第十五届“恩智浦”杯全国大学生

智能汽车竞赛

**技 术 报 告**



学 校：济南大学泉城学院

队伍名称：精神小伙队

参赛队员：张赛豪

孙鑫龙

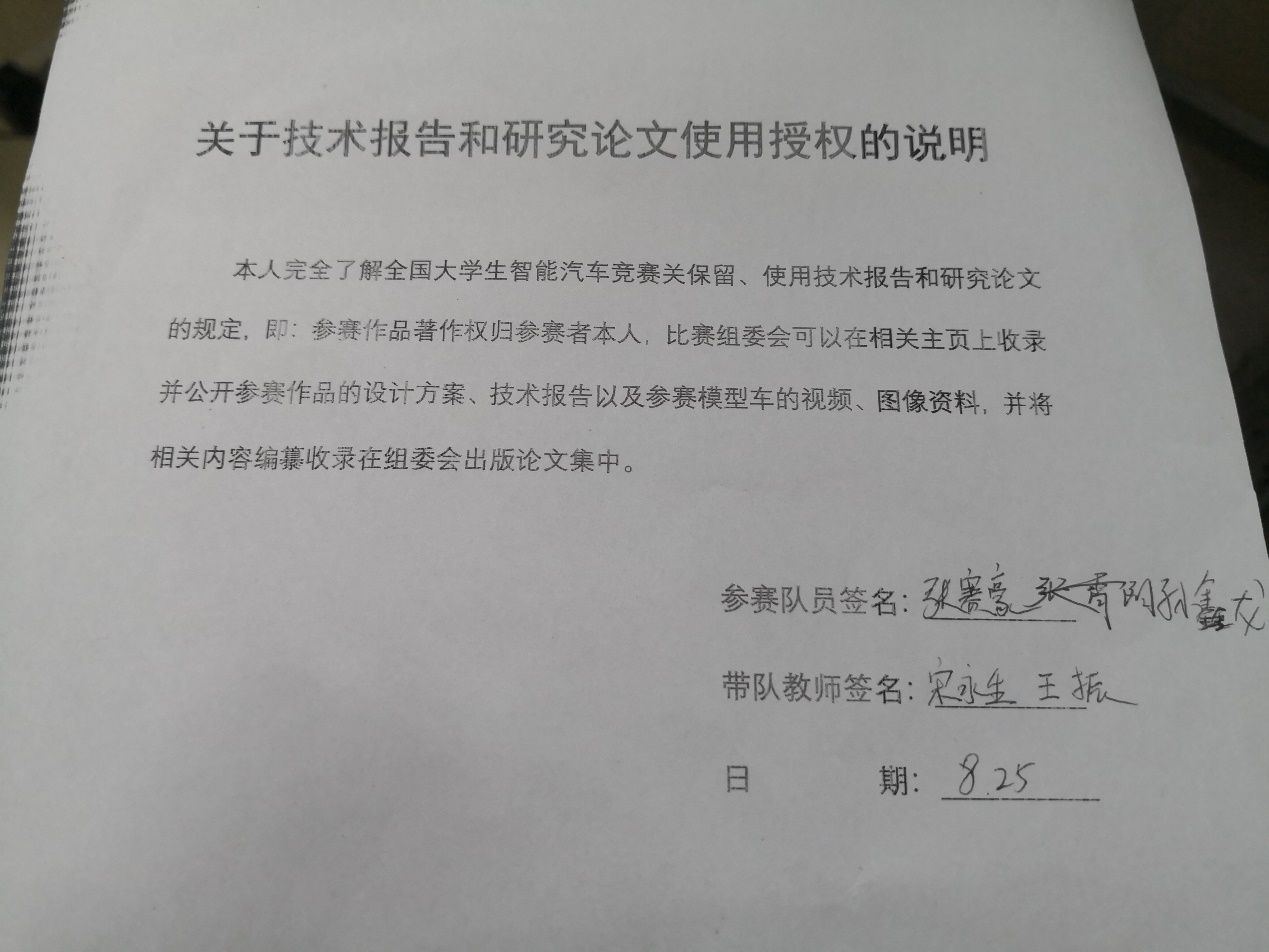
张霄阳

带队教师： 宋永生

王振

关于技术报告和研究论文使用授权的说明

本人完全了解第15届“恩智浦”杯全国大学生智能汽车竞赛关保留、使用技术报告和研究论文的规定，即：参赛作品著作权归参赛者本人，比赛组委会和恩智浦半导体公司可以在相关主页上收录并公开参赛作品的设计方案、技术报告以及参赛模型车的视频、图像资料，并将相关内容编纂收录在组委会出版论文集中。



# 摘要

本文详细介绍了济南大学泉城学院“精神小伙队”在第十五届“恩智浦”杯全国大学生智能汽车竞赛AI电磁组中的系统方案。本次比赛采用大赛组委会统一指定的C型车模，以飞思卡尔半导体公司生产的32位单片机MIMXRT1064DVL6A为核心控制器，要求智能车识别环岛、车库、坡道等多个元素，以最快速度完成整个比赛，智能车采用7个电感对赛道信息检测，根据提取连通域得到赛道中线；通过编码器检测智能车的实时速度，使用电感采集电磁信息，通过干簧管辅助判断车库元素；实现智能车在运动过程中速度控制；为了提高模型车的速度和稳定性，使用上位机调试工具，进行了大量硬件与软件测试。实验结果表明，该系统设计方案确实可行。

**关键词**：MIMXRT1064DVL6A；赛道处理：电感；

# 目录

[摘要 I](#_Toc16534047)

[目录 I](#_Toc16534048)

[第一章 绪 论 1](#_Toc16534049)

[1.1 智能车研究背景 1](#_Toc16534050)

[1.2 方案总体介绍 1](#_Toc16534051)

[1.3 本文结构 2](#_Toc16534052)

[第二章 系统机械设计方案 3](#_Toc16534053)

[2.1 整车布局 3](#_Toc16534054)

[2.2 倾角调节 3](#_Toc16534054)

2.2.1 前轮倾角调节…………………………………………………………………..4

2.2.2 前轮外倾………………………………………………………………………..4

2.2.3 主销后倾………………………………………………………………………..5

2.2.4 主销内倾………………………………………………………………………..5

[2.3 车整体机械结构 5](#_Toc16534055)

2.3.1 重心……………………………………………………………………………..5

2.3.2 轮胎处理………………………………………………………………………..5

[2.4 转向舵机安装 7](#_Toc16534056)

[第三章 硬件电路设计 8](#_Toc16534062)

[3.1 单片机系统设计 8](#_Toc16534063)

[3.2 电源模块设计 9](#_Toc16534064)

[3.3 电机驱动电路 1](#_Toc16534065)1

[3.4 电磁运放电路 11](#_Toc16534066)

[3.5 传感器的选择 12](#_Toc16534068)

[3.6.1 电感 12](#_Toc16534069)

[3.6.2 编码器 12](#_Toc16534070)

[3.7 主控板 13](#_Toc16534073)

[3.8 驱动板 14](#_Toc16534074)

[3.9 电磁一体板 15](#_Toc16534075)

[第四章 软件控制设计 16](#_Toc16534076)

[4.1 滤波算法介绍 16](#_Toc16534077)

[4.2 PID控制算法介绍 17](#_Toc16534084)

[4.2.1 位置式PID 18](#_Toc16534085)

[4.2.2 增量式PID 19](#_Toc16534086)

[4.2.3 PID参数整定 20](#_Toc16534087)

[4.3 转向舵机的神经网络控制算法 20](#_Toc16534088)

[4.4 驱动电机的PID控制算法 22](#_Toc16534089)

[第五章 系统调试 27](#_Toc16534090)

[5.1 开发调试工具 27](#_Toc16534091)

[5.2 蓝牙及上位机调试 27](#_Toc16534093)

[第六章 模型车主要技术参数说明 26](#_Toc16534094)

[结论 27](#_Toc16534095)

[参考文献 28](#_Toc16534096)

核心代码…………………………………………………………………………………………..29

# 第一章 绪 论

## 1.1 智能车研究背景

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。

在我国，教育部为了加强大学生实践、创新能力和团队合作精神的培养，委托教育部高等学校自动化专业教学指导分委员会主办了每年一度的全国大学生智能汽车竞赛。全国大学生智能汽车竞赛是在竞赛组委会提供的统一汽车模型平台上，使用飞思卡尔半导体公司的8位、16位、32位微控制器作为核心控制模块，通过设计道路识别传感器和电机驱动电路、编写相应软件及装配模型车，制作一个能够自主识别道路的模型汽车，按照规定路线（路线赛前未知）行进，以完成时间最短者为优胜。

## 1.2 方案总体介绍

比赛跑道为表面白色，两边有连续黑线作为引导线，黑线宽为25mm。车模通过采集赛道图像进行路径检测比赛规则限定了跑道宽道区宽度40cm，拐角最小半径65cm，并且规定了各赛道标志的具体指标。参赛队员的目标是使智能车按照规则以最短时间完成单圈赛道。

根据竞赛规则相关规定，本智能车系统采用大赛组委会统一提供的C型车模，以恩智浦公司生产的64位微控制器MIMXRT1064DVL6A作为核心控制器，在KEIL开发环境中进行软件开发通过电感将赛道信息输MIMXRT1064DVL6A微控制器，进行进一步处理获得主要的赛道信息；通过编码器来检测车速，并通过单片机脉冲计算获得速度和路程；转向舵机采用神经网络控制；驱动电机采用PID控制，通过PWM控制驱动电路调整电机的功率；而车速的目标值由默认值、运行安全方案和基于图像处理的优化策略进行综合控制；以提高断路元素的速度。

根据比赛的基本要求，我们设计了系统总体思路如图1.1所示。

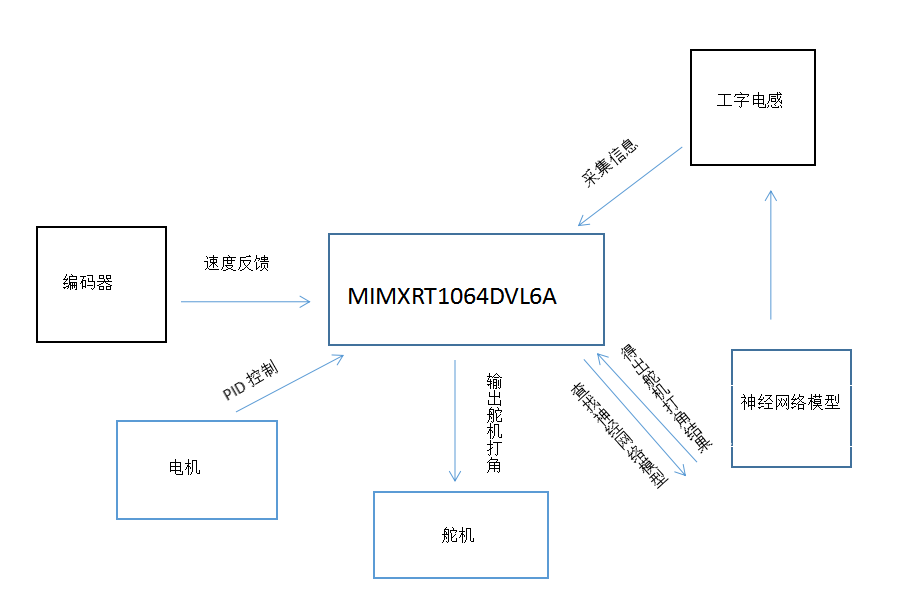


图1.1 系统结构图

## 1.3 本文结构

本文共分为六章。第一章主要是介绍了比赛的背景及智能车系统总体方案的介绍；第二章从智能车系统的机械结构出发，详细阐述了智能车系统各部分机械结构的安装和调整；第三章重点介绍了系统中所涉及的硬件设计方案和原理；第四章是介绍了智能系统的软件算法包括图像处理以及电机舵机的控制策略；第五章对调试过程中的一些手段进行了讲解；第六章则是对智能车一些物理参数进行了简单的汇报。

# 第二章 机械结构设计及调整

智能车的核心是控制策略和算法，但是，机械结构也是限制赛车速度的巨大瓶颈，如果一辆赛车的程序架构很好，但是机械部分做的不好的话，其速度也会被大大的限制。即当车速较高的时候，车模有明显的甩尾和侧滑现象，此时对车的机械结构要求很高。除了对车身姿态的影响外，机械性能影响车的加减速响应速度，运行的对称性和稳定性等，因此，我们在不违反规则的情况下对小车进行了多方面的改造以使小车具有良好的运行性能。

## 2.1 整车布局

我们组采用的是新C车模，对称性好，尺寸为290\*184\*196mm，轮胎尺寸为29\*61mm，通过对轮胎合适地填充海绵和进行软化处理，具有极好的减震性和耐磨性。。

(1) 车模底盘降低，用来降低重心。

(2) 舵机采用竖直姿态，方便控制。

(3) 对前轮倾角进行调节，保证车直线行驶的稳定性。

(4) 调整电池位置使中心尽量在车体中心。

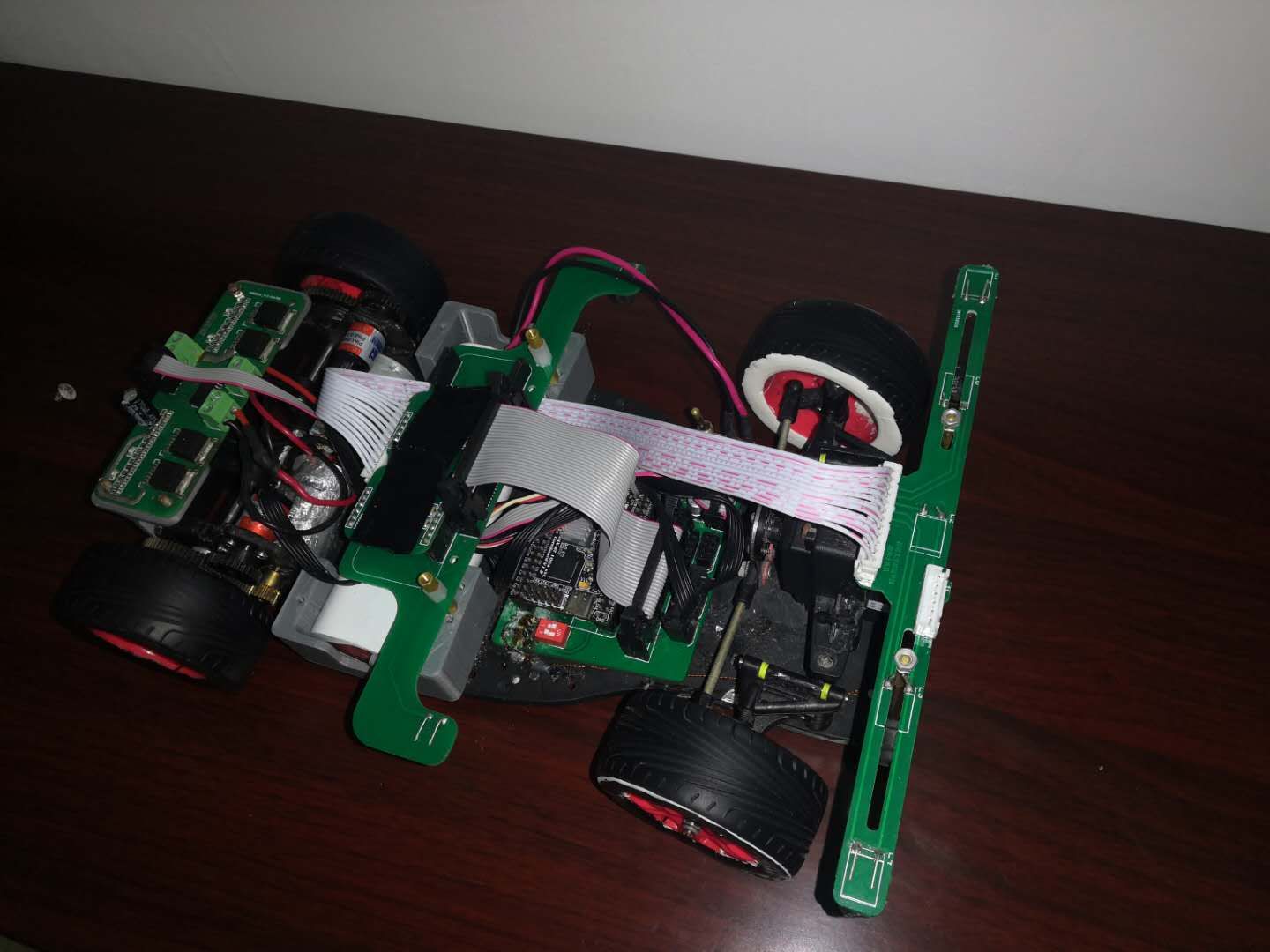


图2.1 智能车外形图

2.2 倾角调节

2.2.1 前轮倾角调节

为了使汽车直线行驶稳定，转向轻便，转向后能自动回正，减少轮胎和转向系零件的磨损等，在转向轮、转向节和前轴之间须形成一定的相对安装位置，叫车轮定位。本系统所采用的智能车通过四条轮胎与地面接触，两个后轮同轴受到限位，无法调整，与智能车的前进方向保持平行，因此要改变智能车与地面的接触方式，调试出利于车转向、直线的四轮定位，只能通过调整前轮倾角各定位参数来实现。它的安装位置由主销内倾、主销后倾、前轮外倾和前轮前束四个项目决定。其中前轮外倾一般使用默认角度，其余三个参数需要在理论的基础上还要参考运行状态进行调节。

2.2.2 前轮外倾

车轮外倾角的是使车轮与地面的动态承载中心得到合理的分配从而达到提高机械零件的使用寿命，减少轮胎的磨损转向轻便，使车轮紧靠轮毅内轴承，以减少外轴承及轮毅螺母的负荷等效果。

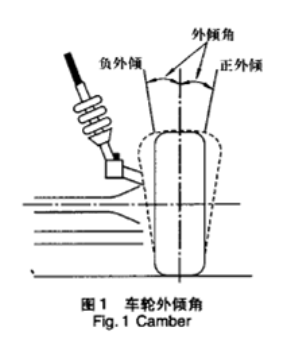


图1.1 前轮内倾

2.2.3 主销后倾

主销后倾如图1.2，是指在纵向平面内主销轴线与地面垂直线之间的夹角，向前为负，向后为正。它在车辆转弯时会产生与车轮偏转方向相反的回正力矩，使车轮自动恢复到原来的中间位置上。所以，主销后倾角越大，车速越高，前轮自动回正的能力就越强，但是过大的回正力矩会使车辆转向沉重。

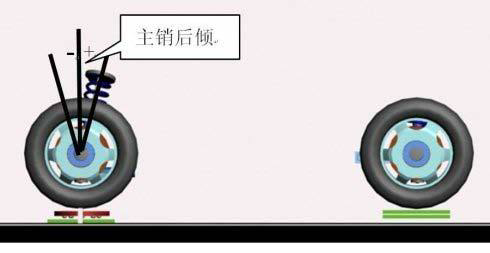


图1.2 主销后倾

2.2.4 主销内倾

主销内倾如图1.3，是指主销在前轴上安装，其上端略向内倾斜，于是主销轴线与地面垂线之间在汽车横向平面内形成一个夹角，称为主销内倾角。该内倾角一般不超过80，主要作用是使转向轮自动回正，转向操纵轻便。

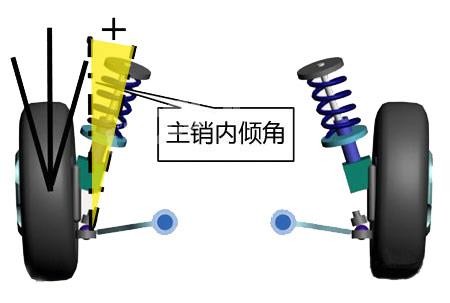


图1.3 主销内倾

2.3 车整体机械结构

2.3.1 重心

车辆重心的位置对于车辆来说尤为重要，重心的位置很大程度上影响车的转向时的姿态，加速，制动。重心过高极有可能引起侧翻，故尽可能把重心降低，我们把电池贴近底盘安装，尽可能降低重心。

2.3.2 轮胎处理

轮胎的处理在车模速度较低的情况下影响不会太大，但是假如是想要把速度提升到2.5m/s以上，轮胎就是不可避免的一个环节，再者c、d车模的轮胎相比起b、e车模轮胎质量相对更好，软化剂对于其效果不是非常明显。因此我们对于轮胎的处理也只是简单的在其上涂抹软化剂，对轮胎边缘用硅橡胶进行了密封。个人感觉轮胎的打磨不如通过车模在赛道上多跑得到的效果好，打磨时内部填充不够，因此主要打磨边缘部分，而在跑车时主要是轮胎中间提供足够的摩擦力。

## 

## 2.4 转向舵机安装

舵机转向是整个控制系统中延迟较大的一个环节，为了减小此时间常数，通过改变舵机的安装位置可以提高舵机的响应速度。通过分析舵机控制转向轮转向的原理可以发现，在相同的舵机转向条件下，转向连杆在舵机一端的连接点离舵机轴心距离越远，转向轮转向变化越快。这相当于增大力臂长度，提高线速度。

舵机安装方式有立式和卧式两种，比较两种方式发现，立式安装效果更好。 舵机安装时要保证左右对称，这样可以保证舵机左右转向时力臂相等且处于最 大范围，提升了舵机的响应速度。经理论分析，功率等于速度与扭矩的乘机，加 大转向速度必然减少输出扭矩，扭矩过小会造成迟钝，所以安装时必须考虑到转 向机构的响应速度与舵机扭矩之间的关系，获得最佳转向效果。经过实验，我们的舵机安装如图2.4所示。

# 第三章 硬件电路设计

硬件是基础，只有一个良好、稳定的硬件环境才能保证车能平稳快速的行驶。我们在整个系统设计过程中严格按照规范进行。本着可靠、高效的原则，在满足各个要求的情况下，尽量使设计的电路简单，PCB的效果简洁。

## 3.1 单片机系统设计

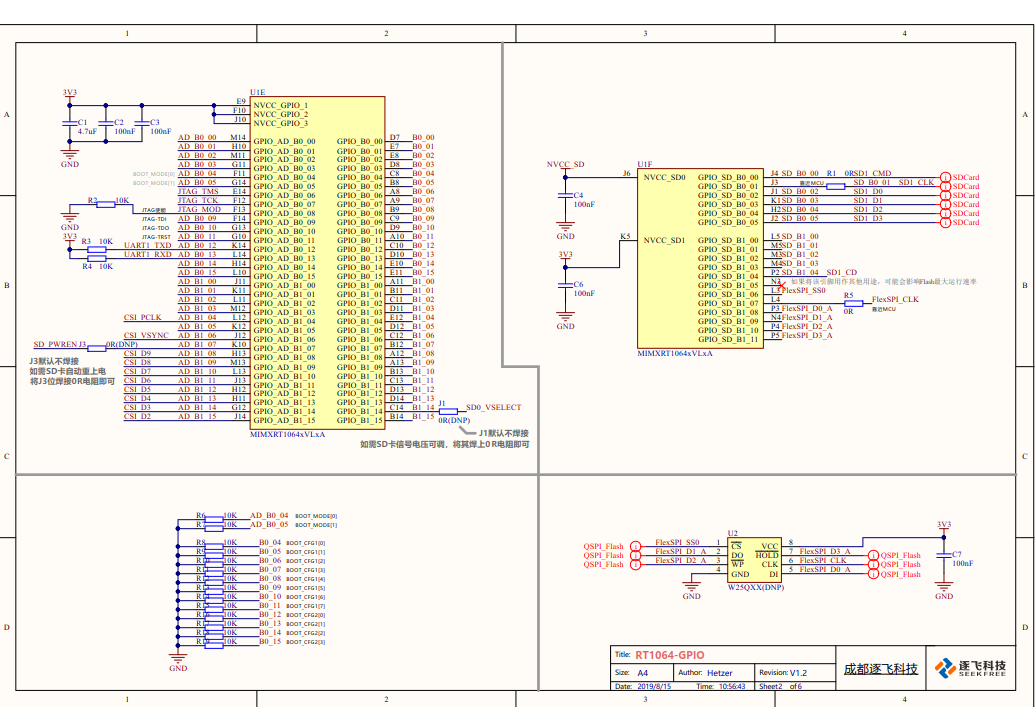
单片机最小系统是智能车系统的核心控制部件。我们采用了逐飞科技M7内核32位的MIMXRT1064DVL6A芯片最小系统。原理图如图3.1所示：

图3.1 单片机系统原理图

## 3.2 电源模块设计

电源模块为系统其他各个模块提供所需要的电源。设计中，除了需要考虑电压范围和电流容量等基本参数之外，还要在电源转换效率、降低噪声、防止干扰和电路简单等方面进行优化。可靠的电源方案是整个硬件电路稳定可靠运行的基础。

全部硬件电路的电源由2000mah镍镉电池提供（额定电压7.4V，满电电压8.4V）。由于电路中的不同电路模块所需要的工作电压和电流容量各不相同，因此电源模块应该包含多个稳压电路，将充电电池电压转换成各个模块所需要的电压。为满足需要，本车模上存在4种供电电压：

(1) 电机驱动使用镍铬电池供电，正常使用时电压在7.4～8.4V。可直接用于电机供电。

(2) 使用稳压芯片LM2940 输出电压5V，用于2.4G无限透传、核心板供电，蓝牙模块、驱动板缓存器等供电。原理图如图3.2.1所示

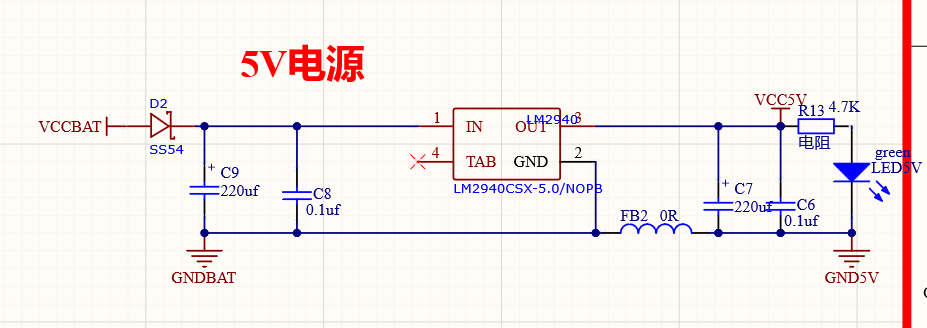


图3.2 转压电路电池电压——5V原理图

(3) 使用稳压芯片RT9013-33GB输出电压3.3V，用于12路运放（7+5）、蜂鸣器、编码器、按键等供电。

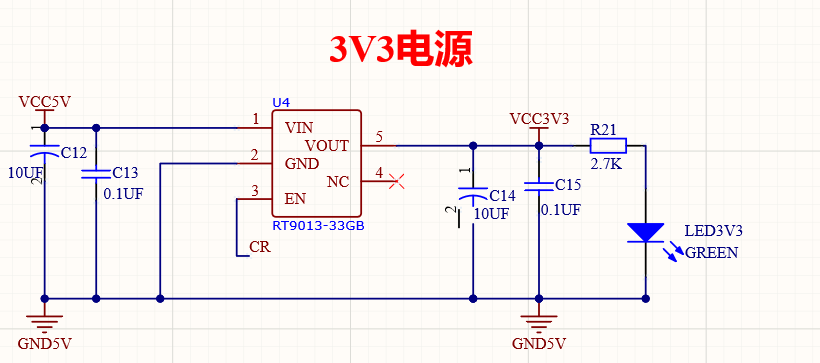


图3.3 转压电路5V-3.3V原理图

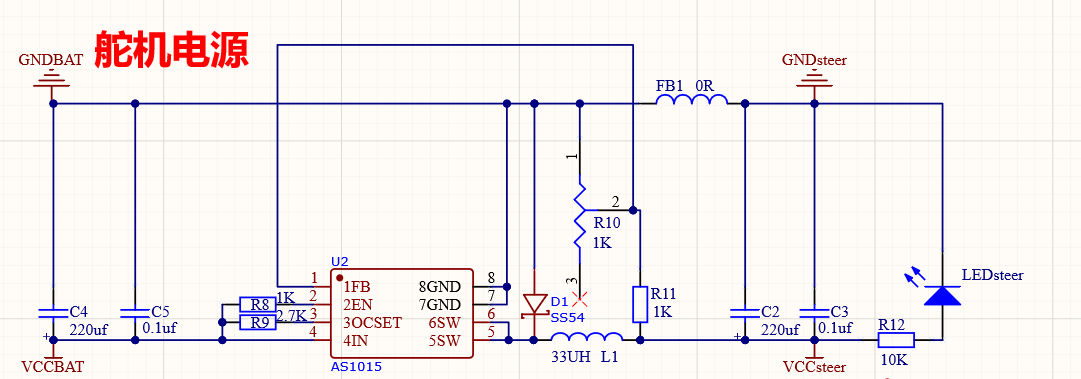
(4) 使用开关稳压芯片AS1015输出6V电压给舵机供电，用电池直接供电容易烧毁舵机。

图3.4 转压电路电池电压-6V原理图

## 3.3 电机驱动电路

在驱动芯片选择方面，我们选择BTN8960芯片，BTN8960芯片可以提高驱动能力，提供较大的驱动电流，并有硬件死区，防止同桥臂导通。使用两片BTN8960控制一路电机，如图3.5所示。

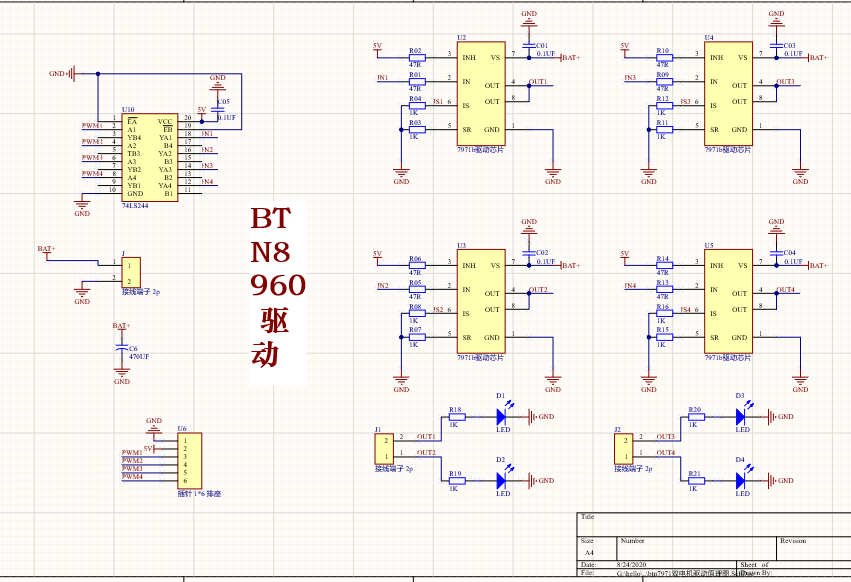


图3.5 BTN8960动原理图

## 3.4 电磁运放电路

采用简单的同向比例放大电路放大倍数，通过对电磁信号进行放大和检波处理和可以直接供单片机的AD口读取。运放芯片使用ad8618芯片

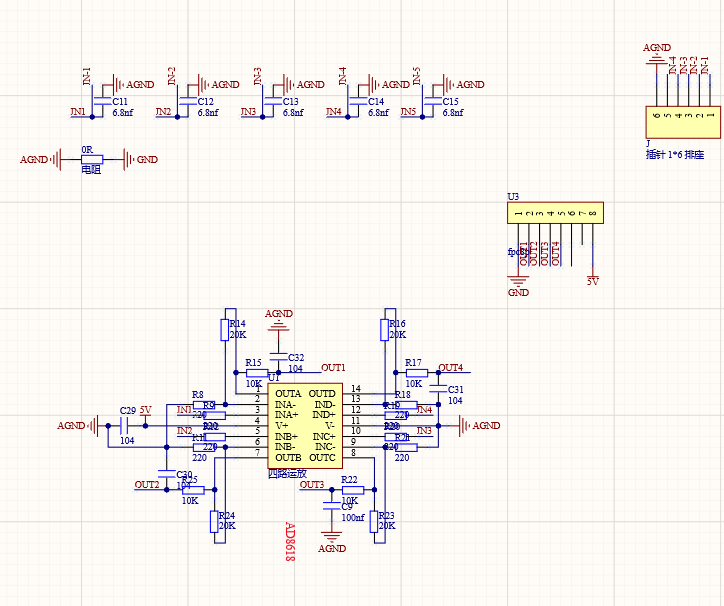


图3.6电磁信号检波放大原理图

## 3.5 传感器的选择

### 3.5.1 编码器

光电编码器是一种通过光电转换将输出轴上的机械几何位移量转换成脉冲或数字量的传感器, 这也是目前应用最多的测速传感器之一。其获取信息准确、精度高、应用简单。

采用增量式512线光电编码器，其供电电压为3.3V或5V，输出为小幅值的正弦信号。为了将此信号放大整形，设计了信号调理电路，其基本原理是使用一个运放做成比较器电路，调节参考电压，使输出变为0V-3.3V的方波信号，送入单片机进行运算。

### 3.5.2 工字电感

采用8\*10mh的黑色工字电感来采集车周围的赛道信息

## 3.6 主控板

在本系统中我们的电路板的制作主要思想是分立工作、追求简洁，便于电路板的调试和安装等工作。主控由电源稳压电路，与按键电路，单片机系统以及各个传感器接口等组成。实物图如图3.10所示



图3.10 主控板实物图

## 3.7 驱动板

驱动板在除了原有的电机驱动电路外，还增加了电流环模块、编码器接口，主要目的是方便编码器的连线，以减少编码器连线过长导致的接触不良等问题，驱动板的实物图如图3.11

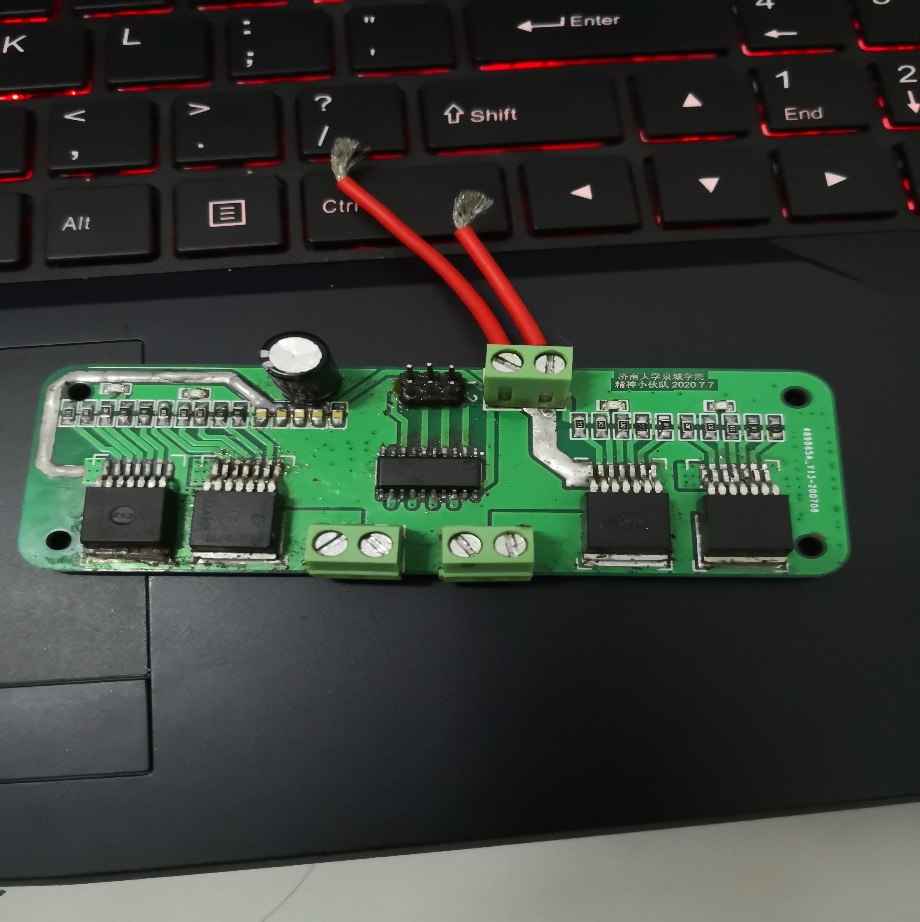


图3.11 驱动板实物图

## 3.8 电磁一体板

将电磁接口与运放画在一个板子上可以减少主控的元件数量，方便主控布局，使主控的电源性能良好，接口减少冲突，使用操作方便；同时缩短电磁信号与运放的走线至几毫米，减少走线或飞线阻抗造成的误差，电磁一体板的实物图如图3.12



图3.12电磁一体板实物图

# 第四章 软件控制设计

高效的软件程序是智能车高速平稳自动寻线的基础。我们设计的系统采用电感进行赛道识别，AD的采集及处理是整个软件的核心内容。在智能车的转向方面，我们使用神经网络控制算法。速度控制方面，我们使用增量式PID控制算法，使智能车能够稳定快速在赛道中循迹并完成比赛。

## 4.1滤波算法介绍

滤波算法采用了中值滤波，其中的排序算法采用了时间复杂度较低的归并排序，有效的降低了采集时间。

具体代码：

void merge\_sort(int ad[],int l,int r)

{

if(l >= r) return;

int mid = (l + r) >> 1;

merge\_sort(ad,l,mid),merge\_sort(ad,mid + 1 , r);

int i = l, j = mid + 1, k = 0;

while(i <= mid && j <= r)

{

if(ad[i] <= ad[j]) temp[k++] = ad[i++];

else temp[k++] = ad[j++];

}

while(i <= mid) temp[k++] = ad[i++];

while(j <= r) temp[k++] = ad[j++];

for(i = l, j = 0 ; i <= r ; i ++, j++) ad[i] = temp[j];

}

## 4.2PID控制算法介绍

在工程实际中，应用最为广泛的调节器控制规律为比例、积分、微分控制，简称PID控制，又称PID调节。PID控制器问世至今已有近70年历史，它以其结构简单、稳定性好、工作可靠、调整方便而成为工业控制的主要技术之一。当被控对象的结构和参数不能完全掌握，或得不到精确的数学模型时，控制理论的其它技术难以采用时，系统控制器的结构和参数必须依靠经验和现场调试来确定，这时应用PID控制技术最为方便。即当我们不完全了解一个系统和被控对象，或不能通过有效的测量手段来获得系统参数时，最适合用PID控制技术。PID控制，实际中也有PI和PD控制。

PID控制器是一种线性控制器，它根据给定值与实际输出值构成控制偏差。将偏差的比例(P)、积分(I)和微分(D)通过线性组合构成控制量，对被控对象进行控制，故称PID控制器，原理框图如图4.19所示。

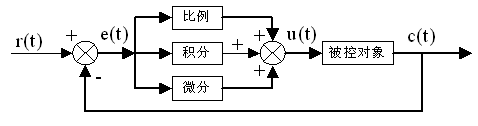


图4.19 PID控制器原理框图

在计算机控制系统中，使用的是数字PID控制器，控制规律为：

 (公式4.1)

 (公式4.2)

式中

k——采样序号，k = 0，1，2…； r(k)——第k次给定值；

c(k)——第k次实际输出值； u(k)—— 第k次输出控制量；

e(k)—— 第k次偏差； e(k-1)—— 第k-1次偏差；

KP——比例系数； TI——积分时间常数；

TD——微分时间常数； T——采样周期。

简单说来，PID控制器各校正环节的作用如下：

比例环节：及时成比例地反映控制系统的偏差信号，偏差一旦产生，控制器立即产生控制作用，以减少偏差。

积分环节：主要用于消除静差，提高系统的无差度。积分作用的强弱取决于积分时间常数，越大，积分作用越弱，反之则越强。

微分环节：能反映偏差信号的变化趋势(变化速率)，并能在该偏差信号变得太大之前，在系统中引入一个有效的早期修正信号，从而加快系统的动作速度，减小调节时间。

数字PID控制算法通常分为位置式PID控制算法和增量式PID控制算法。

### 4.2.1 位置式PID

位置式PID中，由于计算机输出的u (k) 直接去控制执行机构(如阀门)，u(k)的值和执行机构的位置(如阀门开度)是一一对应的，所以通常称公式(4.2)为位置式PID控制算法。

位置式PID控制算法的缺点是：由于全量输出，所以每次输出均与过去的状态有关，计算时要对过去e(k)进行累加，计算机工作量大；而且因为计算机输出的u(k)对应的是执行机构的实际位置，如计算机出现故障，u(k)的大幅度变化，会引起执行机构位置的大幅度变化，这种情况往往是生产实践中不允许的，在某些场合，还可能造成严重的生产事故。因而产生了增量式PID 控制的控制算法，所谓增量式PID 是指数字控制器的输出只是控制量的增量△u(k)。

### 4.2.2 增量式PID

当执行机构需要的是控制量的增量(例如：驱动步进电机)时，可由式(4.2)推导出提供增量的PID控制算式。由式(4.2)可以推出式(4.3)，式(4.2)减去式(4.3)可得式(4.4)。

 (公式4.3)

 (公式4.4)

式中；；

公式(4.4)称为增量式PID控制算法，可以看出由于一般计算机控制系统采用恒定的采样周期T，一旦确定了KP、TI 、TD，只要使用前后三次测量值的偏差，即可由式(4.4)求出控制增量。

增量式PID具有以下优点：

(1) 由于计算机输出增量，所以误动作时影响小，必要时可用逻辑判断的方法关掉。

(2) 手动/自动切换时冲击小，便于实现无扰动切换。此外，当计算机发生故障时，由于输出通道或执行装置具有信号的锁存作用，故能保持原值。

(3) 算式中不需要累加。控制增量△u(k)的确定仅与最近k次的采样值有关，所以较容易通过加权处理而获得比较好的控制效果。

但增量式PID也有其不足之处：积分截断效应大，有静态误差；溢出的影响大。使用时，常选择带死区、积分分离等改进PID控制算法。

### 4.2.3 PID参数整定

运用PID控制的关键是调整KP、KI、KD三个参数，即参数整定。PID参数的整定方法有两大类：一是理论计算整定法。它主要是依据系统的数学模型，经过理论计算确定控制器参数；二是工程整定方法，它主要依赖工程经验，直接在控制系统的试验中进行，且方法简单、易于掌握，在工程实际中被广泛采用。由于智能车系统是机电高耦合的分布式系统，并且要考虑赛道的具体环境，要建立精确的智能车运动控制数学模型有一定难度，而且我们对车身机械结构经常进行修正，模型参数变化较为频繁，理论计算整定法可操作性不强，最终我们采用了工程整定方法。此外，我们先后实验了几种动态改变PID参数的控制方法。

## 4.3 转向舵机的神经网络控制算法

首先，我们使用长的前瞻，采用传统的位置式PD控制让车跑起来，在车能够稳定的完成赛道后，使用串口通信将车周围7个电感的值和舵机的输出值发送到上位机中。将这些数据作为神经网络的输入层，输入到神经网络中，经过训练，得到数据集。我们发现只有loss降低至0.00x时模型才能够使用，并且我们数据采集的划分是30万数据是由车自行采集的，另外有60万数据是经过人手推车完成，最后将数据集使用NNCU模型转换器生成能够使用的模型。

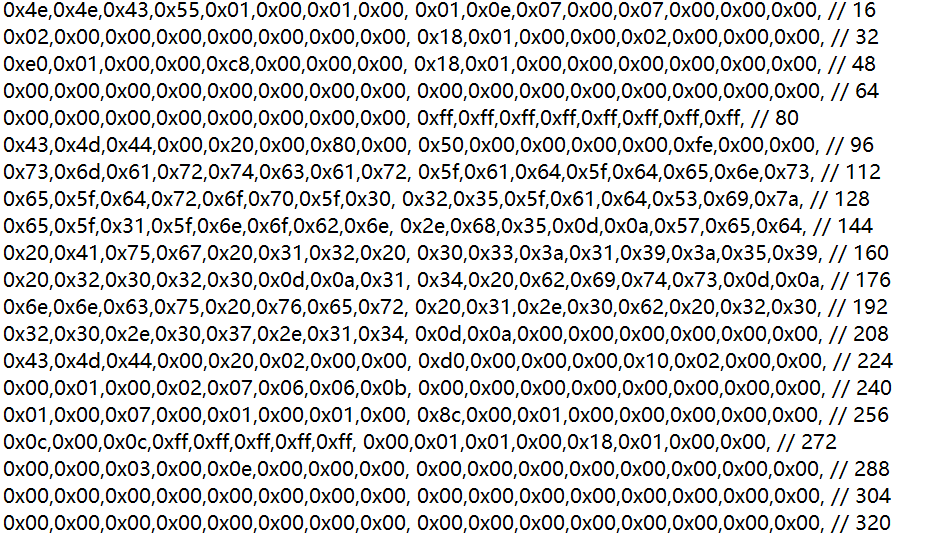


图4.3 部署完成的数据集

最后就是要用模型来实现对舵机的控制，具体代码：

CI\_RunModelXIP(model1, cie\_data, &temp);

int16\_t g\_servoValue;

if(inf.quantBits>7)

{

g\_servoValue = temp >> (inf.quantBits - pOM->fracBits - 1);

}

else

{

g\_servoValue = temp;

}

smotor\_angle = g\_servoValue \* (int32)420 / 128;//放大数据

smotor\_control();//舵机控制

## 4.4 驱动电机的PID控制算法

对于速度控制，我们采用了增量式PID控制算法，基本思想是直道加速，弯道减速。在实际测试中，我们发现小车直道和弯道相互过渡时加减速比较灵敏，与舵机转向控制配合得较好。

在程序中具体代码如下：

a = (int16)(40\*(smotor\_angle - 3750) \* 1.5 / 380);

if(a > 40) a = 40;

if(a < 40) a = 0;

motor\_dutyb = motor\_dutyb - (motor\_dutyb / 70) \* a \* 1; (公式4.5)

该方法相对于传统的差速模型，此方法是采集编码器的值并且将增量式PID的目标值设置为编码器的目标值，这样操作并不会影响电机的直道入弯道减速，并且这样在出弯时由于要补偿一个PWM值以到达减速轮的目标效果，所以出弯进入直道时加速很快，并且灵敏，这样就完成了一个较好的直道加速效果。

# 第五章 系统调试

## 5.1 开发调试工具

软件开发工具选用的是MDK5 for ARM。MDK 源自德国的 KEIL 公司，是 RealView MDK 的简称。在全球 MDK 被超过 10 万的嵌入式开发工程师使用。目前最新版本为：MDK5.14，该版本使用 uVision5 IDE 集成开发环境，是目前针对 ARM 处理器，尤其是 Cortex M 内核处理器的最佳开发工具。通过MDK5工具，可以大大节省软件调试时间。调试界面如图5.1所示：

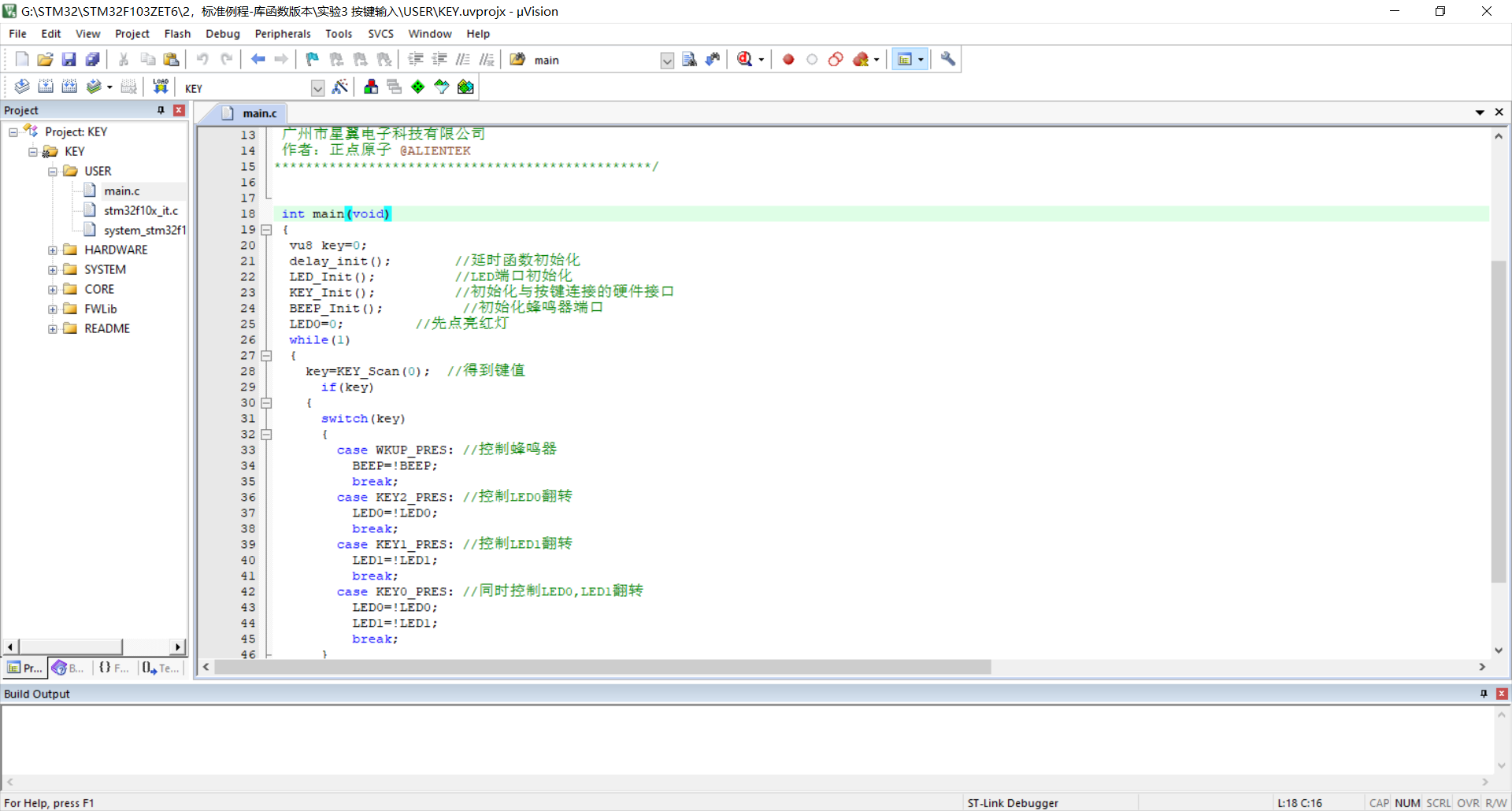


图5.1 KEIL调试界面

## 5.2 蓝牙及上位机调试

车在同样的赛道上走过的路都是不一样的，所以无论怎么考虑车的状态都是不够完全的，因此需要对运行中的车进行实时监控。为了解决这个问题，我们使用蓝牙模块配合上位机进行实时观测车模运行状态。如图5.2所示。

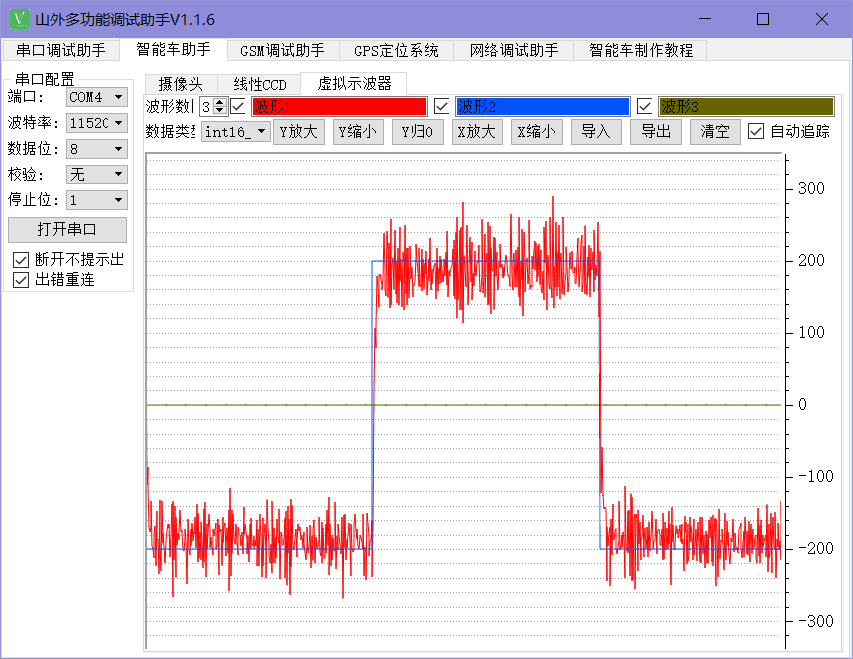


图5.2 上位机监测实时数值

# 第六章 模型车主要技术参数说明

|  |  |  |
| --- | --- | --- |
| 模型车基本参数 | 长 | 29.5cm |
| 宽 | 20cm |
| 高 | 8cm |
| 车重 | 1200g |
| 功耗 | 空载 | 10W |
| 带载 | 大于12W |
| 电容总容量 | | 2200uF |
| 传感器 | 编码器 | 2个 |
| 工字电感 | 7个 |
| 陀螺仪 | 0个 |
| 小型激光雷达 | 0个 |
| Openmv摄像头 | 0个 |
| 除了车模原有的驱动电机、舵机之外伺服电机个数 | | 0 |

# 结论

自报名参加“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，定下了现在这个设计方案。

在此份技术报告中，我们主要介绍了准备比赛时的基本思路，包括机械、电路以及最重要的控制算法的创新思想。在机械结构方面，我们分析了舵机转向系统的改进办法，对前轮倾角进行一系列的改动。在电路方面，我们以模块形式分类，在最小系统、主板、电机驱动、电池使用、电磁采集等模块分别设计，经过不断实验，最后决定了最终的电路图。在程序方面，我们使用C语言编程，利用开发工具调试程序，经过小组成员不断讨论、改进，终于设计出一套比较通用稳定的程序。在这套算法中，我们结合路况调整车速，做到直道加速、弯道减速，保证在最短时间内跑完全程。

在备战过程中，我们遇到了很多挫折，一次次的校内比赛见证我们这个小队艰难的蜕变。非常感谢哈工大智能车创新俱乐部这个像家一样的地方，同时也感谢在场地和经费方面都得到了学校和学院的大力支持，更要特别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指导老师、指导学长，同时也感谢比赛组委会能组织这样一项有意义的比赛。

# 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] 王淑娟，蔡惟铮，齐明．模拟电子技术基础 [M]．北京：高等教育出版社．2009

[3] 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.

[4] 张文春．汽车理论 [M]．北京．机械工业出版社．2005.

[5] 殷剑宏，吴开亚．[图论及其算法](http://202.204.50.68:8080/opac/item.php?marc_no=0000589913) [M] ．中国科学技术大学出版社，2003.

[6] [夏克俭](http://202.204.50.68:8080/opac/openlink.php?author=%CF%C4%BF%CB%BC%F3%B1%E0%D6%F8)．[数据结构及算法](http://202.204.50.68:8080/opac/openlink.php?title=%CA%FD%BE%DD%BD%E1%B9%B9%2B%CB%E3%B7%A8) [M] ．北京：[国防工业出版社](http://202.204.50.68:8080/opac/openlink.php?publisher=%B9%FA%B7%C0%B9%A4%D2%B5%B3%F6%B0%E6%C9%E7)， 2001.

[7] 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.

[8] 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.

# 核心代码

#include "ANO.h"

/\*\*\*\*\*\*\*\*\*\*为了匿名四轴上位机的协议定义的变量\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//cup为小端模式存储，也就是在存储的时候，低位被存在0字节，高位在1字节

#define BYTE0(dwTemp) (\*(char \*)(&dwTemp)) //取出int型变量的低字节

#define BYTE1(dwTemp) (\*((char \*)(&dwTemp) + 1)) // 取存储在此变量下一内存字节的内容，高字节

#define BYTE2(dwTemp) (\*((char \*)(&dwTemp) + 2))

#define BYTE3(dwTemp) (\*((char \*)(&dwTemp) + 3))

/\*

发送给上位机的数据帧定义

@桢头--功能字--长度--数据（一个或多个，具体看协议说明）-校验

@前2个字节为帧头0xAAAA

@第3个字节为帧ID，应设置为0xF1~0xFA中的一个

@第4个字节为报文数据长度(dlc)

@第5个字节开始到第5+dlc-1个字节为要传输的数据内容段，每个数据场为高字节在前，地字节在后

@第5+dlc个字节为CheckSum,为第1个字节到第5+dlc-1个字节所有字节的值相加后，保留结果的低八位作为CheckSum

\*/

/\*void data\_to\_ANO(void)

{ unsigned char Speedset1\_=255;

Speedset1=Speedset1/Speedset1\_;

unsigned char data\_to\_send[18] = {0};

unsigned char cnt = 0;

uint8 sc=0;

uint8 ac=0,i;

data\_to\_send[cnt++]=0xAA; //帧头：AAAA

data\_to\_send[cnt++]=0xFF;

data\_to\_send[cnt++]=0x01; //功能字：OXF2

data\_to\_send[cnt++]=0; //需要发送数据的字节数，暂时给0，后面在赋值。

data\_to\_send[cnt++] = BYTE1(Speedset1\_);//取data[0]数据的高字节，

data\_to\_send[cnt++] = BYTE0(Speedset1\_);

data\_to\_send[3]= cnt-4;//计算总数据的字节数。

for(i=0;i<data\_to\_send[3]+4;i++) //对于for语句，当不写大括号的时候，只执行到下面第一个分号结束。

{

sc+=data\_to\_send[i];

ac+=sc;

}

data\_to\_send[cnt++] =sc; //计算校验位

data\_to\_send[cnt++] =ac;

uart\_putbuff(USART\_8,data\_to\_send,8);

}\*/

void data\_to\_ANO(void)

{ // int aaa=100;

unsigned char set\_speed\_=1;

unsigned char MotorPwm1\_=255;

unsigned char iError1\_=1;

set\_speed\_=set\_speed/set\_speed\_;

MotorPwm1\_=motor\_duty/MotorPwm1\_;

iError1\_=ek/iError1\_;

uart\_putchar(USART\_8,0x03);

uart\_putchar(USART\_8,0xFC);

uart\_putchar(USART\_8,BYTE0(ek));

uart\_putchar(USART\_8,BYTE0(MotorPwm1\_));

uart\_putchar(USART\_8,BYTE0(set\_speed\_));

uart\_putchar(USART\_8,0xFC);

uart\_putchar(USART\_8,0x03);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* COPYRIGHT NOTICE

\* Copyright (c) 2019,逐飞科技

\* All rights reserved.

\* 技术讨论QQ群：一群：179029047(已满) 二群：244861897

\*

\* 以下所有内容版权均属逐飞科技所有，未经允许不得用于商业用途，

\* 欢迎各位使用并传播本程序，修改内容时必须保留逐飞科技的版权声明。

\*

\* @file isr

\* @company 成都逐飞科技有限公司

\* @author 逐飞科技(QQ3184284598)

\* @version 查看doc内version文件 版本说明

\* @Software IAR 8.3 or MDK 5.28

\* @Target core NXP RT1064DVL6A

\* @Taobao https://seekfree.taobao.com/

\* @date 2019-04-30

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include "headfile.h"

#include "isr.h"

void CSI\_IRQHandler(void)

{

CSI\_DriverIRQHandler(); //调用SDK自带的中断函数 这个函数最后会调用我们设置的回调函数

\_\_DSB(); //数据同步隔离

}

float ad1,ad2,ad3,ad4,ad5,ad6,ad7;

float error =0;

float set\_out = 0;

float actual\_in = 0;

float err =0 ;

float invole = 0;

float actual\_out = 0;

float kp = 0.5;// 0.5

float ki = 0;

float kd = 0;

float err\_last = 0;

int collection\_success = 0;

void ad\_collection(void)

{

ad2 = adc\_convert(ADC\_1,ADC1\_CH13\_B24);

//ad3 = lvbo(ADC\_1,ADC1\_CH10\_B21);

//ad4 = lvbo(ADC\_1,ADC1\_CH9\_B20);

ad5 = adc\_convert(ADC\_2,ADC2\_CH1\_B28);

ad1 = adc\_convert(ADC\_2,ADC2\_CH3\_B30);

ad3 = adc\_convert(ADC\_1,ADC1\_CH0\_B27);

ad4 = adc\_convert(ADC\_1,ADC1\_CH11\_B22);

collection\_success = 1;

}

float PID\_place\_duty(float steer)

{

set\_out=0.0;

actual\_in=steer;

err=set\_out-actual\_in;

invole+=err;

actual\_out=kp\*err+ki\*invole+kd\*(err-err\_last);

err\_last=err;

return actual\_out;

}

void PIT\_IRQHandler(void)

{

if(PIT\_FLAG\_GET(PIT\_CH0))

{

PIT\_FLAG\_CLEAR(PIT\_CH0);

}

ad\_collection();

if(ad1 + ad2 == 0)

{

error = 0;

}

else

error = (ad2 - ad1)/(ad1 + ad2);

//PID\_init();

error = PID\_place\_duty(error);

if(PIT\_FLAG\_GET(PIT\_CH1))

{

PIT\_FLAG\_CLEAR(PIT\_CH1);

}

if(PIT\_FLAG\_GET(PIT\_CH2))

{

PIT\_FLAG\_CLEAR(PIT\_CH2);

}

if(PIT\_FLAG\_GET(PIT\_CH3))

{

PIT\_FLAG\_CLEAR(PIT\_CH3);

}

\_\_DSB();

}

void GPIO2\_Combined\_16\_31\_IRQHandler(void)

{

if(GET\_GPIO\_FLAG(C16))

{

CLEAR\_GPIO\_FLAG(C16);//清除中断标志位

}

}

void GPIO2\_Combined\_0\_15\_IRQHandler(void)

{

if(GET\_GPIO\_FLAG(MT9V03X\_VSYNC\_PIN))

{

//不用清除标志位，标志位在mt9v03x\_vsync函数内部会清除

if(1 == flexio\_camera\_type)mt9v03x\_vsync();

}

if(GET\_GPIO\_FLAG(SCC8660\_VSYNC\_PIN))

{

//不用清除标志位，标志位在scc8660\_vsync函数内部会清除

if(2 == flexio\_camera\_type)scc8660\_vsync();

}

}

/\*

中断函数名称，用于设置对应功能的中断函数

Sample usage:当前启用了周期定时器中断

void PIT\_IRQHandler(void)

{

//务必清除标志位

\_\_DSB();

}

记得进入中断后清除标志位

CTI0\_ERROR\_IRQHandler

CTI1\_ERROR\_IRQHandler

CORE\_IRQHandler

FLEXRAM\_IRQHandler

KPP\_IRQHandler

TSC\_DIG\_IRQHandler

GPR\_IRQ\_IRQHandler

LCDIF\_IRQHandler

CSI\_IRQHandler

PXP\_IRQHandler

WDOG2\_IRQHandler

SNVS\_HP\_WRAPPER\_IRQHandler

SNVS\_HP\_WRAPPER\_TZ\_IRQHandler

SNVS\_LP\_WRAPPER\_IRQHandler

CSU\_IRQHandler

DCP\_IRQHandler

DCP\_VMI\_IRQHandler

Reserved68\_IRQHandler

TRNG\_IRQHandler

SJC\_IRQHandler

BEE\_IRQHandler

PMU\_EVENT\_IRQHandler

Reserved78\_IRQHandler

TEMP\_LOW\_HIGH\_IRQHandler

TEMP\_PANIC\_IRQHandler

USB\_PHY1\_IRQHandler

USB\_PHY2\_IRQHandler

ADC1\_IRQHandler

ADC2\_IRQHandler

DCDC\_IRQHandler

Reserved86\_IRQHandler

Reserved87\_IRQHandler

GPIO1\_INT0\_IRQHandler

GPIO1\_INT1\_IRQHandler

GPIO1\_INT2\_IRQHandler

GPIO1\_INT3\_IRQHandler

GPIO1\_INT4\_IRQHandler

GPIO1\_INT5\_IRQHandler

GPIO1\_INT6\_IRQHandler

GPIO1\_INT7\_IRQHandler

GPIO1\_Combined\_0\_15\_IRQHandler

GPIO1\_Combined\_16\_31\_IRQHandler

GPIO2\_Combined\_0\_15\_IRQHandler

GPIO2\_Combined\_16\_31\_IRQHandler

GPIO3\_Combined\_0\_15\_IRQHandler

GPIO3\_Combined\_16\_31\_IRQHandler

GPIO4\_Combined\_0\_15\_IRQHandler

GPIO4\_Combined\_16\_31\_IRQHandler

GPIO5\_Combined\_0\_15\_IRQHandler

GPIO5\_Combined\_16\_31\_IRQHandler

WDOG1\_IRQHandler

RTWDOG\_IRQHandler

EWM\_IRQHandler

CCM\_1\_IRQHandler

CCM\_2\_IRQHandler

GPC\_IRQHandler

SRC\_IRQHandler

Reserved115\_IRQHandler

GPT1\_IRQHandler

GPT2\_IRQHandler

PWM1\_0\_IRQHandler

PWM1\_1\_IRQHandler

PWM1\_2\_IRQHandler

PWM1\_3\_IRQHandler

PWM1\_FAULT\_IRQHandler

SEMC\_IRQHandler

USB\_OTG2\_IRQHandler

USB\_OTG1\_IRQHandler

XBAR1\_IRQ\_0\_1\_IRQHandler

XBAR1\_IRQ\_2\_3\_IRQHandler

ADC\_ETC\_IRQ0\_IRQHandler

ADC\_ETC\_IRQ1\_IRQHandler

ADC\_ETC\_IRQ2\_IRQHandler

ADC\_ETC\_ERROR\_IRQ\_IRQHandler

PIT\_IRQHandler

ACMP1\_IRQHandler

ACMP2\_IRQHandler

ACMP3\_IRQHandler

ACMP4\_IRQHandler

Reserved143\_IRQHandler

Reserved144\_IRQHandler

ENC1\_IRQHandler

ENC2\_IRQHandler

ENC3\_IRQHandler

ENC4\_IRQHandler

TMR1\_IRQHandler

TMR2\_IRQHandler

TMR3\_IRQHandler

TMR4\_IRQHandler

PWM2\_0\_IRQHandler

PWM2\_1\_IRQHandler

PWM2\_2\_IRQHandler

PWM2\_3\_IRQHandler

PWM2\_FAULT\_IRQHandler

PWM3\_0\_IRQHandler

PWM3\_1\_IRQHandler

PWM3\_2\_IRQHandler

PWM3\_3\_IRQHandler

PWM3\_FAULT\_IRQHandler

PWM4\_0\_IRQHandler

PWM4\_1\_IRQHandler

PWM4\_2\_IRQHandler

PWM4\_3\_IRQHandler

PWM4\_FAULT\_IRQHandler

Reserved171\_IRQHandler

GPIO6\_7\_8\_9\_IRQHandler\*/

#include "headfile.h"

#include "fsl\_device\_registers.h"

#include "fsl\_debug\_console.h"

#include "board.h"

#include "isr.h"

#include "pin\_mux.h"

#include "clock\_config.h"

#include "nncie.h"

#define IMAGE\_SIZE (8 + 32\*32\*3) // 3080

//extern void\* RunModel(const void \*in\_buf);

//int16 g\_out[1];

#define SYSTICK\_PRESCALE 10

uint8 send\_buff[20]; //wuxian chuankou?

float smotor\_center = 3750;

#define SMOTOR\_RANGE 400

float smotor\_angle;

float smotor\_end;

void smotor\_control(void)

{

//smotor\_angle = (int16)limit(smotor\_angle,SMOTOR\_RANGE);

if(smotor\_angle > 4130 )

{

pwm\_duty(PWM4\_MODULE3\_CHA\_C31,4130);

}

else if(smotor\_angle < 3370)

{

pwm\_duty(PWM4\_MODULE3\_CHA\_C31,3370);

}

pwm\_duty(PWM4\_MODULE3\_CHA\_C31,smotor\_angle);

//pwm\_init(PWM4\_MODULE3\_CHA\_C31,50,smotor\_angle);

}

int8 cie\_data[7];

void cie\_data\_get(void)

{

cie\_data[0] = ad5;

cie\_data[1] = ad1;

cie\_data[2] = ad3;

cie\_data[3] = ad4;

cie\_data[4] = ad2;

cie\_data[5] = 0;

cie\_data[6] = 0;

}

int main(void)

{

int16 temp;

//chu shi hua

DisableGlobalIRQ();

board\_init();

pit\_init();

pit\_interrupt\_ms(PIT\_CH0,20);

//systick\_delay\_ms(100);

gpio\_init(B9,GPO,1,GPIO\_PIN\_CONFIG);

//chuan kou

seekfree\_wireless\_init();

adc\_init(ADC\_1,ADC1\_CH13\_B24,ADC\_10BIT);

adc\_init(ADC\_2,ADC2\_CH1\_B28,ADC\_10BIT);

adc\_init(ADC\_2,ADC2\_CH3\_B30,ADC\_10BIT);

adc\_init(ADC\_1,ADC1\_CH0\_B27,ADC\_10BIT);

adc\_init(ADC\_1,ADC1\_CH11\_B22,ADC\_10BIT);

//adc\_init(ADC\_1,ADC1\_CH13\_B24,ADC\_8BIT);

//adc\_init(ADC\_1,ADC1\_CH14\_B25,ADC\_8BIT);

ModelInfo\_t inf;

CI\_GetModelInfoXIP(model1, &inf);

pwm\_init(PWM4\_MODULE3\_CHA\_C31,50,3750);

pwm\_init(PWM1\_MODULE3\_CHB\_D1 , 15000, 13500); //14500

pwm\_init(PWM1\_MODULE3\_CHA\_D0 , 15000, 0);

pwm\_init(PWM2\_MODULE3\_CHB\_D3 , 15000, 13500); //14500

pwm\_init(PWM2\_MODULE3\_CHA\_D2 , 15000, 0);

EnableGlobalIRQ(0);

SysTick\_Config(CLOCK\_GetFreq(kCLOCK\_CoreSysClk) / (SYSTICK\_PRESCALE \* 10000U));

const CI\_OutMetricInfo\* pOM = CI\_GetOutMetricInfo(0);

//CI\_GetModelInfoXIP(model1, &inf);

while(1)

{

smotor\_angle = (smotor\_center / 1000 + error)\*1000;

//error\_pre = error;

smotor\_control();

if(collection\_success)

{

collection\_success = 0;

cie\_data\_get();

CI\_RunModelXIP(model1, cie\_data, &temp);

int16\_t g\_servoValue;

if(inf.quantBits>7)

{

//int16\_t \*g\_out16 = (int16\_t \*)g\_out;

g\_servoValue = temp >> (inf.quantBits - pOM->fracBits - 1);

//g\_servoValue = temp>>3;

}

else

{

g\_servoValue = temp;

}

smotor\_angle = g\_servoValue \* (int32)320 / 128;//放大数据

smotor\_control();

}

}

}

#include "common.h"

#include "isr.h"

float mkp,mki,mkd = 0;

float out\_increment = 0;

float out = 0;

int16 speed = 0;

int16 set\_speed = 75;//45

int16 ek,ek1,ek2 = 0;

int32 motor\_duty = 0;

float mkpb,mkib,mkdb = 0;

float out\_incrementb = 0;

float outb = 0;

int16 speedb = 0;

int16 set\_speedb = 75;

int16 ekb,ek1b,ek2b = 0;

int32 motor\_dutyb = 0;

void getmotorpid(void)

{

speed = encoder1;

speedb = encoder2;

ek2 = ek1;

ek2b = ek1b;

ek1 = ek;

ek1b = ekb;

ek = set\_speed - speed;

ekb = set\_speedb - speedb;

mkp = 30;//30

mki = 2.5;//2

mkd = 50; //8

mkpb = 30;

mkib = 2.5;

mkdb = 50;

out\_increment = (int16)(mkp\*(ek-ek1) + mki\*ek + mkd\*(ek - 2\*ek2 + ek2));

out\_incrementb = (int16)(mkpb\*(ekb-ek1b)+mkib\*ekb+mkdb\*(ekb -2\*ek2b+ek2b));

out += out\_increment;

outb += out\_incrementb;

if(out > 50000) out = 50000;

if(outb > 50000) outb = 50000;

motor\_duty = (int32)out;

motor\_dutyb = (int32)outb;

}