第十四届“恩智浦”杯全国大学生智能汽车竞赛

变形金刚三轮组技术报告

学　　校：武汉纺织大学

队伍名称：BBO

参赛队员：赵承浩、傅泉砜、肖华煦

指导老师：刘文琮、张明

授权说明：

目录：

第一章　引言

1.1　大赛指导思想与目的

全国大学生智能汽车竞赛是以智能汽车为研究对象的创意性科技竞赛，是面向全国大学生的一种具有探索性工程实践活动，是教育部倡导的大学生科技竞赛之一。本竞赛以“立足培养，重在参与，鼓励探索，追求卓越”为指导思想，旨在促进高等学校素质教育，培养大学生的综合知识运用能力、基本工程实践能力和创新意识，激发大学生从事科学研究与探索的兴趣和潜能,倡导理论联系实际、求真务实的学风和团队协作的人文精神，为优秀人才的脱颖而出创造条件。

1.2　竞赛特点与特色

本竞赛以竞速赛为基本竞赛形式，辅助以创意赛和技术方案赛等多种形式。竞速赛以统一规范的标准硬软件为技术平台，制作一部能够自主识别道路的模型汽车，按照规定路线行进，并符合预先公布的其他规则，以完成时间最短者为优胜。创意赛是在统一限定的基础平台上，充分发挥参赛队伍想象力，以创意任务为目标，完成研制作品；竞赛评判由专家组、现场观众等综合评定。技术方案赛是以学术为基准，通过现场方案交流、专家质疑评判以及现场参赛队员和专家投票等互动形式，针对参赛队伍的优秀技术方案进行评选,其目标是提高参赛队员创新能力，鼓励队员之间相互学习交流。

本竞赛过程包括理论设计、实际制作、整车调试、现场比赛等环节，要求学生组成团队，协同工作，初步体会一个工程性的研究开发项目从设计到实现的全过程。竞赛融科学性、趣味性和观赏性为一体，是以迅猛发展、前景广阔的汽车电子为背景，涵盖自动控制、模式识别、传感技术、电子、电气、计算机、机械与汽车等多学科专业的创意性比赛。本竞赛规则透明，评价标准客观，坚持公开、公平、公正的原则，保证竞赛向健康、普及、持续的方向发展。

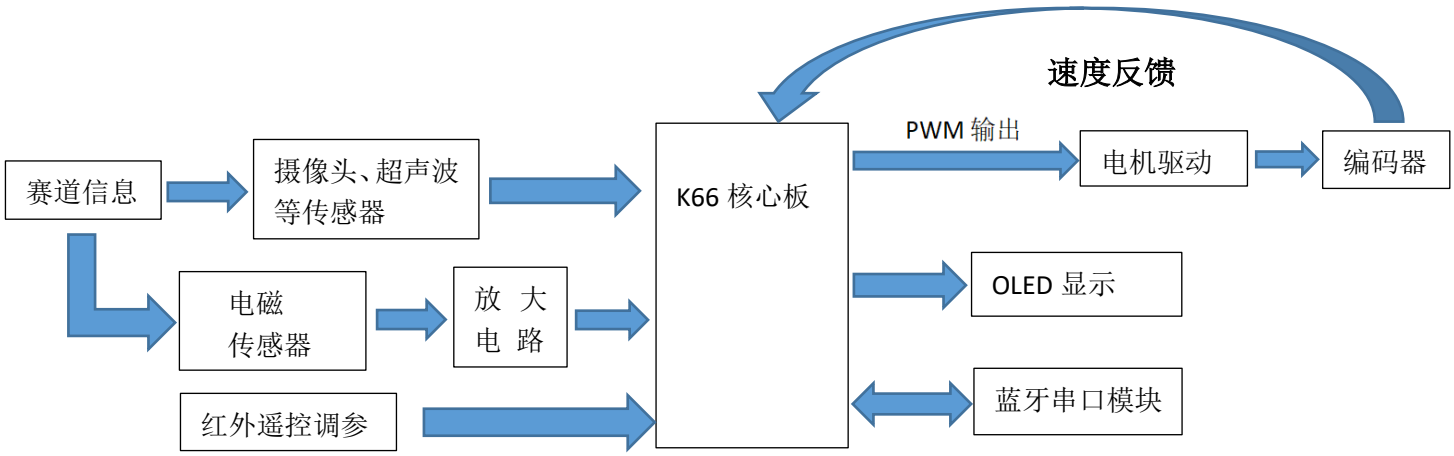
1.3　概述

本文将主要从方案设计、机械结构、硬件设计、软件算法等方面对 BBO 小车进行介绍，阐述了队员们在此次大赛中小车的制作过程，以及过程中所用到的思想和做出的创新。

本文分为五个章节，第一章为引言，介绍了此次大赛和本文的主要内容框架;第二章是小车的机械结构，介绍了小车的整体布局设计、安装调整以及一些物件的处理方法；第三章是小车的硬件电路设计，将分模块地对硬件电路进行说明；第四章是小车的软件算法，主要介绍小车的控制以及赛道元素的处理方法；第五章是开发与调试，将讲述 BBO 在制作过程中调试小车的工具与方法。

第二章：方案设计以及机械结构

2.1 设计方案



2.2 整体布局

分析后，将电池、主板靠后放置，以平衡前瞻的重量，使重心靠近后轮，并尽量降低，以增大后轮摩擦力而防止转向甩尾和提供驱动力。同时其重心应保持在中心轴线上，保证其左右对称性，便于转向。同时为了减少小车整体重量，选用强度高质量轻的碳纤维作为支架。

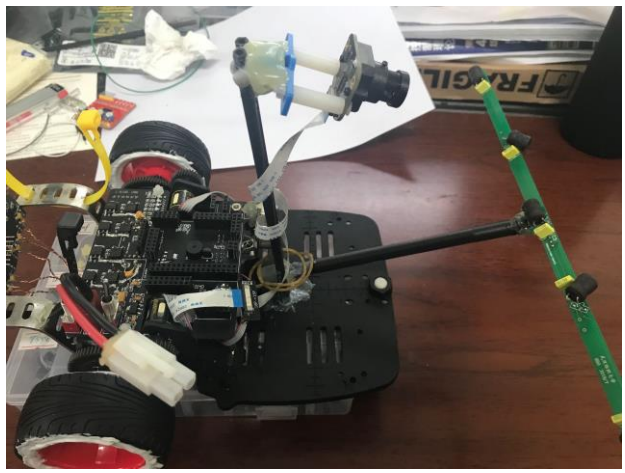
2.3 机械调整

（1）前瞻板的设计与安装

前瞻板的主要作用是作为小车的磁传感器，为小车的循迹采集信息，板上分布有三个水平电杆，和两个八字电感，在不同的路况下起作用，前瞻板中部的凹槽可刚好卡在碳纤维管上，板上分布有五个电感，引出的导线在碳纤维管内部走线，整体显得更为简洁。

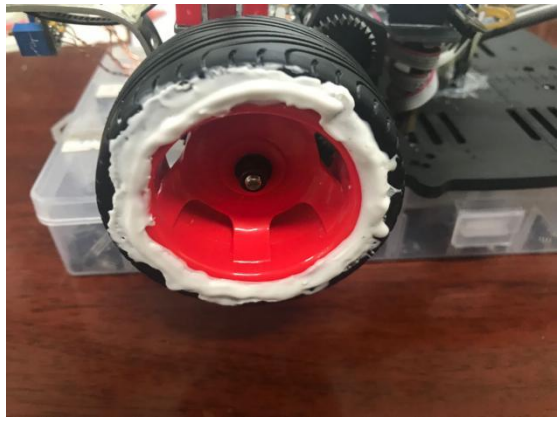
（2）摄像头的安装

摄像头放在小车靠前部分，可以获得更好的视野，其底部与前瞻杆相连，也起到固定前瞻的作用。



（3）后轮的处理

小车通过后轮驱动，轮胎对小车的速度以及过弯性能影响很大，在小车速度达到一定速度以后会发生甩尾打滑的问题，考虑到到小车原轮胎较硬而且内部很空，在转弯时提供不了足够的摩檫力，从而发生甩尾现象，所以我们在原轮胎的基础上做了一些改善，主要是为了使轮胎变软，在轮胎上涂上软化剂，再用热吹风处理，另外为了防止轮胎和轮毂发生相对滑动，我们在接合出涂上一圈硅胶，这样处理之后，确实很好的解决了小车的侧滑问题。此外，轮胎也要勤跑勤擦，每跑完一圈都会进行擦拭。



（4）编码器的安装

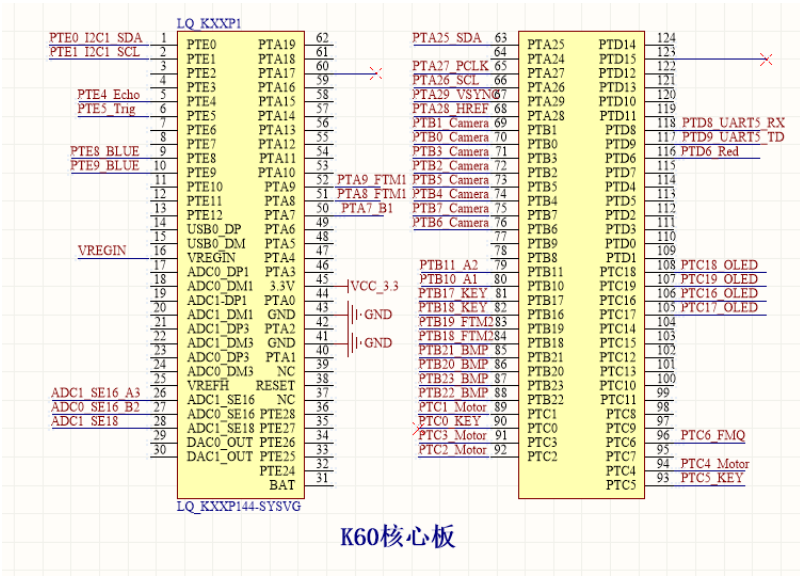
我们选取编码器为速度传感器，齿轮传动机构对车模的驱动能力有很大的影响。齿轮传动部分安装位置的不恰当，会大大增加电机驱动后轮的负载，

会严重影响最终成绩。调整的原则是：两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，浪费动力；传动部分要轻松、顺畅，不能有迟滞或周期性振动的现象。判断齿轮传动是否良好的依据是，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，或者两齿轮轴不平行，电机负载变大。调整好的齿轮传动噪音很小，并且不会有碰撞类的杂音，

第三章： 硬件电路设计

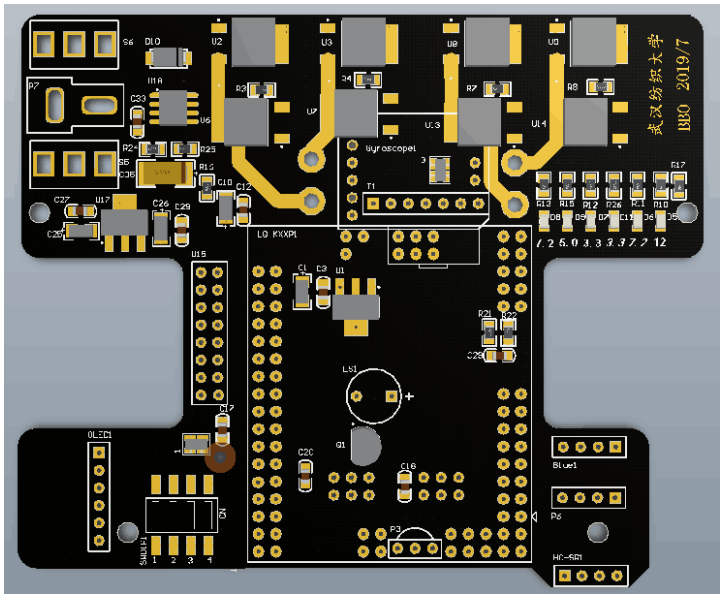
3.1 K66 最小系统

选用 MK66FX1M0VLQ18



3.2 pcb 主板设计

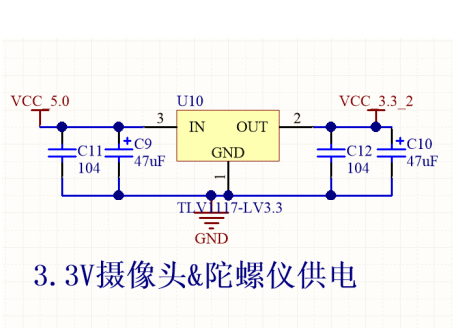
PCB 主板的设计应该要做到结构紧凑，排布有理，并且整体形状要贴合车模结构。



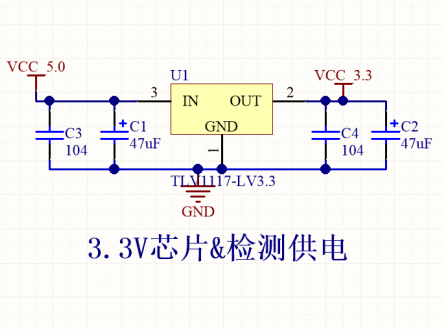
3.3 电源设计

电源模块为整个硬件电路提供稳定的电源，是各个模块的正常稳定工作的基础。在工作时，不同的模块对电源的要求不同，所以我们采用了多路电源，分别对不同的模块进行供电，本次智能车大赛使用的电池为镍镉充电电池，电压在 7.2v 左右，可以直接向电机驱动供电。除此之外，还有 3.3v、5.0v、以及 12v 供电。

（1）3.3v 供电采用了 TLV1117 稳压芯片，为了防止短路，对摄像头、陀螺仪和其它芯片分开供电，这一部分的供电主要是对摄像头、陀螺仪、编码器等模块。

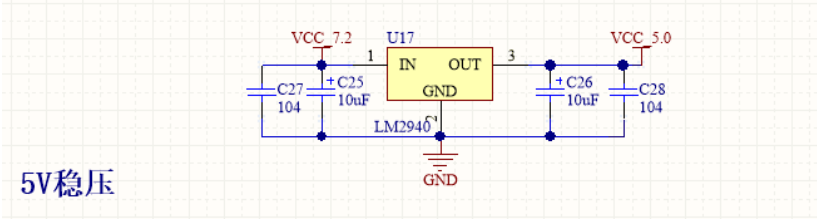


3. 3V摄像头&陀螺仪供电

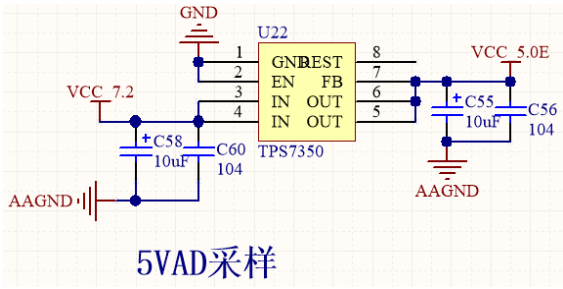


3. 3V芯片&检测供电

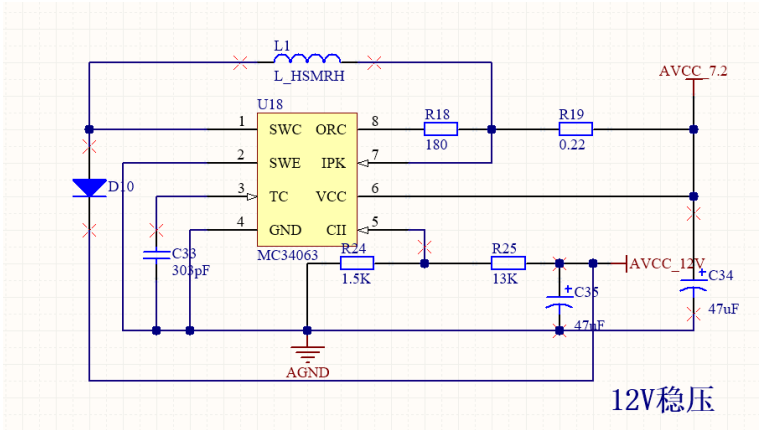
（2）5.0v 供电采用了 LM2940 稳压芯片，主要对超声波、红外传感、激光测距、蜂鸣器等模块供电。另外 TPS7350 芯片 5v 稳压对 AD 采样模块供电。



5V稳压

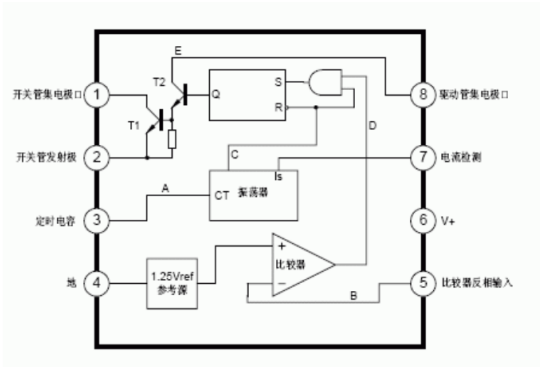


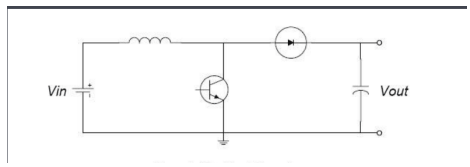
（3）12v 供电采用 MC34063 芯片升压，对电机驱动模块供电。



升压原理：电路中用到的 MC34063 芯片管脚图及 boost 升压电路

2. MC34063 引脚图及原理框图



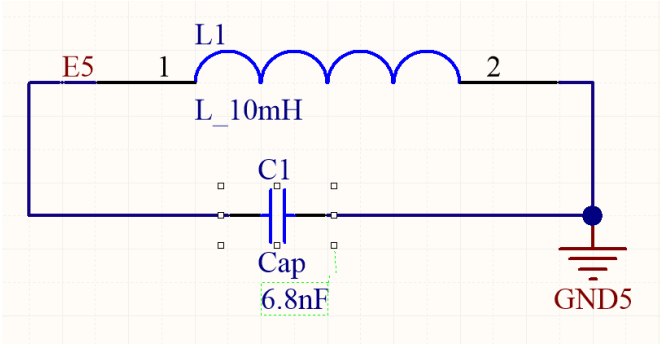


振荡器根据 7 管脚的电流生成占空比不同的方波，并从 C 输出，当 5 管脚低于 1.25v 参考电压时，D 一直为高电平，根绝 RS 触发器特性，Q 与 C 波形一致，而 Q 为高电平时，两个三极管导通，电感会储存能量，等到 Q 低电平时，电感向电路图中的 C35 充电，达到升压效果。

3.4 信号放大设计

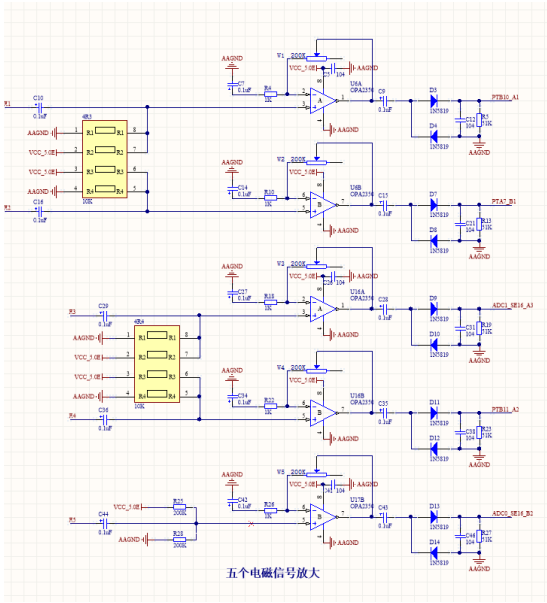
比赛赛道线路导线中为 20KHZ 的交变电流，所以要筛选出 20KHZ 的信号而滤除其他的杂波，所以先用 LC 并联谐振电路实现信号选频，查阅资料然后测试,发现 10mH 的电感产生的感应电动势曲线较为规整，而其他电感产生的信号

变化杂乱，故采用 10mH 的电感和 6.8nF 的电容，如图。



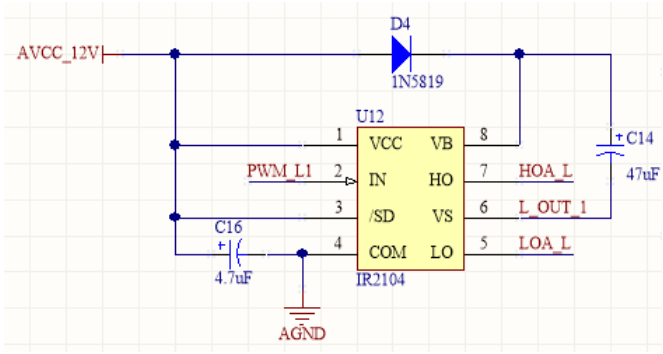
选频之后信号还很微弱，需要对信号进行放大处理，我们选用 OPA2350 进行运放，其优点是能得到稳定的数据且输出稳定。在原先的信号被放大了之后，单片机通过 ad 采样获得正比于感应电动势的数值，该数值反映了前瞻板与导线的相对位置。

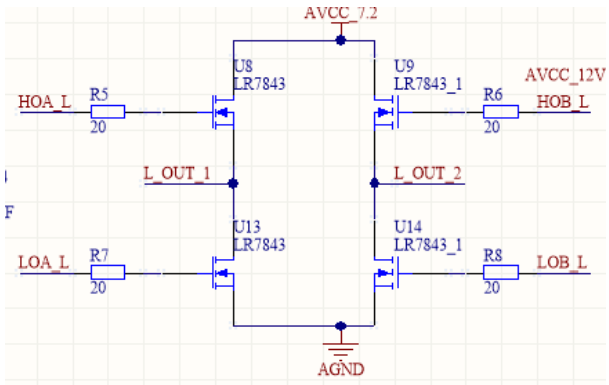
放大电路：



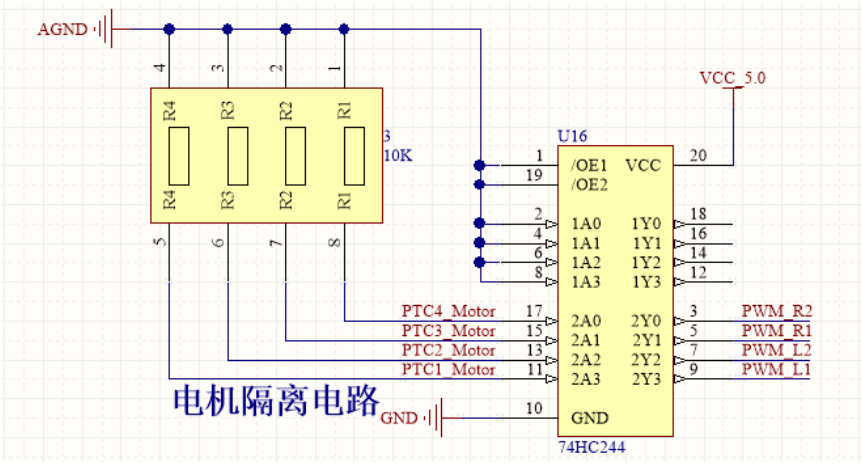
3.5 驱动模块

两个电机提供了小车的所有动力，所以驱动模块的设计对小车的速度影响很大，本次比赛用到的是直流电机，通过控制电流方向可实现电机的正转和反转，我们采用 mos 管与 IR2104 来构成 H 桥控制电路，电路简单高效，控制速度快。IR2104S 驱动一个半桥，结构简单，电路方便，驱动电流较大，很好地满足小车运动的需要，而且电路简单高效，控制速度快。





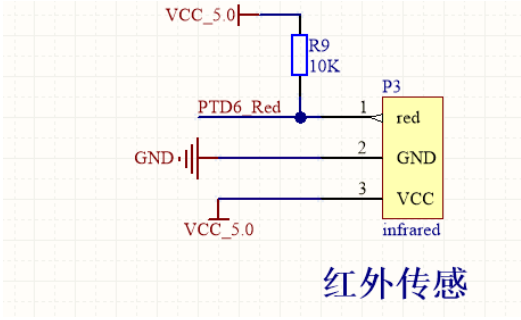
另外，电机在开端瞬间会产生很大电流，如果直接通过导线返回给单片机会造成损坏，所以电机驱动隔离电路在电路设计中是很有必要的。我们采用了74HC224 芯片，具体电路如下：



3.6 其他传感器模块

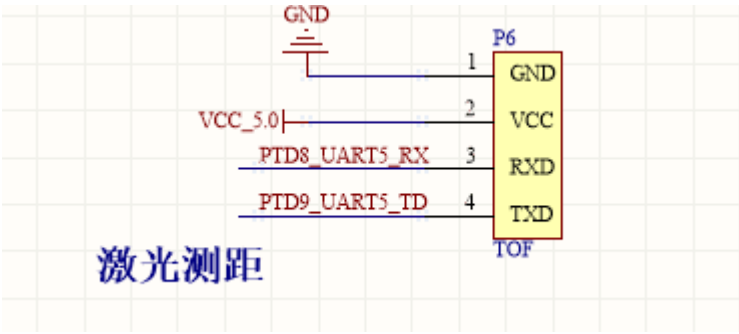
（1）红外传感

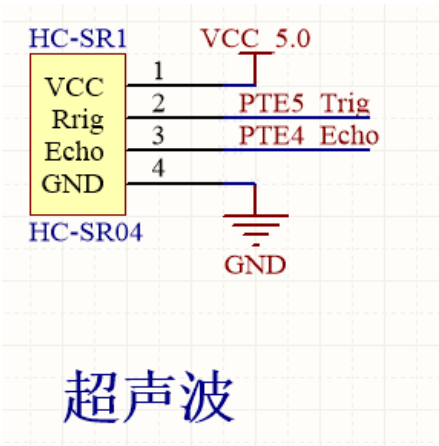
红外传感用于小车的调参以及紧急停车。小车的调试过程中需要不断地调整参数以找到最合适的值，通过红外直接用遥控按键调整就十分便捷，不用一直烧程序。另外为防止小车测试过程中脱离赛道发生碰撞而损坏，编写了紧急停车程序，也通过红外来实现。



（2）超声波和激光测距

比赛中小车有直立和水平两种状态，因为不清楚路障路段出现时小车的具体情况，所以我们用了超声波测距和激光测距分别在两种情况下判断小车与路障的距离。





（3）摄像头

用来检测起跑线和路障

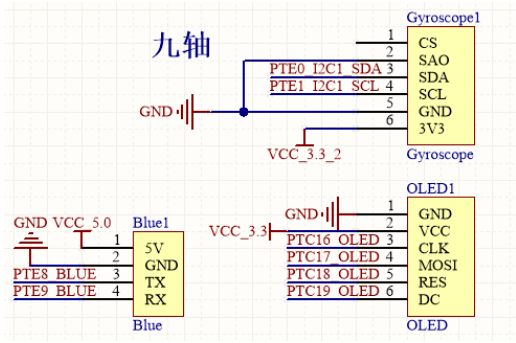
（4）光对管

检测断路。

（5）陀螺仪加速度计

此模块用于检测小车直立的角度以及加速度控制 ，我们采用的是

FSOS8700 和 FXA2100。



第四章： 软件算法

4.1 总体设计

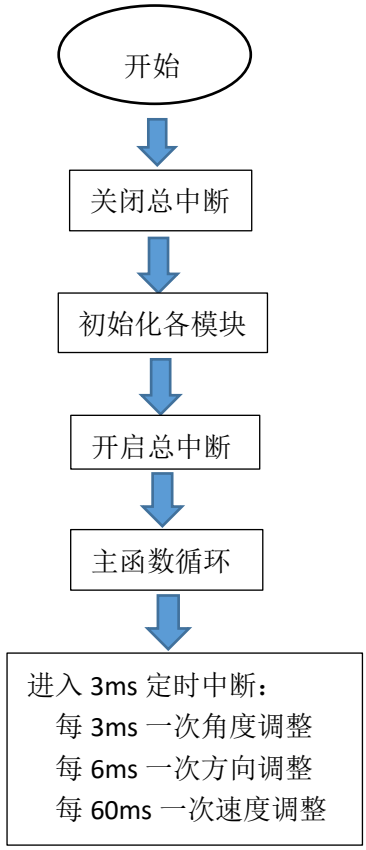
软件是小车的控制系统，根据比赛任务要求，小车有水平和直立两种行进方式，我们可以将车模的控制分为三个方面，直立平衡、速度和方向。小车所有的动力都是来自两个后轮,所以所有的控制都要通过后轮两个电机的转动来实现。为了使系统更加稳定，主要运用了 PID 控制算法，还采用了模糊控制。

（1）小车直立平衡：通过两个电机正反向运动保持车模直立平衡。

（2）小车速度控制：通过调节车模的倾角来实现速度的控制，实际上与小车的平衡控制也有关系，当小车前倾一定角度时，小车为保持平衡，控制两个电机的转速产生加速度，进而有了前进的速度。

（3）小车方向控制：通过控制两个电机之间的转动差速实现车模转向控制。

程序流程图：



4.2 运动控制

（1）水平方向环

我们在前瞻板上放置了三个水平电感和两个八字电感，小车的巡线主要依靠两边的水平电感，中间的水平电感用来判断圆环，而进环需要八字电感和水平电感共同作用。

方向采用 PD 控制，小车偏离中心后，根据电感上的感应电压来生成电机差动控制量，通过左右电机速度差驱动小车转向使其回到赛道中心以消除偏差。一般来说，当小车偏离中心时，跟据偏差值，进行比例控制就可以纠正小车的行驶方向，比例作用越大，调整就越迅速，幅度也越大，但小车具有一定重量，在惯性的作用下，当调整幅度大到一定程度之后发生过冲，所以为了达到迅捷稳定且防止过冲的效果，再增加微分控制。

方向的控制需要测量两个值，其中一个由前瞻板上的电感测出，另一个由陀螺仪测出，当两侧电感测出电感之后，用差比和和方法处理数据，我们想让这个数据与偏离角度具有线性关系以便于比例控制，而直接差比和在一定范围内具有线线性特性，但在偏离距离更远的地方模拟效果不好，故在作差时先作开根号处理，这样得到的数据与偏离角度有更好的线性关系。另外小车转弯时由于前瞻板的长度可能会出现很大的偏差，防止出现突然的急转弯，对偏差进行限幅。陀螺仪测出的是小车转向的角速度，用于微分控制。

（2）直立平衡

根据参考设计方案上给出的倒立摆模型，要保持直立得满足两个条件：

（1）收到与位移（角度）相反得恢复力；

（2）受到与运动速度（角速度）相反得阻尼力；

与运动方向相反的恢复力可以通过控制底部车轮作加速运动得到，即

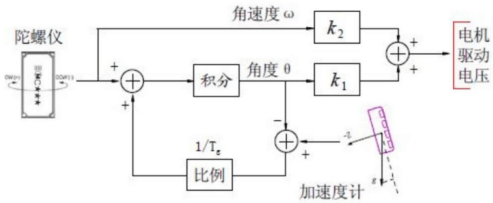
$$F = mg\sin\theta - ma\cos\theta \approx mg\theta - mk_1\theta$$

在加上与偏角速度成正比的阻尼力，即

$$F = mg\sin\theta - ma\cos\theta - mk_2\omega$$，故可得车轮加速度的控制算法即

$$a = k_1\theta + k_2\omega$$

所以对于小车直立，就需要测量出小车的角度和角速度，分别由陀螺仪和加速度计测出，而在基于角度负反馈的 PD 控制中，与角度成比例的控制量为比例控制，与角速度成比例的控制为微分控制。

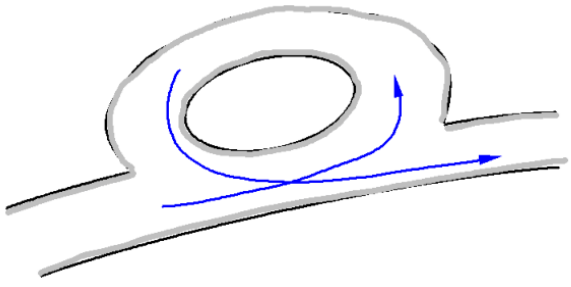


（3）速度控制

速度的控制通过调节小车倾角实现，采用了直立环和速度环的串级控制，小车的直立平衡通过 PD 控制器进行控制。速度的返回值由编码器采集，通过 PI 控制器输出速度控制量作为需要调节的角度作为 PD 控制器的输入，输入到直立控制中，形成直立环和速度环的串级控制。

4.3 赛道元素处理

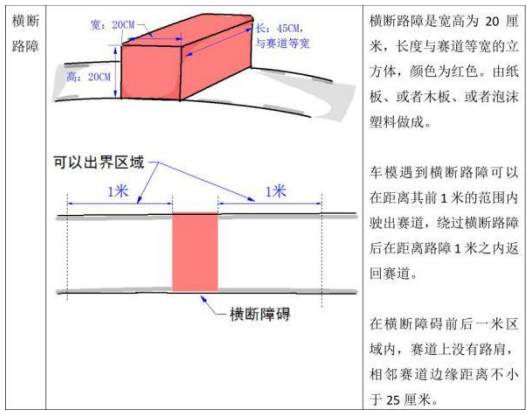
（1）环岛

环岛		赛车经过环岛时需进入环岛绕行一周后继续前行。环岛中心线半径不小于 50 厘米。 电磁导线也是在环岛绕行一周。

在圆环路口，由于导线交叉，中间水平电感所产生的感应电动势要比直道单线的大大约两倍，基于此判断圆环路段，判定圆环之后，小车面前有两跟导

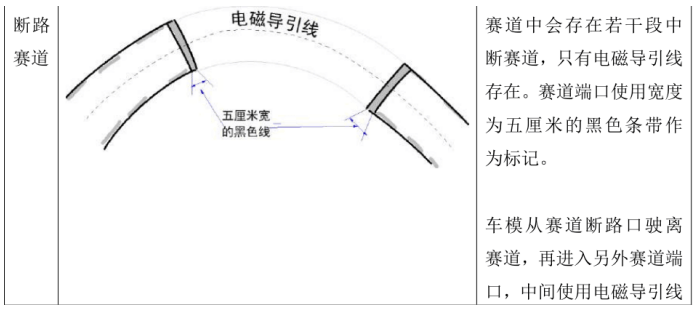
线，其中直导线对水平电感影响较大，弯曲导线对八字电感影响较大，如果不做处理，只使用水平电感，小车会在入环处往入环方向抖动一下然后纠正之后直行，而只使用八字电感，小车会向弯曲导线迅速打角，所以此时，为使小车轨迹更加平滑，改用用八字电感和水平电感共同作用，对两对电感的偏差进行加权计算，入环之后再改用两边的水平电感循迹。小车的出环也需要处理，为防止不断进环，用 flag 语句进行判断所处状况。

（2）路障



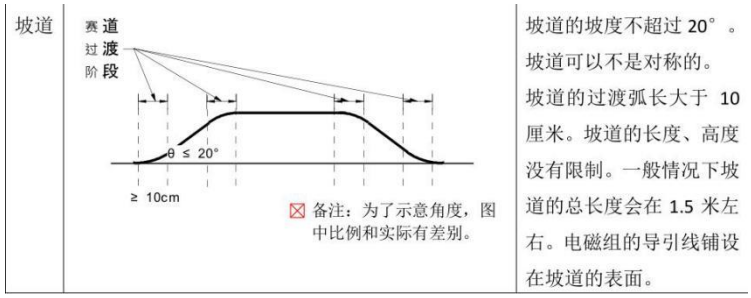
路障的判断用到了超声波和摄像头，超声波可以测定和路障之间的距离，接近到设定距离之后，预先设定一个角度 θ，小车开始打角，行驶一小段距离出界之后，再反方向以 θ 打角回与赛道平行状态，再行驶一小段距离，保证绕过路障，之后再次以 θ 角转回赛道，回赛道最后做一次打角继续正常行驶。编码器采集了小车速度，由积分的值来判断各阶段所设定的行驶距离是否完成。

（3）断路



在断路路段，要求小车三轮变两轮，我们使用光对管判断断路，判定为断路之后，小车的期望角度改变，小车通过平衡控制抬起一定角度，变为两轮，还是根据电感循迹，光对管判断通过断路之后，期望角度恢复为水平状态，小车正常行驶。

（4）坡道



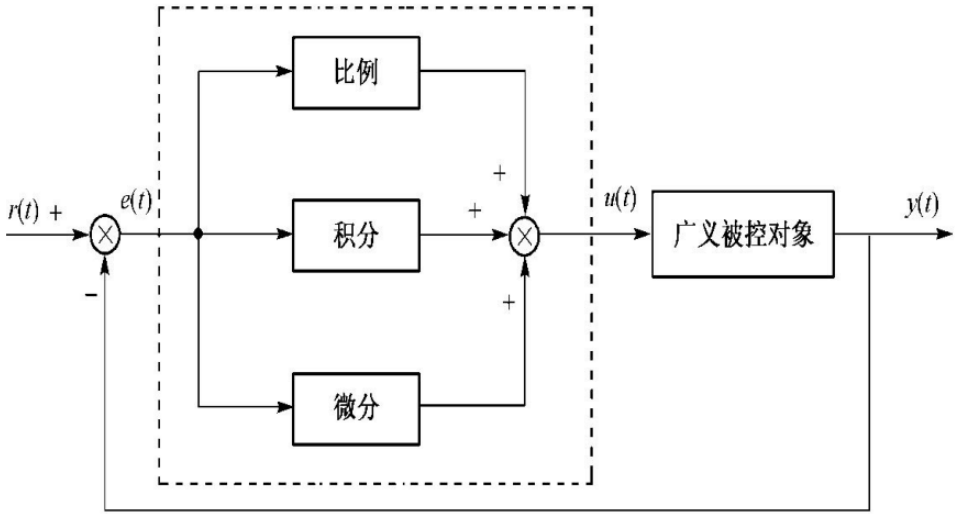
我们发现，由于坡道角度不大于 20 度，当小车速度达到一定值之后可以直接冲上坡道，所以我们对于坡道的判断和处理显得有点多余，所以对于坡道我们并没有作太大处理，下坡时小车会获得较大速度，状态变得不稳定，所以我们对小车的速度进行了限幅，以免失控。

4.5 PID 控制原理

PID 控制以其结构简单、稳定性好、工作可靠、调整方便而成为工业控制的主要技术之一。当被控对象的结构和参数不能完全掌握，或得不到精确的数学模型时，系统控制器的结构和参数必须依靠经验和现场调试来确定，这时应用 PID 控制技术最为方便。

PID 控制就是根据系统的误差，利用比例(P)、积分(I)、微分(D)计算出控制量进行控制的。目前 PID 控制在工业控制系统中无处不见，随着控制效果的要求不断提高，PID 逐渐向智能化发展，但形形色色的现代控制理论最终还是源自经典 PID 理论。

其控制框图如下图所示：



PID 解决了自动控制理论所要解决的最基本问题，既系统的稳定性、快速性和准确性。调节 PID 的参数，可实现在系统稳定的前提下，兼顾系统的带载能力和抗扰能力，同时，在 PID 调节器中引入积分项，系统增加了一个零积点，使之成为一阶或一阶以上的系统，这样系统阶跃响应的稳态误差就为零。

由于自动控制系统被控对象的千差万别，PID 的参数也必须随之变化，以满足系统的性能要求。因此 PID 参数的确定是 PID 控制中重要的部分之一。

PID 控制的基本原理如式 1 所示：

$$u(t)=K_p\left[e(t)+\frac{1}{T_i}\int_0^t e(t)\,\mathrm{d}t+T_d\,\frac{\mathrm{d}e(t)}{\mathrm{d}t}\right] \tag{1}$$

式中，u(t) -- 控制量

KP -- 比例系数

Ti -- 积分时间常数

Td -- 微分时间常数

将式 1 离散化，即将描述连续系统的微分方程代之以等效的描述离散系统的差分方程，就可以得到相应的数字 PID 调节器。当控制周期也就是积分时间足够小的时候，利用矩形法进行数值积分，即以求和代替积分，以差分代替微分，可得到数字形式的 PID 控制方程如式 2 所示：

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

式中 T 为控制周期，当 T 足够小时离散化的控制模型可以逼近连续式的 PID 控制模型。另外，可以将式 2 中的 T 提取出来，把 Kp 分配进去，分别与 1/Ti 和 Td 组成比例项系数 KP，积分项系数 KI 和微分项系数 KD，式 2 可简化为式 3：

$$u(k)=K_P e(k)+K_I\sum_{n=0}^{k}e(n)+K_D\left[e(k)-e(k-1)\right] \tag{3}$$

式 3 也就是常说的位置式 PID 控制方程，但是由于积分项累加计算比较复杂，而且在程序计算中极容易出现溢出现象，因此人们考虑每一次不是给出绝对的控制量，而是根据上一次的控制，给出本次需要增加的控制量即可。根据这一想法，我们首先考虑上一次(即 k-1 时)给出的控制量 u(k-1)为：

$$u(k-1)=K_P e(k-1)+K_I\sum_{n=0}^{k-1}e(n)+K_D\left[e(k-1)-e(k-2)\right] \tag{4}$$

式 3-式 4 可得 k 时刻需要给出的增量

$$u(k)-u(k-1)=K_P\left[e(k)-e(k-1)\right]+K_I e(k)+K_D\left[e(k)-2e(k-1)+e(k-2)\right] \tag{5}$$

由式 5 得输出 u(k)也可表示为：

$$u(k)=u(k-1)+K_P\left[e(k)-e(k-1)\right]+K_I e(k)+K_D\left[e(k)-2e(k-1)+e(k-2)\right] \tag{6}$$

式 6 就是增量式 PID 控制方程，由此可以看出，增量式 PID 消去了积分项

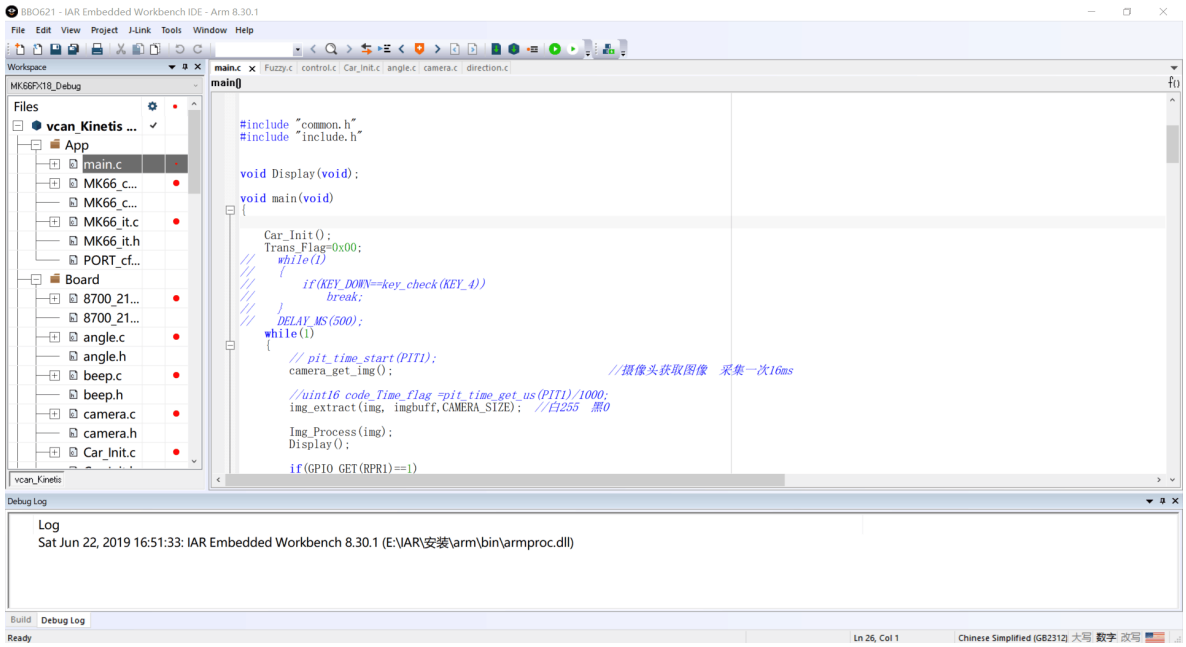
的累加部分，这样便于程序编写。

第五章 软件开发平台与车模调试

5.1 开发环境

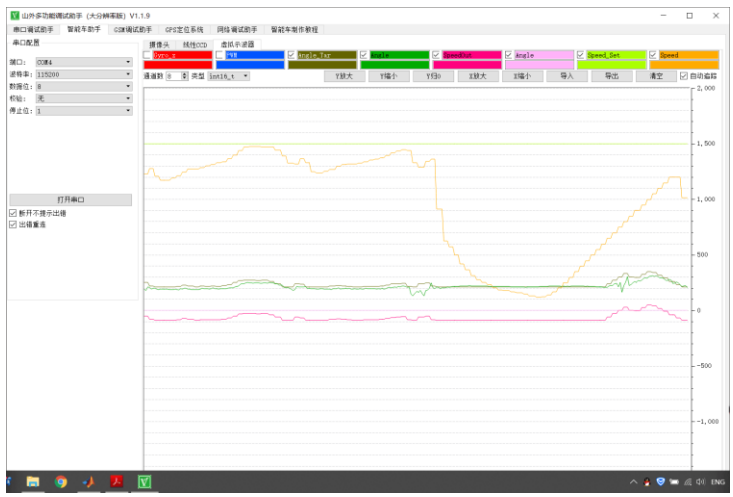
在对程序进行开发和软硬件联调的过程中需要一整套的软件开发与调试工具。程序的开发是在 IAR Embedded Workbench 下进行的，包括源程序的编写、编译和链接，并最终生成可执行文件。包括集成开发环境 IDE、处理器专家库、全芯片显示工具、项目工程管理器、C 交叉编译器、汇编器、链接器以及调试器。

使用 J-LINK 来下载程序，把编译好的程序下载到单片机里运行



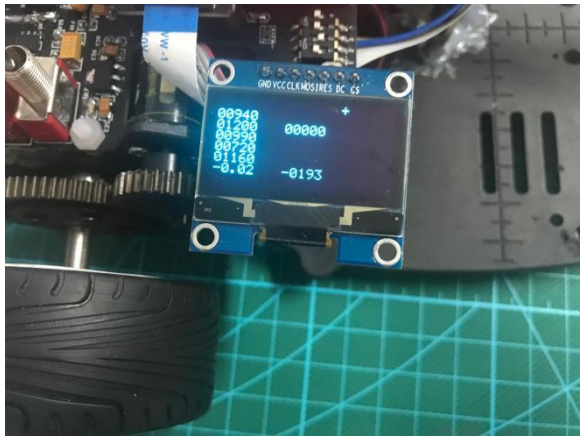
5.2 调试工具

山外调试助手是一款专为飞思卡尔开发板而配套的调试工具，同时集成了一些常用的调试功能，目前集成了 串口调试助手、摄像头调试助手、线性 CCD 调试助手、虚拟示波器、GSM 调试助手、GPS 定位系统、网络调试助手（TCP 服务器、TCP 客户端、UDP）等调试功能。



5.3 OLED 辅助显示

为了能更直观的观察小车的各项参数，再小车上安装了 1.3 寸的 OLED 屏来显示各个电感以及偏差的值，由于空间有限，能显示的项目较少，旁边的拨码盘可以起到切换的作用，用来查看不同的数值。



5.4 参数调整

小车调试过程中，有时候需要频繁修改参数来测试最适合的值，在程序里修改参数值再烧录会很繁琐，编写红外程序，可直接通过遥控按键对小车的参数进行调整，比如 PID 控制中各项参数的值，并且可以在山外助手上看到参数值的变化情况。

除了调参，还可以通过按键对小车进行其他控制，比如紧急停车，当小车失控或者冲出赛道时，可以直接通过按键使小车停止运动，以免造成损坏。

参考文献：

1. 邵贝贝.单片机嵌入式应用的在线开发方法[M].北京.清华大学出版.2004
2. 卓晴.智能汽车自动控制器方案设计[M].北京航空航天大学出版社.2007
3. 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
4. 程鹏著．自动控制原理．高等教育出版社. 2010
5. 卓晴 李马: 基于磁场检测的寻线小车传感器布局研究. 2009.
6. 卓晴 李马: 基于磁场检测的寻线小车传感器布局研究. 2009.
7. 田书林等．电子测量技术．北京．机械工业出版社．2012.
8. 仲志丹，张洛平，张青霞．PID 调节器参数自寻优控制在运动伺服中的应用[J] ．洛 阳工学院学报，2000

总结：

经过半年的努力，我们终于完成了小车的制作，在过程中我们遇到了各种各样的困难，在调试过程中也会遇到很多实际的意想不到的问题，有时会让我们很头疼，这是一个不断学习的过程，重要的是要坚持下来将这些问题一个个解决，一点点的进步。总而言之，此次竞赛让我们学到了很多不拘于书本的知识，让我们的各方面知识和能力都得到了很大提高。感谢一路以来老师们对我们的帮助，也感谢这一次比赛给我们带来的那么多收获。

附源代码

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

extern int32 PWM;
extern int16 Gyro_z;
extern int16 Accle_z;
extern float Angle;
extern float Direction_PD[2][2] ;
extern float Dierct_PWM;
extern int32 Speed_Set;
extern int32 Speed[5];
extern float g_dirControl_P;
extern float g_dirControl_D;
extern int16 g_ValueOfAD[5];
extern uint16 Barrier_Distance;
extern uint8    symbol ;
extern int32 Speed_L, Speed_R;

extern int16 Accle_x;

void Display(void);

uint8 Trans_Flag=0x00;
//   0x04 直立变水平中     0x02 水平变直立中     0X00 直立    0x01 水平

void main(void)
{

      Car_Init();
      Trans_Flag=0x00;
//      while(1)
//      {
//            if(KEY_DOWN==key_check(KEY_4))
//                  break;
//      }
//      DELAY_MS(500);
      while(1)
      {
           // pit_time_start(PIT1);
           camera_get_img();                                          //摄像头获取图像    采
集一次 16ms
```

```
            //uint16 code_Time_flag =pit_time_get_us(PIT1)/1000;
            img_extract(img, imgbuff,CAMERA_SIZE);    //白 255    黑 0

            Img_Process(img);
            Display();

            if(GPIO_GET(RPR1)==1)
                led (LED0,LED_ON);
            else
                led (LED0,LED_OFF);

            if(GPIO_GET(RPR2)==1)
                led (LED1,LED_ON);
            else
                led (LED1,LED_OFF);

            if(GPIO_GET(RPR2)==1&&GPIO_GET(RPR1)==1&&
(Star_Flag&0x20)&&Break_Flag==0)//低电平亮
            {
//                  beep_on();
//                  DELAY_MS(100);
//                  beep_off();
                if((Bar_State == 0 || Bar_State == 5))
                {
                    while(GPIO_GET(RPR2)==1);
                    Break_Flag=1;
                }
            }
//           img_recontract(img, imgbuff,CAMERA_SIZE);          //  图像压缩
//           send_camera(imgbuff,CAMERA_SIZE) ;

/*电感的*/
#if 0
            Send_data[0]=(int16)Direct_PWM;
            Send_data[1]=g_ValueOfAD[0];
            Send_data[2]=g_ValueOfAD[1];
            Send_data[3]=g_ValueOfAD[2];
            Send_data[4]=g_ValueOfAD[3];
            Send_data[5]=g_ValueOfAD[4];
            Send_data[6]=(int16)(g_fDirectionError[0]*1000);
            Send_data[7]=g_fDirectionError_dot*10000;
            send_oscilloscope((uint8 *)Send_data, sizeof(Send_data));
            #endif
/*路障的*/
```

```
#if 0
        Send_data[0]=(int16)Direct_PWM;
        Send_data[1]=Accle_x;
        Send_data[2]=E_Gyro;
        Send_data[3]=Cont;
        Send_data[4]=Barrier_Distance;
        Send_data[5]=Bar_State*1000;
        Send_data[6]=Pulse_Cont;
        Send_data[7]=g_ValueOfAD[4];
        send_oscilloscope((uint8 *)Send_data, sizeof(Send_data));
        #endif
/*坡道的*/
#if 0
        Send_data[0]=(int16)g_ValueOfAD[4];
        Send_data[1]=(int16)Speed_Filter;
        Send_data[2]=g_ValueOfAD[0];
        Send_data[3]=Slope_State*1000;
        Send_data[4]=g_ValueOfAD[2];
        Send_data[5]=Angle_Y;
        Send_data[6]=Slope_Flag*100;
        Send_data[7]=(int16)TPWM*100;
        send_oscilloscope((uint8 *)Send_data, sizeof(Send_data));
        #endif

/*速度的*/
#if 0
        Send_data[0]=(int16)Speed_PWM;
        Send_data[1]=(int16)Speed_Filter;
        Send_data[2]=Last_TGyro;
        Send_data[3]=AimAngle-10000;
        Send_data[4]=TAngle-10000;
        Send_data[5]=APWM;
        Send_data[6]=TSpeed_Set;
        Send_data[7]=(int16)TPWM*100;
        send_oscilloscope((uint8 *)Send_data, sizeof(Send_data));
        #endif
/*模糊的*/
#if 0
        Send_data[0]=(int16)PPWM;
        Send_data[1]=E_Gyro;
        Send_data[2]=(int16)DirFuzzy_P;
        Send_data[3]=(int16)(DirFuzzy_D*100000);
        Send_data[4]=(int16)DPWM;
        Send_data[5]=(int16)(Last_Errordot*10000);
```

```
            Send_data[6]=(int16)(g_fDirectionError[0]*10000);
            Send_data[7]=(int16)Direct_PWM;
            send_oscilloscope((uint8_t *)Send_data, sizeof(Send_data));

#endif
/*圆环的*/
#if 0
            Send_data[0]=Flag_Round*1000;
            Send_data[1]=g_ValueOfAD[1];
            Send_data[2]=g_ValueOfAD[2];
            Send_data[3]=g_ValueOfAD[3];
            Send_data[4]=Round_State*1000;
            Send_data[5]=g_fDirectionError[0]*10000;
            Send_data[6]=g_fDirectionError[1]*1000;
            Send_data[7]=Direct_PWM;
            send_oscilloscope((uint8 *)Send_data, sizeof(Send_data));

#endif
//extern uint8 time200ms;

extern float fGyroscopeAngleIntegral;
extern float Angle2, Angle3;
extern int32    Speed_Set,Speed_Out;
#if 1
            Send_data[0]=(int16)Gyro_z;
            Send_data[1]=PWM;
            Send_data[2]=Angle_Tar;
            Send_data[3]=(int16)Angle;
            Send_data[4]=(int16)Speed_Out;
            //Send_data[5]=(int16)Angle-Angle_Tar;
            Send_data[6]=(int16)Speed_Set;//ACC_RealZ;// Speed_Set
            Send_data[7]=(int16)Speed[0];//Speed[4];
            send_oscilloscope((uint8 *)Send_data, sizeof(Send_data));

#endif

    }
}


/*****************************************************************************
*****************************
函数名称: Display
功      能: 通过拨码盘进行菜单选择
```

```
参      数:
返      回: void
*************************************************************************************
*********************************/
void Display(void)
{
    char txt[16]=" ";

    uint8 Boma=0;
    Boma |= key_get(SW_4)<<3;
    Boma |= key_get(SW_3)<<2;
    Boma |= key_get(SW_2)<<1;
    Boma |= key_get(SW_1)<<0;
     if(symbol)
        LCD_P6x8Str(90,0,"+");
    else
        LCD_P6x8Str(90,0,"-");
    switch(Boma)
    {
        case 0x01:
            dis_bmp(60,80,img,0x7F);

            sprintf(txt,"%05d",Barrier_Distance);
            LCD_P6x8Str(90,1,(uint8 *)txt);
            sprintf(txt,"%05d",Emit_Flag);
            LCD_P6x8Str(90,2,(uint8 *)txt);
            sprintf(txt,"%05.1f",Dolly_now_angle);
            LCD_P6x8Str(90,3,(uint8 *)txt);
            sprintf(txt,"%05.1f",Width_Average);
            LCD_P6x8Str(90,4,(uint8 *)txt);

            break;

        case 0x02:
            sprintf(txt,"%05d",g_ValueOfAD[0]);
            LCD_P6x8Str(0,1,(uint8 *)txt);
            sprintf(txt,"%05d",g_ValueOfAD[1]);
            LCD_P6x8Str(0,2,(uint8 *)txt);
            sprintf(txt,"%05d",g_ValueOfAD[2]);
            LCD_P6x8Str(0,3,(uint8 *)txt);
            sprintf(txt,"%05d",g_ValueOfAD[3]);
            LCD_P6x8Str(0,4,(uint8 *)txt);
            sprintf(txt,"%05d",g_ValueOfAD[4]);
            LCD_P6x8Str(0,5,(uint8 *)txt);
```

```
            sprintf(txt,"%05d",Speed[4]);
            LCD_P6x8Str(50,2,(uint8 *)txt);

            sprintf(txt,"%05.2f",g_fDirectionError[0]);
            LCD_P6x8Str(0,6,(uint8 *)txt);

            sprintf(txt,"%05d",(int16)Direct_PWM);
            LCD_P6x8Str(50,6,(uint8 *)txt);

            break;

        case 0x04:

              LCD_P6x8Str(0,1,"IRD");
            LCD_P6x8Str(0,2,"SpdSet");
            LCD_P6x8Str(0,3,"Speed");

            LCD_P6x8Str(0,4,"KP");
            LCD_P6x8Str(0,5,"KI");
            LCD_P6x8Str(0,6,"KD");


//              sprintf(txt0,"%05X",Data_IRD);
//            LCD_P6x8Str(40,0,(uint8 *)txt0);




            sprintf(txt,"%05.0f",g_dirControl_P);
            LCD_P6x8Str(80,4,(uint8 *)txt);

            LCD_P6x8Str(80,5,"Spedir");

            sprintf(txt,"%05.4f",g_dirControl_D);
            LCD_P6x8Str(80,6,(uint8 *)txt);
            break;

        case 0x08:
            LCD_P6x8Str(0,1,"GySpS");
            LCD_P6x8Str(0,2,"AngSp");
            LCD_P6x8Str(0,3,"pwmLR");
            LCD_P6x8Str(0,4,"KP");
```

```
                LCD_P6x8Str(0,5,"KI");
                LCD_P6x8Str(0,6,"KD");

                sprintf(txt,"%05d",Gyro_z);
                LCD_P6x8Str(40,1,(uint8 *)txt);
                sprintf(txt,"%05d ",(int16)Angle);
                LCD_P6x8Str(40,2,(uint8 *)txt);

                sprintf(txt,"%05d",Speed_Set);
                LCD_P6x8Str(80,1,(uint8 *)txt);
                sprintf(txt,"%05d",Speed[0]);
                LCD_P6x8Str(80,2,(uint8 *)txt);

                sprintf(txt,"%05d ",Speed_L);
                LCD_P6x8Str(40,3,(uint8 *)txt);
                sprintf(txt,"%05d ",Speed_R);
                LCD_P6x8Str(80,3,(uint8 *)txt);

                sprintf(txt,"%05.2f",Speed_PID.KP);
                LCD_P6x8Str(30,4,(uint8 *)txt);
                sprintf(txt,"%05d",Trans_Flag);
                LCD_P6x8Str(30,5,(uint8 *)txt);
                sprintf(txt,"%05.3f",Speed_PID.KD);
                LCD_P6x8Str(30,6,(uint8 *)txt);

                sprintf(txt,"%05.3f",Angle_PID.KP);
                LCD_P6x8Str(80,4,(uint8 *)txt);
                sprintf(txt,"%05.4f",Angle_PID.KI);
                LCD_P6x8Str(80,5,(uint8 *)txt);
                sprintf(txt,"%05.4f",Angle_PID.KD);
                LCD_P6x8Str(80,6,(uint8 *)txt);

//                  sprintf(txt,"%05.4f",Gyro_Rate);
//                  LCD_P6x8Str(80,1,(uint8 *)txt);

                break;

            default:
                LCD_CLS();
        }

}
```