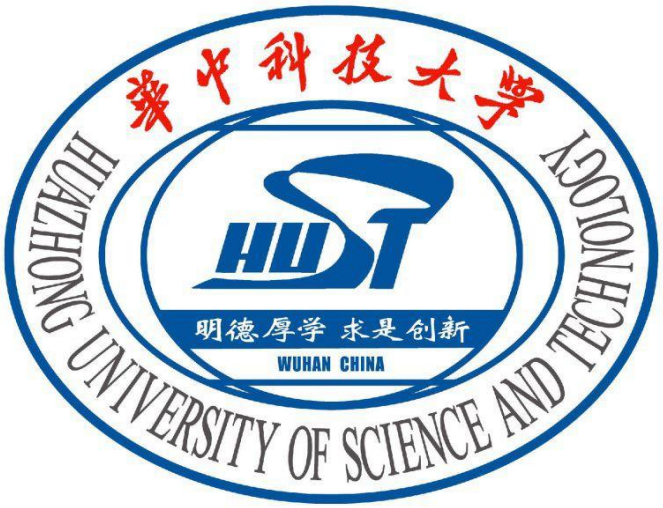
第十三届“恩智浦”杯全国大学生智能汽车竞赛

技 术 报 告



学　　校：　　华中科技大学

队伍名称：华中科技大学三轮电磁一队

参赛队员：　　宗　睿

　　黎　瑞

　　蒋世聪

带队教师：　　何顶新

　　彭　刚

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 13 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：宗睿 黎涛 蒋世彪

带队教师签名：

日　　期：2018年8月7日

摘 要

本文介绍了基于恩智浦32位微控制器的基于电磁场检测巡线智能车系统。针对比赛的具体情况，我们建立了赛车、赛道和自主控制系统的基本模型，给出了理论分析、仿真计算、在线调试的基本开发方法，在比较各种算法的性能特点后，我们确定最终方案，并完成了智能车的制作和调试。

本系统以微控制器KEA128为核心，软件平台为IAR EWARM开发环境，车模为组委会统一提供的F车模。

论文介绍了整个智能车系统的硬件和软件设计开发过程。使用KEA128作为主控芯片，用安装在车头的电感来检测赛道信息，用陀螺仪检测小车姿态，用光电编码器检测车模速度，用干簧管检测起跑线信息。整个系统的工作原理是由磁感应传感器采集赛道信息并经放大处理，与陀螺仪采集的车模姿态信息和光电编码器采集的车模速度信息一起送给单片机，通过程序设计控制优化算法，控制电机的转速以达到车模在赛道上的稳定高速行驶。

关键字：智能车　KEA128　循迹　控制

Abstract

This paper introduces the intelligent vehicle system based on EMF detection and patrol based on Enzhipu 32-bit microcontroller. In view of the specific situation of the competition, we set up the basic model of the car, track and autonomous control system, and give the basic development methods of theoretical analysis, simulation calculation and on-line debugging. After comparing the performance characteristics of various algorithms, we determine the final scheme and complete the manufacture and debugging of the intelligent vehicle.

The system is based on the microcontroller KEA128 as the core, the software platform is IAR EWARM development environment, and the vehicle model is F model provided by the organizing committee.

This paper introduces the design and development process of hardware and software of the whole intelligent vehicle system. KEA128 is used as the main control chip, the track information is detected by the inductor installed at the head of the car, the attitude of the car is detected by the gyroscope, the speed of the car model is detected by the photoelectric encoder and the information of the starting line is detected by the dry reed tube. The principle of the whole system is to collect the track information from the magnetic induction sensor and to be magnified. It is sent to the single chip with the attitude information of the vehicle model and the speed information collected by the photoelectric encoder. The optimization algorithm is controlled by the program design, and the speed of the motor is controlled to achieve the stability of the model on the track. Drive quickly.

Key word: KEA128 tracking control of intelligent vehicle

目录

第 1 章 引 言

“恩智浦”杯全国大学生智能车竞赛以“立足培养，重在参与，鼓励探索，追求卓越”为指导思想，系统涵盖了机械、电子、电气、传感、计算机、自动化控制等多方面知识，一定程度上反映了高校学生科研水平。本章节详细阐述了智能车系统的研究背景和本智能小车的设计总体概况。

1.1 概述

随着现代科技的飞速发展，人们对智能化的要求已越来越高，而智能化在汽车相关产业上的应用最典型的例子就是汽车电子行业，汽车的电子化程度则被看作是衡量现代汽车水平的重要标志。同时，汽车生产商推出越来越智能的汽车，来满足各种各样的市场需求。

第十三届“恩智浦”杯全国大学生智能汽车竞赛就是在这个背景下举行的。比赛要求在大赛组委会统一提供的竞赛车模，我们选择了飞思卡尔微控制器KEA128为核心控制单元的基础上，自主构思控制方案及系统设计，包括传感器信号采集处理、控制算法及执行、动力电机驱动等，最终实现能够自我识别路线，并且可以实时输出车体状态的智能车控制硬件系统。

本文先从总体上介绍了智能车的设计思想和方案论证，然后分别从机械、硬件、软件等方面的设计进行论述，重点介绍了芯片的选择和路径识别的方法，接着描述了智能车的制作及调试过程，其中包含本队在制作和调试过程中遇到的问题及其解决方法。

1.2 整车设计思路

本组电磁车使用KEA128芯片作为核心控制单元，使用多传感器大前瞻进行巡线控制，并且检测起跑线。使用了光电编码器对小车速度进行检测。并且增加了OLED屏和按键，以便更方便的查看小车数据，修改小车参数，以及修改整体策略。根据采集到的中线信息以及小车速度信息，对小车的转向以及速度进行控制，从而实现小车的循迹，整体结构框图如图所示：

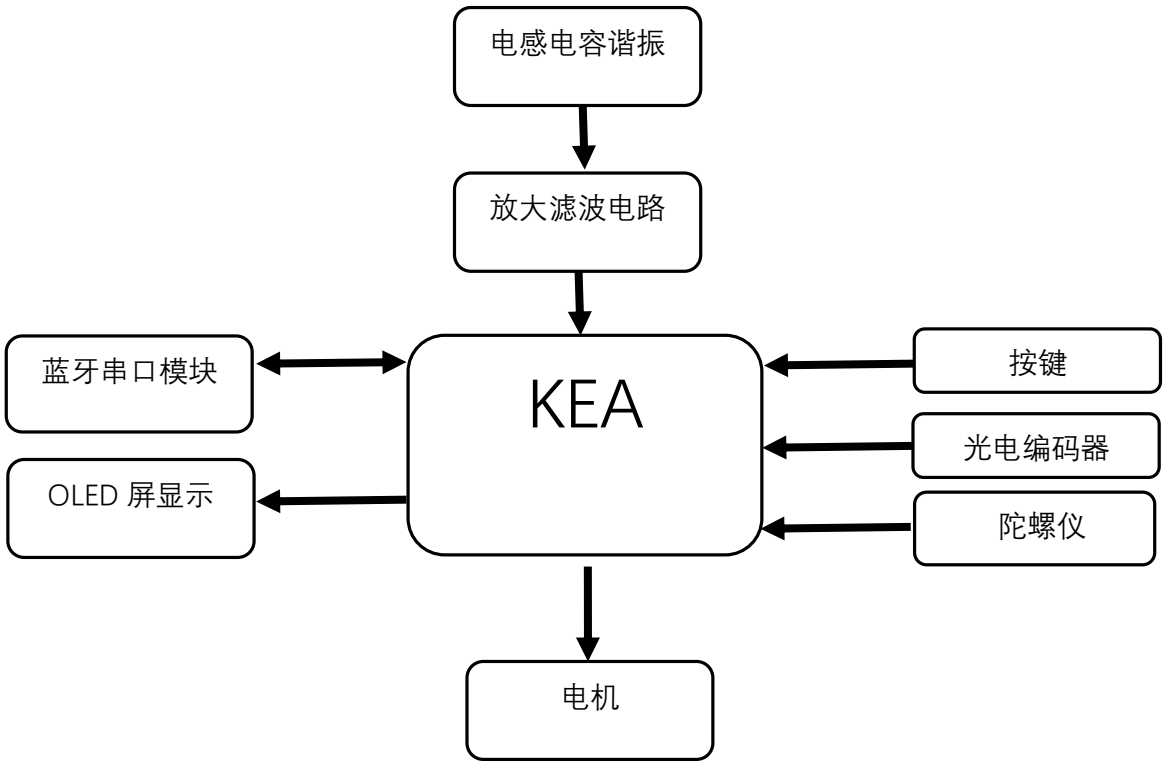


图 1.1　整体结构框图

1.3 文本撰写框架

本文简述小车的设计过程，主要分为：智能车车模机械设计、智能车系统硬件设计、智能车控制策略和软件设计、总结等四部分。并且附录中加入小车的算法程序。

第二章：智能车机械结构设计。主要介绍电感支架的设计安装、重心的调整等。

第三章：智能车硬件电路设计。主要介绍KEA128单片机系统、底层主电路、电感传感器、驱动电路等。

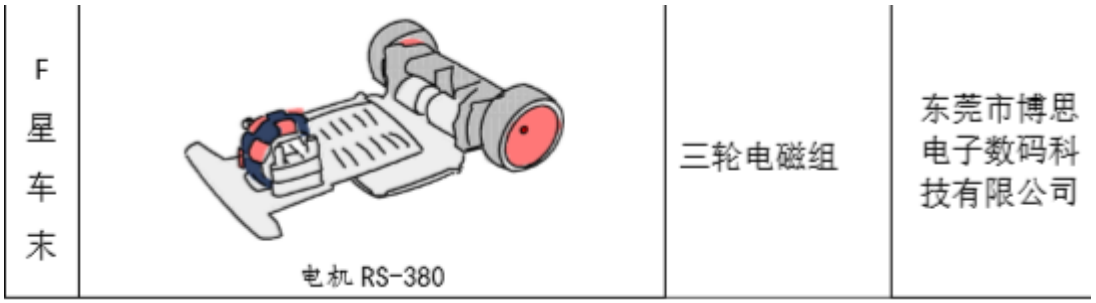
第四章：软件设计。主要介绍主程序流程、信号分析与识别、控制策略等。

第 2 章　机械设计

对智能车而言，一个好的机械结构至关重要，它是小车提速的基础。从控制的角度来说，这部分既是系统的执行机构又是被控对象。软件驾驭硬件，而硬件却依赖于机械结构。要想提高智能车的速度，软件要调的好，而机械结构上的一些优化，在很大程度上可以简化软件的编写。合理优良的机械结构能让智能车在直道和弯道上高速稳定的通过，而且转弯灵巧，快速。车模的机械部分是影响其行驶性能最直接的部分，同等控制能力的情况下，机械的性能会直接影响比赛的成绩，一个不良的机械系统会增加控制的难度，会为车模的速度提升带来障碍。所以车模的机械性能应该是优先考虑的问题，需要精心地调整，只有保证车模的机械参数调整在最佳状态，小车的性能才能得到完全发挥。 在满足组委会规定参数的情况下，我们对小车的各个部分进行了调校测试。本章主要介绍赛车车模的机械结构设计和调整优化方案，并按车模的机械安装优化顺序逐一对小车的机械结构进行详细的说明。

2.1 车模的整体框图

本届电磁竞速比赛采用了F型车模，让车模以两个后轮驱动进行行走,前面万向轮提供支撑。如图所示：



车模图

2.2 传感器的安装

车模传感器包括有：速度传感器（编码器），车模姿态传感器（陀螺仪），以及电磁检测感应线圈。小车的传感器相当于人的五官，它利用速度传感器来感知行进的速度，通过陀螺仪和加速度计获得车模倾角，检测坡道以及颠簸路面，而小车前方的电磁传感器就像人的两只眼睛，用来探知前方的路况，然后将信号传给单片机进行分析，判断下一步的行走方向。传感器与小车的其他部分，配合单片机共同完成对小车智能控制，实现小车的行走，自动循迹。

2.2.1 速度传感器的安装

速度传感主要由编码器构成，我们组车模用的是1024线编码器。通过编码器将旋转位移转换成一串数字脉冲信号，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反映电机的转速。去掉后轮，安装电机测速编码器。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。如图所示：

安装编码器后的车模

2.2.2 车模姿态传感器的安装

车模姿态传感器使用了陀螺仪。为了最大程度上减少车模运行前后振动对于测量的干扰，通常尽可能的将陀螺仪固定在整个车模最底部比较平稳的位置。

安装角度传感器电路板时应该尽量保证陀螺仪传感器水平安装。若是陀螺仪安装不能够保证水平，则车模在过弯道时速度会变快或者变慢。因为车模在过弯道时同时具有两种运动：平动和转动。其中转动会带动陀螺仪转动。如果陀螺仪安装不是绝对的水平，那么这个转动就会在陀螺仪的Z轴方向存在一个分量。根据陀螺仪倾斜的方向不同，这个分量有可能是正，有可能是负。从而会使得车模控制“仿佛感觉到在上坡或者是在下坡”，引起车模的速度变慢或者变快。所以我们将陀螺仪安装在车模的中轴线上。如图所示：

安装陀螺仪加速度器后的车模

2.2.3 电磁检测传感器

由于本届比赛限制了车模的宽度以及长度，要宽度小于25厘米，从轮胎中心到最前面传感器小于40厘米，为了能够更好地探知前方的路况，令单片机及时做出判断，防止小车冲出跑道，所以我们将小车的工字型的10mH电感尽可能地在规则允许范围内安装在车模运行前方较远的地方。但在实际的实验过程中，我们也发现电感支架过长的话会使小车的电感在过弯道时容易丢失信号，冲出跑道。这是因为小车的车体相对于车前方的电感有滞后性，导致小车在拐弯时车体还在跑道中间，电感却已探出跑道，检测不到正确的信号，导致小车做出错误的判断，使小车冲出跑道。所以最后我们经过反复的尝试，小车的支架长度在39cm左右效果最好。左右电感以车模中轴线为基准对称水平安放，电感横架平行于车模板。如果电感安放的高低、前后不同，或是偏离车模中轴线，都会影响车模方向控制，令单片机做出错误的判断，走错方向。一是因为车模方向控制算法是在电感水平安装的前提下，左右电感感应电动势通过差比和运算得到结果。在电感安装高度不同的情况下测到的感应电动势会相差很多，计算值与实际情况相差甚远，会使小车本应在中线加速直走，单片机却做出误判拐弯，小车会冲出跑道。二是因为电磁线圈如果未安装在同一条水平线上，车模在过十字交叉路口的时候，水平方向的电线中的磁场就会在线圈上产生额外的感应电动势，这些电动势会造成车模方向控制的不稳定。

为了减轻电感支架的重量我们使用了碳纤管支架，使小车跑起来更加的灵活。如图所示：

电感安装

2.2.4 永磁铁的安装

为了能够触发计时系统，需要在车模底盘安装一块永磁铁作为标签。永磁铁距离地面高度在 2cm 以内。计时磁标可以永久粘在车模的地盘上，也可以在比

赛前临时固定在车模的底地盘上。 具体磁标固定的位置并不要求精确，计时的过程是检测该磁标前后通过磁感应线圈的时间间隔。安装位置如图

2.2.5 干簧管的安装

干簧管的位置注意不要和永磁铁的安装位置太靠近，边移动边用万用表笔测试，直到蜂鸣器不响为止。

干簧管的安装

2.3 车模机械结构的优化

2.3.1 调整重心

通过重心的调整，可使模型车转弯时更加稳定、高速。其调整主要分为重心高度的调整以及重心在整车上局部分布的调整。考虑到车子的稳定性，在保证车模顺利通过坡道的前提下，我们尽可能的降低车子的重心。同时均匀车身重量，使重心在整车的中轴线上。由于靠前的重心会影响小车转弯的灵活性，过后的重心又会导致侧滑，经过多次试验，我们找到了一个合理的位置安排重心。

为了降低电池的安装高度，我们将电池用魔术贴固定在底盘上，这样也有利

于电池的更换；同时，直接将主板用热熔胶固定在底盘面板支架上。

主板与电池的固定

2.3.2 其他部分优化

关于三轮车的机械结构方面可以改进的地方还有，比如说车轮、传感器的保护等方面。由于小车的直立行驶及转向都是通过后轮实现的，因此当小车在转向时，模型车的轮胎与轮毂之间很容易发生相对位移，可能导致在加速时会损失部分驱动力，而且使小车的状态不稳。因此，我们在实际调试过程中对车轮进行了粘胎处理，可以有效地防止由于轮胎与轮毂错位而引起的驱动力损失的情况。另外，由于新购进的轮胎较硬，速度快时拐弯处明显打滑，因此需要对轮胎外胎进行适当的处理：软化或打磨。

2.4 齿轮啮合

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，听电机带动后轮空转的声音，若声音刺耳，则是齿轮撞齿的声音，说明齿轮间隙过大；声音沉闷，则表明电机负载过重，说明明齿轮间隙太小。

同时应注意齿轮互相平行，并尽量保证表面在同一平面。因为有电机和差速

盘，编码器和差速盘两处齿轮啮合，可以先调整电机与差速盘的间隙，调致无明显噪声后再调节编码器与差速盘间的啮合。调节至电机全速转动时齿轮间的噪声很小。

齿轮传动机构对车模的驱动能力有很大的影响。由于差速器是不能改变的， 所以我们只通过涂润滑油使传动更加顺畅。

2.5 轮胎固定

模型车在较高速的条件下（大于 2.3m/s），由于快速变化的加减速过程，使得模型车的轮胎与轮毂之间很容易发生相对位移，可能导致在加速时会损失部分驱动力。在实验中调试表明，赛车在高速下每跑完一圈，轮胎与轮毂之间通常会产生几个厘米的相对位移，严重影响了赛车的加速过程。为了解决这个问题，我们在实际调试过程中对车轮进行了粘胎处理，可以有效地防止由于轮胎与轮毂错位而引起的驱动力损失的情况。

第 3 章 电路部分

3.1 电路设计总述

电路部分是硬件平台的重要组成部分，稳定的电路是后续软件顺利运行的基础。我们在设计电路之时对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。我们的电路的设计思想是在保证正确检测信号的前提下，尽可能使用精简、高效的电路。

这次电路采用电磁信号处理、主控、驱动一体的电路板设计，避免了电路板的占地面积以及纷繁的连接线。

3.2 电源管理部分

为满足需要，本车模上存在 4 种供电电压：

1）智能车使用镍镉充电电池，充满时电压在 7.8～8.2V。可直接用于电机供电。

2）使用直流 3.3V 为主控板器件、运算放大器以及单片机供电。为了隔离电机的干扰，稳压芯片的输入端加并联电容。其中根据稳压芯片的功率不同，各部分所用的稳压芯片有所不同：

选用线性稳压芯片 TPS76833 为主控板的各个元器件供电；

选用另一块线性稳压芯片 TPS76833 单独为单片机供电以保证稳定性及功率要求；

选用线性稳压芯片 TPS76850 为信号处理模块供电。

3）电机驱动模块使用直流 12V，使用 5V 稳压芯片 LM2940 以及 5-12V 升压电源模块。加若干 LED 显示各类电源工作状况。

为避免电源反接带来的严重后果，采用 XT60 型头作为电源接头，并且由于整流二极管 1N4007 拥有良好的单向导电性及反向截止特性，故使用整流二极管在电源最开始的地方作为反接保护器件。

电源管理部分原理图如图 3.2 所示：

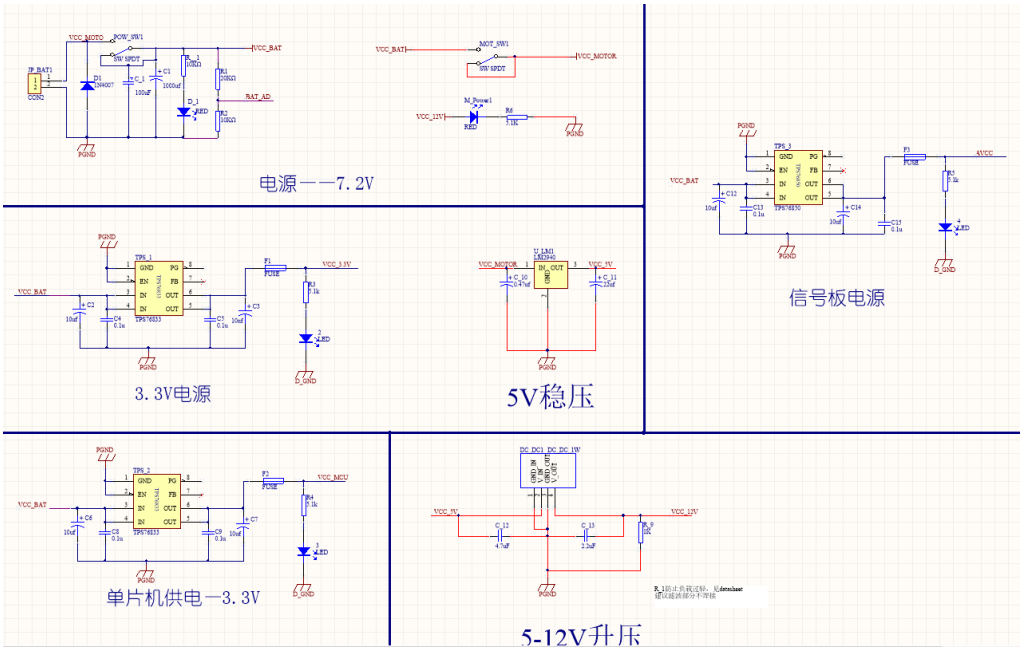


图 3.1 电源管理模块原理图

3.3 电机驱动模块

电机驱动模块可以说是智能车电路模块中一个最重要的部分，驱动的设计尤为重要。

常用的电机驱动有两种方式：1.采用集成电机驱动芯片；2.采用 N 沟道 MOSFET 和专用栅极驱动芯片设计。市面上常见的集成 H 桥式电机驱动芯片中，飞思卡尔公司的 33886 型芯片性能较为出色，该芯片具有完善的过流、欠压、过温保护等功能，内部 MOSFET 导通电阻为 120 毫欧，具有最大 5A 的连续工作电流。使用集成芯片的电路设计简单，可靠性高，但是性能受限。由于比赛电机内阻仅为几毫欧，而集成芯片内部的每个 MOSFET 导通电阻在 120 毫欧以上，大大增加了电枢回路总电阻，此时直流电动机转速降落较大，驱动电路效率较低，电机性能不能充分发挥。而分立的 N 沟道 MOSFET 具有极低的导通电阻，大大减小了电枢回路总电阻，

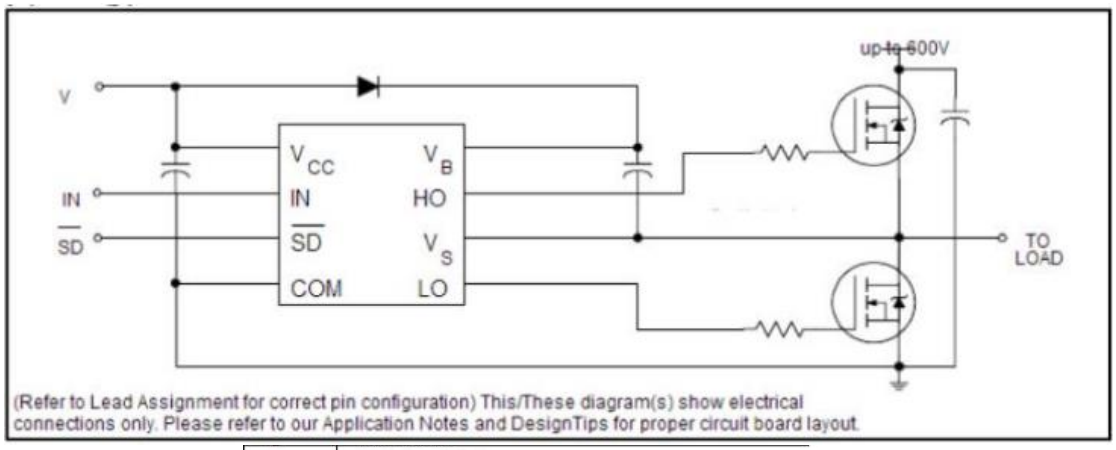
所以本系统电机驱动电路为一个由分立元件制作的直流电机可逆双极型桥式驱动器：

电机驱动信号经过芯片 74HC244。74HC244 是一款高速 CMOS 器件，74HC244 引脚兼容低功耗肖特基 TTL（LSTTL）系列。 74HC244 是八路正相缓冲器/线路驱动器，具有三态输出。该三态输出由输出使能端 1OE 和 2OE 控制。任意 nOE

上的高电平将使输出端呈现高阻态。芯片对驱动信号起数据缓冲作用，同时提高了信号的带负载能力。

驱动部分：使用 IR2104 驱动芯片和 N 沟道 MOSFET 组成完整的直流电机 H 桥式驱动电路，分立的 N 沟道 MOSFET 具有极低的导通电阻，大大减小了电枢回路总电阻。驱动芯片采用芯片 IR21014，IR2104 芯片可以驱动高端和低端两个 N 沟道 MOSFET，能提供较大的栅极驱动电流，并具有硬件死区、硬件防同臂导通等功能，提高 MOSFET 的开关速度，使 PWM 控制方式的调制频率可以得到提高，从而减少电枢电流脉动。

图 3.2 IR2104 引脚图及功能



HO	高侧栅极驱动输出
V_S	高侧浮动电源返回
V_{CC}	低压侧和逻辑固定电源
LO	低侧栅极驱动输出
COM	低端回报

图 3.3 IR2104 应用图

MOSFET 的选择主要需要考虑耐压、导通内阻和封装。电源是额定电压为 7.2V 的电池组，由于电机工作时可能处于再生发电状态，所以驱动部分的元件耐压值最好取两倍电源电压值以上，即耐压在 16V 以上。而导通内阻则越

	Parameter	Max.	Units
V_{DS}	Drain-to-Source Voltage	30	V
V_{GS}	Gate-to-Source Voltage	± 20	
I_D @ T_C = 25℃	Continuous Drain Current, V_{GS} @ 10V	161④	A
I_D @ T_C = 100℃	Continuous Drain Current, V_{GS} @ 10V	113④	
I_{DM}	Pulsed Drain Current ①	620	
P_D @T_C = 25℃	Maximum Power Dissipation ③	140	W
P_D @T_C = 100℃	Maximum Power Dissipation ③	71	
	Linear Derating Factor	0.95	W/℃
T_J T_{STG}	Operating Junction and Storage Temperature Range	-55 to + 175	℃
	Soldering Temperature, for 10 seconds	300 (1.6mm from case)	

小越好。封装越大功率越大，即同样导通电阻下通过电流更大，但封装越大栅极电荷越大，会影响导通速度。常用的 MOSFET 封装有 TO-220、TO-252、SO-8 等，TO-252 封装功率较大、而栅极电荷较小。于是我们最终选择了 IR 公司 TO-252 封装的 LR7843 型 N 沟道 MOSFET。

图 3.4 LR7843 的电气参数

为避免电机反向电动势的影响，在 MOSFET 两端接二极管，避免在电路开闭瞬间产生的反向电动势对电路造成影响甚至烧毁元件。

驱动模块电路如图 3.5 所示。

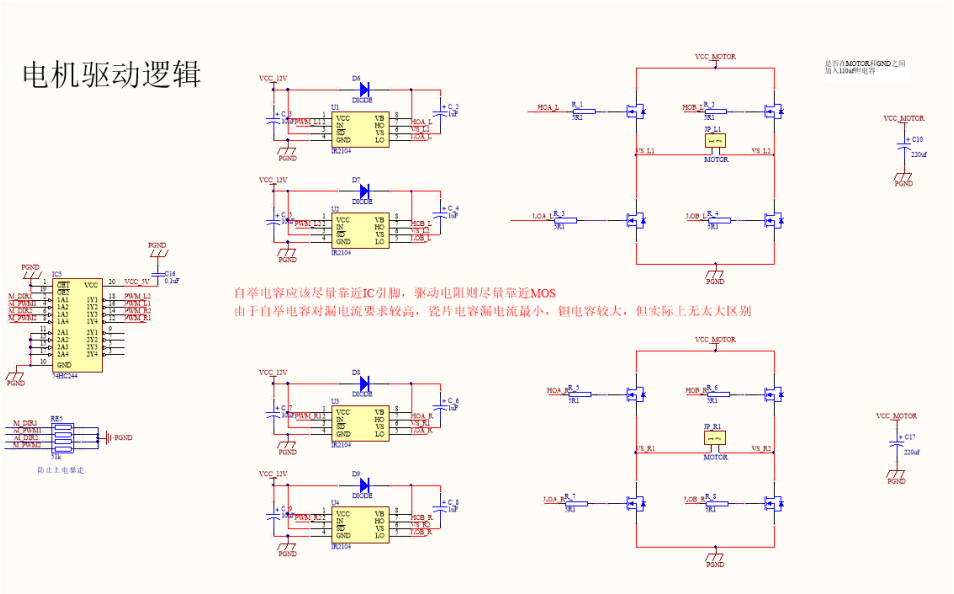


图 3.5 电机驱动模块电路

3.4 电磁信号采集与处理模块

3.4.1 电磁信号采集原理

传感器是电磁车的眼睛，所以传感器模块及放大电路对于小车能否快速准确的捕捉路况信息十分重要。根据麦克斯韦电磁场理论，变化的磁场可以激发涡旋电场，变化的电场可以激发涡旋磁场；电场和磁场不是彼此孤立的，它们相互联系、相互激发组成一个统一的电磁场。

智能车比赛选择 20kHz 的交变磁场作为路径导航信号，本系统根据 LC 谐振电路检测小信号的原理，选取工字电感和电容搭建 LC 谐振电路进行信号采集。

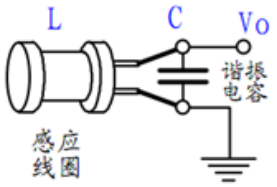


图 3.6 LC 谐振模型

电感和电容的选择满足 $f0=\frac{1}{2\pi\sqrt{LC}}=20kHZ$.

LC 电路输出电阻 R0 公式为

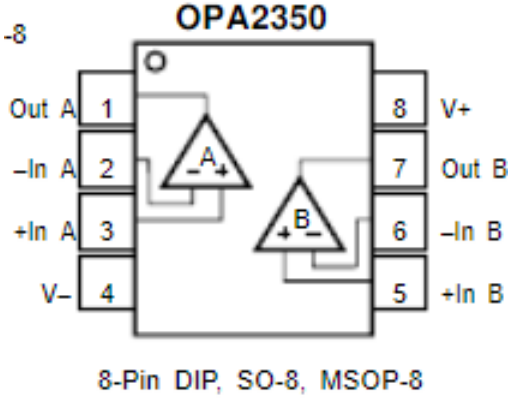
$$R_0=\sqrt{\frac{(\frac{L}{RC})^2+(\frac{1}{2\pi fC})^2}{1+Q^2(\frac{f}{f_0}-\frac{f_0}{f})^2}}$$

由公式可知，为了使检测到的信号的输出幅值变大，应考虑两点：第一，输出越大，R0 就要越大大，为了配合信号源的输出频率 20KHz，电感越大，电感的阻抗越高， 第二，谐振的频率要在 20KHz 附近，幅值衰减最小，电路的品质因数 Q 要大一 些，可以大幅减小其他频率的幅值而保持所选频率的幅值衰减不大，达到一个选频的目的。经过粗略计算后，本系统使用 10mH 和 6.8nF 的电感和电容，但这只是大致范围，在实际选用中应对电感和电容值进行精确测量，使得到的频率尽量靠近 20kHz，并做到对称。

3.4.2 电磁信号处理模块

1） 信号的放大

由于三极管放大存在温漂较大，且静电现象严重等问题，本系统采用集成运放芯片。信号处理模块中运放采用芯片 OPA2350，OPA*350 系列轨至轨 CMOS



运算放大器针对低电压单电源运行进行了优化，轨至轨输入和输出、低噪声（5nV/√Hz）和高速运行（38MHz，22V/μs）使得运算放大器非常适合驱动采样模数（A/D）转换器。使用芯片 REF3125 为数据转换器提供 2.5V 的基准电压。

图 3.7　OPA2350 引脚图

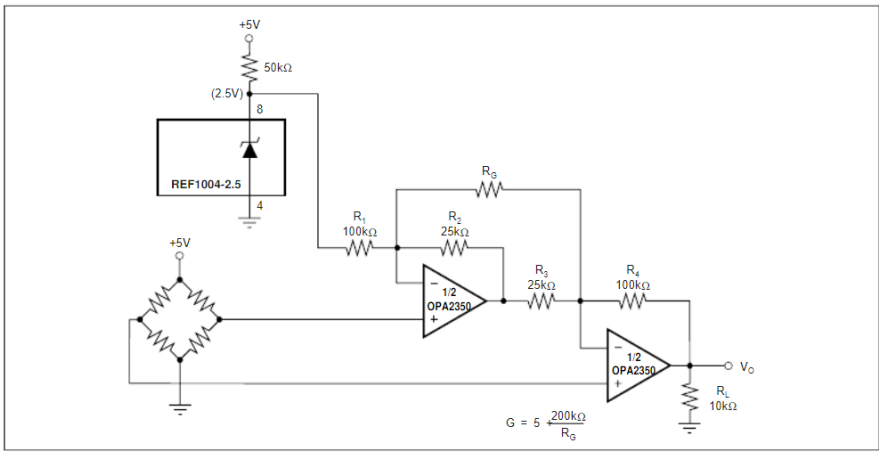


FIGURE 7. Two Op-Amp Instrumentation Amplifier With Improved High Frequency Common-Mode Rejection.

图 3.8　OPA2350 应用图

2）信号的检波

使用二极管检波电路将交变的电压信号检波形成直流信号，然后再通过单片机的 AD 采集获得正比于感应电压幅值的数值。使用两个二极管进行倍压检波，可以获得正比于交流电压信号峰峰值的直流信号。为了能够获得更大的动态范围，倍压检波电路中的二极管推荐使用开启电压较小的肖特基二极管。后接电压跟随器以稳定输出和带负载能力。

最终信号处理模块电路图如图 3.4.3。

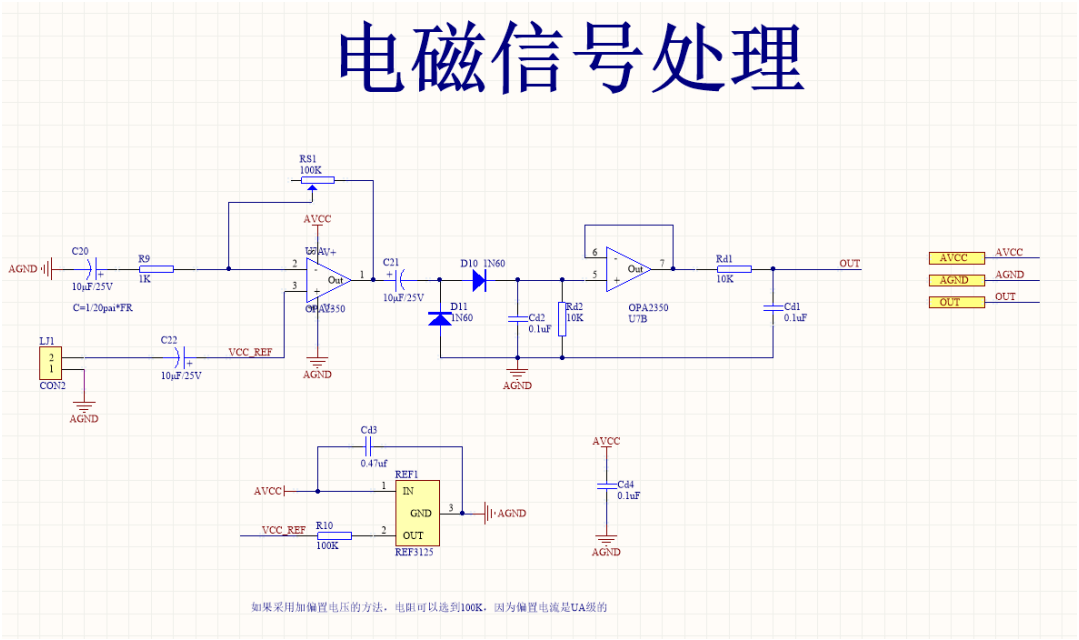


图 3.9 信号处理模块

3.5 起跑线检测模块

采用干簧管作为起跑线永磁铁检测元件。

检测模块如图 3.5.1，图 3.5.2 所示

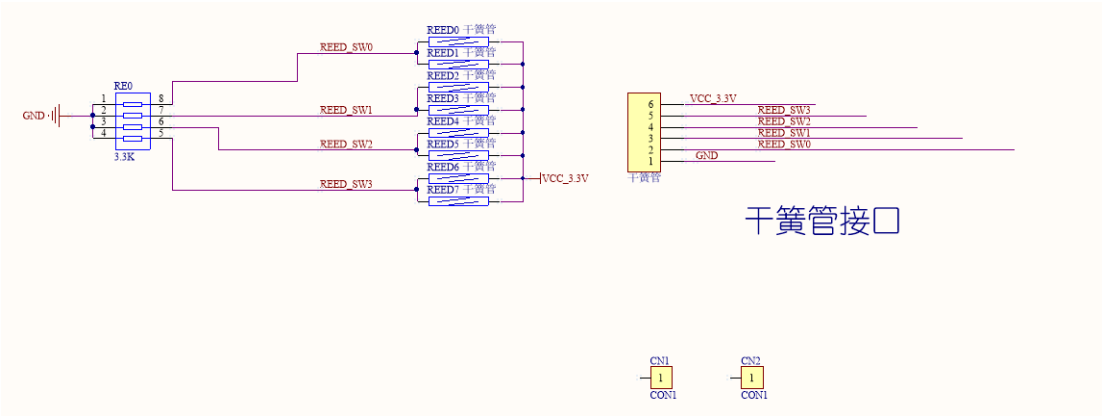


图 3.10 干簧管检测电路

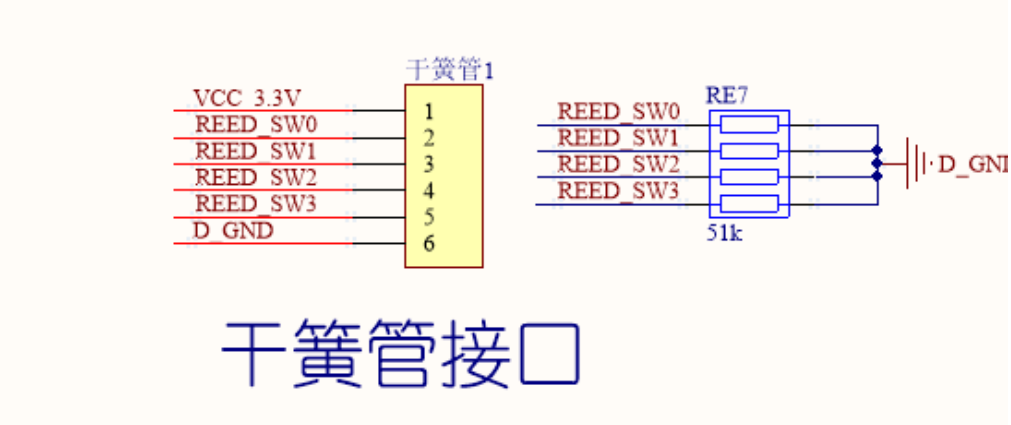


图 3.11 干簧管接口

3.6 其他部分

3.6.1 MCU 最小系统

MCU 最小系统部分主要包括： MCU 、复位、时钟、与外围器件接口。选用单片机型号：S9KEA128P80M48SF0 。

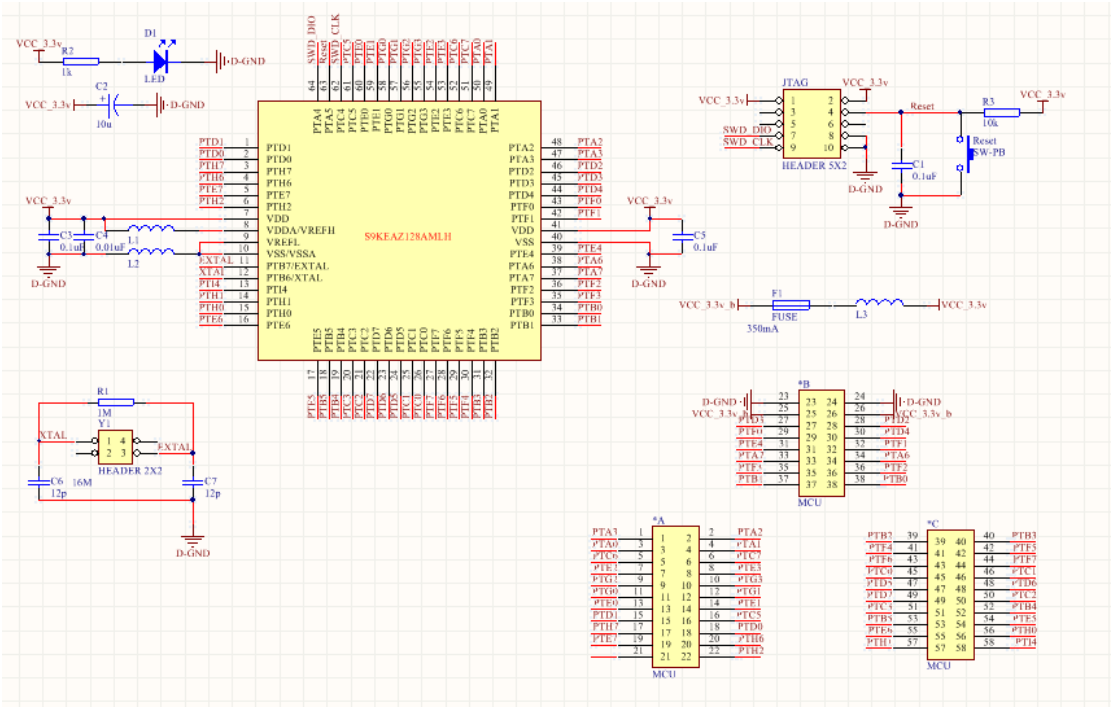


图 3.12 最小系统原理图

3.6.2 编码器模块

在电机上方架编码器，使之随电机转动而转动。左、右编码器得到的信号

均通过异或门和 D 触发器进行硬件解码，输入单片机。

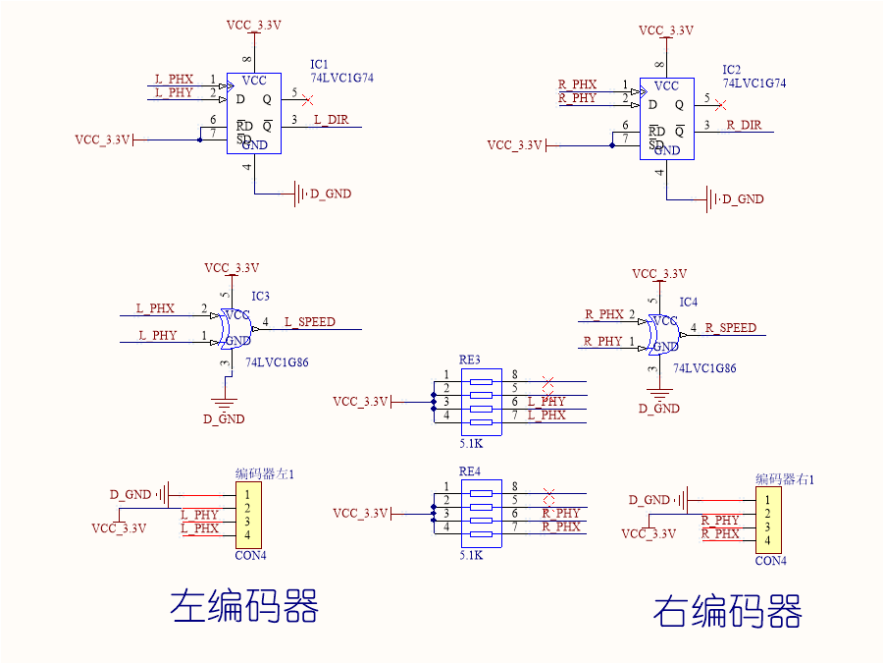


图 3.13 编码器正交解码

3.6.3 加速度计 陀螺仪

选用 MPU6050，相较于多组件方案，免除了组合陀螺仪与加速器时间轴之差的问题。

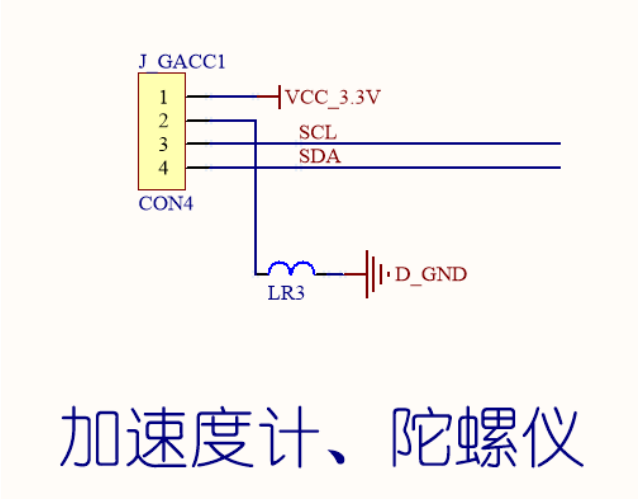
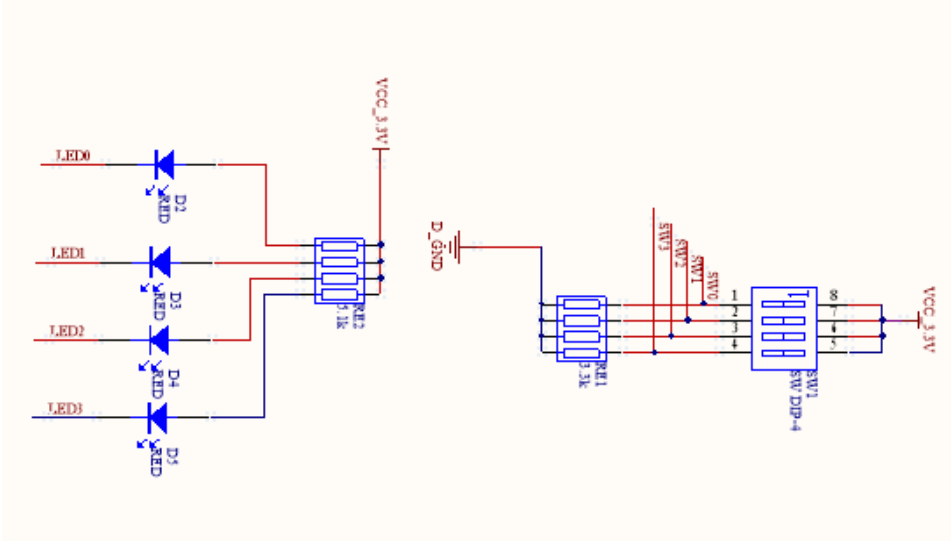


图 3.14 MPU6050 接口

3.6.4 人机交互模块

人机交互模块包括液晶显示，五向按键进行参数调试，拨码开关选择档位，LED 灯以及蜂鸣器显示工作状态。

图 3.15 LED 和拨码按键



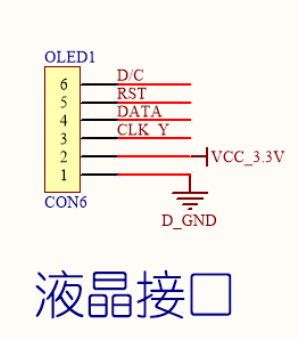


图 3.16 液晶接口

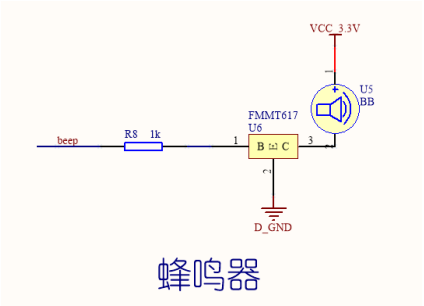


图 3.17 蜂鸣器

第 4 章 软件设计

4.1 程序整体框架

整个系统由恩智浦半导体公司的 32 位 KEA128 系列微控制器中的 S9KEA128P80M48SF0 ，对电磁传感器信号进行采集和处理，并对输出 的两路 PWM 分别控制，进行左轮和右轮电机的控制，决定小车的速度以及小车左右轮的差速导致的小车整体的角速度。我们使用编码器采集速度值，主要进行速度环的控制；使用陀螺仪 mpu6050 采集角速度，主要进行串级控制中角速度环（差速环）的控制。

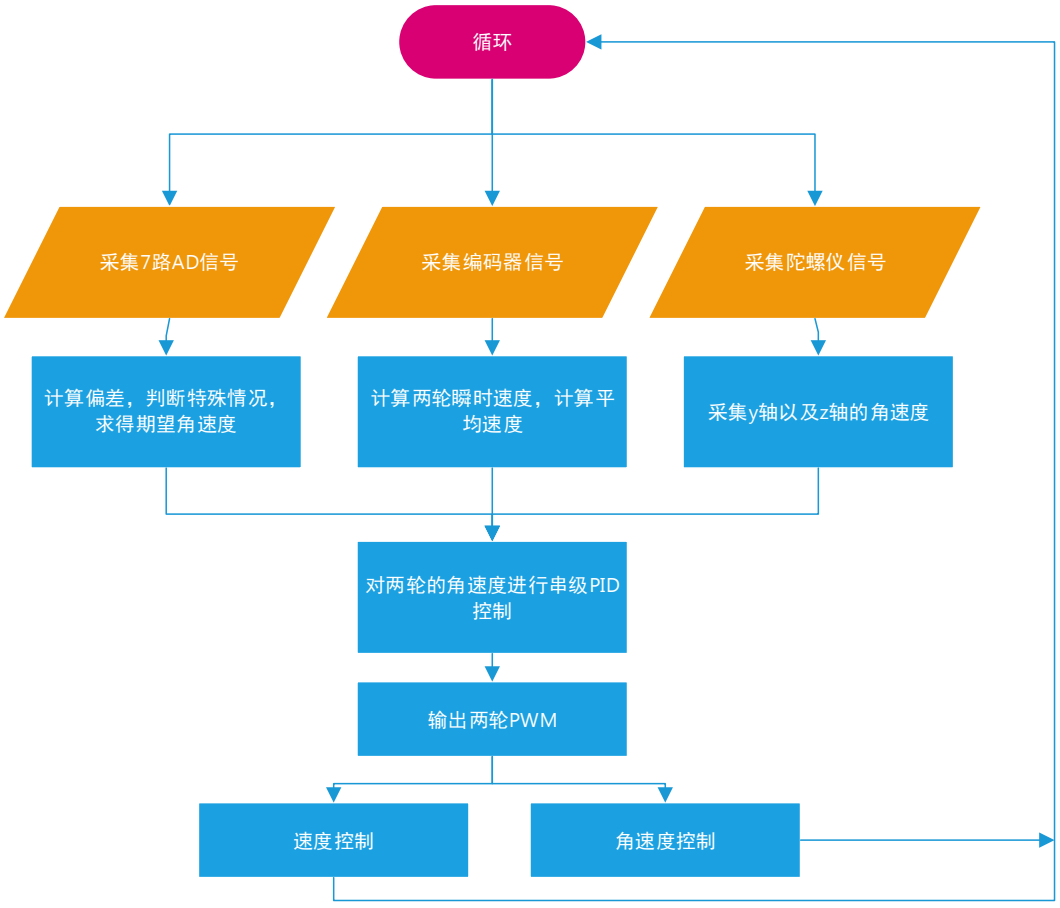


图 4-1 系统框图

4.2 系统分析

4.2.1 问题获取

对于竞速巡线智能车，我们参加的是电磁竞速组比赛，首先仔细阅读比赛的规则 以及相关规定，我们要做的是在此限制的基础上控制小车以最快的速度完成比 赛全程。系统的主要信息量为赛道中心通有 20KHz 电流方波产生交变磁场的漆包线产生的磁场，电流的理论值大小有 100mA，根据赛场环境的不同以及信号源的质量差异，电流的实际测量值可能会有 20%上下的差异或者波动，因此在进行程序编写的时候要注意到这一点。

4.2.2 分析

对于电磁三轮来讲，其识别道路情况全靠前面的电感传感器的值，由于磁感应强度是矢量，并且 AD 采样值为模拟量，因此通过将不同的电感用不同的姿势摆在不同的位置，就会在不同的路况下产生不一样的 AD 电感值的组合，我们可以通过这个来识别道路情况。由于三轮重心偏后，加上三轮前方全向轮中部有两个沟槽，三轮在赛道中线运行时可能会有轻微颠簸导致电感信号轻微晃动，因此电感的滤波需要进一步加强。

4.2.3 功能需求

为了更好的控制，三轮电磁智能车控制系统需要有：

（1） 高速、高性能控制芯片；

（2） 双轮速度反馈；

（3） 水平角速度反馈；

（4） 垂直角速度反馈；

（5） 需要实现基本的人机交互；

（6） 需要返回车在运行时的实时状态；

（7） 可以对返回的数据进行快速的数据分析。

4.2.4 性能需求

对于性能方面的需求，包括：

（1） 高速、高性能控制芯片；

（2） 稳定的电源供电模块；

（3） 强劲的电源放电能力；

（4） 速度响应快速，转向特性优越；

（5） 很好的轮胎摩擦力；

（6） 机械性能稳定、轻量化；

（7） 机械重心尽可能靠近两轮轴线；

（8） 较小的转动惯量；

（9） 较合适的控制周期；

（10） 程序高效稳定、发挥出机械与电路上的最高速度；

（11） 高效高速稳定的数据通讯实现。

4.3 信号获取与处理

4.3.1 AD 读取

我们采用如图分布的传感器方案，将七个传感器信号放大后输入 MCU 的七个 AD 通道，即可获得对应传感器的信号。软件上 4ms 周期性读取一次 AD 值， 用于后面的各种处理。

图 4-2 传感器布局

4.3.2 AD 滤波

在三轮车运行的过程中，由于温度变化、静电影响、特别是机械振动的关系，其获取的 AD 采样数组与实际期望采样到的数值之间会有轻微振动以及一定的静差。

除了在硬件上对传感器和周围电路进行一定调整之外，软件上也必须进行处理。我们的处理方法是中值滤波。

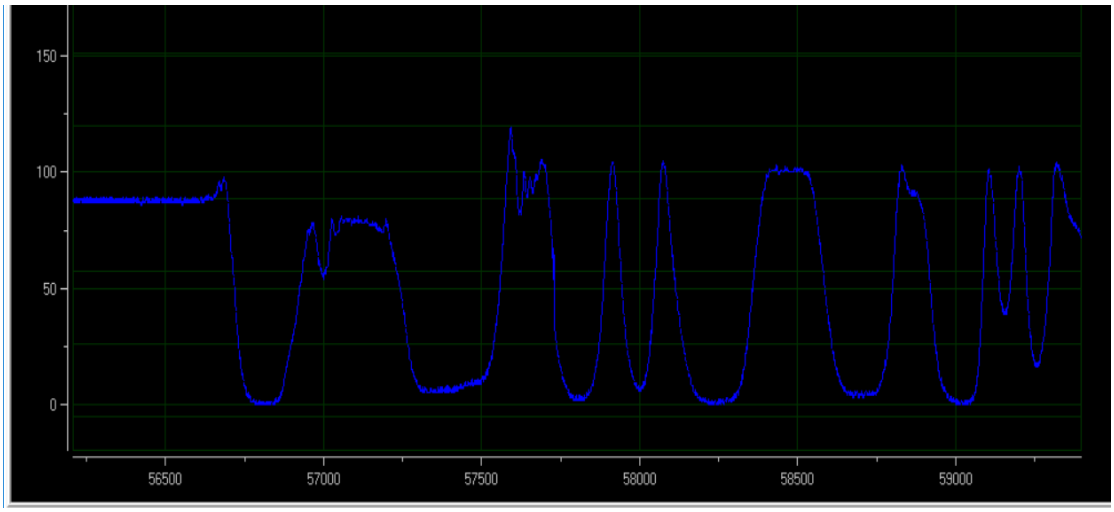


图 4-3　AD 原始信号

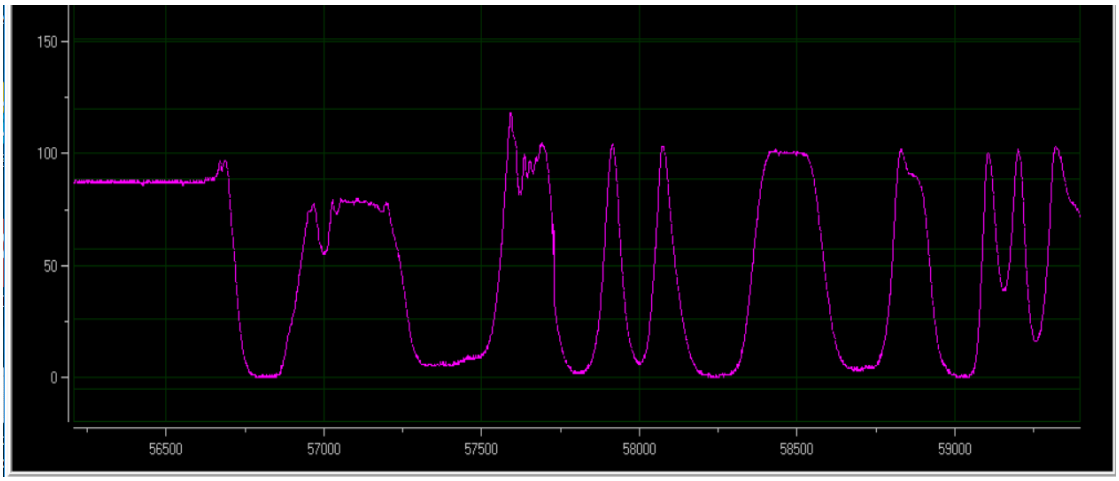


图 4-4 AD 滤波后信号

我们用的是匹配好的电感电容，因此滤波前与滤波后没什么明显的区别。之所以没有用到最小二乘法滤波是因为我们使用中值信号滤波后得出的信号与最小二乘法滤波得到的信号差别不超过 1%（用上图的 x0y 范围肉眼都难以分辨区别），并且该误差远远小于 PID 控制双轮转向所产生的误差，因此我们直接使用中值加权滤波的方法进行信号的处理。

动态标定是在赛道不同位置进行标定，把原始信号处理成一套我们习惯的信 号，这样就能适用于一套参数。但是由于我们使用的是差比和的偏差算取方法，因此当 ad 值等比例放大或者缩小时，理论上偏差是不变的（当然特殊元素判断除外），经过我们实际测量，在不改变 7 路电感放大倍数的情况下，程序能够承受 80%到 125%信号变动范围的大小变化，因此我们在实验了几次之后，直接去除了信号的动态标定，如果比赛场地的信号变动实在太大，我们将通过现场手调放大倍数的方式来调整，之所以不使用一个系数 K 来决定，是因为我们发现当信号源缩小时，不同位置、不同方向的电感并不是等比例缩小的，甚至也不是近似等比例，因此我们放弃了动态标定，保持程序尽可能简洁。

4.3.3 陀螺仪滤波

我们使用的 mpu6050 测量 z 轴（水平）以及 y 轴（垂直）角速度，但是由于其测量误差较大，因此我们直接采取加权递推滤波的方法，越靠近当前时态权重就越大，由此进行滤波。

以 y 轴（垂直）角速度为例：

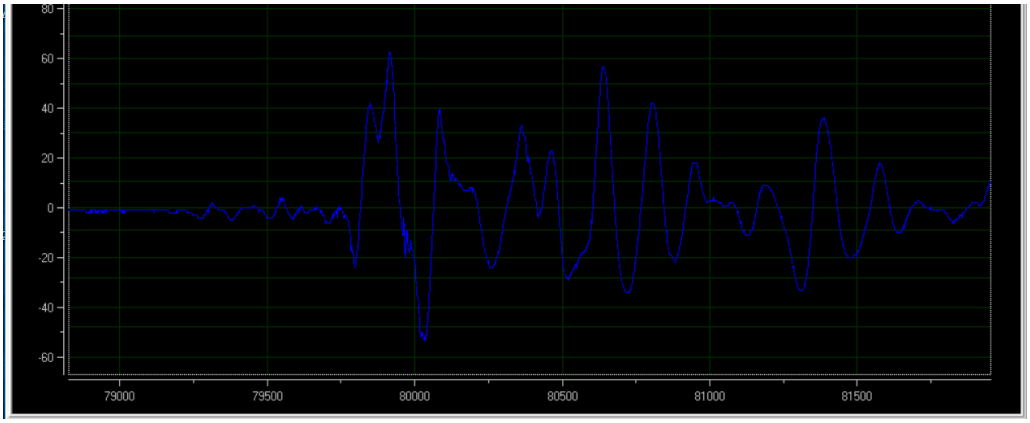


图 4-5 陀螺仪原始信号

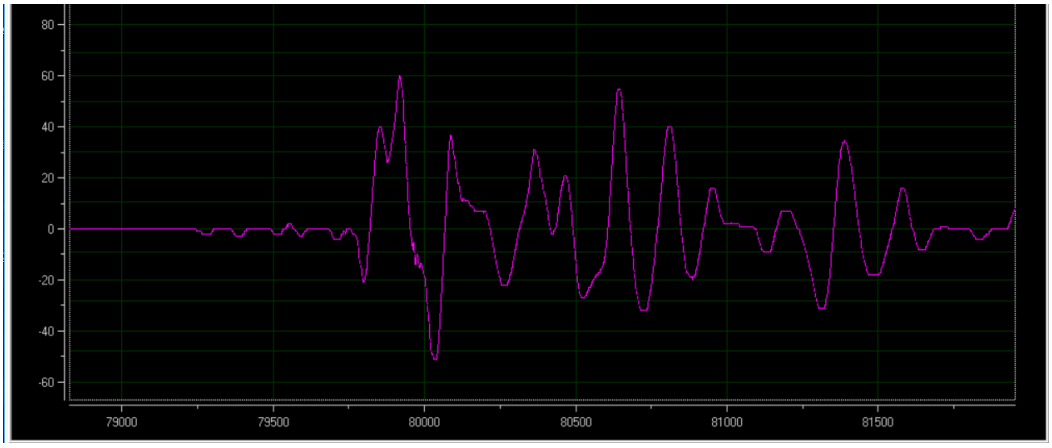


图 4-5 陀螺仪滤波后信号

可以很明显的看出，陀螺仪的滤波处理使得陀螺仪的信号毛刺减少，并使得它的静差有所下降，其效果以及必要性要比 AD 传感器滤波的效果及必要性要高出很多。

4.4 串级处理

4.4.1 转向环内环处理

4.4.1.1 偏差获取

由于我们使用的传感器方案用了两个 x 方向电感、两个八字形方向、两个 Y 方向电感，我们对这几个偏差进行加权平均，来得到当前的偏差，类似于

```
offset1 =
(float)(kt1*(t1-t2)+kx1*(float)(x1-x2)+ky1*(float)(y1-y2))
/(float)(kx2*(float)(x1+x2)+kt2*(float)(t1+t2)+(ky2>-0.0001?ky2:-ky2)*(y1+y2));
```

通过不断的调试 kx1、kx2、kt1、kt2、ky1、ky2 来修正偏差，以期望获取匹配于赛道期望角速度的数值。

然而单纯的使用差比和可能会使得在角度小的时候偏差变化率反而有点大，角度大的时候偏差变化率反而有点小的情况，为了缓解这种情况，我们引入了第二个偏差：

```
offset2 =
((float)kt3*(t1+t2)-ktx*kx3*(float)(x1+x2))
/(float)(kx4*(float)(x1+x2)+kt4*(float)(t1+t2)+ky3*(float)(y1+y2));
```

其中 ktx 就是我们 8 字方向电感对于横向电感在 offset1 为 0 的时候的比

值，这样可以保证在 offset1 不为零时，offset2 随着 offset 绝对值的增大而非线性增大（恒为正值），当偏差大于零就加上 offset2，当偏差小于零就减去 offset2，这样就可以解决我们上述所讲的偏差变化率不能达到我们理想期望趋势的问题。

4.4.1.2 偏差处理

众所周知电磁车的传感器一般比较长，过急弯时传感器不可避免地要伸 出赛道，但是由于今年限定了 AD 传感器的长度，因此电磁车只要在赛道上正常的运行，就不会出现丢信号这种情况，在我们调试阶段的时候可能会加上单边丢信号以及双边丢信号的处理，但在正式比赛时，这些处理都是要屏蔽掉的，因为正常跑完肯定不会出现这种情况，一旦出现这种情况说明小车的路径已经非常差劲了，肯定不会有什么很好的成绩。因此，我们在正式比赛的时候并没有加上什么丢线处理。只需要加上一个偏差限幅，以及限制变化率即可。、

```
current_offset=(int)-temp_offset;
if(current_offset>offset_history[0]+10)
        current_offset=offset_history[0]+10;
    else if(current_offset<offset_history[0]-10)
        current_offset=offset_history[0]-10;
```

4.4.1.3 期望的角速度获取

期望的角速度是由获取的偏差值决定的，然而众多周知在舵机的 PID 控制中我们一般并不使用 I 以防止控制太过滞后，在我们的串级控制中如果内外环控制都使用 PD 控制的话，就容易产生系统信号高频震荡，导致两轮的差速虽然反应是变得更加灵敏了但是它们的静态误差增大了；并且有电机迅速过热的危险存在，因此我们决定，将偏差乘上一个系数作为此次控制周期的理想角速度，即我们在内环里面直接使用单 P 控制的方式来求得期望角速度。

4.4.2 转向环外环处理

4.4.2.1 偏差获取

我们是直接通过由内环求得的期望角速度直接减去处理好的 mpu6050 测量到的水平角速度来取得外环的偏差，而偏差的差分我们则是使用了 6 个周期中近三个周期的偏差减去 6 个周期中前三个周期的偏差除以 3，这样可以有效的减少因偏差振动过大而引起的轮胎高频振动现象。

4.4.2.2 差速 PWM 获取

由于机械转动惯量以及电机因素，即使是限制为 200%的 PWM 的差速也在很多时候不能满足小车的转向需求（2.5m/s 以上时），因此我们的差速环并不进行任何限制（在数据调整阶段直接将限幅变成 200%的满 PWM 值），因此我们并没有外环的偏差处理。

差速的 PWM 是由外环的偏差由 PD 控制求得的，因此我们直接使其为：

```
D_pwm =
(int32)((gyro_error*kp/100.0+gyro_error_delt*kd/10.0));
```

即可。

考虑到单一电机过快变化较大 PWM 电机容易发热烧坏的问题，我们对每一个电机的 PWM 变化率进行了限幅，即（以左轮为例）：

```
if(speed_pwm+D_pwm>last_PWM_L+s_parameter.max_lPWM)
                L_PWM=last_PWM_L+s_parameter.max_lPWM;
else if(speed_pwm+D_pwm<last_PWM_L-s_parameter.max_lPWM)
                L_PWM=last_PWM_L-s_parameter.max_lPWM;
else
                L_PWM=speed_pwm+D_pwm;
```

4.4.3 转向环其他特殊处理

考虑到内环偏差可能并不能适应我们实际需求，参照过程控制系统中的理

想系统变量图，我们曾经使用过模糊 PD 的方式来处理所需求的偏差，后来发现由于三轮车本身的机械特性，其滞后实在太严重，导致模糊 PD 处理起来相对麻烦，并且没有很好的效果，这让我们意识到其与平常的四轮车不能等闲视之，因此我们直接使用了相对较为直观的分段线性连续 PD 法：其实就是现将偏差与车的偏离角处理成线性变化，再在这种线性函数的基础上进行各种各样的函数拟合。后来我们发现，当偏差较大时，其最佳拟合方案就是线性，因此我们最终选择的分段线性 PD 只是将偏差绝对值较小的时候的 PD 稍微缩小了一点，并没有太大的改进。

当我们遇到需要停车的时候，由于不能直接两个车轮差速给 0（防止其停下车之后冲出赛道），并且也不能原原本本的直接将速度环的 PWM 分量给零（万一速度很低的时候再遇到弯道可能会转向过大），因此我们采用的方式就是使速度环变成 0 的情况下，差速环的大小减少一半。这样可以让其较稳定的停车。

4.5 特殊情况处理

这里指的特殊情况主要有：坡道和颠簸、起跑线、十字、环岛。其中坡道以及颠簸是通过陀螺仪识别的，起跑线是通过干簧管识别的，十字、环岛则是通过 AD 采集信号识别的。

4.5.1 坡道、颠簸的识别与处理

颠簸和坡道之所以要处理很大程度上是来自于两个原因，第一：怕自己的车子被颠歪以至于冲出赛道；第二：怕自己的车子浮空太久或者卡到 KT 板严重影响自己的速度并损耗机械。因此我们先说一下处理方案，针对第一条，我们只需要识别到之后将自己的 PD 参数稍微给小一些就能取得不错的效果；针对第二条，我们的解决方案不在于降速，而在于减小速度环中积分环的限幅，这样车子不会出现“猛冲”的情况，就简单的解决了问题。

至于识别，颠簸的识别我们是看垂直传感器的值是否超出规定阈值，一旦超出，之后的 50 个周期小车都会被默认在颠簸或坡道的状态，如果再次超出阈值，50 个周期会被刷新，即从第二次超出阈值的周期再向后数 50 个周期当做

颠簸状态，坡道的识别类似，不过这样只能检测到上坡和下坡的一段时间，但实际上我们只是需要对这两段时间进行处理，因此识别上下坡道的那一段时间已经足够了。

4.5.2 十字的识别与处理

十字要识别主要是因为我们的差比和求得的偏差值如果不在十字特殊处理的话就会偏差突变两次，造成车体抖动，在十字弯处可能并没有太多问题，但在直道贯穿的十字处，存在有两个风险：第一个就是小车可能因为轻微的抖动而突然打一个错误的直角弯；第二个就是小车每次在路过十字时都会轻微抖动，造成即使是直道也根本无法加速的情况。十字处理比较简单，因为实际上十字那里有不对称变化的只有 8 字电感，可能一个为接近极大值一个接近为 0，因此处理的时候只需要将计算的偏差中有关于 8 字电感的所有系数都设置为 0 即可。

识别的时候我们是看 AD 的采样值，如果在两侧 8 字电感的差值大于一定值的情况下，两个竖直电感的值差值小于一定值，并且竖直电感的和大于一定值，就能够判定其在十字中。当然由于有些时候在大弯处可能会误判等问题，我们还对每一个电感值增添了一个判断范围。不在范围内即肯定不是十字。

4.5.3 起跑线的识别与处理

由于我们在实际测试中从来没有遇到过干簧管误触发的问题，倒是遇到过很多次“不触发”的问题，因此我们直接将干簧管的防误触发滤波处理去除掉，只要检测到一次“有磁铁”，就让车运行半米后停止。

当然我们也有后备方案，一旦小车真的干簧管误触发导致了运行一半停下来的问题，我们就启用定距停车方案，在一定程度上能够使小车勉强停下来。

4.5.4 环岛的识别与处理

环岛的识别与处理是这些特殊元素识别中最难精准识别，也是最难精准处理的一个特殊元素，与其他元素不同，环岛要分成好几段分别识别和处理的方式，才能够使小车过的比较圆润。

第一阶段：入环前。由于小车在入环前不进行任何处理的话小车就会向环

的方向抖动，这对于一个车的流畅入环是很不利的，因此需要进行处理，处理方法为将这一特殊段的 PD 参数稍微给小一些。识别方法为中间横电感大于 1.25 倍的平时值，但小于入环判断阈值。

第二阶段：入环。通过小车中间横电感达到两倍的平常值，并且两边横电感的和大于 1.5 倍的平常的和的值（防止坡道上坡前等误判），并辅以一系列的 flag 判断语句，即可判断为入环阶段。处理办法即为将偏差计算中和竖电感有关的系数全部翻 3 到 4 倍，即可使得小车比较切内，且不会提前切内的入环。

第三阶段：环内。环内的信号事实上是和正常道路上的信号是差不多的，因此直接按照在正常赛道上的处理方式即可。判断第三阶段我们使用的是，当第二阶段历经 1.2m 以上路程时直接判断进入第三阶段，或者直接省略第三阶段，将第二阶段的处理方法用在环内一样可行。

第四阶段：出环。当入环后一定距离时，中间横电感又到达了一定值，比如平时 1.4 倍的值，那么就可以判断其为出环阶段，此时的出环阶段我们使用单边处理，比如如果环在左边我们出环时就按右边信号单边处理，如何判断左右是靠第二阶段不断的累加 y1-y2 的值，第二阶段结束时，通过累加值的正负就能很准确的得到环的方向。

第五阶段：出环之后。和第四阶段一样，只不过第五阶段是第四阶段之后各个信号初次回复正常后的一个过渡阶段，如果不进行处理第五阶段还是会向环的方向偏离，这就会造成车身抖动，因此我们还是使用第四阶段的处理方式，但是由于我们在第四阶段与第五阶段的信号标定值并不一样，因此我们只有在第五阶段初期加上 PD 弱化以使得其不会抖动的同时，能正常沿着路径行驶。

4.6 速度控制

在四轮时我们会总体上用偏差决定速度的算法，偏差越大速度越慢。而我们发现对于三轮而言，其在弯道上本身就能承受 3m/s 左右的期望速度（然而通过蓝牙我们知道实际上弯道根本到不了这个速度），单纯的直道能够承受

3.5m/s 以上的速度，而当使用直道 3.5m/s 的速度时，直入弯道的那一瞬间电机的减速环严重影响到了角速度环，因此导致转向不流畅、分段甚至停顿，我们测试过，在弯道转向不流畅时，耽误的时间至少为 0.4s 左右，即使是一个 7m 长的直道，3.5m/s 的直线速度比之 3m/s 的直线速度也不会节约 0.4s，更不用说我们比赛时直道一般有 5m 就算很长了，于是我们发现使用巨大加减速的加速阶段省下的时间还不如弯道耽误的时间多，因此我们决定选用全程匀速或者稍微加减一点速度的准则，即不能影响转向，这就使得我们不能讲速度分成很多段，因为持续调整速度环会使得转向频繁受影响得不偿失，因此我们的方案就是只分直道和弯道速度两个速度，并且直道速度要满足一定时间的小偏差的条件才能达到。

速度环的控制周期不能和角度环重合，一定要比角度环的周期要长，我们选择 20ms，即 5 个角度环的控制周期来作为速度环的控制周期，这样可以有效的防止两环耦合。

电机 PWM 的确定使用 PI 控制，根据当前速度和期望速度偏差以及偏差的积累来确定 PI 参数，从而确定电机输出 PWM。

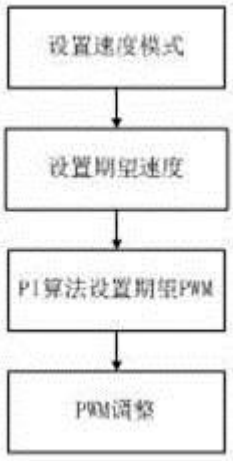


图 4-8　速度控制流程图

第 6 章 全文总结

6.1 智能车主要技术参数

表 6- 1 智能车主要技术参数

项目	参数
路径检测方法（赛题组）	电磁传感器
车模几何尺寸（长/宽/高）（厘米）	53/27/20
车模轴距/轮距（厘米）	轮距15
车模平均电流（匀速行驶）(毫安)	×
电路电容总量（微法）	无
传感器种类及个数	电感：7 陀螺仪 1 编码器 2 干簧管：1
新增加伺服电机个数	无
赛道信息检测频率（次/秒）	250
主要集成电路种类/数量	运放/7， 稳压/4， 电机驱动/4
车模重量（带有电池）（千克）	1.1

6.2 不足与改进

机械改装和支架设计中，其强度和稳定性需要更加注意，以免造成支架容易撞断和运行不稳定的情况；

硬件设计的兼容性和易扩展性考虑，在电路设计的过程中，端口分配是一个很重要的步骤，设计时需要预留一些可能用到的资源，以备后面增加是方便扩

展；在程序设计和调试的过程中，不能太模糊化，需要进行实际的建模和仿真； 对智能车的运行一定要进行定量的分析和计算，对智能车运行中出现的各种情 况都能做到理解—解决，站在理论支持的角度进行分析和调试。

在备赛过程中，对于学校内部应加强不同组别和不同传感器方案队间的交流，同时应加强与其他兄弟院校的交流，使智能车往更好的方向发展。

6.3 致谢与总结

经过半年多的设计、制作和调试，在队员们的分工和配合下，最终顺利完成了智能车电路、机械、程序等方面的工作。在这个过程中，大家学到了很多，积累了很多，能够一步一步走下来，靠的是整个团队的协作和团结。总体而言， 大家学到了很多，提高了很多，这将对以后的学习和工作都会起到很多有意义 的作用。

作为一个融合多学科交叉的的复杂系统，完成整智能车的设计、制作和调试是一个非常庞大的工程，仅靠几名队员是很难完成的，它需要一个高效运作、规范管理的团队的紧密合作才能完成。本文所论述的内容不仅仅是作者个人工作的结晶，同时也蕴含了他人的劳动、汗水和智慧。在本文即将结束之际，特向整个智能车团队的成员以及其它提供过帮助、建议和意见的人们表示衷心的感谢。

感谢指导老师的指导、支持和帮助。没有他的指导和帮助，整个智能车队就很难高效稳定运行下去，设计的方向也很难把握得这么准确。

感谢华中科技大学启明学院和自动化学院在调试场地，物资上的支持和帮助，正是有了学校的大力支持，才使我们能专心做事，仔细研究。

还要感谢教育部、自动化分教指委会给大家提供的这个锻炼团队合作和创新能力的竞赛平台，感谢各个赛区承办学校，感谢各个院校为大家提供优良的竞赛环境，使大家能够在竞赛中发挥出高水平。

参考文献

[1] 卓晴 黄开胜 邵贝贝 等.学做智能车. 北京：北京航空航天出版社，2007 年 3 月 第 1 版

[2] 第九届全国大学生“飞思卡尔”杯智能汽车竞赛秘书处，竞速比赛规则与赛场纪 律，2013/12

[3] National Semiconductor，LM2940 Datasheet，1999/06

[4] National Semiconductor，LM1117 Datasheet，2004/06

[5] 杨杰 吴凡.粗糙表面可见光散射特性的实验研究. 中国测试，2009，02 期 [6] HAMAMATSU，Light modulation photo IC S7136，2007/07

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

[8] 邵贝贝.UCOS-II 嵌入式实时操作系统. 北京：清华大学出版社，2002 年 10 月第 1 版

附录 A 源代码

因篇幅所限，只附录部分代码

```
/* ============================================================
    @brief  偏差计算
    @function 得到偏差、差分
    @parameter 无
    @note 内部无需修改
============================================================
*/
void Calculate_Turn_Offset(void)
{
    //static int16 ad_history_value[SENSOR_NUM][8];
    static int offset_history[6];
    uint16 x1,x2,y1,y2,t1,t2,m2;
    int i,j;
    float temp_offset,offset1,offset2;
        int offset_0;
        //Search_0();
        float ky=1;
        float kt=1.0;
        MAX_OFFSET=s_parameter.max_offset;
    for(i=0;i<SENSOR_NUM;i++)
    {
        for(j=7;j>0;j--)
        {
            ad_history_value[i][j]=ad_history_value[i][j-1];
        }
        ad_history_value[i][0]=sensor_value[i];
    }
```

```
x1=sensor_value[2];
x2=sensor_value[5];

y1=sensor_value[1];
y2=sensor_value[6];

t1=sensor_value[0];
t2=sensor_value[7];

    m2=sensor_value[4];



    if(x1<=3&&x2<=3&&t1<=3&&t2<=3&&y1<=3&&y2<=3)
    {
            blank_flag=1;
            lose_flag=0;
    }
    else
    {
        blank_flag=0;

        if(x1<10&&x2<10)
    {

            lose_flag=1;
    }
    else if((x1<=5)&&(x2<20)&&(t1<=5) && (t2<35)&&(y1<=35)&&(y2<=35) )
    {

            lose_flag=1;
    }
```

```
        else if((x2<=5)&&(x1<20)&&(t2<=5) && (t1<35)&&(y2<=35)&(y1<=35) )
        {

            lose_flag=1;
        }
        else if((x1<40)&&(x2<40)&&(t1<65)&&(t2<65)&(y1<35)&(y2<35))  //t57
y35  30
        {
            if(x1<24&&t1<12&&y1<18)
            {

                lose_flag=1;
            }
            else if(x2<24&&t2<12&&y2<18)
            {

                lose_flag=1;
            }
            else
                {
                    lose_flag=0;
                }
        }
        else
        {
            lose_flag=0;
        }




        }
```

```
        if(distance_0>=20&&distance_0<=60)
          BEEP_On();
        else
          BEEP_Off();

        Special_0();

        if(flag_LorR)
        {

            offset_0=0;
                      for(i=7;i>=0;i--)
            {
                          if(ad_history_value[7][i]>ad_history_value[0][i]+10)
                            offset_0++;
                          else
if(ad_history_value[7][i]<ad_history_value[0][i]-10)
                            offset_0--;

                          if(ad_history_value[6][i]>ad_history_value[1][i]+10)
                            offset_0++;
                          else
if(ad_history_value[6][i]<ad_history_value[1][i]-10)
                            offset_0--;

                          if(ad_history_value[5][i]>ad_history_value[2][i]+10)
                            offset_0++;
                          else
if(ad_history_value[5][i]<ad_history_value[2][i]-10)
                            offset_0--;
```

```
            }
                    if(offset_0>=0)
                    {
                       RorL=1;
                    }
                    else
                    {
                       RorL=-1;
                    }
                flag_LorR=0;
        }




         if(blank_flag)
         {
                    if(offset_history[3]==0)
                        {
                                current_offset=0;
            }
            else if(offset_history[3]>0)
                current_offset=-MAX_OFFSET;
            else
                current_offset=MAX_OFFSET;
    }

        else if(bend_cross_flag==1)
        {
```

```
            current_offset=offset_history[1];
        }


                else if(lose_flag==1)
        {
                 if(offset_history[1] > 0)
                        current_offset=MAX_OFFSET;
                else if(offset_history[1] < 0)
                        current_offset=-MAX_OFFSET;
        }
        else
                {
                  if(distance_0>10&&distance_0<=60)
                   {
                       if(RorL>=0)
                       {
                                        t2=(int16)(t2*((1-distance_0/90.0)>0?(1-
distance_0/90.0):0));
                                        y2=(int16)(y2*((1-distance_0/90.0)>0?(1-
distance_0/90.0):0));
                                        x2=(int16)(x2*((1-distance_0/90.0)>0?(1-
distance_0/90.0):0));
                       }
                       else
                       {
                                        t1=(int16)(t1*((1-distance_0/90.0)>0?(1-
distance_0/90.0):0));
                                        y1=(int16)(y1*((1-distance_0/90.0)>0?(1-
distance_0/90.0):0));
                                        x1=(int16)(x1*((1-distance_0/90.0)>0?(1-
distance_0/90.0):0));
```

```
                    }

                                test_flag=1;
                }


                    if(distance_0>90)
                      ky=0;

                  if(distance_0<=90)
                  {
                     if(y1>50&&y2>50)
                       kt=0.3;
                     else
                       kt=1.0;
          offset1 = (float)(1.35*kt*(float)(t1-t2)+1.2*(float)(x1-x2)
                          + 3.2*ky*(float)(y1-y2))

    /(float)(0.70*(float)(x1+x2)+0.75*kt*(float)(t1+t2)+0.75*ky*(float)(y1+y2))
;
          offset2 = ((float)kt*(t1+t2)-0.69*(float)(x1+x2))

    /(float)(0.70*ky*(float)(x1+x2)+0.75*kt*(float)(t1+t2)+0.75*(float)(y1+y2))
;


          if(offset1 >= 0)
          {
              temp_offset = (float)27*(offset1 + offset2);   //52  //45
          }
          else
```

```
            {
                temp_offset = (float)27*(offset1 - offset2);    //52
            }

            if((x1+x2 < 50) && (t1+t2 < 60) && (y1+y2 < 100))
            {
                temp_offset = (float)temp_offset * 1.08;//*1.3
            }
                    }
                    else
                    {
                       temp_offset=27*1.35*(float)(t1-
t2)/(0.75*(t1+t2))+s_parameter.0_kp*RorL;
                    }
                 current_offset=(int)-temp_offset;
        }

         if(current_offset>MAX_OFFSET)   //120.0
        {
            current_offset=MAX_OFFSET;

        }
        else if(current_offset<-MAX_OFFSET)
        {
            current_offset=-MAX_OFFSET;
        }
            for(i=5;i>0;i--)
            {
                offset_history[i]=offset_history[i-1];
            }
            offset_history[0]=(int)current_offset;
```

```
                if(!distance_0&&!flag_0)
                RorL=0;
}



void Calculate_Turn_Offset_delta(void)
{
    static int offset_history[OFFSET_HISTORY_NUM];
    int i;
    for(i=OFFSET_HISTORY_NUM-1;i>0;i--)
    {
        offset_history[i]=offset_history[i-1];
    }
    offset_history[0]=current_offset;

    current_offset_delt=offset_history[0]-offset_history[OFFSET_HISTORY_NUM-1];

}



/* ================================================================
    @brief  速度控制
    @function 得到左右轮的PWM
    @parameter 无
    @note 内部无需修改
================================================================
*/
uint8 Inc_speed_control(void)
{

        static int gyro_error_history[GYRO_HISTORY_NUM];
```

```
int16 left_speed_error = 0;
static int16 left_speed_error_last = 0;
int16 temp_left_pwm1 = 0;
int CurrentLeftSpeed;

uint8 i;
int16 right_speed_error = 0;
static int16 right_speed_error_last = 0;
int16 temp_right_pwm1 = 0;
int CurrentRightSpeed;



int Current_D_Speed;
int speed_D_error;
int16 speed_error = 0;
int16 temp_pwm1 = 0;
int16 temp_D_pwm1 = 0;
static int16 speed_error_last = 0;
static int16 gyro_error_last  =0;
static int16 speed_D_error_last =0;
float kp;
float kd;



float z1,z2;

int R_L1=(int)Get_current_offset();
int R_L2=(int)Get_current_offset_delta();
float x1,x2;
/*
if ( start_flag == 0)
```

```
{
    start_flag = 1;//发车
    soft_start_flag = 1;
}

*/

CurrentLeftSpeed=GetLeftSpeed();
CurrentRightSpeed=GetRightSpeed();
CurrentSpeed=(CurrentRightSpeed+CurrentLeftSpeed)/2;
Current_D_Speed=(CurrentLeftSpeed-CurrentRightSpeed)/2;
/*
x1=s_parameter.dir_kp*0.1*s_parameter.null_3_2*0.16*R_L1/120;
x2=s_parameter.dir_kd*0.1*s_parameter.null_3_2*0.16*R_L2/60;
*/


Get_Expected_PD(R_L1,R_L2,&kp,&kd);
/*
kp=(float)s_parameter.dir_kp;
kd=(float)s_parameter.dir_kd;
fuzzy_PD(R_L1,R_L2,&kp,&kd);
fuzzy_PD(R_L1,R_L2,&kp,&kd);

kp=kp*s_parameter.dir_kp;
kd=kd*s_parameter.dir_kd;
*/
x1=kp*100*R_L1;
x2=kd*100*R_L2;

x1=x1*0.1*0.16/120;
```

```
        x2=x2*0.1*0.16/120;
        z1=kp*120*R_L1;
        z2=kd*120*R_L2;
        z1=z1/1000;
        z2=z2/1000;
        //设定期望中心速度
        //set_expect_speed(Get_current_offset(), &left_expect_speed);
        left_expect_speed=L_test;
        right_expect_speed=R_test;

        //left_expect_speed=(int16)(s_parameter.cam_cnst+x1+x2);
        //right_expect_speed=(int16)(s_parameter.cam_cnst-x1-x2);

        if(R_L1>=40||R_L1<=-40)
          expect_speed = s_parameter.DN_speed;
        else if(R_L1>=30||R_L1<=-30)
          expect_speed = s_parameter.DN_speed+(s_parameter.UP_speed-
s_parameter.DN_speed)/4;
        else if(R_L1>=20||R_L1<=-20)
          expect_speed = s_parameter.DN_speed+(s_parameter.UP_speed-
s_parameter.DN_speed)*2/4;
        else if(R_L1>=10||R_L1<=-10)
          expect_speed = s_parameter.DN_speed+(s_parameter.UP_speed-
s_parameter.DN_speed)*3/4;
        else
          expect_speed = s_parameter.DN_speed+(s_parameter.UP_speed-
s_parameter.DN_speed);

        if(distance_0>0)
          expect_speed = s_parameter.DN_speed;
```

```
expect_D_speed =x1+x2;
expect_gyro=(int16)(z1+z2);



 if(left_expect_speed>left_last_speed+10)
   left_expect_speed>left_last_speed+10;
 else if(left_expect_speed<left_last_speed-10)
   left_expect_speed>left_last_speed-10;
 if(right_expect_speed>right_last_speed+10)
   right_expect_speed>right_last_speed+10;
 else if(right_expect_speed<right_last_speed-10)
   right_expect_speed>right_last_speed-10;

 if(expect_speed>last_speed+10)
   expect_speed=last_speed+10;
 else if(expect_speed<last_speed-10)
   expect_speed=last_speed-10;
 /*
 if(expect_gyro>last_gyro+10)
  expect_gyro=last_gyro+10;
 else if(expect_gyro<last_gyro-10)
   expect_gyro=last_speed-10;
 */
 left_last_speed=left_expect_speed;
 right_last_speed=right_expect_speed;
 last_speed=expect_speed;
 last_gyro=expect_gyro;
//left_expect_speed=L_test;
```

```
        //right_expect_speed=R_test;
        //left_expect_speed=150;
        //right_expect_speed=150;

//left_expect_speed=(int16)(s_parameter.cam_cnst+s_parameter.dir_kp*0.1*s_param
eter.null_3_2*0.16*R_L1/120+s_parameter.dir_kd*0.1*s_parameter.null_3_2*0.16*R_
L2/60);
        //right_expect_speed=(int16)(s_parameter.cam_cnst-
s_parameter.dir_kp*0.1*s_parameter.null_3_2*0.16*R_L1/120-
s_parameter.dir_kd*0.1*s_parameter.null_3_2*0.16*R_L2/60);


        //left_expect_speed=(int16)s_parameter.cam_cnst;
        //right_expect_speed=(int16)s_parameter.cam_cnst;

//left_expect_speed=(int16)(150+80*0.1*150*0.16*R_L1/120+10000*0.1*150*0.16*R_L
2/60);
        //right_expect_speed=(int16)(150-80*0.1*150*0.16*R_L1/120-
10000*0.1*150*0.16*R_L2/60);
        //left_expect_speed=(int16)s_parameter.cam_cnst;
        //right_expect_speed=(int16)s_parameter.cam_cnst;
        //left_expect_speed=L_test;
        //right_expect_speed=R_test;

        if(right_expect_speed>(s_parameter.speed_max))
            right_expect_speed=(s_parameter.speed_max);
        else if(right_expect_speed<(s_parameter.speed_min))
            right_expect_speed=(s_parameter.speed_min);
        if(left_expect_speed>(s_parameter.speed_max))
            left_expect_speed=(s_parameter.speed_max);
        else if(left_expect_speed<(s_parameter.speed_min))
```

```
            left_expect_speed=(s_parameter.speed_min);
        /*
        if(expect_speed>(s_parameter.speed_max))
            expect_speed=(s_parameter.speed_max);
        else if(expect_speed<(s_parameter.speed_min))
            expect_speed=(s_parameter.speed_min);
        */
        if(expect_gyro>(1000))
            expect_gyro=(1000);
        else if(expect_gyro<(-1000))
            expect_gyro=(-1000);
        if((GetLeftDistance()>150||GetLeftDistance()<-
150)&&CurrentLeftSpeed<50&&CurrentRightSpeed<50)
        {

          if(Stop<100)
            Stop++;

        }
        else if(Stop<50)
          Stop=0;



        left_speed_error = left_expect_speed - CurrentLeftSpeed;
        right_speed_error = right_expect_speed - CurrentRightSpeed;
        speed_error = expect_speed - CurrentSpeed;
        speed_D_error = expect_D_speed - Current_D_Speed;
        gyro_error=expect_gyro-gyro_value;
        if(gyro_error>1000)
          gyro_error=1000;
```

```
        else if(gyro_error<-1000)
          gyro_error=-1000;

        if(gyro_error-gyro_error_last>100)
          gyro_error=gyro_error_last+100;
        else if(gyro_error-gyro_error_last<-100)
          gyro_error=gyro_error_last-100;
        //gyro_error_delt=gyro_error-gyro_error_last;
        for(i=GYRO_HISTORY_NUM;i>0;i--)
            {
                gyro_error_history[i]=gyro_error_history[i-1];
            }
            gyro_error_history[0]=gyro_error;


        temp_left_pwm1 = (int16)Inc_get_expected_PWM(left_speed_error,
left_speed_error - left_speed_error_last);
        temp_right_pwm1 = (int16)Inc_get_expected_PWM(right_speed_error,
right_speed_error - right_speed_error_last);
        temp_pwm1 = (int16)Inc_get_expected_PWM(speed_error, speed_error -
speed_error_last);
        temp_D_pwm1 = (int16)Inc_get_expected_PWM(speed_D_error, speed_D_error
- speed_D_error_last);
        //temp_D_pwm1 = (int16)Inc_get_expected_PWM(gyro_error, gyro_error -
gyro_error_history[GYRO_HISTORY_NUM-1]);
        if(temp_left_pwm1>100)
          temp_left_pwm1=100;
        else if(temp_left_pwm1<-100)
          temp_left_pwm1=-100;
        if(temp_right_pwm1>100)
          temp_right_pwm1=100;
```

```
else if(temp_right_pwm1<-100)
  temp_right_pwm1=-100;

if(temp_pwm1>100)
  temp_pwm1=100;
else if(temp_pwm1<-100)
  temp_pwm1=-100;


left_speed_error_last = left_speed_error;
right_speed_error_last = right_speed_error;
speed_error_last = speed_error;
speed_D_error_last = speed_D_error;
Calculate_gyro_error_delta();
left_pwm  = (int16)(left_pwm + temp_left_pwm1);
right_pwm = (int16)(right_pwm + temp_right_pwm1);
speed_pwm = (int16)(speed_pwm + temp_pwm1);
D_pwm     = (int16)(gyro_error*7000/1000.0+gyro_error_delt*0/1000.0);
//D_pwm     = (int16)(D_pwm + temp_D_pwm1);

gyro_error_last=gyro_error;
if(left_pwm>700)
  left_pwm=700;
else if(left_pwm<-700)
  left_pwm=-700;
if(right_pwm>700)
  right_pwm=700;
else if(right_pwm<-700)
  right_pwm=-700;

if(speed_pwm>600)
```

```
  speed_pwm=600;
else if(speed_pwm<-600)
  speed_pwm=-600;

/*
if ( soft_start_flag == 1)
{
      if (GetLeftSpeed() <= 20)
      {
          left_pwm = start_pwm;
          right_pwm = start_pwm;
      }
      else if (GetLeftSpeed() <= 30)
      {
          left_pwm = start_pwm+20;
          right_pwm = start_pwm+20;
      }
      else if (GetLeftSpeed() <= 50)
      {
          left_pwm = start_pwm+50;
          right_pwm = start_pwm+50;
      }
      else if (GetLeftSpeed() <= 80)
      {
          left_pwm = start_pwm+100;
          right_pwm = start_pwm+100;
      }
      else if (GetLeftSpeed() <= 120)
      {
          left_pwm = start_pwm+200;
          right_pwm = start_pwm+200;
```

```
        }
        else if (GetLeftSpeed() <= 140)
        {
            left_pwm = start_pwm+300;
            right_pwm = start_pwm+300;
        }
        else
        {
            soft_start_flag = 2;
        }

        if (GetLeftDistance() >= 150)
        {
            soft_start_flag = 2;
        }
 }
*/
if(D_pwm>2000)
  D_pwm=2000;
else if(D_pwm<-2000)
  D_pwm=-2000;
/*
if(counter1/200%2==1)
{
  D_pwm=s_parameter.null_5_3;
}
else
{
  D_pwm=-s_parameter.null_5_3;
}
  */
```

```
if(Stop<50||SWITCH_ONE(SW_3))
{

  if(SWITCH_ONE(SW_3))
  {
    if(D_pwm>=0)
    {
      LeftMotor_PWM(D_pwm*10/s_parameter.null_5_0);
      RightMotor_PWM(-D_pwm*s_parameter.null_5_0/10);

    }
    else
    {
      LeftMotor_PWM(D_pwm*s_parameter.null_5_0/10);
      RightMotor_PWM(-D_pwm*10/s_parameter.null_5_0);
    }
  }
  else
  {
    if(D_pwm>=0)
    {
      LeftMotor_PWM(speed_pwm+D_pwm);
      RightMotor_PWM(speed_pwm-D_pwm);
    }
    else
    {
      LeftMotor_PWM(speed_pwm+D_pwm);
      RightMotor_PWM(speed_pwm-D_pwm);
    }
  }
```

```
          /*
          LeftMotor_PWM(200);
          RightMotor_PWM(200);
          */
          /*
          LeftMotor_PWM(L_test);
          RightMotor_PWM(R_test);
          */
        }
        else
        {
           LeftMotor_PWM(0);
          RightMotor_PWM(0);
        }

        return 1;
}//Inc_speed_control(void)
void Calculate_gyro_error_delta(void)
{
    static int gyro_error_history[GYRO_HISTORY_NUM];
    int i;
    for(i=GYRO_HISTORY_NUM-1;i>0;i--)
    {
        gyro_error_history[i]=gyro_error_history[i-1];
    }
    gyro_error_history[0]=gyro_error;

    gyro_error_delt=gyro_error_history[0]-gyro_error_history[GYRO_HISTORY_NUM-
1];

}
```