第十三届“恩智浦”杯全国大学生

智能汽车竞赛

技 术 报 告

学　　校：成都信息工程大学

队伍名称：成信 WD 队

参赛队员：穆天增

向科治

李卓

带队教师：邓昌建

关于技术报告和研究论文使用授权的说明

本人完全了解第 13 届“恩智浦”杯全国大学生智能汽车竞赛关保留、使用技术报告和研究论文的规定，即：参赛作品著作权归参赛者本人，比赛组委会和恩智浦半导体公司可以在相关主页上收录并公开参赛作品的设计方案、技术报告以及参赛模型车的视频、图像资料，并将相关内容编纂收录在组委会出版论文集中。

参赛队员签名：穆天增、向科治、李卓

带队教师签名：邓昌建

日　　期：2018-8-18

摘 要

本文设计的智能车系统以KEA128微控制器为核心控制单元，通过10mh的电感检测赛道信息，通过软件对电感值进行处理，提取电磁线偏差，用于赛道识别；通过编码器检测模型车的实时速度，使用PID、ADRC控制算法调节驱动左右电机的转速，实现了对车运动速度和运动方向的闭环控制。为了提高模型车的速度和稳定性，使用自制上位机、OLED模块、键盘模块等调试工具，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。

目录

引　　言

第十三届全国大学生“恩智浦”杯智能汽车竞赛是以“立足培养、重在参与、鼓励探索、追求卓越”为宗旨，鼓励创新的移向科技竞赛活动，竞赛分为光电四轮组、两轮直立组、三轮电磁组、节能组、双车组、信标组六个组别，两轮直立组要求在规定的汽车模型平台上，使用恩智浦半导体公司的微控制器作为核心控制模块，通过增加姿态传感器、道路识别传感器、电机驱动模块以及编写相应的控制程序，制作完成一个可以保持两轮直立姿态，能够自主识别道路的平衡车。智能汽车竞赛的赛道路面为宽度不小于 45cm 的白色面板，赛道中间有一根通有 100MA，20KHZ 的交流电的导线作为引导线，参赛队员的目标是模型汽车需要按照规则以最短的时间完成单圈赛道。本次比赛中，我们使用大赛组委会统一提供的 D 车模，采用恩智浦 KEA128 单片机为核心控制器，采用 MOS 桥驱动车模电机，MPU6050 作为姿态传感器， 10MH 电感作为道路识别传感器，自主构思姿态控制，速度控制和转向控制方案，引导车模按照规定路线识别行进。

在这份报告中，我们主要通过对整体方案、机械、硬件、算法等方面的介绍，详细阐述我队在此次智能汽车竞赛中的思想和创新。具体表现在电路的创新设计、算法以及辅助调试模块等方面的创新。我队成员涉及自动化、机械、计算机等专业，在准备比赛的过程中，队员查阅了大量的专业资料，反复地调试汽车模型的各项参数。所有队员都为此次智能汽车竞赛付出了艰苦的劳动。

第一章　方案设计

1.1. 系统概述

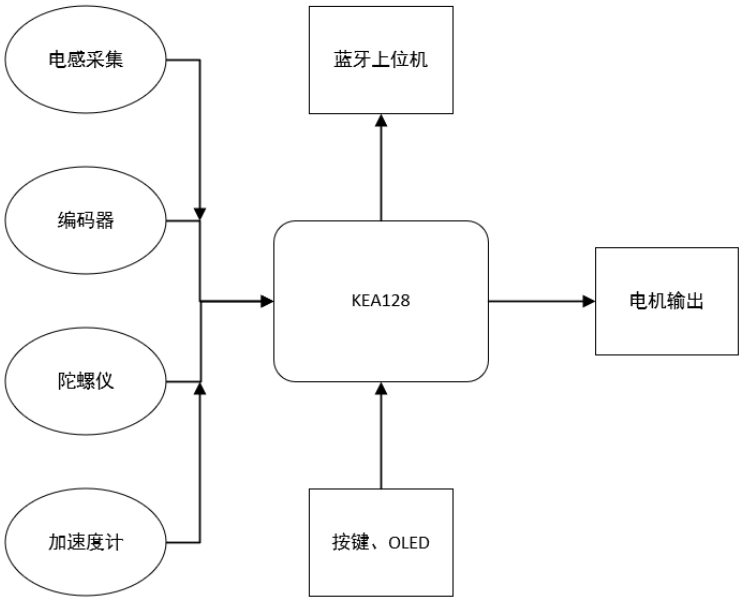

图 1.1 系统框图

电磁直立车的系统整体结构如图所示。KEA128 微处理器通过三个电感采集后，通过差比和得到车体和赛道中线的偏差信息；通过采集陀螺仪数据分析计算过桥信息；通过采集编码器对车轮转速的脉冲计数，得到车行进的速度数据。通过微处理器对电感偏差处理，对角度、速度进行 PID 控制，最后 PWM 波输出驱动电机。在调试过程中，通过蓝牙传输给上位机观察电感采集的偏差、车身角度等实时数据，便于了解车的实时状态控制进行调试。

第二章　智能车机械结构调整与优化

智能汽车各系统的控制都是在机械结构的基础上实现的，因此在设计整个软件架构和算法之前一定要对整个模型车的机械结构有一个全面清晰的认识，然后建立相应的数学模型，从而再针对具体的设计方案来调整赛车的机械结构，并在实际的调试过程中不断的改进优化和提高结构的稳定性。本章将主要介绍智能汽模型车型车的机械结构和调整方案。

2.1. 智能汽车车体机械建模

此次竞赛选用的是东莞博思公司生产的智能车竞赛专用模型车(D1　型模型车)，配套的电机型号为RS-380。智能车的控制采用的是双后轮驱动方案。智能车的外形大致如下：

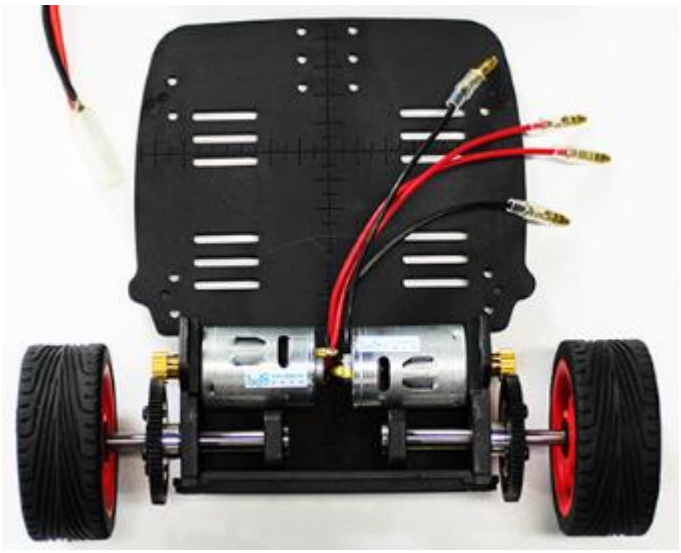

图 2.1 D 车模外形图

2.2. 智能汽车传感器的安装

车模中的传感器包括有：速度传感器，车模姿态传感器（陀螺仪、加速度计）以及光电传感器（摄像头）。下面分别介绍这些传感器的安装。

2.2.1. 速度传感器的安装

速度传感器我们使用了迷你编码器，安装在D车模设计的编码器固定位置，安装可靠性高，且便于安装和拆卸。

图 2.2 编码器安装图

2.2.2. 电感的安装

电感是小车获取路径信息的传感器，需要一个稳定的安装方式以获得更稳定的数据，我们采用塑料支架，并用空心 3K 碳纤维杆做前瞻，以尽可能减小前面的重量。另外我们还用斜撑杆对主杆进行了固定，用 AB 胶对车模的后板和主杆进行固定，这样能使主杆与车模完全固定在一起。下面是前瞻图：

图 2.3 电磁传感器安装图

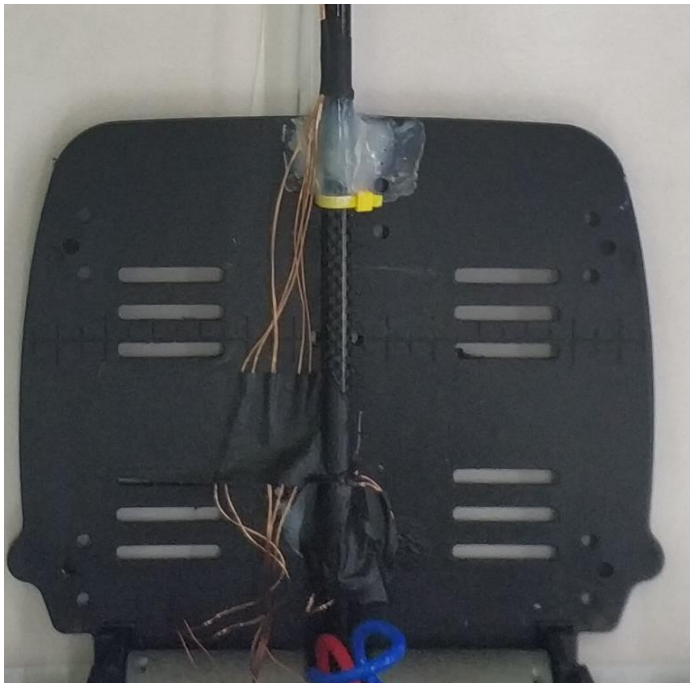

图 2.4 电感杆安装图

2.2.3. 车模倾角传感器

陀螺仪和加速度计用于测量车模姿态，需要和小车紧密相连，对于加速度计，为了减少运动产生的噪声，需要安装的尽可能靠近车轴，对于陀螺仪，则需要保证完全水平安装，否则转弯时会产生竖直方向的分量，形成加减速的现

象，陀螺仪同时也测量了车模转弯时的角速度用来进行转向微分控制，最终我们决定将它放在两个电机中间，并且垂直于车模正常跑时的角度，安装图如下：

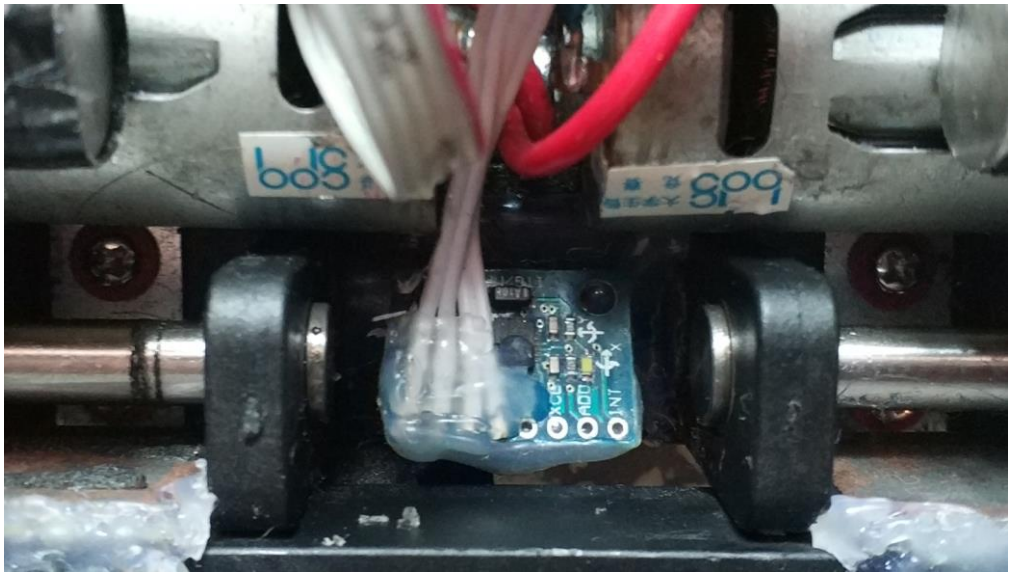

图 2.5 陀螺仪传感器安装图

2.3. 重心高度调整

重心的高度是影响智能车稳定性的因素之一。当重心高度偏高时，智能车在转弯过程中会发生抬轮现象，严重时甚至翻车。因此，从小车稳定性出发，我们尽量降低重心高度，从而保证小车可靠稳定。

2.3.1. 电路板的安装

为了使小车具有较好的稳定性及转向性能，我们在搭建小车时尽量选择降低重心，因此也将电路板安装在了电机上方，从而实现降低重心，提高小车的稳定性。

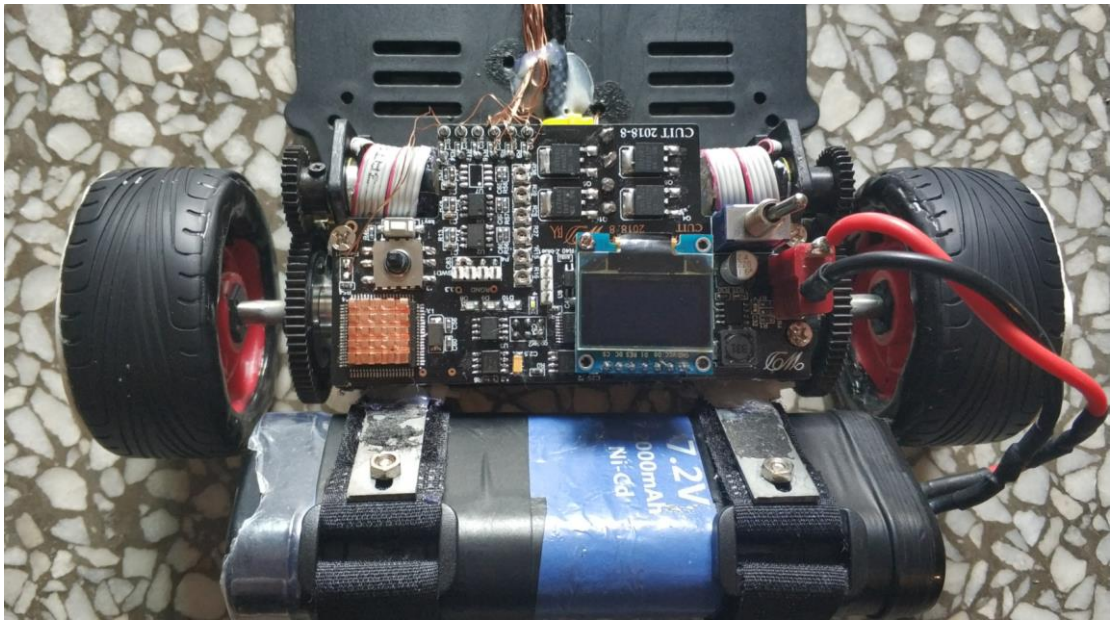

图2-6 主板安装图

2.3.2. 电池安放

同样为实现降低重心，提高小车稳定性的目的，我们制作了铝条，然后固定在车模后方，进而用于固定电池，最大程度的降低了小车的重心。

图2-7 电池安装图

2.4. 其他机械结构的调整

另外，在模型车的机械结构方面还有很多可以改进的地方，比如说车轮、传感器的保护等方面。由于直立小车的直立行驶及转向都是通过两个轮实现的，因此当小车在转向时，模型车的轮胎与轮毂之间很容易发生相对位移，可能导致在加速时会损失部分驱动力，而且使小车的状态不稳。因此，我们在实际调试过程中对车轮进行了粘胎处理，可以有效地防止由于轮胎与轮毂错位而引起的驱动力损失的情况。

2.5. 小结

模型车的性能与机械结构有着非常密切的联系。良好的机械结构是模型车提高速度的关键基础。在同等的控制环境下，机械机构的好坏对其速度的影响十分显著。我们非常重视对智能汽车的机械结构的改进，经过大量的理论研究和实践，我们小车的大部分质量都集中在两轮前后，达到降低重心的目的，从而提高了小车整体的稳定性和可靠性。

第三章　电路设计说明

3.1. 主控板设计

3.1.1. 电源管理模块

电源分为开关电源和线性电源，线性电源的电压反馈电路是工作在线性状态，开关电源是指用于电压调整的管子工作在饱和和截至区即开关状态的。线性电源一般是将输出电压取样然后与参考电压送入比较电压放大器，此电压放大器的输出作为电压调整管的输入，用以控制调整管使其结电压随输入的变化而变化，从而调整其输出电压，但开关电源是通过改变调整管的开和关的时间即占空比来改变输出电压的。从其主要特点上看：线性电源技术很成熟，制作成本较低，可以达到很高的稳定度，波纹也很小，而且没有开关电源具有的干扰与噪音，开关电源效率高、损耗小、可以降压也可以升压，但是交流纹波稍大些。电源模块对于一个控制系统来说极其重要，关系到整个系统是否能够正常工作，因此在设计控制系统时应选好合适的电源模块。

竞赛规则规定，比赛使用智能汽车竞赛统一配发的标准车模 7.2V 2000mAh Ni-cd 供电。系统中 3.3V 电路功耗较小，考虑到姿态传感器对于电源低纹波的要求，我们决定使用线性稳压芯片。此外，当直流电机在高速运行时的，电池作为一个恒功率源输出电流增大，势必带来输出电压减小。因此，为了提高系统工作的稳定性，我们选择了 TPS7350 和 TPS7333 对各个模块进行供电。

具体电源模块原理图如下：

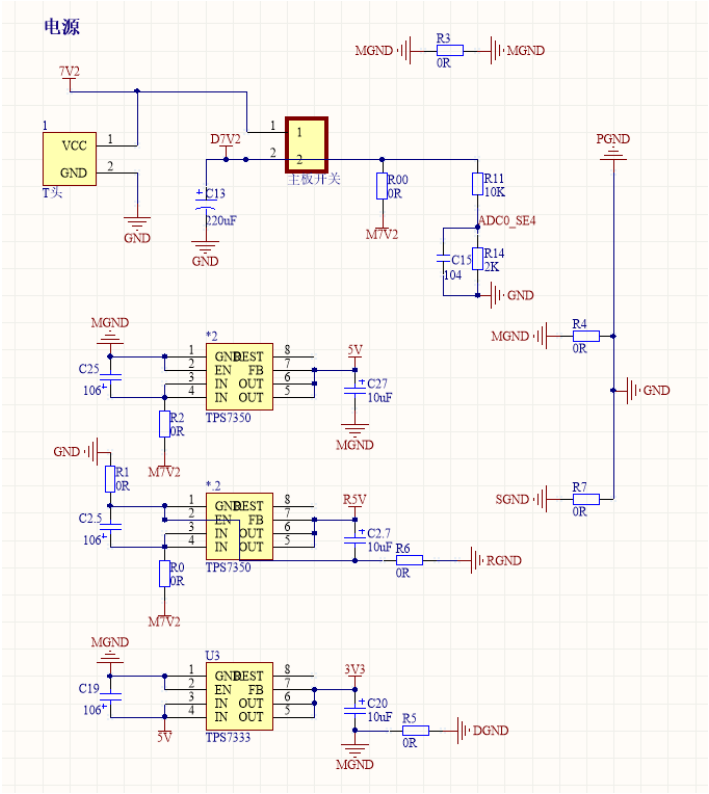

图 3-1 电源管理模块原理图

3.1.2. 主板

我们采用的是恩智浦基于 Cortex-M0+内核的 KEA128 单片机，市面上有很多此芯片的最小系统销售，但这些模块外设较多，质量较重，更注重扩展性，为了使车模总重更轻，重心更低，要求主板尽可能的小而轻， 所以我们根据 KEA128 技术手册列举的外围电路自行设计最小系统板。经过测试可以稳定运行，原理图如下：

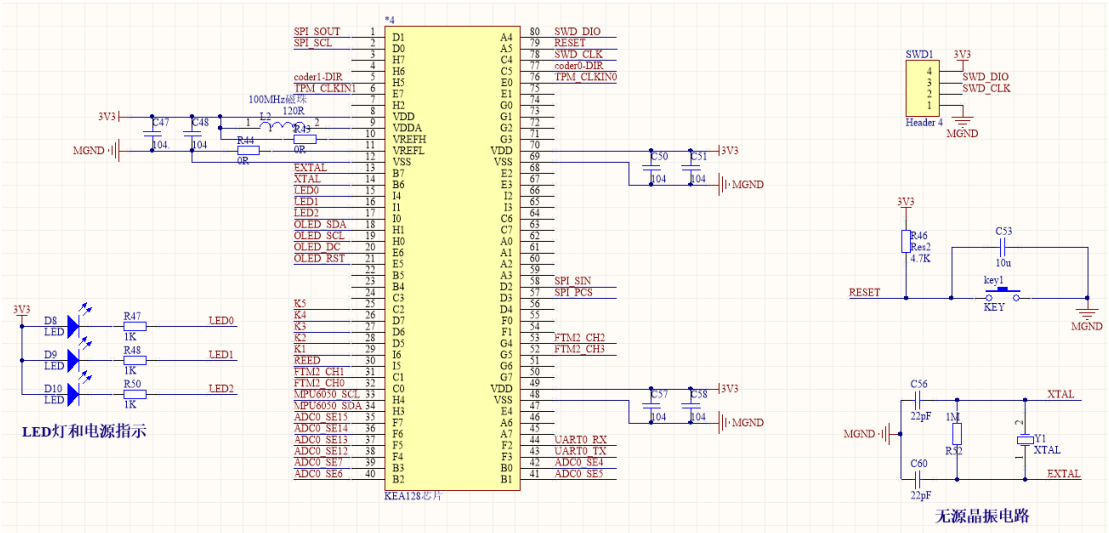

图3-2 最小系统原理图

信号放大模块：

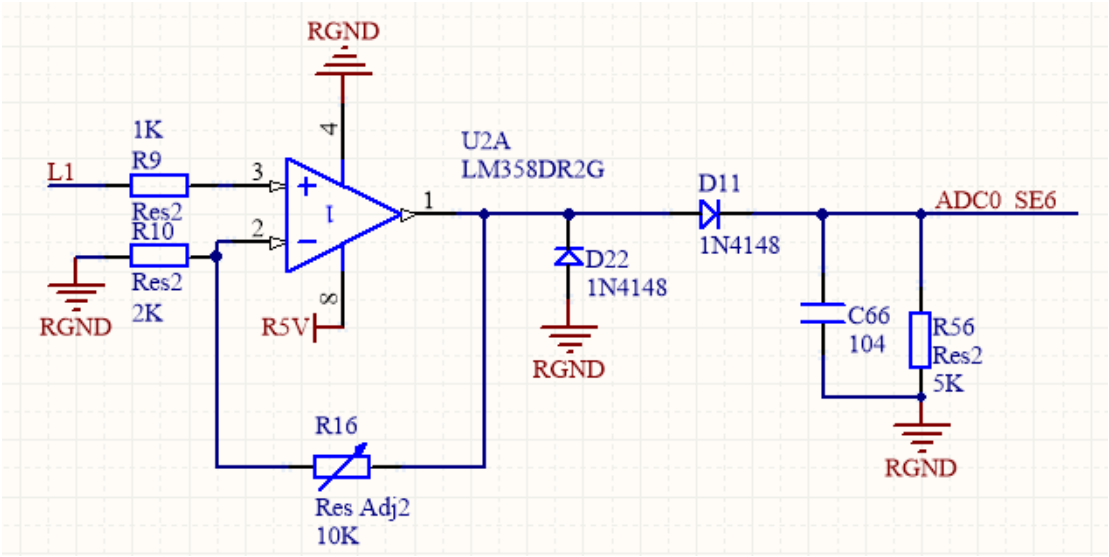

图3-3 信号放大原理图

按键&接口:

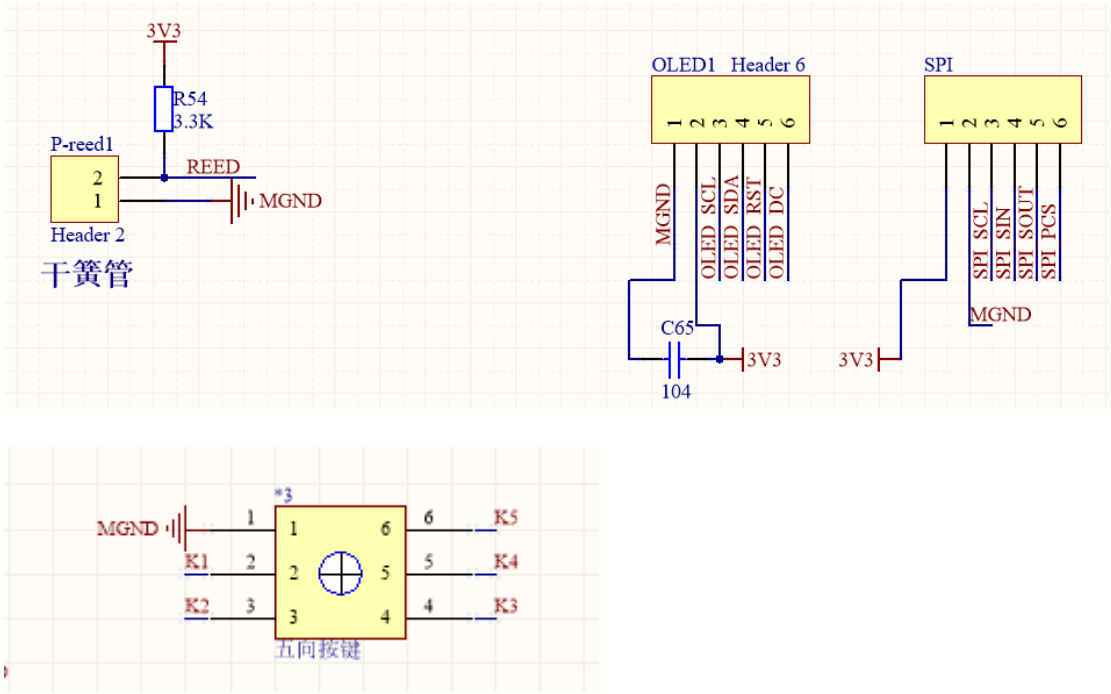

图3-4 按键及接口原理图

3.1.3. 电机驱动模块

本次电磁直立组需要驱动 2 个电机，驱动的设计尤为重要。常用的电机驱动有两种方式：一、采用集成电机驱动芯片；二、采用 N 沟道 MOSFET 和专用栅极驱动芯片设计。市面上常见的集成 H 桥式电机驱动芯片中，飞思卡尔公司的 33886 型芯片性能较为出色，该芯片具有完善的过流、欠压、过温保护等功能，内部 MOSFET 导通电阻为 120 毫欧，具有最大 5A 的连续工作电流。使用集成芯片的电路设计简单，可靠性高，但是性能受限。由于比赛电机内阻仅为几毫欧，而集成芯片内部的每个 MOSFET 导通电阻在 120 毫欧以上，大大增加了电枢回路总电阻，此时直流电动机转速降落较大，驱动电路效率较低，电机性能不能充分发挥。

由于分立的 N 沟道 MOSFET 具有极低的导通电阻，大大减小了电枢回路总电阻。另外，专门设计的栅极驱动电路可以提高 MOSFET 的开关速度，使 PWM 控制方式的调制频率可以得到提高，从而减少电枢电流脉动。

并且专用栅极驱动芯片通常具有防同臂导通、硬件死区、欠电压保护等功能，可以提高电路工作的可靠性。

1.专用栅极驱动芯片的选择：

IR 公司号称功率半导体领袖，所以我们主要在 IR 公司的产品中进行选择。其中 IR2184 型半桥驱动芯片可以驱动高端和低端两个 N 沟道 MOSFET，能提供较大的栅极驱动电流，并具有硬件死区、硬件防同臂导通等功能。使用两片 IR2184 型半桥驱动芯片可以组成完整的直流电机 H 桥式驱动电路。由于其功能完善，价格低廉容易采购，所以我们选择它进行设计，如图 3.2 所示。

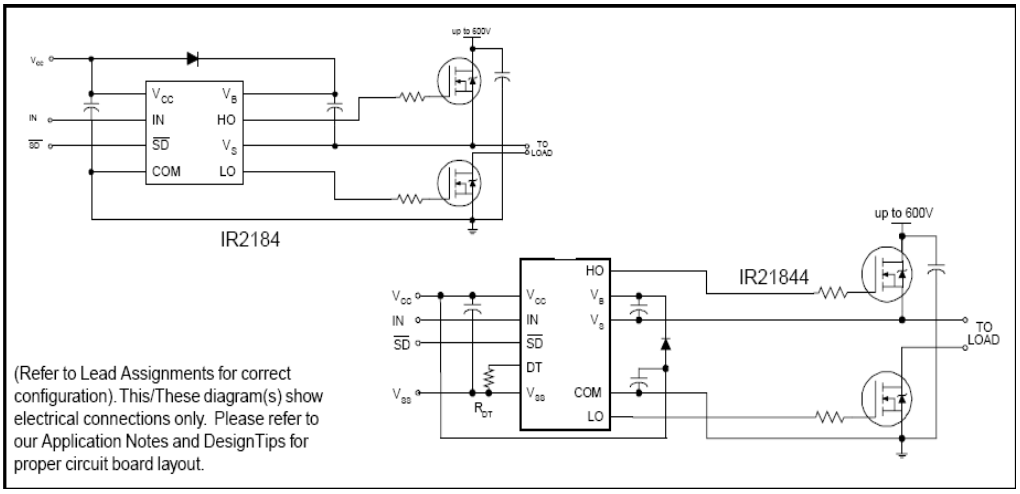

图 3.5 IR2184 应用图

2. MOSFET 的选择：

选择 MOSFET 时主要考虑的因素有：耐压、导通内阻和封装。智能汽车电源是额定电压为 7.2V 的电池组，由于电机工作时可能处于再生发电状态，所以驱动部分的元件耐压值最好取两倍电源电压值以上，即耐压在 16V 以上。而导通内阻则越小越好。封装越大功率越大，即同样导通电阻下通过电流更大，但封装越大栅极电荷越大，会影响导通速度。常用的 MOSFET 封装有 TO-220、TO-252、SO-8 等，TO-252 封装功率较大、而栅极电荷较小。于是我们最终选择了 IR 公司 TO-252 封装的 LR7843 型 N 沟道 MOSFET，VDSS = 55 伏、RDS(on) = 8.0 毫欧、ID= 110 安。

3. 控制逻辑电路设计：

IR2184 的控制信号有两个管脚：IN 和 SD。IR2184 输入输出信号关系图如图 3.3 所示：

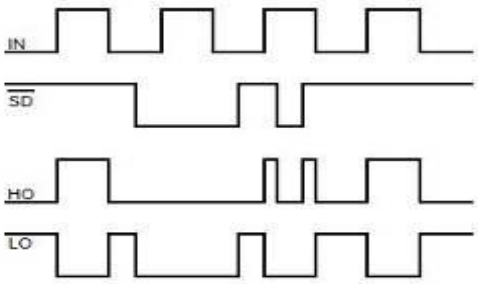

图 3.6 IR2184 输入输出关系图

而当两片 IR2184 驱动如图 3.4 所示可逆桥式电路时，其真值表为表 3.1：

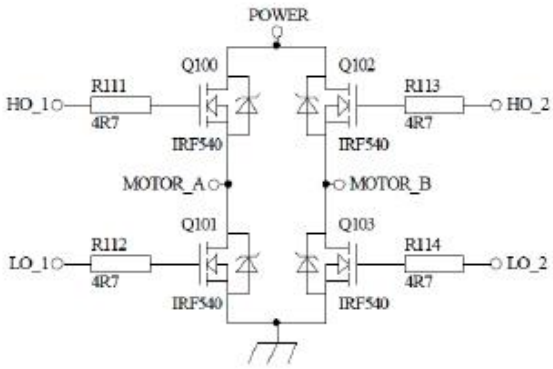

图 3.7 可逆桥式电路

表 3.1 可逆桥式电路中 IR2184 输入输出信号真值表

状态	输入				输出			
	IN1	SD1	IN2	SD2	HO1	LO1	HO2	LO2

正转	H	H	L	H	H	L	L	H
反转	L	H	H	H	L	H	H	L
上桥臂制动	H	H	H	H	H	L	H	L
下桥臂制动	L	H	L	H	L	H	L	H
关闭	X	L	X	L	L	L	L	L

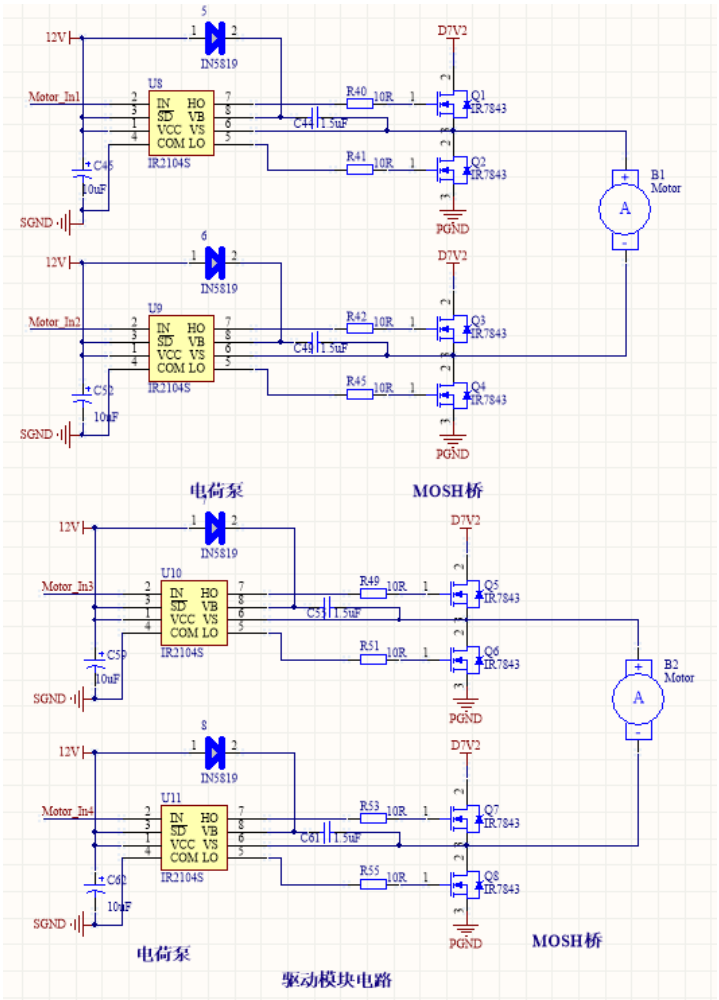

图 3.8 MOS 桥驱动电路原理图

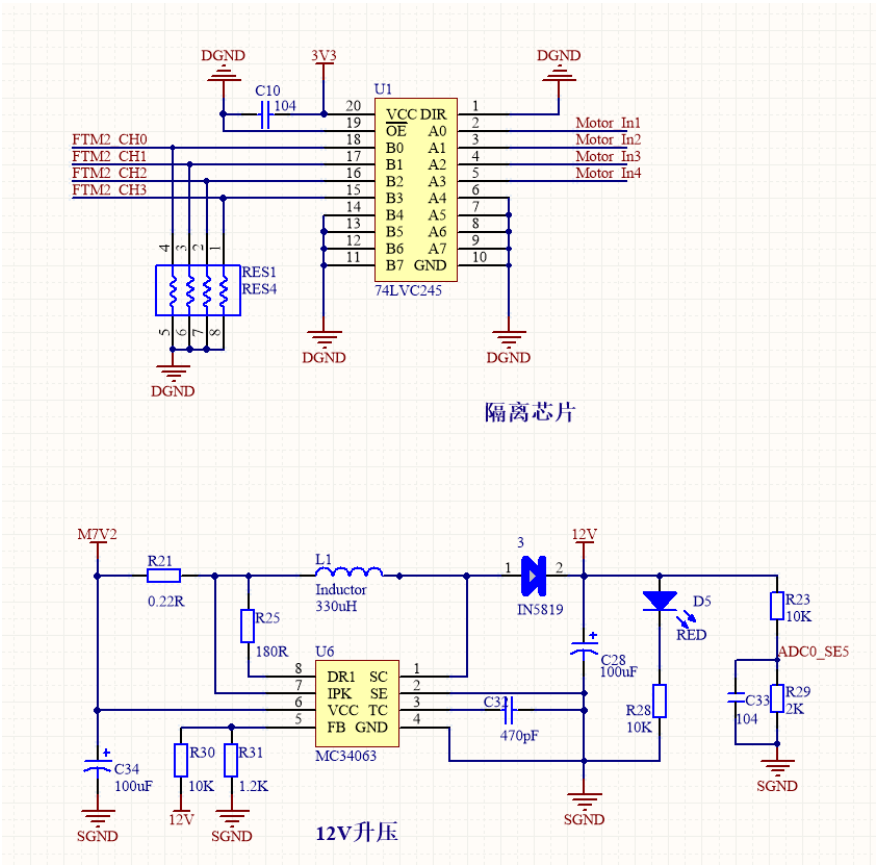

图 3.9 12V 升压电路原理图

我们用两片 IR2104 及四个 MOSFET 组成全桥的电机驱动电路，并且采用单极性的控制模式，PWM 的占空比在 0%之时电机速度为零，0~100%控制转速，H_D 信号控制转向。

4、驱动电路的原理分析及元件参数确定

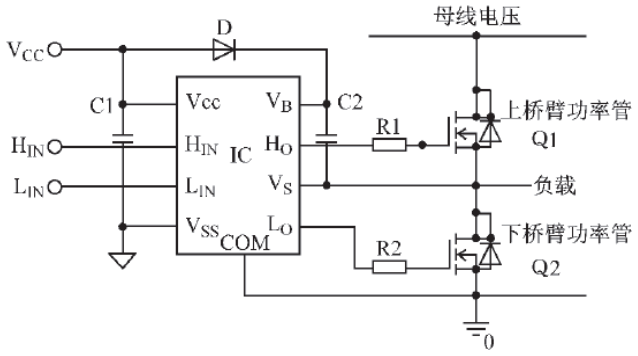

图 3.10 电机驱动分析图

这个驱动设计单从信号逻辑上分析比较容易理解，但要深入的理解和更好的应用，就需要对电路做较深入的分析，对一些外围元件的参数确定做理论分析计算。图 3.6 中 IC 是一个高压驱动芯片,驱动 2 个半桥 MOSFET。Vb,Vs 为高压端供电;Ho 为高压端驱动输出;COM 为低压端驱动供电,Lo 为低压端驱动输出;Vss 为数字电路供电.此半桥电路的上下桥臂是交替导通的,每当下桥臂开通,上桥臂关断时 Vs 脚的电位为下桥臂功率管 Q2 的饱和导通压降,基本上接近地电位,此时 Vcc 通过自举二极管 D 对自举电容 C2 充电使其接近 Vcc 电压.当 Q2 关断时 Vs 端的电压就会升高,由于电容两端的电压不能突变,因此 Vb 端的电平接近于 Vs 和 Vcc 端电压之和,而 Vb 和 Vs 之间的电压还是接近 Vcc 电压.当 Q2 开通时,C2 作为一个浮动的电压源驱动 Q2;而 C2 在 Q2 开通其间损失的电荷在下一个周期又会得到补充,这种自举供电方式就是利用 Vs 端的电平在高低电平之间不停地摆动来实现的.由于自举电路无需浮动电源,因此是最便宜的，如图所示自举电路给一只电容器充电，电容器上的电压基于高端输出晶体管源极电压上下浮动。图 3.6 中的 D 和 C2 是 IR2184 在脉宽调制（PWM）应用时应严格挑选和设计的元器件,根据一定的规则进行计算分析;并在电路实验时进行调整,使电路工作处于最佳状态,其中 D 是一个重要的自举器件,应能阻断直流干线上的高压,其承受的电流是栅极电荷与开关频率之积,为了减少电荷损失,应选择反向漏电流小的快恢复二极管,芯片内高压部分的供电都来自图中自举电容 C2 上的电荷;为保证高压部分电路有足够的能量供给,应适当选取 C2 的大小。

MOSFET具有相似的门极特性,开通时需要在极短的时间内向门极提供足够的栅电荷,在自举电容的充电路径上,分布电感影响了充电的速率,下桥

臂功率管的最窄导通时间应保证自举电容有足够的电荷#以满足栅极所需要的电荷量再加上功率器件稳态导通时漏电流所失去的电荷量.因此,从最窄导通时间为最小值考虑,自举电容应足够小;综上所述,在选择自举电容大小时应考虑,既不能太大影响窄脉冲的驱动性能;也不能太小影响宽脉冲的驱动要求,应从功率器件的工作频率、开关速度、门极特性等方面进行选择、估算后调试而定。

3.1.4. 接口模块

1.模数转换接口

由于 KEA128 的内部 AD 比较精确，我们在测试外部 AD 后结果显示其结果与内部 AD 相差不大，而且方差较内部 AD 更大，当然这也有可能是测试方法有问题。内部 AD 较外部 AD 可以简化电路设计，综合考虑这下我们还是决定使用内部 AD。

2．陀螺仪接口

陀螺仪在保持车身的平衡方面极其重要，为了方便更换，我们并未将陀螺仪直接画在主板上。由于使用的是单片机本身的软件 IIC，所以陀螺仪的接口很简单。电路如下所示。

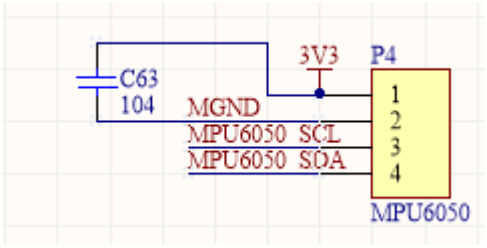

图 3.11 陀螺仪接口

3.1.5. 陀螺仪和加速度计

今年大赛没有规定陀螺仪的型号，所以使用经常使用的 MPU6050。

MPU6050 芯片它是由 InvenSense 公司生产的，又称为微惯性器件单元，它内部集成了三轴加速度计传感器和三轴陀螺仪传感器， 它不仅消除了我们

在焊接电路时，非常容易造成加速度计和陀螺仪之间的对准误差，而且还由于它的内部设置了数字可编程的低通滤波器，当飞行器有较大振动时，可以认为的用程序设置适当频率的低通滤波器，用来滤掉不需要的高频振动噪声，这是用来减小车模机体振动对姿态的影响的一个很有效的方法。MPU6050 六轴传感器被广泛应用于物体的姿态检测与校正中，其电路如下：

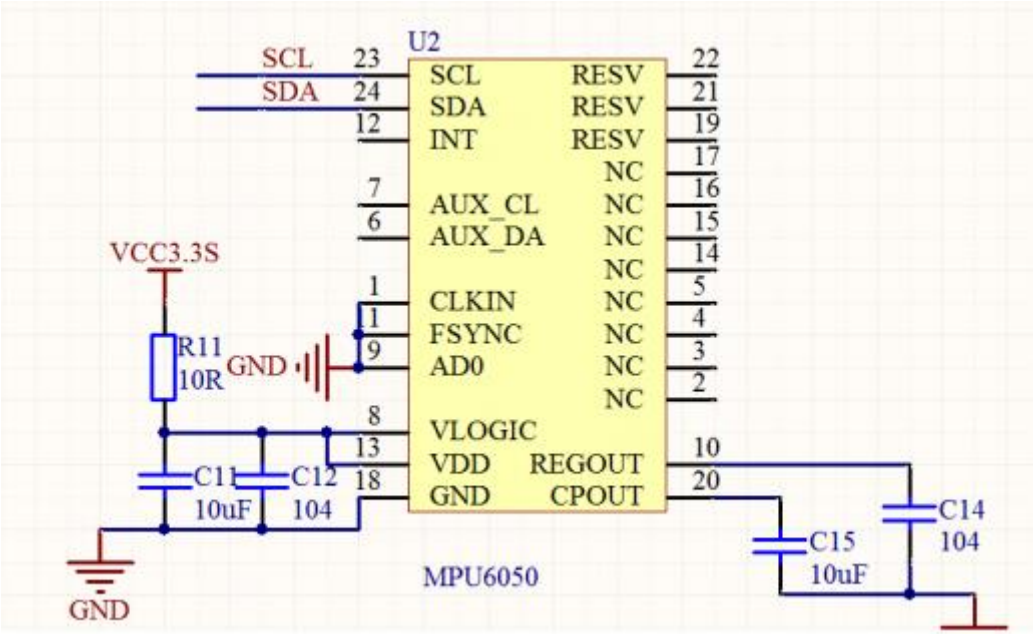

图 3.12 陀螺仪电路图

3.2. 速度传感器

为了使用闭环控制，我们在汽车模型上附加了编码器。和其他元件相比，选用编码器可以使电路更加完善，信号更加精确。编码器功耗低，重量轻，抗冲击抗震动，精度高，寿命长，非常实用。迷你编码器内部有上拉电阻，因此编码器接口不需要设计上拉电阻。KEA128 自身不具有正交解码功能，迷你编码器自带脉冲输出和方向输出，只需要将接口连接到单片机上相应的接口即可。接口如图 3.12 所示。

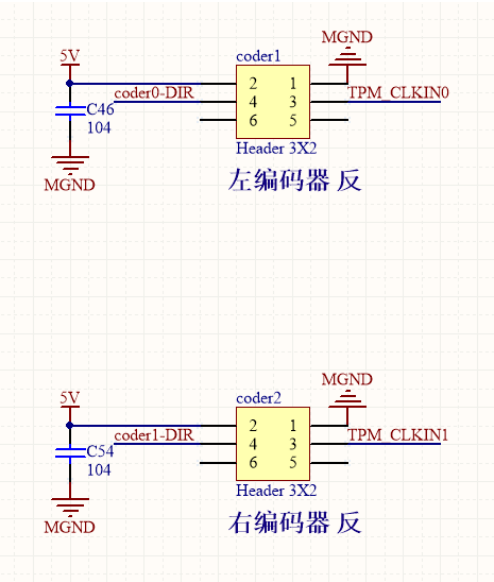

图 3.13 编码器的接口部分

3.3. OLED 显示屏

在调试过程之中，我们需要实时的了解与掌握一些车的运行状态，比如说传感器的状态，对管的返回值等，调试时用 OLED 显示屏将这些参数显示出来，让我们实时的监测车的状态，从而做出判断，这样很大程度的方便了对车的调试。有时候需要对参数作修改处理，如果每修改一个数据就下载一次程序的话，就会浪费时间，这时应用五向按键，它就起到一个人机交互的作用。

3.4. 小结

硬件电路是模型汽车系统的必备部分。只有稳定的硬件电路才能保证程序的正确控制。为此，我们在设计电路之时，考虑了很多问题，采用了模拟部分与数字部分隔离等措施。我们的硬件电路的设计思想是在保证正确检测信号的前提下，尽可能精简电路。

第四章　智能车控制软件设计说明

高效的软件程序是智能车高速平稳自动寻线的基础。我们设计的智能车系统采用数字摄像头进行赛道识别，图像采集及校正处理就成了整个软件的核心内容。在智能车的转向和速度控制方面，我们使用了鲁棒性很好的经典 PID 控制算法，配合使用理论计算和实际参数补偿的办法，使智能车能够稳定快速寻线。

车模姿态控制包括直立环、速度环和方向环三个闭环，这三个环相互影响，相互制约，为了使车模高速稳定运行，需要综合考虑很多因素，确定这三个环的控制方法，下面将介绍我们的控制方式。

4.1. 姿态控制

4.1.1. 直立控制

车模直立控制包括姿态检测、直立 PD 控制、姿态检测使用加速度计和陀螺仪互补滤波完成，通过外环角度、内环角速度使车模保持直立状态。

对于一个倒立摆系统，使它处在平衡位置需要两个力，一个是回到原点的回复力，另一个则是使它停下来的粘斥力，类似单摆，我们使用角度值和角速度值进行位置式 PD 控制可使小车稳定的直立。直立控制就是下面的代码，原型来自位置式 PD 公式。

外环角度环：

```
Tar_Ang_Vel.Y = -PID_Realize(&Angle_PID, Angle,imu_data.pit*100, ADRC_SPEED_Controller.x1);
Tar_Ang_Vel.Y = range_protect(Tar_Ang_Vel.Y,-1500, 1500);   // 注意正负号
```

内环角速度环：

```
Theory_Duty += -PID_Increase(&Ang_gyro_PID, Ang_gyro, (int32)(GYRO_Real.Y*10),(int32)Tar_Ang_Vel.Y);
Theory_Duty = range_protect(Theory_Duty, -950, 950);                                    // 直立PI
```

4.1.2. 互补滤波

为了实现车模直立控制，需要获得车模的倾角和倾角速度，测量车模倾角和倾角速度通过安装在车模上的加速度计和陀螺仪实现。

1：加速度传感器

加速度计可以测量由地球引力作用或物体运动所产生的加速度。我们使用了 MPU6050 的加速度计测量加速度，读取车模垂直方向的加速度值可得出与之成正比的车模角度。但是在实际车模运动过程中，由于车模本身摆动所产生的加速度会产生很大的干扰信号，它叠加在上述测量型号上会让输出信号无法准确反映车模倾角。

2：陀螺仪

陀螺仪可以测量物体旋转角速度，我们使用了 MPU6050 的陀螺仪测量角速度，对车模垂直方向的角速度积分也可以获得车模角度，该信号中噪声很小，使角度信号更稳定。

3：互补滤波

我们通过加速度传感器获得的角度信息对陀螺仪积分得到的角度进行校正，通过对比积分所得到的角度和重力加速度所得到的角度，使用它们之间的偏差改变陀螺仪的输出，从而积分的角度逐步跟踪到加速度传感器所得到的角度。

4.1.3. 速度控制

为了获得车模速度，我们使用迷你编码器进行速度检测，车模速度控制是加在直立角度上的，相当于在直立得到的角度上直接加上一个角度，然后输出给角度，同时为了角度输出平滑，我们将角度变化率进行限幅， 也对最大最小角进行限幅， 让车子在运行过程中倾角不进行剧烈变化，

代码如下：

```
Target_Angle.Y = -PID_Realize(&Speed_PID, Speed, ADRC_SPEED_MIN_Controller.x1,Speed_Set);/
Target_Angle.Y +=1200;        //重心为24度
if (ABS(Middle_Err)>40||sysTickUptime<7000)//弯道或者起跑前几秒满速
    Target_Angle.Y = -1300;
else
    Target_Angle.Y = range_protect(Target_Angle.Y,Speed_Limit_Min, Speed_Limit_Max);    //
Fhan_ADRC(&ADRC_SPEED_Controller,Target_Angle.Y);
```

4.1.4. 转向控制

车模的方向控制是先由电感差比和得到偏差，然后外环对偏差做出处理，输出量作为内环期望值，内环控制对象为陀螺仪的 Z 轴角速度，内外环均使用 PD 控制。

外环：

```
Speed_Calculate();                                                    //根据当前速度判断转向PID参数
Radius = -PlacePID_Control(&Turn_PID, Turn[Fres], Middle_Err, 0);    //转向外环PID
```

内环：

```
Direct_Parameter = PID_Realize(&Turn_gyro_PID, Turn_gyro, (int32)(GYRO_Real.Z*100),(int32)Radius*142);
Direct_Parameter = range_protect(Direct_Parameter, -1100, 1100);
```

4.2. 特殊元素处理

4.2.1. 环岛

在本届智能汽车竞赛中，出现了‘环岛’元素，这是一种全新的元素，对于此元素，有以下处理方案：

一、利用电感值的变化以及电感的阈值来判断，但是此种方案容易出现误判，因为此方案需要车模在高速运动下依然保持稳定，一旦出现跳动等不稳定的现象，环岛就会误判。

二、只利用电感的阈值，加之人工判断环岛方向和大小，这种方案可以稳定识别出环岛的存在，而且环岛方向不会出现误判，且环岛大小现场可调，具有很高的灵活性。

三、利用八字电感寻迹入环，这种方案需要在前瞻上再加两个八字电感，会使车模重心大大向前移，导致车模不可控性增加，从而导致不稳定。

综上所述，我们选择第二种方案，利用电感的阈值，加之人工判断设置，经过大量测试和验证，此方案可行。

4.2.2. 坡道和颠簸

由于现在直立车的速度越来越快，很多时候过坡道时，车子会飞到空中然后再着地，当车子在空中时，由于车子失去了赛道摩擦带来的阻力以及电感离赛道很远导致偏差检测也不准确，此时轮速也会发生突变，此时轮速和车速是不吻合的，甚至相差很多，所以当着地一瞬间，编码器返回的数据仅仅是轮子的速度而不是车子本身的速度，所以这会导致落地一瞬间车子的姿态控制出现偏差，导致车子的平衡系统不稳定。

对此我们限制了轮速的变化率，使得车模在腾空后落地时也能保持稳定。

4.3. PID 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。

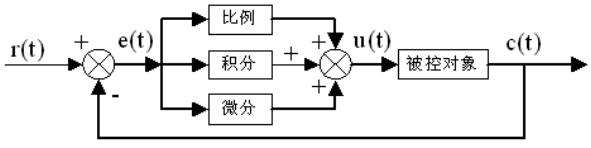

图 4.1 PID 控制器原理框图

在计算机控制系统中，使用的是数字 PID 控制器，控制规律为：

$$e(k) = r(k) - c(k) \quad \text{(公式 4.4)}$$

$$u(k) = K_P\{e(k) + \frac{T}{T_I}\sum_{j=0}^{k} e(j) + \frac{T_D}{T}[e(k) - e(k-1)]\} \quad \text{(公式 4.5)}$$

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。

4.3.1. 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(4.5)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)。

4.3.2. 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(4.5)推导出提供增量的 PID 控制算式。由式(4.5)可以推出式(4.6)，式(4.5)减去式(4.6)可得式(4.4)。

$$u(k-1)=K_P\{e(k-1)+\frac{T}{T_I}\sum_{j=0}^{k-1}e(j)+\frac{T_D}{T}[e(k-1)-e(k-2)]\} \quad \text{(公式 4.6)}$$

$$\Delta u(k)=K_P\{[e(k)-e(k-1)]+\frac{T}{T_I}e(k)+\frac{T_D}{T}[e(k)-2e(k-1)+e(k-2)]\}$$
$$=K_P\Delta e(k)+K_I e(k)+K_D[\Delta e(k)-\Delta e(k-1)]\} \quad \text{(公式 4.7)}$$

式中 $\Delta e(k)=e(k)-e(k-1)$； $K_I=K_P\frac{T}{T_I}$； $K_D=K_P\frac{T_D}{T}$

公式(4.7)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(4.7)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。

4.3.3. 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。

4.4. PID 控制算法

在小车跑动中，因为不需要考虑小车之前走过的路线，所以，我们舍弃了 I 控制，将小车转向的 PID 控制简化成 PD 控制。本方案中通过双电机的差速控制采用位置式的 PD 控制，速度闭环控制采用了增量式 PID 控制。

工程整定方法是通过闭环试验，观察系统响应曲线，根据各控制参数对系统响应的大致影响，反复试凑参数，以达到满意的响应，最后确定 PID 控制参数。整定不是盲目的，而是在控制理论指导下进行的。在控制理论中已获得如下定性知识：

比例调节（P）作用：是按比例反应系统的偏差，系统一旦出现了偏差，比例调节立即产生调节作用用以减少偏差。比例作用大，可以加快调节，减少误差，但是过大的比例，使系统的稳定性下降，甚至造成系统的不稳定。

积分调节（I）作用：是使系统消除稳态误差，提高无差度。因为有误差，积分调节就进行，直至无差，积分调节停止，积分调节输出一常值。积分作用的强

弱取决与积分时间常数 Ti，Ti 越小，积分作用就越强。反之 Ti 大则积分作用弱，加入积分调节可使系统稳定性下降，动态响应变慢。积分作用常与另两种调节规律结合，组成 PI 调节器或 PID 调节器。

微分调节（D）作用：微分作用反映系统偏差信号的变化率，具有预见性，能预见偏差变化的趋势，因此能产生超前的控制作用，在偏差还没有形成之前，已被微分调节作用消除。因此，可以改善系统的动态性能。在微分时间选择合适情况下，可以减少超调，减少调节时间。微分作用对噪声干扰有放大作用，因此过强的加微分调节，对系统抗干扰不利。此外，微分反应的是变化率，而当输入没有变化时，微分作用输出为零。微分作用不能单独使用，需要与另外两种调节规律相结合，组成 PD 或 PID 控制器。

4.5. ADRC 控制器

ADRC 自抗扰控制器是一种新型的控制器，其性能优异，被称为 21 世纪的工业革命，ADRC 控制器克服了 PID 的多数缺点，使用时无需考虑被控对象是否非线性，是否是时变系统。ADRC 由三部分组成：

（1）跟踪微分器（TD），由称之为安排过渡过程，其作用在于对控制器的输入期望值信号进行跟踪和算微分，比起 PID 直接作差算微分的方法，跟踪微分器不会明显放大噪声，并且对控制信号进行平滑的跟踪，减小了输入的突变对控制器干扰 。

（2）扩展状态观测器（ESO），该部分对 ADRC 控制器的性能有着决定性的作用，也是需整定参数最多的环节。ESO 实现的功能为对误差的跟踪、微分，和对系统内扰、外扰及总扰动的量化。

（3）非线性状态误差反馈控制率（NLSEF），ADRC 控制器相对 PID 控制器改进之处还在于 PID 控制器最终的控制量是 P、I、D 三项的线性加权和得到的，而 NLSEF 通过非线性函数来组合 P、D 得到输出量，可以实现小误差大增益输出，提高抗扰能力和打舵响应 。

尽管 ADRC 控制器有着远超 PID 控制器的性能，但是 ADRC 控制器需要整定的参数繁多，至今没有可靠科学的参数整定方法，所以 ADRC 的工程应用实例很少。

在本次设计中，由于 KEA128 运算能力有限，且 ESO 环节计算量过大，我只使用了 TD 环节来对速度环的输出、编码器速度的输出做了处理，效果如图：

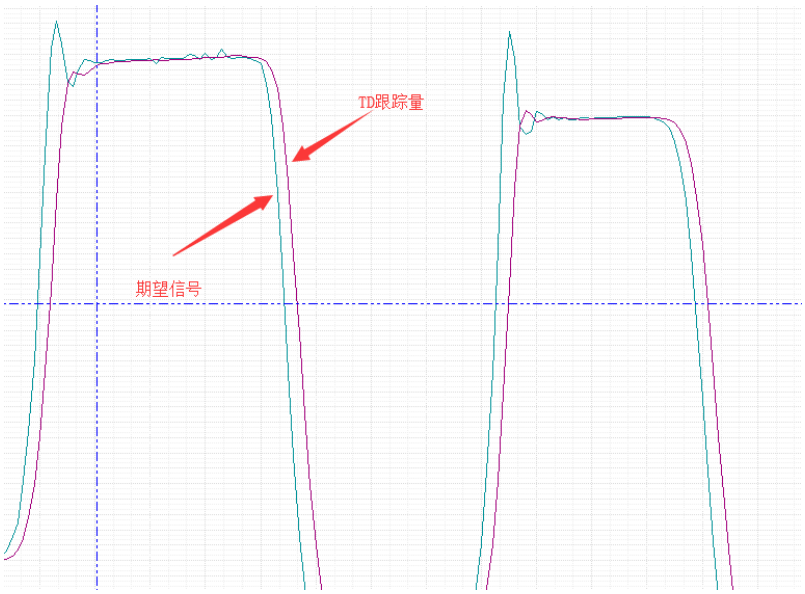

图 5.1 TD 滤波效果图

直立车的速度控制是滞后的，在坡道和颠簸路段由于车速过快经常会出现轮子离地空转的情况，此时速度环的输出必将突变而导致车体不稳定，而 TD 跟踪微分器可将非线性变化转为线性曲线并且去除突变，可使车模姿态更稳定。

第五章　开发工具、制作、安装、调试过程说明

5.1. 开发工具

程序开放在 IAR Embedded Workbench IDE 下进行， Embedded Workbench for ARM 是 IAR Systems 公司为 ARM 微处理器开发的一个集成开发环境(下面简称 IAR EWARM)。比较其他的 ARM 开发环境，IAR EWARM 具有入门容易、使用方便和代码紧凑等特点。

EWARM 中包含一个全软件的模拟程序(simulator)。用户不需要任何硬件支持就可以模拟各种 ARM 内核、外部设备甚至中断的软件运行环境。从中可以了解和评估 IAR EWARM 的功能和使用方法。

5.2. 上位机

我们使用了 QT 来制作一个简易上位机，可在线调参，波形显示等功能，使用蓝牙模块可将数据实时传输到上位机。

第六章　模型车的主要技术参数说明

赛车基本参数	长	47.5cm
	宽	25cm
	高	32cm
车重		915g
功耗	空载	10W
	带载	大于 12W
电容总容量		2080uF
传感器	数字摄像头	0 个
	陀螺仪	1 个
	编码器	2 个
	加速度计	1 个
赛道信息检测	视野范围（近瞻/远瞻）	10cm/500cm
	精度(近/远)	2/100mm
	频率	50Hz

结论

自报名参加“恩智浦”杯智能汽车竞赛以来，我们队成员从查找资料、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，我们分析了整车质量分布，调整重心位置，优化机械结构。在电路方面，我们以模块形式分类，在最小系统、主板、电机驱动等模块分别设计，在查找资料的基础上各准备了几套方案；然后我们分别实验，最后以报告中所提到的形式决定了我们最终的电路图。在程序方面，我们使用 C 语言编程，利用比赛推荐的开发工具调试程序，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。

在这几个月的备战过程中，场地和经费方面都得到了学校和学院的大力支持，在此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。

现在，面对即将到来的大赛，在历时近五个月的充分准备以及华北赛的考验之后，我们有信心在全国比赛中取得优异成绩。也许我们的知识还不够丰富，考虑问题也不够全面，但是这份技术报告作为我们小组辛勤汗水的结晶，凝聚着我们小组每个人的心血和智慧，随着它的诞生，这份经验将永伴我们一生，成为我们最珍贵的回忆。

参考文献

[1]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.

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

[3]王晓明．电动机的单片机控制 [M] ．北京：北京航空航天大学出版社. 2002.

[4]安鹏，马伟．S12 单片机模块应用及程序调试[J] ．电子产品世界．2006．第 211 期． 162-163.

[5]张文春．汽车理论[M]．北京．机械工业出版社．2005.

[6]童诗白，华成英．模拟电子技术基础 [M] ．北京: 高等教育出版社，2001.

[7]阎石．数字电子技术基础 [M] ．北京: 高等教育出版社，2000.

[8]谭浩强著．C 程序设计．北京：清华大学出版社，2003．

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

[10]Park K.H ，Bien Z，Hwang D.H. A study on the robustness of a PID - type iterative learning controller against initial state error [J]. Int. J. Syst. Sci. 1999, 30(1) ，102～135.

[11]殷剑宏，吴开亚．图论及其算法 [M] ．中国科学技术大学出版社，2003.

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

[13]尹怡欣，陶永华．新型 PID 控制及其应用．北京：机械工业出版社，1998 年.

[14]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．

[15]仲志丹，张洛平，张青霞．PID 调节器参数自寻优控制在运动伺服中的应用[J] ．洛阳工学院学报，2000，21（1）：57～60.

[16]卓晴，黄开胜，邵贝贝．学做智能车：挑战“飞思卡尔”杯．北京：北京航空航天大学出版社，2007 年.

附录：程序源代码

```
static void Duty_2ms(void)//2.4ms
{
    Start_Control();
    MPU6050_GetData(&GYRO, &ACC);    // 读取陀螺仪数据
    Data_steepest();                 //原始数据梯度下降滤波
}

static void Duty_4ms(void)//0.5ms
{
L_AD_Sample();
    Theory_Duty += -PID_Increase(&Ang_gyro_PID, Ang_gyro,
(int32)(GYRO_Real.Y*10),(int32)Tar_Ang_Vel.Y);    // 计算直立 PWM     内环角速度环
    Theory_Duty = range_protect(Theory_Duty, -950, 950);
// 直立 PWM 限幅

    Direct_Parameter = PID_Realize(&Turn_gyro_PID, Turn_gyro,
(int32)(GYRO_Real.Z*100),(int32)Radius*142); // 转向环左正右负
(int32)Radius*Speed_Min
    Direct_Parameter = range_protect(Direct_Parameter, -1100, 1100);

    Direct_Last = Direct_Last*0.2 + Direct_Parameter*0.8;  // 更新上次角速度环结果
```

```
        MOTOR_Duty_Left    = Theory_Duty - Direct_Last;  // 左右电机根据转向系数调整差速
        MOTOR_Duty_Right = Theory_Duty + Direct_Last;

        if (Run_Flag&&sysTickUptime>5000)
            MOTOR_Control(MOTOR_Duty_Left, MOTOR_Duty_Right);  // 控制左右电机
        //MOTOR_Control(500, 500); // 控制左右电机
        else
        {
            if (Stop_Flag)
            {
                if (Speed_Now > 60) MOTOR_Control(-850, -850);
                else                            MOTOR_Control(0, 0);
            }
            else
                MOTOR_Control(0,0);
        }

    }

    static void Duty_8ms(void)//0.6ms
    {
        //Road_Find();
```

```
        IMU_update(0.008f,&(sensor.Gyro_deg), &(sensor.Acc_mmss),&imu_data);
//姿态解算
        Speed_Measure();
        Distance += Speed_Now/55.0f;
        Speed_Calculate();
//根据当前速度判断转向 PID 参数
        Radius = -PlacePID_Control(&Turn_PID, Turn[Fres], Middle_Err, 0);          //
转向外环 PID
        Tar_Ang_Vel.Y = -PID_Realize(&Angle_PID, Angle,imu_data.pit*100,
ADRC_SPEED_Controller.x1);  /*  角度环加到角速度环上串级控制
*///ADRC_SPEED_Controller.x1
        Tar_Ang_Vel.Y = range_protect(Tar_Ang_Vel.Y,-1500, 1500);      //  注意正
负号
        ANO_DT_Data_Exchange();
    }

    static void Duty_40ms(void)    //0.2ms
    {

        Target_Angle.Y = -PID_Realize(&Speed_PID, Speed,
ADRC_SPEED_MIN_Controller.x1,Speed_Set);/*  速度环加到角度环上串级控制
ADRC_SPEED_MIN_Controller.x1 Speed_Now*/
        Target_Angle.Y +=1200;              //重心为 24 度
        if (ABS(Middle_Err)>40||sysTickUptime<7000)//弯道或者起跑前几秒满速
            Target_Angle.Y = -1300;
```

```
        else
            Target_Angle.Y = range_protect(Target_Angle.Y,Speed_Limit_Min,
Speed_Limit_Max);  //   Speed_Limit_Min, Speed_Limit_Max
        Fhan_ADRC(&ADRC_SPEED_Controller,Target_Angle.Y);

    }
    static void Duty_100ms(void)
    {
        gpio_turn(C2);
        if (OLED_Refresh)
        OLED_Draw_UI();
        Check_BottonPress();
    }

    //系统任务配置，创建不同执行周期的“线程”
    static sched_task_t sched_tasks[] =
    {
        {Duty_100ms ,   100, 0},
        {Duty_40ms   ,    40, 0},
        {Duty_8ms    ,     8, 0},
        {Duty_4ms    ,     4, 0},
        {Duty_2ms    ,     2, 0}
```

```
};
//根据数组长度，判断线程数量
#define TASK_NUM (sizeof(sched_tasks)/sizeof(sched_task_t))

//这个函数放到 main 函数的 while(1)中，不停判断是否有线程应该执行
void Loop_Run(void)
{
    uint8_t index = 0;

    //循环判断所有线程，是否应该执行
    for(index=0;index < TASK_NUM;index++)
    {
        //获取系统当前时间，单位 MS --
        uint32_t tnow = sysTickUptime;
        //进行判断，如果当前时间减去上一次执行的时间，大于等于该线程的执行周期，则执行线程
        if(tnow - sched_tasks[index].last_run >= sched_tasks[index].interval_ticks)
        {
            //更新线程的执行时间，用于下一次判断
            sched_tasks[index].last_run = tnow;
            //执行线程函数，使用的是函数指针
            sched_tasks[index].task_func();
        }
```

```
    }
}
```