第十六届全国大学生

智能汽车竞赛

**技 术 报 告**

学 校： 常熟理工学院

队伍名称： 盘龙队

参赛队员： 王彦栋

李其轩

文财明

带队教师： 李鑫

# 摘要

本文设计的电单车智能车系统以[SAK-TC264DA-40F200W BC](https://www.infineon.com/cms/cn/product/microcontroller/32-bit-tricore-microcontroller/32-bit-tricore-aurix-tc2xx/aurix-family-tc264da-adas/sak-tc264da-40f200w-bc/)微控制器为核心控制单元，通过车体上方的摄像头检测赛道信息；通过编码器检测模型车的实时速度，使用改进的PID 控制算法调节驱动电机的转速和舵机的角度，使用串级PID控制算法调节无刷电机的转速和转动方向，实现了对车运动速度和运动方向的闭环控制。为了提高模型车的速度和稳定性，使用匿名上位机、虚拟示波器、蓝牙模块、按键模块等调试工具，进行了大量硬件与软件测试。使得对各种赛道元素有较好的适应能力，能够更好，更快的完成竞赛要求。

目录

[摘要 2](#_Toc112336963)

[引言 1](#_Toc112336964)

[总体方案设计 2](#_Toc112336965)

[(一) 系统的总体方案设计 2](#_Toc112336966)

[(二) 系统的组成 2](#_Toc112336967)

[第一章 机械结构部分 4](#_Toc112336968)

[1.1 智能车车模 4](#_Toc112336969)

[1.2 智能车传感器安装 4](#_Toc112336970)

[1.2.1 编码器 4](#_Toc112336971)

[1.2.2 陀螺仪 4](#_Toc112336972)

[1.2.3 摄像头 4](#_Toc112336973)

[1.3 智能车轮胎的保养和选型 4](#_Toc112336974)

[1.4 电池及电池板的安装 5](#_Toc112336975)

[1.4.1 电池板的安装 5](#_Toc112336976)

[1.4.2 电池的固定 5](#_Toc112336977)

[1.5 其他结构调整 5](#_Toc112336978)

[第二章 硬件电路设计 6](#_Toc112336979)

[2.1 硬件设计方案 6](#_Toc112336980)

[2.1.1 电源管理模块 6](#_Toc112336981)

[2.1.2 电机驱动模块 6](#_Toc112336982)

[2.1.3 主控电路模块 6](#_Toc112336983)

[2.1.4 无刷电机模块 6](#_Toc112336984)

[2.2 智能车传感器模块设计 6](#_Toc112336985)

[第三章 小车软件控制 7](#_Toc112336986)

[3.1 图像处理算法 7](#_Toc112336987)

[3.2 PID算法简介 7](#_Toc112336988)

[3.3 舵机PID算法 7](#_Toc112336989)

[3.4 串级PID算法 8](#_Toc112336990)

[第四章 调试过程概述 9](#_Toc112336991)

[4.1 开发工具 9](#_Toc112336992)

[4.2 调试过程 9](#_Toc112336993)

[4.2.1 上位机调试 9](#_Toc112336994)

[4.2.2 串口屏、蓝牙、按键调试 9](#_Toc112336995)

[4.3 整车机械方面的调整 9](#_Toc112336996)

[第五章 模型车的主要技术参数 11](#_Toc112336997)

[结论 12](#_Toc112336998)

[参考文献 13](#_Toc112336999)

[附录A：原理图 14](#_Toc112337000)

[附录B：程序源代码 14](#_Toc112337001)

# 引言

随着科技的逐年发展，电子技术在各行各业中所起到的作用逐渐凸显，相关的技术研究和领域也受到了人们的高度重视，这同时为智能研究提供了更加广阔的平台。

作为全国范围内高质量、高水平的全国大学生智能车竞赛，它是涵盖了包括模块控制、智能识别、传感技术、机械原理、电子芯片、电气理论、计算机设计等多个学科的大型比赛，参赛选手利用指定的材料和工具设计开发适应指定比赛类型的赛车，比赛的公平性得以保障，同时对于学生的知识融合和实践能力的提高，具有良好的推动作用，因此在各大高校中都有着很高的地位。

本届全国大学生智能汽车竞赛分为四轮电磁组、四轮摄像头组、多车编队组、平衡单车组、无线充电组、平衡信标组、智能视觉组、极速越野组、完全模型组九个组别，分别进行比赛。我队参与了平衡单车组的比赛。此次比赛严格按照官方规定需要使用K型车模，使用英飞凌的TC264系列单片机以及其他指定物品，完成本次比赛设计。

本篇技术报告主要包括了赛车机械结构、硬件系统和软件系统等方面的内容，详细介绍了我们在整个准备过程中有关单车的制定方案和设计思路。

# 总体方案设计

本章主要简要地介绍智能车系统总体设计思路，在后面的章节中将整个系统分为机械结构、控制模块、控制算法等三部分对智能车控制系统进行深入的介绍分析。

1. 系统的总体方案设计

根据十六届智能车的规则，单车拉力组可采用K型车模。车模作品尺寸不限。但是车模上不允许安装额外的用于保持车模平衡的惯量轮。车模微控制器指定使用 STC 系列单片机。允许使用电感、红外光电、摄像头传感器、激光传感器等。比赛没有赛道，只有电磁线。但赛道元素包括有直道、弯道、坡道、十字路口。赛道一般铺设在路面相对平整的室外运动场内，路面不会有颠簸路面和沙石路段。

赛车的位置信号由车体前方的电感电磁杆采集，经过OPA4377搭建的放大电路放大之后，输入到控制核心，用于赛车的运动控制决策。通过编码器模块来采值检测车速；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转速，完成智能车速度的闭环控制。使用OLED屏幕来看小车相应调试的参数。此外，还增加了键盘作为小车启停和开关屏幕的操作，便于小车的发车。

1. 系统的组成

系统主要包括平衡控制、图像处理、驱动控制、数据处理四个部分。各个部分的作用如下：

1.数据处理：作为小车的主体部分，它扮演者整个单车的“大脑”，我们的小车的微处理器使用的是官方指定的英飞凌的TC264芯片，小车使用了两块TC264芯片，它们虽然都用于数据的采集和处理，但是作用不尽相同，位于无刷驱动板上的TC264芯片单独服务于动量轮，通过实时数据处理以保证小车的平衡，而位于主控板上的TC264芯片，由于主控板上连接了各种外设，因此处理的数据也相对更加庞大，复杂程度也更加高。

2.平衡控制：通过陀螺仪采集到的数据处理后经过串级PID控制将数据反馈给无刷电机，动量轮转动所带来的力矩使小车能够保持平衡，在车子行驶过程中我们将处理后的数据反馈给了舵机，以保证车子在转向时能够根据当前车身倾斜角度来调整舵机的转向PID参数。

3.驱动控制：除了对于控制后轮的电机，还单独为用于控制车身平衡的无刷电机制作了一块无刷驱动板，以满足无刷电机的供电控制的需求。

4.图像处理：图像采集使用了逐飞科技MT9V03X系列总钻风摄像头，将获取原始的灰度图像通过转化变成图像处理用到的二值化图像，然后通过具体的程序检测图像中包含的赛道信息，通过对这些信息进行判断分析并取出我们需要的重要信息。

1. 机械结构部分

智能汽车各系统的控制都是在机械结构的基础上实现的，因此在设计整个软件架构和算法之前一定要对整个模型车的机械结构有一个全面清晰的认识，然后建立相应的数学模型，从而再针对具体的设计方案来调整赛车的机械结构，并在实际的调试过程中不断的改进优化和提高结构的稳定性。本章将主要介绍新K车模的机械结构和调整方案。

* 1. 智能车车模

本届比赛使用的车模型号为K车模.

* 1. 智能车传感器安装

小车上的传感器包括：陀螺仪，编码器，摄像头。以下将介绍各传感器的详细安装。由于单车上的位置有限，所以各传感器在固定上大都使用的热熔胶来固定。

* + 1. 编码器

速度编码器使用的是1024线带方向mini编码器。

该编码器的特点是精度高、抗干扰能力强，在采集速度信息实现闭环控制上有着显著的作用，安装时要保证齿轮之间相互啮合，齿间缝隙较小。

* + 1. 陀螺仪

陀螺仪采用的是六轴加速度计陀螺仪ICM-20602。

陀螺仪不仅要采集x轴、y轴方向上的角度值，同时还有加速度值，这很大程度上影响了后面车子的稳定性，所以关于陀螺仪的安装要十分关心，不要将陀螺仪装的离地面高度太高或者偏离车中心位置太远。

* + 1. 摄像头

摄像头使用的是MT9V03X系列总钻风数字摄像头。

摄像头的安装决定了后续步骤中图像的采集处理，作用与陀螺仪同样重要，首先要选择镜头度数合适的摄像头，当拿到摄像头后，先要调整好成像清晰度，之后调整好摄像头离地高度以及摄像头俯仰角度，当摄像头能够看到远处合适距离的地方时，就可以开始固定了。

* 1. 智能车轮胎的保养和选型

本此比赛车模采用的“轮胎”材料为橡胶与海绵，在使用的过程中主要暴露出的问题有：高速过弯时发生抬轮现象、长期使用容易磨损等。在“轮胎”粘接时，我们尽量保证其圆度；在不调试的时间里，尽量避免“轮胎”与地面接触，保证轮胎的形状不发生改变。

* 1. 电池及电池板的安装
     1. 电池板的安装

为了在不影响其他硬件的安装的同时能够安装电池，我们将为电池向外另外拓展了一块无电气作用的板子，用于电池的单独安装。

* + 1. 电池的固定

电池安装在电池板上，用扎带固定。

* 1. 其他结构调整

由于需要安装无刷电机，在智能车竞赛组委会许可的情况下，我们对K车模的底部进行了切割处理，让K车模底部刚好能够容纳下动量轮跟电机，利用热熔胶和绑带进行固定处理。

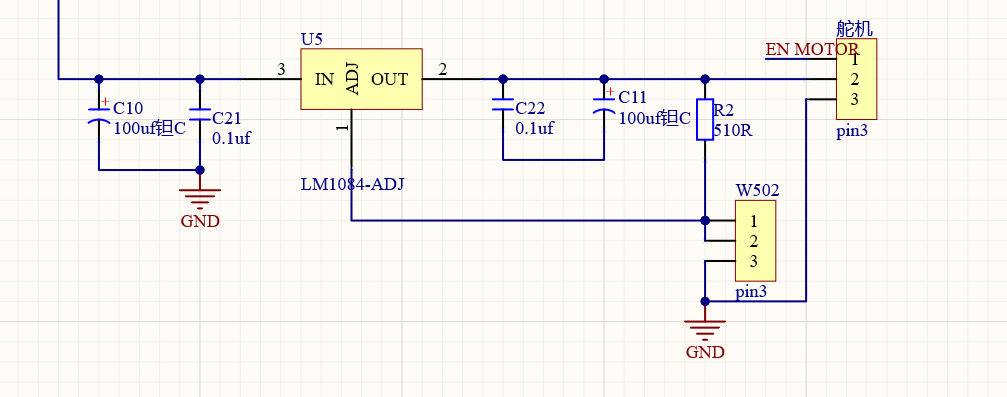
1. 硬件电路设计
   1. 硬件设计方案

可靠、高效、简洁是我们硬件设计的原则，选用性能优秀而且性价比好的芯片，设计稳定可靠兼容性好的电路板。 可靠性是系统设计的第一要求，在电路设计的所有环节都进行了电磁兼容性设计，做好各部分的接地、屏蔽、滤波等工作，将高速数字电路与模拟电路分开，使本系统工作的可靠性达到了设计要求。 高效是指本系统的性能要足够强劲，使用了由分立元件制作的直流电动机可逆双极型桥式驱动器，该驱动器的额定工作电流可以轻易达到几十安培以上，保证了电动机的工作转矩和转速。 简洁是指在满足了可靠、高效的要求后，为了尽量减轻整车重量，降低模型车的重心位置，应使电路设计尽量简洁，尽量减少元器件使用数量，缩小电路板面积，使电路部分重量轻，易于安装。

* + 1. 电源管理模块

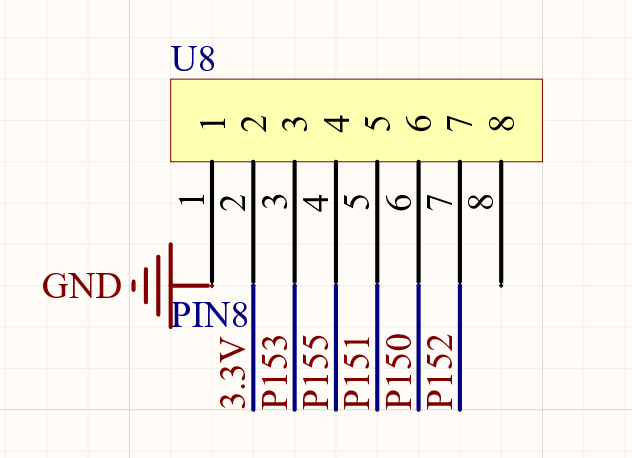
电源模块对于一个控制系统来说极其重要，关系到整个系统是否能够正常工作，因此在设计控制系统时应选好合适的电源模块。竞赛规则规定，车模所使用的电池可以是原车模厂商配套的镍镉动力电（7.2V，2AH），也可以是3S锂电池等等。本单车使用了高放电倍率的3S1500mah的锂电池一个。

舵机供电模块：



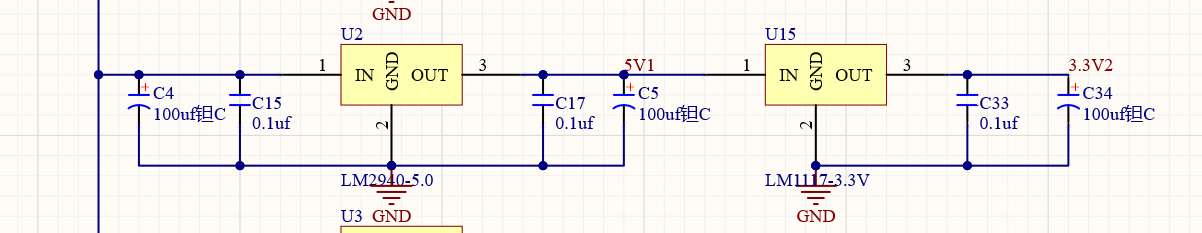
舵机供电模块 1

IPS供电模块、电机接口：



IPS供电模块、电机接口 1

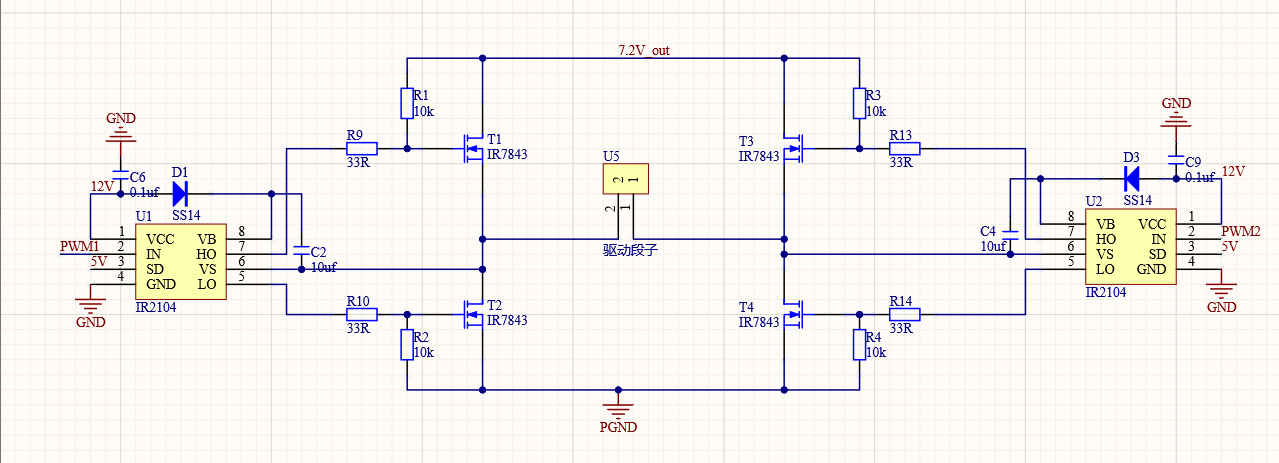
3.3V稳压模块



3.3V稳压模块 1

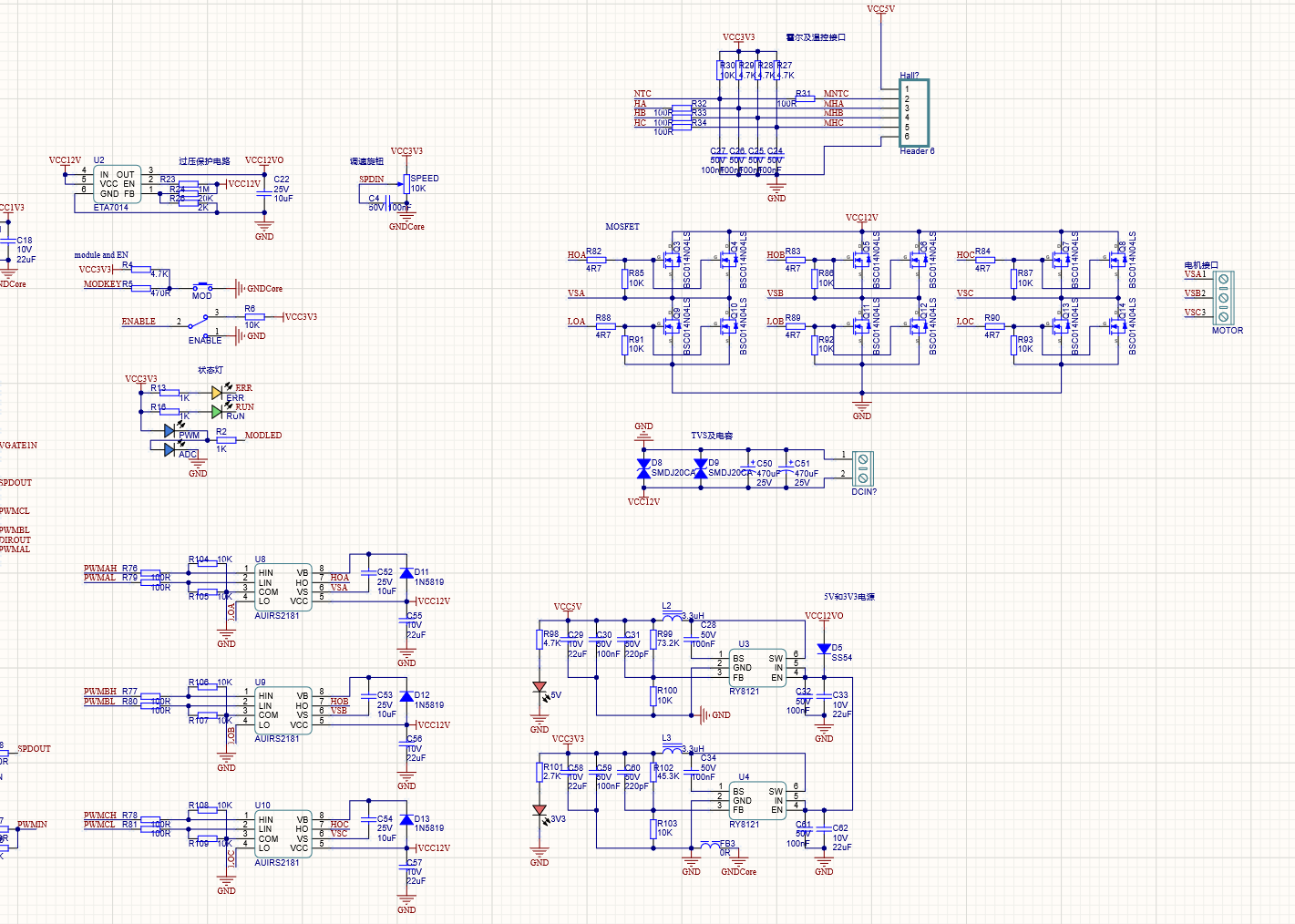
* + 1. 电机驱动模块

直流有刷驱动电路原理图：



直流有刷驱动电路原理图 1

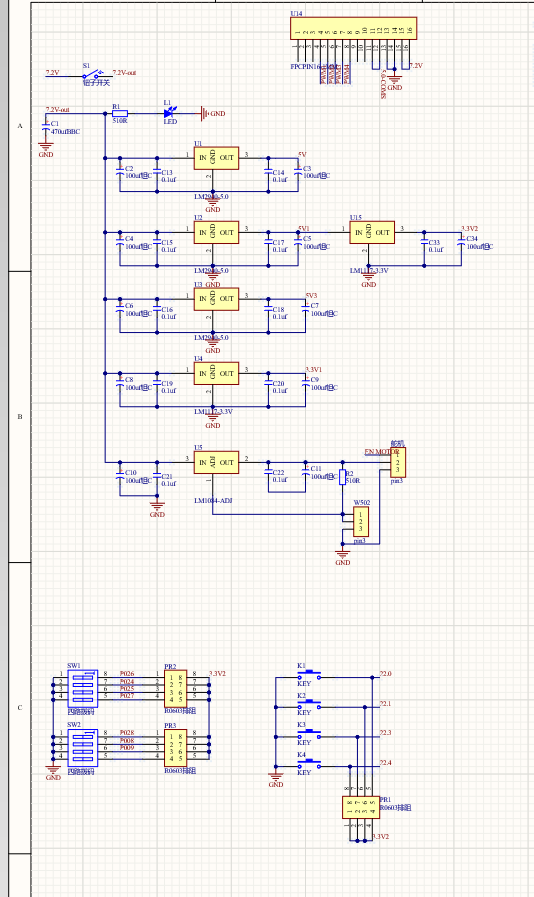
无刷驱动原理图：



无刷驱动原理图 1

* + 1. 主控电路模块

主控原理图：



主控原理图 1

* 1. 智能车传感器模块设计

小车使用的传感器为MT9V03X总钻风摄像头。

1. 小车软件控制
   1. 图像处理算法

图像处理大致可以分为信息提取和处理两个过程：

1.信息提取：通过采取合适的扫线方式，提取完整的赛道信息，我们在扫线方面方面使用的是通用的两种扫线方式，一种是从上一行的终点出发向两边扫线找到当前所在行的边界位置，另一种是从上一行的边界位置附近出发扫线找到当前行的边界位置，第二行明显能够节省时间，测试中我们发现第二种方法对于三岔路的处理有着很好的效果，但是受到的干扰太大，对于更多的赛道情形往往是不太适用的，所以我们采取了两种方法相结合。

上面说的扫线只是在每一行上做的横向处理，但这样找到的信息有限，有时候我们还需要进行纵向寻线，在每一列从某行出发向上搜寻边界，用纵向寻线补足横向寻线的不足，最终获取的图像信息已经足够完整了，当然，摄像头的远瞻（能看到的最远距离）很大程度影响着纵向寻线的最终效果，所以在拿到摄像头后一定记得调整好摄像头的位置并保证之后的调试过程中摄像头不再调整。

2.处理：处理部分主要是对图像进行补线处理，对于某些类型的赛道，往往需要进行补线才能让它的边界轨迹达到我们的预期效果，比如三岔我们如果要向某一边转向是，需要将另一个方向的路口堵住，或者是入圆环时我们需要把车子前面的路堵死，这就涉及到了补线的问题。通过求斜率或者其他特殊方法，拟合出一条边界路径，这样就可以找到车子行驶的最佳轨迹。

* 1. PID算法简介

比例积分微分控制，简称PID控制，其中P表示比例、I表示积分、D表示微分。PID控制算法是最早发展起来的控制策略之一，由于其算法简单、鲁棒性好和可靠性高，被广泛应用于工业过程控制。而且也衍生出多种相关的控制算法：P控制、PI控制、PD控制、增量式PID控制、模糊PID控制等。虽然这些算法各不相同，但都是基于最基本的PID控制算法为原型的。

* 1. 舵机PID算法

对于舵机的闭环控制，我们采用了位置式PID控制算法，根据往届的技术资料和实际测试，将每场图像的黑线中心加权平均值与舵机PID参考角度值构成一次线性关系。

经过反复测试，我们选择的PID调节策略是：

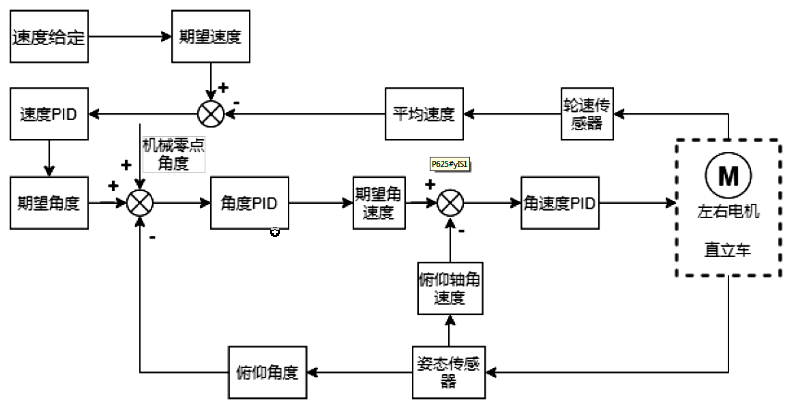
(1) 将积分项系数置零，我们发现相比稳定性和精确性，舵机在这种随动系统中对动态响应性能的要求更高。更重要的是，在KI置零的情况下，我们通过合理调节Kp，发现车能够在直线高速行驶时仍能保持车身非常稳定，没有震荡，基本没有必要使用KI参数；

(2) 微分项系数KD使用定值，原因是舵机在一般赛道中都需要较好的动态响应能力；

(3) 对Kp，我们使用了二次函数曲线，Kp随中心位置与中心值的偏差呈二次函数关系增大，在程序中具体代码如下：loca\_Kp = (loca\_error \* loca\_error)/2 + 1000。

* 1. 串级PID算法

串级PID控制系统结构图：



串级PID控制系统结构图 1

串级[PID算法](https://so.csdn.net/so/search?q=PID%E7%AE%97%E6%B3%95&spm=1001.2101.3001.7020" \t "_blank)的实现在于角速度环、角度环和速度环控制的不同周期。初始化程序完成之后，启动周期为2 ms的定时中断。PIT 中断要为循环中的算法提供程序执行的时序标志位控制，根据标志位实现不同周期执行不同控制环节，这种程序结构有助于提高算法的执行效率以及实时性。

1. 调试过程概述
   1. 开发工具

程序的开发是在AURIX Development Studio下进行的，包括源程序的编写、编译和链接，并最终生成可执行文件。

ADS集成开发环境是 ARM公司推出的 ARM核微控制器集成开发工具，英文全称为 ARM DeveloperSuite。ADS主要包括 4个模块:SIMULATOR、C编译器、实时调试器和应用函数库。成熟版本为 ADS1.2。ADS 1.2支持 ARM10之前的所有 ARM系列微控制器,支持软件调试及硬件仿真调试,支持汇编、C、C++源程序,具有编译效率高、系统库功能强等特点，可以在 Windows98、WindowsXP、Windows7、Windows8以及 RedHat Linux上运行。

* 1. 调试过程
     1. 上位机调试

在观察图像、调试波形、测试参数的过程中我们分别采用了由逐飞科技提供的“逐飞科技智能车传感器调试助手”、“Serial\_Digital\_Scope V2\_Cracked（虚拟示波器）”和“sscom5.13.1”，通过对传感器实时采集数据并做相应处理，以取得我们希望的数据。在实际测试中，“sscom5.13.1”能够很好的把每个时刻的数据反馈给我们，且精确度较高；“Serial\_Digital\_Scope V2\_Cracked（虚拟示波器）”用于显示波形，为调节单车平衡提供了很大的帮助；“逐飞科技智能车传感器调试助手”用于显示图像，能够逐帧清晰地观察到图像，但是过于占用资源导致图像更新速度过慢，并不适合用于处理连续变化的图像。合理使用上位机，通过上位机分析才能取得有效的结果。

* + 1. 串口屏、蓝牙、按键调试

由于上位机需要在电脑上使用，为了方便，对于部分数据我们采用了显示屏和蓝牙、按键结合的形式进行调参，通过蓝牙或者按键来改变数据，大大节省了反复烧写代码的时间，便于观察小范围变化的数据对小车的整体影响。

* 1. 整车机械方面的调整

车模的机械也是很重要的一方面，针对平衡单车组车模，必须有几个特别需要关注的地方：

1）摄像头支架必须结实耐撞，震动尽量小。震动会导致传感器得到图像“抖动”，导致车模会在赛道上“颤抖”。

2）重心尽可能低，并且分布均匀。把车模地盘降低，车上装配的器件也尽可能的放低，转弯效果会比较好。分布均匀可以防止车模在赛道甩动。本组单车讲电池放到地盘下面有效的使得小车的重心大大降低。

3）为了防止小车侧面安装的驱动电机和电机不会因为小车频繁倒地而损坏，我们在车上加了一根碳素杆，在不影响比赛的前提下，保护小车的安全。

1. 模型车的主要技术参数

车模类型 K车模

长 28cm

宽 8.5cm

高 22cm

车重 约3kg

摄像头距地面高度 20cm

传感器 总钻风摄像头 1个

微处理器 TC264 两个

无刷电机 1个

直流有刷电机 1个

舵机 1个

陀螺仪 1个

编码器 1个

电池 3s1500mah12V锂电池 1个

赛道信息检测 精度（远瞻/近瞻） 150cm/30cm

频率 5ms

# 结论

自报名参加全国大学生智能汽车竞赛以来，我们小组成员从查找资料、设计机构、组装车模、编写程序一步一步的进行，最后终于完成了最初目标，定下了现在这个设计方案。

我们在这份技术报告中从智能车的机械结构、硬件电路和软件控制等方面介绍了我们为十七届全国大学生智能车大赛所做的准备。

相较于上一届智能单车，由于增设了动量轮，车子稳定性能大大提高，但相应地对于硬件电路的设计要求也更加严格，在经历了无数次的炸板后，我们才得以见证车子从原地直立平衡到蹒跚起步再到能够流畅的跑完整个赛道。

在传感器方面，我们用摄像头取代了电磁传感器，由于没有可继承的代码，只能多方寻找帮助，在众多车友和前辈的指点下我们取得了很大的进步。最终效果也很明显，尽管单车的行驶过程中会带来图像不同程度畸变，但是摄像头相比电磁传感器能够超前采集到赛道信息，车速的上限大大提高。

这次的做车经历对于我们而言有着不同程度的影响，我们产生过纠纷、经历过低谷、也体会过共同解决出难题后的喜悦，所有这些或许不可名状，但却给我们的回忆留下了印记，将会伴随着我们一生。

最后，感谢队伍中每位成员的不离不弃，感谢广大车友和前辈在这个过程中的帮助，感谢老师、学校的关注和关心。

# 参考文献

[1] 卓晴，黄开胜，邵贝贝．学做智能车：挑战“飞思卡尔”杯．北京：北京航空航天大学出版社，2007年.

[2] 王晓明．电动机的单片机控制 [M] ．北京：北京航空航天大学出版社. 2002.

[3] 谭浩强著．C程序设计．北京：清华大学出版社，2003．

[4] 童诗白，华成英．模拟电子技术基础[M]．北京. 高等教育出版社．2000.

[5] 阎石．数字电子技术基础 [M] ．北京: 高等教育出版社，2000.

[6] 尹怡欣，陶永华．新型PID控制及其应用．北京：机械工业出版社，1998年.

[7] 胡寿松.自动控制原理(第六版)[M].科学出版社,2014.

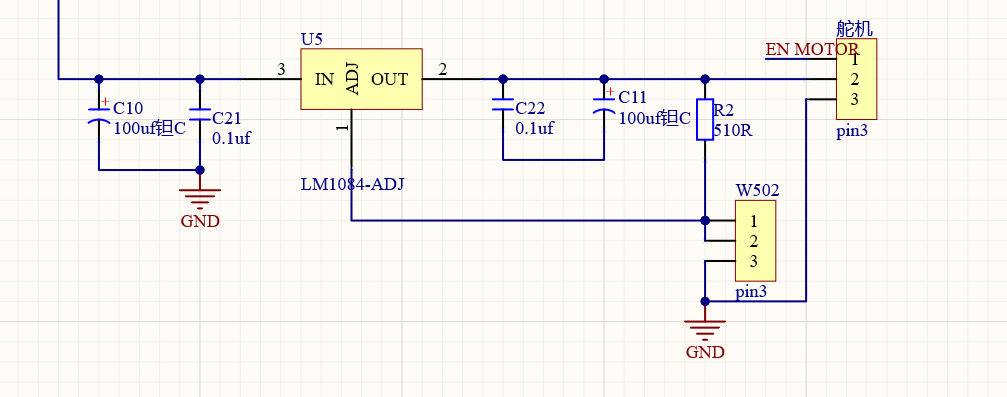
[8] 沈长生．常用电子元器件使用一读通[M]．北京. 人民邮电出版社．2004.

[9]张文春. 汽车理论[M]．北京．机械工业出版社．2005.

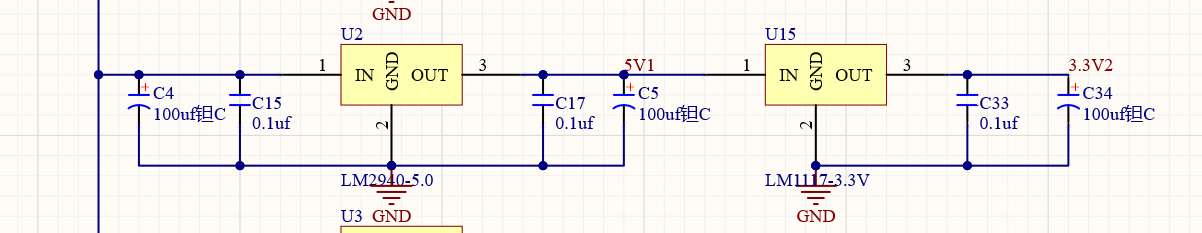
[10] 杨素行,模拟电子技术基础简明教程（第三版）.清华大学电子学教研组编北京：高等教育出版社,2005.

[11] 邓惜仁，宋学树．基于PID算法和 ABS点杀思想的智能车高速行驶研究[J].软件，2013，34(5):60-63.

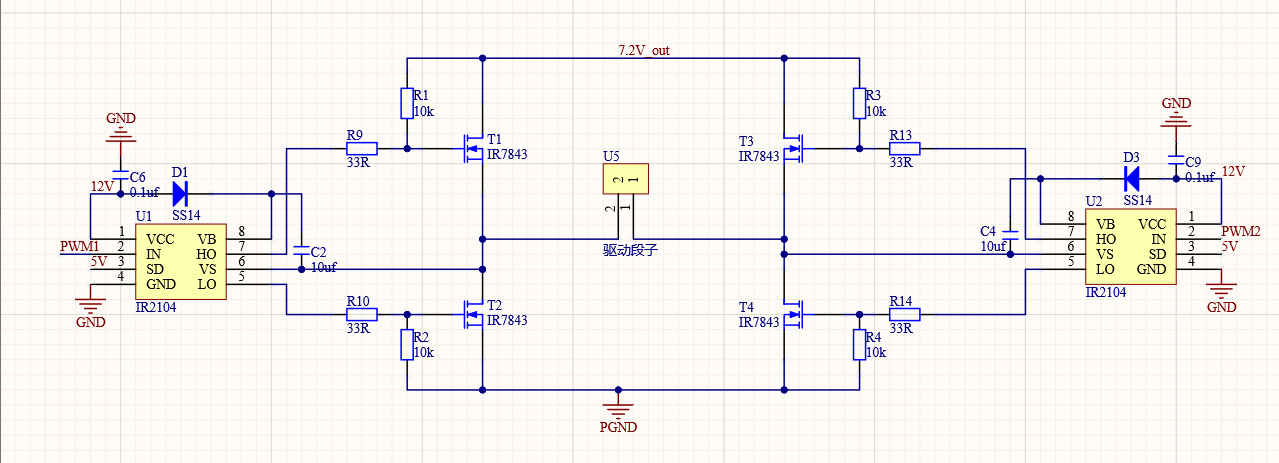
# 附录A：原理图



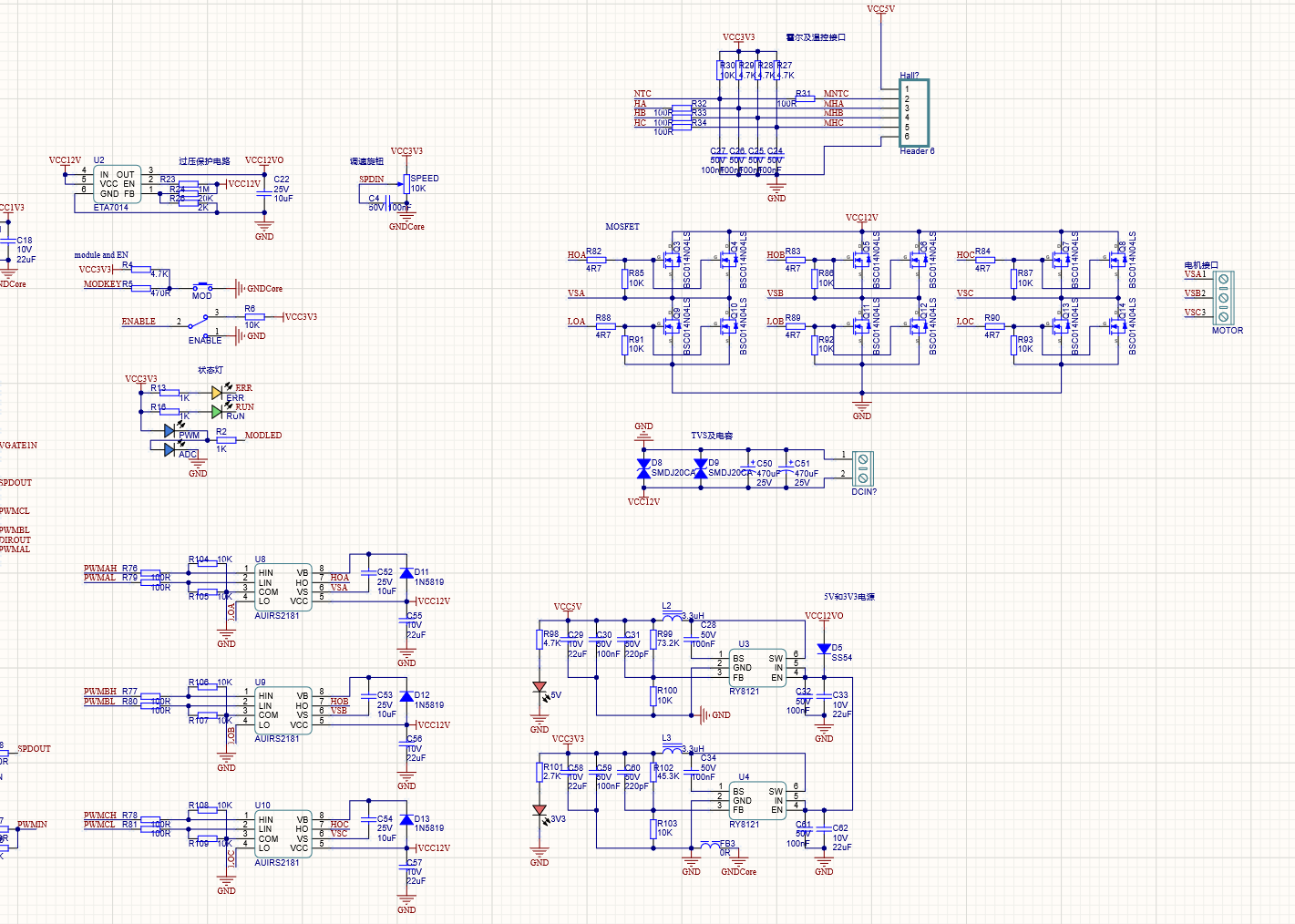
舵机供电模块 2



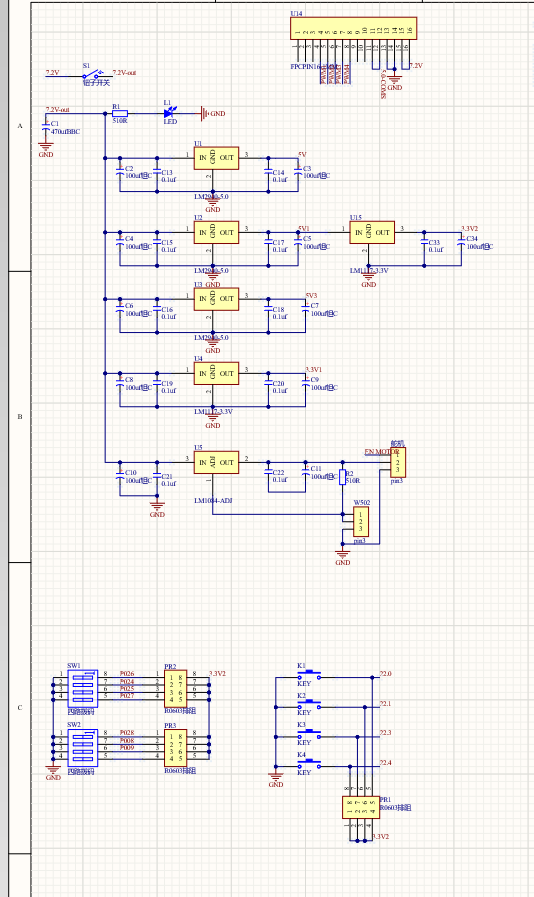
3.3V稳压模块 2



直流有刷驱动电路原理图 2



无刷驱动原理图 2



主控原理图 2

# 附录B：程序源代码

：特别的，当使用功能1时，还需要方向标志位trend\_flag，当该标志位值为1时，向左转弯，需要求左斜率，当该标志位值为2时，向右转弯，需要右斜率。

//函数输出：有效值value

//备注：无

//时间：2022-7-9-0:56

int16 **Vertical\_search**(uint16 start\_col, uint16 end\_col, uint16 start\_row, uint8 function\_flag, uint8 trend\_flag)

{

//通用的变量

sint16 row, col; //扫线行、列

int16 value = 0; //函数输出结果

//实现功能1时需要使用的变量

uint16 vertical\_length = 0, vertical\_longest\_length\_1 = 0, vertical\_longest\_length\_2 = 0; //vertical\_length 表示每列从扫线起始行到顶部边界的距离，vertical\_longest\_length\_1和vertical\_longest\_length\_2则分别记录两列最远的距离。

uint16 vertical\_search\_row = 0, vertical\_search\_col = 0, vertical\_search\_row\_1 = 0, vertical\_search\_col\_1 = 0,/\* vertical\_search\_row\_2 = 0,\*/ vertical\_search\_col\_2 = 0; //vertical\_search\_row和vertical\_search\_col是最终找到的要用于计算的距离最远点的行值和列值，而vertical\_search\_row\_1, vertical\_search\_col\_1, vertical\_search\_row\_2, vertical\_search\_col\_2则是中间记录的距离最远点的行和列，还需要转换后得到vertical\_search\_row, vertical\_search\_col

//实现功能2时需要使用的变量

uint8 found\_edge\_flag = 0; //功能2是统计找到顶（找到第0行都没有边界）的列数，所以需要对每一列能否找到顶作判断。

**if**(function\_flag == 1) //功能标志位为1时，此函数就用来求纵向斜率。

{

**for**(col = start\_col; col < end\_col; col ++) //该嵌套循环用以寻找纵向最远距离的点

{

**for**(row = start\_row; row > 0; row --)

{

**if**(mt9v03x\_image[row][col] == 0 && mt9v03x\_image[row - 1][col] == 0) //因为图像二值化处理是直接在mt9v03x\_image的基础上改变的（mt9v03x\_image的内容已经改变了），所以可以直接用mt9v03x\_image。

{

vertical\_length = start\_row - row;

**if**(vertical\_length > vertical\_longest\_length\_1)

{

vertical\_longest\_length\_1 = vertical\_length;

vertical\_search\_row\_1 = row;

vertical\_search\_col\_1 = col;

}

**else** **if**(vertical\_length == vertical\_longest\_length\_1)

{

vertical\_longest\_length\_2 = vertical\_length;

vertical\_search\_col\_2 = col;

}

**break**;

}

}

}

**if**(vertical\_longest\_length\_1 == vertical\_longest\_length\_2)

{

vertical\_search\_col = (vertical\_search\_col\_1 + vertical\_search\_col\_2) / 2;

}

**else**

{

vertical\_search\_col = vertical\_search\_col\_1;

}

vertical\_search\_row = vertical\_search\_row\_1;

//下面的部分用于求纵向斜率

**if**(function\_flag == 1)

{

**if**(trend\_flag == 1)

{

value = (vertical\_search\_col - r\_line[start\_row]) / (start\_row - vertical\_search\_row); //求左斜率时，减的是end\_col，求得的结果小于0。

}

**else** **if**(trend\_flag == 2)

{

value = (vertical\_search\_col - l\_line[start\_row]) / (start\_row - vertical\_search\_row); //求右斜率时，减的是start\_col，求得的结果大于0。

}

}

}

**else** **if**(function\_flag == 2) //功能标志位为2时，此函数用来找列值范围（start\_col~end\_col）内能找到顶的列数。

{

**for**(col = start\_col; col <= end\_col; col ++)

{

found\_edge\_flag = 0;

**for**(row = start\_row; row > 0; row --)

{

**if**(mt9v03x\_image[row][col] == 0 && mt9v03x\_image[row - 1][col] == 0)

{

found\_edge\_flag = 1;

**break**;

}

}

**if**(!found\_edge\_flag)

{

value ++; //当能找到第0行都没找到边界时，值加1。

}

}

}

**else** **if**(function\_flag == 3)

{

**for**(col = start\_col; col <= end\_col; col ++)

{

found\_edge\_flag = 0;

**for**(row = start\_row; row > 0; row --)

{

**if**(mt9v03x\_image[row][col] == 0 && mt9v03x\_image[row - 1][col] == 0)

{

found\_edge\_flag = 1;

vertical\_length = start\_row - row;

**if**(vertical\_length > vertical\_longest\_length\_1)

{

vertical\_longest\_length\_1 = vertical\_length;

vertical\_search\_row\_1 = row;

vertical\_search\_col\_1 = col;

}

**else** **if**(vertical\_length == vertical\_longest\_length\_1)

{

**if**(!Cross\_flag)

{

vertical\_longest\_length\_2 = vertical\_length;

vertical\_search\_col\_2 = col;

}

**else** **if**(Cross\_flag)

{

vertical\_longest\_length\_1 = vertical\_length;

vertical\_search\_col\_1 = col;

}

}

**break**;

}

}

**if**(!found\_edge\_flag) //没有找到边界，就是找到了第一行

{

vertical\_length = start\_row - 0;

**if**(vertical\_length > vertical\_longest\_length\_1)

{

vertical\_longest\_length\_1 = vertical\_length;

vertical\_search\_row\_1 = 0;

vertical\_search\_col\_1 = col;

}

**else** **if**(vertical\_length == vertical\_longest\_length\_1)

{

**if**(!Cross\_flag)

{

vertical\_longest\_length\_2 = vertical\_length;

vertical\_search\_col\_2 = col;

}

**else** **if**(Cross\_flag)

{

vertical\_longest\_length\_1 = vertical\_length;

vertical\_search\_col\_1 = col;

}

}

}

}

**if**(vertical\_longest\_length\_1 == vertical\_longest\_length\_2)

{

vertical\_search\_col = (vertical\_search\_col\_1 + vertical\_search\_col\_2) / 2;

}

**else**

{

vertical\_search\_col = vertical\_search\_col\_1;

}

vertical\_search\_row = vertical\_search\_row\_1;

**if**(trend\_flag == 1)

{

value = vertical\_search\_row;

}

**else** **if**(trend\_flag == 2)

{

value = vertical\_search\_col;

}

}

**else** **if**(function\_flag == 4) //实现功能4，确定向上纵向搜索能找到上拐点，trend\_flag用来确定对应的返回值，trend\_flag为0时返回数值“1”，表示能够找到上拐点；trend\_flag为1时返回值为上左拐点的行值，它的列值为上左拐点对应行的左边界值；trend\_flag为2时返回值为上右拐点的行值，它的列值为上右拐点对应行的右边界值。

{

//可以改变的变量

uint8 N = 3, M = 5;

//通用变量

sint16 i = 0, j = 0;

//用以找上边界值的变量

uint8 found\_boundary\_flag = 0;

uint16 Boundary\_value[MT9V03X\_W] = {0}; //定义用来存储上边界边界点的行值数组，初始时每一列的上边界边界都是第0行。

//用以找上左拐点、上右拐点的变量

uint8 may\_found\_upleft\_infpoint\_flag = 0, may\_found\_upright\_infpoint\_flag = 0; //可能找到上左拐点、上右拐点的标志位，完全确定已经找到上左拐点、上右拐点还需要进一步验证。

uint8 found\_upleft\_infpoint\_flag = 0, found\_upright\_infpoint\_flag = 0; //分别是找到上左拐点的标志位和找到上右拐点的标志位。

uint8 found\_upright\_infpoint\_end\_flag = 0;

uint16 cnt\_1 = 0, cnt\_2 = 0; //检验找到上左拐点或上右拐点的次数，超过某个值后可以判断已经找到了对应的做拐点和右拐点。

uint16 Upleft\_inflection\_point\_row = 0, Upright\_inflection\_point\_row = 0; //上左拐点所在行值和上右拐点所在行值。

uint16 Left\_linear = 0, Right\_linear = 0; //上左拐点所在行的左边界值和上右拐点所在行的右边界值。

uint16 Upleft\_inflection\_point\_row\_1 = 0, Upleft\_inflection\_point\_row\_2 = 0, Upright\_inflection\_point\_row\_1 = 0, Upright\_inflection\_point\_row\_2 = 0;

uint16 Left\_linear\_1 = 0, Left\_linear\_2 = 0, Right\_linear\_1 = 0, Right\_linear\_2 = 0;

uint8 found\_first\_upleft\_infpoint\_flag = 0, found\_end\_upright\_infpoint\_flag = 0; //两个变量分别用来判断上左拐点在第0列的情形和上右拐点在第MT9V03X\_W - 1列的情形。

uint8 enable\_left\_infpoint\_flag = 0, enable\_right\_infpoint\_flag = 0; //上左拐点找到的标志、上右拐点找到的标志

uint16 Far\_from\_midline = 0;

uint8 found\_once\_far\_flag = 0; //第一次对上左拐点和上右拐点相对于中线距离进行判断的标志位。

//找到每一列的上边界边界值

**for**(col = 0; col <= MT9V03X\_W - 1; col ++)

{

Boundary\_value[col] = MT9V03X\_H - 1;

}

**for**(col = start\_col; col <= end\_col; col ++)

{

found\_boundary\_flag = 0;

**for**(row = start\_row; row > 0; row --)

{

**if**(mt9v03x\_image[row][col] == 0 && mt9v03x\_image[row - 1][col] == 0) //找到上边界点

{

found\_boundary\_flag = 1;

Boundary\_value[col] = row;

**break**;

}

}

**if**(!found\_boundary\_flag)

{

Boundary\_value[col] = 0;

}

}

//对每一列的上边界边界值进行比较分析，确定是否存在上左拐点、上右拐点（【注】：可能只能找到上左拐点或上右拐点其中之一）

**for**(i = start\_col; i < end\_col; i ++)

{

//\*\*\*寻找上左拐点

**if**((Boundary\_value[i] > Boundary\_value[i + 1]) && (Boundary\_value[i] - Boundary\_value[i + 1] < N) && !may\_found\_upleft\_infpoint\_flag)

{

may\_found\_upleft\_infpoint\_flag = 1;

Upleft\_inflection\_point\_row = Boundary\_value[i];

Left\_linear = i;

}

**if**(may\_found\_upleft\_infpoint\_flag)

{

cnt\_1 ++;

**if**(cnt\_1 > M)

{

may\_found\_upleft\_infpoint\_flag = 0;

cnt\_1 = 0;

found\_upleft\_infpoint\_flag = 1;

}

}

**if**(may\_found\_upleft\_infpoint\_flag && (Boundary\_value[i] <= Boundary\_value[i + 1]))

{

cnt\_1 = 0;

may\_found\_upleft\_infpoint\_flag = 0;

}

//\*\*\*寻找上右拐点

**if**((Boundary\_value[i] < Boundary\_value[i + 1]) && (Boundary\_value[i] - Boundary\_value[i + 1] > -N))

{

**if**(!may\_found\_upright\_infpoint\_flag && !found\_upright\_infpoint\_flag)

{

may\_found\_upright\_infpoint\_flag = 1;

}

**if**(!found\_upright\_infpoint\_end\_flag)

{

Upright\_inflection\_point\_row = Boundary\_value[i + 1];

Right\_linear = i;

}

}

**if**(may\_found\_upright\_infpoint\_flag && !found\_upright\_infpoint\_flag)

{

cnt\_2 ++;

**if**(cnt\_2 > M)

{

may\_found\_upright\_infpoint\_flag = 0;

cnt\_2 = 0;

found\_upright\_infpoint\_flag = 1;

}

}

**if**(may\_found\_upright\_infpoint\_flag && (Boundary\_value[i] >= Boundary\_value[i + 1]) && !found\_upright\_infpoint\_flag)

{

cnt\_2 = 0;

may\_found\_upright\_infpoint\_flag = 0;

}

**if**(found\_upright\_infpoint\_flag && (Boundary\_value[i] >= Boundary\_value[i + 1]) && !found\_upright\_infpoint\_end\_flag)

{

found\_upright\_infpoint\_end\_flag = 1;

}

**if**(found\_upright\_infpoint\_end\_flag && found\_upright\_infpoint\_flag && ((Boundary\_value[i] < Boundary\_value[i + 1]) && (Boundary\_value[i] - Boundary\_value[i + 1] > -N)))

{

found\_upright\_infpoint\_end\_flag = 0;

found\_upright\_infpoint\_flag = 0;

}

//\*\*\*对找到的上左拐点和上右拐点进行判定（找到的上拐点需要满足以下特点：1.左拐点列值一定比右拐点列值小；2.左右拐点相对于中线的位置距离最近；3.左右拐点中间的列的上边界行值一定接近第0行（【注】2.的优先级高于3.的优先级））

**if**(found\_upleft\_infpoint\_flag)

{

Upleft\_inflection\_point\_row\_1 = Upleft\_inflection\_point\_row;

Left\_linear\_1 = Left\_linear;

enable\_left\_infpoint\_flag = 1;

}

**if**(found\_upright\_infpoint\_flag)

{

Upright\_inflection\_point\_row\_1 = Upright\_inflection\_point\_row;

Right\_linear\_1 = Right\_linear;

enable\_right\_infpoint\_flag = 1;

}

**if**(!found\_upleft\_infpoint\_flag && !found\_first\_upleft\_infpoint\_flag && found\_upright\_infpoint\_flag) //如果在已经找到了上右拐点的情况下却没有找到上左拐点，那么说明上左拐点的位置应该在第0列

{

found\_first\_upleft\_infpoint\_flag = 1;

found\_upleft\_infpoint\_flag = 1;

enable\_left\_infpoint\_flag = 1;

Upleft\_inflection\_point\_row\_1 = Boundary\_value[0];

Left\_linear\_1 = 0;

}

**if**(found\_upleft\_infpoint\_flag && !found\_upright\_infpoint\_flag && !found\_end\_upright\_infpoint\_flag && i == end\_col - 1)

{

found\_end\_upright\_infpoint\_flag = 1;

found\_upright\_infpoint\_flag = 1;

enable\_right\_infpoint\_flag = 1;

Upright\_inflection\_point\_row\_1 = Boundary\_value[end\_col];

Right\_linear\_1 = end\_col;

}

**if**(enable\_left\_infpoint\_flag && enable\_right\_infpoint\_flag)

{

**if**(!found\_once\_far\_flag)

{

found\_once\_far\_flag = 1;

Far\_from\_midline = abs\_1((Left\_linear\_1 + Right\_linear\_1)/2 - (MT9V03X\_W - 1)/2);

Upleft\_inflection\_point\_row\_2 = Upleft\_inflection\_point\_row\_1;

Left\_linear\_2 = Left\_linear\_1;

Upright\_inflection\_point\_row\_2 = Upright\_inflection\_point\_row\_1;

Right\_linear\_2 = Right\_linear\_1;

}

**else**

{

**if**(abs\_1((Left\_linear\_1 + Right\_linear\_1)/2 - (MT9V03X\_W - 1)/2) < Far\_from\_midline)

{

Far\_from\_midline = abs\_1((Left\_linear\_1 + Right\_linear\_1)/2 - (MT9V03X\_W - 1)/2);

Upleft\_inflection\_point\_row\_2 = Upleft\_inflection\_point\_row\_1;

Left\_linear\_2 = Left\_linear\_1;

Upright\_inflection\_point\_row\_2 = Upright\_inflection\_point\_row\_1;

Right\_linear\_2 = Right\_linear\_1;

}

}

enable\_left\_infpoint\_flag = 0; //清除找到上左拐点和上右拐点的标志位

enable\_right\_infpoint\_flag = 0;

}

}

**if**(found\_upleft\_infpoint\_flag)

{

l\_line[Upleft\_inflection\_point\_row] = Left\_linear\_2;

}

**if**(found\_upright\_infpoint\_flag)

{

r\_line[Upright\_inflection\_point\_row] = Right\_linear\_2;

}

**if**(trend\_flag == 0)

{

**if**(found\_upleft\_infpoint\_flag && found\_upright\_infpoint\_flag)

{

value = 1;

}

**else**

{

value = 0;

}

}

**else** **if**(trend\_flag == 1)

{

value = Upleft\_inflection\_point\_row\_2;

}

**else** **if**(trend\_flag == 2)

{

value = Upright\_inflection\_point\_row\_2;

}

// printf("Upleft\_inflection\_point\_row = %d\n",Upleft\_inflection\_point\_row);

// printf("Left\_linear = %d\n",Left\_linear);

// printf("Upright\_inflection\_point\_row = %d\n",Upright\_inflection\_point\_row);

// printf("Right\_linear = %d\n",Right\_linear);

}

**return** value;

}