第十四届“恩智浦”杯全国大学生智能车竞赛

技术报告

学校：　　四川大学

队伍名称：　　OUTRAGE

参赛队员：　凌涌、龚兵、杨海鑫

带队教师：　　涂海燕

目录

第十四届“恩智浦”杯全国大学生智能车竞赛技术报告

凌涌 龚兵 杨海鑫

（四川大学电气工程学院，四川 成都 610065）

摘 要：本文介绍了四川大学室外越野电磁组队员们在参与第十四届“恩智浦杯”智能车大赛过程中的软件、硬件的设计思路和调试方法。智能车系统以MK60DN512为处理器芯片，软件平台为IAR开发环境，车模采用的是大赛组委会统一提供的L车模。整个系统涉及车模机械结构调整、电磁传感器设计及信号处理、控制策略和算法优化等多个方面。赛车采用谐振电路对赛道进行检测。

关键词：K60、PID 控制、赛道类型识别、电磁循迹

引言

1.智能车比赛背景

为加强大学生实践、创新能力和团队精神的培养，促进高等教育教学改革，受教育部高等教育司委托，由教育部高等自动化专业教学指导分委员会（以下简称自动化分教指委）主办全国大学生智能汽车竞赛。该竞赛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。

该竞赛以“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。

该竞赛由竞赛秘书处为各参赛队提供/购置规定范围内的标准硬软件技术平台，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。该竞赛融科学性、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。

该竞赛规则透明，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。该竞赛以飞思卡尔半导体公司为协办方，得到了教育部相关领导、飞思卡尔公司领导与各高校师生的高度评价，已发展成全国 30 个省市自治区近 300 所高校广泛参与的全国大学生智能汽车竞赛。

2008 年起被教育部批准列入国家教学质量与教学改革工程资助项目中科技人文竞赛之一（教高函[2007]30 号文）。全国大学生智能汽车竞赛原则上由全国有自动化专业的高等学校（包括港、澳地区的高校）参赛。竞赛首先在各个分赛区进行报名、预赛，各分赛区的优胜队将参加全国总决赛。

第一章 系统整体设计

1.1 方案设计思路

系统是以检测电磁场信号为基础，通过单片机处理信号实现对车体控制，实现车体能够准确沿着预设路径寻迹。系统电路部分需要包括单片机控制单元、电机驱动电路、电磁传感器电路等部分，除此之外系统还需要一些外部设备，例如编码器测速、伺服器控制转向、直流电机驱动车体，OLED 液晶显示屏显示赛道参数。

1.2 方案设计结构

由上述结果知，本智能车系统包含了以下几个模块:

1.电源模块
2.单片机最小系统模块
3.电磁传感器模块
4.电机驱动模块
5.测速模块
6.起跑线检测（霍尔元件）模块
7.人机交互（OLED 显示屏）模块
8.陀螺仪模块

系统整体模块图如图 1.1.1 所示：

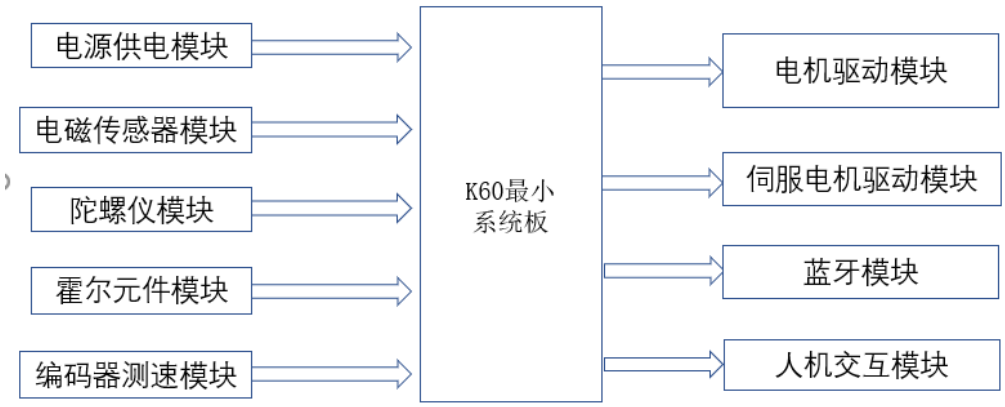


图 1.1.1 整体模块图

系统主要含有9个模块，各模块作用如下：

1. 电源模块：为整个小车的控制系统提供电能。
2. 单片机 K60 主控模块：作为整个智能车的大脑，对传感器采集回来的信息进行数据处理，控制小车的运行。
3. 测速模块：用于检测当前小车车速，与设定值进行比较处理，形成闭环控制，小车根据偏差进行相应的调整运行车速。
4. 起跑线检测模块：使用霍尔元件，检测起跑线。
5. 人机交互模块：包括 OLED 液晶显示模块、蓝牙模块、拨码开关、按键开关等，主要用于智能车的功能调试、赛车赛道状态监控等。
6. 电机驱动模块：驱动电机运转。
7. 陀螺仪模块：可以检测小车方向转动的角速度。
8. 电磁传感器模块：用于检测赛道信息，通过前瞻提前感知前方赛道信息，为智能车的下一步动作提供参考依据。

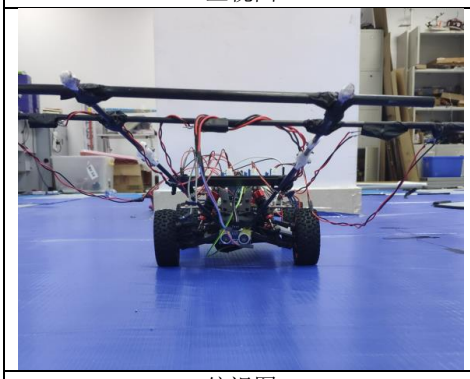
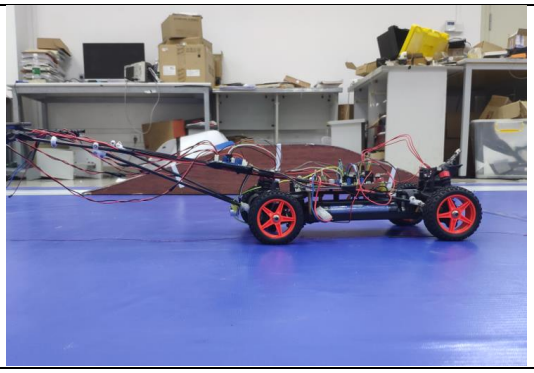
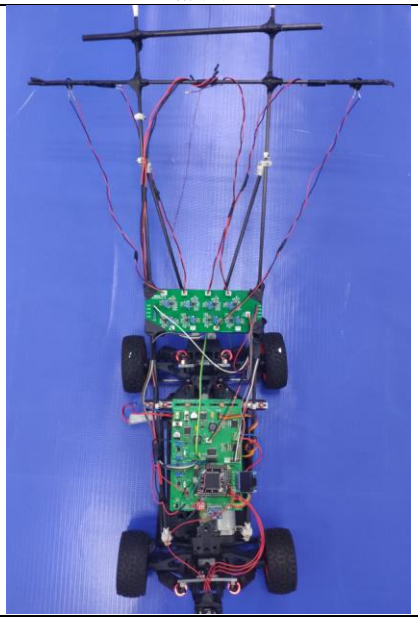
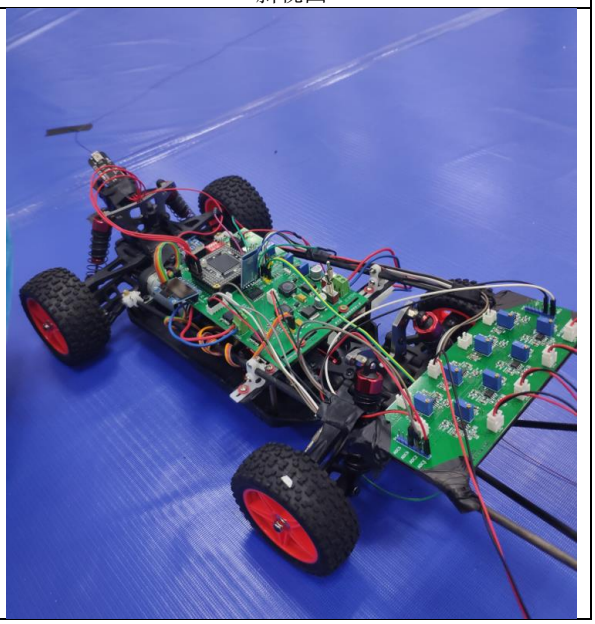
第二章 机械结构调整与优化

2.1 智能车整体布局

智能车的整体布局包括传感器的排布，电机放置的位置、高度，车体重心的分布，这些都将对智能车的运行性能产生重要影响。因此智能车的布局，应尽量使小车左右平衡，因我们的车为四轮驱动，故重心尽量落在车体中心，保证小车既能够可靠地抓牢地面，又能快速转弯，且将大电流电路和小电流电路画在了同一块板子上，方便更改硬件结构。

整体架构布局如下所示：

表 2.1.1 智能车整体结构图

正视图	侧视图
	
俯视图	斜视图
	

2.2 电磁传感器

电磁传感器作为智能车的“眼睛”，应尽量使传感器往前伸。本届室外越野电磁组参赛车模具没有尺寸限制，我们设置整个车模长度从后轮中心到最前方传感器部位距离为 66cm，因此我们的传感器延伸长度为 29cm。使用 5 个工字电感，最左、中间、最右三个工字电感平行于赛道方向放置，用于检测包括直行赛道、转弯赛道。左中、右中的工字电感垂直赛道方向放置，用于检测十字交叉赛道。五个电感均匀放置，总宽度为 40cm。为了扩大电感检测范围，应使电感尽可能高，调整放大器增益，调整高度为 17cm。

第三章 硬件电路设计说明

3.1 单片机最小系统板

本车主控芯片采用恩智浦公司 K60，其最小系统板中带有晶振电路、复位电路、调试接口、串行通信接口、无线收发接口。

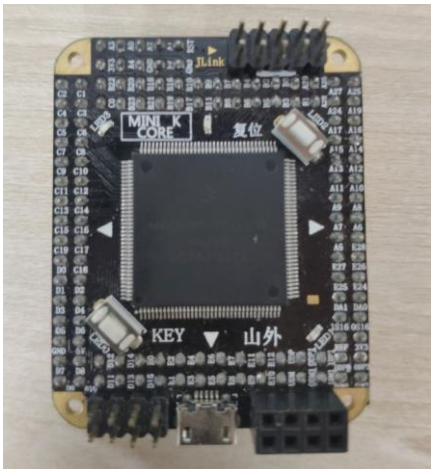


图 3.1.1 K60 最小核心板

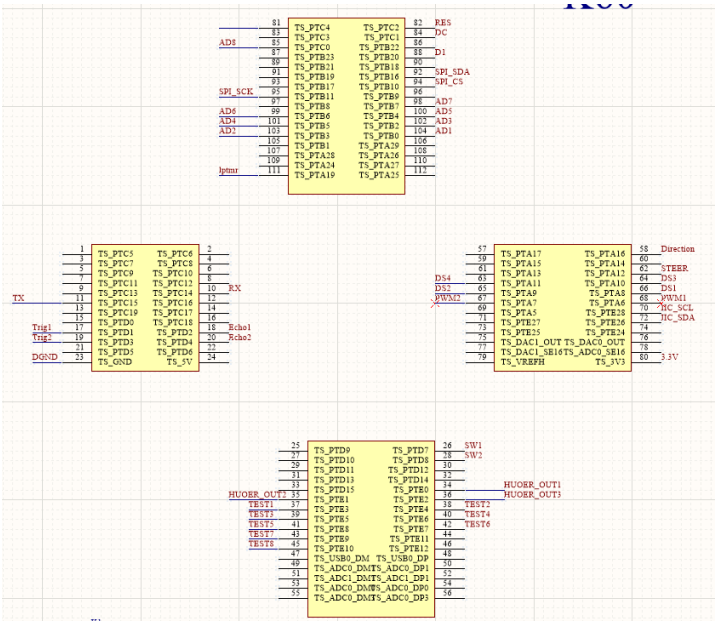


图 3.1.2 K60 最小核心板接口原理图

3.2 电源模块

在电源模块设计中，由于受到电源转换效率、相互干扰和降低噪声等多方面因素的影响，需考虑电源模块由若干相互独立的稳压电路组成。根据各模块电压的需求，合理设计不同的稳压模块。小车系统电源采用比赛组织提供的 7.2V-2000mAh 可充电镍镉蓄电池，满电时电池电压在 7.8-8.2V 之间，该电压可直接用于直流电机驱动模块供电。我们使用 LM2596 芯片提供 5V 降压，用于电磁传感器和单片机、编码器等供电。由于舵机功率需求较大，瞬时电流较大，若采用同一 5V 电源供电，可能会出现供电不足，单片机重启等情况，故舵机和单片机、编码器等数字电路部分分开供电，设计两路 LM2596 实现模拟和数字电路分别 5V 供电。使用 AMS1117-3.3 芯片实现 3.3V 降压，主要用于陀螺仪和 OLED 模块供电。电源模块主要电路如下所示：

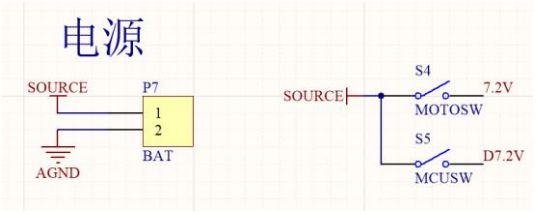


图 3.2.1 电池供电电路

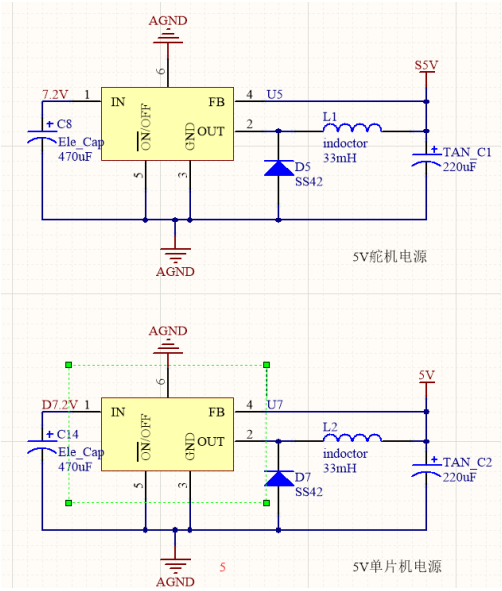


图 3.2.2 双路 LM2596 供电电路

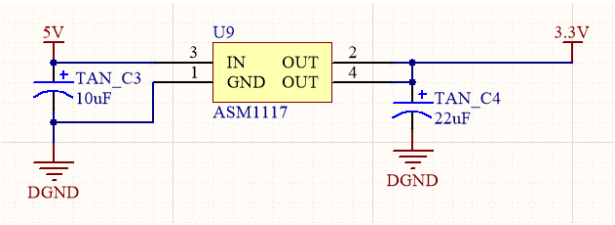


图 3.2.2 ASM1117-3.3V 供电电路

3.3 电磁传感器模块

电磁传感器用于小车的循迹，电磁检测的性能的好坏直接决定了智能车循迹的好坏。赛道中心铺设通有 100mA，20KHz 交变电流的漆包线，根据电磁感应定律，选用的 10mH 工频电感与 6.8nF 的电容并联，组成 LC 并联谐振电路，输出频率为 20KHz 的交流电压信号。由于电感线圈感应出的电动势信号微弱，只有十几毫伏，故我们使用带宽增益为 1MHz 的运算放大器 LMV358 芯片进行信号放大。电磁传感器模块原理图如下图所示：

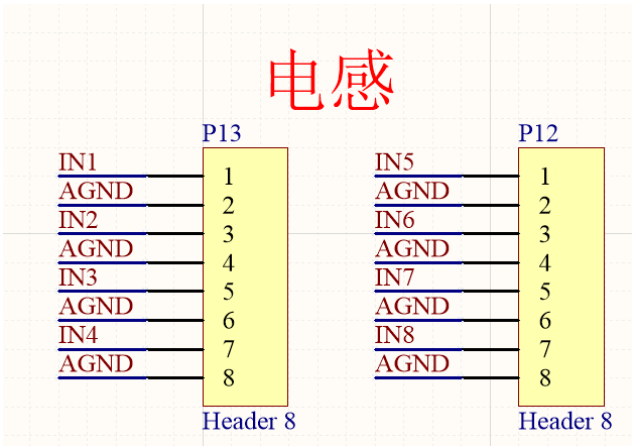


图 3.3.1 电感接口

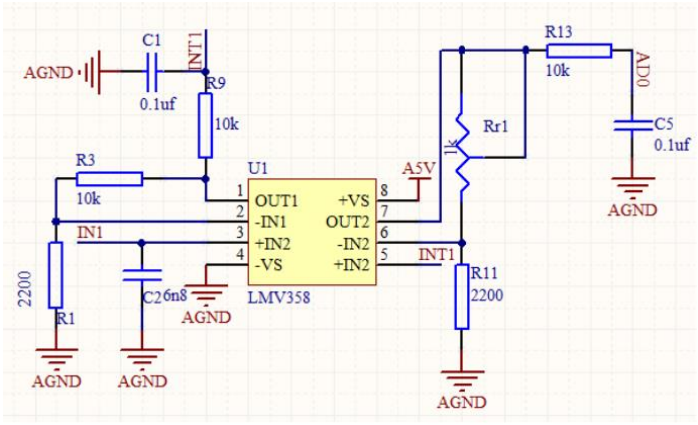


图 3.3.2 电磁信号二级放大电路

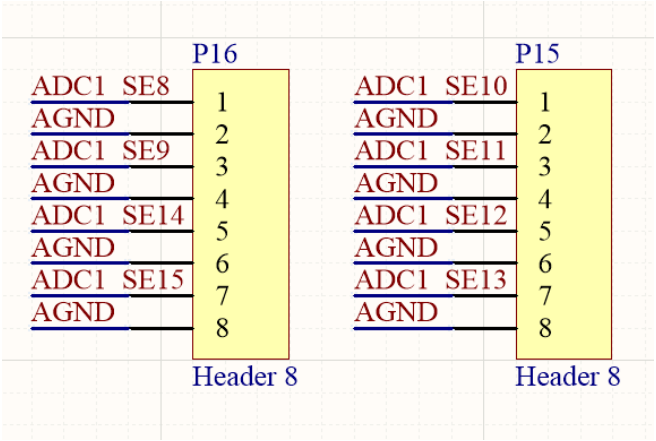


图 3.3.3 电磁信号输出接口

3.4 电机驱动与隔离模块

由于单片机产生的 PWM 脉冲信号无法驱动智能车提供的直流电机，需要设立独立的驱动电路来实现电机驱动，以获得足够的功率。考虑到车模电机电流较大，故驱动电路要有较大的电流输出能力。故选驱动芯片 BTS7960 搭建 H 桥，此种方法搭建的 H 桥最大可驱动电流为 43A，足够提供电机所需电流，且根据后期测试，发热情况并不严重，且电机变速能力较好。电机驱动电路如下所示：

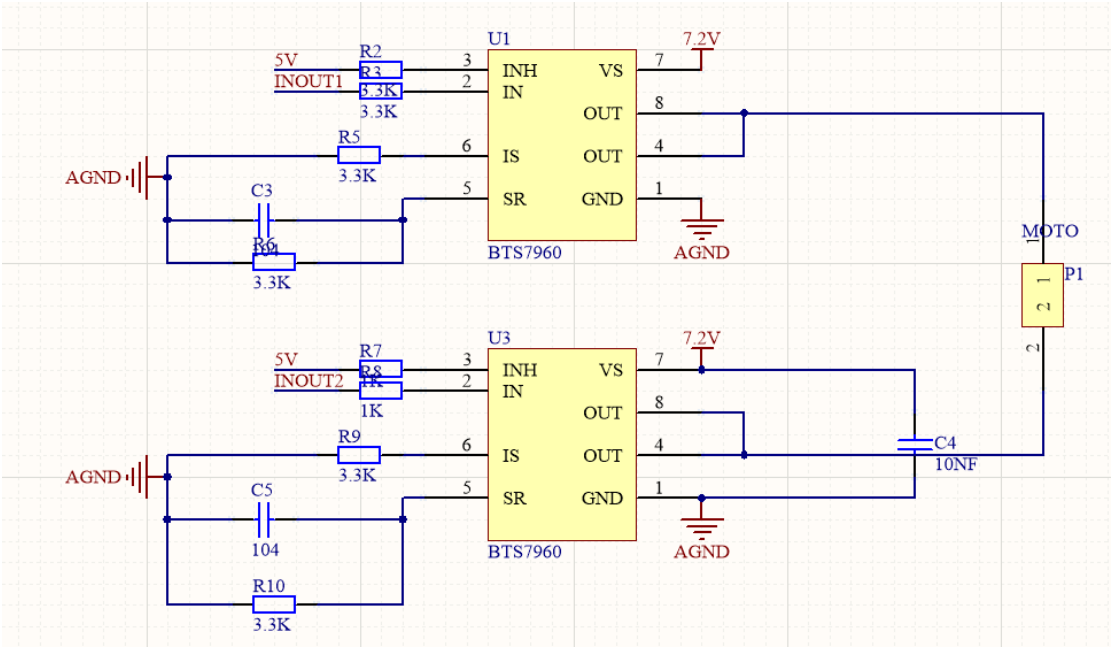


图 3.4.1 电机驱动电路

由于小车电机的驱动信号由单片机输出 PWM 信号进行控制，电机驱动部分属于大电流部分，为防止驱动部分的大电流流回单片机导致烧毁单片机，应加入隔离模块，将大电流部分与单片机进行隔离，且能改变电平等级，将单片机输出的 3.3V 电平提升为 5V 电平。我们使用三态门电路芯片 74HC244D 进行隔离，A1-A3 引脚输入单片机输出信号，Y1-Y3 则输出对应的信号，其原理图接线如图 3.4.2 所示：

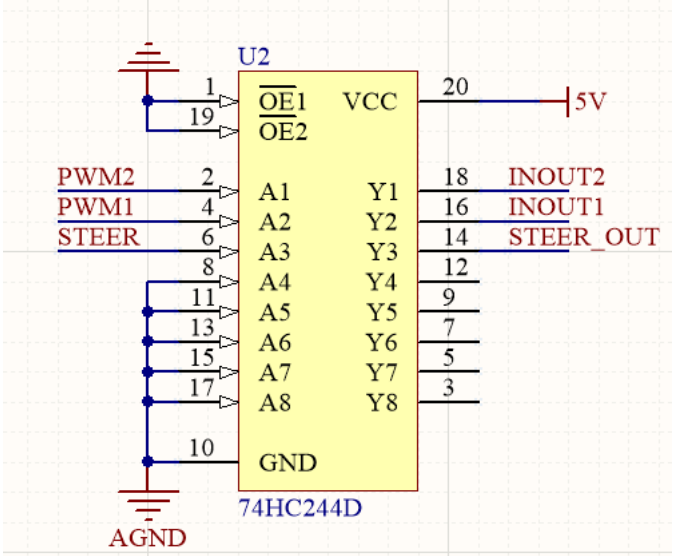


图 3.4.2 信号隔离电路

3.5 起跑线检测霍尔模块

起跑线检测一般有两种方案。

方案一：使用干簧管，干簧管内部由两片端点重叠可磁化的簧片组成，一般情况下两簧片分开几微米，当小车经过贴有磁铁的起跑线时，两簧片在磁场作用下产生不同极性，使两片不同极性的簧片因异性相吸而闭合，从而导致簧片的端接口电平发生改变，触发停车程序。

方案二：使用霍尔传感器，通过霍尔传感器来检测安装于起跑线下方的磁铁，便可以稳定、简单的检测出起跑线，并触发停车程序。

因干簧管怕震动，导通原理其实是通过簧片机械连接，震动对它来说是致命的，故综合考虑，我们采用了霍尔传感器来实现起跑线检测，并通过将霍尔元件布置在不同的位置和方向，来对起跑线进行精确检测。

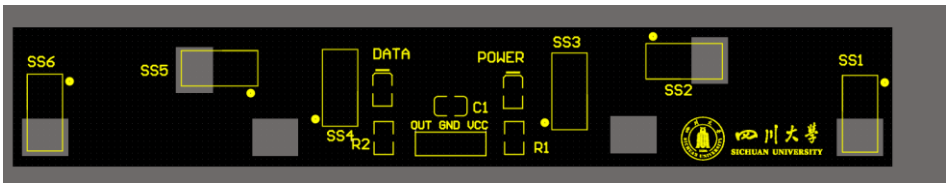


图 3.5.1 霍尔传感器布局示意图

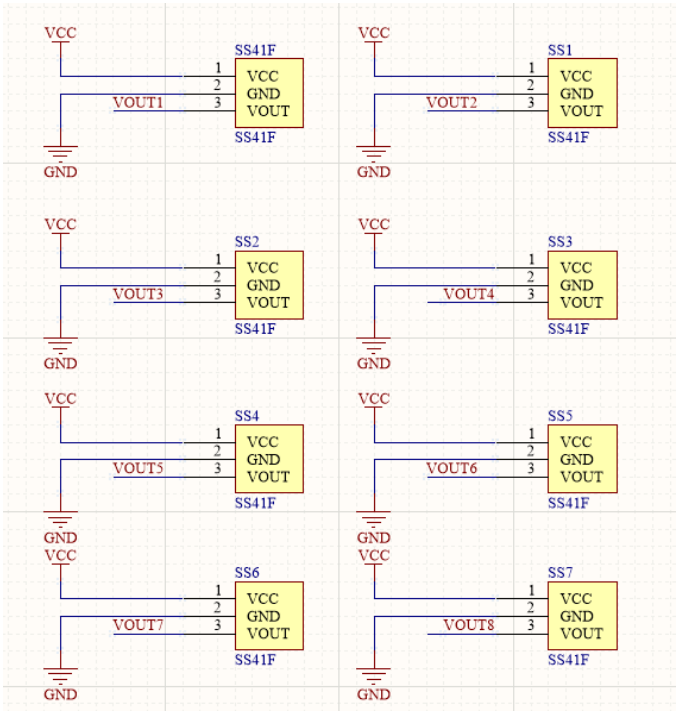


图 3.5.2 霍尔传感器原理图

3.6 人机交互模块

人机交互模块包括蓝牙模块、OLED 液晶显示模块、拨码开关、按键开关（加入硬件滤波消抖）。按键和 OLED 模块主要用于小车的功能调试、赛道状态检测，陀螺仪模块主要用于实现转向闭环。除蓝牙模块和陀螺仪模块之外，其余功能均使用的是普通 IO 口，使用模拟 I2C 等方式读取数据，其原理图如下所示：

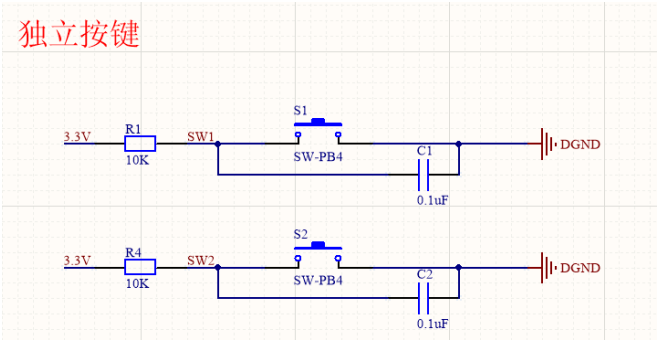


图 3.6.1 带硬件消抖的按键模块

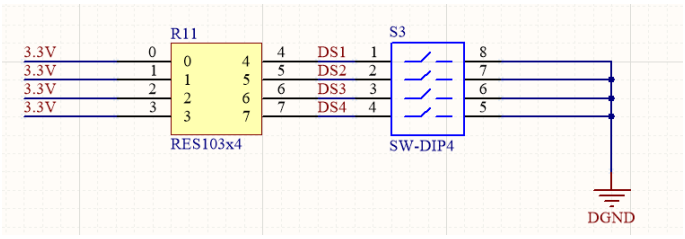


图 3.6.2 拨码开关模块

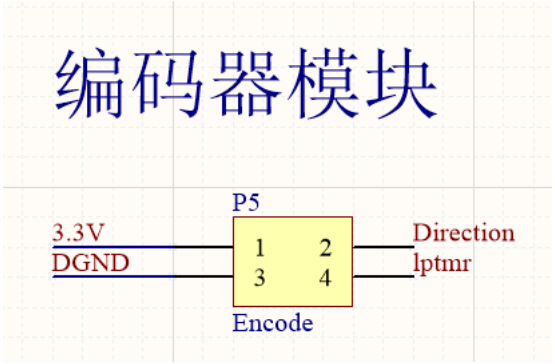


图 3.6.3 编码器模块

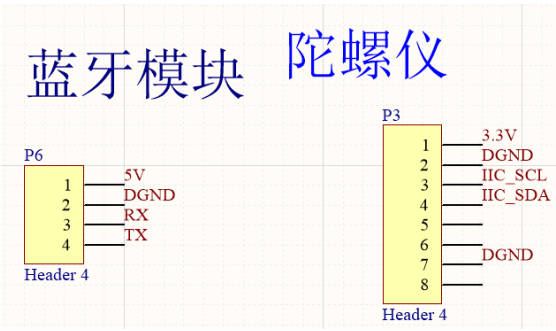


图 3.6.4 蓝牙和陀螺仪模块

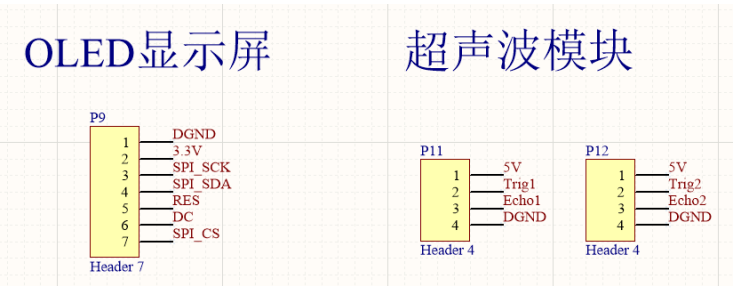


图 3.6.5 OLED 和超声波模块

第四章 软件系统及控制策略设计

4.1 软件开发平台

此次智能车软件开发环境采用了恩智浦 K60 系列单片机开发软件 IAR。该软件具有支持多种语言、开发界面统一、交叉开发平台以及支持插件工具等特点。

4.2 软件系统整体设计

在整个系统设计中，主要用到了单片机六个基本功能模块，包括：A/D 转换模块，FTM 模块，GPIO 端口模块，UART 模块，I2C 模块，PIT 定时器模块。通过配置寄存器先对所使用的模块进行硬件初始化，并通过相应的数据寄存器或状态寄存器的读写，实现期望的功能。为实现期望的功能所需芯片资源如下表所示：

表 4.2.1 芯片资源分配表

模块名称	模块号	连接器件或设备
AD 模块	ADC0、ADC1	电磁传感器
GPIO 模块	PORTA-PORTE	拨码开关、按键开关、霍尔元件、OLED 显示屏、电机控制、编码器
PIT 模块	PIT0	定时器中断
FTM 模块	FTM0、FTM1	电机控制
IIC 模块	IIC1	陀螺仪
UART 模块	UART0	蓝牙

4.3 电磁巡线原理

比赛中存在两种磁场：一种是用来作为起跑线的永磁铁产生的固定磁场，该磁场不影响交变磁场。一种是赛道中通过 100mA 交变电流的漆包线所产生的电磁场，用于赛道路径识别。前者可以用霍尔元件来检测，霍尔元件在平时输出为高电平，在经过永磁体时输出变为低电平，通过 I/O 输入读取其值，即可判断终点线。在交变的磁场中，工频电感产生的交变电压随着距离增加而减少，根据这个特性我们可以判断小车偏离中线的距离。

设置五个电感排布在前瞻上，工频电感的排列有两种，一种是与交变电磁线平行，一种是与交变电磁线垂直。与赛道垂直的电感用于判断偏差，如图 4.1 所示的 1 号、2 号、3 号电感，与赛道平行的电感用于判断赛道类型：十字路、直角等特殊路况，如图 4.1 所示的 4 号、5 号电感。尤其是直角弯路段，4、5 号电感起到了关键的作用。

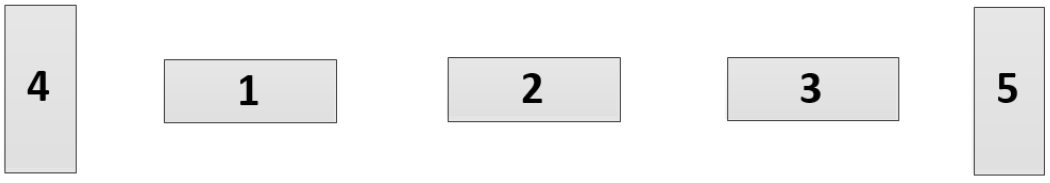


图 4.1 电感分布图

4.4 控制策略

四轮车可以利用舵机控制方向，故可以使用一个位置环 PD 控制器来控制，将电感传感器传回的数据进行处理后得到的离中心线的偏差，通过 PD 控制器的计算，舵机需要执行的打角值，而由于期望小车在直线运行时速度尽可能的快，而在过弯时速度要适当降低以避免发生事故，故位置环要输出一个负值送给速度环来降低期望速度值，达

到过弯减速的目的。而速度环可以采用 PID 控制器，输入为从编码器采集到的脉冲数经过运算转换后得到当前转速，输出为PWM波，控制H桥驱动电路以实现对电机转速的调节。

方框图如图 4.4.1 所示：

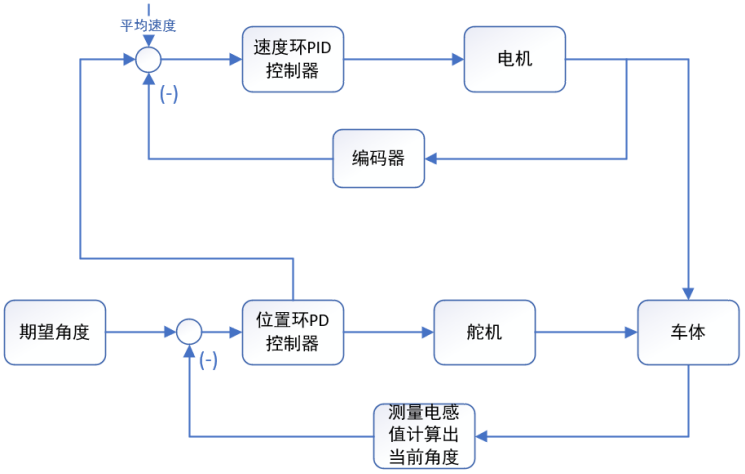


图 4.4.1 控制方框图

4.5 中心偏差计算方法

中心偏差的计算是控制小车巡线的核心，如果得到的偏差不合适，则小车很容易出现丢线、摆头等现象。理论上，我们希望小车的偏差曲线呈线性的关系，但实际上一个电感的电压值与距离中线距离的关系如下：

$$U=\frac{k}{x^2} \qquad \text{（公式 1）}$$

其中 U 为实测电压值，可以通过 A/D 转换变为数字量，x为实际距离，k为放大的增益。很明显计算偏差不是线性变换，并且电感随小车在平面上的运动不是相对轴线原理的运动，这让偏差的计算更难整定到理想的线性。

为了使偏差的曲线体现出左右方向的极性，传统的偏差计算一般使用差比和，计算公式如下：

$$偏差=\frac{左电感值（1）\text{-}右电感值（5）}{左电感值（1）+右电感值（5）} \qquad \text{（公式 2）}$$

数字对应于图 4.1 的各电感排布位置序号。

当车体向右偏时，左电感靠近中心线电压值增大，右电感远离中心线电压值减少，得到的偏差计算值为正。当车体向左偏时，左电感远离中心线电压值减少，右电感靠近中心线电压值增大，得到的偏差计算值为负。分母求和保证偏差值在车体靠左和靠右时为增加的。

但是传统的偏差计算值并不是很优良，其容易出现丢线等情况，会增加控制器设计的难度。所以我们使用了另一种可以简化控制器设计的算法：

$$偏差=\frac{左电感值（1）\text{-}右电感值（5）}{（左电感值（1）+中电感值（3）+右电感值（5）\times 中电感值（3）} \qquad \text{（公式 3）}$$

以上的偏差的原理类似于传统偏差计算，由于引入了中间电感值，中间电感越偏离中线，其值越小，这让大偏差越大，小偏差较小，得到一个类似于三次函数的偏差曲线

以下为实测数据的 matlab 拟合曲线：

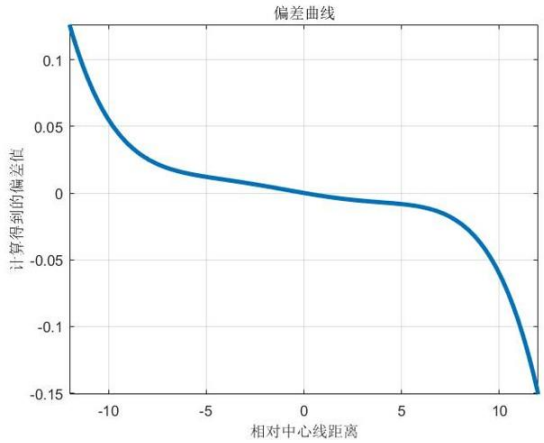


图 4.5.1　实测偏差计算值曲线

以上偏差计算值不是理想的线性，但是它单调且斜率一直增加，这可以帮助我们实现小弯转小角，大弯转大角的优良特性。

位置环使用 PD 控制器，由于这种曲线的特性，会使高速入大弯时，转向输出非常灵 敏，微分项 D 的输入（即偏差变化率）也会变得很大，相当于有一个动态的修正系数改变 D 得取值，使控制难度大大降低。

4.6 控制器的设计

控制器的设计决定了小车速度的上限，小车中速度环和位置环分别使用一套控制器。他们的具体参数和设计方法均不相同，核心算法是 PID。

在生产过程自动控制的发展历程中，PID 控制是历史最悠久、生命力最强的基本控制方式。它以其原理简单、使用方便、鲁棒性强、适应性广等优点，从而对大多数控制系统具有广泛的适用性。PID 控制主要由三个部分组成：比例、积分、微分。

比例控制能迅速反应误差，从而减小误差，但比例控制不能消除稳态误差，Kp的加大，易引起系统的不稳定；积分控制的作用是，只要系统存在误差，积分控制作用就不断地积累，输出控制量以消除误差，当积分作用太强易引起振荡；微分控制能够预测误差变化的趋势，提前使误差变为 0，甚至为负值，从而避免了被控量的严重超调。

PID 控制器结构如图 4.6.1 所示：

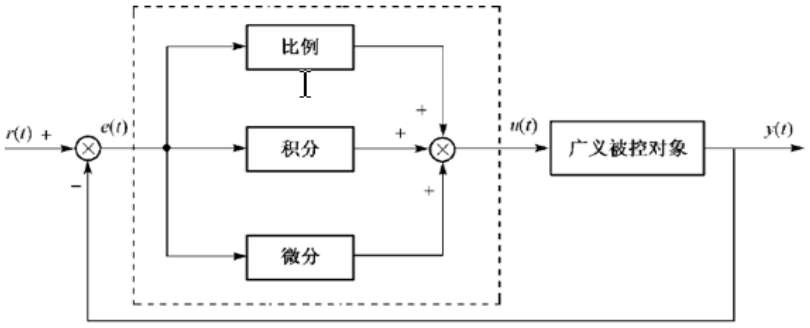


图 4.6.1 PID 控制器

对应的误差传递函数为：

$$\frac{U(s)}{E(s)} = K_p(1+\frac{1}{T_i}+T_d) \qquad \text{（公式 4）}$$

式中，Kp 为比例增益，Ti 为积分时间常数，Td 为微分时间常数，U(s)为控制量，E(s)为被控量 Y(s)与设定值 R(s)的偏差,时域表达式为:

$$u(t) = K_p\left[e(t)+\frac{1}{T_i}\int_0^t e(t)dt + T_d\frac{de(t)}{dt}\right] \qquad \text{（公式 5）}$$

由于单片机仅能对数字信号进行处理，将连续 PID 控制改为数字 PID 控制，将公式 5 进行离散化，得：

$$u(t)=K_p\left\{e(t)+\frac{T}{T_i}\sum_{j=0}^{k}e(j)+\frac{T_d}{T}\left[e(k)-e(k-1)\right]\right\} \quad \text{（公式 6）}$$

4.6.1 位置式 PID 控制算法

直接利用上述离散化公式 6 计算。由于积分项（PI）是将所有采集值偏差相加，在一段时间后会很浪费单片机资源。且位置算法要用到过去的误差的累加值，容易产生较大的累加误差。位置算法的输出是控制量的全量输出，误动作影响较大。对其稍加改进，得到增量型 PID 算法。

4.6.2 增量式 PID 控制算法

由公式 6 知，第 k-1 个采样周期的控制量为：

$$u(k-1)=K_p\left\{e(k-1)+\frac{T}{T_i}\sum_{j=0}^{k-1}e(j)+\frac{T_d}{T}\left[e(k-1)-e(k-2)\right]\right\} \quad \text{（公式 7）}$$

公式 7 减去公式 6 可得：

$$\Delta u(k)=q_0e(k)+q_1e(k-1)+q_2e(k-2) \quad \text{（公式 8）}$$

其中，$q_0=K_p(1+\frac{T}{T_i}),q_1=-K_p,q_2=K_p\frac{Td}{T}$。由公式 8 可知，利用两个历史数据，递推使用，即可完成增量式 PID 控制，框图如下所示：

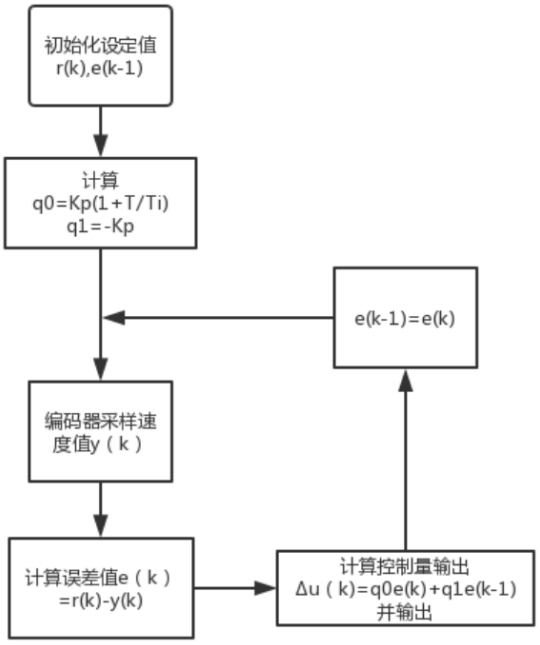


图 4.6.2 增量式 PID 控制器

相对于位置式 PID 而言，增量算法不需要做累加，控制量增量的确定仅与最近几次误差采样值有关，计算误差或计算精度问题，对控制量的计算影响较小。且增量式算法得出的是控制量的增量，误动作影响小，必要时通过逻辑判断限制或禁止本次输出，不会严重影响系统的工作。

4.6.3 位置巡线环设计

位置环采用的是位置式 PD，由于位置的设定值为一个随动值且 I 的值难以整定，使用 PD 控制。位置环的输入恒为零，误差就是通过公式（3）计算得到的值，即为小车偏离中心线的程度，送入位置环 PD 控

制器计算，得到的结果即为在此偏离程度下小车舵机需要的打角值，送给舵机执行打角操作。

4.6.4 速度跟踪环设计

速度环的控制采用的是位置式 PID 控制器， 由于编码器的采样值具有很多噪声，即采样回来的转速会频繁抖动，若不处理直接送给控制器运算，将会导致电机运行更为不平稳，故需要对编码器采集回来的转速信息进行滤波处理，我们采用惯性滤波的方式来处理后，从编码器读取回来的转速信息抖动情况大有缓解，但同时滤波处理后的数据将会比实际稍有滞后，但并不影响整个系统的运行。

4.7 停车线检测算法

在小车底盘上安装有八个霍尔元件，且这八个霍尔元件的输出端均连接在一片八输入与非门上，八输入与非门的输出连接到单片机的 IO 口，在正常运行时霍尔元件输出均为高电平，这样与非门的输出为低电平，当小车经过如下图所示的停车线时，八个霍尔元件总有一个会检测到磁场而输出低电平，此时与非门的输出将有低电平跳变为高电平，小车将停车。

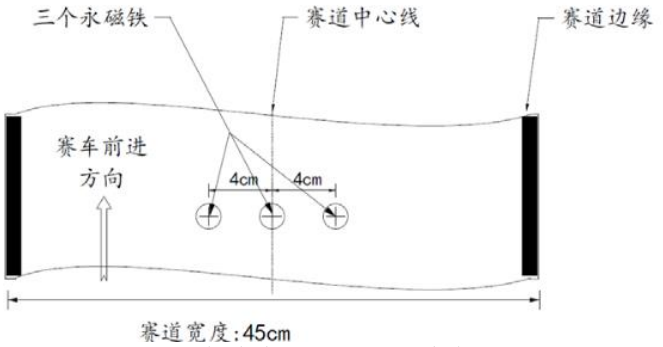


图 4.7 停车线中间的永磁体安装位置

4.8 特殊赛道元素处理

4.8.1 十字路段、折线路段

车辆通过十字交叉路口需要直行，不允许左转、右转。如下图 4.8.1 所示： 其特征是磁场方向合成，4、5 号电感输出增加，而 1、2、3 号电感不受影响，所以使用上述偏差计算公式获取偏差即可顺利通过十字路段，不会误判。

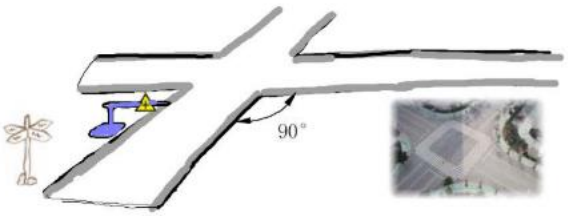


图 4.8.1.1 十字交叉路口

室外越野组的弯道均由折线构成，线夹角不小于 90°，折线长度不小于 25 厘米，对于折线角度大于 120° 的路段，只需要适当加长电磁杆前瞻长度即可使用上述偏差计算公式正常通过，而对于折线角度为 90° 的路段，由于此时电磁线已经与电感水平，使用上述偏差计算公式已经无法正常通过，而此时竖直的 4、5 号电感与电磁线垂直，故可用 4 号电感值-5 号电感值得到偏差，若为正值则表明车子舵机需要向左打死转弯，若为负值则表明车子舵机需要向右打死转弯。当 1、2、3 号电感采样到的电压值重新回到一定阈值有则表明车子已经转弯完成则释放原打死的舵机，转由偏差计算公式所得到的转弯信息控制舵机使小车继续正常运行，小车运行流程图如图 4.8.1.3 所示。

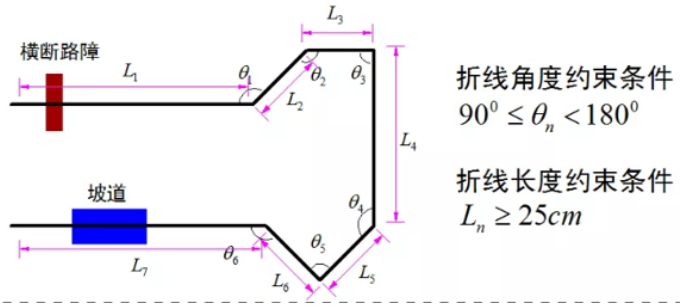


图 4.8.1.2 折线路段

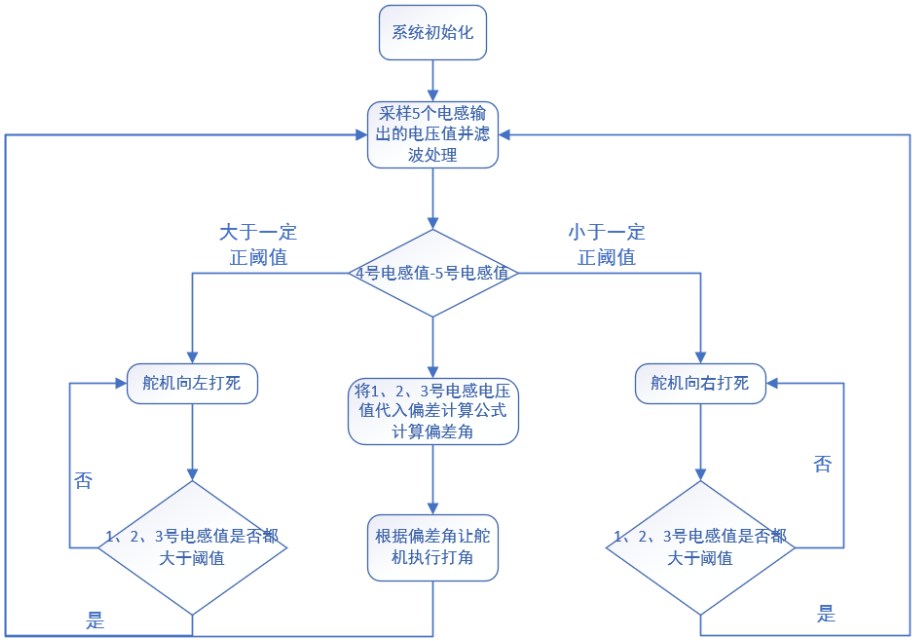


图 4.8.1.2 折线路段程序判别执行流程图

4.8.2 参数动态化

在智能车巡线行驶过程中，在经过了一长段直道加速后可能遇到急弯，如果整条赛道只使用一套 PID，很可能导致小车在过弯时由于打角不够而冲出赛道。在我们的策略中，通过编码器的采样值得到的速度和偏差变化率作为判断条件。根据速度等级和偏差变化率等级，修改位置环 PID 的参数，在实际应用中得到了较好的效果。

第五章 系统调试

5.1 开发环境

IAR Embedded Workbench 是瑞典 IAR Systems 公司为微处理器开发的一个集成开发环境（下面简称 IAR EW），支持 ARM，AVG，MSP 等芯片内核平台。EWARM 中包含一个软件的模拟程序（simulator）。用户不需要任何硬件支持就可以模拟各种 ARM 内核、外设甚至中断的软件运行环境。它为用户提供一个易学和具有大量代码继承能力的开发环境。

IAR EWARM 的主要特点如下：

1. 高度优化的IAR ARM C/C++ Compiler
2. IAR ARM Assembler
3. 一个通用的IAR 5LINK Linker
4. IAR 5AR 和5LIB 建库程序和IAR DLIB C/C++运行库
5. 功能强大的编辑器
6. 项目管理器
7. 命令行实用程序
8. IAR C-SPY 调试器(先进的高级语言调试器）

下图为IAR界面：

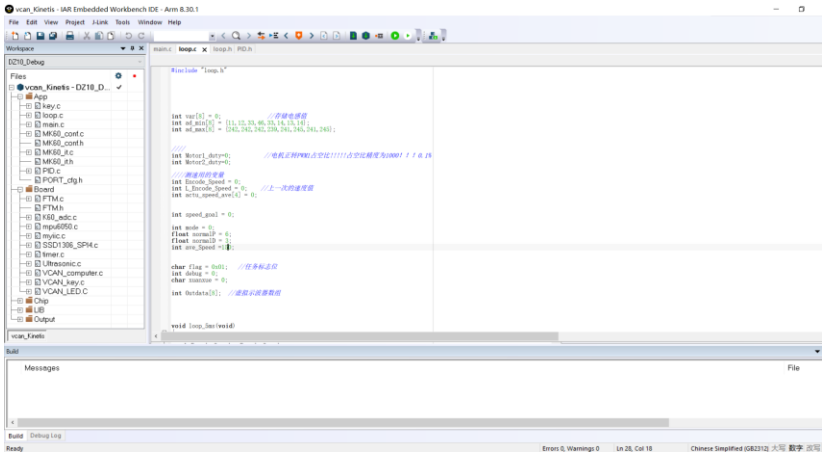


图5.1.1IAR界面

5.2 上位机调试软件

上位机调试软件使用的山外多功能调试助手 V1.1.9，这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，网络调试助手等调

试功能，满足多种需求。图为上位机读取到的某一个波形：

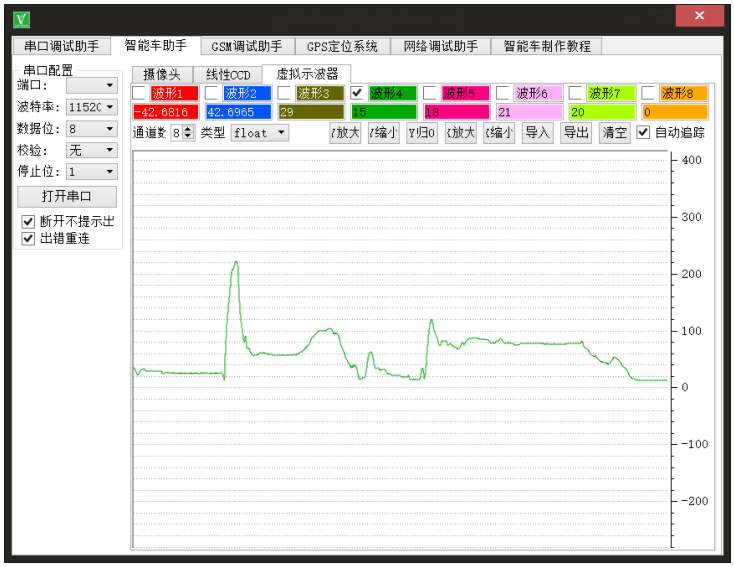


图 5.2.1 上位机调试助手界面

5.3 蓝牙调试

调试方法：我们使用蓝牙模块实现智能车和上位机的通讯，将智能车行进过程中的数据发回到上位机并绘制曲线，用以分析和修改参数。显示陀螺仪、加速度计的波形，显示赛道检测电感的波形，从而达到参数的采集。

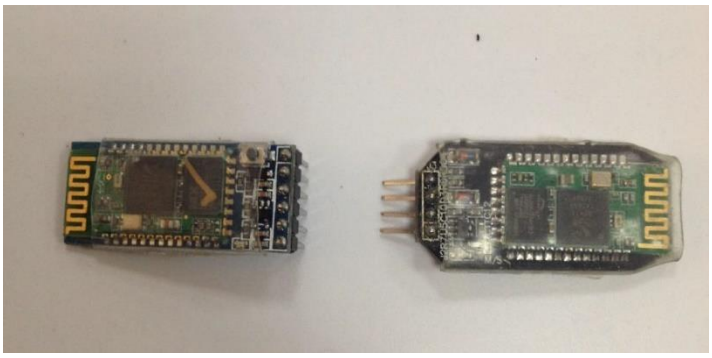


图 5-2 蓝牙模块

5.4 速度环调试

在许多智能小车中，均有使用 PID 调节器来实现速度闭环，以使得小车的实际速度值能够紧跟期望速度值，接下来将以一台寻迹小车为例简单讲解在实际工程应用中如何通过经验法来进行 PID 参数的整定。

如图 5-1 所示，这是小车控制系统在 P=1，I 和 D 均为 0 时，即控制器的增益为 1 时的闭环调速系统阶跃响应曲线。

图 5-1 原始系统速度响应曲线

其中蓝线为期望速度值，红线为实际速度值，从上图中可以看到，跟踪稳态误差为 50%左右，系统的响应速度也很不理想，这样的系统在智能小车上是根本无法使用的，故需要引入 PID 控制器来实现对电机速度的精确控制。

```
speed_erL=speed_er;
speed_er = (speed_goal-speed_actual);    //本次误差
speed_erI += speed_er;

speed_out = Kp*speed_er + Ki*(speed_erI) + Kd*(speed_er-speed_erL);    //位置式PID计算

if(speed_erI >= 1000)
  speed_erI = 1000;
if(speed_erI <= -1000)
  speed_erI = -1000;
```

图 5-2 位置式 PID 计算公式

理想的 PID 计算公式是在时域上连续的，而在数字控制系统中，信号量均为离散信号，故需将连续 PID 公式转换为如上图所示的数字离散式 PID 计算公式。从图 5-1 中可见，当 P 为 1 时，系统具有很大稳态误差，故需要增大 P 值来降低跟踪稳态误差，故可将 P 设置为 10，此时闭环调速系统的阶跃响应曲线如图 5-3 所示。

图 5-3 P=10 时阶跃响应曲线

从上图中可以看见，此时系统的跟踪稳态误差已经降低为 10%，速度响应曲线也较为平稳，故可以继续增大 P 值，在将 P 值增大为 15 时可以得到图 5-4 所示的阶跃响应曲线。

图 5-4 P=15 时阶跃响应曲线

从上图中可以发现，此时速度响应曲线已经发生了明显的震荡，这在实际速度控制系统

中是不可使用的，故需将 P 调回 10，而此时系统仍有 10%的跟踪稳态误差不能通过提高 P 来消除，故可通过加入 I 来进一步消除系统的跟踪稳态误差。将 P 适当增加一点至 11，I 为 2，此时系统的响应曲线如图 5-5 所示。

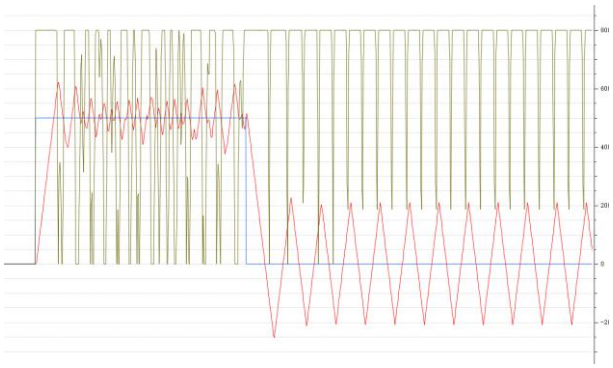


图 5-5 P=11 I=2 阶跃响应曲线

从上图中可以看到，红色的速度响应曲线在蓝色的期望速度周围发生了等幅震荡，而棕色的线为 PWM 占空比值，产生这样的现象是由于 I 值过大导致系统积分作用过强，使得系统发生震荡，故需要降低 I 值。将 P 调节为 9，I 为 0.8 可以得到如图 5-6 所示的阶跃响应曲线。

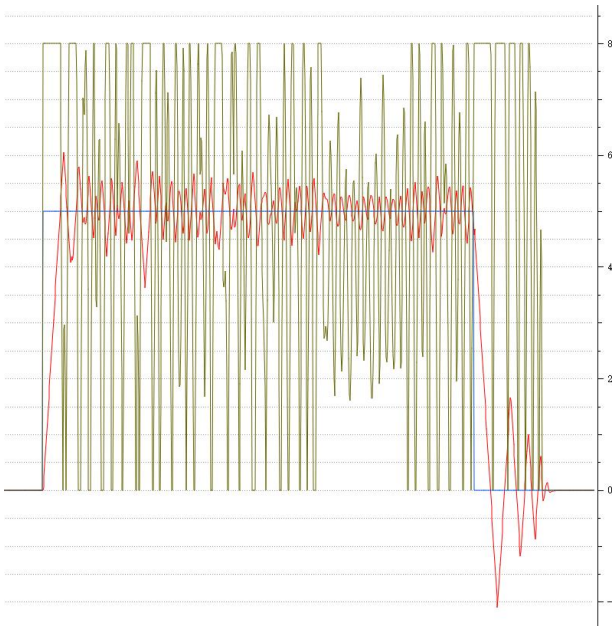


图 5-6 P=9 I=0.8 阶跃响应曲线

从红色的速度响应曲线中可以发现，系统能以一种近似为震荡的形式较为稳定的通过，此时电机运行也有很强烈的顿挫感，观察棕色的 PWM 占空比变化曲线可见其变化十分剧烈，此时可以通过加入 D 来降低这种变化。将 P 调节为 9，I 为 0.8，D 为 2.5 可以得到如图 5-7 所示的阶跃响应曲线。

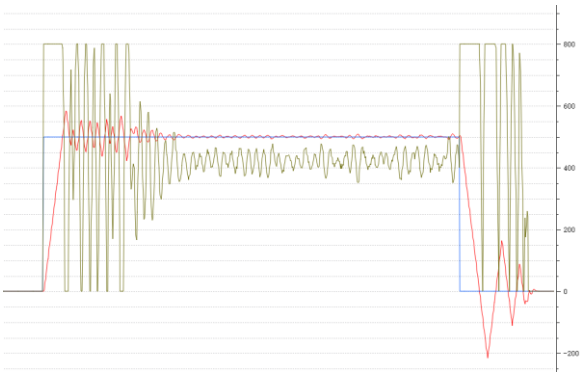


图 5-7 P=9 I=0.8 D=2.5 阶跃响应曲线

从上图中可以看到，加入 D 后速度值的震荡得到了有效的抑制，此时的速度响应曲线已经可以使用，但此时的超调量稍大，故将 P 降低为 8，同时为了兼顾快速性，将 D 增大至 3 可以得到如图 5-8 所示的阶跃响应曲线。

图 5-8 P=8 I=0.8 D=3 阶跃响应曲线

此时从曲线中可以看到，速度响应迅速，稳态误差低，但超调量达到 20%，但考虑到此时是在轮子悬空，电机空载时测量的，而车子自重较大，在实际运行中相当于加入了一个惯性较大的环节，故这些超调量可以得到抑制，车子使用本套 PID 参数在路面上进行运行时超调量得到了明显的降低，运行平稳，在将 PID 参数在本套 PID 参数值周围浮动时发现均没有本套参数理想，故可认为本套 PID 参数在当前运行环境下为较优参数值，参数整定工作至此结束。

第六章 越野电磁车主要参数

表 6.1

项目	参数
赛题组	室外越野电磁组
车模几何尺寸（长、宽、高）	150，30，25
传感器种类及个数	电感*5，霍尔元件*8，编码器*1，陀螺仪*1
主要集成电路种类/数量	最小核心板*1，电机驱动*2，放大电路*5

第七章 总结与评估

在一次偶然的机会下，我们组参与了室外越野电磁组的比赛。自报名参加第十四届“恩智浦”杯智能汽车竞赛以来，我们小组成员从查找资料、设计机构、组装车模、编写程序一步一步的进行，最后终于完成了最初目标。从最开始的直立调试，到之后的路径识别，以及提速等过程中，遇见了很多困难，也有想过放弃，不过最终在老师的厚望，以及队友的鼓励下，让我们坚持了下来。

智能车比赛是一个非常锻炼人的全国大学生竞赛，其不仅锻炼了我们的思维能力，动手能力，还对我们的学习能力、团队协作能力和组织实践能力有很大的提升。

在初始准备阶段，由于室外越野电磁组是今年的新赛题，我们小组可借鉴的资料比较少，只能重新设计电路板。在重新设计电路板的过程中我们遇到了各种各样的问题，但在队友的不懈努力下，我们最终解决了问题。

在调试的过程中，我们不断提高车速，紧接着变出现了各式各样的问题，比如说冲出赛道，进入圆环不稳定等等。可见，车子的速度与稳定性需要一定的取舍，并且在适当的取舍中不断调整，继续提高，达到更好的平衡。

当然，虽然我们在不断进步，但是我们仍然存在很多问题，比如说 PID 调试的经验的不足，导致一开始调试速度环花费了很多时间，并且还没有得到很好的结果。但是后来通过老师和学长的指导，做到了理想的效果。

在比赛之前的准备阶段，我们花费心思，不断努力，虽然还有很多问题没有解决，但是我们没有遗憾，因为人生本就应该不断地学习，在学习中不断地进步！

2019 年四川省赛四川大学参赛队员合影

附录 部分程序源码

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


#define Stop_Thres 5

int var[8] = 0;                         //存储电感值
int ad_min[8] = {11,12,33,46,33,14,13,14};
int ad_max[8] = {200,200,200,200,200,245,241,245};


////
int Motor1_duty=0;                  //电机正转 PWM1 占空比!!!!!占空比精度为 1000!!!  0.1%
int Motor2_duty=0;

////测速用的变量
int Encode_Speed = 0;
int L_Encode_Speed = 0;       //上一次的速度值
int actu_speed_ave[4] = 0;


int speed_goal = 0;

int mode = 0;     //模式
float normalP = 6;
float normalD = 6;
int ave_Speed =150;
int turnspeed = 70;
int turnThres = 35;
int turnThres_1 = 35;   //横断阈值

char flag = 0x01;     //任务标志位
int debug = 0;

int Outdata[8];   //虚拟示波器数组

int stopcount = 0;

//10ms
unsigned short    count = 0;
float dis;
char avoid_step = 0;     //过横断执行步骤
```

```
int    dajiao =30;

//100       50
//110     1.6    55

void main()
{
      DisableInterrupts;
      OLED_Init();     //OLED 初始化
      ADC_Init();   //ADC 初始化
      Motor_initial();        //电机舵机初始化
       KEY_Init();
      uart_init(UART4,115200);//   蓝牙串口初始化
      uart_rx_irq_en(UART4);
      set_vector_handler(UART4_RX_TX_VECTORn , uart4_test_handler);

       while(MPU_Init() == 1);
       MPU6050.check = 1;
      MPU6050.offset_gZ = 0.574;

       OLED_Init();
      gpio_init (PTA16, GPI,1);   //编码器正反转信号    PTA16
      lptmr_pulse_init(LPT0_ALT1,0xffff,LPT_Rising);       //LPTMR  编码器计数  PTA19
      lptmr_pulse_clean();     //lptmr 清零
      Ultrasonic_Init();
      Timer_Init();     //PIT0 初始化
      EnableInterrupts;
      mode = getMode();

       while(    (mode & 0x08) == 0x08    )     //归一化扫描
       {
            mode = getMode();
            var[0] = adc_ave(ADC1_SE8,    ADC_8bit,6);     //左
            var[1] = adc_ave(ADC0_SE13, ADC_8bit,6);    //中
            var[2] = adc_ave(ADC0_SE12, ADC_8bit,6);     //  右
            var[3]  =  adc_ave(ADC1_SE11,  ADC_8bit,6);       //       左转的电感          陀螺仪是正方向转
150 要过 90 度
            var[4] = adc_ave(ADC1_SE10, ADC_8bit,6);       //         右转的电感       陀螺仪是负方向转

            if (var[0]>ad_max[0])         //mode0  用于测量赛道中电感采样到的最大值和最小值
                  ad_max[0]=var[0];
            if (var[1]>ad_max[1])
```

```
            ad_max[1]=var[1];
        if (var[2]>ad_max[2])
            ad_max[2]=var[2];
        if (var[3]>ad_max[3])
            ad_max[3]=var[3];
        if (var[4]>ad_max[4])
            ad_max[4]=var[4];

        if (var[0]<ad_min[0])
            ad_min[0]=var[0];
        if (var[1]<ad_min[1])
            ad_min[1]=var[1];
        if (var[2]<ad_min[2])
            ad_min[2]=var[2];
        if (var[3]<ad_min[3])
            ad_min[3]=var[3];
        if (var[4]<ad_min[4])
            ad_min[4]=var[4];

        OLED_printf(0,0,"%d %d %d %d",var[0],ad_min[0],ad_max[0],var[4] );
        OLED_printf(0,2,"%d %d %d %d",var[1],ad_min[1],ad_max[1],ad_min[4] );
        OLED_printf(0,4,"%d %d %d %d",var[2],ad_min[2],ad_max[2],ad_max[4] );
        OLED_printf(0,6,"%d %d %d %d",var[3],ad_min[3],ad_max[3],mode);
}

OLED_Clear();

mode = getMode();


switch(mode)
{
case 1:
   ave_Speed = 110;
   shuaijian = 1.6;
   turnspeed = 50;
   normalP = 5;
  normalD = 9;
   turnThres = 25;
   turnThres_1 = 45;
   dajiao = 30;
   break;
case 2:
   ave_Speed = 110;
```

```
        shuaijian = 1.6;
        turnspeed = 45;
        normalP = 5;
       normalD = 8;
        turnThres = 30;
        turnThres_1 = 45;
        dajiao = 30;
        break;
    case 4:
        ave_Speed = 110;
        shuaijian = 1.5;
        turnspeed = 50;
        normalP = 5;
       normalD = 8;
        turnThres = 30;
        turnThres_1 = 45;
        dajiao = 22;
        break;
    }

    while(1)
   {


////////********** 1ms **************

            if( loopflag_5ms >= 1)
            {

                L_Encode_Speed = Encode_Speed;
                Encode_Speed = lptmr_pulse_get();    //获取计数值
                lptmr_pulse_clean();      //LPTMR 清零

                if(PTA16_IN == 0)           //反转时测回转速值反向
                    Encode_Speed = -Encode_Speed;
                if(Encode_Speed - L_Encode_Speed > 10)     //本次测量出的速度减去上一次测量出的速
度
                    Encode_Speed=L_Encode_Speed + 10;          //限幅滤波 防止毛刺   20 这个应该根
据采样周期设置
                else if(Encode_Speed - L_Encode_Speed < -10)
                    Encode_Speed=L_Encode_Speed - 10;

                actu_speed_ave[3] = actu_speed_ave[2];
                actu_speed_ave[2] = actu_speed_ave[1];
```

```
                actu_speed_ave[1] = actu_speed_ave[0];
                actu_speed_ave[0] = Encode_Speed;
                actu_speed_ave[0]                                                                    =
actu_speed_ave[0]*0.7+actu_speed_ave[1]*0.15+actu_speed_ave[2]*0.1+actu_speed_ave[3]*0.05;     //低通
滤波

                motorPID(actu_speed_ave[0],speed_goal);

                loopflag_5ms = 0;

        }

////////********** 10ms **************

        if(loopflag_10ms >= 10)
        {



                var[0] = adc_ave(ADC1_SE8,   ADC_8bit,6);    //左
                var[1] = adc_ave(ADC0_SE13,   ADC_8bit,6);   //中
                var[2] = adc_ave(ADC0_SE12, ADC_8bit,6);    // 右
                var[3] =  adc_ave(ADC1_SE11,  ADC_8bit,6);    //    左转的电感      陀螺仪是正方向转
150 要过 90 度
                var[4] = adc_ave(ADC1_SE10, ADC_8bit,6);     //      右转的电感     陀螺仪是负方向转


                if(var[0] <= ad_min[0] + 3)
                        var[0] = ad_min[0];
                if(var[1] <= ad_min[1] + 3)
                        var[1] = ad_min[1];
                if(var[2] <= ad_min[2] + 3)
                        var[2] = ad_min[2];
                if(var[3] <= ad_min[3] + 3)
                        var[3] = ad_min[3];
                if(var[4] <= ad_min[4] + 3)
                        var[4] = ad_min[4];

                ad_Difference1();



                if(ad_nomalize[4] >=   turnThres_1 && ad_nomalize[3] >= turnThres_1)
                        count = 0;
```

```
            else if( (flag & 0x10) != 0x10 && (ad_nomalize[4]-ad_nomalize[3] )>= 35)          //右转一
个 90 度的弯弯
            {
                flag |=   0x08;
                count = 0;
            }
            else if( (flag & 0x10) != 0x10 && (ad_nomalize[3]-ad_nomalize[4] )>= 35)      //左转一个 90
度的弯弯
            {
                flag |=   0x04;
                count = 0;
            }



            if((flag & 0x04) == 0x04)      //左转
            {
                if(stopcount <= 3000)
                    stopcount++;
                count++;
                if( count >= 30 &&   ad_nomalize[0] > 50 || ad_nomalize[1] > 51 || ad_nomalize[2] >27 )
//转弯了一个小弯弯   又检测到电磁线
                    flag &=   ~0x04;
                else if( count >= 500 && ad_nomalize[0] <= Stop_Thres && ad_nomalize[1] <=
Stop_Thres && ad_nomalize[2] <= Stop_Thres   )    //转了一个大弯弯后    还没有检测到电磁线   停车
                {
                    flag &=   ~0x04;
                    flag |=   0x02;
                }


                ftm_pwm_duty(FTM1, FTM_CH0, -dajiao + Mid_PID_Steer_Output ); //改变舵机 PWM
占空
                speed_goal = turnspeed;
            }
            else if( (flag & 0x08) == 0x08   )    //右转
            {
                if(stopcount <= 3000)
                    stopcount++;
                count++;
                if( count >= 30 &&   ad_nomalize[0] > 50 || ad_nomalize[1] > 51 || ad_nomalize[2] >27 )
//转弯了一个小弯弯   又检测到电磁线
                    flag &=   ~0x08;
                else if( count >= 500 && ad_nomalize[0] <= Stop_Thres && ad_nomalize[1] <=
```

```
Stop_Thres && ad_nomalize[2] <= Stop_Thres    )    //转了一个大弯弯后    还没有检测到电磁线   停车
                    {
                        flag &=   ~0x08;
                        flag |=   0x02;
                    }
                    ftm_pwm_duty(FTM1, FTM_CH0, dajiao + Mid_PID_Steer_Output ); //改变舵机 PWM 占
空
                    speed_goal = turnspeed;
                }
                else if( (flag & 0x01) == 0x01 || (flag & 0x02) == 0x02 || (flag & 0x10) == 0x10   )     //停车
                {

                    if(          ad_nomalize[0] <= Stop_Thres && ad_nomalize[1] <= Stop_Thres &&
ad_nomalize[2] <= Stop_Thres )       //常规超出电磁线 停车
                        flag |=   0x02;
                    else
                        flag &=   ~0x02;
                    ad_Difference1_control( );
                    speed_goal = 0;
                }
                else    //正常运行
                {
                    if(stopcount <= 3000)
                        stopcount++;
                    if(          ad_nomalize[0] <= Stop_Thres && ad_nomalize[1] <= Stop_Thres &&
ad_nomalize[2] <= Stop_Thres )       //常规超出电磁线 停车
                        flag |=   0x02;
                    else
                        flag &=   ~0x02;
                    speed_goal = ad_Difference1_control( );   //计算出期望速度 并执行舵机打角
                }



                Outdata[0] = ad_nomalize[0];
                Outdata[1] = ad_nomalize[1];
                Outdata[2] = ad_nomalize[2];
                Outdata[3] = ad_nomalize[3];
                Outdata[4] = ad_nomalize[4];
                Outdata[5] = Encode_Speed;
                Outdata[6] = speed_goal;
                Outdata[7] = MPU6050.angleZ;
```

```
                vcan_sendware((uint8_t *)Outdata,sizeof(Outdata));


                loopflag_10ms = 0;
            }

////////********** 20ms **************
            if(loopflag_20ms >= 20)
            {
                MPU6050_Get_Gyroscope(&MPU6050.gyroZ);//获取 MPU6050 陀螺仪数据
//                  MPU6050.angleZ+=0.020f*MPU6050.gyroZ;
                OLED_printf(0,0,"YAW:%.1f",MPU6050.angleZ);
                OLED_printf(0,2,"flag:%x",flag);
                loopflag_20ms = 0;
            }

////////********** 100ms **************
            if(loopflag_100ms >= 100)
            {
                int i = 0;
                i = KEY_Scan(0);
                switch(i)
                {
                    case 1:
                      //debug++;
                    flag = 0;
                    //speed_goal = 150;
                    break;
                    case 2:
                    // debug--;
                    flag = 0x01;
                    //speed_goal = 0;
                break;
                }


                OLED_printf(0,6,"0:%.1f 1:%.1f ",Wave0.Dis,Wave1.Dis);

                Ultrasonic_Wave_Measure();
                loopflag_100ms = 0;
            }
        }
}
```

```
#include "PID.h"



//速度环 PID 用的变量
int speed_actual=0;

int speed_er=0;
int speed_erL=0;
int speed_erI=0;

int speed_out=0;
int pout=0;

float   Kp=9;
float   Ki=1;
float   Kd=20;//11

int pulseFull = 960;   //PWM 脉冲满量值
int M_Speed = 0;       //pwm 占空比


int shuaijian = 0;


//电机速度环 PID
//Encode_Speed  实际速度
//speed_goal  期望速度
void motorPID(int Encode_Speed, int speed_goal)
{
   speed_actual=speed_actual*0.3+0.7*Encode_Speed;      //只取速度变化量的 0.7   消除毛刺？？？？
   speed_erL=speed_er;
   speed_er = (speed_goal-speed_actual);     //本次误差
   speed_erI += speed_er;

   speed_out = Kp*speed_er + Ki*(speed_erI) + Kd*(speed_er-speed_erL);     //增量式 PID 计算

   if(speed_erI >= 1000)
      speed_erI = 1000;
   if(speed_erI <= -1000)
      speed_erI = -1000;

   if(speed_out >= 160 )       //正转
   {
```

```
        M_Speed = speed_out; //    120 是死区
        if(M_Speed > pulseFull)
            M_Speed = pulseFull;
        ftm_pwm_duty(FTM0, FTM_CH3, M_Speed);
        ftm_pwm_duty(FTM0, FTM_CH4, 0);
    }
    else if(speed_out <= -160)   //反转
    {
        speed_out = -speed_out;
        M_Speed = speed_out;
        if(M_Speed > pulseFull)
            M_Speed = pulseFull;
        else if(M_Speed < 0)
            M_Speed = 0;
        ftm_pwm_duty(FTM0, FTM_CH3, 0);
        ftm_pwm_duty(FTM0, FTM_CH4, M_Speed);
    }
    else
    {
       M_Speed = 0;
       ftm_pwm_duty(FTM0, FTM_CH3, 0);
       ftm_pwm_duty(FTM0, FTM_CH4, 0);
    }

}




/****************电磁差和比以及舵机角度 目标车速计算**********************

****************电磁差和比以及舵机角度 目标车速计算**********************/
float ad_nomalize[8]={0};

double Angle_er=0;
double Angle_erL=0;
double Angle_erLL=0;

int Steer_out=0;

float p_dev=0;//330
float d_dev=0;//230

float ad_cal[3] = 0;
```

```
void ad_Difference1(void)   //计算打角值
{

    double delta = 0;

    //以下函数功能为动态归一化   归化为 1~200
    //这个 200 是因为过环岛的时候有两根电磁线 电感电压值会乘以二
    ad_nomalize[0] = 100.0*(var[0]-ad_min[0])/(ad_max[0]-ad_min[0]);   //AD7     A7
    if(ad_nomalize[0]<1.0)
    ad_nomalize[0]=1.0;
    if(ad_nomalize[0]>200)
    ad_nomalize[0]=200;

    ad_nomalize[1] = 100.0*(var[1]-ad_min[1])/(ad_max[1]-ad_min[1]);   //AD5    F4
    if(ad_nomalize[1]<1.0)
    ad_nomalize[1]=1.0;
    if(ad_nomalize[1]>200)
    ad_nomalize[1]=200
      ;
    ad_nomalize[2] = 100.0*(var[2]-ad_min[2])/(ad_max[2]-ad_min[2]);   //AD4    F5
    if(ad_nomalize[2]<=1.0)
    ad_nomalize[2]=1.0;
    if(ad_nomalize[2]>200)
    ad_nomalize[2]=200;

    ad_nomalize[3] = 100.0*(var[3]-ad_min[3])/(ad_max[3]-ad_min[3]);    //AD1    C0
    if(ad_nomalize[3]<=1.0)
    ad_nomalize[3]=1.0;
    if(ad_nomalize[3]>200)
    ad_nomalize[3]=200;

    ad_nomalize[4] = 100.0*(var[4]-ad_min[4])/(ad_max[4]-ad_min[4]);    //AD0    C1
    if(ad_nomalize[4]<=1.0)
    ad_nomalize[4]=1.0;
    if(ad_nomalize[4]>200)
    ad_nomalize[4]=200;

     delta = 1000 * (ad_nomalize[2]-ad_nomalize[0]) / (   (ad_nomalize[0]+ad_nomalize[1]+ad_nomalize[2])
*   ad_nomalize[1]);   //报告中的角度偏差计算公式   乘以 300 根据实际情况调节


    p_dev = normalP;
    d_dev = normalD;
```

```
    Angle_erL = Angle_er;
    Angle_er = (-delta);     //本次误差

    Steer_out = p_dev * ( Angle_er ) + d_dev*( Angle_er - Angle_erL);     //增量式 PID 计算


/*
          if(Steer_out > Max_PID_Steer_Output)              //输出限幅
        Steer_out = Max_PID_Steer_Output;
    if(Steer_out < Min_PID_Steer_Output)
        Steer_out = Min_PID_Steer_Output;

    ftm_pwm_duty(FTM1, FTM_CH0, Steer_out + Mid_PID_Steer_Output ); //改变舵机 PWM 占空比

*/


}

int ad_Difference1_control()
{
    int goal = 0;
     if(Steer_out > Max_PID_Steer_Output)              //输出限幅
        Steer_out = Max_PID_Steer_Output;
    if(Steer_out < Min_PID_Steer_Output)
        Steer_out = Min_PID_Steer_Output;



    ftm_pwm_duty(FTM1, FTM_CH0, Steer_out + Mid_PID_Steer_Output ); //改变舵机 PWM 占空比


    if(Steer_out >= 0)
        goal = ave_Speed - shuaijian *Steer_out;      //注意这里的平均速度一定要大于打脚的最大值!!!!!
    else
        goal = ave_Speed + shuaijian *Steer_out;

     OLED_printf(0,4,"st:%d",goal);

    return goal;      //返回目标速度

}
```

```
int angle_control(float angle_goal,float angle_actu)
{
    int goal = 0;
    p_dev = normalP;
    d_dev = normalD;

    Angle_erL = Angle_er;
    Angle_er = angle_actu - angle_goal;    //本次误差

    Steer_out = p_dev * ( Angle_er ) + d_dev*( Angle_er - Angle_erL);    //增量式 PID 计算



    if(Steer_out > Max_PID_Steer_Output)            //输出限幅
        Steer_out = Max_PID_Steer_Output;
    if(Steer_out < Min_PID_Steer_Output)
        Steer_out = Min_PID_Steer_Output;

    ftm_pwm_duty(FTM1, FTM_CH0, Steer_out + Mid_PID_Steer_Output ); //改变舵机 PWM 占空比



    if(Steer_out >= 0)
        goal = ave_Speed - 0.5*Steer_out;      //注意这里的平均速度一定要大于打脚的最大值!!!!!
    else
        goal = ave_Speed + 0.5*Steer_out;



    return goal;      //返回目标速度
}

void ADC_Init(void)
{
    adc_init(ADC1_SE8);        // PTB0     ad1
    adc_init(ADC0_SE13);        // PTB3     ad2
    adc_init(ADC0_SE12);       // PTB2      ad3
    adc_init(ADC1_SE11);       // PTB5      ad4
    adc_init(ADC1_SE10);       // PTB4      ad5
     adc_init(ADC1_SE12);       // PTB6      ad6
     adc_init(ADC1_SE13);       // PTB7      ad7
     adc_init(ADC0_SE14);       // PTC0      ad8
```

```
}

uint16 adc_mid(ADCn_Ch_e adcn_ch, ADC_nbit bit)
{
    uint16 i,j,k,tmp;
    //1.取 3 次 AD 转换结果
    i=adc_once(adcn_ch,bit);
    j=adc_once(adcn_ch,bit);
    k=adc_once(adcn_ch,bit);
    if(i>j)
    {
      tmp=i;i=j;j=tmp;
    }
    if(k>j)
      tmp=j;
    else if(k>i)
      tmp=k;
    else
      tmp=i;
    return tmp;
}

/*!
 *  @brief      AD 采样均值滤波
 *  @since      2019 年 1 月 19 日
 *  @note       1 月 19 日底层
 */
uint16 adc_ave(ADCn_Ch_e adcn_ch, ADC_nbit bit,uint8 N)
  {
    uint16 tmp=0;
    uint16 i;
    uint16 AD_value;
    for(i=0;i<N;i++)
    {
      tmp+=adc_mid(adcn_ch,bit);    //还需要加权值
    }
    tmp=tmp/N;
    AD_value=(uint16)tmp;
    return AD_value;
  }

/*
引脚说明
```

```
OLED 的 DC 接至 PTC1
        RES 接至 PTC2
使用的是 SPI1
SPI1_SCK_PIN       PTB11
SPI1_SOUT_PIN      PTB16
SPI1_PCS0_PIN      PTB10
*/


void OLED_DC(OLED_DC_Status status)
{
    gpio_set (PTC1, status);
}
void OLED_RES_PORT(OLED_RST_Status status)
{
    gpio_set (PTC2, status);
}

void delay_ms(unsigned int ms)
{
     unsigned int a;
     while(ms)
     {
          a=1800;
          while(a--);
          ms--;
     }
     return;
}

//向 SSD1306 写入一个字节。
//dat:要写入的数据/命令
//cmd:数据/命令标志  0,表示命令;1,表示数据;
void OLED_WR_Byte(uint8 dat,OLED_DC_Status cmd)
{
    uint8 buf[1];
    buf[0] = dat;
    OLED_DC(cmd);
    spi_mosi(SPI1,SPI_PCS0,buf,NULL,1);
}

void OLED_Set_Pos(unsigned char x, unsigned char y)
{
     OLED_WR_Byte(0xb0+y,OLED_CMD);
```

```
        OLED_WR_Byte(((x&0xf0)>>4)|0x10,OLED_CMD);
        OLED_WR_Byte((x&0x0f)|0x01,OLED_CMD);
}
//开启 OLED 显示
void OLED_Display_On(void)
{
        OLED_WR_Byte(0X8D,OLED_CMD);    //SET DCDC 命令
        OLED_WR_Byte(0X14,OLED_CMD);    //DCDC ON
        OLED_WR_Byte(0XAF,OLED_CMD);    //DISPLAY ON
}
//关闭 OLED 显示
void OLED_Display_Off(void)
{
        OLED_WR_Byte(0X8D,OLED_CMD);    //SET DCDC 命令
        OLED_WR_Byte(0X10,OLED_CMD);    //DCDC OFF
        OLED_WR_Byte(0XAE,OLED_CMD);    //DISPLAY OFF
}
//清屏函数,清完屏,整个屏幕是黑色的!和没点亮一样!!!
void OLED_Clear(void)
{
        u8 i,n;
        for(i=0;i<8;i++)
        {
                OLED_WR_Byte (0xb0+i,OLED_CMD);       //设置页地址（0~7）
                OLED_WR_Byte (0x00,OLED_CMD);         //设置显示位置—列低地址
                OLED_WR_Byte (0x10,OLED_CMD);         //设置显示位置—列高地址
                for(n=0;n<128;n++)OLED_WR_Byte(0,OLED_DATA);
        } //更新显示
}


//在指定位置显示一个字符,包括部分字符
//x:0~127
//y:0~63
//mode:0,反白显示;1,正常显示
//size:选择字体  16/12
void OLED_ShowChar(u8 x,u8 y,u8 chr)
{
      unsigned char c=0,i=0;
      c=chr-' ';//得到偏移后的值
      if(x>Max_Column-1){x=0;y=y+2;}
      if(SIZE ==16)
      {
            OLED_Set_Pos(x,y);
```

```
            for(i=0;i<8;i++)
            OLED_WR_Byte(F8X16[c*16+i],OLED_DATA);
            OLED_Set_Pos(x,y+1);
            for(i=0;i<8;i++)
            OLED_WR_Byte(F8X16[c*16+i+8],OLED_DATA);
      }
}
//m^n 函数
u32 oled_pow(u8 m,u8 n)
{
        u32 result=1;
        while(n--)result*=m;
        return result;
}

//显示一个字符号串
void OLED_ShowString(u8 x,u8 y,u8 *chr)
{
        unsigned char j=0;
        while (chr[j]!='\0')
        {               OLED_ShowChar(x,y,chr[j]);
                        x+=8;
                if(x>120){x=0;y+=2;}
                        j++;
        }
}

/***********功能描述：显示显示 BMP 图片 128×64 起始点坐标(x,y),x 的范围 0～127，y 为页的范围 0～
7*****************/
void OLED_DrawBMP(unsigned char x0, unsigned char y0,unsigned char x1, unsigned char y1,unsigned char
BMP[])
{
  unsigned int j=0;
  unsigned char x,y;

    if(y1%8==0) y=y1/8;
    else y=y1/8+1;
        for(y=y0;y<y1;y++)
        {
                OLED_Set_Pos(x0,y);
       for(x=x0;x<x1;x++)
              {
               OLED_WR_Byte(BMP[j++],OLED_DATA);
              }
```

```
    }
}


//初始化 SSD1306
void OLED_Init(void)
{
    uint32 baud;
    gpio_init(PTC1, GPO, 0);   //初始化 OLED_DC 的引脚
    gpio_init(PTC2, GPO, 0);   //初始化 OLED_RST 的引脚
    baud = spi_init(SPI1, SPI_PCS0, MASTER, 4*1000*1000);  //初始化 SPI,选择 CS0,主机模式, 波特率为 500kpbs ,返回真实波特率到 baud 变量

    OLED_RES_PORT(OLED_SET);
    delay_ms(100);
    OLED_RES_PORT(OLED_RST);
    delay_ms(100);
    OLED_RES_PORT(OLED_SET);
    delay_ms(100);

    OLED_WR_Byte(0xAE,OLED_CMD);//--turn off oled panel
    OLED_WR_Byte(0x00,OLED_CMD);//---set low column address
    OLED_WR_Byte(0x10,OLED_CMD);//---set high column address
    OLED_WR_Byte(0x40,OLED_CMD);//--set start line address  Set Mapping RAM Display Start Line (0x00~0x3F)
    OLED_WR_Byte(0x81,OLED_CMD);//--set contrast control register
    OLED_WR_Byte(0xCF,OLED_CMD); // Set SEG Output Current Brightness
    OLED_WR_Byte(0xA1,OLED_CMD);//--Set SEG/Column Mapping     0xa0 左右反置 0xa1 正常
    OLED_WR_Byte(0xC8,OLED_CMD);//Set COM/Row Scan Direction   0xc0 上下反置 0xc8 正常
    OLED_WR_Byte(0xA6,OLED_CMD);//--set normal display
    OLED_WR_Byte(0xA8,OLED_CMD);//--set multiplex ratio(1 to 64)
    OLED_WR_Byte(0x3f,OLED_CMD);//--1/64 duty
    OLED_WR_Byte(0xD3,OLED_CMD);//-set display offset Shift Mapping RAM Counter (0x00~0x3F)
    OLED_WR_Byte(0x00,OLED_CMD);//-not offset
    OLED_WR_Byte(0xd5,OLED_CMD);//--set display clock divide ratio/oscillator frequency
    OLED_WR_Byte(0x80,OLED_CMD);//--set divide ratio, Set Clock as 100 Frames/Sec
    OLED_WR_Byte(0xD9,OLED_CMD);//--set pre-charge period
    OLED_WR_Byte(0xF1,OLED_CMD);//Set Pre-Charge as 15 Clocks & Discharge as 1 Clock
    OLED_WR_Byte(0xDA,OLED_CMD);//--set com pins hardware configuration
    OLED_WR_Byte(0x12,OLED_CMD);
    OLED_WR_Byte(0xDB,OLED_CMD);//--set vcomh
    OLED_WR_Byte(0x40,OLED_CMD);//Set VCOM Deselect Level
    OLED_WR_Byte(0x20,OLED_CMD);//-Set Page Addressing Mode (0x00/0x01/0x02)
    OLED_WR_Byte(0x02,OLED_CMD);//
```

```
    OLED_WR_Byte(0x8D,OLED_CMD);//--set Charge Pump enable/disable
    OLED_WR_Byte(0x14,OLED_CMD);//--set(0x10) disable
    OLED_WR_Byte(0xA4,OLED_CMD);// Disable Entire Display On (0xa4/0xa5)
    OLED_WR_Byte(0xA6,OLED_CMD);// Disable Inverse Display On (0xa6/a7)
    OLED_WR_Byte(0xAF,OLED_CMD);//--turn on oled panel

    OLED_WR_Byte(0xAF,OLED_CMD); /*display ON*/
    OLED_Clear();
    OLED_Set_Pos(0,0);
}

signed int OLED_printf(u8 x,u8 y,const char *pFormat, ...)
{
     u8 i = 0;
    unsigned char pStr[16] = {'\0'};
    va_list ap;
    signed int result;

    // Forward call to vprintf
    va_start(ap, pFormat);
    result = vsprintf((char *)pStr, pFormat, ap);
    va_end(ap);

     for(i=0;i<16;i++)
    {
        if(pStr[i]=='\0')break;
    }
    for(i=i;i<16;i++)
    {
        pStr[i]=' ';
    }
     pStr[15]='\0';

    OLED_ShowString(x,y,pStr);

    return result;
}
```