第十六届全国大学生
智能汽车竞赛

技 术 报 告

学　　校：扬州大学

队伍名称：这个电磁不懂 AI

参赛队员：薛鹏

牛恺锐 张驰

带队教师：张正华

陈万培

关于技术报告和研究论文使用授权的说明

本人完全了解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。

参赛队员签字：

薛鹏 张驰 韩德亮

指导教师签字：

张洋 [illegible]

日期：2021 年 8 月 12 日

摘 要

本文设计的智能车系统以 TC377 微控制器为主控，通过电磁传感器检测赛道信息并进行赛道识别，采用干簧管检测起跑线完成停车任务，用神经网络拟合参数，结合 PID 控制算法，实现了对车模运动的速度控制和方向控制。利用无线串口、上位机、SD 卡模块、以及 TFT 屏幕等对程序进行调试，以获得更好的控制效果。

目录

第一章 绪论

1.1 研究背景

智能车是一种高新技术密集型的新型汽车，以汽车电子为背景涵盖控制、模式识别、传感技术、电子、电气、计算机、机械等多科学的科技创意性设计，一般主要由路径识别、速度采集、角度控制及车速控制等模块组成。可实现自动驾驶、自动变速及自动识别道路等功能，是现代电子产业发展中一项重要的组成部分。

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，涵盖多个学科，从一定程度上反映了当代大学生综合运用所学知识和探索创新的精神。同时该赛事是教育部高等教育司委托(教高司函[2005]201 号文)，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)主办的全国性、多学科交叉、趣味性、创新性赛事，旨在加强大学生实践与团队合作精神，促进高等教育改革。竞赛规则透明，评价客观标准，坚持公开、公平、公正的原则，从而保持了竞赛的健康、普及、持续的发展。

1.2 方案总体介绍

比赛赛道不限场地，铺设有通上 20k Hz 交流信号的电磁线，并且设有路障、草皮、砂石等越野元素。

根据比赛的相关规定，本智能车系统采用组委会统一提供的 L 车模，车模采用 Infineon 的 TC377 芯片作为控制器，通过长前瞻电感寻迹、采集短前瞻数据，并离线训练神经网络，然后将神经网络模型用 NNCU 转换工具部署到单片机上，利用短前瞻电感的数据，推理出小车的舵机角度。通过编码器来检测车速，并通过单片机上的正交解码功能获得速度和路程，采用 PI 控制算法驱动电机，通过 PWM 调整电机功率，由神经网络输出的角度和角度的变化对车速进行综合控制。

根据比赛要求，我们设计的系统总体方案如下图：

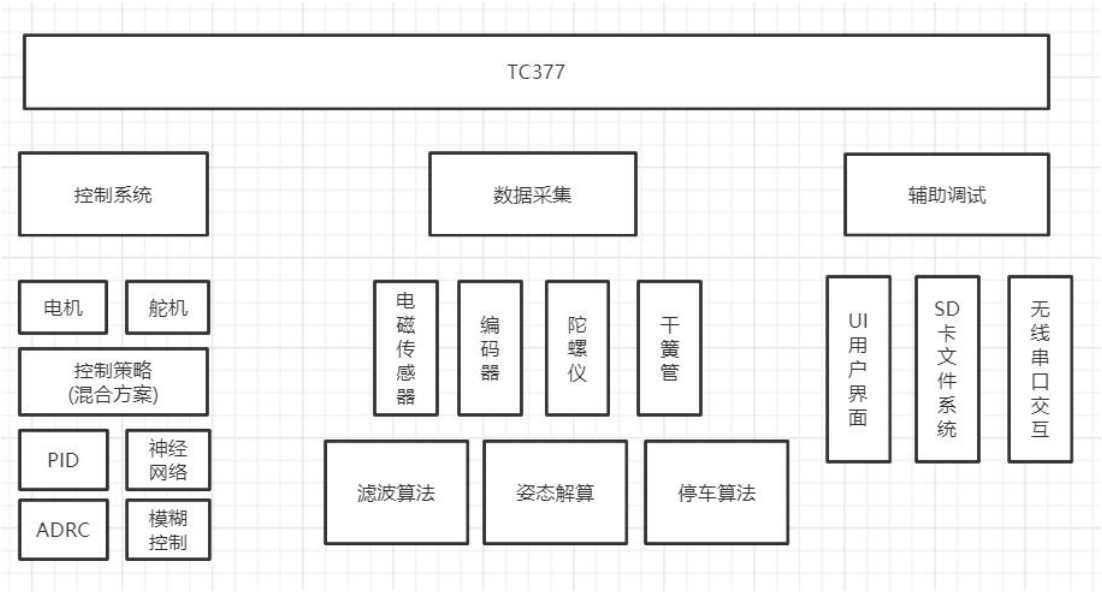


图 1.2.1 智能车系统框图

程序流程图如下：

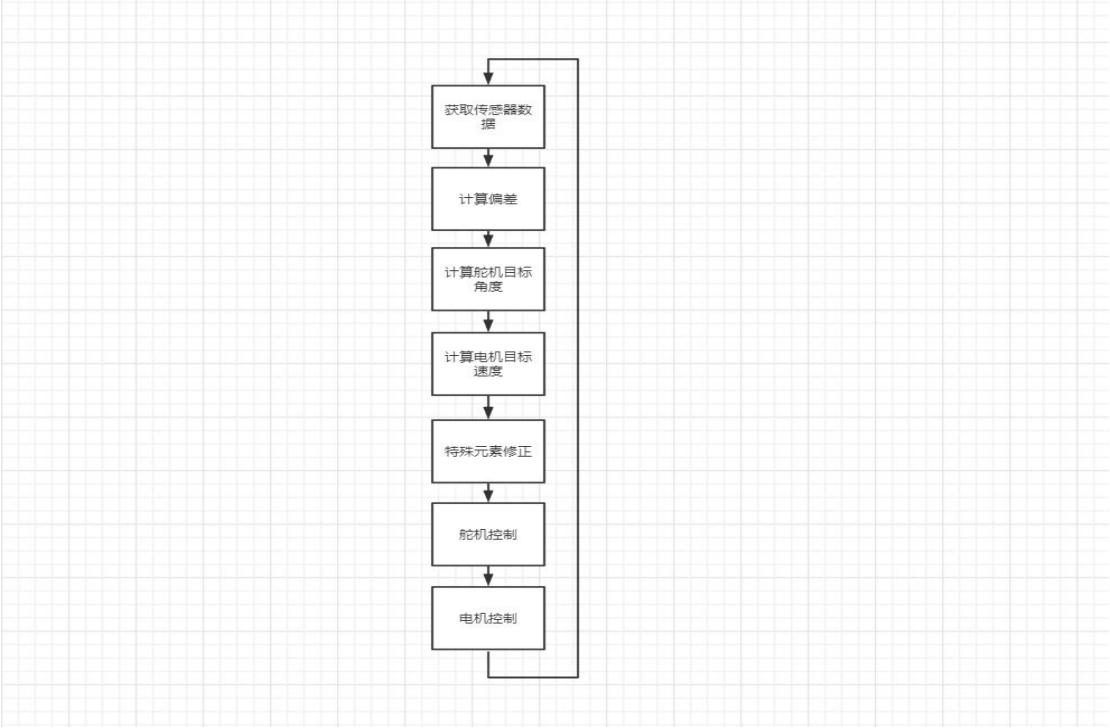


图 1.2.2 程序流程图

第二章 机械结构的设计及调整

机械设计中，我们需要考虑转向灵敏度、电感的保护、电路板放置、以及车身强度等问题。

2.1 转向结构设计

舵机转向控制系统是整个系统中延迟较大、控制精度较低的环节。为了减小延迟、增大控制精度，我们尽可能的在智能车走直线时，舵机的力臂与推拉杆垂直，这样可以尽可能的增大力臂长度，减小推拉杆的旷量对舵机控制的影响。我们也使用带塑料球头扣的推拉杆以减小旷量。

我们向前顶了约 1 厘米的车头，并磨去了一部分前车轮的固定件以增大转向的角度。

2.2 电路板安放

为了便于调试，我们把电路板放在车体的中心位置，并使用铜柱加螺丝固定。而铜柱则固定在中间塑料件的洞里，并辅以热熔胶加固。

2.3 传感器安装

电磁传感器用热熔胶固定在碳素杆上形成长前瞻，长前瞻使用约 50 厘米的碳素杆从车架的尾部架到车前约 30 厘米处，并使用三通加铜柱固定在车模的底板上。

电磁传感器固定在电路板上形成短前瞻，短前瞻使用 3D 打印件进行固定，今年的规则对于车模的长度进行了严格的限制，我们在规则允许的范围内把短前瞻尽可能的伸长。

同时，由于短前瞻上电磁传感器的位置微小变动都会使神经网络的控制效果变差，所以我们在焊接的同时使用热熔胶辅以加固。

2.4 车身强度

我们在车身强度明显不足的地方试图使用热熔胶加固。车子前端装有塑料泡沫防撞。

2.5 轮胎处理

我们使 4 个轮子都微微内倾，使车模稳定运行。我们适当的软化轮胎，增大了车模与地面的摩擦力。

第三章 硬件电路设计

良好、稳定的硬件电路是保证车子能快速、平稳运行的一个重要因素。我们在设计的过程中本着可靠性高、简洁高效的原则，在满足需求的情况下，尽量使得电路简单、整体效果简洁。

3.1 电源管理模块

我们的硬件电路采用一块锂电池(额定电压 7.4V，满电 8.4V)供电。我们使用 TPS7350 提供 5 V 电压。

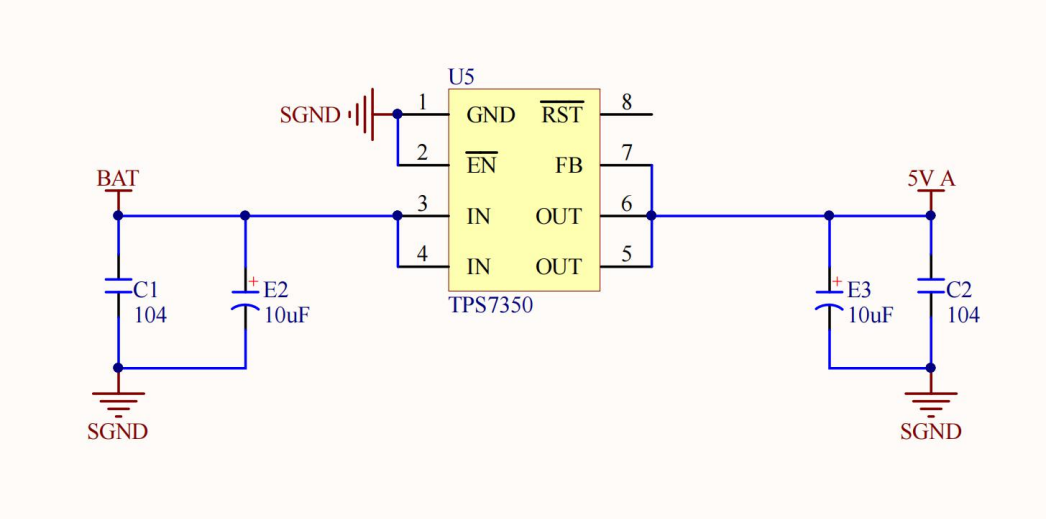


图 3.1.1 5V 稳压电路

使用 MP1584 得到 3.3 V 电压，完成对单片机以及其他外设的供电。

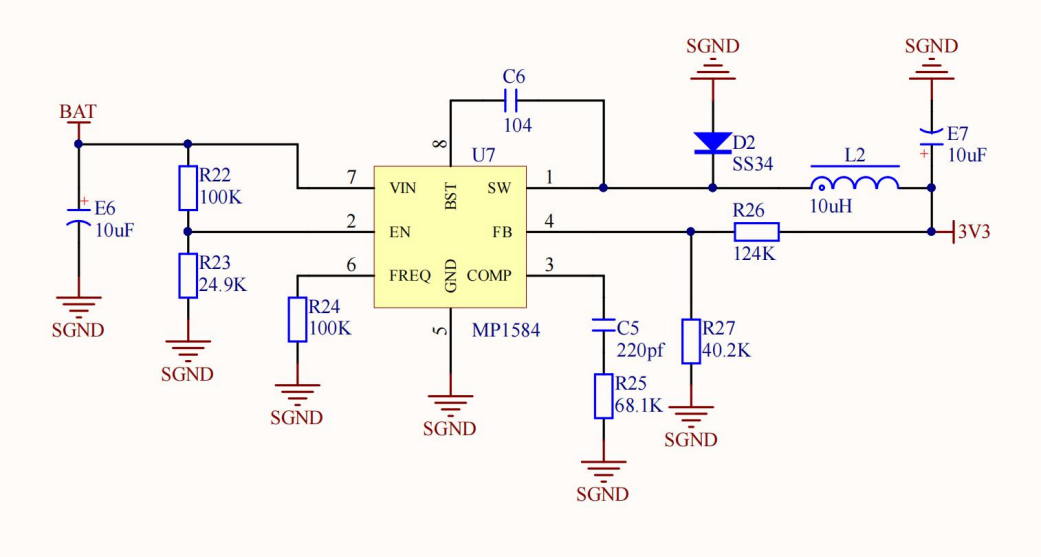


图 3.1.2 3.3V 稳压电路

使用 SPX29302 提供 6 V 电压，完成舵机供电。

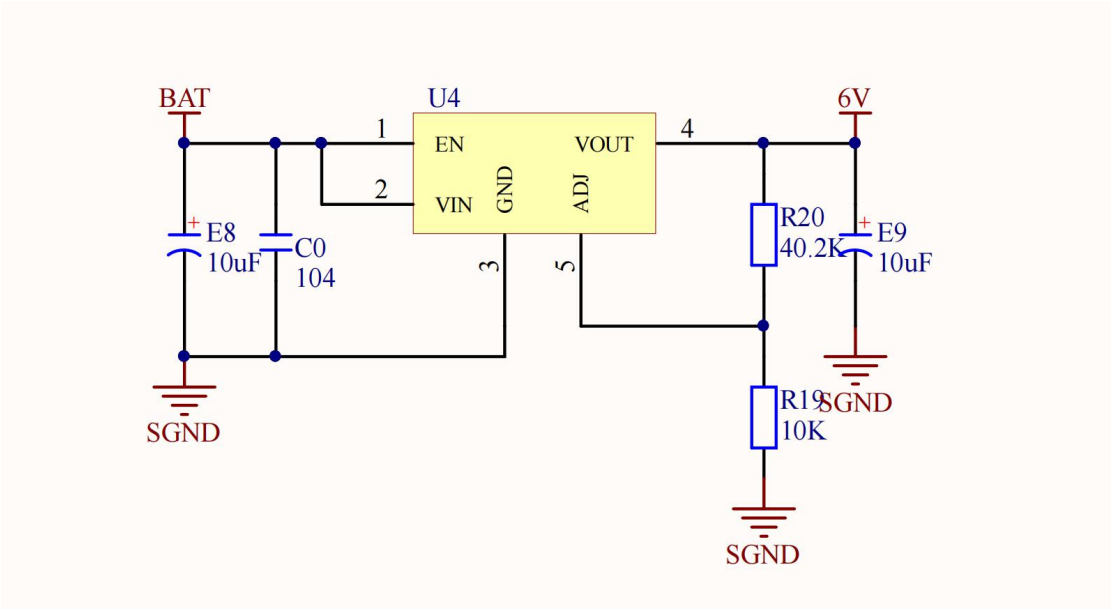


图 3.1.3 舵机供电电路

3.2 电机驱动模块

在驱动芯片的选择方面，我们采用 DRV8701 作为驱动芯片，驱动两个 N 沟道 MOS 管，能够提供较大的驱动电流。

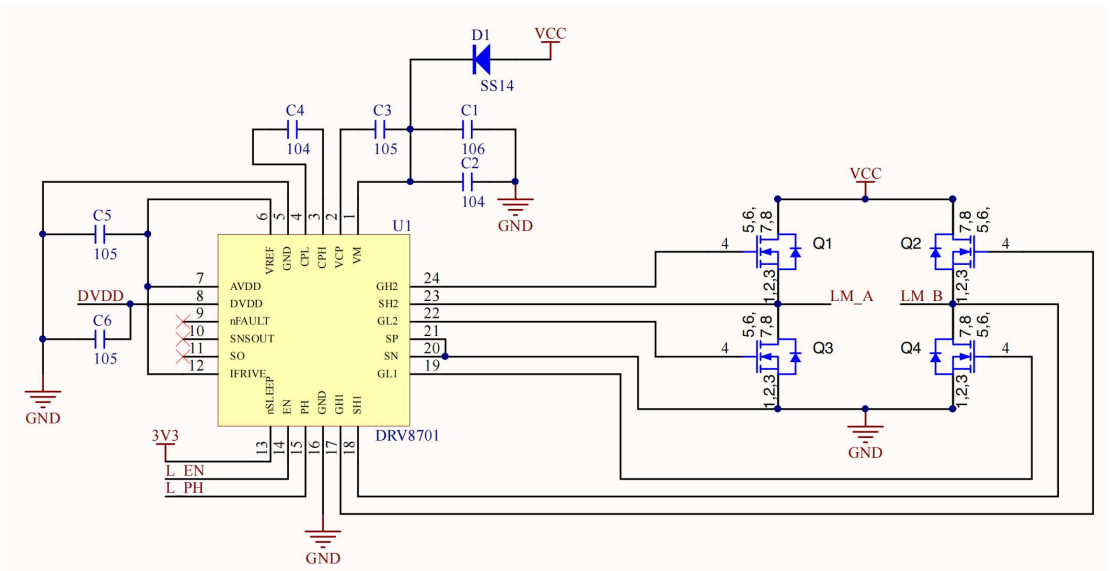


图 3.2.1 电机驱动电路

3.3 舵机模块

我们采用 SPX29302 为舵机提供 6V 电压，使用 74HC244 作为缓冲芯片。

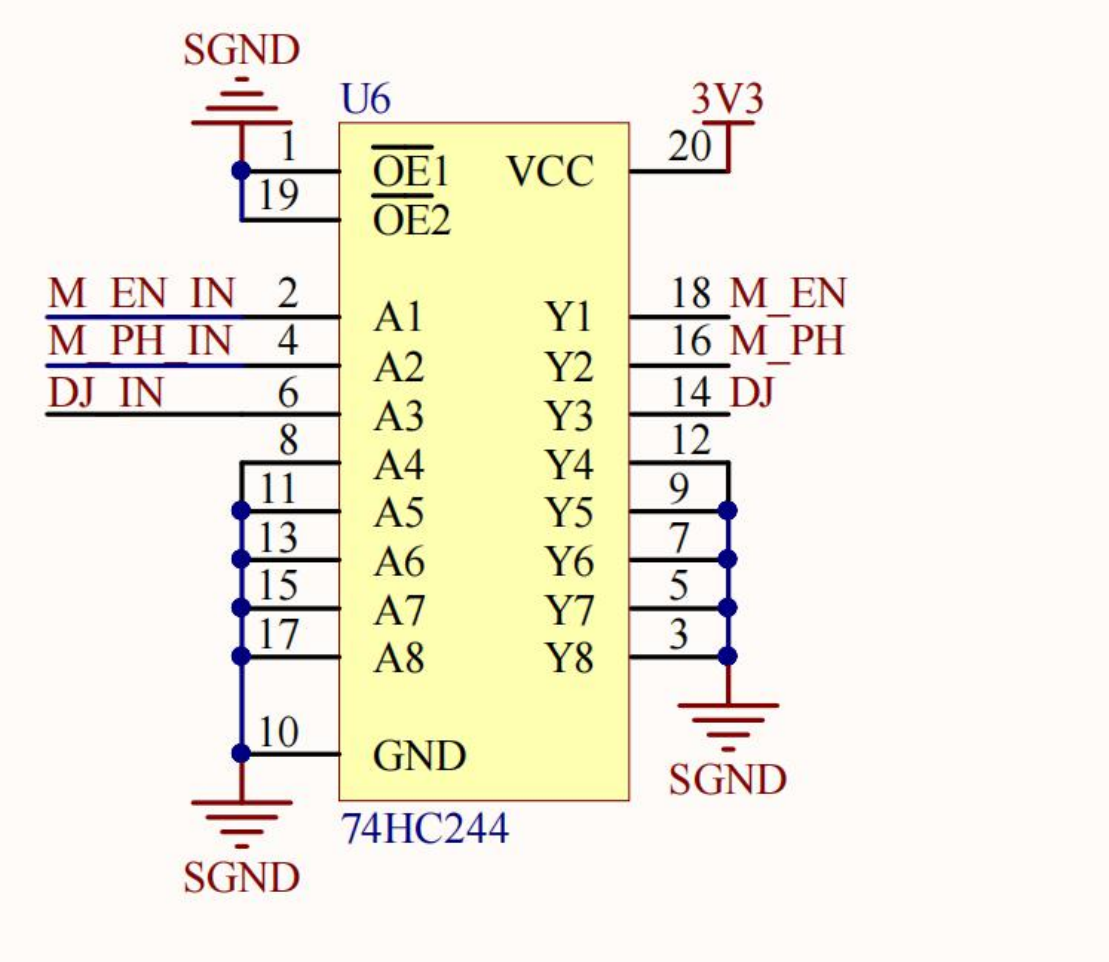


图 3.3.1 舵机缓冲器电路

3.4 电磁传感器模块

电磁传感器是电磁组最重要的模块之一，能够灵敏地检测出磁场信号的变化，对车子在赛道上的位置计算以及控制都起到了很大的作用。本模块将电磁信号进行滤波、放大、检波后，连接到单片机的 ADC 引脚进行读取并处理。放大电路如图所示：

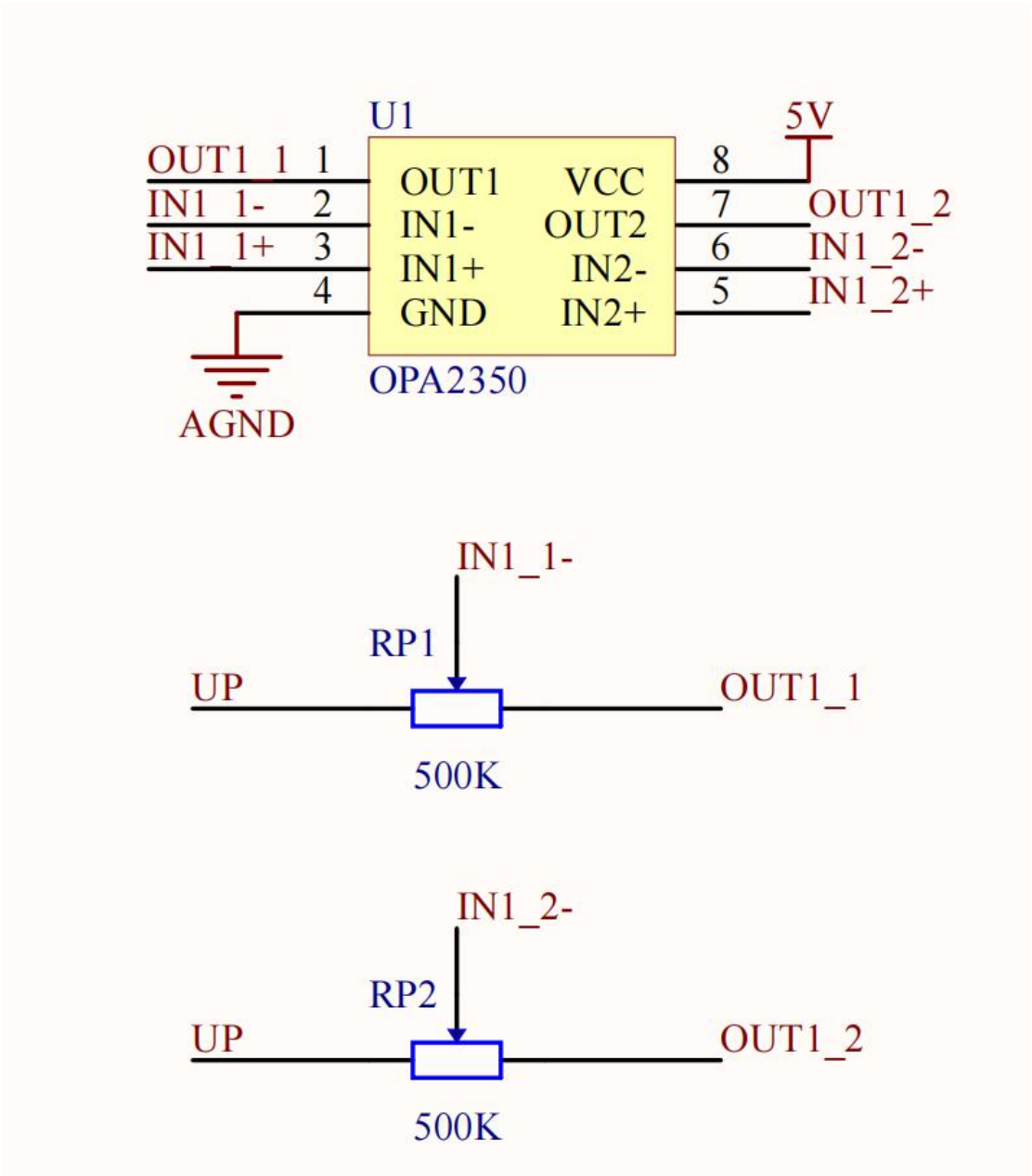


图 3.4.1 运放电路

3.5 编码器模块

我们采用龙邱科技的 1024 线编码器对小车进行测速，工作电压为 3.3V-5V。通过单片机的正交解码功能，来实现小车的测速。

3.6 陀螺仪模块

我们采用 MPU9250 姿态传感器，配合姿态解算算法以及卡尔曼滤波算法，准确得出当前的姿态，用于小车的元素处理中。

3.7 停车模块

我们采用干簧管进行停车检测，当小车靠近起跑线附近时，干簧管会闭合，从而可以通过检测单片机引脚电平的方式来停车。

3.8 人机交互模块

为了方便调试，我们装有无限串口模块，以便快速、方便的传输数据。使用键盘、TFT 显示屏等模块查看以及修改参数，方便调试。

此外，我们使用 SD 卡模块用于数据采集。

第四章 传统算法设计

我们采用传统的控制算法，基于长前瞻电感电压的数据，完成对舵机转向和电机转速的控制，保证车子能够平稳运行。并且，我们对此状态下小车的运动数据进行采集，用于搭建神经网络训练所需的数据集。

4.1 传感器数据采集

4.1.1 电感排布

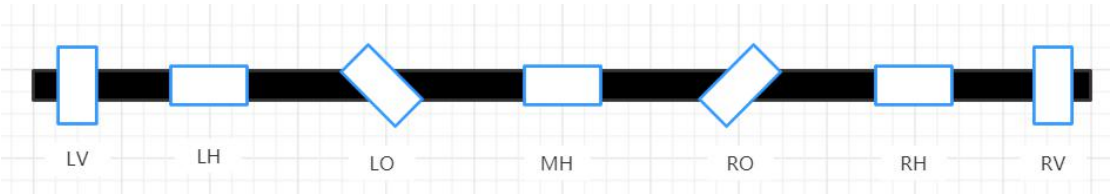


图 4.1.1 长前瞻电感分布图

从左到右依次命名为:

LV,LH,LO,MH,RO,RH,RV

4.1.2 数据处理

我们对采集到的数据进行滤波并归一化处理。采用滑动均值滤波的方法对电感数据进行滤波，取得了较好的滤波效果。

4.2 舵机转向控制

4.2.1 中线偏差的计算

对于中线偏差，我们采用如下方法计算得到：

$$Bias = \frac{A \times vDifference + B \times hDifference}{hSum}$$

其中，A、B 为系数，需要根据实际情况调整。

vDifference = LV - RV，为两侧竖电感的差值。

hDifference = LH - RH，为两侧横电感的差值。

hSum = LH + RH，为两侧横电感的和。

4.2.2 舵机转向的控制策略

对于舵机的转向控制，由于电磁越野的弯道基本都呈折线状，对控制精度的要求不是很高，为了达到更快的响应速度以及更好的稳定性，我们采用模糊控制作为舵机转向的控制策略，取得了较好的控制效果。

模糊控制器主要分为三个模块，模糊化、模糊推理、以及清晰化，如下图所示：

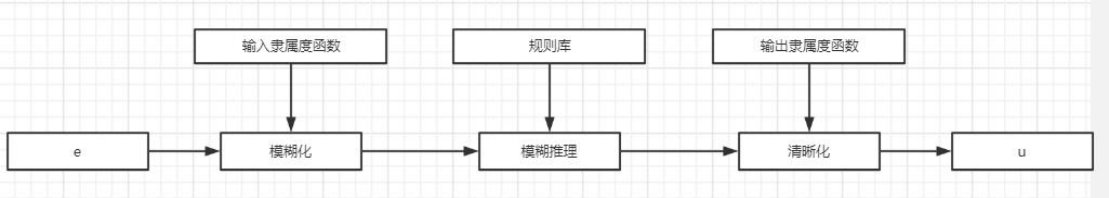


图 4.2.1 模糊控制流程

我们将偏差以及偏差的变化作为输入，以角度作为输出，建立模糊规则。主要分为以下三个过程：

(1) 模糊化

我们采用三角形隶属度函数对误差和误差变化分别进行模糊化。得到隶度关系。

特征点：{-75.0,-50.0,-25.0,0.0,25.0,50.0,75.0}

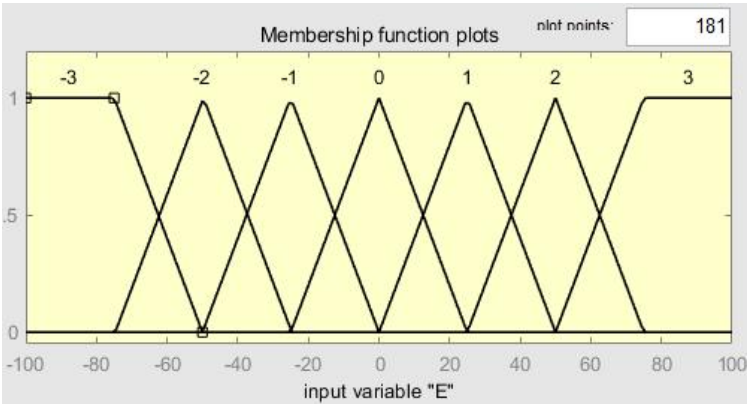


图 4.2.2 输入 E(偏差)的隶属度函数

特征点：{-20.0,-10.0,-5.0,0.0,5.0,10.0,20.0}

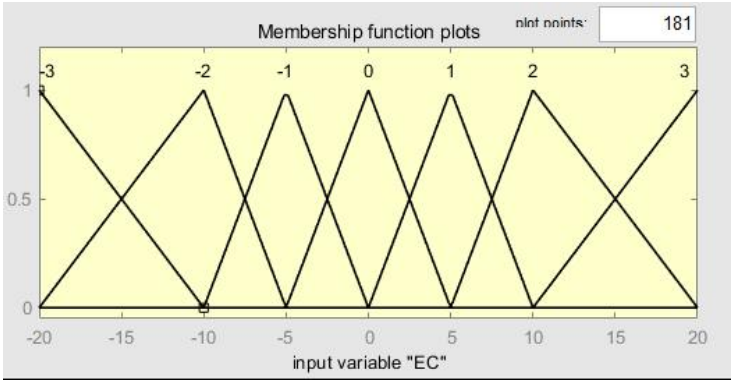


图 4.2.3 输入 EC(偏差变化)的隶属度函数

特征点：{-1.0,-0.8,-0.6,-0.32,-0.18,-0.08,0.0,0.08,0.18,0.32,0.6,0.8,1.0}

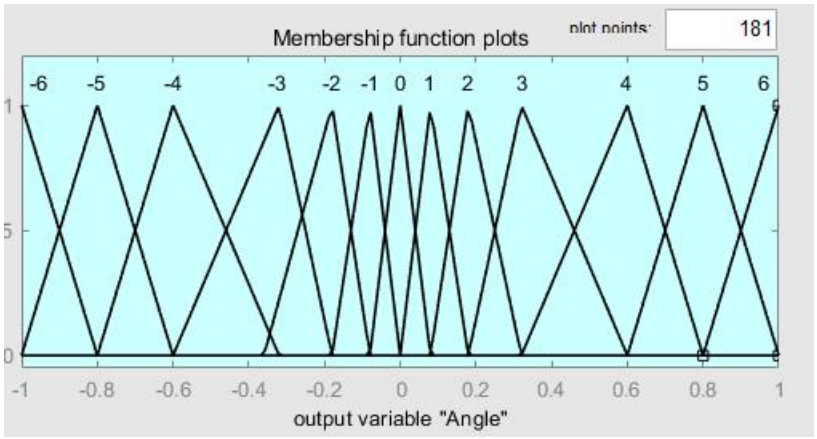


图 4.2.4 输出 Angle 的隶属度函数

(2) 模糊推理

根据输入的 E 和 EC，我们可以推断出它们各自的隶属度，然后根据模 糊规则表找到输出值所对应的隶属度。我们采用的模糊规则表如下

Angle		EC						
		-3	-2	-1	0	1	2	3
E	-3	-6	-6	-6	-6	-4	-4	-3
	-2	-6	-5	-5	-4	-3	-2	-1
	-1	-5	-4	-3	-1	0	1	2
	0	-4	-3	-1	0	1	3	4
	1	-2	-1	0	1	3	4	5
	2	1	2	3	4	5	5	6
	3	3	4	4	6	6	6	6

表 4.2.1 模糊规则表

(3) 清晰化

我们采用重心法进行反模糊（清晰化），得到归一化后的角度输出。

$$z_0 = \frac{\sum_{i=0}^{n} u_c(z_i) \times z_i}{\sum_{i=0}^{n} u_c(z_i)}$$

4.3 电机控制

4.3.1 电机速度控制

对应速度闭环控制我们采用增量式 PI 控制算法。PID 控制是工程中实际应用最为广泛的控制方法。其结构简单、稳定性好，因此成为工业控制的主要方法之一。其原理框图如下：

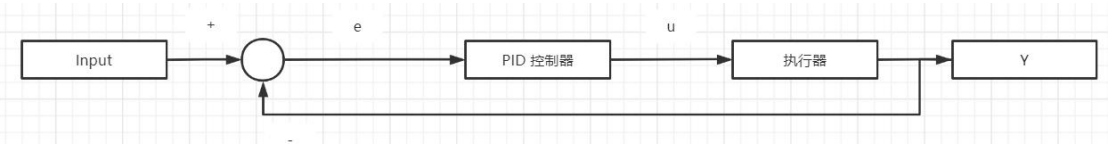


图 4.3.1 PID 闭环控制框图

e 代表目标输入与实际输出的误差，这个误差被送到 PID 控制器，控制器

算出输出量 u：

$$u = k_p \times e + k_i \times \int edt + k_d \times \frac{de}{dt}$$

而在数字系统中，其误差是经过采样、量化后的数字量，PID 控制器采用离散形式：

$$u_k = k_p \times (e_n + \frac{1}{T_i} \times \sum_{k=1}^{n} e_k T + Td \times \frac{e_n - e_{n-1}}{T})$$

在电机控制中，常采用增量式 PID 控制：

$$\Delta u_k = k_p \times (e_n - e_{n-1}) + k_i \times e_n + k_d \times (e_n - 2e_{n-1} + e_{n-2})$$

其中，Kp、Ki、Kd 为三个待调的参数，分别称为比例系数、积分系数、微分系数。

在代码实现中，可以写成：

```
float PID_Inc = 0.0;

PID_CtrlStr->PID_Error[0] = PID_CtrlStr->PID_Error[1];
PID_CtrlStr->PID_Error[1] = PID_CtrlStr->PID_Error[2];
PID_CtrlStr->PID_Error[2] = (TargetValue - ActualValue);

PID_Inc = PID_CtrlStr->Kp * (PID_CtrlStr->PID_Error[2] - PID_CtrlStr->PID_Error[1])+\
                  PID_CtrlStr->Ki * (PID_CtrlStr->PID_Error[2])+\
                  PID_CtrlStr->Kd * (PID_CtrlStr->PID_Error[2] - 2*PID_CtrlStr->PID_Error[1]+PID_CtrlStr->PID_Error[0]);

PID_CtrlStr->Result += PID_Inc;
```

除了使用 PI 控制算法外，我们还尝试了使用 ADRC 控制算法，来完成对电机的闭环控制，以达到快速响应并且无超调的效果。但由于时间限制，未能将该控制器调到最优状态，因此仍然采用 PI 算法。

4.3.2 PID 参数的调节

对于电机 PID 的三个参数，我们采用 Matlab 进行调节，找到一套较好的参数。

首先，给电机一个阶跃信号作为输入，用编码器记录下电机的响应，然后将输入和输出波形放入到 Matlab 的系统辨识工具箱中，辨识出系统的传递函数：

$$\frac{0.08588\ s + 1.496}{s^2 + 8.797\ s + 26.64}$$

辨识系统的阶跃响应和电机的阶跃响应对比如下图所示：

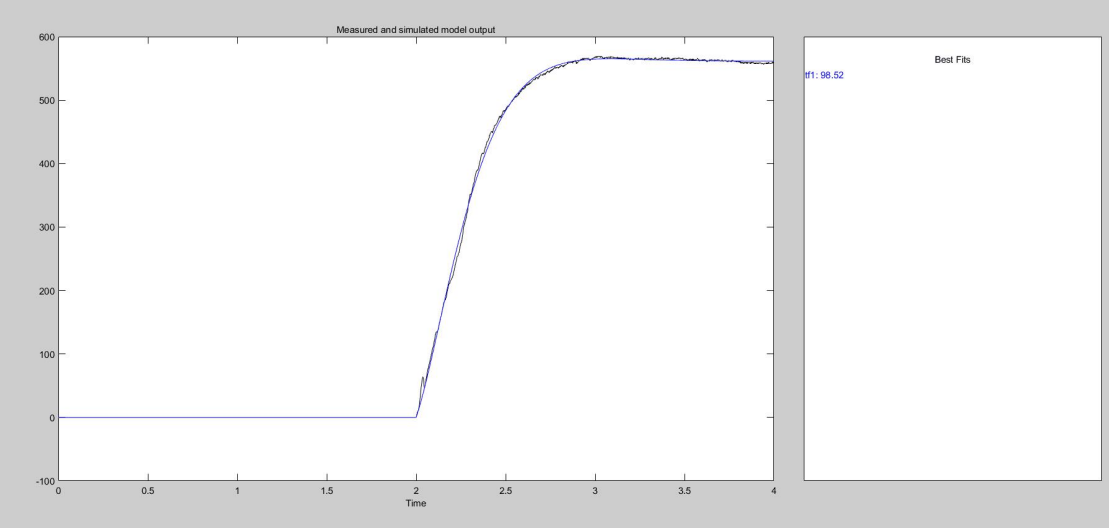


图 4.3.2 辨识系统函数的阶跃响应与电机的阶跃响应对比

可以看到，辨识结果和电机的阶跃响应基本一致。

然后在 Simulink 中搭建控制模型，进行参数调节：

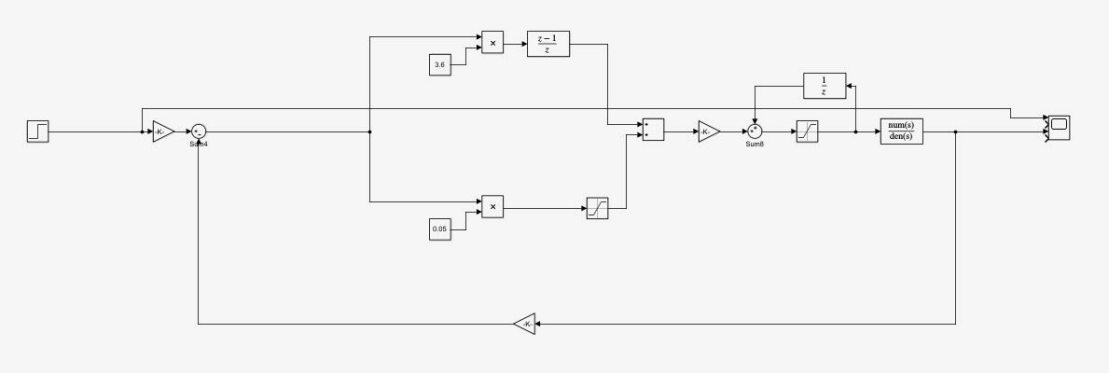


图 4.3.3 PID 控制模型

当 Kp = 3.6 , Ki = 0.05 时，PID 控制器的响应曲线如下，具有良好的动态性能：

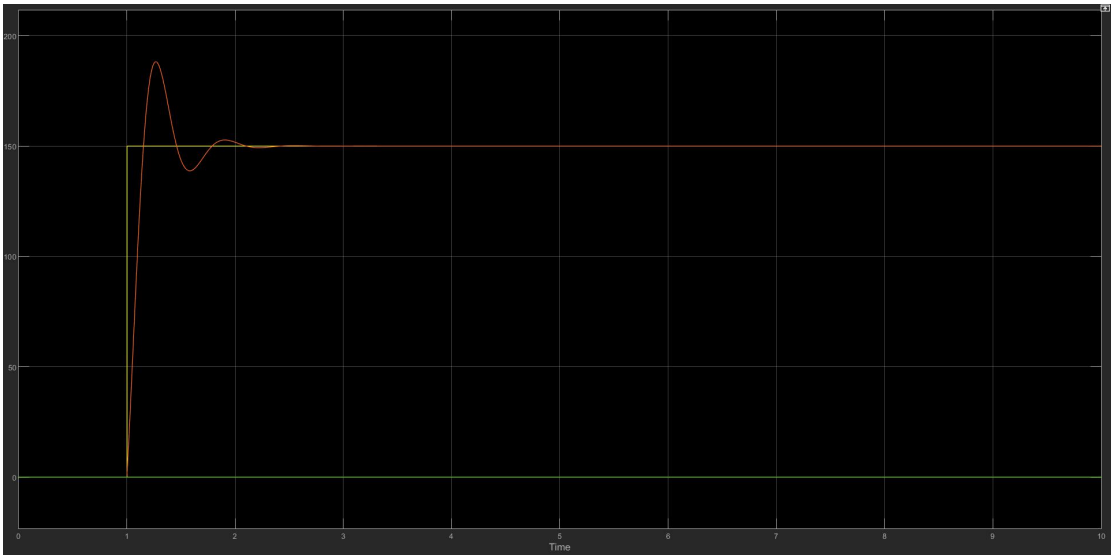


图 4.3.4 PID 响应曲线

因此，采用该参数进行电机的闭环控制。

4.3.3 变速策略

我们采用模糊控制的方法，以角度以及角度的变化作为输入，速度为输出，建立模糊规则，计算出模糊查询表。在单片机上通过查表获取车速。

第五章 神经网络控制

我们采用神经网络算法，基于短前瞻电感电压的数据，完成对舵机转向和电机转速的控制，以达到长前瞻的引导巡线效果。

5.1 数据集制作

5.1.1 电感排布

图 5.1.1 电感排布

如图为车体采集电磁数据的短电磁杆上的 8 个电感单元排布。

5.1.2 数据采集

小车在长前瞻的引导下完成常规的元素识别和循迹功能。在小车运行过程中利用相关的判别条件来更新当前数据的类别属性。具体分为：“RigthtAngle”（直角）、“Corner”（弯道）、“Cross”（十字）、“Cycle”（苯环）、“Other”（其他）、“Straight”（直道）、“None”（未识别）。

通过 TC377 写入 SD 卡，并建立.txt 格式的文件。采集数据文件参见如下：

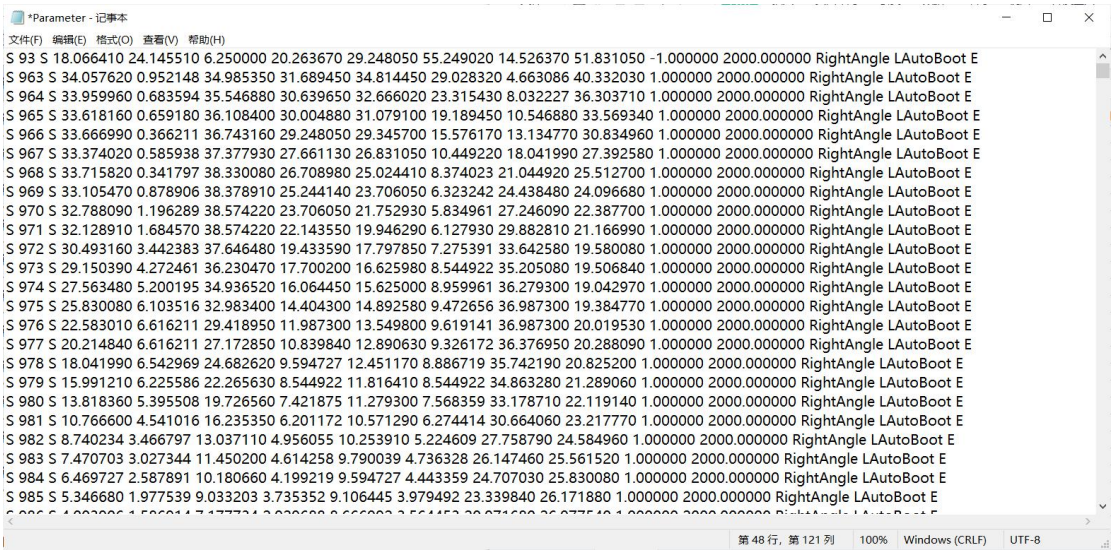

*Parameter - 记事本
文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)
S 93 S 18.066410 24.145510 6.250000 20.263670 29.248050 55.249020 14.526370 51.831050 -1.000000 2000.000000 RightAngle LAutoBoot E
S 963 S 34.057620 0.952148 34.985350 31.689450 34.814450 29.028320 4.663086 40.332030 1.000000 2000.000000 RightAngle LAutoBoot E
S 964 S 33.959960 0.683594 35.546880 30.639650 32.666020 23.315430 8.032227 36.303710 1.000000 2000.000000 RightAngle LAutoBoot E
S 965 S 33.618160 0.659180 36.108400 30.004880 31.079100 19.189450 10.546880 33.569340 1.000000 2000.000000 RightAngle LAutoBoot E
S 966 S 33.666990 0.366211 36.743160 29.248050 29.345700 15.576170 13.134770 30.834960 1.000000 2000.000000 RightAngle LAutoBoot E
S 967 S 33.374020 0.585938 37.377930 27.661130 26.831050 10.449220 18.041990 27.392580 1.000000 2000.000000 RightAngle LAutoBoot E
S 968 S 33.715820 0.341797 38.330080 26.708980 25.024410 8.374023 21.044920 25.512700 1.000000 2000.000000 RightAngle LAutoBoot E
S 969 S 33.105470 0.878906 38.378910 25.244140 23.706050 6.323242 24.438480 24.096680 1.000000 2000.000000 RightAngle LAutoBoot E
S 970 S 32.788090 1.196289 38.574220 23.706050 21.752930 5.834961 27.246090 22.387700 1.000000 2000.000000 RightAngle LAutoBoot E
S 971 S 32.128910 1.684570 38.574220 22.143550 19.946290 6.127930 29.882810 21.166990 1.000000 2000.000000 RightAngle LAutoBoot E
S 972 S 30.493160 3.442383 37.646480 19.433590 17.797850 7.275391 33.642580 19.580080 1.000000 2000.000000 RightAngle LAutoBoot E
S 973 S 29.150390 4.272461 36.230470 17.700200 16.625980 8.544922 35.205080 19.506840 1.000000 2000.000000 RightAngle LAutoBoot E
S 974 S 27.563480 5.200195 34.936520 16.064450 15.625000 8.959961 36.279300 19.042970 1.000000 2000.000000 RightAngle LAutoBoot E
S 975 S 25.830080 6.103516 32.983400 14.404300 14.892580 9.472656 36.987300 19.384770 1.000000 2000.000000 RightAngle LAutoBoot E
S 976 S 22.583010 6.616211 29.418950 11.987300 13.549800 9.619141 36.987300 20.019530 1.000000 2000.000000 RightAngle LAutoBoot E
S 977 S 20.214840 6.616211 27.172850 10.839840 12.890630 9.326172 36.376950 20.288090 1.000000 2000.000000 RightAngle LAutoBoot E
S 978 S 18.041990 6.542969 24.682620 9.594727 12.451170 8.886719 35.742190 20.825200 1.000000 2000.000000 RightAngle LAutoBoot E
S 979 S 15.991210 6.225586 22.265630 8.544922 11.816410 8.544922 34.863280 21.289060 1.000000 2000.000000 RightAngle LAutoBoot E
S 980 S 13.818360 5.395508 19.726560 7.421875 11.279300 7.568359 33.178710 22.119140 1.000000 2000.000000 RightAngle LAutoBoot E
S 981 S 10.766600 4.541016 16.235350 6.201172 10.571290 6.274414 30.664060 23.217770 1.000000 2000.000000 RightAngle LAutoBoot E
S 982 S 8.740234 3.466797 13.037110 4.956055 10.253910 5.224609 27.758790 24.584960 1.000000 2000.000000 RightAngle LAutoBoot E
S 983 S 7.470703 3.027344 11.450200 4.614258 9.790039 4.736328 26.147460 25.561520 1.000000 2000.000000 RightAngle LAutoBoot E
S 984 S 6.469727 2.587891 10.180660 4.199219 9.594727 4.443359 24.707030 25.830080 1.000000 2000.000000 RightAngle LAutoBoot E
S 985 S 5.346680 1.977539 9.033203 3.735352 9.106445 3.979492 23.339840 26.171880 1.000000 2000.000000 RightAngle LAutoBoot E
第 48 行，第 121 列 100% Windows (CRLF) UTF-8

图 5.1.2 电感数据.txt 文件

此外每个字节的具体含义如下：

字节	名称	说明	数据类型
字节 1	识别位 1	字符 S	char
字节 2	数据编号	每位数据的编号	uint16
字节 3	识别位 1	字符 S	char
字节 4	电感 1	左 1 横电感（归一值*100）	float32
字节 5	电感 2	左 2 竖电感	float32
字节 6	电感 3	左 3 斜电感	float32
字节 7	电感 4	左 4 横电感	float32
字节 8	电感 5	右 4 横电感	float32
字节 9	电感 6	右 3 斜电感	float32
字节 10	电感 7	右 2 竖电感	float32
字节 11	电感 8	右 1 横电感	float32
字节 12	角度	归一值±1	float32
字节 13	速度	归一值*10000	float32
字节 14	元素标签	元素标签	char
字节 15	识别模式	采集模式	char
字节 16	标识位 3	字符 E	char

以上是原始数据的采集，通过 Python 脚本转换生成 8 电感与打角值的预测模型以及赛道当前状态的多元非线性分类器的训练集和测试集。

图 5.1.3 分类别采集数据

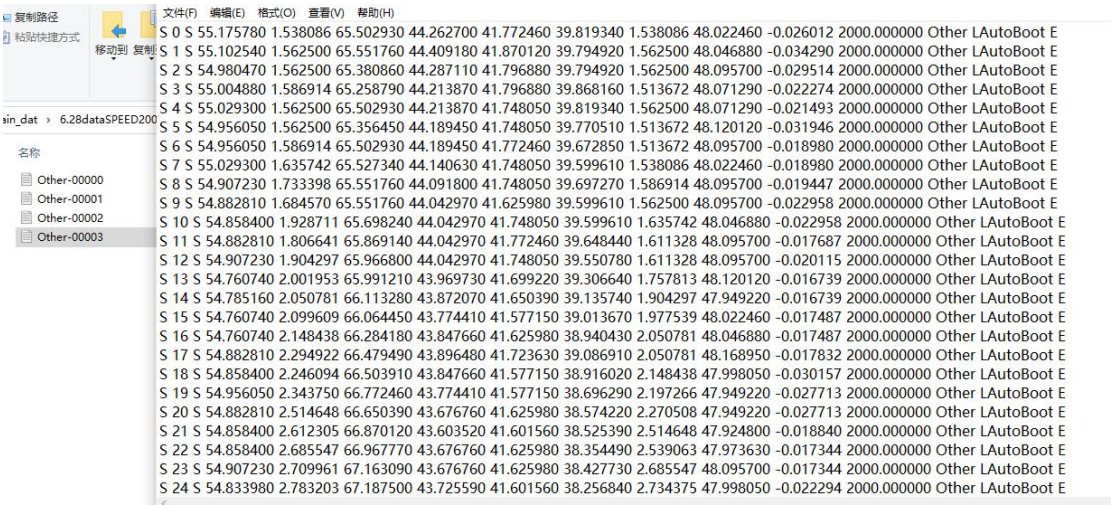


图 5.1.4 单一类别数据采集

首次采集综合数据，开始第一次训练，此后根据车模在赛道上的识别情况来分类别采集更多的特殊数据。并且加入数据集，重新训练。

此外，可利用 excel 来处理 txt 文本数据集，方便分析数据的统计特性和占比。

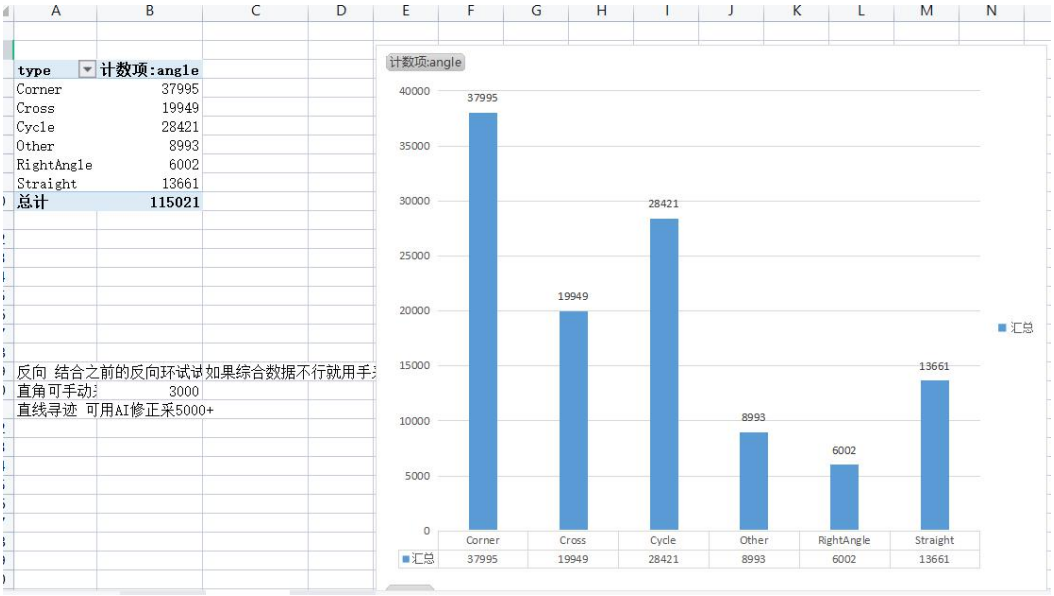


图 5.1.5 分析数据集

同时我们通过统计分析工具得出电感中的多重共线性参数，以此修正高相关性的电感分布原始数据，减小赛道特征学习的难度。

其中斜电感最为重要，左右 1、2 号电感次之。在通过采集多种特殊数据后计算其与打角值的皮尔逊相关系数来分析权重的初步分布。

相关性

		ad1	ad2	ad3	ad4	ad5	ad6	ad7	ad8	angle
ad1	皮尔逊相关性	1	.130**	.844**	.876**	.643**	.253**	.202**	.377**	.266**
	Sig.（双尾）		.000	.000	.000	.000	.000	.000	.000	.000
	个案数	23025	23025	23025	23025	23025	23025	23025	23025	23025
ad2	皮尔逊相关性	.130**	1	-.026**	-.046**	-.012	.205**	.714**	.236**	-.249**
	Sig.（双尾）	.000		.000	.000	.080	.000	.000	.000	.000
	个案数	23025	23025	23025	23025	23025	23025	23025	23025	23025
ad3	皮尔逊相关性	.844**	-.026**	1	.719**	.504**	-.092**	.248**	.187**	.590**
	Sig.（双尾）	.000	.000		.000	.000	.000	.000	.000	.000
	个案数	23025	23025	23025	23025	23025	23025	23025	23025	23025
ad4	皮尔逊相关性	.876**	-.046**	.719**	1	.895**	.549**	.057**	.627**	.091**
	Sig.（双尾）	.000	.000	.000		.000	.000	.000	.000	.000
	个案数	23025	23025	23025	23025	23025	23025	23025	23025	23025
ad5	皮尔逊相关性	.643**	-.012	.504**	.895**	1	.771**	.064**	.865**	-.041**
	Sig.（双尾）	.000	.080	.000	.000		.000	.000	.000	.000
	个案数	23025	23025	23025	23025	23025	23025	23025	23025	23025
ad6	皮尔逊相关性	.253**	.205**	-.092**	.549**	.771**	1	.068**	.901**	-.524**
	Sig.（双尾）	.000	.000	.000	.000	.000		.000	.000	.000
	个案数	23025	23025	23025	23025	23025	23025	23025	23025	23025
ad7	皮尔逊相关性	.202**	.714**	.248**	.057**	.064**	.068**	1	.250**	.082**
	Sig.（双尾）	.000	.000	.000	.000	.000	.000		.000	.000
	个案数	23025	23025	23025	23025	23025	23025	23025	23025	23025
ad8	皮尔逊相关性	.377**	.236**	.187**	.627**	.865**	.901**	.250**	1	-.269**
	Sig.（双尾）	.000	.000	.000	.000	.000	.000	.000		.000
	个案数	23025	23025	23025	23025	23025	23025	23025	23025	23025
angle	皮尔逊相关性	.266**	-.249**	.590**	.091**	-.041**	-.524**	.082**	-.269**	1
	Sig.（双尾）	.000	.000	.000	.000	.000	.000	.000	.000	
	个案数	23025	23025	23025	23025	23025	23025	23025	23025	23025

**. 在 0.01 级别（双尾），相关性显著。

图 5.1.3 数据中电感的相关性

此外还要对采集的原始数据的增加一定的噪声，增强此后训练模型的鲁棒性。因此要将正常寻迹和特殊元素分开来采集。

对于直道，我们选择给打角值随机的扰动，让车模偏离赛道，然后采集修正的打角值，这是十分重要的，这样的数据会使车模寻迹时的修正效果比 PID 的效果更好，也更快。

对于特殊元素，则需人工抓取车模在赛道上晃动采集，以此提升采集数据的数量和质量，直角、苯环、十字均需这样处理。

直角需要用手推动采集数据，苯环则需要用手推采集正向、反向的入环、出环数据，而十字则要从四个方向分别采集数据保存，完善数据集。

5.2 建立神经网络模型

5.2.1 舵机打角预测模型

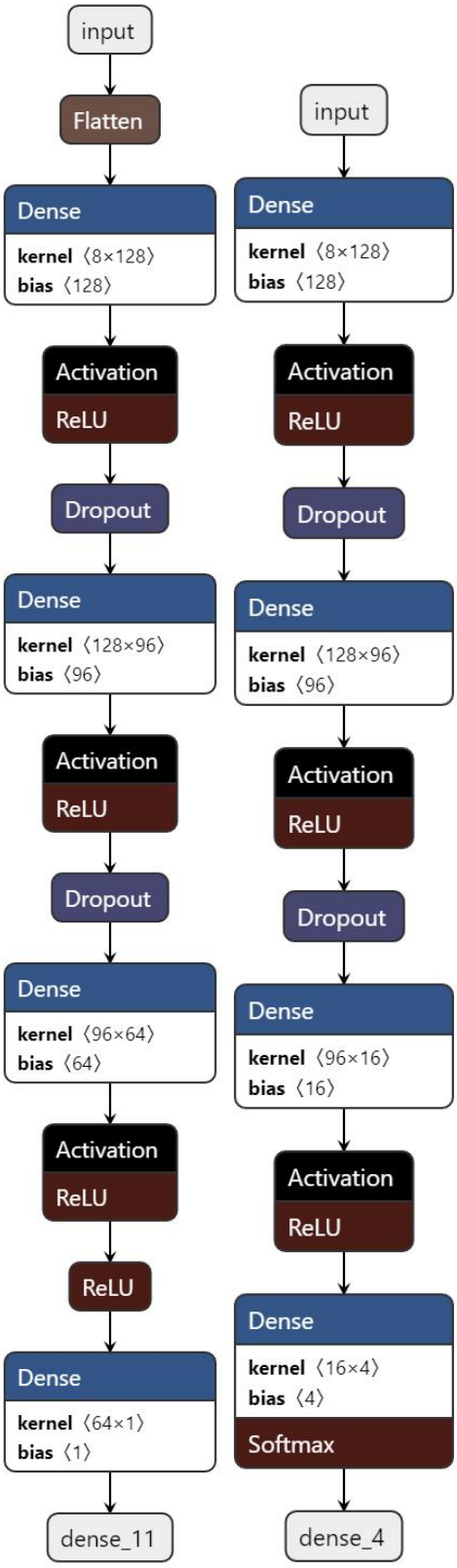
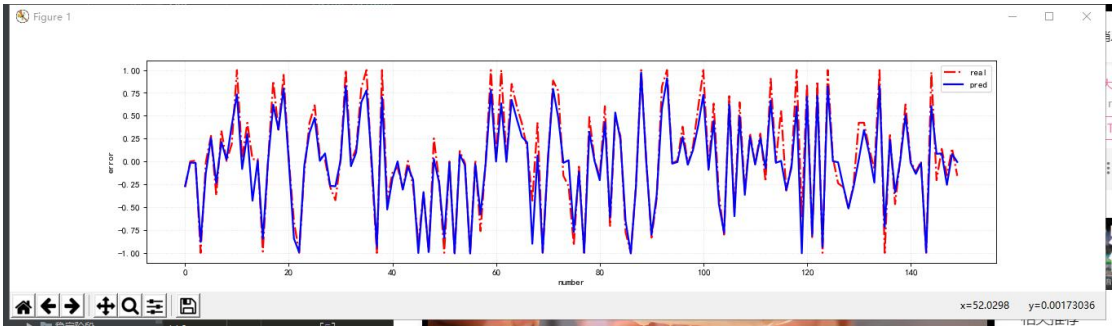


图 5.2.1 打角预测模型及元素分类器网络结构

5.3 模型训练

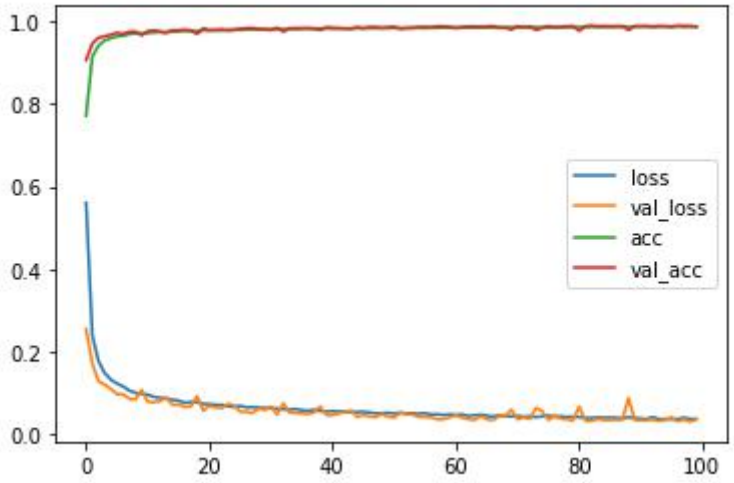
```
print('Train %d times' % (i + 1))
early_stopping = EarlyStopping(monitor='val_loss', patience=50, verbose=2)
history = model.fit(x_train, y_train, batch_size=128, epochs=epocsPerTrain,validation_data=(x_test, y_test), \
shuffle = True, class_weight="auto",callbacks=[early_stopping])  # callbacks=[TensorBoard(log_dir='./Log_' + mo
```

fit 函数参数配置如上图，其中根据学习进度调整 batch_size：根据模型 loss 和 mse、mae 的参数收敛情况选择增大还是缩小：常选 32、64、128 三种规格。



5.3.1 直接预测模型的测试集可视化

该模型学习提取的特征可以满足车体识别所有元素的打角值。



5.3.2 元素分类模型的训练 history 可视化

该模型的分类功能能够识别所有车体的状态准确率 98.8%及以上。

5.4 模型部署

将以上的模型保持为.h5 格式，通过 nncu 转换为 c 文件推算模型即可。

其中参数注意输入数据移位指数，选用 12 位，使得模型输出的范围为 4096 以内。分类模型的转换注意切换输出后处理类型为分类。设定命令，点击“干活”生成模型和权重文件放入车体工程中调用 RunModel 函数即可。

5.5 控制算法设计

在项目工程中，直接调用 RunModel()函数即可得到输出。但其中模型的输出误差收以下因素影响：

1、模型转换过程的量化误差；

2、模型本身的拟合误差；

3、车体运行时电感采集的细微差别等。

要想尽可能的减小以上误差，我们通过增加修正函数来使输出数据的拟合性更强，更接近长前瞻的效果，并且将长前瞻相对短前瞻的优势，即对于一些特殊元素如直角等可相对能提前打角，使车体稳定在使用 AI 循迹时更稳定。

我们通过一种自优化的方式来生产修正函数。

当通过长前瞻和短前瞻 AI 推算循迹的效果相差不大时，采用该方法来进一步拟合，在利用 AI 模式循迹的过程中，同时启用长前瞻，输出打角值，同时保持 AI 推算的打角值，两者做差保存。其中若误差小于所设定的阈值，如差值小于长前瞻打角的 10%浮动范围，则舍去，其余则认为推算误差过大需修正的数据。

其中推算误差大的数据可保存并加入模型重新训练，若整体效果不错，可直接用 excel 把误差数据导入，并生成一个补偿函数即可。

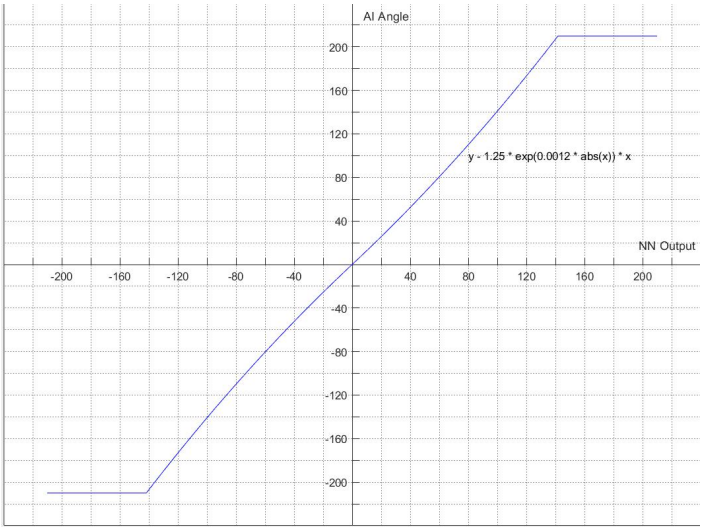


图 5.5.1　AI 输出预测修正函数

如图所示，NN Output 为神经网络的输出结果，AI Angle 为修正后的角度，对角度进行修正后取得了较好的寻迹效果。

5.6 融合算法

我们将 AI 模型预测出的角度(AI Angle)，以及运用传统算法+短前瞻得出的角度(Cor Ange)进行结合，以得到最终的舵机角度(Angle)。

```
data->AIAngle = NeuralNetworkReasoning(data);

data->AIAngle = ConstrainFloat(data->AIAngle,Servo.MinAngle,Servo.MaxAngle);

if(!data->Is_AdjustAngle)
    data->CorAngle = FuzzyControl(&data->S_Fuzzy,0.0,data->Bias) * Servo.MaxAngle;

data->CorAngle = ConstrainFloat(data->CorAngle,Servo.MinAngle,Servo.MaxAngle);

if(Is_LoseLine(data))
{
    data->Angle = fsign(data->A[9] * 0.6 + data->A[8] * 0.4) * Servo.MaxAngle;//LowPass Filter.
}
else if(fabs(data->AIAngle) <= 20.0)
{
    data->Angle = data->CorAngle;
}
else
{
    data->Angle = data->AIAngle;
}
```

当 |AI Angle| <= 20.0 时，Angle = Cor Angle;

当 |AI Angle| >= 20.0 时，Angle = AI Angle;

如果传统算法判断成丢线状态或者偏离过大，则根据前两次角度的方向，

将角度设置成左侧(或右侧)最大值，即通过前两次的角度进行加权处理，取加权后的符号，乘以最大角度得到。

Angle = MaxAngle * Sign(A[9] * 0.6 + A[8] * 0.4);

然后根据舵机的角度输出，采用 4.3.3 的方法完成速度控制。

第六章 结论

本文从机械、硬件、软件等方面详细介绍了车模的设计和制作方案。从机械和硬件方面，我们力求电路和结构的稳定可靠，尽量将车模做的轻快灵活，尽可能降低重心;在软件方面，我们努力追求最有效的行驶路径以及控制算法，不断优化各方面策略。经过比赛证明，此方案可行。

自从参加智能车竞赛以来，我们小队都积极查阅资料，学习提高自己的能力，为智能车制作奉献了许多时间和经力。在车模制作过程中，我们学到很多知识，逐渐对单片机控制、电路设计等有了比较好的认识，很感谢比赛能够给我们这样的平台，感谢各位指导老师和学长、学姐的指导。经过近一年的辛苦准备，我们也积累了很多经验，深刻认识到坚持不懈的重要性，培养了发现问题、解决问题的能力。

最后，再次感谢扬州大学创新实验室这个大家庭，感谢一直支持和关注智能车比赛的学校和学院领导以及各位老师，也感谢比赛组委会能组织这样一项很有意义的比赛。

参考文献

[1] 卓晴，黄开胜，邵贝贝.学做智能车 [M].北京:北京航空航天大学出版 社.2007.

[2] 张军.AVR 单片机应用系统开发典型实例 [M].北京:中国电力出版社，2005.

[3] 邵贝贝.单片机嵌入式应用的在线开发方法 [M].北京.清华大学出版 社.2004.

[4] 张文春.汽车理论 [M].北京.机械工业出版社.2005.

[5] 蔡述庭.“飞思卡尔”杯智能汽车竞赛设计与实践 [M].北京:北京航空航天大学出版社.2012.

[6]尹勇. Protel DXP 电路设计入门与进阶 [M] . 北京: 科学出版社，2004.

[7]夏克俭. 数据结构及算法 [M] . 北京：国防工业出版社， 2001.

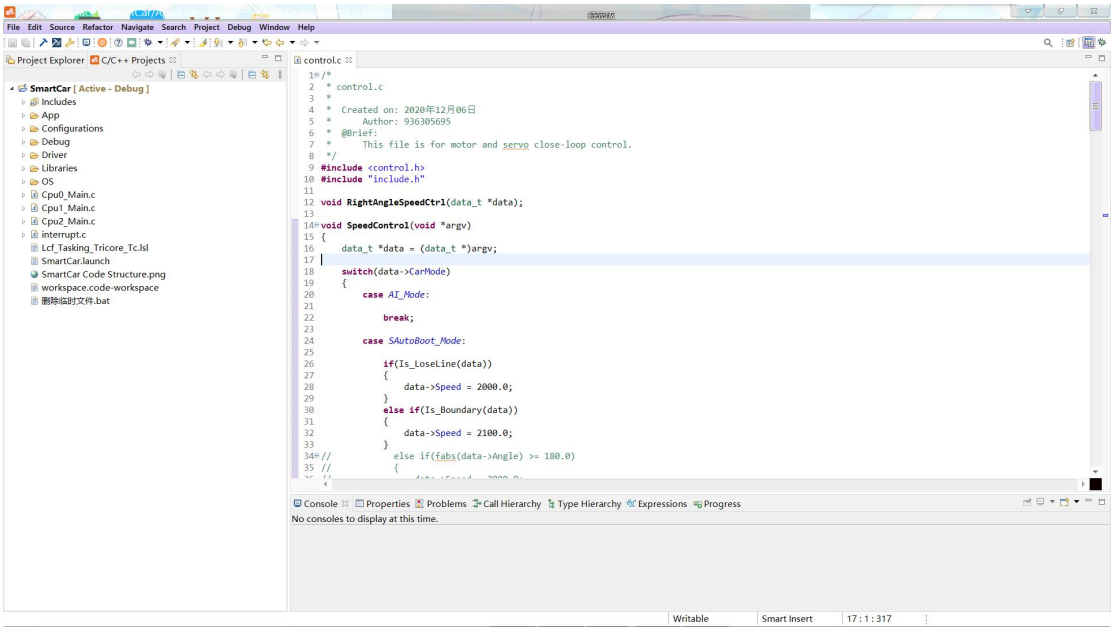
[8]孙文韬. 反向传播神经网络的原理研究及优化方法[J]. 中国科技博览，2011,000(032)：511-511.

[9]李太福. 基于在线参数自整定的模糊 PID 伺服控制系统[J] . 交流伺服系统，2005，4：203～215.

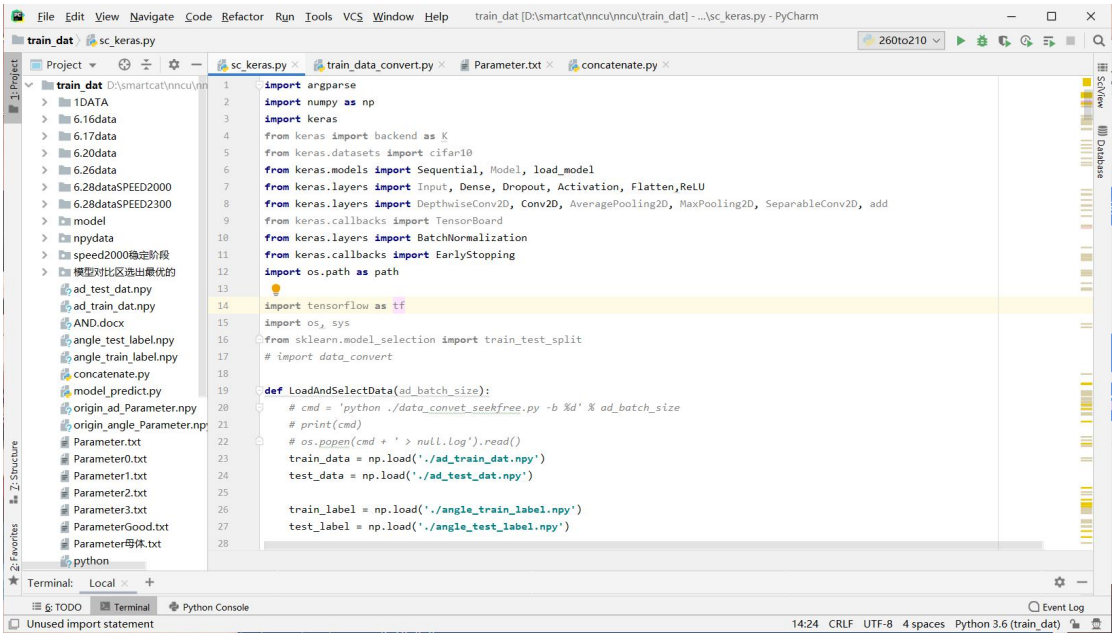
[10]刘天舒. BP 神经网络的改进研究及应用[D].东北农业大学,2011.

附录 A 调试软件

1. 软件开发平台：AURIX Development Studio

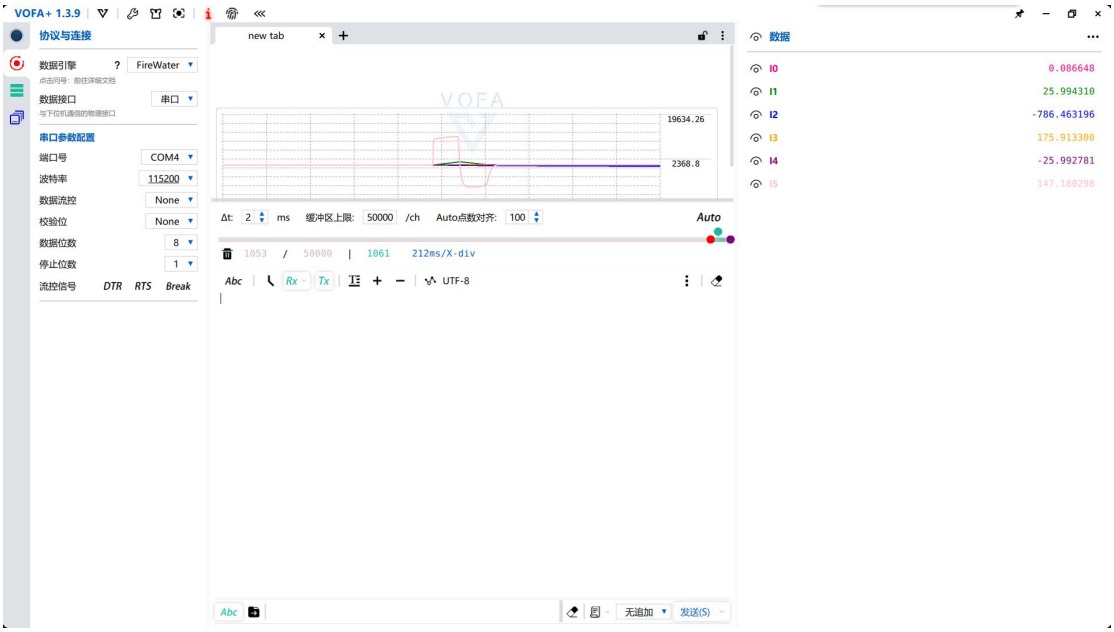


2. 神经网络算法编写：Pycharm



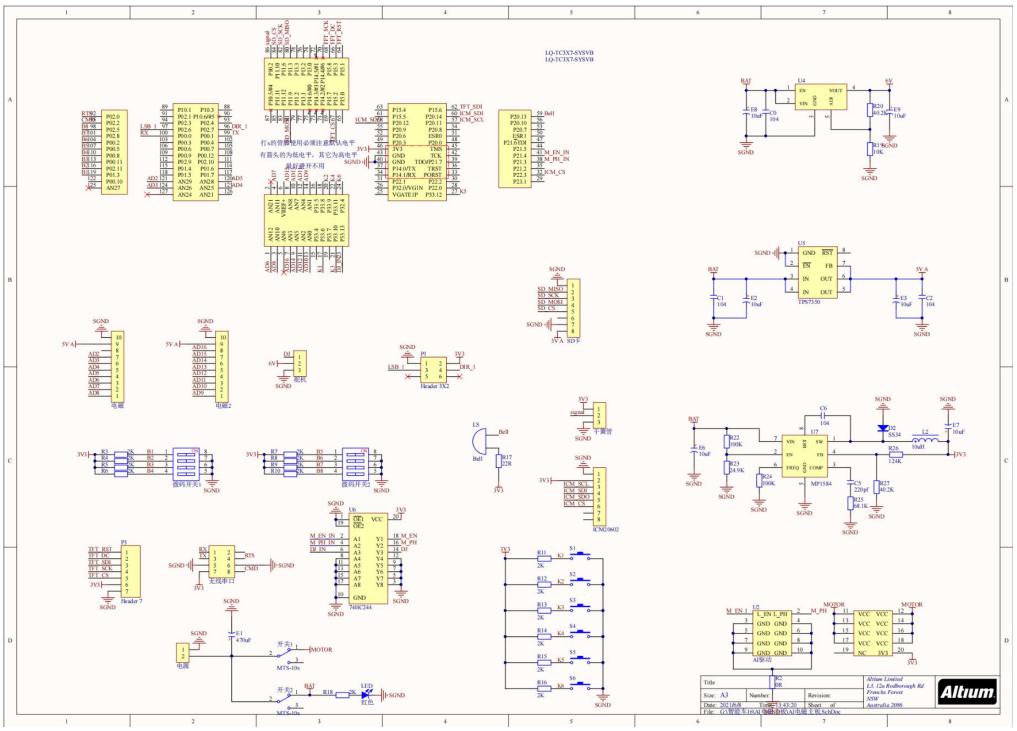
3. 神经网络 h5 文件转换工具:NNCU

4.调试上位机:VOFA+

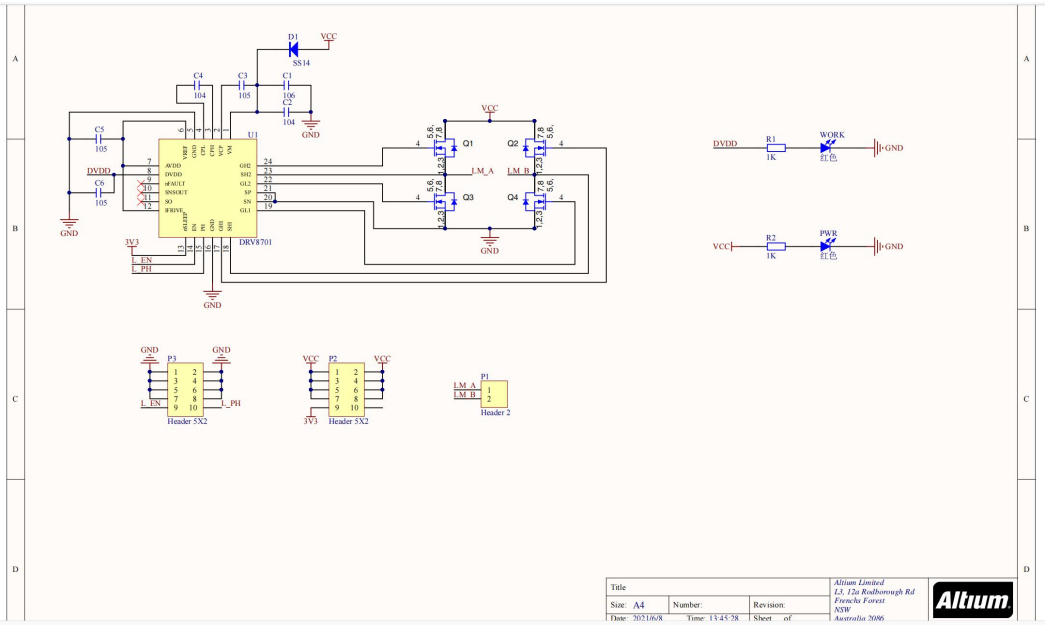


附录 B 电路板原理图

1.主板

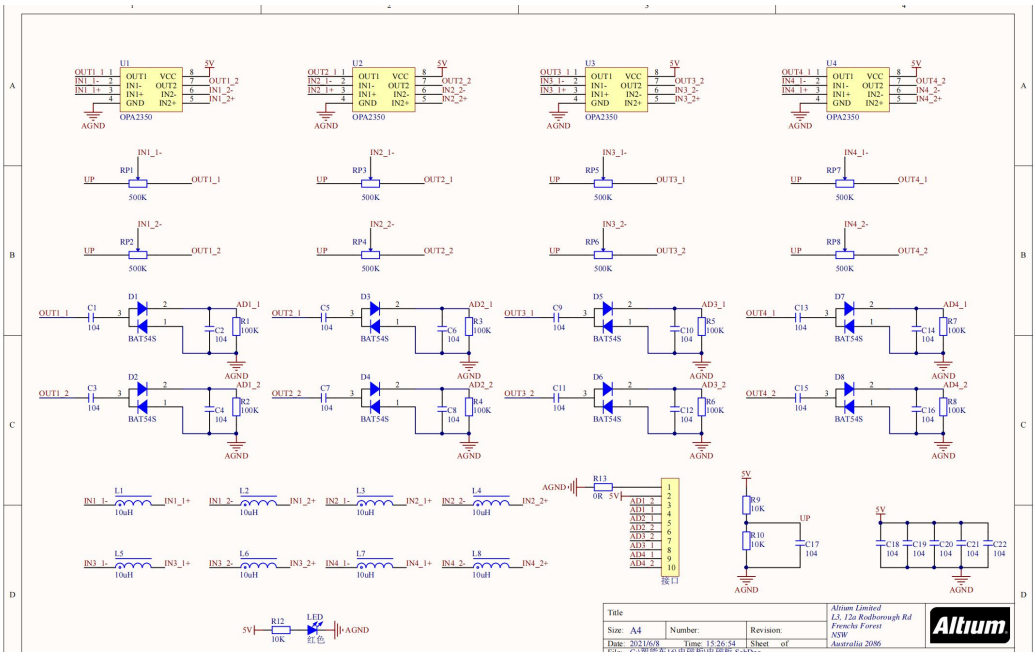


2. 驱动板


D1
SS14
VCC
C4
104
C3
105
C1
106
C2
104
GND
C5
105
C6
105
U1
DRV8701
DVDD
3V3
L_EN
L_PH
Q1
Q2
Q3
Q4
LM_A
LM_B
R1
1K
WORK
R2
1K
PWR
P3
Header 5X2
P2
Header 5X2
P1
Header 2
Title
Size: A4
Number:
Revision:
Date: 2021/6/8
Time: 13:45:58
Sheet of
Altium Limited
L3, 12a Rodborough Rd
Frenchs Forest
NSW
Australia 2086
Altium

1 / 2

3.电磁板


U1
U2
U3
U4
OPA2350
AGND
5V
RP1
RP2
RP3
RP4
RP5
RP6
RP7
RP8
500K
D1
D2
D3
D4
D5
D6
D7
D8
BAT54S
L1
L2
L3
L4
L5
L6
L7
L8
10uH
R13
0R
R9
10K
R10
10K
C17
104
C18
C19
C20
C21
C22
R12
10K
LED
红色
Title
Size: A4
Number:
Revision:
Date: 2021/6/8
Time: 15:26:54
Sheet of
Altium Limited
L3, 12a Rodborough Rd
Frenchs Forest
NSW
Australia 2086
Altium

附录 C 部分源程序代码

1.神经网络部分

```
float NeuralNetworkCalAngle(void *argv)
{
    data_t *data = (data_t *)argv;

    sint16_t *angle = NNForWardReasoning(NULL,data->SESensor_SampleValue,0);

    float res = ((*angle) >> (14 - 10 - 1) ) * 1.0;

    data->NNOutput = (res);

    return 1.25 * exp(0.0012 * fabs(res)) * res;

    //return 1.4 * exp(0.0006 * fabs(res)) * res;

    //return 1.32 * exp(0.008 * fabs(res)) * res;

    //return 1.2 * exp(0.0015 * fabs(res)) * res;

    //return res * 1.25;
}
```

2.控制部分

```
/*
 * control.c
 *
 *  Created on: 2020年12月06日
 *
 *      Author: 936305695
 *      Github：https://github.com/GTshenmi/AI-SmartCar
 *
 *  @Brief:
 *      This file is for motor and servo close-loop control.
 */
#include <control.h>
#include "include.h"

void RightAngleSpeedCtrl(data_t *data);

void SpeedControl(void *argv)
{
    data_t *data = (data_t *)argv;

    switch(data->CarMode)
    {
        case AI_Mode:

            break;

        case SAutoBoot_Mode:
```

```
            if(Is_LoseLine(data))
            {
                data->Speed = 2000.0;
            }
            else if(Is_Boundary(data))
            {
                data->Speed = 2100.0;
            }
//            else if(fabs(data->Angle) >= 180.0)
//            {
//                data->Speed = 2000.0;
//            }
            else
            {
                data->Speed                                                    =
FuzzySpeedControl(&data->M_FuzzySpeed,0.0,data->Angle * 100.0/210.0);
            }

            RightAngleSpeedCtrl(data);

            break;

        case DebugMode:

            Motor.SetPwmValue(Motor.Self,data->Speed);

            break;
```

```
        case LAutoBoot_Mode:default:

            break;

    }

    /*set speed*/
    if(data->CarMode != DebugMode)
    {
        float formatedSpeed = 0.0;

        formatedSpeed = (data->Speed * Motor.GetMaxSpeed(Motor.Self))/10000.0;
// target * 0.055

        data->ActualSpeed = Motor.GetSpeed(Motor.Self);

        data->x += data->ActualSpeed * 0.000028 * 100.0;

        Motor.SetSpeed(Motor.Self,formatedSpeed);

        Motor.Update(Motor.Self);
    }
}

void AngleControl(void *argv)
{
    data_t *data = (data_t *)argv;

    /*calculate angle according to bias or use nn.*/
```

```
    switch(data->CarMode)
    {
        case AI_Mode:

            data->AIAngle = NeuralNetworkReasoning(data);

            data->AIAngle                                                    =
ConstrainFloat(data->AIAngle,Servo.MinAngle,Servo.MaxAngle);

            if(!data->Is_AdjustAngle)
                data->CorAngle = FuzzyControl(&data->S_Fuzzy,0.0,data->Bias) *
Servo.MaxAngle;

            data->CorAngle                                                   =
ConstrainFloat(data->CorAngle,Servo.MinAngle,Servo.MaxAngle);

            data->Angle = data->AIAngle;

            //data->Angle = data->AIAngle + data->CorAngle;

            break;

        case SAutoBoot_Mode:

            data->AIAngle = NeuralNetworkReasoning(data);

            data->AIAngle                                                    =
ConstrainFloat(data->AIAngle,Servo.MinAngle,Servo.MaxAngle);
```

```
            if(!data->Is_AdjustAngle)
                data->CorAngle = FuzzyControl(&data->S_Fuzzy,0.0,data->Bias) *
Servo.MaxAngle;

            data->CorAngle                                                    =
ConstrainFloat(data->CorAngle,Servo.MinAngle,Servo.MaxAngle);

            if(Is_LoseLine(data))
            {
                data->Angle = fsign(data->A[9] * 0.6 + data->A[8] * 0.4) *
Servo.MaxAngle;//LowPass Filter.
            }
            else if(fabs(data->AIAngle) <= 20.0)
            {
                data->Angle = data->CorAngle;
            }
            else
            {
                data->Angle = data->AIAngle;
            }

            break;

        case DebugMode:

            Servo.SetPwmValue(Servo.Self,data->SPwmValue);

            break;
```

```
        case LAutoBoot_Mode:default:

            if(!data->Is_AdjustAngle)
                data->Angle  =  FuzzyControl(&data->S_Fuzzy,0.0,data->Bias)  *
Servo.MaxAngle;

            data->Angle                                                    =
ConstrainFloat(data->Angle,Servo.MinAngle,Servo.MaxAngle);

            break;

    }

    /*record historical angle*/
    for(int i = 0 ; i < 9 ;i++)
    {
         data->A[i] = data->A[i + 1];
    }

    data->A[9] = data->Angle;

    /*set angle*/
    if(data->CarMode != DebugMode)
    {
        Servo.SetAngle(Servo.Self,data->Angle);

        Servo.Update(Servo.Self);
    }
```

```
}

void RightAngleSpeedCtrl(data_t *data)
{
    rightangle_speedctrl RASCState = RASC_Wait;

    switch(RASCState)
    {
        case RASC_Wait:

            if((data->A[8] - data->A[7]) >= 90.0 && data->Angle >= 140.0)
            {
                RASCState = RASC_Tracking;
            }

            break;
        case RASC_Tracking:
            data->Speed = 2000.0;

            if(data->h_bias <= 30.0 || data->Angle <= 20.0)
            {
                RASCState = RASC_Out;
            }
            break;

        case RASC_Out:
            RASCState = RASC_Wait;

            break;
```

```
        default:
            break;
    }
}

/*speed close loop control.*/
sint16_t  MotorCtrlStrategy(struct  motor_ctrl  *self,float  target_speed,float
actual_speed,void *argv,uint16_t argc)
{
    sint16_t PwmValue = 0;

    data_t *data =(data_t *)argv;

    float tspeed,aspeed = 0.0;

    data->SpeedLoopCtrlType = PID;

    tspeed = 100.0 * NormalizeFloat(target_speed,0.0,self->MaxSpeed);

    aspeed = 100.0 * NormalizeFloat(actual_speed,0.0,self->MaxSpeed);

    //aspeed = 100.0 * random(0.0,1.0);

    if(0)
    {
        PwmValue = (sint16_t)(PWMx.MaxPwmDuty * fsign(data->M_PID.PID_Error[2])
* 0.6);
    }
```

```
    else
    {
        switch(data->SpeedLoopCtrlType)
        {
            case PID:

                PID_Ctrl(&data->M_PID,tspeed,aspeed);

                PwmValue                                                    =
(sint16_t)ConstrainFloat(data->M_PID.Result,-10000.0,10000.0);

                break;

            case FuzzyPID:


FuzzyPIDCtrl(&data->M_FuzzyKp,&data->M_FuzzyKi,tspeed,aspeed);

                data->M_PID.Kp = 3.2 + data->M_FuzzyKp.U;
                data->M_PID.Ki = 0.5 + data->M_FuzzyKi.U;

                PID_Ctrl(&data->M_PID,tspeed,aspeed);

                PwmValue                                                    =
(sint16_t)ConstrainFloat(data->M_PID.Result,-10000.0,10000.0);

                break;

            case ADRC:
```

```
                ADRC_Control(&data->M_ADRC,target_speed,actual_speed);

                PwmValue = (sint16_t)ConstrainFloat(data->M_ADRC.u,0,10000.0);

                break;

            case OpeningLoop:

                PwmValue                                                    =
(sint16_t)ConstrainFloat(target_speed,-10000.0,10000.0);

                break;

        }

        //Console.WriteLine("MPID:%.3f,%.3f",data->ActualSpeed,data->Speed   *
550.0 /10000.0);



    }

    //PwmValue = 3000;

    return PwmValue;
}

uint16_t  ServoCtrlStrategy(struct  servo_ctrl  *self,float  target_angle,float
actual_angle,void *argv,uint16_t argc)
```

```
{
    float angle = 0.0;

    angle = target_angle + self->PwmCentValue;

    return (uint16_t)angle;
}
```

3.元素部分

```
/*
 * element.c
 *
 *  Created on: 2020年12月6日
 *      Author: 936305695
 *  @Brief:
 *          This file is for the special element.
 */
#include <element.h>
#include "include.h"

void LoseLine_Handler(data_t *data);
void Cycle_Handler(data_t *data);
void Cross_Handler(data_t *data);
void RightAngle_Handler(data_t *data);
void Ramp_Handler(data_t *data);
void RustAngle_Handler(data_t *data);
void StopSituationDetect(data_t *data);

//typedef bool (*Element_HandlerAI_F)(data_t *data);

bool RightAngle_HandlerAI(data_t *data);
bool Cycle_HandlerAI(data_t *data);
```

```
bool Cross_HandlerAI(data_t *data);

//bool            (*Element_HandlerAI)(data_t            *data)[5]            =
{RightAngle_HandlerAI,Cycle_HandlerAI,Cross_HandlerAI,Cycle_HandlerAI,RightAn
gle_HandlerAI};

//Element_HandlerAI_F                    Element_HandlerAI[5]                    =
{RightAngle_HandlerAI,Cycle_HandlerAI,Cross_HandlerAI,Cycle_HandlerAI,RightAn
gle_HandlerAI};

uint GameElement[5] = {RightAngle,Cycle,Cross,Cycle,RightAngle};

typedef enum
{
    WaitElement,
    ElementSwitch,
    HandlerElement,

}element_switch_t;

float ElementDetermineAI(void *argv)
{
//    data_t *data = (data_t *)argv;
//
//    static uint elementPointer = 0;

    static element_switch_t elementState;

    //static uint elementId = 0;
```

```
    switch(elementState)
    {
        case WaitElement:

            break;

        case ElementSwitch:

            break;

        case HandlerElement:

          //  if(Element_HandlerAI[elementId](argv))
            {
                elementState = WaitElement;
            }

            break;
    }

    return 0.0;
}

bool Cross_HandlerAI(data_t *data)
{

    return false;
}
```

```
bool Cycle_HandlerAI(data_t *data)
{

    return false;
}

bool RightAngle_HandlerAI(data_t *data)
{
    static rightangle_state_t rightAngleState = RA_Wait;

    switch(rightAngleState)
    {
        case RA_Wait:

            break;

        case RA_Config:

            goto RA_CONFIRM;

            break;

        case RA_Confirm:    //confirm and config some parameter.

            RA_CONFIRM:

            if(1)
            {
```

```
                GLED.ON(GLED.Self);

                rightAngleState = RA_Tracking;

            }

            break;

        case RA_Tracking:



            break;

        case RA_Out:

            break;

        default:

            break;
    }

    if(rightAngleState == RA_Out)
        return true;
    else
        return false;
}

float ElementDetermine(void *argv)
```

```
{
    data_t *data = (data_t *)argv;

    static uint32_t loseLineCnt = 0;

    if(Is_Ramp(data))
    {
        data->Element.Type = Ramp;
    }

    if(Is_RightAngle(data))
    {
        data->Element.Type = RightAngle;
    }

    if(Is_Cycle(data))
    {
        data->Element.Type = Cycle;
    }

    if(Is_Cross(data) && data->Element.Type != Cycle && data->Element.Type !=
RightAngle)
    {
        data->Element.Type = Cross;
    }

    if(Is_LoseLine(data)    &&    data->Element.Type    !=    RightAngle    &&
data->Element.Type != Cycle)
```

```
    {
        data->TrackingState = LoseLine;
        loseLineCnt++;

        //DebugBeepOn;
    }
    else
    {
        data->TrackingState = Normal_Tracking;
        loseLineCnt = 0;

        //DebugBeepOff;
    }

//    if(loseLineCnt >= 1500 && data->CarMode != DebugMode)
//    {
//        Motor.Break(Motor.Self);
//        Servo.Break(Servo.Self);
//    }

    ElementDetermineAI(data);

    return data->Element.Type * 1.0;

}

/*
 * @Brief:特殊元素处理接口函数
 * */
```

```
void SpecialElementHandler(void *argv)
{
    data_t *data = (data_t *)argv;

    RightAngle_Handler(data);

    Cross_Handler(data);

    Cycle_Handler(data);

    LoseLine_Handler(data);

    Ramp_Handler(data);

    RustAngle_Handler(data);

    StopSituationDetect(data);
}

void StopSituationDetect(data_t *data) {
    //GPIO Resources[27]

    static _Bool firstExecute = 1;
    static float startTime = 0.0;
    static float startPos = 0.0;

    static int times = 2;

    switch(data->GameMode)
```

```
    {
        case OneLopMode:

            if (firstExecute) {
                GPIOx.Init(&GPIO_Resources[27].GPION);
                firstExecute = 0;
                startTime =os.time.getTime(s);
                startPos = data->x;
            }

            if ((!GPIOx.Read(&GPIO_Resources[27].GPION)) && ((os.time.getTimes()
- startTime) >= 10 && ((data->x - startPos) >= 100.0))) {

                startTime =os.time.getTime(s);
                startPos = data->x;

                times--;
            }

            if(times <= 0)
            {
                Motor.Break(Motor.Self);
            }

            break;

        case ScoringMode:

            if((os.time.getTime(s) - data->GameStartTime) >= 60.0 * 3)
```

```
            {
                Motor.Break(Motor.Self);
                data->GameOverTime = os.time.getTime(s);
            }

            break;
    }
}

void Ramp_Handler(data_t *data)
{
    //data->Bias = 0.0;
}

void RustAngle_Handler(data_t *data)
{
    //data->Bias          =          Servo.GetMaxAngle(Servo.Self)          *
fsign(data->HESensor[0].Value - data->HESensor[1].Value);
}

void Cycle_Handler(data_t *data)
{
    static cycle_state_t cycleState = CC_Wait;
    static cycle_dir_t   cycleDir = CC_DirUndefined;

    static cycle_cnt cycleCnt = {0,0,0,0,0,0,0};

    static cycle_flag_t cycleFlag = {false,false,false,false,false,false};
```

```
static cycle_config  cycleConfig = {0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,0.0};

switch(cycleState)
{
    case CC_Undefined:

        cycleState = CC_Exception_Handler;

        break;

    case CC_Exception_Handler:

        CycleClearConfig(&cycleConfig);

        CycleClearCnt(&cycleCnt);

        CycleClearFlag(&cycleFlag);

        BLED.OFF(BLED.Self);

        data->Element.Type = None;

        cycleCnt.Wait = 2000;

        cycleState = CC_Wait;

        break;

    case CC_Wait:
```

```
            cycleCnt.Wait--;

            if(cycleCnt.Wait <= 0)
                cycleCnt.Wait = 0;

            if(data->Element.Type == Cycle && cycleCnt.Wait <= 0)
            {
                cycleState = CC_Config;
            }

            break;

        case CC_Config:

            CycleClearConfig(&cycleConfig);
            CycleClearCnt(&cycleCnt);
            CycleClearFlag(&cycleFlag);

            cycleConfig.inCyclePointYaw = 0.0;

            cycleConfig.dYaw = 0.0;

            cycleConfig.waitInDistance = data->CycleWaitInDistance;
            cycleConfig.inDistance = data->CycleInDistance;

            cycleConfig.sum_l        =        data->HESensor[0].Value        +
data->VESensor[0].Value;
```

```
            cycleConfig.sum_r        =        data->HESensor[3].Value        +
data->VESensor[1].Value;

            if(cycleConfig.sum_l > cycleConfig.sum_r)//Judge the direction of
cycle.
            {
                cycleDir = CC_DirLeft;
            }
            else
            {
                cycleDir = CC_DirRight;
            }

            if(cycleDir == CC_DirLeft)
            {
                cycleConfig.bias = 100.0;
            }
            else
            {
                cycleConfig.bias = -100.0;
            }

            cycleConfig.isCyclePos = data->x;

            BLED.ON(BLED.Self);

            cycleState = CC_Confirm;

            break;
```

```
        case CC_Confirm:

            cycleCnt.Confirm--;


if(Is_CycleConfirmed(data,&cycleCnt,&cycleFlag,cycleDir))//Confirm.
            {
                cycleConfig.inCyclePointYaw = data->attitude.yaw;

                //cycleConfig.isCyclePos = data->x;

                cycleState = CC_WaitIn;
            }

            //if(cycleCnt.Confirm <= -2000.0)//Misjudge.
            //    cycleState = CC_Undefined;

            if((data->x - cycleConfig.isCyclePos) >= 3 *
cycleConfig.waitInDistance)//Misjudge.
            {
                cycleState = CC_Undefined;
            }

            break;

        case CC_WaitIn:

            cycleCnt.WaitIn--;
```

```
            if(Is_ArriveCycleInPoint(data,&cycleConfig))//Wait  for  in  cycle
point.
            {
                cycleState = CC_In;
            }

            if((data->x     -     cycleConfig.isCyclePos)     >=     5     *
cycleConfig.waitInDistance)//Misjudge.
            {
                cycleState = CC_Undefined;
            }

//            if(cycleCnt.WaitIn <= -2000.0)//Misjudge.
//            {
//                cycleState = CC_Undefined;
//            }

            break;


        case CC_In:

            cycleCnt.In--;

            cycleConfig.dYaw          =          data->attitude.yaw          -
cycleConfig.inCyclePointYaw;

            data->Bias = cycleConfig.bias;    //Give the servo a fixed angle
```

```
within a certain distance,to enter cycle.

            if(Is_CycleIn(data,&cycleConfig))
            {
                cycleState = CC_Tracking;//in cycle success.
            }

            if((data->x - cycleConfig.isCyclePos) >= 5 *
(cycleConfig.waitInDistance + cycleConfig.inDistance))//Misjudge.
            {
                cycleState = CC_Undefined;
            }

//            if(cycleCnt.In <= -2000)
//            {
//                cycleState = CC_Undefined;//in cycle fail.
//            }

            break;

        case CC_Tracking:

            cycleCnt.Tracking++;

            cycleConfig.dYaw = data->attitude.yaw -
cycleConfig.inCyclePointYaw;

            data->Bias = ((data->h_difference + data->v_difference) /
data->h_sum) * 100.0;
```

```
            data->Bias = data->Bias * 0.95 + fsign(cycleConfig.bias) * 100.0 *
0.05;

            data->Bias = ConstrainFloat(data->Bias,-100.0,100.0);

            if(Is_CycleOut(data,&cycleCnt,&cycleConfig)/*Normal*/           ||
cycleCnt.Tracking  >=  6000  ||  (data->x  -  cycleConfig.isCyclePos)  >=
2000.0/*Abnormal,force out*/)
            {
                cycleState = CC_Out;//out cycle.
            }

            break;

        case CC_Out:

            cycleCnt.Out++;

            cycleCnt.Wait = 2000;

            if(Is_CycleBackToStraight(data))/*Normal Out.*/
            {
                //back to straight.
                cycleState = CC_Wait;
                data->Element.Type = None;

                BLED.OFF(BLED.Self);
            }
```

```
            if(cycleCnt.Out >= 500 || cycleCnt.Tracking >= 6000 || (data->x -
cycleConfig.isCyclePos) >= 2000.0)//Force out of the cycle.
            {
                cycleState = CC_Undefined;
            }

            break;

        default:

            break;
    }

    data->CycleState = cycleState;
}

void RightAngle_Handler(data_t *data)
{
    static rightangle_state_t rightAngleState = RA_Wait;

    static rightangle_cnt rightAngleCnt = {0};

    static rightangle_config rightAngleConfig = {0};

    switch(rightAngleState)
    {
        case RA_Undefined:
```

```
    rightAngleState = RA_ExceptionHandler;

    break;

case RA_ExceptionHandler:

    RightAngleClearCnt(&rightAngleCnt);
    RightAngleClearConfig(&rightAngleConfig);
    GLED.OFF(GLED.Self);
    rightAngleState = RA_Wait;

    break;

case RA_Wait:

    if(data->Element.Type == RightAngle)
    {
        rightAngleState = RA_Config;

        goto RA_CONFIG;
    }

    break;

case RA_Config:

    RA_CONFIG:

    RightAngleClearCnt(&rightAngleCnt);
```

```
            RightAngleClearConfig(&rightAngleConfig);

            rightAngleCnt.Tracking = 100;

            rightAngleConfig.bias      =      fsign(data->v_difference)      *
100.0;//Determine the direction of the right angle.

            rightAngleState = RA_Confirm;

            goto RA_CONFIRM;

            break;

        case RA_Confirm:    //confirm and config some parameter.

            RA_CONFIRM:

            if(1)
            {
                GLED.ON(GLED.Self);

                rightAngleState = RA_Tracking;

                goto RA_TRACKING;
            }

            break;

        case RA_Tracking:
```

```
            RA_TRACKING:

            if((data->v_difference  >=  15.0)  &&  (data->v_sum  >=  15.0))
//for continuous right angle,update bias.
                rightAngleConfig.bias = fsign(data->v_difference) * 100.0;

            data->Bias = rightAngleConfig.bias;//give the servo max angle by
force.

            rightAngleCnt.Tracking--;

            if(Is_RightAngleOut(data,&rightAngleCnt,&rightAngleConfig))//if
find line,hand over to the control algorithm to patrol the line.
            {
                rightAngleState = RA_Out;
            }

            if(rightAngleCnt.Tracking <= -1000) //force out of rightangle.
            {
                rightAngleState = RA_Out;
            }

            break;

        case RA_Out:

            if(Is_RightAngleBackToStraight(data))
            {
```

```
                GLED.OFF(GLED.Self);
                rightAngleState = RA_Wait;
                data->Element.Type = None;
            }

            break;

        default:

            break;
    }
}

void LoseLine_Handler(data_t *data)
{
    static loseline_state_t loseLineState = LL_Wait;

    static loseline_config  loseLineConfig = {0};

    if(data->TrackingState == Normal_Tracking)
    {
        loseLineState = LL_Wait;
    }

    switch(loseLineState)
    {
        case LL_Undefined:

            loseLineState = LL_ExceptionHandler;
```

```
    break;

case LL_ExceptionHandler:

    LoseLineClearConfig(&loseLineConfig);

    loseLineState = LL_Wait;

    break;

case LL_Wait:

    if(data->TrackingState == LoseLine)
    {
        loseLineState = LL_Config;
    }

    break;

case LL_Config:

    LoseLineClearConfig(&loseLineConfig);
    loseLineConfig.bias = data->B[8];
    loseLineState = LL_Confirm;

    break;

case LL_Confirm:
```

```
    if(Is_LoseLine(data))
    {
        loseLineState = LL_SearchLine;
    }
    else
    {
        loseLineState = LL_Undefined;
    }

    break;

case LL_BackSearchLine:

    break;


case LL_SearchLine:

    data->Bias = loseLineConfig.bias;

    if(Is_SearchedLine(data))
    {
        loseLineState = LL_Searched;
    }

    break;

case LL_Searched:
```

```
            loseLineState = LL_Wait;
            data->TrackingState = Normal_Tracking;

            break;

        default:

            break;
    }
}

void Cross_Handler(data_t *data)
{
    static cross_state_t crossState = CS_Wait;

    static cross_cnt     crossCnt = {0};

    static cross_config  crossConfig = {0};

    switch(crossState)
    {

        case CS_Undefined:

            crossState = CS_ExceptionHandler;

            break;
```

```
case CS_ExceptionHandler:

    CrossClearCnt(&crossCnt);
    CrossClearConfig(&crossConfig);
    crossState = CS_Wait;

    break;

case CS_Wait:

    if(data->Element.Type == Cross)
    {
        BEEP.ON(BEEP.Self);

        crossState = CS_Config;
    }

    break;

case CS_Config:

    CrossClearCnt(&crossCnt);
    CrossClearConfig(&crossConfig);

    crossConfig.interval = 125;
    crossCnt.cnt = crossConfig.interval;

    crossState = CS_Confirm;
    break;
```

```
case CS_Confirm:

    crossCnt.cnt --;

    if(crossCnt.cnt <= 0)
    {
        crossCnt.cnt = crossConfig.interval;

        crossState = CS_In;

        BEEP.Toggle(BEEP.Self);
    }


    break;

case CS_In:
    crossCnt.cnt --;

    if(crossCnt.cnt <= 0)
    {
        crossCnt.cnt = crossConfig.interval;

        crossState = CS_Tracking;

        BEEP.Toggle(BEEP.Self);
    }
```

```
        break;

    case CS_Tracking:

        crossCnt.cnt --;


        if(crossCnt.cnt <= 0)
        {
            crossCnt.cnt = crossConfig.interval;

            crossState = CS_Out;

            BEEP.Toggle(BEEP.Self);
        }

        break;
    case CS_Out:

        crossCnt.cnt --;

        if(crossCnt.cnt <= 0)
        {
            crossCnt.cnt = crossConfig.interval;

            crossState = CS_Tracking;

            BEEP.Toggle(BEEP.Self);
```

```
                BEEP.OFF(BEEP.Self);


                crossState = CS_Wait;
                data->Element.Type = None;
            }



            break;

        default:

            break;
    }
}
```