第十一届“恩智浦”杯全国大学生
智能汽车竞赛

技 术 报 告

学　　校：南京师范大学

队伍名称：先驱者

参赛队员：徐锋　吴西伟　王鑫悦

带队教师：沈世斌　张亮

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：徐　锋

吴西伟

王鑫悦

带队教师签名：　沈世斌

张　亮

日　　期：2016 年 8 月 13 日

摘　要

本文以第十一届全国大学生智能车竞赛为背景，介绍了四轮摄像头智能车系统的软硬件结构和开发流程。该比赛采用大赛组委会统一指定的B型车模，以恩智浦半导体公司生产的32位单片机MK60FX512VLQ15为核心控制器，在IAR6.5开发环境中进行软件开发，使用 MT9V022 CMOS摄像头进行赛道信息采集。系统主要根据MT9V022采集到的赛道信息进行软件二值化后路径规划，通过舵机连接前轮实现转向控制，通过电机连接后轮实现差速及速度控制。此系统是一个软硬件与机械相结合的复杂整体，其中硬件主要包括电源管理模块、电机驱动模块、速度测量模块、辅助调试模块、图像采集处理模块、舵机控制模块和单片机模块等；软件主要包括单片机初始化程序、速度测量程序、速度设定程序、速度控制程序、舵机控制程序、图像识别程序等方面的内容，为提高在高速运行下的稳定性，进行了不同方案的设计，并使用Matlab进行了大量的数据分析以及上位机的设计调试，确定了现有的机械结构和相关控制参数。

关键词：恩智浦，智能车，MT9V022，摄像头，MK60FX512VLQ15

目　　录

引　言

全国大学生“恩智浦”杯智能汽车竞赛以“立足培养、重在参与、鼓励探索、追求卓越”为宗旨，是一项鼓励创新的科技竞赛活动。智能车设计内容涵盖了控制、模式识别、传感技术、汽车电子、电气、计算机、机械、能源等多个学科的知识，对学生的知识融合和实践动手能力的培养，具有良好的推动作用。竞赛要求在规定的汽车模型平台上，使用恩智浦半导体公司的微控制器作为核心控制模块，通过增加道路传感器、电机驱动模块以及编写相应控制程序，制作完成一个能够自主识别道路的模型汽车。

本次车模采用恩智浦半导体公司生产的32位单片机MK60FX512VLQ15为核心控制器，通过MT9V022数字摄像头采集赛道信息，自主设计并制作主控板、驱动板等功能电路，尝试使用多种转向及速度控制方案，精确控制四轮摄像头模型赛车识别赛道，并按照规定路线行进。车模设计制作过程中，深入研究车模特点，设计整体结构及控制系统，并根据实际运行情况及时调整。

硬件机械设计时，针对小车结构特点，在传感器、电路模块安装时多次尝试，及时优化；软件系统设计时，使用了经典PID以及模糊PID等方法，实现转向及速度控制。结合陀螺仪等模块，识别当前车模运行位置，不断改进，提升车模适应性，最终达到良好的效果。

本次报告主要介绍摄像头智能车硬件电路及车模机械的设计方案，以及软件控制系统建立的方式方法。

第一章　车模机械设计

根据大赛组委会规定，本届比赛摄像头组使用 B 型车模。B 型车模具有 SD5 舵机与 B540 电机，是一种相对灵活的四轮车模，对于赛道的适应性高，而且稳定易于控制，但后轮差速的调整较为繁琐复杂，且需通过硬件机械进行调整，一旦调整不合理，会使车子整体性能下降，无法正常运行。又由于摄像头采集道路信息较为全面的特点，使得本届摄像头组智能车对速度会有更高的要求。所以机械结构对于整车的性能至关重要，只有在车模拥有较好机械结构的前提下，控制算法才能发挥出其应有的效果。使智能车可以更好的适应高速运行的情况，保证其在高速时任然具有较高的机械强度与灵活度，是本次比赛研究重点。本章将介绍根据实际情况对车模机械做出的改进。

1.1 前轮定位调整

四轮智能车出现直线走偏、转弯费力、轮胎磨损快等情况时大多与轮胎安装角度有关，涉及到一个非常重要的转向轮位置角度定位问题，叫做“前轮定位”。它的作用是保障智能车直线运行时的稳定性，使其转向轻便并减少轮胎的磨损。前轮是转向轮，它的安装位置由主销内倾、主销后倾、前轮外倾和前轮前束等四个项目决定，反映了转向轮、主销和前轴等三者在车架上的位置关系。

1.1.1 主销后倾角

从侧面看车轮，转向主销(车轮转向时的旋转中心)向后倾倒，称为主销后倾角。设置主销后倾角后，主销中心线的接地点与车轮中心的地面投影点之间产生距离(称作主销纵倾移距，与自行车的前轮叉梁向后倾斜的原理相同)，使车轮的接地点位于转向主销延长线的后端，车轮就靠行驶中的滚动阻力被向后拉，使车轮的方向自然朝向行驶方向。设定很大的主销后倾角可提高直线行驶性能，同时主销纵倾移距也增大。但主销纵倾移距过大，会使转向沉重，而且由于路面干扰而加剧车轮的前后颠簸。

1.1.2 主销内倾角

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车模前部向上抬起一个相应的高度，这样车模本身的重力有使转向车轮回复到原来中间位置的效应，因而车模容易回正。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，从而减小转向时舵机上的力，使转向更加轻便，同时也可减少从转向轮传到舵机上的冲击力。

1.1.3 前轮外倾

从前后方向看车轮时，轮胎并非垂直安装，而是稍微倾倒呈现“八”字形张开，称为负外倾，而朝反方向张开时称正外倾。车模一般将外倾角设定得很小，接近垂直。若设定大外倾角会使轮胎磨偏，降低轮胎摩擦力。

1.1.4 前轮前束

四轮定位前束值脚尖向内，所谓“内八字脚”的意思，指的是左右前轮分别向内。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，转向变得容易。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。

在智能车程序基本完整，可以完成各类赛道元素后，根据实际情况对前轮定位进行调整。通过改变上横梁垫片的数目，增大后倾角。在智能车长期调试后，发现适当增大内倾角，可使转弯时车轮和地面有更大的接触面积，继而增大车与地面的摩擦程度，使车转向更灵活，减小因摩擦不够而引起的转向不足的情况。

1.2 舵机安装调整

目前舵机安装主要有两种方式，一种是卧式安装，一种是立式安装。车模默认的卧式安装方式会使左右两边轮子连杆不等长，易造成舵机对左右两边转向响应时间不一样，目前使用较少。另外一种较为普遍的卧式安装方式，改进了以上缺点，相对减小负载力矩，加快舵机的响应速度。而立式安装方式具有负载力矩大，响应区间广，控制精度高的特点。

根据杠杆原理，将与舵机连接的舵盘适当加长，再将转向传动杆连接在加长的输出盘的末端，就可以在舵机输出较小的转角下，取得更大的前轮转角，提高舵机的响应的速度。结合舵机重心，结构件安装等问题，最终使用了立式安装方式。舵机安装效果如图 1.1 所示。

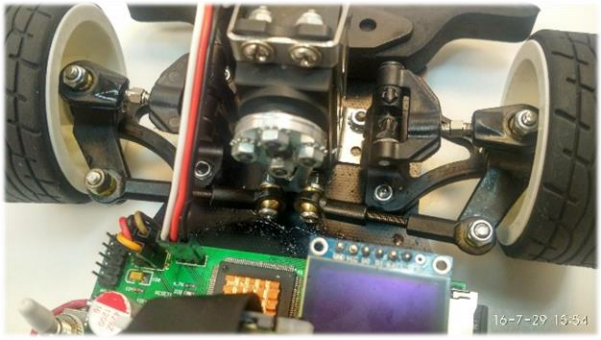

图 1.1　舵机安装效果图

1.3 摄像头安装调整

本次车模采用 MT9V022 数字摄像头采集赛道信息。作为最主要的道路信息采集传感器，摄像头的安装方式是否稳固，是否合理，是其能否采集到优质图像的关键。经过多次尝试调整，最终使用一根碳素杆搭架摄像头的方式，以减轻可能产生的不必要的重量。使用防松螺母将摄像头支撑杆底部固定在车模转向的轴线上，保证杆子与车身保持垂直。同时，碳素杆固定在车身中心位置，使车模转向时不存在水平分量，保证图像的稳定性。

摄像头位于碳素杆较高位置时，会提高车模重心位置，为使设想头在碳素杆较低位置时任然可以采集到较多的图像信息，故使用广角镜头增加摄像头视野宽度，使摄像头可以安装在一个相对低的位置，降低重心。在摄像头采集图象时，周围的光照对摄像头采集的图像会有影响，摄像头会接收到来自镜头之外的光，对赛道图像的正常采集产生影响，所以对摄像头进行了处理。

1.4 编码器安装调整

为使智能车能够平稳地沿着赛道运行，需要对车速进行精确控制，使赛车在急转弯时速度不至过快而冲出赛道。理论上，通过控制驱动电机上的平均电压即可控制智能车速度，但是如果开环控制电机转速，会受很多因素影响，例如电池电压、电机传动摩擦力、道路摩擦力和前轮转向角度等。这些因素会造成赛车运行不稳定，需要检测出智能车当前速度。通过速度检测，对车模速度进行闭环反馈控制，即可消除上述各种因素的影响，使得车模运行得更稳定。

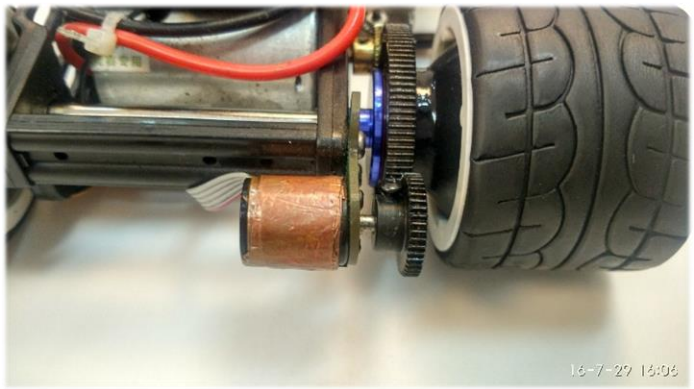

图 1.2　旋编安装效果图

本次使用绝对式 mini 编码器，根据车模自身实际结构，自行制作了编码器的安装结构件，使编码器使用时不对电池的安装产生影响。在使用时注意到，编码器容易受到静电或连接松动的影响，所以需要对编码器进行保护，在编码器各连接处进行加固处理，防止以上情况发生。编码器安装效果如图 1.2 所示。

1.5 车模重心调整

车模重心位置对赛车加减速性能、转向性能和稳定性都有较大影响，重心调整主要包括重心高度和前后位置的调整。智能车车身较轻，重心较低时可以增加智能车高速运行时的稳定性。除此之外，车辆重心前后方向的调整，对赛车行驶性能也有很大的影响。根据车辆运动学理论，车身重心前移，会增加转向，但降低转向的灵敏度（因为大部分重量压在前轮，转向负载增大），同时降低后轮的抓地力，影响加减速性能；重心后移，会减少转向，但增大转向灵敏度，后轮抓地力也会增加，提高加减速性能。因此，确定合适的车体重心，让车模更加适应比赛赛道是很关键

的。

基于以上思路，对智能车车模重心进行调整。首先，通过不断简化主板、驱动等自行制作的电路板的重量，并根据安装实际情况，设计电路板接线位置，减少接线长度，尽可能减少连接线引起的不必要的重量。同时，将车身板在不改变其结构的前提下进行适当打磨，将车身板上各类元器件尽可能接近车身板，降低车模重心，最终使车模除摄像头安装较高，其他各个部件安装高度都很低。安装时减少车模车身板打孔数目，并将各类器件均匀分布安装在车身上，而不是集中在车身中心，这样使车模前后轮位置都具有一定重量，保证车模在高速运行时具有一定的抓地力，使车模可以快速稳定运行。

第二章　硬件电路设计

硬件电路的可靠性是车模正常运行的前提，在尝试使用多种硬件电路后，最终确定了一套可靠、高效的设计方法，并可满足强劲的性能需求。实现硬件电路可靠、高效的要求后，再将电路设计尽量简洁，尽量减少元器件使用数量，以缩小电路板面积，减轻电路重量，从而减轻整车重量，降低车模重心。

2.1 电源管理电路设计

目前主要使用的电源分为开关电源和线性电源。线性电源的电压反馈电路是工作在线性状态；开关电源是指用于电压调整的管子工作在饱和和截至区即开关状态的。线性电源一般是将输出电压取样然 后与参考电压送入比较电压放大器，此电压放大器的输出作为电压调整管的输入，用以控制调整管使其结电压随输入的变化而变化，从而调整其输出电压。开关电源是通过改变调整管的开和关的时间即占空比来改变输出电压的。从其主要特点上看：线性电源技术很成熟，制作成本较低，可以达到很高的稳定度，波纹也很小，而且没有开关电源具有的干扰与噪音，开关电源效率高、损耗小、可以降压也可以升压，但是交流纹波稍大些。电源管理模块是智能车控制系统的基础，因此选择使用可靠、合理、实用的电源管理模块在设计控制系统时至关重要。

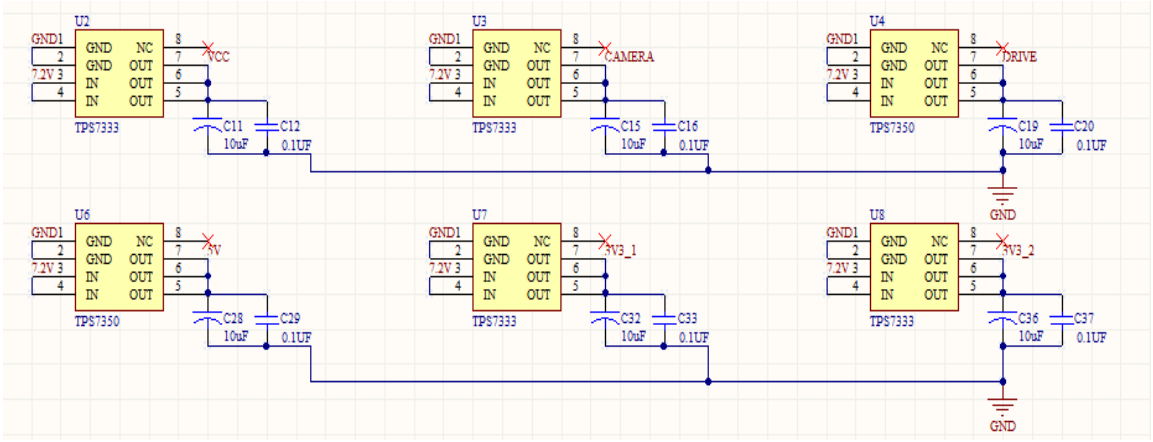

图 2.1 供电管理模块原理图

根据需求，系统中电源稳压电路分别有+3.3V、+5V、+6.5V、+7.2V、+12V 供电。其中，单片机供电、摄像头供电、陀螺仪供电、五向按键供电、TF 卡供电使用+3.3V;、显示屏供电、编码器供电使用+5V；舵机供电使用+6.5V；+12V 和+5V 共同给电机驱动电路中 IR2104 供电。供电模块电路原理图分配图如图 2.1、图 2.2 所示。

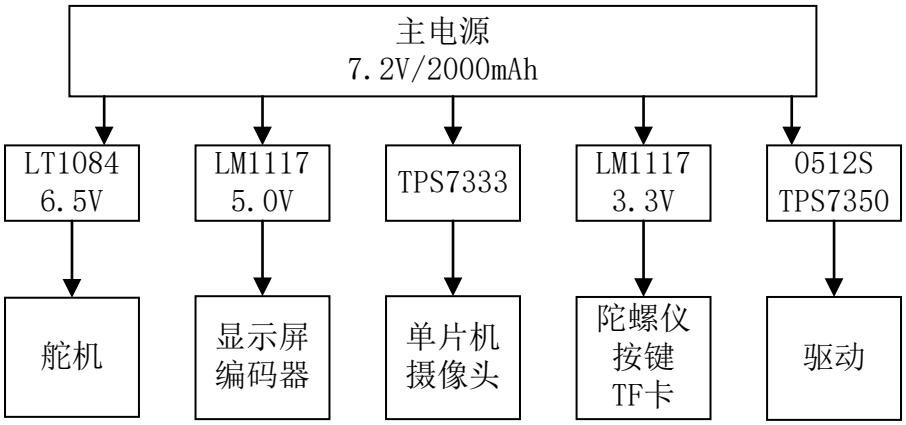

图 2.2　供电管理模块分配图

线性稳压芯片具有稳定度高、波纹小、干扰与噪音小的特点，而开关电源具有效率高、损耗小、纹波大的特点。考虑到当直流电机在高速运行时，电池输出电流增大，输出电压减小的情况。为提高系统在高速运行时的稳定性，选择具有完善的过流、过压、电压反接保护功能的 TPS 系列低压差线性稳压芯片为单片机、摄像头、驱动模块供电。这个系列的芯片只需要极少的外围元件就能构成高效稳压电路。具有更低的工作压降和更小的静态工作电流，更好的满足供电稳定的需求。

2.2 单片机最小系统电路设计

本次比赛使用飞思卡尔公司的 32 位单片机 MK60FX512ZVLL15。该单片机功能较 XS128 系列、KL26 系列单片机具有更强的性能。具有独立的闪存 Bank，可以并发执行代码和固件更新，既不会影响性能，也无需复杂的编码例程。多达 32 通道的 DMA 用于外设和存储器，可以降低 CPU 负载，实现更快的系统吞吐。

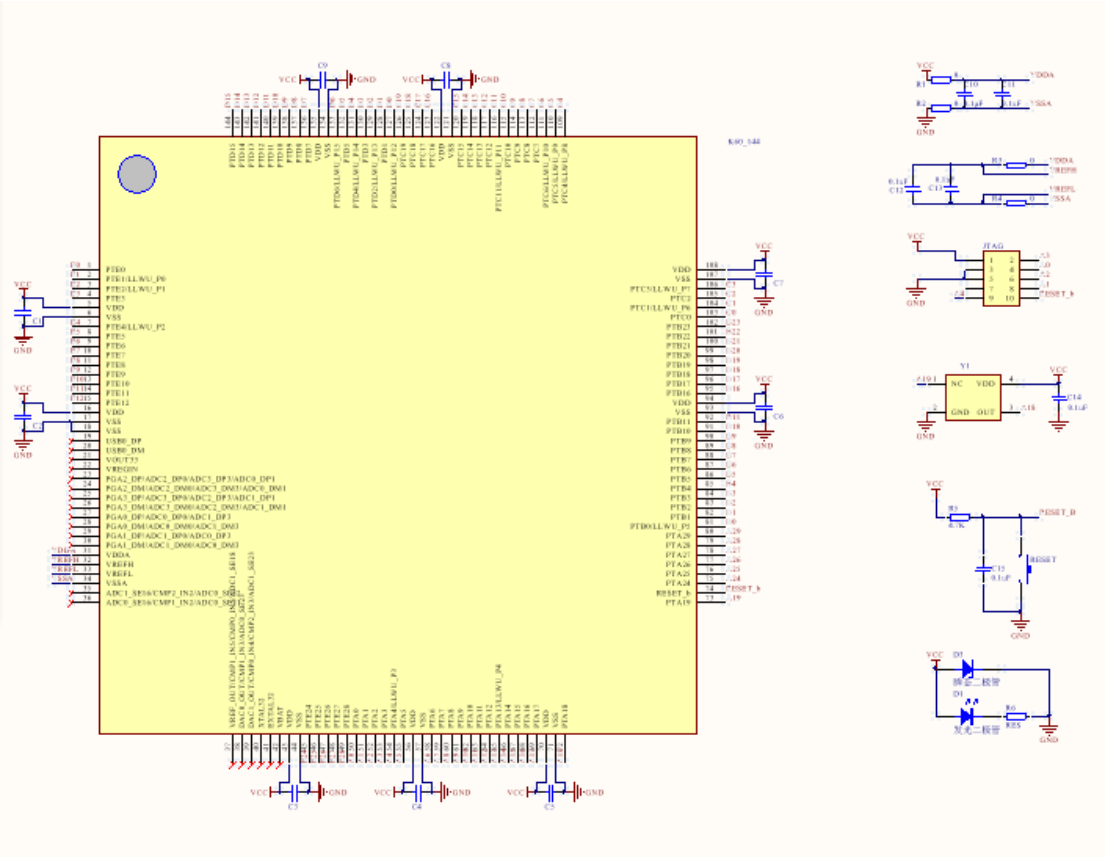

图 2.3 最小系统原理图

本次自行设计最小系统，使用 144PIN QFP 封装，并集成在主板上，主板上仅将所用到的引脚引出，与其他模块在主板上直接连接。单片机最小系统原理图如 2.3 所示。该芯片上许多引脚具有复用功能。这些引脚在复位后，立即被配置为高阻状态，且为通用输入引脚，没有内部上拉电阻。需要注意的是，为了避免来自浮空输入引脚额外的漏电流，应用程序中的复位初始化例程需尽快使能上拉或下拉，也可改变不常用引脚的方向为输出，以使该引脚不再浮空。

在使用自行设计最小系统电路时，会遇到有某些功能无法使用的情况，如原理图没有错误，需要注意在电路制作时可能出现的错误。先要确保使用元器件可靠易用，尽量不多次重复使用某一元器件。在绘制 PCB 时，布局布线最好都手动完成，不使用自动布线，开始布局时就要考虑到器件的位置, PCB 叠层的安排，重要连线

的走法，元器件封装选择等问题。布局时，模拟电路和数字电路要分区域放置，使得模拟电路在模拟电路区域，数字电路在数字区域内，这样就不会相互影响。模拟地和数字地处理的出发点是类似的，不能让数字信号的回流流到模拟地上去。模拟电路对地的主要要求是，完整、回路小、阻抗匹配。数字信号如果低频没有特别要求；如果速度高，也需要考虑阻抗匹配和地完整。使用高速逻辑器件时，走线不对称或者有比较大的差时，容易因为时延造成错误的逻辑。

由于单片机超频使用，程序容易停在单片机时钟初始化处，大部分是由于单片机晶振不起振或频率异常造成的，故在晶振电路设计时要特别注意。除原理、元器件质量问题，晶振电路的走线过长，晶振两脚之间有走线，外围电路也会对晶振使用产生影响。在排除电路错误的可能性后，在 PCB 布线时，晶振尽量与单片机在同一层，晶振电路的走线应尽量短且尽可能靠近单片机，杜绝在晶振两脚间走线，并尽量减少过孔使用。

2.3 电机驱动电路设计

本次电机驱动电路设计时尝试使用了两种方式：N 沟道 MOSFET LR7843 和专用栅极驱动芯片 IR2104 组成的驱动电路，以及采用集成电机驱动芯片BTN7971组成的驱动电路。

MOSFET 驱动电路：

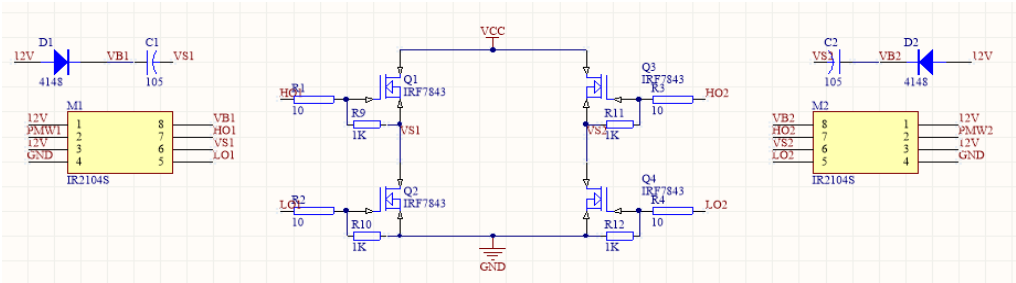

图 2.4　　MOS 驱动原理图

用 LR7843 MOS管搭建 H桥电路，配合使用IR2104控制MOSFET栅极导通，构成了直流电动机可逆双极型桥式驱动电路。驱动电路原理图如图2.4所示。

BTN 驱动电路：

使用两片集成电机驱动芯片 BTN7971 即可构成一个全桥电路驱动电机便可驱动电机转动，且相对以往使用的 BTN7970 性能更加卓越。驱动电路原理图如图 2.5 所示。

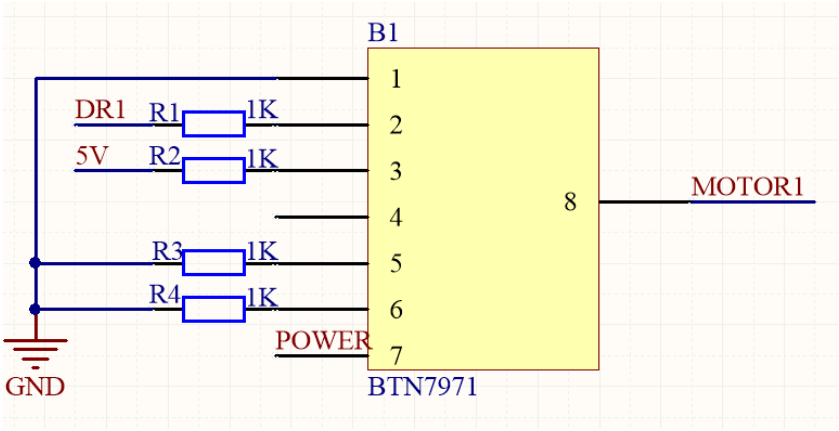

图 2.5 BTN 驱动原理图

经过对以上两种驱动电路制作及具体使用后，最终选择 MOS 驱动电路。LR7843 到同时最大内阻仅为 3.3mΩ，其电流量大，加减速性能好，使用贴片型同样也可以使驱动电路很简洁。由于流过电机的电流相对较大，在绘制驱动电路 PCB 时，通过大电流的线路要尽量短粗，并且尽量避免经过过孔，如果要经过过孔的话要把过孔做大一些（>1mm）并且在焊盘上做一圈小的过孔，在焊接时用焊锡填满，否则可能会烧断。

第三章　软件系统设计

完善的软件控制系统是智能车高速稳定运行的核心。通过使用数字式摄像头采集赛道信息，并对采集的图像进行矫正处理，以获取准确的道路信息。通过分析采集到的图像数据，对智能车的转向和速度进行控制，使用经典 PID 控制算法，并结合实际情况不断完善修改，实现对摄像头智能车的精确控制。

3.1 软件控制流程

软件控制流程图如图 3.1 所示。对摄像头采集图像进行预处理，之后根据图像搜索赛道边界，并计算赛道中心，由此编写程序获得所需偏差，实现转向与速度控制。

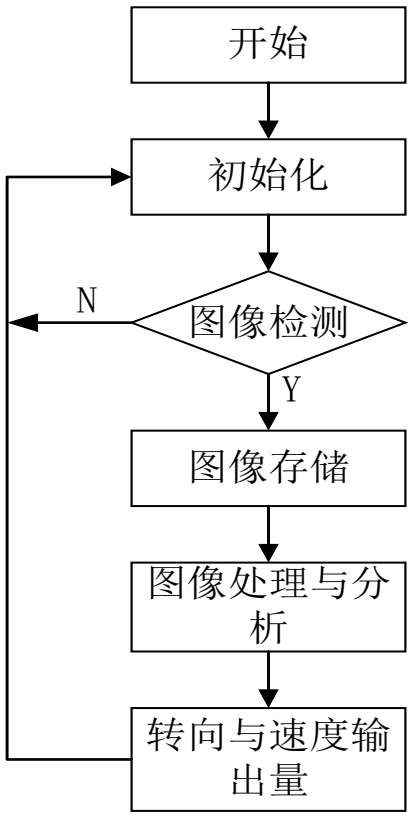

图 3.1　初始化流程图

3.2 图像采集处理

本届比赛赛道由白色 PVC 跑道与黑色边沿组成。数字摄像头可直接采集赛道灰度值，其中灰度值高的为白色，灰度值低的为黑色。因此在图像中白色与黑色灰度值会有较大区别，在有明显区别处分割阈值，进行图像的二值化处理。对原始图像进行逐行扫描，根据设定的阈值从二值化图像中部向两边搜寻黑白跳变点，从而确定左右边界位置，处理效果如图 3.2。

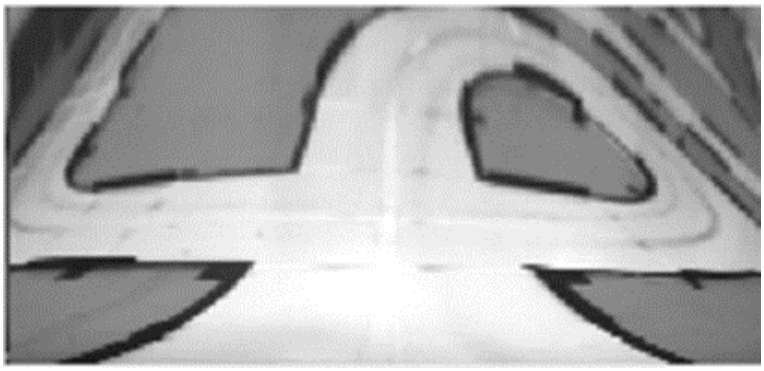

图 3.2　图像采集处理图

3.3 图像矫正处理

为通过摄像头获取更多的赛道信息，将摄像头安装在车身上方位置，同时，由于摄像头光轴与地面呈一定夹角，会使摄像头采集图像中赛道会呈现“近大远小”的情况，即摄像头采集的图像中存在梯形失真的情况。梯形失真的存在会增加图像处理的难度。通过分析摄像头采集的图像，并使用 Matlab 编写程序，对摄像头进梯形校正，矫正效果如图 3.3。

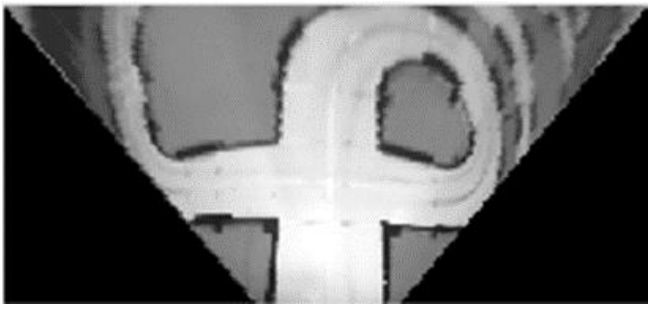

图 3.3　图像矫正效果图

3.4 黑线边界处理

通过对采集到的图像进行梯形校正后，由提取的赛道边沿数据推算中心。当左右两边都有效时计算出中心作为中心点。当单边有有效数据时，通过平移赛道宽度一半的距离矫正单边数据，将其作为中心点。当左右边沿点总数少于设定值时舍弃。推算完整幅图像中心点后，将中心点进行处理，保证其连续性。处理效果如图 3.4。

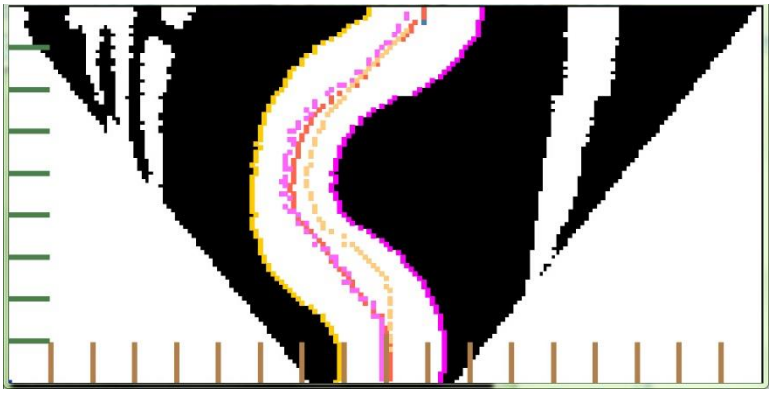

图 3.4　黑线边界处理图

3.5 速度控制处理

速度选用闭环控制方式。通过闭环反馈使系统的精确度提高, 响应时间缩短, 控制稳定性高。而开环控制没有反馈环节，稳定性、精确度不高。为了达到好的速度控制效果，速度控制要选用闭环控制方式。

由于设定速度主要由道路与直道的偏差来决定，道路越接近直道，设定速度越高，反之越低。设定速度控制系统中的最低速度，需令系统以较低的速度匀速行使，在保证安全的前提下，逐渐提高匀速行使的速度，直到系统出现不稳定行为，此速度再减去一个安全量，即为设定的最低速度。最高速度需在确定最低速度以后，加入变速策略，不断提高最高速度的设定值，直到系统出现不稳定行为，此速度再减去一个安全量，即为设定的最高速度。即最低速度等于匀速运行时的最高速度，最高速度等于变速运行时的最高速度。

基于以上速度控制策略，由单片机在每次中断程序中读取当前旋转编码器的计

数值，然后与上一次计数值相减即可得到当前速度值。并根据摄像头采集图像，对当前道型判断，对速度进行控制。速度控制的最高速度和最低速度需根据不同的赛道组合类型设定，没有明确的最优值。经过实际运行测试，使用 PID 控制算法能够很好的控制小车的运行速度，并且在运行过程中也非常稳定。对于系统运行过程中难免出现的“失去道路”的情况，对此需要采取一定的安全策略，防止系统“盲跑”而导致车身结构因撞击发生改变。处理效果如图 3.5 所示。

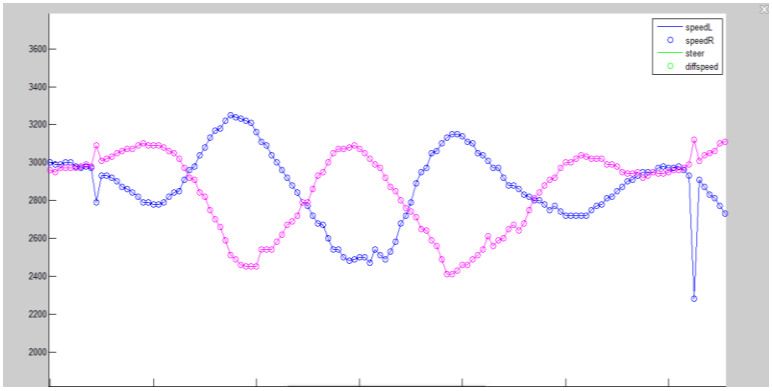

图 3.5　速度控制处理图

3.6 转向控制处理

智能车能够在高速情况下，根据各类赛道及时动作，主要取决于对智能车转向的控制。在转弯时，合适的转向角度可以使智能车具有更好的路径，同时，也可以提高智能车运行时的稳定性，缩短时间。根据实际情况，采用图像计算值加权的算法。通过计算权值，使车模切赛道内道，提高了车模速度。

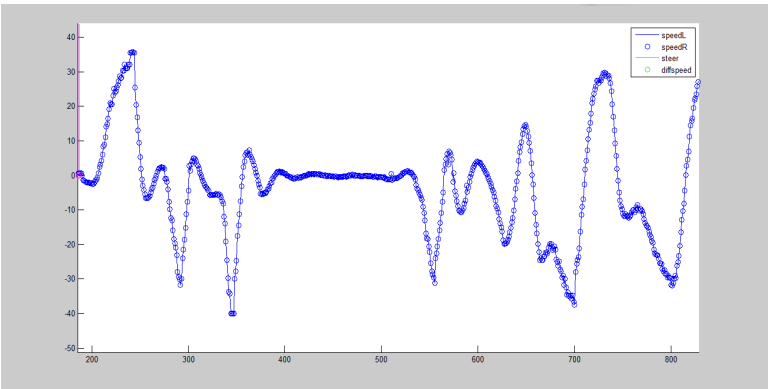

图 3.6　转向控制处理图

对整场图像计算出的有效行进行加权算法处理，对每一行图像乘以一组权值，从而计算出新的赛道中心，可以使有效获得更好的数据。但对整幅图像都乘以求出的加权平均值计算出的中线数据，不能很好的处理高速运行时采集的图像，故对有效行及加权方式进一步优化。减小车模近处权重，增加远处权重。需要注意的是，图像的长度越多，可以采集到的有效行就越多，增加图像的长度就很有必要，在不损失近处图像的同时将摄像头尽量抬高，并使用广角镜头，即可采集到更多有效数据。

提前判断道路，需要车模具有更远的前瞻，而前瞻越长，计算中线时越可能受到赛道周围或其他信号的干扰而使车模运行出错，所以在控制舵机时要通过软件消除。当计算出新的中线信息时，要与上次计算值进行比较，防止运行时出现较大错误。转向控制根据前瞻不同参数不同，需要根据车模运行表现做相应的调整，远端的参数影响入弯的时机，但也影响车模运行在直道和弯道时的稳定性，近端的参数影响过弯的路线，这些参数需要经过反复调试才能得到较完美的参数。处理效果如 3.6 所示。

第四章　系统开发及调试工具

整个智能车是一个复杂的系统模型，所以在开发中使用合适有效的调试工 具进行测试与调试显得尤为重要。为此使用了 TF 卡存储和在线调试两种方法，并且结合 EEPROM 掉电不丢失数据的特性进行数据保存，在调试中收到了很好的效果。

4.1 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。

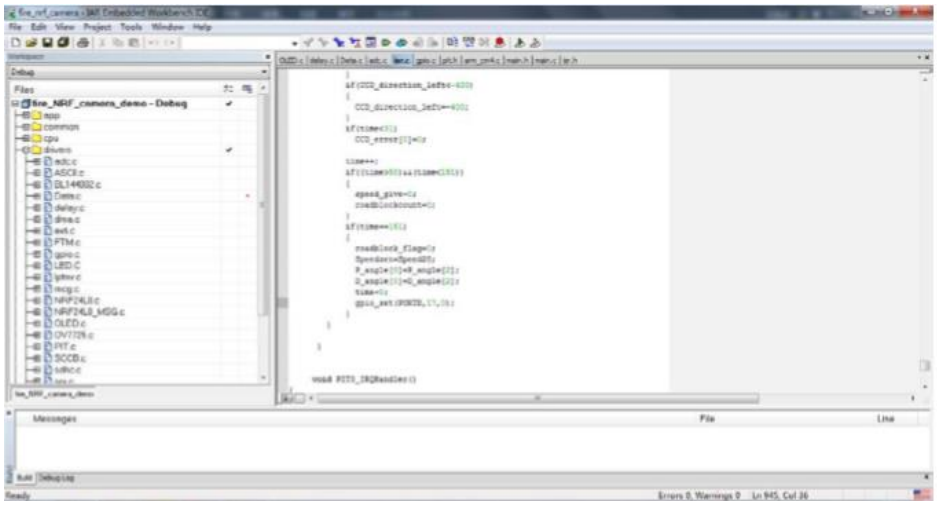

图 4.1　IAR 调试界面

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。

4.2 MATLAB 数据处理

MATLAB 是矩阵实验室（Matrix Laboratory）的简称，是美国 MathWorks 公司出品的商业数学软件 ，用于算法开发、数据可视化、数据分析以及数值计算的高级技术计算语言和交互式环境，主要包括 MATLAB 和 Simulink 两大 部分。本次主要使用 MATLAB 完成数据的可视化处理以及算法仿真。通过蓝牙串口将需要观测的数据发至电脑，使用 MATLAB 进行绘图处理。根据采集到的数据进行相关参数的整定。

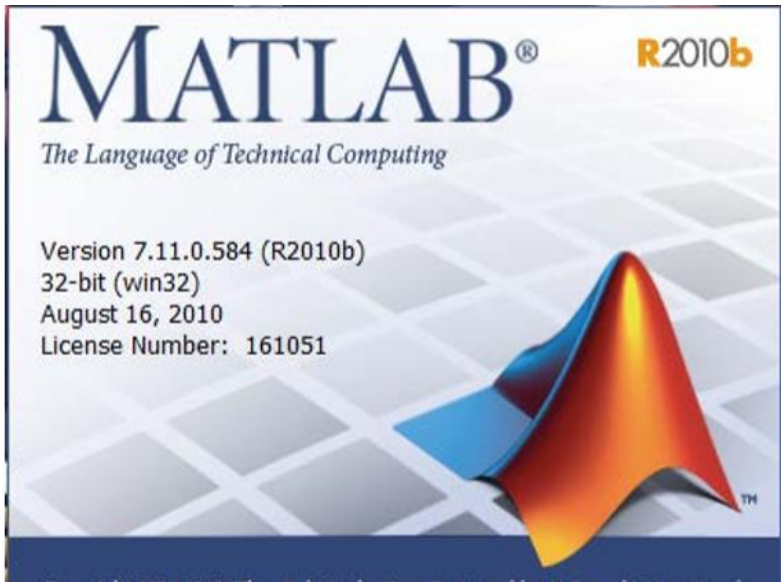

图 4.2　Matlab 软件界面图

4.3 TF 卡数据存储

TF卡 (Micro SD卡) 是一种极细小的快闪存储器卡，其格式源自 SanDisk 创造，原本这种记忆卡称为 T-Flash，及后改称为 Trans Flash。其主要应用于移动电话，但因它的体积微小和储存容量的不断提高，已经使用于GPS设备、便携式音乐播放器和一些快闪存储器盘中。它的体积为 15mm x 11mm x1mm ，差不多相等于手指甲的大小，是现时最细小的记忆卡。本次使用TF卡存储图像等数据。

Pin	SD Mode			SPI Mode		
	Name	Type(1)	Description	Name	Type	Description
1	DAT2	I/O/PP	Data Line [Bit 2]	RSV		Reserved
2	CD/DAT3(2)	I/O/PP (3)	Card Detect / Data Line [Bit 3]	CS	I	Chip Select (neg true)
3	CMD	PP	Command/Response	DI	I	Data In
4	V_{DD}	S	Supply voltage	V_{DD}	S	Supply voltage
5	CLK	I	Clock	SCLK	I	Clock
6	V_{SS}	S	Supply voltage ground	V_{SS}	S	Supply voltage ground
7	DAT0	I/O/PP	Data Line [Bit 0]	DO	O/PP	Data Out
8	DAT1			RSV		Reserved

表4.1 TF卡引脚定义

第五章　车模参数汇总

本届比赛使用 B 型车模搭架摄像头智能车，按照组委会要求设计车模车身结构及制作硬件电路，车模具体参数如图 5.1 所示。

图 5.1　车模参数表

第六章　心得总结

智能汽车是交通现代化的重要标志之一。智能车辆是一个集环境感知、规划决策、多等级辅助驾驶等功能于一体的综合系统，它集中运用了计算机、现代传感、信息融合、通讯、人工智能及自动控制等技术，是典型的高新技术综合体。我们则做的是智能汽车的微缩模型。在整个制作车模，参与比赛的过程中，加强了动手实践能力，丰富了专业知识积累，在整个过程中都得到了锻炼。不仅是知识方面对的，更是做人做事方面的 。

整个车模制作过程，我们进行了很多次尝试，在研究自身已有软硬件设计基础上，进行改进、发展，并研究各个兄弟高校研究制作智能车的宝贵经验，并与自身设计，弥补不足，缩小差距。从车模理论设计、实际制作、整车调试，再到比赛现场调试，每一个环节都需要团队成员一起努力。到赛场上的车模只有一个，但它承载的是每位为智能车付出同学的梦想与希望。

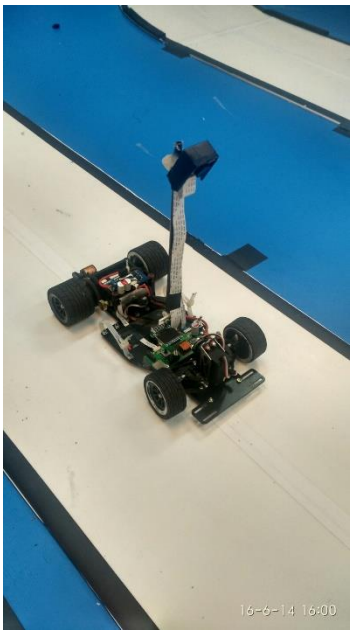

图 6.1 车模整体图

全国大学生智能车竞赛举办至今已有 11 年了，这十一年，见证了多少关于小车的欢笑与泪水，这十一年，体验了多少关于青春的快乐与悲伤。而我们从开始接触智能车到现在已经两年了，不管中间的过程如何，我们都是身有体会，对其有着特殊的情怀。来到赛场上的每一辆车都是有灵魂的，它们身上的每一颗螺丝，每一点焊锡，每一种速度，每一种形象，都凝结着每一个智能车人夜以继日的辛劳与记忆。

车模在赛场上终将完成它的使命，但关于智能车的故事还将继续。不论智能车承载了多少关于梦的希望，愿以后回想起这一切的时候，不只记得它将带来的荣耀或屈辱，还能够记得曾有段青春叫智能车。

参考文献

[1]康华光.电子技术基础模拟部分[M].北京:高等教育出版社,2006,第五版.

[2]罗华飞.MATLAB GUI 设计学习手记[M].北京:北京航空航天大学出版社,2010.

[3]谭浩强.C程序设计[M].北京:清华大学出版社,2003.

[4]谢文策.线阵CCD图像不失真采集算法及实现[J].信息通信,2009/9.

[5]罗华飞.MATLAB GUI设计学习手记[M].北京:北京航空航天大学出版社,2010.

[6]梁慧冰,孙炳达编著.现代控制理论[M].北京:机械工业出版社,2007,第二版.

[7]屠运武,徐俊艳,张培仁等.自平衡控制系统的建模与仿真[J].系统仿真学报,2004.

附录 A 控制程序代码

```
void SpeedControl(void)
{
  static int16 velocity_array_R[Avg_count]={0};
  static float32 DeviateInt_R=0;
  static float32 DeviateInt_SumR=0;
  static float32 SpeedAverOut_R=0;
  static int16     R_Speednow=0;
  static float32 Speeed_UdR[3]={0};
  static float32 Speeed_KpR[3]={0};
  static int16     Speedold_R[5]={0};
  static u8          Speederr=0;
  static u8          SpeedControlCount=0;
  int16 i=0;
  SpeedControlCount+=PITTime;
  R_Speednow+=ftm_quad_gets();
  if(!StopFlag&&(trackstart)&&(R_Speednow==0)&&(TimeStartCount>5000))
  {
    ++Speederr;
    if((Speederr>=50)&&(imgmode!=1))
    {
      Speederr=0;
      if(DirectionControlFlag&&SpeedControlFlag)
      {
        ring_off();
        DisableInterrupts;
        FTM0CH2_Out(0);FTM0CH3_Out(0);
        OLED_Clear();
        if(!Stopflag)
        OLED_ShowString12(30,20,CodeInfo);
        OLED_ShowString16(10,32,TimeInfo);
        OLED_ShowString16(10,48,PulsInfo);
        if(StartCtrlMod&&Stopflag)
          OLED_ShowNum(60,32,ImgCount,5,16);
        else
          OLED_ShowNum(60,32,TimeCount,5,16);
        OLED_ShowNum(60,48,Hardcount,5,16);
        OLED_Refresh_Gram();
        FTM2CH0_Out((int)(DirectionOutMid));
        while(1);
      }
    }
  }
  else Speederr=0;
```

```
  if((SpeedControlCount>=SpeedControlPeriod))
 {
     SpeedControlCount=0;
     PreSpeedControlOut_R=SpeedControlOut_R;
     PreSpeedControlOutNew_R=SpeedControlOutNew_R ;
     SetSpeedMaxOld=SetSpeedMax;
     if(ErrErr>20.0f)
        ErrErr=20.0f;
     else if(ErrErr<-20.0f)
        ErrErr=-20.0f;
     FuzzyErr=Error/33.33f;
     FuzzyErrErr=ErrErr/10.0f;
     if(StrightAcc&&!EdLine&&!PoDaoFlag)
     {
        SetSpeedMax=StrightSpeed;
     }
     else if(PoDaoFlag&&!Stopflag)
        SetSpeedMax=PoDaoSpeed;
     else if(Stopflag&&EdLine)
        SetSpeedMax=0;
     else
        SetSpeedMax=FuzzySpdSet();
     //SetSpeedMax=220;
     SetSpeedMax=(int16)(0.2*SetSpeedMaxOld+0.8*SetSpeedMax);
     SetSpeed=SetSpeedMax;
     if((!trackstart)&&((TimeStartCount>RunWaitTime) && (!SDOrNormFlag)) ||
((TimeStartCount>RunWaitTime)&&(SDOrNormFlag)))
     {
        trackstart=1;
     }
     if(StartCtrlMod&&trackstart&&(TimePITCount>10000)&&!StLine)
StLine=1;

     if(TimeCount>=StartCheckTime)
        Stopflag=1;
     if(!StartCtrlMod&&Stopflag&&(displaycheck<=20))
     {   //OLED_Clear();
        trackstart=0;
        DeviateInt_R=0;
        DeviateInt_SumR=0;
        //SetSpeed=1;
        displaycheck++;
        OLED_Clear();
        OLED_ShowString16(10,32,"Time's Up!!!");
        OLED_ShowNum(60,48,Hardcount,5,16);
```

```
        if(displaycheck>19)
          OLED_Refresh_Gram();
      }

      if(StartCtrlMod&&Stopflag&&(displaycheck<=20))
      {      //OLED_Clear();
        DeviateInt_R=0;
        DeviateInt_SumR=0;
        //SetSpeed=1;
        displaycheck++;
        OLED_Clear();
        if(EdLine)
          OLED_ShowString16(10,32,"EdLine Stop!!!");
        else
          OLED_ShowString16(10,32,"EdLine Error!!!");
        OLED_ShowNum(60,48,ImgCount,5,16);
        OLED_ShowNum(60,0,Hardcount,5,16);
        if(displaycheck>19)
          OLED_Refresh_Gram();
      }
      Speedold_R[4]=Speedold_R[3];
      Speedold_R[3]=Speedold_R[2];
      Speedold_R[2]=Speedold_R[1];
      Speedold_R[1]=Speedold_R[0];
      Speedold_R[0]=R_Speednow;
      if((R_Speednow-Speedold_R[4])>SpeedError)
R_Speednow=Speedold_R[4];
      if(R_Speednow>GetSpeedMax) R_Speednow=GetSpeedMax;
      if(!Stopflag&&!EdLine&&(R_Speednow<GetSpeedMin))
R_Speednow=GetSpeedMin;
      for(i=1;i<Avg_count;i++)
      {

        velocity_array_R[i-1]=velocity_array_R[i];
      }
      velocity_array_R[Avg_count-1]=R_Speednow;


      R_Speednow=0;
      for(i=0;i<Avg_count;i++)
      {
        R_Speednow+=velocity_array_R[i];
      }

      if(SpeedAverOut_R>SpeedAveMax)          SpeedAverOut_R=SpeedAveMax;
      else if(SpeedAverOut_R<SpeedAveMin) SpeedAverOut_R=SpeedAveMin;
```

```
      SpeedControlOut_R=PreSpeedControlOut_R+SpeedAverOut_R;
      SpeedControlOut_R=SpeedControlOutNew_R;
    if(SpeedControlOutNew_R<0)
    {
      if(PoDaoFlag)
        PWMOUTMINTemp=3.0f*DeviateSpeed_R[0];
      else
        PWMOUTMINTemp=1.5f*DeviateSpeed_R[0];
    }
    else
       PWMOUTMINTemp=PWMOUTMIN;
     if(!Stopflag&&!EdLine)
      {
       if(SpeedControlOut_R>PWMOUTMAX)
{SpeedControlOut_R=PWMOUTMAX;}
       else
if(SpeedControlOut_R<PWMOUTMINTemp)
{SpeedControlOut_R=PWMOUTMINTemp;}
      }
      else if(Stopflag&&EdLine)
      {
       if(SpeedControlOut_R>PWMOUTMAX)
{SpeedControlOut_R=PWMOUTMAX;}
       else                    if(SpeedControlOut_R<(PWMOUTMIN-1000))
{SpeedControlOut_R=PWMOUTMIN-1000;}
      }

     if(Stopflag&&!EdLine)
     {
       SpeedControlOut_R=0;
       Speeed_UdR[2]=0;
       DeviateSpeed_R[0]=0;
     }

     Temp_Save=(int16)(Forward_Angle+10000);          //          Forward_Angle
Speeed_UdR[2]   Speed_KpTmp
     SpeedLR[2]=(u8)(Temp_Save&0x00ff);
     SpeedLR[3]=(u8)((Temp_Save&0xff00) >> 8);

     Temp_Save=(int16)(Speednow+1000);
     SpeedLR[4]=(u8)(Temp_Save&0x00ff);
     SpeedLR[5]=(u8)((Temp_Save&0xff00) >> 8);

     Temp_Save=(int16)(Speeed_KpR[2]+10000);
     SpeedLR[6]=(u8)(Temp_Save&0x00ff);
```

```
        SpeedLR[7]=(u8)((Temp_Save&0xff00) >> 8);

        Temp_Save=(int16)(DeviateInt_SumR+10000);
        SpeedLR[8]=(u8)(Temp_Save&0x00ff);
        SpeedLR[9]=(u8)((Temp_Save&0xff00) >> 8);

        Temp_Save=(int16)(10000+(GyroValXx));
        SpeedLR[10]=(u8)(Temp_Save&0x00ff);
        SpeedLR[11]=(u8)((Temp_Save&0xff00) >> 8);

        Temp_Save=(int16)(SpeedControlOut_R+10000);
        SpeedLR[12]=(u8)(Temp_Save&0x00ff);
        SpeedLR[13]=(u8)((Temp_Save&0xff00) >> 8);

        Temp_Save=SetSpeed+1000;
        SpeedLR[14]=(u8)(Temp_Save&0x00ff);
        SpeedLR[15]=(u8)((Temp_Save&0xff00) >> 8);
    }
    if(SpeedControlFlag==0)
    {

        DeviateInt_R=0;
        DeviateInt_SumR=0;
        SpeedControlOutNew_R=0;
        return ;
    }
}
```