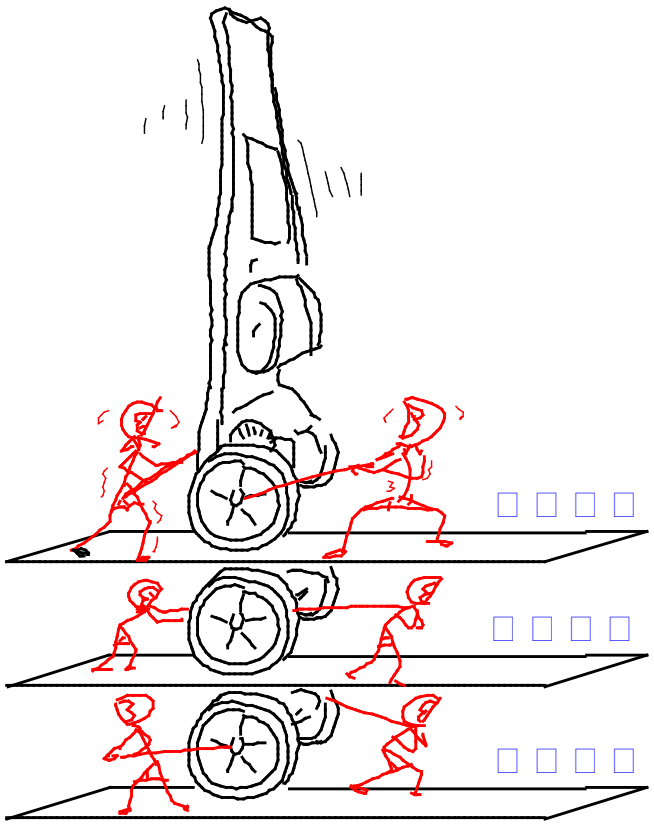
第七届全国大学生“飞思卡尔”杯智能汽车竞赛

电磁组直立行车参考设计方案

（版本 1.0）



竞赛秘书处

2011-12-22

目　　录

一、前言

为了提高全国大学生智能汽车竞赛创新性和趣味性，激发高校学生参与比赛的兴趣，提高学生的动手能力、创新能力和接受挑战能力，智能汽车竞赛组委会将电磁组比赛规定为车模直立行走（如图 1.1 所示），其它两个组别的车模行走方式保持不变。

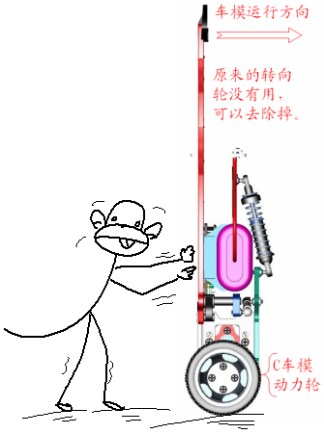


图 1.1 电磁组车模直立运行模式

车模直立行走比赛是要求仿照两轮自平衡电动车的行进模式，让车模以两个后轮驱动进行直立行走。近年来，两轮自平衡电动车以其行走灵活、便利、节能等特点得到了很大的发展。国内外有很多这方面的研究，也有相应的产品。在电磁组比赛中，利用了原来 C 型车模双后轮驱动的特点，实现两轮自平衡行走。相对于传统的四轮行走的车模竞赛模式，车模直立行走在车体检测、控制算法等方面提出了更高的要求。为了能够帮助参赛同学尽快制作车模参加比赛，竞赛秘书处编写了 C 型车模直立行走的参考设计方案。参赛队员可以在此基础上，进一步改进硬件和软件方案，提高竞赛水平。

为了适应初学者，方案介绍过程中，尽可能减少公式推导，使用通俗科学的语言介绍控制原理和方法，给出 C 型车模制作过程中的核心环节。本文的主要内容如图 1.2 所示。

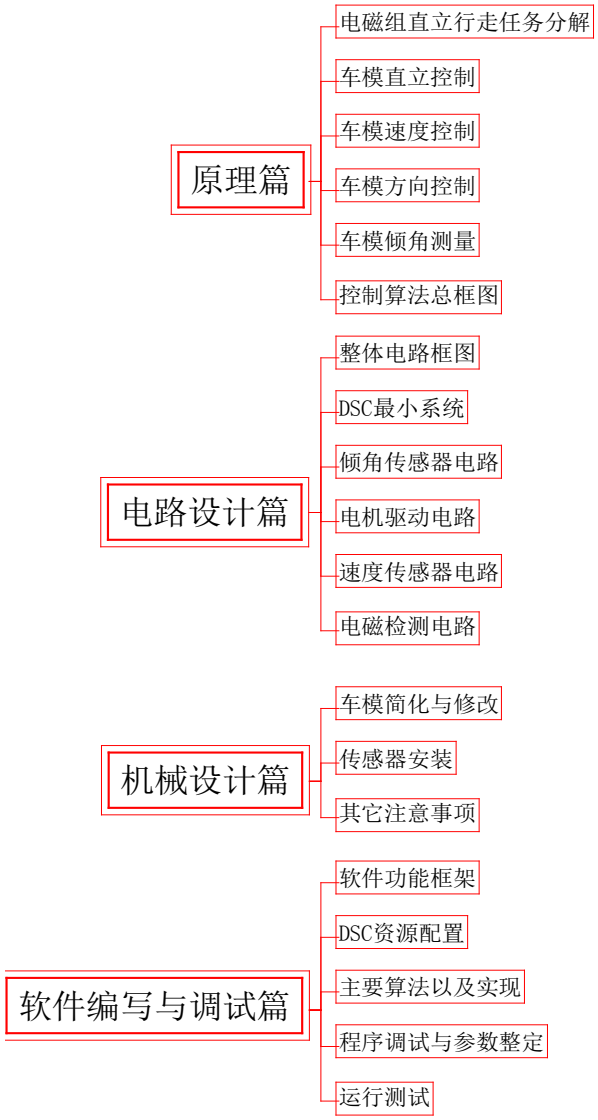


图 1.2 参考设计方案内容

参考设计方案经过了实际验证测试，车模测试运行视频请在竞赛网站上下载。参考设计方案最后附录中给出了用于下载资料的相关网站。

二、原理篇

2.1 直立行走任务分解

电磁组比赛要求车模在直立的状态下以两个轮子着地沿着赛道进行比赛，相比四轮着地状态，车模控制任务更为复杂。为了能够方便找到解决问题的办法，首先将复杂的问题分解成简单的问题进行讨论。

为了分析方便，根据比赛规则，假设维持车模直立、运行的动力都来自于车模的两个后车轮，后轮转动由两个直流电机驱动。因此从控制角度来看，由控制车模两个电机旋转方向及速度实现对车模的控制。车模运动控制任务可以分解成以下三个基本任务：

（1） 控制车模直立：通过控制两个电机正反向运动保持车模直立状态；

（2） 控制车模速度：通过控制两个电机转速速度实现车模行进控制；

（3） 控制车模转向：通过控制两个电机之间的转动差速实现车模转向控制。

以上三个任务都是通过控制车模两个后轮驱动电机完成的。可以假设车模的电机可以虚拟地被拆解成三个不同功能的驱动电机，它们同轴相连，分别控制车模的直立平衡、前进行走、左右转向，如图 2.1 所示。

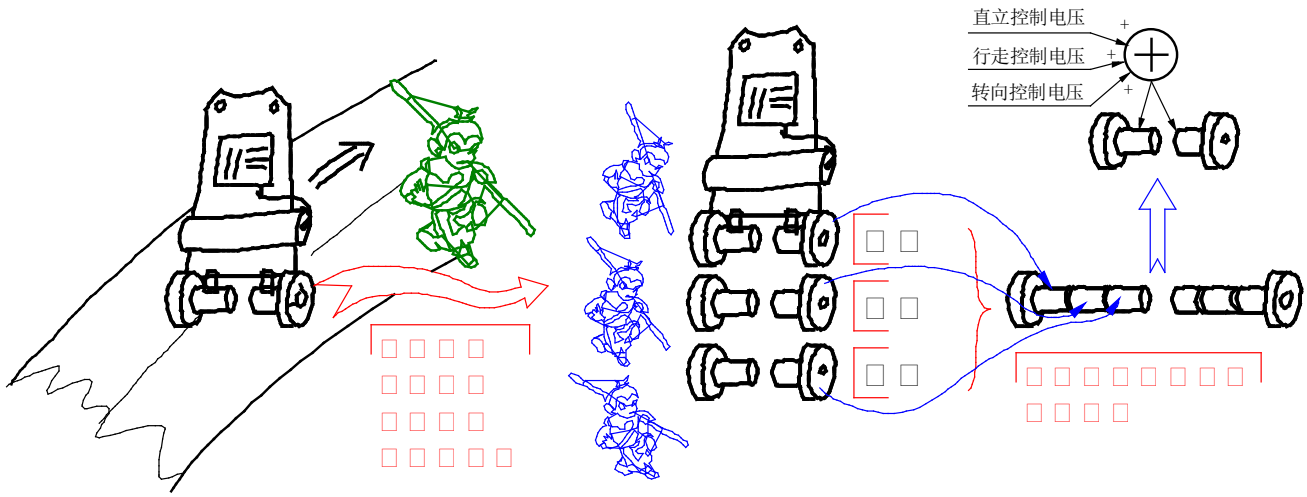


图 2.1 车模运动控制分解示意图

直流电机的力矩最终来自于电机驱动电压产生的电流。因此只要电机处于**线性状态**，上述拆解可以等效成三种不同控制目标的电压叠加之后，施加在电机上。

在上述三个任务中保持车模直立是关键。由于车模同时受到三种控制的影响，从车模直立控制的角度，其它两个控制就成为它的干扰。因此在速度、方向控制的时候，应该尽量平滑，以减少对于直立控制的干扰。三者之间的配合如图 2.1 所示。

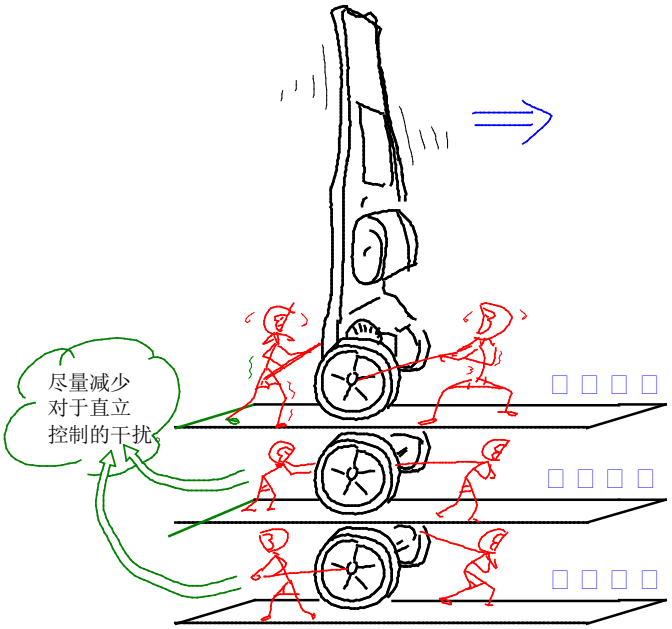


图 2.2 三层控制之间相互配合，底层尽量减少对于上层的干扰

上述三个控制各自独立进行控制，它们各自假设其它两个控制都已经达到稳定。比如速度控制时，假设车模已经在直立控制下保持了直立稳定，通过改变电机的电压控制车模加速和减速。车模在加速和减速的时候，直立控制一直在起作用，它会自动改变车模的倾角，移动车模的重心，使得车模实现加速和减速。

下面分别讨论三个任务的实现原理。

2.2 车模直立控制

控制车模直立的直观经验来自于杂技表演。一般的人通过简单练习就可以让一个直木棒在手指尖上保持直立。这需要两个条件：一个是托着木棒的手掌可以移动；另一个是眼睛可以观察到木棒的倾斜角度和倾斜趋势（角加速度）。通过手掌移动抵消木棒的倾斜角度和趋势，从而保持木棒的直立。这两个条件缺一不可，实际上就是控制中的负反馈机制，参见图 2.3。

世界上还没有任何一个天才杂技演员可以蒙着眼睛使得木棒在自己手指上直立，因为没有了负反馈。

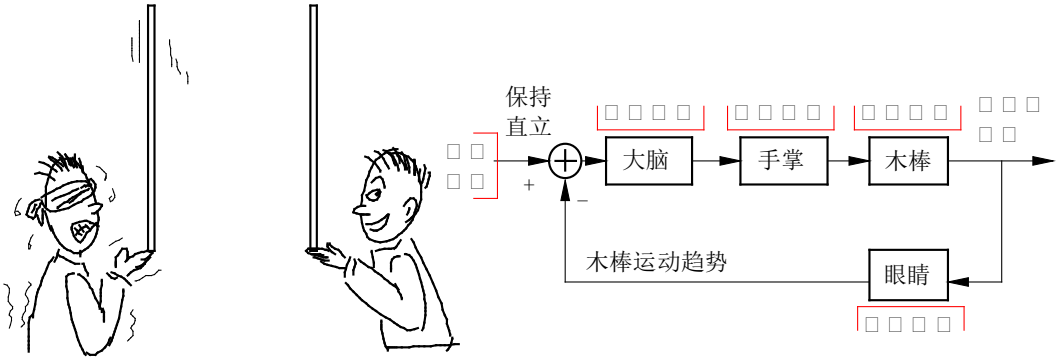


图 2.3 通过反馈保持木棒的直立

车模直立也是通过负反馈实现的。但相对于上面的木棒直立相对简单。因为车模有两个轮子着地，因此车体只会在轮子滚动的方向上发生倾斜。控制轮子转动，抵消倾斜的趋势便可以保持车体直立了。如图 2.4 所示。

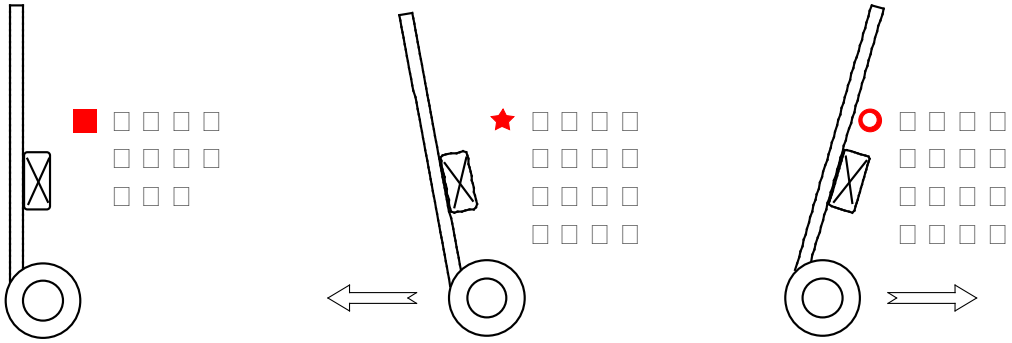


图 2.4 通过车轮运动控制保持车体直立

那么车轮如何运行，才能够最终保持车体垂直稳定？为了回答这个问题，一般的做法需要建立车模的运动学和动力学数学模型，通过设计最优控制来保证车模的稳定。为了使得同学们能够比较清楚理解其中的物理过程。下面通过对比单摆模型来说明保持车模稳定的控制规律。

重力场中使用细线悬挂着重物经过简化便形成理想化的单摆模型。直立着的车模可以看成放置在可以左右移动平台上的倒立着的单摆。如图 2.5 所示。

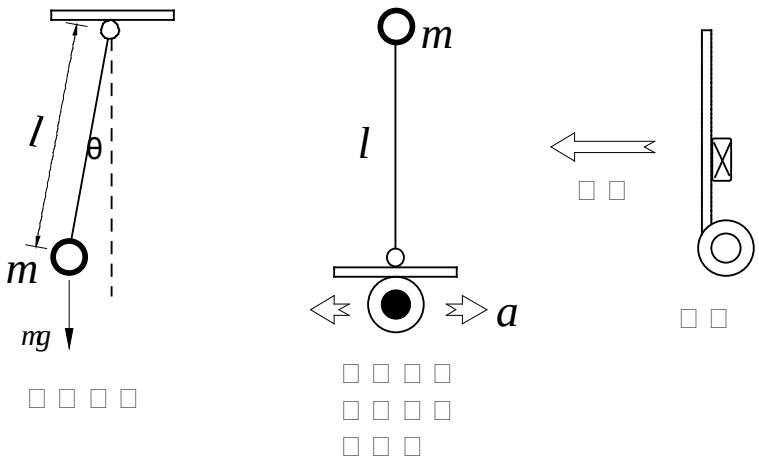


图 2.5 车模可以简化成倒立的单摆

普通的单摆受力分析如图 2.6 所示。

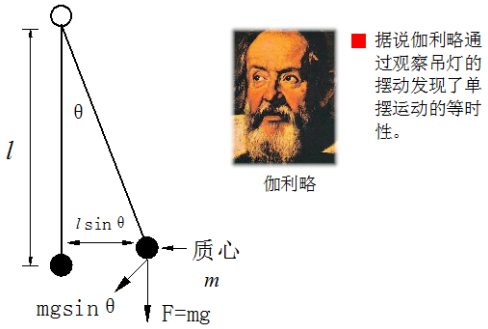


图 2.6 普通的单摆受力分析

当物体离开垂直的平衡位置之后，便会受到重力与悬线的作用合力，驱动重物回复平衡位置。这个力称之为回复力，其大小为

$$F = -mg\sin\theta \approx -mg\theta$$

在此回复力作用下，单摆便进行周期运动。在空气中运动的单摆，由于受到空气的阻尼力，单摆最终会停止在垂直平衡位置。空气的阻尼力与单摆运行速度成正比，方向相反。阻尼力越大，单摆越会尽快在垂直位置稳定下来。图 2.7 显示出不同阻尼系数下，单摆的运动曲线。

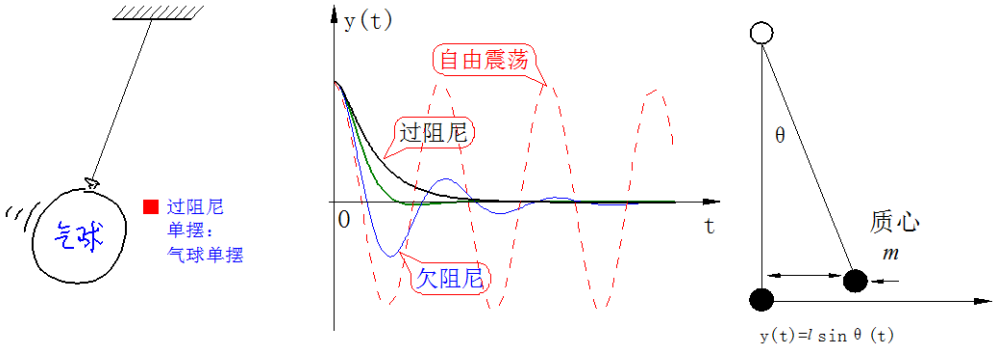


图 2.7 单摆在不同阻尼下的运动情况

总结单摆能够稳定在垂直位置的条件有两个：

（1） 受到与位移（角度）相反的恢复力；

（2） 受到与运动速度相反的阻尼力。

如果没有阻尼力，单摆会在垂直位置左右摆动。阻尼力会使得单摆最终停止在垂直位置。阻尼力过小（欠阻尼）会使得单摆产生震荡，阻尼力过大（过阻尼）会使得单摆到达平衡位置时间拉长。存在一个阻尼临界阻尼系数，使得单摆最快稳定在平衡位置。

为什么倒立摆在垂直位置时，在受到外部扰动的情况下，无法保持稳定呢？分析倒立摆的受力，如图 2.8 所示。

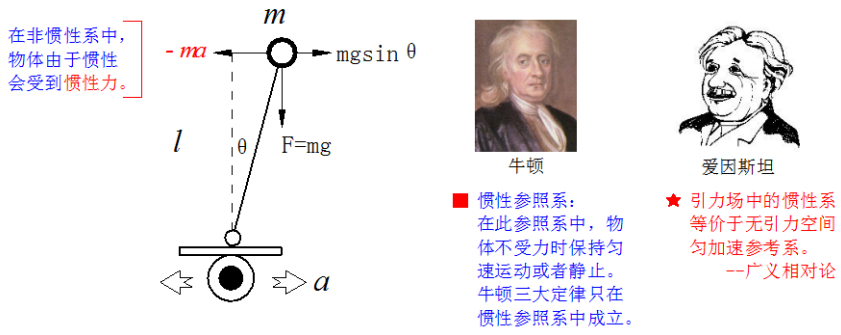


图 2.8 在车轮上参照系中车体受力分析

倒立摆之所以不能象单摆一样可以稳定在垂直位置，就是因为在它偏离平衡位置的时候，所受到的回复力与位移方向相同，而不是相反！因此，倒立摆便会加速偏离垂直位置，直到倒下。

如何通过控制使得倒立摆能够像单摆一样，稳定在垂直位置呢？要达到这一目的，只有两个办法：一个是改变重力的方向；另一个是增加额外的受力，使得恢复力与位移方向相反才行。由此，能够做的显然只有第二种方式。

控制倒立摆底部车轮，使得它作加速运动。这样站在小车上（非惯性系）看倒立摆，它就会受到额外的力(惯性力)，该力与车轮的加速度方向相反，大小成正比。这样倒立摆所受到的回复力为

$$F = mg\sin\theta - ma \approx mg\theta - mk_1\theta \qquad (2\text{-}1)$$

式中，假设控制车轮加速度与偏角θ成正比，比例为k_1。显然，如果$k_1 > g$，（g是重力加速度）那么回复力的方向便于位移方向相反了。

此外，为了使得倒立摆能够尽快地在垂直位置稳定下来，还需要增加阻尼力，与偏角的速度成正比，方向相反。因此式（2-1）可变为

$$F = mg\theta - mk_1\theta - mk_2\theta' \qquad (2\text{-}2)$$

按照上面的控制方法，可把倒立摆模型变为单摆模型，能够稳定在垂直位置。因此，可得控制车轮加速度的控制算法

$$a = k_1\theta + k_2\theta' \qquad (2\text{-}3)$$

式中，θ为车模倾角；θ'为角速度；k_1、k_2均为比例系数；两项相加后作为车轮加速度的控制量。只要保证在$k_1 > g$、$k_2 > 0$条件下，可以维持车模直立状态。其中，k_1决定了车模是否能够稳定到垂直位置，它必须大于重力加速度；k_2决定了车模回到垂直位置的阻尼系数，选取合适的阻尼系数可以保证车模尽快稳定在垂直位置。

因此控制车模稳定，需要下列两个条件：

（1）能够精确测量车模倾角θ的大小和角速度θ'的大小；

（2）可以控制车轮的加速度。

如何测量车模倾角和倾角速度θ,θ'，参见第五小节“车模倾角测量”。如何确定参数k_1、k_2参见“软件调试篇”中的参数调节。

如何控制车模车轮的加速度，参见下一节“车模速度控制”。

2.3 车模速度控制

车模运行速度是通过控制车轮速度实现的，车轮通过车模两个后轮电机经由减速齿

轮箱驱动，因此通过控制电机转速可以实现对车轮的运动控制。

电机的运动控制有三个作用：

（1） 通过电机加速度控制实现车模直立稳定。其中控制规律由上一节给出；

（2）通过电机速度控制，实现车模恒速运行和静止。虽然本届比赛规则中没有要求车模速度恒定，也没有要求车模在比赛之前和冲过终点之后保持静止状态。但是通过速度控制，可以提高车模稳定性。在将来的比赛中，如果规则增加了静止要求，或者需要通过路桥等障碍，速度控制将会发挥作用。

（3）通过电机差速控制，可以实现车模方向控制。差速的控制方法参见下一小节“车模方向控制”。

电机运动控制是通过改变施加在其上的驱动电压实现的。对于电机的电磁模型、动力学模型以及车模的动力学模型进行分析和简化，可以将电机运动模型简化成如下的一阶惯性环节模型。施加在电机上一个阶跃电压 $Eu(t)$，电机的速度变化曲线为

$$\omega(t)=Ek_m\left(1-e^{-\frac{t}{T_1}}\right)u(t) \qquad (2\text{-}4)$$

式中，E 为电压；$u(t)$ 为单位阶跃函数；T_1 为惯性环节时间常数；k_m 为电机转速常数。

对应不同的电压，电机的速度变化曲线如图 2.9 所示。

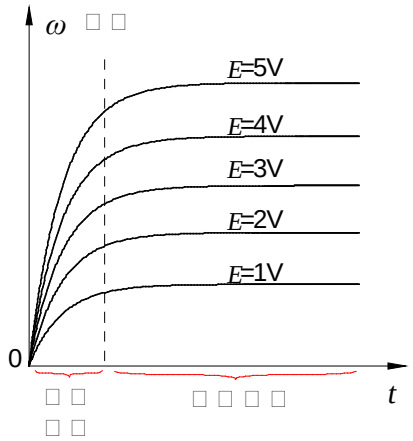


图 2.9 电机在不同电压下的速度

由图 2.9 可以看出，电机运动明显分为两个阶段：第一个阶段是加速阶段；第二个阶段为恒速阶段。其中，在加速阶段，电机带动车模后轮进行加速运动，加速度近似和施加在电机上的电压成正比，加速阶段的时间长度取决于时间常数 T_1，该常数由电机转

动惯量、减速齿轮箱、车模的转动惯量决定；在恒速阶段，电机带动车模后轮进行恒速运行，运行速度与施加在电机上的电压成正比。

调整车模直立时间常数很小，此时电机基本上运行在加速阶段。由上一节式（2-3）计算所得到的加速度控制量 a 再乘以一个比例系数，即为施加在电机上的控制电压，这样便可以控制车模保持直立状态。

车模运行速度调整时间相对很长，此时，电机速度与施加在其上的电压成正比。通过传统的 PID 反馈控制，便可以精确控制电机的运行速度，从而控制车模的运行速度。

电机速度控制需要测量电机的转速，电机旋转速度可以通过安装在电机输出轴上的光电编码盘方便获得。如图 2.10 所示。

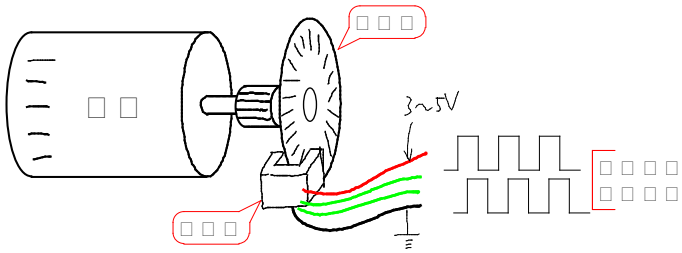


图 2.10 电机速度检测

利用控制单片机的计数器测量在固定时间间隔内速度脉冲信号的个数可以测量电机的转速。

对于电机速度 PID 控制方法如图 2.11 所示。

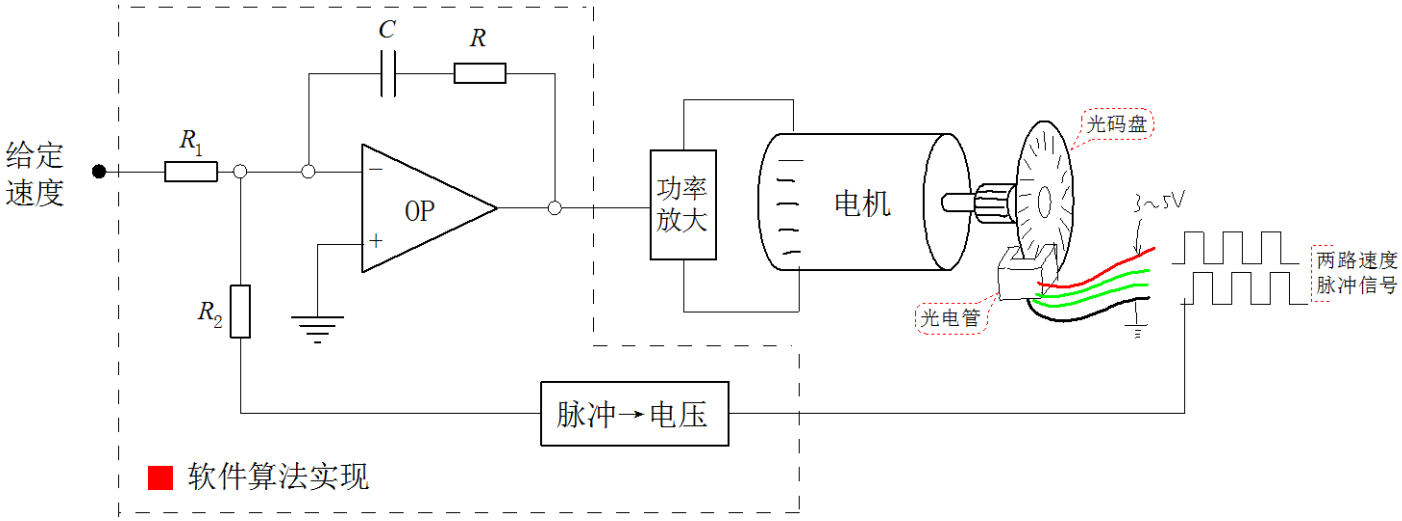


图 2.11 电机 PI 反馈控制

电机速度控制采用了 PI 调节器，具体实现可以通过单片机软件编程实现。

2.4 车模方向控制

实现车模方向控制是保证车模沿着竞赛道路比赛的关键。通过道路电磁中心线偏差检测与电机差动控制实现方向控制。将在下面分别进行介绍。

（1） 道路电磁中心线的偏差检测

道路电磁中心线检测简单的方法可以通过安装在车模前方的两个电磁感应线圈实现。线圈一般采用 10mH 的工字型电感。如图 2.12 所示。

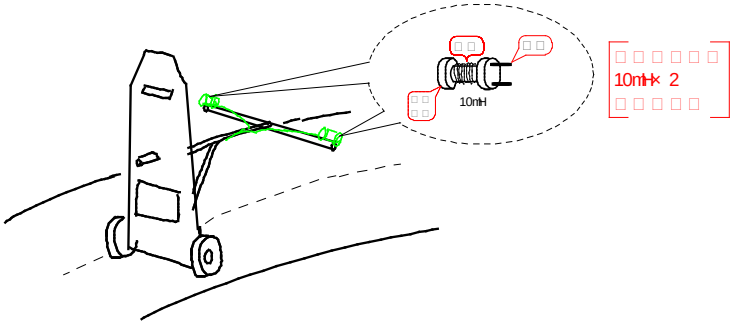


图 2.12 检测道路中心电磁线的方式

详细的参考设计方案请参见文档《电磁组竞赛车模设计参考方案，2010》。

（2） 电机差动控制

利用电磁线偏差检测信号分别与车模速度控制信号进行加和减，形成左右轮差动控制电压，使得车模左右轮运行角速度不一致进而控制车模方向。如图 2.13 所示。

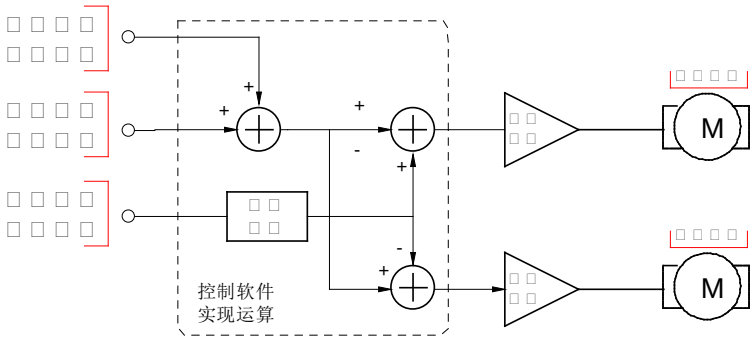


图 2.13 通过差动控制左右电机驱动电压控制车模方向

通过左右电机速度差驱动车模转向消除车模偏差，这个过程是一个积分过程。因此车模差动控制一般只需要进行简单的比例控制就可以完成车模方向控制。

2.5 车模倾角测量

在 2.2 节车模直立控制中介绍了控制车模直立的算法，通过测量车模的倾角和倾角加速度控制车模车轮的加速度来消除车模的倾角。因此车模倾角以及倾角加速度的测量成为控制车模直立的关键。测量车模倾角和倾角加速度可以通过加速度传感器和陀螺仪实现。

（1）加速度传感器

加速度传感器可以测量由地球引力作用或者物体运动所产生的加速度。竞赛规则规定如果车模使用加速度传感器必须使用飞思卡尔公司产生的加速度传感器。该系列的传感器采用了半导体表面微机械加工和集成电路技术，传感器体积小，重量轻。它的基本原理如图 2.14 所示。

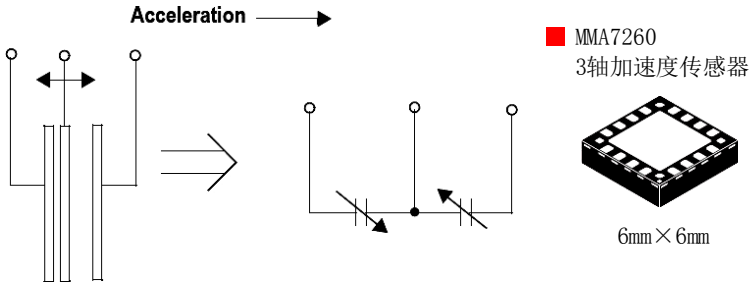


图 2.14 加速度传感器

通过微机械加工技术在硅片上加工形成了一个机械悬臂。它与相邻的电极形成了两个电容。由于加速度使得机械悬臂与两个电极之间的距离发生变化，从而改变了两个电容的参数。通过集成的开关电容放大电路量测电容参数的变化，形成了与加速度成正比的电压输出。MMA7260 是一款三轴低 g 半导体加速度计，可以同时输出三个方向上的加速度模拟信号，如图 2.15 所示。

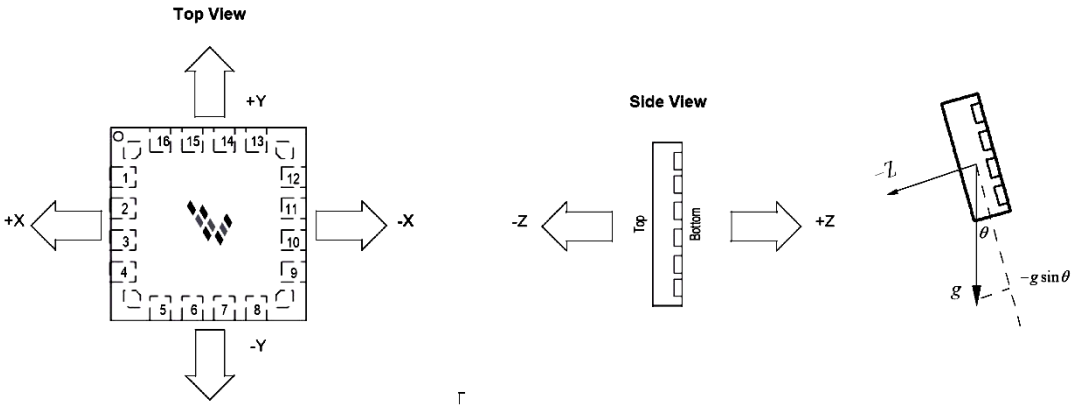


图 2.15 三轴加速度传感器

通过设置可以使得 MMA7260 最大输出灵敏度为 800mV/g。

只需要测量其中一个方向上的加速度值，就可以计算出车模倾角，比如使用 Z 轴方向上的加速度信号。车模直立时，固定加速度器在 Z 轴水平方向，此时输出信号为零偏电压信号。当车模发生倾斜时，重力加速度 g 便会在 Z 轴方向形成加速度分量，从而引起该轴输出电压变化。变化的规律为

$$\Delta u = kg\sin\theta$$

式中， g 为重力加速度； θ 为车模倾角； k 为比例系数。当倾角 θ 比较小的时候，输出电压的变化可以近似与倾角成正比。

似乎只需要加速度就可以获得车模的倾角，再对此信号进行微分便可以获得倾角加速度。但在实际车模运行过程中，由于车模本身的运动所产生的加速度会产生很大的干扰信号叠加在上述测量信号上，使得输出信号无法准确反映车模的倾角，如图 2.16 所示。

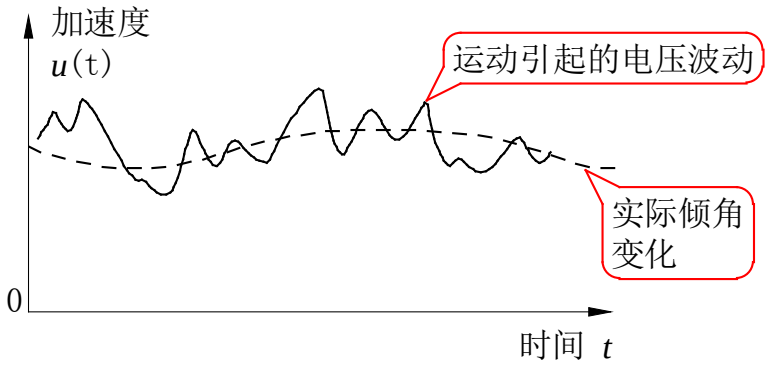


图 2.16 车模运动引起加速度计信号波动

车模运动产生的振动加速度使得输出电压在实际倾角电压附近波动，可以通过数据平滑滤波将其滤除。但是平滑滤波也会使得信号无法实时反映车模倾角的变化，从而减缓对于车模车轮控制，使得车模无法保持平衡。因此对于车模直立控制所需要的倾角信息需要通过另外一种器件获得，那就是角速度传感器-陀螺仪，如图 2.17 所示。

（2）角速度传感器-陀螺仪

陀螺仪可以用来测量物体的旋转角速度。竞赛允许选用村田公司出品的 EN-03 系列的加速度传感器。它利用了旋转坐标系中的物体会受到克里利奥力的原理，在器件中利用压电陶瓷做成振动单元。当旋转器件时会改变振动频率从而反映出物体旋转的角速度。

ENC-03
角速度传感器

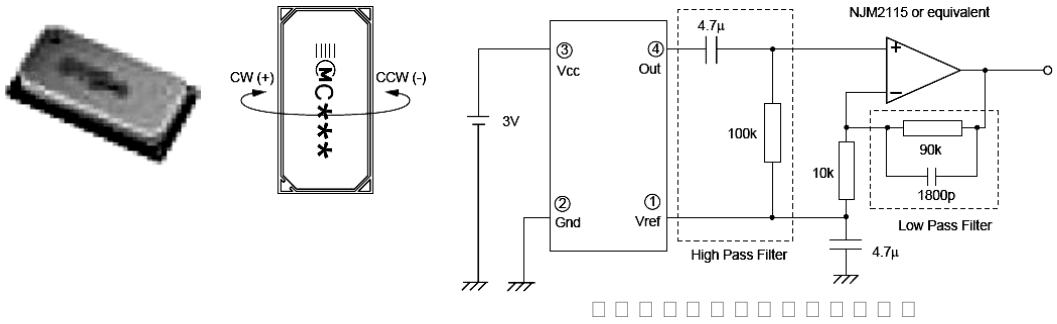


图 2.17 角速度传感器

在车模上安装陀螺仪，可以测量车模倾斜的角速度，将角速度信号进行积分便可以得到车模的倾角。如图 2.18 所示。

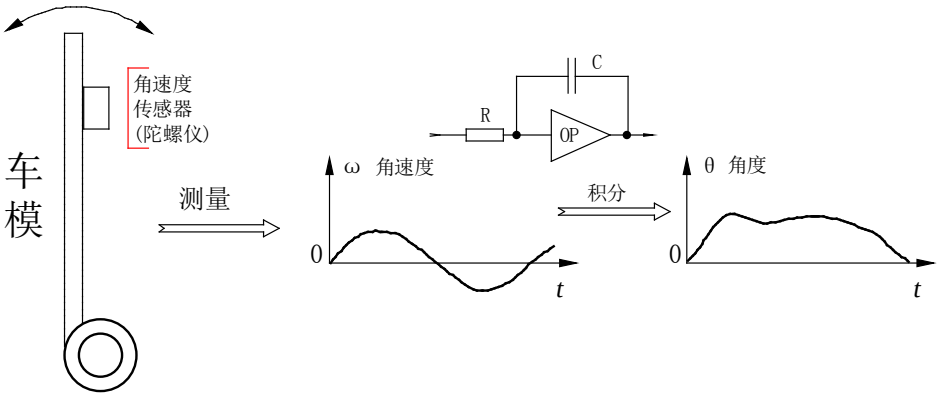


图 2.18 测量车模角速度和角度

由于陀螺仪输出的是车模的角速度，不会受到车体振动影响。因此该信号中噪声很小。车模的角度又是通过对角速度积分而得，这可进一步平滑信号，从而使得角度信号更加稳定。因此车模控制所需要的角度和角速度可以使用陀螺仪所得到的信号。

由于从陀螺仪的角速度获得角度信息，需要经过积分运算。如果角速度信号存在微小的偏差，经过积分运算之后，变化形成积累误差。这个误差会随着时间延长逐步增加，最终导致电路饱和，无法形成正确的角度信号，如图 2.19 所示。

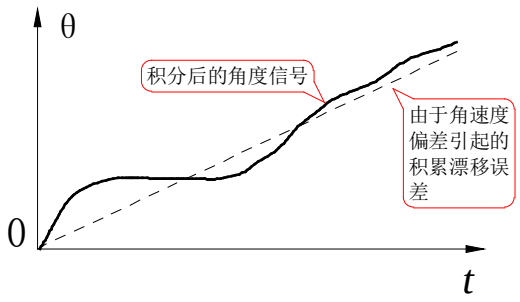


图 2.19 角度积分漂移

如何消除这个累积误差呢？

可以通过上面的加速度传感器获得的角度信息对此进行校正，如图 2.20 所示。

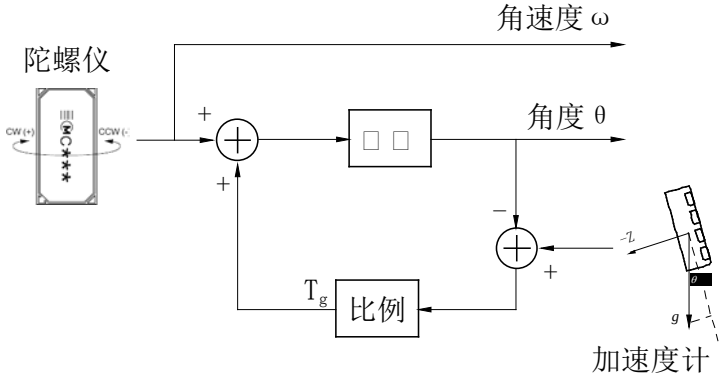


图 2.20 角度积分漂移校正

利用加速度计所获得的角度信息θ_g与陀螺仪积分后的角度θ进行比较，将比较的误差信号经过比例T_g放大之后与陀螺仪输出的角速度信号叠加之后再进行积分。从图 2.20 中的框图可以看出，对于加速度计给定的角度θ_g，经过比例、积分环节之后产生的角度θ必然最终等于θ_g。由于加速度计获得的角度信息不会存在积累误差，所以最终将输出角度θ中的积累误差消除了。

加速度计所产生的角度信息θ_g中会叠加很强的有车模运动加速度噪声信号。为了避免该信号对于角度θ的影响，因此比例系数T_g应该非常小。这样，加速度的噪声信号经过比例、积分后，在输出角度信息中就会非常小了。由于存在积分环节，所以无论比例T_g多么小，最终输出角度θ必然与加速度计测量的角度θ_g相等，只是这个调节过程会随着T_g的减小而延长。

为了避免输出角度θ跟着θ_g过长，可以采取以下两个方面的措施：

（1）仔细调整陀螺仪的放大电路，使得它的零点偏置尽量接近于设定值，并且稳定。

（2）在控制电路和程序运行的开始，尽量保持车模处于直立状态，这样一开始就使得输出角度θ与θ_g相等。此后，加速度计的输出只是消除积分的偏移，输出角度不会出现很大的偏差。

使用加速度计来矫正陀螺仪的积分漂移只是其中一种方法。还可以通过测量车模的运行速度和加速度来矫正陀螺仪的积分漂移，这样就可以省略加速度器件。这种控制方法请同学们在掌握了整个控制方案之后自行设计实现。

2.6 车模直立行走控制算法总图

通过上面介绍，将车模直立行走主要的控制算法集中起来，形成控制算法总框图，如图 2.21 所示。

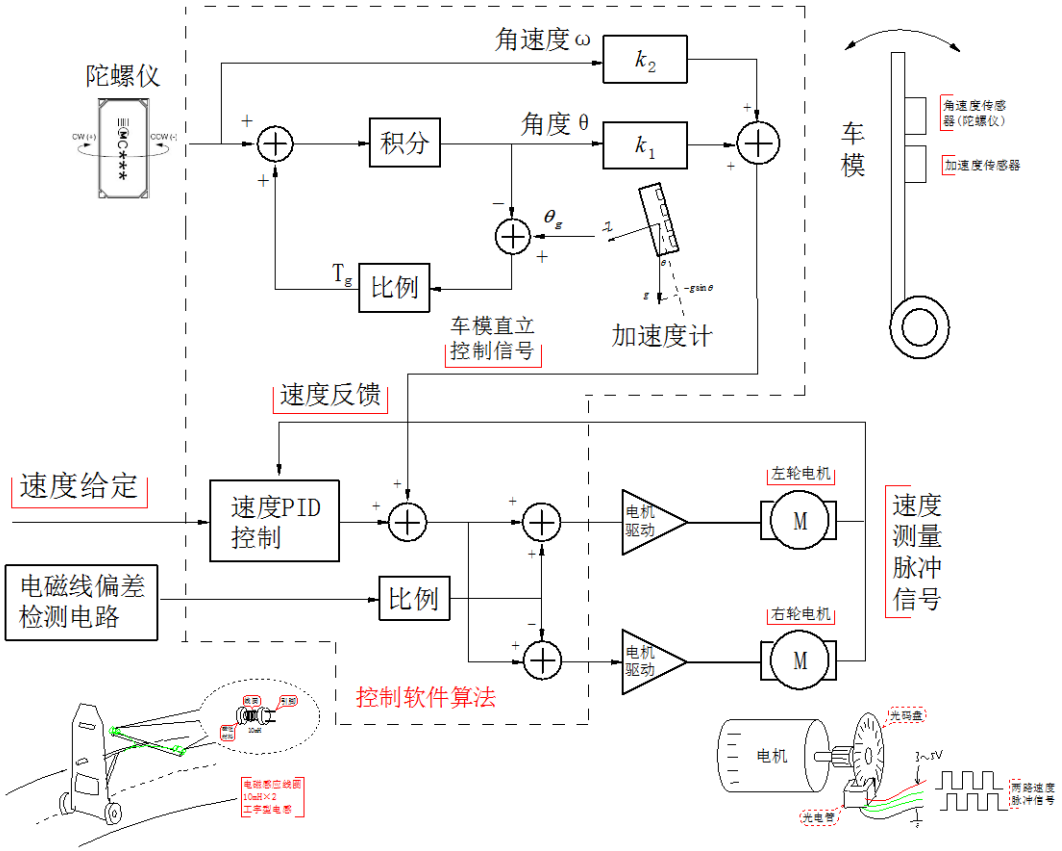


图 2.21 车模运动控制总框图

图 2.21 中，为了实现车模直立行走，需要采集如下信号：

（1） 车模陀螺仪信号；

（2） 车模加速度计信号（z 轴信号）；

（3） 车模电机转速脉冲信号；

（4） 车模电磁偏差信号（两路）。

需要进行如下控制环节，控制车模电机转动：

（1）车模直立控制：使用车模倾角的 PD（比例、微分）控制；

（2）车模速度控制：使用 PI（比例、积分）控制；

（3）车模方向控制：使用 P（比例）控制。

可通过单片机软件实现上述控制算法。

车模的三种控制（直立、速度、方向）最终是将控制量叠加在一起作为电机输出电压控制量。直立控制是基础，它的调整速度非常快，速度和方向控制相对调整速度慢。速度和方向控制的输出量是直接叠加在电机控制电压上。它们假定直立控制会始终保持车模不跌倒，直立控制会自动调节车模的倾角以适应车模的加速、减速和转弯的需要。稍作分析如下：

车模加速前进时，由速度控制算法给出电机增加的正向电压，电机开始逐步加速旋转。在此同时，车模直立控制会迅速进行调整，使得车模往前倾斜，车模开始加速。当车模速度达到设定值，由车模速到控制算法使得电机进入恒速运行。此时车模直立控制算法也会相应调整车模出于直立状态，车模恒速运行。车模减速过程与此类似，由速度控制算法减少了电机的电压，电机开始减速运行。直立控制算法会自动调整车模往后倾斜，使得车模减速。车模转向控制是在车速控制基础之上，调节两个电机驱动电压差使得电机运行速度出现差动，进而调整车模的方向。

在此控制算法中，直立控制一直维持车模的直立状态，速度与方向控制将会成为直立控制的外部干扰。为了确保车模不会跌倒，因此外部的速度和方向控制算法调整速度不能够过快，过于剧烈。这一点在后面软件实现的时候需要注意。

三、电路设计篇

3.1 整体电路框图

设计车模控制系统的电路，首先需要分析系统的输入、输出信号，然后选择合适的核心控制嵌入式计算机（单片机），逐步设计各个电路子模块，最后形成完整的控制电路。

系统的输入输出包括：

（1） AD 转换接口（至少 4 路）

a) 电磁监测：左右两路，用于测量左右两个感应线圈电压。

b) 陀螺仪：一路，测量陀螺仪输出电压。

c) 加速度计：一路，测量加速度 Z 轴输出电压。

d) 辅助调试：（备用）1 到 3 路，用于车模调试、设置作用。

（2） PWM 接口（4 路）

a) 控制左右两个电极双方向运行，需要四路 PWM 接口。

（3） 定时器接口（2 路）

a) 测量两个电机转速，需要两个定时器脉冲输入端口。

（4） 通讯接口（备用）

a) SCI（UART）：一路，用于程序下载和调试接口；

b) I2C:（备用）如果选择飞思卡尔公司的数字加速度计，可以通过 I2C 接口直接读取加速度值。

（5） **IO 接口**（备用）

4 到 8 路输入输出，应用车模运行状态显示，功能设置等。

竞赛允许使用飞思卡尔公司处理器系列，绝大部分都能够满足上面的控制要求。本文选择其中的 DSC 16 位处理器 MC56F8013 作为核心的控制处理器为例加以说明，车模控制电路整体框图如图 3.1 所示。

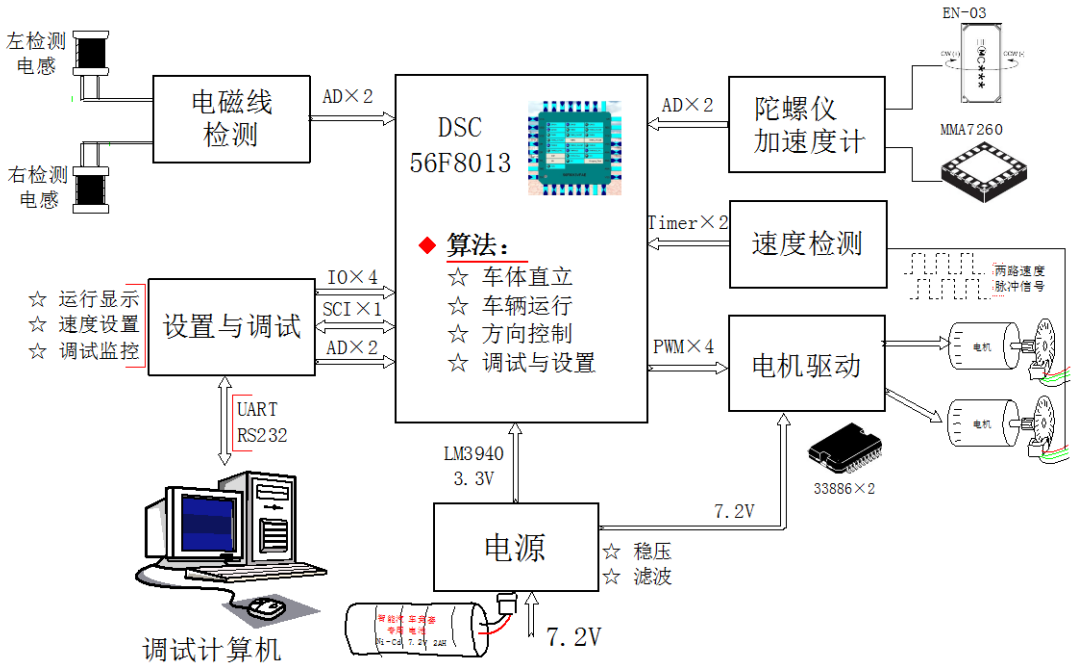


图 3.1 车模控制电路整体框图

根据图 3.1，控制电路划分为如下子模块：

（1） 单片机最小系统：包括 DSC 处理器，程序下载调试接口等；

（2） 电磁线检测：包括两路相同的电磁感应信号放大与检波电路；

（3） 陀螺仪与加速度计：包括两个姿态传感器信号放大滤波电路；

（4） 速度检测：检测电机光电码盘脉冲频率；

（5） 电机驱动：驱动两个电极运行电路；

（6） 电源：电源电压转换、稳压、滤波电路；

（7） 设置与调试：显示系统运行状态、速度设定、程序下载与监控。

以下将分别对以上电路给出设计参考方案。

3.2 DSC 介绍与单片机最小系统

单片机选择飞思卡尔公司 DSC MC56F8013，它体积小（32PIN TQFP），功耗低（3.3V 工作电压），运算速度快（32MIPS，DSP 结构），具有丰富的外设模块，非常适合控制车模运行。它的主要外设包括：

（1）PWM 6 通道；

（2）AD 6 通道，12bit；

（3）定时器，16bit，4 通道；

（4）外部串行接口：SCI，I2C，SPI；

（5）IO 口：最多可以提供 26 路。

此外，内部还集成了时钟电路、电源检测电路以及看门狗电路等。

内部存储器资源包括：16k 程序 Flash，4k 数据 RAM。图 3.2 显示该单片机的内部资源情况。

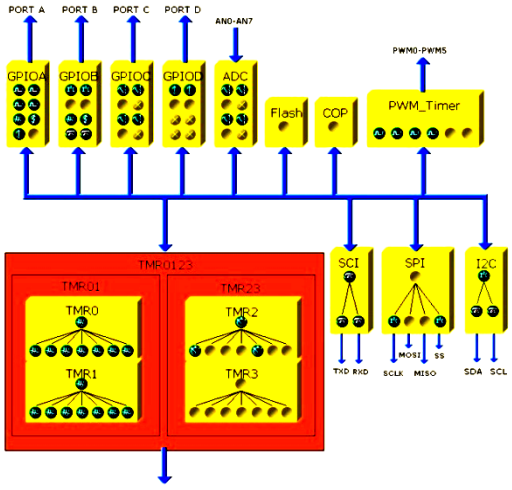


图 3.2 56F8013 内部资源示意图

单片机的最小系统电路如图 3.3 所示。

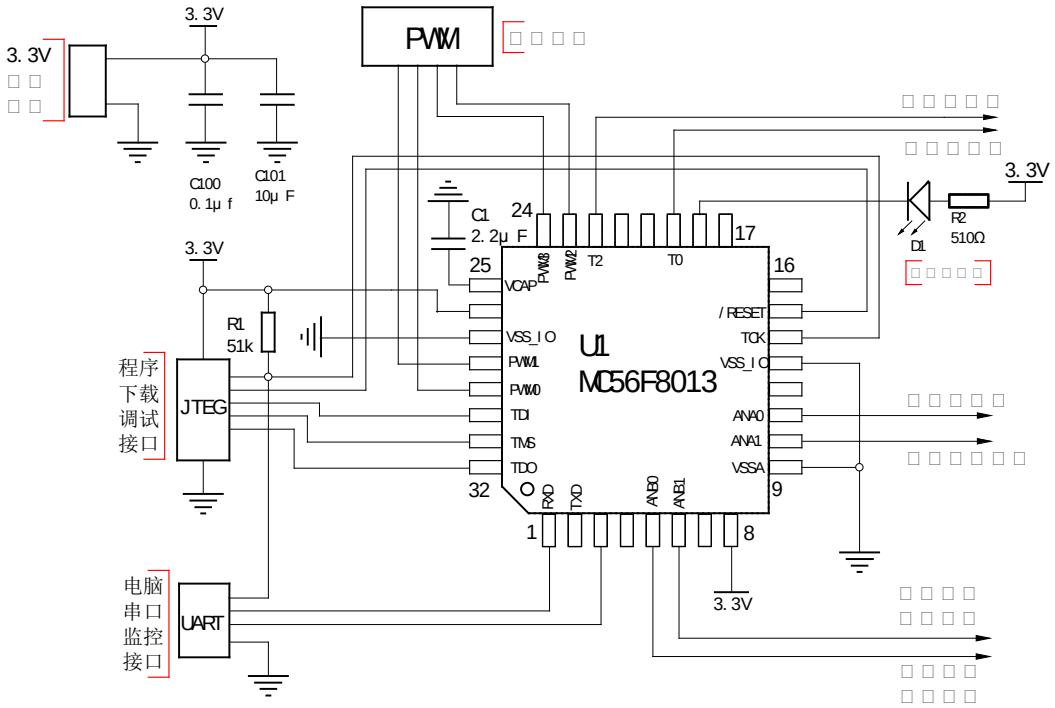


图 3.3 DSC 最小系统电路图

所使用 F8013 单片机的资源包括：

（1） 模拟量检测接口

a) ANA0（PIN12）：陀螺仪电压；

b) ANA1（PIN11）：加速度计电压；

c) ANB0（PIN5）：右侧电感检波电压；

d) ANB1（PIN6）：左侧电感检波电压。

（2） 电机转速脉冲接口

a) T0（PIN19）：右侧电机光电码盘脉冲；

b) T2（PIN22）：左侧电机光电码盘脉冲。

（3） 电机 PWM 驱动接口

a) PWM0-3（PIN23,24,28,29）：电机驱动。

（4） 程序下载调试 JTEG 接口

a) TDI（PIN30）；

b) TDO（PIN32）；

c) TMS（PIN31）；

d) TCK（PIN14）；

e) /RESET(PIN15)。

（5） 串口监控 UART 接口：

a) RXD（PIN1）；

b) TXD（PIN3）。

其它没有使用的 IO 和模拟量口可以用作状态显示、运行设置以及辅助调试作用。

使用快速制板方法制作的最小单片机系统实物图如图 3.4 所示。

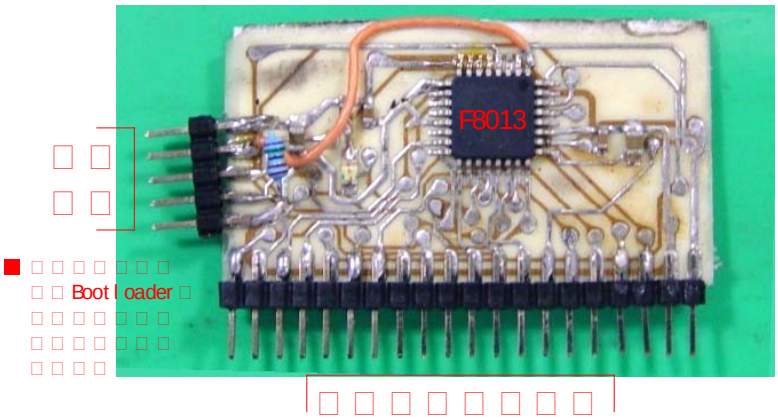


图 3.4 最小 DSC 系统实物图

图 3.4 显示的实际电路中，将外部所需要的各个端口通过总线方式引出，通过统一的接口底板连接其他各子电路。

上面电路板也没有 JTEG 的程序接口。程序的开发主要是通过单片机内部的 Bootloader 程序完成程序的下载和调试的，因此单片机最小系统只需要一个 UART 接口便可以进行，无需额外的调试器。相关的资料可以参见飞思卡尔公司网站中的介绍。

3.3 倾角传感器电路

车模倾角传感器电路主要是将陀螺仪信号进行放大滤波。由于加速度传感器采用是低 g 值的传感器 MMA7260，它的输出信号非常大，不需要再进行放大。电路图如图 3.5 所示。

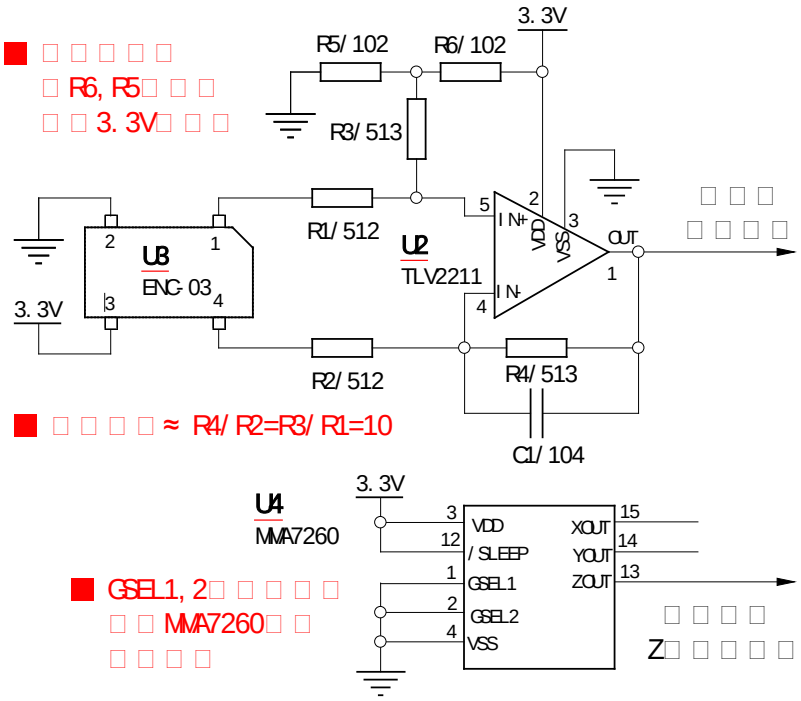


图 3.5 陀螺仪、加速度计电路

图 3.5 中，将陀螺仪的输出信号放大了 10 倍左右，并将零点偏置电压调整到工作电源的一半（1.65V）左右。放大倍数需要根据选取的传感器输出灵敏度设计。

将上述电路单独制作成小的电路板，可以比较方便放置在车模的最稳定的位置。实际的电路图如图 3.6 所示。

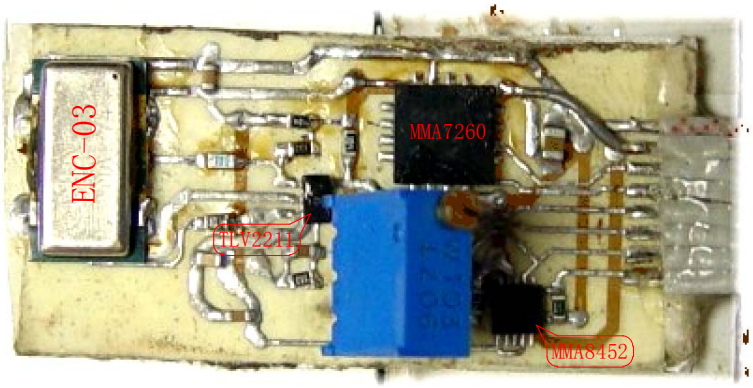


图 3.6 传感器电路

请注意，上面的电路除了陀螺仪和加速度器之外，还包括了一个用于电压设定的电位器以及一个 I2C 总结接口的加速度计 MMA8452，这些器件并没有包含在上面电路原理图中，它们只是用来作为对比实验。

3.4 电机驱动电路

由于车模具有两个后轮驱动电机，因此需要两组电机驱动桥电路。图 3.7 选用了两片飞思卡尔公司专用电机驱动芯片 33886 组成了电机驱动电路。

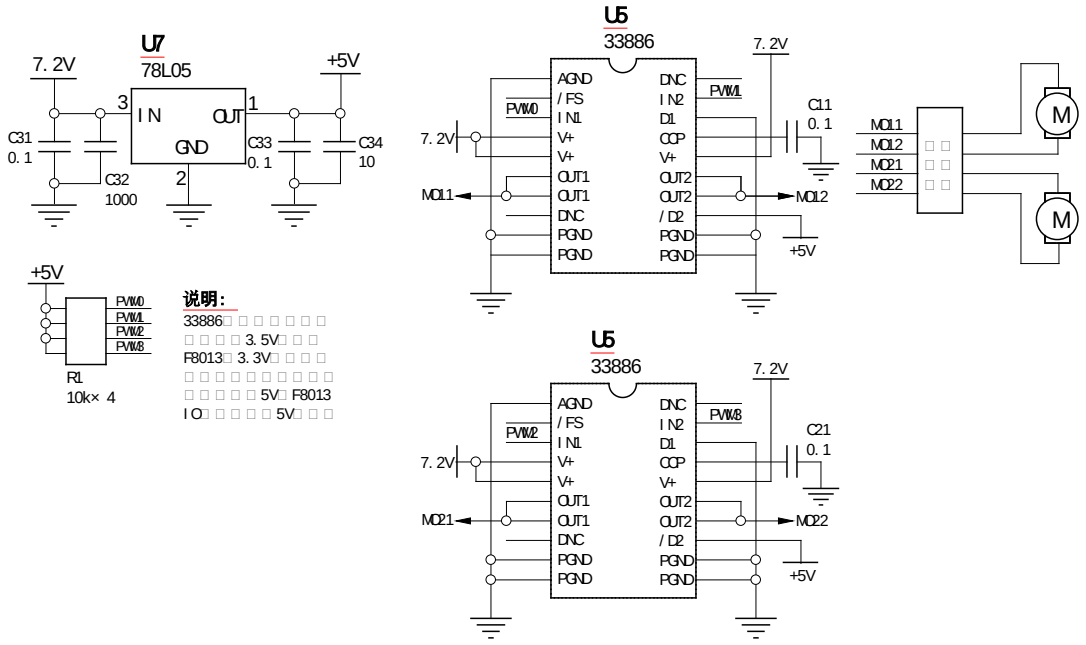


图 3.7 电机驱动电路

图 3.7 中，由于 DSC F8013 是 3.3V 器件，它的 IO 输出电压最高位 3.3V，达不到 33886 对于高电平必须大于 3.5V 的要求，所以在电路中专门设计了 5V 电源，将 33886 的驱动信号上拉至 5V。由于 F8013 的 IO 端口可以容忍 5V 电压，所以上面的电路便可以使得 33886 的驱动信号电压达到 5V。

为了提高电源的应用效率，驱动电机的 PWM 波形采用了单极性的驱动方式。也就是在一个 PWM 周期内，施加在电机上的电压为一种电压，如图 3.8 所示。

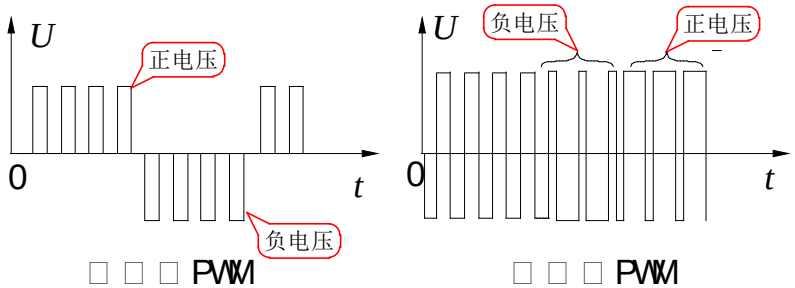


图 3.8 单极性 PWM、双极性 PWM

因此每一路电机为了能够实现正反转，都需要两个 PWM 信号。两个电机总共需要

4 路 PWM 信号。具体实现的驱动电路如图 3.9 所示。

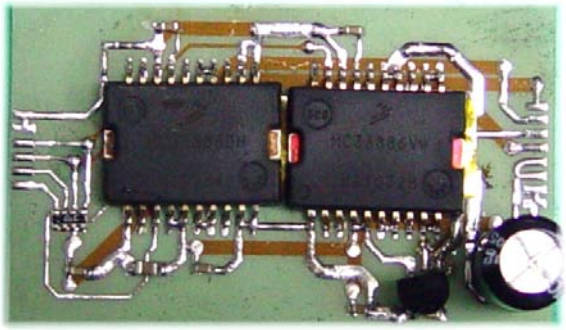
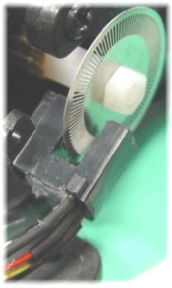


图 3.9 两片 33886 组成的电机驱动电路

图 3.9 中，为了防止电机输出电流对于电源的冲击，在电路板的电源输入（7.2V）端口并联了一个 1000 微法的电容。

3.5 速度传感器

电机速度传感器使用了固定在电机输出轴上的光码盘以及相互配合的光电对管器件，如图 3.10 所示。



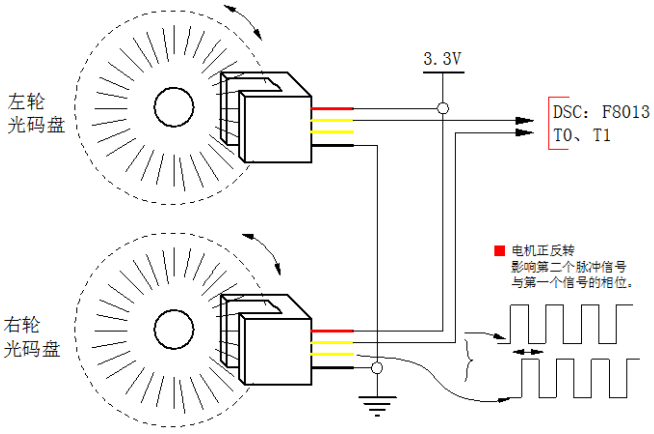


图 3.10 速度传感器电路

由于光电管器件直接输出数字脉冲信号，因此可以直接将这些脉冲信号连接到单片机的计数器端口。

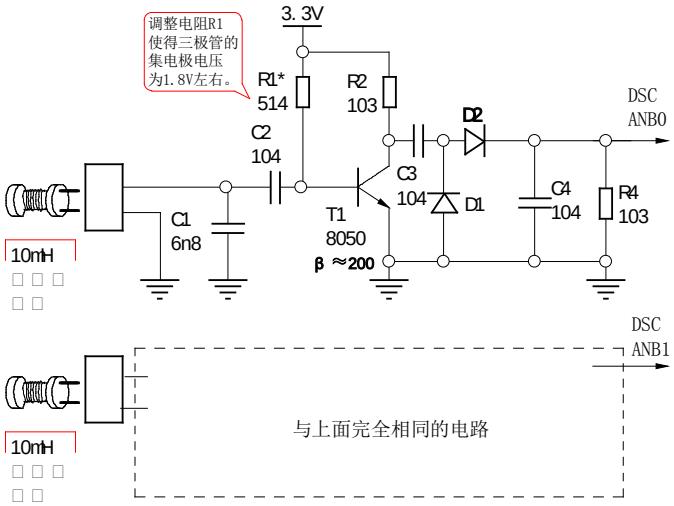
每个光电管输出两个脉冲信号，它们波形相同，只是相位相差 90°。如果电机正转，第二个脉冲落后 90°；如果电机反转，第二个脉冲超前 90°。可以通过这个关系判断电机是否正反转。在实际电路中，只检测了一路脉冲信号。通过他的频率测量得到电机的转速。电机的转向是通过施加在电机上的电压正负进行判断的。通过实验验证这个方

法可以有效判断电机的转动方向并进行速度控制。

3.6 电磁线检测电路

道路中心线的电磁线检测是保证车模能够运行在赛道上。由于电磁组在第五届竞赛中已经设立，在 2010 年竞赛秘书处公布了电磁线检测的参考设计方案。详细设计原理和电路请参见附录 1 中的参考文献。

图 3.11 给出了其方案的电路图。



3.11 电磁检测电路

根据上面电路制作的电路图如图 3.12 所示。

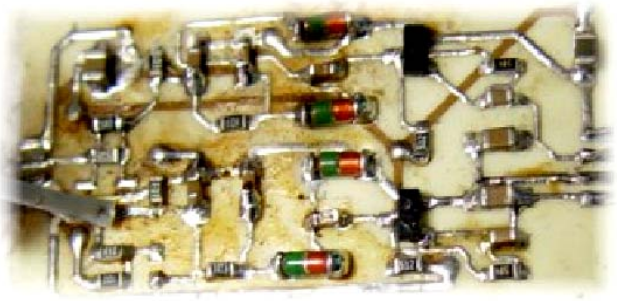


图 3.12 电机检测电路板

以上给出了车模控制电路的主要电路设计，此外还有电源管理电路、设置与监控电路、程序调试与运行接口电路等。同学们可以根据往届竞赛技术报告进行相关的设计。

四、机械设计篇

良好的车模机械设计与制作，对于车模稳定运行、安全调试都非常重要。如下仅就车模简化改装与传感器安装两个方面进行讨论。

4.1 车模简化改装

由于今年电磁组车模采用了原来竞赛 C 型车模，它是双后轮驱动，前轮舵机转向的运动模式，而竞赛规定 C 型车模直立行走，因此车模前轮以及部分相关部件都可以进行简化。具体可以参照以下改装步骤：

（1） 去掉前轮及其支撑部件，去掉后轮悬挂缓冲支架

拆卸后的情况如图 4.1 所示。

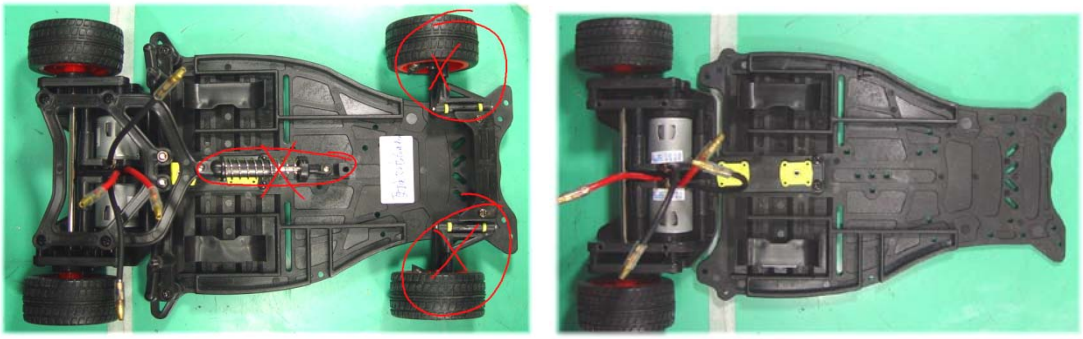


图 4.1 左图：完整的 C 型车模底盘；右图：简化后的 C 型车模底盘。

（2） 固定车模底盘与后轮支架

原有车模为了减轻后轮振动对于车体的影响，后轮的支架与底盘之间采用了活动连接方式。但是，为了保证车模直立车体稳定性，需要将原有车模地盘与后轮支架固定在一起。最简便的方式就是可以使用热熔胶在后轮支架与底盘之间的缝隙处进行粘接。这样后轮与车体之间形成一个刚体，便于进行直立控制。图 4.2 所示为热熔胶粘接的位置。

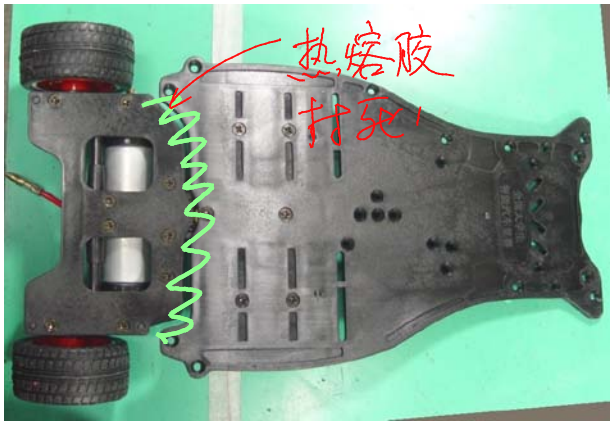


图 4.2 使用热熔胶粘接后轮支架与底盘。

（3） 去掉后轮，准备安装电机测速光码盘

拆装后轮可以使用原车模配备的十字套筒扳手。拆卸后的车模如图 4.3 所示。

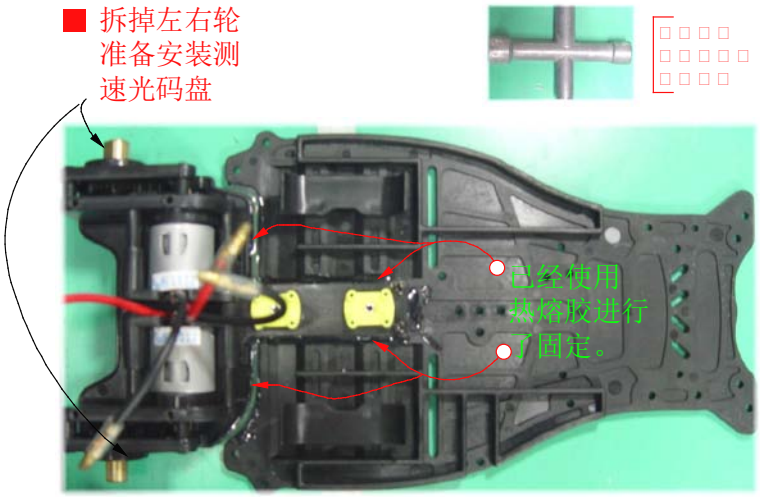


图 4.3 拆掉后轮之后的车模

在图 4.3 中，请注意使用热熔胶固定后轮支架与车模底盘的情况。

在安装测速传感器之后，再将两个后轮重新安装在后轮支架上。

4.2 传感器安装

车模中的传感器包括有：速度传感器，车模姿态传感器（陀螺仪、加速度计）以及电磁检测感应线圈。下面分别介绍这些传感器的安装。

（1）速度传感器安装

速度传感器是安装在驱动电机输出轴上的光电码盘以及相配合的光电对管。固定光

电码盘可以使用复合胶水，将光电码盘的塑料轴与电机的输出铜轴粘合。如图 404 所示。

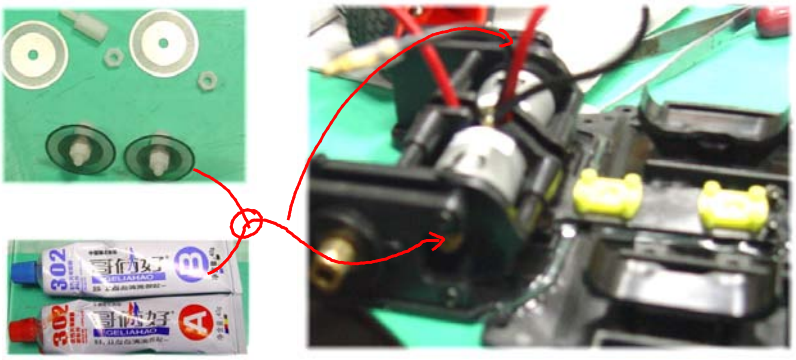


图 4.4 使用复合胶水将光电码盘固定在电机输出轴上

粘合码盘之后，再固定检测光电对管。此时可以使用热熔胶将光电对管固定在后轮支架上固定光电对管，使得光电码盘在对管的中间缝隙中通过。如图 4.5 所示。

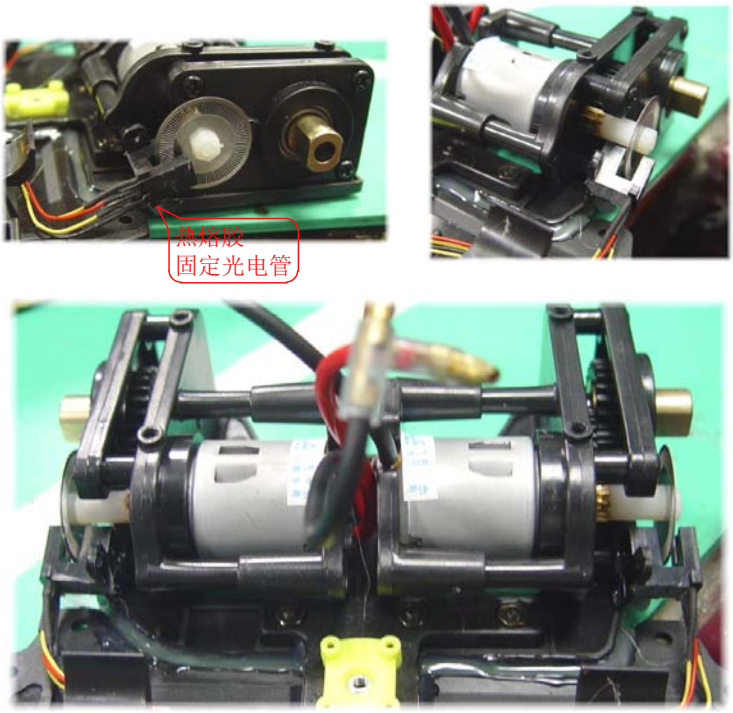


图 4.5 固定好码盘和光电对管的后轮

请注意，码盘的固定一定要使用热稳定性比较好复合胶水，比如“哥俩好”。由于电机在运行过程中发热，而且振动大，所以千万不要使用热熔胶固定码盘。

上述测速传感器固定完毕之后，便可以将后轮重新安装在后轮的支架上。

（2）电磁传感器安装

电磁传感器为两个工字型的 10 毫亨电感。为了能够更好的检测前面的道路，一般

将这两个工字型的电感尽可能的安装在车模运行前方较远的地方。由于车模是直立运行，可以考虑如下的参考方案，如 4.6 所示。

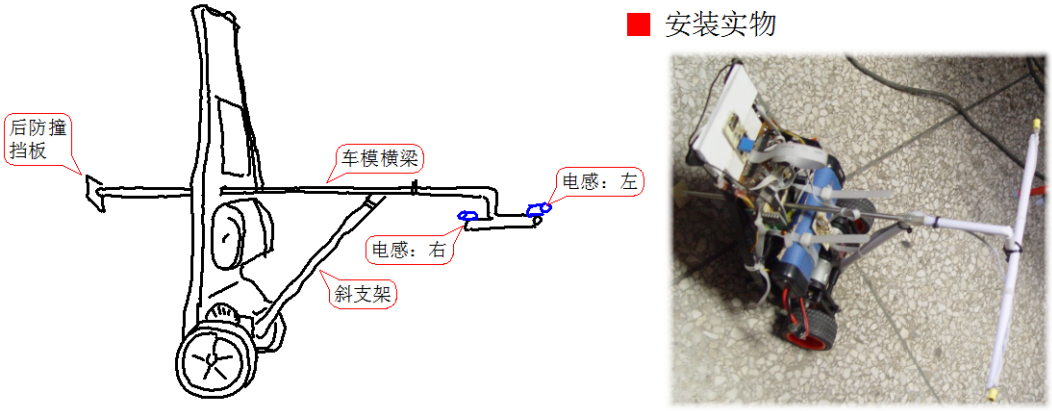


图 4.6 电磁传感器安装支架示意图

请注意，竞赛规则对于车模外形尺寸的限制。

由于本届比赛没有要求车模在竞赛完毕后自动停止在其实现后三米的赛道上，因此无需检测赛道起始线上的永磁铁。

为了避免车模在通过十字交叉线路口时受到的影响，在固定两个电感线圈的时候，尽量保持这两个线圈呈水平位置。

（3）车模倾角传感器

车模倾角传感器包括陀螺仪和加速度计。它们都是表贴元器件，固定在电路板上。一般建议将这块带有陀螺仪和加速度计的电路板固定在整个车模中间质心的位置。这样可以最大程度减少车模运行时前后振动对于测量倾角的干扰。

4.3 注意事项

由于车模直立运行，在制作电路板的时候尽可能减少电路板的尺寸，一方面便于固定，另一方面可以减少车模的惯量。

固定电路板应尽可能贴近车模的底盘，使其能够稳固。

为了避免车模运行过程中倾倒，摔坏车模及其上的电路板，在车模机械设计的时候，

需要考虑在车模前后安装有防撞支架或者缓冲物，一旦车模倾倒或者失控，防撞支架可以保护车模机械的安全性。

此外，也可以在防撞支架上安装车模跌倒检测开关，一旦车模倾倒，控制电路便立即停止运行。

五、软件编写与调试篇

通过前面的介绍，车模控制电路制作与安装均已完毕。车模是否能够正常高速稳定运行，需要通过软件编写和调试来完成。软件编写与调试主要任务包括：

（1）建立软件工程，配置 DSC 资源；

（2）编写单片机软件程序框架，编写上位机监控软件，建立软件编译、下载、调试的环境；

（3）实现并测试各个子模块的功能正确性；

（3）逐步完成车模闭环控制，整定各个待定参数；

（4）进行车模整体运行性能测试与提高。

开发飞思卡尔公司 56800/E 系列 DSC 单片机应用程序可以使用 CodeWarrior 集成开发环境（CodeWarrior for 56800/E Digital Signal Controllers，目前版本 v5.9）。与其它版本的 CodeWrrior 一样，DSC 版 CodeWarrior 也提供了 Processeor Expert 功能模块，可以通过工程配置非常方便地生成单片机的各个外设的初始化代码和接口程序，帮助开发者将精力集中在应用程序的开发上。

5.1 软件功能与框架

软件的主要功能包括有：

（1）车模运行状态检测；

（2）电机 PWM 输出；

（3）车模运行控制：直立控制、速度控制、方向控制；

（4）车模运行流程控制：程序初始化、车模启动与结束；

（5）系统界面：状态显示、上位机监控、参数设定等。

上述功能可以分成两大类：

第一类包括 1-3 功能，它们属于需要精确时间周期执行，因此可以在一个周期定时

中断里完成。第二类包括 4-5 功能。它的执行不需要精确的时间周期。可以放在程序的主程序中完成。这两类任务之间可以通过全局变量实现相互的通讯。

主程序框架如图 5.1 所示。

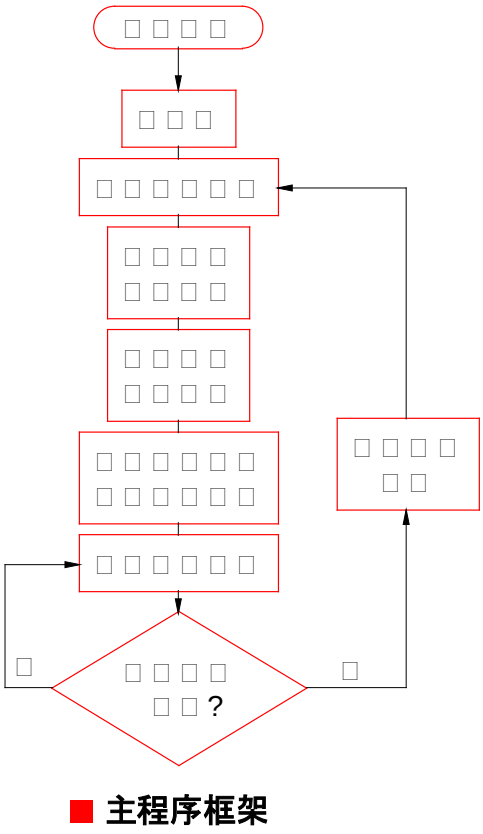


图 5.1 主程序框架

图 5.1 中，程序上电运行后，便进行单片机的初始化。初始化的工作包括有两部分，一部分是对于单片机各个应用到的模块进行初始化。这部分的代码由 CodeWarrior 集成环境的 ProcessorExpert 工具生成。第二部分是应用程序初始化，是对于车模控制程序中应用到的变量值进行初始化。

初始化完成后，首先进入车模直立检测子程序。该程序通过读取加速度计的数值判断车模是否处于直立状态。如果一旦处于直立状态则启动车模直立控制、方向控制以及速度控制。

程序在主循环中不停发送监控数据，在通过串口发送到上位机进行监控。同时检查车模是否跌倒。跌倒判断可以通过车模倾角是否超过一定范围进行确定，或者通过安装在车模前后防撞支架上的微动开关来判断。一定车模跌倒，则停止车模运行。包括车模

直立控制、速度控制以及方向控制。然后重新进入车模直立判断过程。

车模的直立控制、速度控制以及方向控制都是在中断程序中完成。通过全局标志变量确定是否进行这些闭环控制。

中断程序框架如图 5.2 所示。

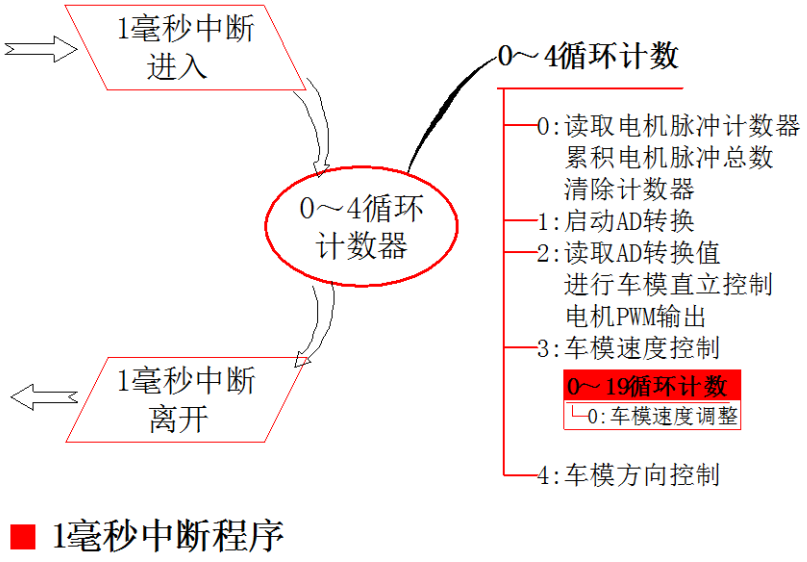


图 5.2 中断服务程序框架

图 5.2 中，使用 DSC 的一个定时器，产生一毫秒的周期中断。中断服务程序的任务被均匀分配在 0-4 的中断片段中。因此每个中断片段中的任务执行的频率为 200Hz。

将任务分配到不同的中断片段中，一方面防止这些任务累积执行时间超过 1 毫秒，扰乱一毫秒中断的时序，同时也考虑到这些任务之间的时间先后顺序。

这些任务包括：

（1）电机测速脉冲计数器读取与清除。累积电机转动角度。累积电机速度，为后面车模速度控制提供平均数；

（2）启动 AD 转换。由于 AD 转换启动到完成需要一定时间。所以读取 AD 转换在下一个时间片段中。

（3）读取 AD 转换值。这些值包括有陀螺仪、加速度计数值、电磁场检测电压值等。读取完毕之后，便进行车模直立控制过程。包括车模角度计算、直立控制计算、电机 PWM 输出等。

（4）车模速度控制：在这个时间片段中，又进行 0-19 计数。在其中第 0 片段中，进行速度 PID 调节。因此，速度调节的周期为 100 毫秒。也就是每秒钟调节 10 次。

（5）车模方向控制：根据前面读取的电磁场检波数值，计算偏差数值。然后计算电机差模控制电压数值。

5.2 DSC 的资源配置

下面说明程序所用到 DSC 的模块资源及其配置。如果使用其它系列的处理器，可以参照这些配置进行设置。

1）毫秒定时中断：TI1

硬件模块：TMR2_Compare

中短周期：Period:1ms

触发事件：Event:Interrupt

2）电机ＰＷＭ输出控制：PWMC1

硬件模块：PWM_Timer

输出频率：Frequence:10Khz

输出模式：PWM0,1,2,3 Independent

死区时间：Dead Time:0

3）ADC 采集通道：ADC

硬件模块：ANA0,1,ANB0,1

数值范围：Range:0 - 0x7ff0

转换时间：Conversion Time:1.594us

采集模式：Mode: Sequency

转换分辨率：Resolution:12bit

4）电机测速脉冲计数器：Counter1

硬件模块：TMR0

计数范围：16bit

信号触发沿：Rising Edge

5）电机测速脉冲计数器：Counter2

硬件模块：TMR1

计数范围：16bit

信号触发沿：Rising Edge

6） 监控 UART 串口

硬件模块 SCI

通讯速率：Baud:117647（实际值）

7）读取 MMA8452Q 的 I2C 总线（备用）

硬件模块：I2C

中断模式：Interrupt Service:Disable

I2C 地址：Address:4c

通讯速率：SCL Frequency:200kHz

注：这个 I2C 总线控制器用在读取飞思卡尔公司数字接口的加速度传感器上。

注意：上面的这些配置主要是通过 CodeWarrior 的 Processor Expert（PE）来完成，PE 会自动生成这些模块的初始化程序代码，以及接口程序代码。

5.3 主要算法及其实现

下面将主程序框架、中断程序框架中的各个主要子程序功能及其程序实现进行介绍。

1) 电机 PWM 输出子程序

包括两个子程序：

第一个程序：SetMotorVoltage

设置四个 PWM 输出数值。

函数输入参数：

nLeftVol，nRightVol：分别表示左右两个电机输出电压数值。

取值范围：-0x8000 至 0x7fff：代表定点小数 -1.0 至 1.0。正负号表示电机的正反转。

子程序中调用了 DSC 的专门定点小数乘法指令 mult()，它可以利用 DSC 中的硬件乘法器完成定点小数的乘法运算。具体的程序参见如程序 5.1 所示。

```
void SetMotorVoltage(int nLeftVol, int nRightVol) {
    short nPeriod;
    nPeriod = (short)getReg(PWM_PWMCM);
    if(nLeftVol > 0) {
        setReg(PWM_PWMVAL0, 0);
        nLeftVol = mult(nLeftVol, nPeriod);
        setReg(PWM_PWMVAL1, nLeftVol);
    } else {
        nLeftVol = -nLeftVol;
        setReg(PWM_PWMVAL1, 0);
        nLeftVol = mult(nLeftVol, nPeriod);
        setReg(PWM_PWMVAL0, nLeftVol);
    }
    if(nRightVol > 0) {
        setReg(PWM_PWMVAL3, 0);
        nRightVol = mult(nRightVol, nPeriod);
        setReg(PWM_PWMVAL2, nRightVol);
    } else {
        setReg(PWM_PWMVAL2, 0);
        nRightVol = -nRightVol;
        nRightVol = mult(nRightVol, nPeriod);
        setReg(PWM_PWMVAL3, nRightVol);
    }
}
```

● mult: DSC□□□□□ □□□□ □□□□□□ nLeftVol * nPeriod/0x7fff

程序 5.1 电机 PWM 输出子程序

这个子程序 5.1 由下面的第二个程序调用。

第二个子程序：MotorSpeedOut()

它的输入参数为两个全局变量：g_nLeftMotorOut, g_nRightMotorOut。

分别表示左右两个电机输出电压。

取值范围： -0x7ff 至 0x7ff。

输出参数：调用 MOTOR_SET()实际上就是上面的子程序。

该程序主要功能为包括：

（1） 在输出数值上添加克服死区的数值，利用宏定义 MOTOR_OUT_DEAD_VAL（缺省为 1000）表示。这个数值需要通过实验确定。对于电机施加比较小的电压时，由于存在静摩擦力，车模实际上是不运动的。因此，在输出的时候，需要根据输出的数值，增加一个固定量，克服静摩擦力，使得车模运行比较平稳。死区补偿关系如图 5.3 所示。

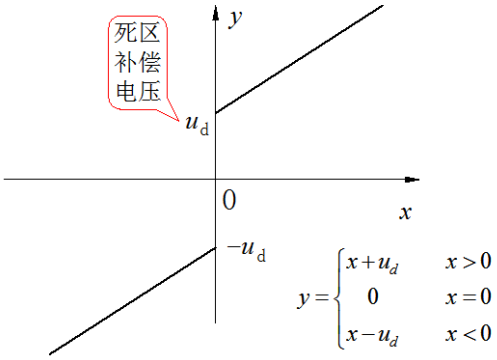


图 5.3 电机死区补偿

（2） 进行输出数值的饱和判断，确保在调用 MOTOR_SET 的时候，输入参量在 -0x8000 至 0x7fff 之间。

详细程序参见程序 5.2 代码。

```
void MotorSpeedOut(void) {
    int nLeftVal, nRightVal;

    nLeftVal = g_nLeftMotorOut;
    nRightVal = g_nRightMotorOut;
    if(nLeftVal > 0)        nLeftVal += MOTOR_OUT_DEAD_VAL;
    else if(nLeftVal < 0)   nLeftVal -= MOTOR_OUT_DEAD_VAL;
    if(nRightVal > 0)       nRightVal += MOTOR_OUT_DEAD_VAL;
    else if(nRightVal < 0)  nRightVal -= MOTOR_OUT_DEAD_VAL;

    if(nLeftVal > MOTOR_OUT_MAX)    nLeftVal = MOTOR_OUT_MAX;
    if(nLeftVal < MOTOR_OUT_MIN)    nLeftVal = MOTOR_OUT_MIN;
    if(nRightVal > MOTOR_OUT_MAX)   nRightVal = MOTOR_OUT_MAX;
    if(nRightVal < MOTOR_OUT_MIN)   nRightVal = MOTOR_OUT_MIN;

    nLeftVal = nLeftVal << 4;
    nRightVal = nRightVal << 4;

    MOTOR_SET(nLeftVal, nRightVal);
}
```

程序 5.2 电机速度输出子程序

这个子程序由车模直立控制子程序调用，每 5ms 调用一次。

2) 模拟电压采集及车模倾角计算子程序

本程序读取 DSC 的 AD 采用数值，然后计算车模的倾角。

对于读取的陀螺仪和加速度计的数值需要减去零偏值。这个数值需要通过实验确定。在车模保持直立静止时，读出两个通道的数值，便是相应的零偏值。这个数值会带

有一定的误差，往往会使得车模往一个方向加速行驶。这个误差可以通过后面的速度控制加以消除。

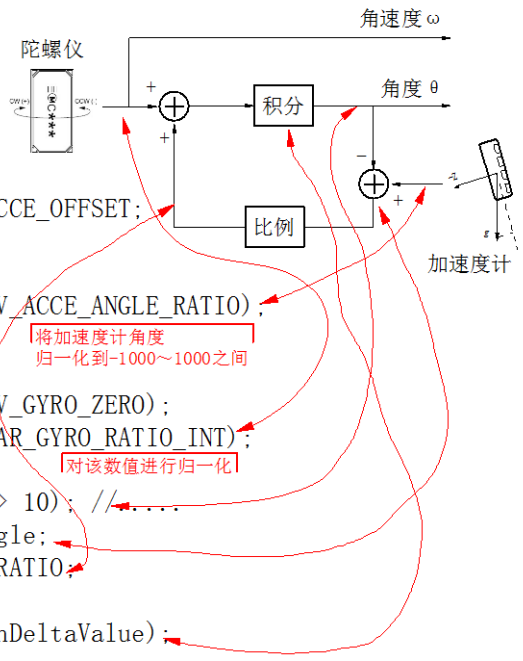
对于加速度计所得到的数值，通过一个比例系数(CV_ACCE_ANGLE_RATIO)将它归一化到-1000 至 1000 之间。同样，后面的陀螺仪的读出的数据也需要通过一个比例系数进行归一化。这个数值也是通过实验确定的。具体的方式为：对于陀螺仪的输出进行积分，将车模由垂直状态放置为水平状态，得到积分的数值。与 1000 相比较，便可以得到归一化的比例系数。

由于本函数调用的时间周期为 5ms。为了不损失积分的精度，使用一个长整型数字(g_lnCarAngleSigma)进行积分。对于数值通过右移 10 位（相当于除以 1024）得到最终的角度值。因此前面的 CAR_GYRO_RATIO_INT 也是需要考虑到这个比例的。

请注意前面 DSC 中的 AD 配置的时候，将所有 AD 通道的输入数据都归一化到 0 至 0x7ff0 之间。所以在下面程序中，所有的常量定义都是根据这个范围确定的。

详细程序请参见程序 5.3 代码：

```
void CarVoltageGet(void) {
    long lnDeltaValue;

    ADC_GetValue16(g_nCarVoltage);

读取加速度值减去零偏数值
    lnDeltaValue = (int)CV_ACCE_VAL;
    lnDeltaValue = lnDeltaValue - (int)CV_ACCE_OFFSET;
    g_nCarAcceVal = (int)lnDeltaValue;

    g_nCarAcceVal = mult_r(g_nCarAcceVal, CV_ACCE_ANGLE_RATIO);

读取陀螺仪数值值减去零偏数值
    g_nCarGyroVal = (int)CV_GYRO_VAL;
    g_nCarGyroVal = (int)(g_nCarGyroVal - CV_GYRO_ZERO);
    g_nCarGyroVal = mult_r(g_nCarGyroVal, CAR_GYRO_RATIO_INT);

    g_nCarAngle = (int)(g_lnCarAngleSigma >> 10); //....
    lnDeltaValue = g_nCarAcceVal - g_nCarAngle;
    lnDeltaValue = lnDeltaValue * CAR_ACCE_RATIO;

    g_lnCarAngleSigma += (g_nCarGyroVal + lnDeltaValue);
}
```



程序 5.3 车模倾角计算子程序

3) 车模直立控制子程序

车模直立控制是关键子程序。其中涉及到两个关键控制系数：

k_1:CAR_AA_P_INT：倾角比例，（0.75*0x7fff）

k_2:CAR_AA_D_INT：角速度比例，（0.125*0x7fff）

注意上面的乘以 0x7fff 是为了将小数化成定点小数表示形式。

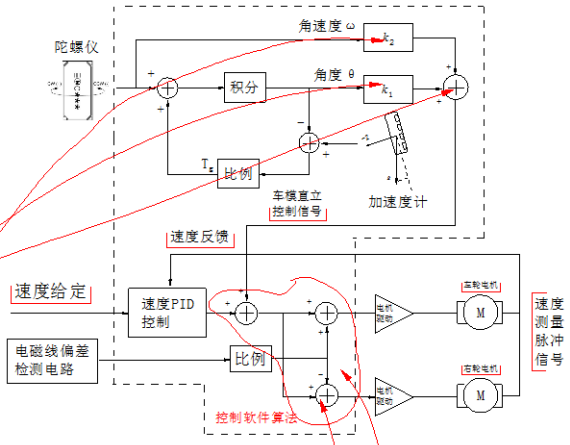
这两个系数是通过实验调试逐步确定的。具体确定的方式如下：

首先将 CAR_AA_D_INT 置为 0。逐步加大 CAR_AA_P_INT 数值，直到车模开始前后震荡。此时再逐步增大 CAR_AA_D_INT，消除震荡。然后逐步增加这两个数值，直到车模开始抖动，然后略减少这两个数值即可。

这个过程需要反复调整，直到车模可以比较稳定的达到直立平衡。

注意：由于现在还没有加入速度闭环，所以由于加速度传感器零偏的误差，会导致车模在直立的时候会往一个方向加速行驶。

如下是车模直立控制子程序代码：




```
void CarAngleAdjust(void) {
    int nLeft, nRight;
    int nSpeed;
    int nP, nD;

    nP = g_nCarAngle; (-1000~1000)
    nP = (int)mult_r(nP, CAR_AA_P_INT);
    nD = g_nCarGyroVal >> 5;
    nD = (int)mult_r(nD, CAR_AA_D_INT);

    nSpeed = nD + nP;

    if(nSpeed > MOTOR_SPEED_SET_MAX)        nSpeed = MOTOR_SPEED_SET_MAX;
    else if(nSpeed < MOTOR_SPEED_SET_MIN)   nSpeed = MOTOR_SPEED_SET_MIN;

    nLeft = nSpeed + g_nLeftMotorOutSpeed - g_nMotorLeftRightDiff;
    nRight = nSpeed + g_nRightMotorOutSpeed + g_nMotorLeftRightDiff;

输出数值调整以及饱和保护
    g_nLeftMotorOut = nLeft << 6;
    g_nRightMotorOut = nRight << 6;
    if(g_nLeftMotorOut > MOTOR_OUT_MAX) g_nLeftMotorOut = MOTOR_OUT_MAX;
    if(g_nLeftMotorOut < MOTOR_OUT_MIN) g_nLeftMotorOut = MOTOR_OUT_MIN;
    if(g_nRightMotorOut > MOTOR_OUT_MAX) g_nRightMotorOut = MOTOR_OUT_MAX;
    if(g_nRightMotorOut < MOTOR_OUT_MIN) g_nRightMotorOut = MOTOR_OUT_MIN;
    MotorSpeedOut(); PWM输出
}
```

程序 5.4 车模直立控制子程序

4) 车模速度控制子程序

车模速度控制采用了典型的 PI 控制算法。其中速度采用了左右电机速度的平均值进行计算。

对于 PI 调节算法中的参数整定可以参照一般 PI 参数整定的方法进行。

该函数的输入为 g_nLeftMotorSpeedCount，g_nRightMotorSpeedCount。

计算得到 g_nLeftMotorSpeedOut，g_nRightMotorSpeedOut。

由于速度控制子程序是每 100 毫秒才调用一次。所以程序并没有直接更新 g_nLeftMotorSpeedOut， g_nRightMotorSpeedOut 的数值，而是通过调用函数 CalculateMotorSpeedOut()更新上面的数值。函数 CalculateMotorSpeedOut()是每 5 毫秒被调用一次，所以程序是将速度控制的变化量平均到 20 次进行更新。这样可以降低速度控制对于车模直立控制的影响。

```
void MotorSpeedAdjustCal(void) {
    int nLeftSpeed, nRightSpeed;
    int nDeltaValue, nP, nI;
    int nSpeed;

    nLeftSpeed = (int)g_nLeftMotorSpeedCount;
    nRightSpeed = (int)g_nRightMotorSpeedCount;
    nSpeed = (nLeftSpeed + nRightSpeed) / 2;

    nDeltaValue = g_nMotorSpeedSet - nSpeed;
    nP = mult_r(nDeltaValue, MOTOR_SPEED_P_INT);
    nI = mult_r(nDeltaValue, MOTOR_SPEED_I_INT);

    g_nMotorOutSpeedOld = g_nMotorOutSpeedNew;

    g_nMotorOutSpeedKeep -= nI;
    g_nMotorOutSpeedNew = (g_nMotorOutSpeedKeep >> 3) - nP;
    if(g_nMotorOutSpeedKeep > MOTOR_OUT_MAX)
        g_nMotorOutSpeedKeep = MOTOR_OUT_MAX;
    if(g_nMotorOutSpeedKeep < MOTOR_OUT_MIN)
        g_nMotorOutSpeedKeep = MOTOR_OUT_MIN;
}
void CalculateMotorOutSpeed(void) {
    int nValue;
    nValue = g_nMotorOutSpeedNew - g_nMotorOutSpeedOld;
    nValue = nValue * (g_nCarMotionCount + 1) /
             (CAR_MOTION_PERIOD - 1) + g_nMotorOutSpeedOld;
    g_nLeftMotorOutSpeed = g_nRightMotorOutSpeed = nValue;
}
```

程序 5.5 车模速度控制子程序

5) 车模方向控制子程序

车模方向控制是通过比较车模两个电感检波电压进行的。这个调节函数调用周期为

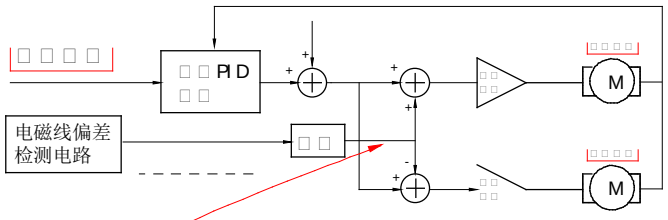
5 毫秒。

请注意，下面的算法中，采用了比例控制算法。

首先计算两个电感电压差除以两个电感电压之和所得到的比例。这个数值在-1.0 至 1.0 之间。然后再乘以一个比例系数 CMA_P_MAX。

CMA_P_MAX 可以通过实验逐步确定一个合适的取值。

详细程序请参见程序 5.6 的代码：




```
void CarMagneticAdjust(void) {
    int nP;
    long lnDelta;
    int nSigma;

    lnDelta = g_nCarMagneticRightAverage - g_nCarMagneticLeftAverage;
    nSigma = (g_nCarMagneticLeftAverage / 2) + (g_nCarMagneticRightAverage / 2);
    if(nSigma == 0) return;

    nP = (int)(lnDelta * CMA_P_MAX / nSigma) / 2;

    g_nMotorLeftRightDiff += nP;
}
```

程序 5.6 车模速度控制子程序

6) 中断服务程序

下面的 1 毫秒中断服务程序 5.7，请参见前面中断服务程序框架进行阅读。

```
void TI1_OnInterrupt(void){
    unsigned int nLeftMotorSpeed, nRightMotorSpeed;

    g_nCarSpeedCount ++;
    if(g_nCarSpeedCount >= CAR_SPEED_PERIOD) {
        g_nCarSpeedCount = 0;
  □□
        GetMotorSpeed(&nLeftMotorSpeed, &nRightMotorSpeed);
        ClearMotorSpeed();
        g_nLeftMotorSpeed = (int)nLeftMotorSpeed;
        g_nRightMotorSpeed = (int)nRightMotorSpeed;
        if(!MOTOR_LEFT_SPEED_POSITIVE) □
            g_nLeftMotorSpeed = -g_nLeftMotorSpeed;
        if(!MOTOR_RIGHT_SPEED_POSITIVE) □
            g_nRightMotorSpeed = -g_nRightMotorSpeed;
        g_lnCarLeftPosition □+= g_nLeftMotorSpeed;
        g_lnCarRightPosition □+= g_nRightMotorSpeed;
        g_nLeftMotorSpeedCount += g_nLeftMotorSpeed;
        g_nRightMotorSpeedCount += g_nRightMotorSpeed;
    } else if(g_nCarSpeedCount == 1) {
        ADC_Measure(0);
    } else if(g_nCarSpeedCount == 2) {
        CarVoltageGet();
        CarAngleAdjust();
    } else if(g_nCarSpeedCount == 3) {
        g_nCarMotionCount ++;
        if(g_nCarMotionCount >= CAR_MOTION_PERIOD) {
            g_nCarMotionCount = 0;
            MotorSpeedAdjust();
            g_nLeftMotorSpeedCount = 0;
            g_nRightMotorSpeedCount = 0;
            CalculateMotorLeftRightDiff();
            CalculateMotorOutSpeed();
    } else if(g_nCarSpeedCount == 4) {
        g_nCarMagneticLeftAverage = (int)CV_MAGNETLEFT_VAL;
        g_nCarMagneticRightAverage = (int)CV_MAGNETRIGHT_VAL;
        CarMagneticAdjust();
    }
}
```

程序 5.7 中断服务子程序

5.4 程序调试与参数整定

前面给出的算法程序存在很多参数，虽然从理论上可以对这些参数进行优化计算。但是由于受到车模模型精度的影响，计算所得到的参数也只能够作为参考值——调试的起始范围。实际优化参数需要通过一定的工程步骤最终确定，这个过程称为参数整定。

为了保证调试顺利，一般需要配合上位机串口监控程序，能够实时显示程序运行采集到的各种数据，通过曲线或者数字显示出来，帮助确定一些待定参数，判断程序 BUG，加快程序调试，确定控制参数的优化数值。俗话说，欲善其工，必先利其器。开发制作相应的辅助调试工作，不仅可以大大加快调试的进度，同时也会锻炼同学们解决工程问题的能力。

下面给出程序调试的一般步骤，作为参考。

（1）电机 PWM 输出调试：程序给出两个电机固定的 PWM 输出。测试电机转速以及转向是否符合设定的要求。在此过程中，测试车模电机死区补偿电压数值：MOTOR_OUT_DEAD_VAL。

（2）陀螺仪和加速度计零偏调试：获得这两个传感器零点的偏移量。保持车模直立静止。读取两个传感器 AD 转换值，记录下来，作为后面程序中的零偏常量。

（3）测试车模直立判断程序和跌倒判断程序。此时禁止车模直立控制环节，通过读取加速度计的数值，判断车模是否直立或者跌倒，通过外部的 LED 显示结果。一般情况下，车模直立要求，车模倾角在正负 10 度之内，保持 2 秒钟。

车模跌倒判断车模的倾角大于 45 度。

（4）车模直立控制调试：

只启动车模直立控制，调整两个控制参数：倾角比例和角速度比例，使得车模能够稳定的直立。此时由于没有速度控制，车模可能会朝一个方向加速行驶。

可以通过微调加速度计的零偏值，减小车模在直立的时候朝一个方向的加速度。尽量控制车模静止。

（5）车模速度控制调试：

启动车模的直立控制和速度控制，调节速度控制的 PI 参数，使得车模能够静止稳定在一个地点。然后再给定一个设定速度，车模可以稳定的前行或者后退。

（6）车模方向控制调试：

启动车模的直立控制、速度控制和方向控制。将车模放在电磁导引跑道上，车模可以稳定在跑道上运行。

通过上面几个步骤，逐步确定各个控制环节中的参数。但是是否能够在将来的比赛中胜出，还需要“是骡子是马，拉出来溜溜”，进行下面的现场测试。

5.5 现场运行测试

车模经过制作、编程、调试之后，需要在真实环境中进行运行，进一步发现存在的错误，优化各个控制参数，逐步提高车模的运行速度。通过不断的磨合、修改获得一组最优的控制策略。

在运行测试中，也可以对控制算法做进一步的改进。增加车模状态观测量，修正其中非线性环节，提高车模在高速运行时的稳定性和抗干扰能力。

在测试过程中，对于车模的机械结构、控制电路、信号检测等各个环节进行整体的优化。采用新颖的控制思想，突破现有观念的限制，从而在未来的竞争中获胜。

六、结束语

本文使用了尽量少的理论知识介绍了车模直立行走的参考控制方案。通过这个方案，同学们会觉得制作直立行走的车模参加竞赛不是困难的事情，而是相当容易和有趣的过程。

通过制作，同学们可以基本上实现完成比赛的车模。完成之后，同学们会逐步发现该方案的不足之处，进而萌发改进的想法，去查阅更多的相关文献，学习先进的现代控制理论知识，提高自己的理论水平和实践能力，最终产生对于自动控制专业的强烈兴趣，这正是同学们应该得到的收获。

在制作竞赛车模的方案中，应该尽量选择价格便宜，容易购买到的器件。为了避免参赛队伍在选择传感器方面使用昂贵的器件，竞赛规则中限定加速度传感器和陀螺仪使用的型号。考虑到将来传感器市场供货情况的变化，组委会将扩大传感器选择的范围。请参赛队员及时留意竞赛网站公布的信息。

由于今年车模直立行走竞赛是首次设立，同学们制作的电磁组直立行走的车模运行速度可能相对比较慢，故此在竞赛规则中限制了比赛赛道的长度和难度。同时，在比赛

过程中，不要求车模从静止两秒钟自动开始比赛，也取消了比赛结束时车模停止在起跑线后三米内赛道上的要求。竞赛组委会将会根据同学们制作车模的具体情况，对电磁组比赛赛道、比赛规则做进一步的修订，修订信息会及时发布在竞赛网站上。

附录：

1、 本文版权声明

《第七届全国大学生“飞思卡尔”杯智能汽车竞赛电磁组参考设计方案》文章版权属于智能汽车竞赛秘书处。

2、 参考文献

（1）《电磁组竞赛车模路径检测设计参考方案》竞赛秘书处 2010-1, 版本 1.1

（2）《第五届全国大学生智能汽车竞赛 20kHz 电源参考设计方案》

竞赛秘书处技术组，版本 1.1

3、直立车模参考设计方案视频

可以在竞赛网站下载到本文介绍的参考设计方案车模演示视频：

http://www.smartcar.au.tsinghua.edu.cn/web/index.jsp：“直立车模参考设计方案视频”

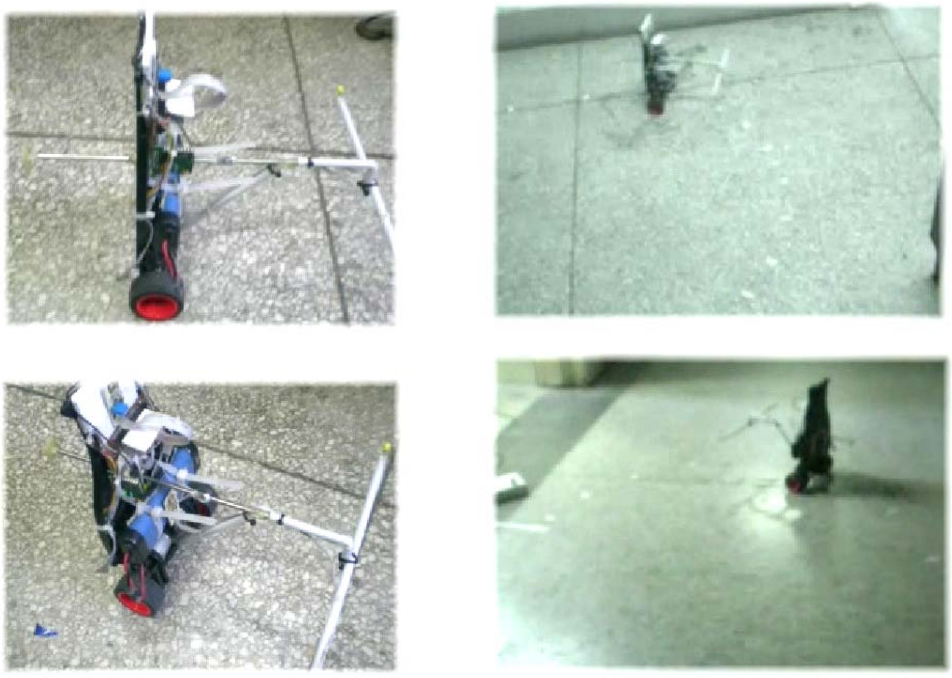


图 6.1 参考设计方案视频截图

4、**参考设计工程设计文件**

可以在竞赛网站下载到本文介绍参考设计方案软件工程文件。

“直立车模参考设计方案 DSC 软件工程文件.RAR”