机个数	0 个
赛道信息检测精度	22.5mm
赛道信息检测频度	75（次 / 秒）

表 6.1 车模部分主要技术.参数

6. 2. 电路中芯片的种类及数量

芯片名称	使用数量	功能简述
MK60DN512ZVLQ10	1	微控制器
TPS54428	1	开关电源芯片
TPS79633	1	线性稳压器
TPS7a4501	1	线性稳压器
IR2104	2	栅极驱动器
LR7843	4	MOS管
Lm2733	1	开关电源芯片
tpd4e1u06	1	四通道ESD防护芯片
sn65220	2	USB瞬态抑制芯片

表 6.2 电路中芯片的种类及数量

第七章. 总结

本文介绍了摄像头智能小车的总体设计方案，从机械布局、架构设计，硬件系统设计和软件设计三个部分详细介绍了我们设计小车的方案和一些创新。在基本的设计流程中，我们也进行了不断地尝试，并且取得了较理想的成果。车模的设计要综合考虑各个方面的问题和可能遇到的问题。在设计的时候要胆大心细，求实创新。飞思卡尔举办到现在已经是第十届了。在技术和经验上都是比较成熟了，尤其是摄像头，比赛的差距更是非常小。想要在比赛中取得好成绩就要在平时注重机械、硬件、软件的综合配合调节和优化。而这一届我们的小车在软件、硬件和机械上都有了一些新的突破。摄像头的架构安装，与以前相比更加稳固，而且拆装方便；底盘高度的调节，PCB 板配合车模形状，液晶贴底，降低底盘高度的同时使整个车身的重量全部下降，相比以前，整个车身的重心要远远下降，这样在弯道就有明显的优势；芯片选型也较好，PCB 布局布线贴合车模车身设计，各个模块设计布线也是相对优化的。软件设计部分，相对以前已经有了的基础之上有了较大的优化。通过图像处理，速度反馈和速度控制，有效地让小车进行循迹。

第八章. 致谢

在为本次大赛制作智能车期间，我们遇到过很多问题，从最初的传感器选型与方案确定，到后来的软硬件联合调试。在解决一个个问题之后，我们发现，

我们技术上在不断成长，思想上不断成熟。而在这过程中，离不开学校，老和同学的支持。首先，我们要感谢学校对这次比赛的重视，感谢学校教务处对我们比赛的大力支持。没有他们的支持，我们的车模绝对上不了赛场。其次，我们要感谢指导老师的悉心教导。没有他们在思想上的指导和整体的规划安排，我们很难有今天的成果。最后，我们要感谢同实验室的其他同学，感谢我们同时工作在这么和谐的实验室。

第九章. 参考文献

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

[2] 邵贝贝. 单片机嵌入式应用的在线开发方法. 北京-清华大学出版社 2004年 10 月第 1 版

[3] 卓晴，黄开胜，邵贝贝 学做智能车 北京-北京航空航天大学出版社 2007

[4] 王威 HCS12 微控制器原理及应用 北京-北京航空航天大学出版社 2007

[5] 李宁，刘启新 电机自动控制系统 北京-机械工业出版社，2003

[6] 潘松，黄继业 现代数字电路基础教程 北京-科学出版社 2008

[7] 魏彪，盛新志 激光原理及应用 重庆-重庆大学出版社 2007

[8]科大中冶队第四届“飞思卡尔”杯全国大学生智能车大赛技术报告北京科技大学 2009

[9] 赵先奎 汽车前轮定位 黑龙江 黑龙江出入境检验检疫局 2008

[10]刘锐宁 Visual C++编程之道 人民邮电出版社 2011

[11] 赵先奎 汽车前轮定位 黑龙江 黑龙江出入境检验检疫局 2008

[12]姚领田 精通 MFC 程序设计 人民邮电出版社 2006

[13]刘锐宁 Visual C++编程之道 人民邮电出版社 2011

附录 A

小车程序：

```
int main(void)
{
    SystemClockSetup(ClockSource_EX50M,CoreClock_100M);
    NVIC_Rank();
    OLED_Init();
    Camera_GPIOInit();
    Camera_DMAInit();
    Motor_Init();
    Encoder_Init();
    Time_Init();
    Engine_Init();
    Menu();
    FTM_PWM_ChangeDuty(MotorAZ,3500);
    FTM_PWM_ChangeDuty(MotorBZ,3500);
    //图像传输初始化
    //UART_Send_Init();
    //Image_Send_DMA_Init();
    //虚拟示波器传输数据初始化
    //Oscilloscope_Init();
    //Oscilloscope_PIT_Init();
    //上位机传图初始化
    Data_Uart_Init();
    if((Mode == 2) || (Mode == 1))
    {
        //SD 卡初始化
        MySD_Init();
    }
    if(Mode == 2)
```

```
        {
            //SD 读取函数
            SD_Read();
        }
        if(Mode == 1)
        {
            //SD 多块写入模式初始化
            SD_ManyWT_Init();
        }

        while(1)
        {
            InformationConvert();

            InformationDeal();

            ControlAdjustment();
        }
    }
    /********************摄像头的 D 图像采集场中断函数（A26）**********************/
    void PORTA_IRQHandler(void)
    {

        if(GPIO_GetITStates ( PTA, GPIO_Pin_26 ))//场中断
        {
            GPIO_ClearITPendingBit(PTA,GPIO_Pin_26);  //清除场中断标志位

            DMA_SetEnableReq(DMA_CH0,ENABLE);        //打开 DMA0 传输
```

```
            Set_Camera_DMA_DAddr((uint32_t)date); //重设地址
        }



}


/*******************按键中断函数（C）**********************/
void PORTC_IRQHandler(void)       //
{
    IT_KeyScan();//中断按键检测
}



void PIT0_IRQHandler(void)
{


  //GPIO_SetBits(FMQ_PORT,LED_PIN);

//      if(Stop_flag != 1)
//      {
////            LED1 =   ADC_GetConversionValue(ADC1_SE17_PA17);
////            LED2 =   ADC_GetConversionValue(ADC0_SE18_E25);
////            LED3 =   ADC_GetConversionValue(ADC0_SE15_PC1);
////            LED4 =   ADC_GetConversionValue(ADC0_SE1_DP1);
//      }
```

```
        if(Ramp_flag == 0 && Obstacle_location == 0 && Time_count >= Stop_delay && Stop_flag != 1)
        {
            //if((LED1 >= 20||LED2 >= 20)&&(LED3 >= 20||LED4 >= 20))
            if((GPIO_ReadInputDataBit(PTD,GPIO_Pin_12) == 0||GPIO_ReadInputDataBit(PTD,GPIO_Pin_13) == 0)&&(GPIO_ReadInputDataBit(PTD,GPIO_Pin_11) == 0||GPIO_ReadInputDataBit(PTD,GPIO_Pin_10) == 0))
            {
                LED_IRQ = 1;
                Stop_flag = 1;
            }
        }

    //GPIO_ResetBits(FMQ_PORT,LED_PIN);


    PIT_ClearITPendingBit(PIT0,PIT_IT_TIF);//重置
}


void PIT1_IRQHandler(void)
{
        Time_count++;

    if(Time_count== Time_Scanf_count)
    {
        Time_count= 0;

        Stop_flag = 1;
    }
```

```
    PIT_ClearITPendingBit(PIT1,PIT_IT_TIF);//重置
}
uint32_t Data_SD_RCA = 0;
uint64_t Data_Count=NULL;//数据块长度

uint16_t Blocks_Num=0;//扇区标记地址

uint8_t Data_SD_Init(void)
{
    uint32_t delay_cnt = 0;
    uint8_t result;   //存储函数返回结果
    uint8_t i=0;
    uint8_t hc=0;        //是否为 SDHC 标志
    //SD 卡命令结构体定义
    SD_CommandTypeDef SD_CommandStruct1;
    //SD 卡参数设置结构体定义
    SD_InitTypeDef SD_InitStruct;

    SD_InitStruct.SD_BaudRate=SD_WR_BaudRate;//设置波特率

    //开启 GPIO 时钟
    SIM->SCGC5 |= SIM_SCGC5_PORTE_MASK;
    //开启 SDHC 时钟
    SIM->SCGC3 |= SIM_SCGC3_SDHC_MASK;
    //复位 SDHC  设置超时时间
    SDHC->SYSCTL = SDHC_SYSCTL_RSTA_MASK |
SDHC_SYSCTL_SDCLKFS(0x80); //超时时间设置为 2^13   //2 分频
    //等待复位完成
    while(SDHC->SYSCTL & SDHC_SYSCTL_RSTA_MASK){};
    //初始化 SDHC 相关寄存器
```

SDHC->VENDOR = 0;

SDHC->BLKATTR = SDHC_BLKATTR_BLKCNT(1) | SDHC_BLKATTR_BLKSIZE(512); //默认传输 1 个 Block 每个 Block 512 字节

SDHC->PROCTL = SDHC_PROCTL_EMODE(2)| SDHC_PROCTL_D3CD_MASK; //LSB 格式 使用 DATA3 检测卡插入移除

SDHC->WML = SDHC_WML_RDWML(1) | SDHC_WML_WRWML(1); //设置读写总线缓冲区容量为 1word

//设置 SDHC 模块的通信速率

SD_SetBaudRate(SD_InitStruct.SD_BaudRate); //设置 SD 卡通信速率为 25MHZ

while (SDHC->PRSSTAT & (SDHC_PRSSTAT_CIHB_MASK | SDHC_PRSSTAT_CDIHB_MASK)) ;

//初始化 GPIO

SD_PortType->PCR[D1] = (PORT_PCR_MUX(4) | PORT_PCR_PS_MASK | PORT_PCR_PE_MASK | PORT_PCR_DSE_MASK); /* ESDHC.D1 */

SD_PortType->PCR[D0] = (PORT_PCR_MUX(4) | PORT_PCR_PS_MASK | PORT_PCR_PE_MASK | PORT_PCR_DSE_MASK); /* ESDHC.D0 */

SD_PortType->PCR[CLK] = (PORT_PCR_MUX(4) | PORT_PCR_DSE_MASK); /* ESDHC.CLK */

SD_PortType->PCR[CMD] = (PORT_PCR_MUX(4) | PORT_PCR_PS_MASK | PORT_PCR_PE_MASK | PORT_PCR_DSE_MASK); /* ESDHC.CMD */

SD_PortType->PCR[D3] = (PORT_PCR_MUX(4) | PORT_PCR_PS_MASK | PORT_PCR_PE_MASK | PORT_PCR_DSE_MASK); /* ESDHC.D3 */

SD_PortType->PCR[D2] = (PORT_PCR_MUX(4) | PORT_PCR_PS_MASK | PORT_PCR_PE_MASK | PORT_PCR_DSE_MASK); /* ESDHC.D2 */

SDHC->IRQSTAT = 0xFFFF;

//使能中断位

```
SDHC->IRQSTATEN = SDHC_IRQSTATEN_DEBESEN_MASK
                | SDHC_IRQSTATEN_DCESEN_MASK
                | SDHC_IRQSTATEN_DTOESEN_MASK
                | SDHC_IRQSTATEN_CIESEN_MASK
                | SDHC_IRQSTATEN_CEBESEN_MASK
                | SDHC_IRQSTATEN_CCESEN_MASK
                | SDHC_IRQSTATEN_CTOESEN_MASK
                | SDHC_IRQSTATEN_BRRSEN_MASK
                | SDHC_IRQSTATEN_BWRSEN_MASK
                | SDHC_IRQSTATEN_CRMSEN_MASK
                | SDHC_IRQSTATEN_TCSEN_MASK
                | SDHC_IRQSTATEN_CCSEN_MASK;
  //延时
    for(delay_cnt=0;delay_cnt<10000;delay_cnt++);
    //80 个时钟周期的初始化
    SDHC->SYSCTL |= SDHC_SYSCTL_INITA_MASK; //确保初始化在 80 个 sd 卡时钟周期内完成 ，并且在此期间总线允许发布命令，不需要等待 SD 卡自我清除完毕
    while (SDHC->SYSCTL & SDHC_SYSCTL_INITA_MASK){}; //等待初始化完成
    //--------------以下开始 SD 卡初始化 物理层协议--------------------------
    //开始 SD 卡初始化进程 --------------------------------
    //说明      CMD0 -> CMD8 -> while(CMD55+ACMD41) ->CMD2 -> CMD3 ->CMD9
    //                          -> CMD7( 选 中 卡 )-> CMD16( 设 置 块 大 小)->(CMD55+ACMD6)设置位 4 线位宽
    //--------------------------- 正 式 开 始 ------------------------------  now Let's begin !
    //CMD0  使所有卡进入 IDLE
    SD_CommandStruct1.COMMAND = ESDHC_CMD0;
    SD_CommandStruct1.ARGUMENT = 0;
    SD_CommandStruct1.BLOCKS = 0;
```

```
result = SD_SendCommand(&SD_CommandStruct1);
if(result != ESDHC_OK) return ESDHC_ERROR_INIT_FAILED;        //
错误则返回


//CMD8   判断是 V1.0 还是 V2.0 的卡
SD_CommandStruct1.COMMAND = ESDHC_CMD8;
SD_CommandStruct1.ARGUMENT =0x000001AA;
SD_CommandStruct1.BLOCKS = 0;
result = SD_SendCommand(&SD_CommandStruct1);
if (result > 0)   //CMD8 无响应   错误或者普通卡
{
    result = ESDHC_ERROR_INIT_FAILED;
}
if (result == 0) //SDHC  卡
{
    hc = TRUE;
}
//反复发送 55+ACDM41  直到卡准备好
do
{
    //延时
    for(delay_cnt=0;delay_cnt<10000;delay_cnt++);
    i++;
    SD_CommandStruct1.COMMAND = ESDHC_CMD55;
    SD_CommandStruct1.ARGUMENT =0;
SD_CommandStruct1.BLOCKS = 0;
  result = SD_SendCommand(&SD_CommandStruct1);

    SD_CommandStruct1.COMMAND = ESDHC_ACMD41;
    if(hc)
    {
        SD_CommandStruct1.ARGUMENT  =  0x40300000;   //若是大容
```

```
量 SD 卡    //0~2G            SDSC      标准容量 SD 卡
            }
//2~32G         SDHC      大容量 SDCH 储存卡
            else
//32~2T         SDXC      容量扩大化的安全储存卡
            {
                SD_CommandStruct1.ARGUMENT = 0x00300000;
            }
            result = SD_SendCommand(&SD_CommandStruct1);
        }while ((0 == (SD_CommandStruct1.RESPONSE[0] & 0x80000000)) && (i
< 30));
        //CMD2  取 CID
        SD_CommandStruct1.COMMAND = ESDHC_CMD2;
        SD_CommandStruct1.ARGUMENT = 0;
        SD_CommandStruct1.BLOCKS = 0;
        result = SD_SendCommand(&SD_CommandStruct1);



        if(result != ESDHC_OK) return ESDHC_ERROR_INIT_FAILED;
        SD_InitStruct.CID[0] = SD_CommandStruct1.RESPONSE[0];
        SD_InitStruct.CID[1] = SD_CommandStruct1.RESPONSE[1];
        SD_InitStruct.CID[2] = SD_CommandStruct1.RESPONSE[2];
        SD_InitStruct.CID[3] = SD_CommandStruct1.RESPONSE[3];

        //CMD3  取 RCA
        SD_CommandStruct1.COMMAND = ESDHC_CMD3;
        SD_CommandStruct1.ARGUMENT = 0;
        SD_CommandStruct1.BLOCKS = 0;
        result = SD_SendCommand(&SD_CommandStruct1);
        if(result != ESDHC_OK) return ESDHC_ERROR_INIT_FAILED;
        SD_InitStruct.RCA = SD_CommandStruct1.RESPONSE[0]>>16;
```

```
Data_SD_RCA = SD_CommandStruct1.RESPONSE[0];
//CMD9  取 CSD
SD_CommandStruct1.COMMAND = ESDHC_CMD9;
SD_CommandStruct1.ARGUMENT = SD_InitStruct.RCA<<16;
SD_CommandStruct1.BLOCKS = 0;
result = SD_SendCommand(&SD_CommandStruct1);
if(result != ESDHC_OK) return ESDHC_ERROR_INIT_FAILED;
SD_InitStruct.CSD[0] = SD_CommandStruct1.RESPONSE[0];
SD_InitStruct.CSD[1] = SD_CommandStruct1.RESPONSE[1];
SD_InitStruct.CSD[2] = SD_CommandStruct1.RESPONSE[2];
SD_InitStruct.CSD[3] = SD_CommandStruct1.RESPONSE[3];

//CMD7  选中卡
SD_CommandStruct1.COMMAND = ESDHC_CMD7;
SD_CommandStruct1.ARGUMENT = SD_InitStruct.RCA<<16;
SD_CommandStruct1.BLOCKS = 0;
result = SD_SendCommand(&SD_CommandStruct1);
if(result != ESDHC_OK) return ESDHC_ERROR_INIT_FAILED;
//CMD16  设置块大小
SD_CommandStruct1.COMMAND = ESDHC_CMD16;
SD_CommandStruct1.ARGUMENT = 512;
SD_CommandStruct1.BLOCKS = 0;
result = SD_SendCommand(&SD_CommandStruct1);
if(result != ESDHC_OK) return ESDHC_ERROR_INIT_FAILED;

//CMD55  使用 ACMD 命令
SD_CommandStruct1.COMMAND = ESDHC_CMD55;
SD_CommandStruct1.ARGUMENT = SD_InitStruct.RCA<<16;
SD_CommandStruct1.BLOCKS = 0;
result = SD_SendCommand(&SD_CommandStruct1);
if(result != ESDHC_OK) return ESDHC_ERROR_INIT_FAILED;
//ACMD6  修改 SD 卡通讯位宽
```

```
        SD_CommandStruct1.COMMAND = ESDHC_ACMD6;
        SD_CommandStruct1.ARGUMENT = 2;
        SD_CommandStruct1.BLOCKS = 0;
        result = SD_SendCommand(&SD_CommandStruct1);//修改 SD 卡位 4 位通
讯位宽
        if(result != ESDHC_OK) return ESDHC_ERROR_INIT_FAILED;
         //设置 Kinetis 的 SDIO 模块位 4 线模式
        SDHC->PROCTL &= (~SDHC_PROCTL_DTW_MASK);
        SDHC->PROCTL                                                    |=
SDHC_PROCTL_DTW(ESDHC_PROCTL_DTW_4BIT);
        //判断卡类型
        if((SD_InitStruct.CSD[3]>>22)&0x03)
        {
            Data_SD_CardType = SD_CARD_TYPE_SDHC;
        }
        else
        {
            Data_SD_CardType = SD_CARD_TYPE_SD;
        }

            //顺利初始化操作
        SD_InitStruct.SD_Size = SD_GetCapacity(&SD_InitStruct);

        Data_Count=(SD_InitStruct.SD_Size*1024*1024/512);//计算长度 为了保
持 sd 能够高速存储

        return ESDHC_OK;
    }
```

```
//多块写指令初始化
void Data_SD_WT_Init(void)
{
    SD_CommandTypeDef SD_CommandStruct1;

    SD_CommandStruct1.COMMAND = ESDHC_CMD25;
    SD_CommandStruct1.BLOCKS =Data_Count;
    SD_CommandStruct1.BLOCKSIZE = 512;
    SD_CommandStruct1.ARGUMENT = 0;//sd 卡初始地址
    SD_SendCommand(&SD_CommandStruct1);

}


//多块写指令
void Data_SD_WriteMultiBlock(const uint8_t *pbuffer,uint16_t Count)
{
    uint32_t j;
    uint8_t* ptr=(uint8_t *) pbuffer;

    for (j=0 ;j<(SD_WT_BlocksNum*((SD_Block_Size+3)>>2));j++)
    {
        while (0 == (SDHC->PRSSTAT & SDHC_PRSSTAT_BWEN_MASK)){}; //等待数据准备好

        if((j<<2)<Count)
        {
            SDHC->DATPORT =  *(uint32_t *)ptr;
            ptr+=4;
        }
        else
```

```
            {
                SDHC->DATPORT = 0xFFFFFFFF;
            }
        }
    }
```

```
    //读 SD 卡的一个 block

    void Data_SD_ReadSingleBlock(uint32_t sector, uint8_t *buffer,uint16_t Count)
    {
        uint32_t j;
       volatile uint32_t   temp=0xFFFFFFFF;//用来消耗一下剩余的块
         uint8_t *ptr = (uint8_t*)buffer;
        SD_CommandTypeDef SD_CommandStruct1;
        if(Data_SD_CardType == SD_CARD_TYPE_SD) //如果是普通 SD 卡 把块地址转换成字节地址
        {
            sector = sector<<9;
        }
         while (SDHC->PRSSTAT & SDHC_PRSSTAT_DLA_MASK){};//等待 DATA 线空闲

         SD_CommandStruct1.COMMAND = ESDHC_CMD17;
         SD_CommandStruct1.ARGUMENT = sector;
         SD_CommandStruct1.BLOCKS = 1;
         SD_CommandStruct1.BLOCKSIZE = SD_Block_Size;
```

```
        SD_SendCommand(&SD_CommandStruct1);


                for (j =0 ;j<(SD_Block_Size+3)>>2;j++)
                {

                                //等待数据准备好
                        while (0 == (SDHC->PRSSTAT & SDHC_PRSSTAT_BREN_MASK)) {};
                        if((j<<2)<Count)
                        {
                                *(uint32_t *)ptr = (uint32_t)SDHC->DATPORT;

                                ptr+=4; //这里取代 *ptr++=SDHC->DATPORT; 因为这句有 BUG
                        }
                        else
                        {
                            temp=SDHC->DATPORT;
                        }

                }

    }


    void Data_Read_Init(void)//读 SD 卡初始化
    {
        Blocks_Num=0;
    }

    void Data_SD_Save(void)//SD 卡存储信息
```

```
{
    Set_DataToCon();

    SetFlashDataToCon();

    Compressed_Data();//压缩数据

    Data_SD_WriteMultiBlock((uint8_t  *)Sent_Data,Data_All_Length);/*存取数据*/

    if(SD_ALL_WT_Count%50==0)  INI_Flash_Count_Save();/*100 场刷新计数一次*/
}



void Uart_Read_SD(void)//读取 SD 卡信息  Uart 发送
{
    int i,j;
    INI_Flash_Count_Read(); //读取场数
    OLED_Clear();//清屏
    Data_Read_Init();//SD 传图初始化
    OLED_Write_String(3,1,(uint8_t*)"ALL:");
    OLED_Write_Num4(7,1,SD_ALL_WT_Count);
    OLED_Write_String(3,3,(uint8_t*)"NUM:");
    for(i=0;i<=SD_ALL_WT_Count;i++)// 上传到一定数量自动关闭 SD_ALL_WT_Count
    {
        Data_Read();//读 SD 卡
        OLED_Write_Num4(7,3,i);
        Data_UpLoad();//传数据
```

```
            for(j=0;j<500000;j++){;}
            Blocks_Num+=SD_WT_BlocksNum;
        }
        OLED_Write_String(1,5,(uint8_t*)"READ SD IS OK.");
        while(1);
    }


    void OLED_Read_SD(void)//读取 SD 卡信息  OLED 显示
    {
        uint8_t Key_temp;
        int i,Par_Page;
        uint8_t Mode=0,Display_Flag=0;
        INI_Flash_Count_Read(); //读取场数
        IT_Image_Mode_Key();//开启中断按键
        EnableInterrupts();//开启总中断
        OLED_Clear();//清屏
        Data_Read_Init();//SD 传图初始化
        for(i=0;i<=SD_ALL_WT_Count;)// 上 传 到 一 定 数 量 自 动 关 闭
SD_ALL_WT_Count
        {
            Key_temp=get_Image_Mode_Key();//获取中断按键
        //                  for(j=0;j<Key_Delay_Count;j++)      {DelayMs(15);
get_Image_Mode_Key();}//延时中断按键过于灵敏


            switch(Key_temp)
            {
                case
1:Display_Flag=~Display_Flag;Key_temp=0;OLED_Clear();break;
                case
6:if(i>0&&i<=SD_ALL_WT_Count){Blocks_Num+=SD_WT_BlocksNum;i++;}Ke
y_temp=0;OLED_Clear();break;
```

```
                case
3:if(i>0&&i<=SD_ALL_WT_Count){Blocks_Num-=SD_WT_BlocksNum;i--;}Key_
temp=0;OLED_Clear();break;
                case 2:Mode++;OLED_Clear();Key_temp=0;break;
                case
4:if(Par_Page<(DataNum+3)/4-1){Par_Page++;}else{Par_Page=0;}OLED_Clear();K
ey_temp=0;break;
                case
5:if(Par_Page==0){Par_Page=((DataNum+3)/4-1);}else{Par_Page--;}OLED_Clear();
Key_temp=0;break;
                default:break;
            }

            if(Mode>3)
            {
                Mode=Mode%3;
            }
            if(Display_Flag)
            {
                if(Mode==1)
                {
                    Data_Read();
                    Unpack_Data(); //解压数据
                    OLED_Real_Image_Dispaly();
                }
                if(Mode==2)
                {
                    Data_Read();
                    OLED_Fit_Image_Dispaly();
                }
                if(Mode==3)
                {
                    Data_Read();
```

```
                OLED_Par_Dispaly_Host(Par_Page);//显示参数  本地
            }
            i++;
            Blocks_Num+=SD_WT_BlocksNum;
            DelayUs(100000/Image_SD_Speed);//调节速度
        }
        else
        {
            if(Mode==1)
            {
                Data_Read();
                Unpack_Data(); //解压数据
                OLED_Real_Image_Dispaly();
            }
            if(Mode==2)
            {
                Data_Read();
                OLED_Fit_Image_Dispaly();
            }
            if(Mode==3)
            {
                Data_Read();
                OLED_Par_Dispaly_Host(Par_Page);//显示参数  本地
            }
            //DelayUs(100000/Image_SD_Speed);//调节速度
        }



}
OLED_Clear();
OLED_Write_String(3,1,(uint8_t*)"ALL:");
```

```
        OLED_Write_Num4(7,1,SD_ALL_WT_Count);
        OLED_Write_String(3,3,(uint8_t*)"NUM:");
        OLED_Write_Num4(7,3,i-1);
        OLED_Write_String(1,5,(uint8_t*)"READ SD IS OK.");
        while(1);
    }


    void Data_Read(void)//读取数据到 Data 数组
    {
        uint8_t i=0;
        uint16_t Data_Count =Data_All_Length;//记录长度


        for(i=0;i<((Data_All_Length+SD_Block_Size-1)>>9);i++)//读一次所需的
数据块数
        {
            Data_SD_ReadSingleBlock(Blocks_Num+i,(uint8_t
*)Sent_Data+(i<<9),Data_Count>=SD_Block_Size?SD_Block_Size:Data_Count );
            Data_Count-=SD_Block_Size;
        }
    }
```