

**SIRB硬件组考核**

**设计报告**

*学 校： 成都信息工程大学*

*小 组： 硬件组*

*题目：遥控+避障+巡线小车*

**摘要**

本设计以学习PID常见的平衡车为平台，以STM32F103C8T6为主控，实现了平衡车的遥控，巡线，避障功能。通过串口蓝牙模块实现了小车端和遥控端的两个STM32之间的通信，通过超声波模块检测距离，实现避障功能，通过线性CCD传感器，实现了巡线功能，通过蓝牙遥控实现3种模式切换。

该作品使用三个PID环，进行简单的线性叠加，较好的实现了控制功能。通过PD控制，用MPU6050读取到俯仰角，实现了直立环；通过PI控制， 读取编码器，实现了速度环；在遥控模式下，通过PD控制，读取偏航角，实现了转向环；在巡线模式下，通过PD控制，处理CCD传感器的数据得到偏差，实现了转向环。

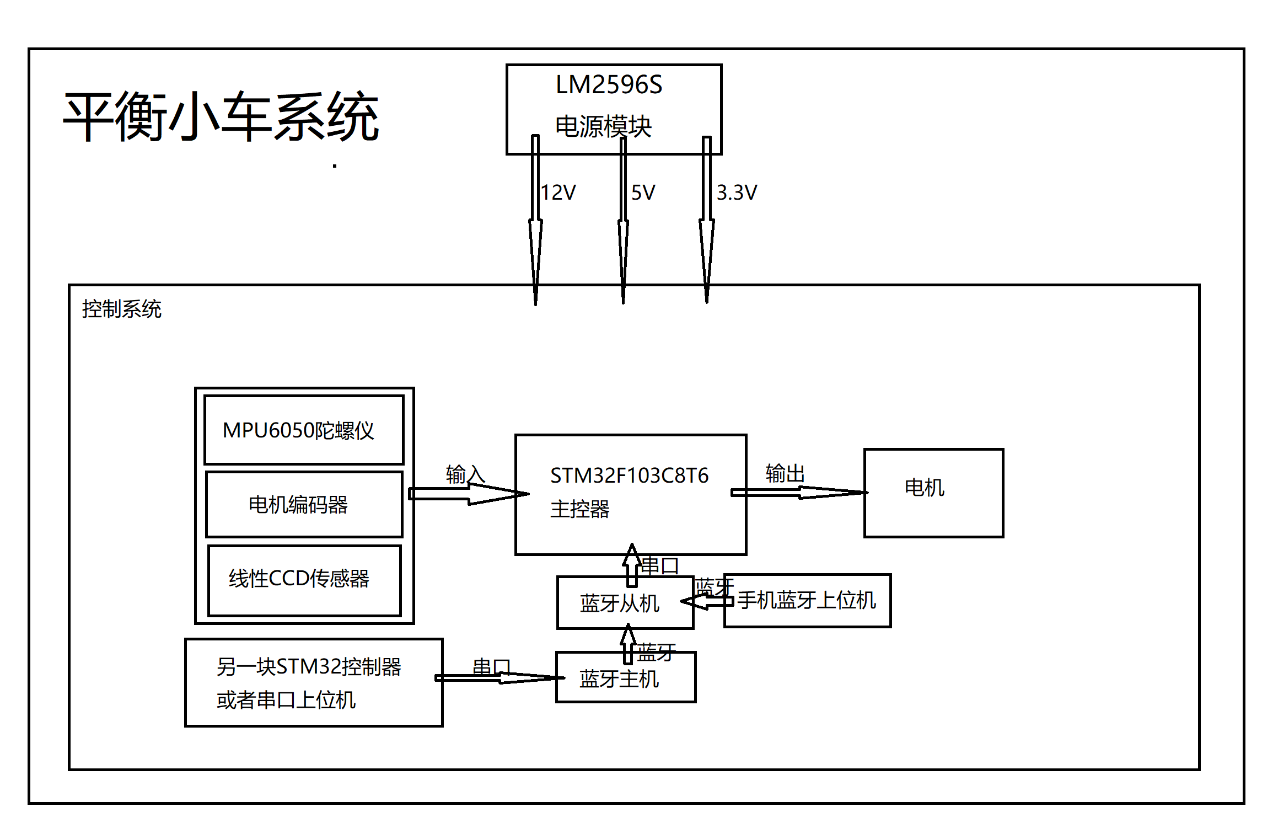
关键词：STM32平衡车，CCD巡线，超声波避障，PID算法

**第一章 系统总体设计方案**

智能车作为一个整体系统，主要包括机械设计、硬件电路、软件算法三个主要部分，三部分互相联系，相互影响，相互促进。

对这三个部分统筹设计是贯穿始终的原则，机械部分设计决定智能车能力的极限，硬件电路为智能车提供实现机械潜能所必需的电源、检测设备及控制调试能力，软件算法对数据进行处理最终实现智能车机械和电路的控制，达到智能化。

系统结构框如图所示：



**1.1智能车机械结构整体设计方案**

我们对于智能车机械设计部分总体的设计原则为：车身对称、连接牢固、减轻重量、重心合理、车体美观五个方面。根据机械构造原理，对智能车的底盘、传动、车轮等进行规则允许范围内的设计调整。

**1.2 智能车硬件电路整体设计方案**

硬件电路是整个智能车系统中相当重要的一部分，是系统运行的基础。主要设计原则是最简、实用、可靠和模块。因为电路复杂就增加了故障几率，只要符合要求，提供足够的应用功能就足够，而智能车最容易出现故障的环节往往是硬件电路部分，这部分出现问题的后果也比较致命，所以可靠的电路设计的重要原则之一，另外，模块化设计便于整个系统的修改、升级、更替，而整体化有利于减小电路板的尺寸，使电路更加的简洁，稳定，可靠。

智能车中的电路包括：微处理器最小系统、电源、传感器电路、电机驱动电路、测速器以、信号硬件处理及其他周边电路。各模块的详细说明见第三章。

**1.3 智能车软件整体设计方案**

软件部分的设计主要是对微处理器STM32F103C8T6的程序编写，通过单片机采集线性CCD、编码器等的信号，并通过控制算法处理，最终通过单片机端口输出至硬件电路，实现遥控、循迹、避障等功能。

软件的设计原则主要是：高效率、结构化、规范化、易读改。软件执行结果的输出直接控制硬件，因此提高处理效率能够使控制更加及时。而程序编写时的结构化和规范化能使逻辑关系复杂的程序更加易看易懂，便于调试和修改以及扩展。

程序编译环境选用KEIL+CubeMX ，主要程序编写语言为C 语言。软件设计具体思路及算法将在第四章中进行详细介绍。编译环境及调试方法详见第五章。具体代码详见附录A。

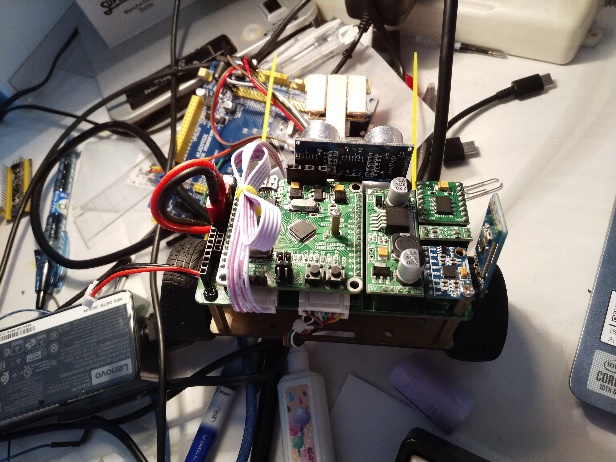
**第二章 机械设计和模块选型**

一个好的机械结构对于智能车的性能提升是毋庸置疑的，尤其对于C车来说。所以为了智能车的性能提升，我们对智能车的机械部分做了一系列的设计和调整。本章主要对机械布局设计、车模机械结构的改装、摄像头选型、安装、编码器的选取与安装、对抗的机械设计做一些详解。

**2.1 平衡小车整体的机械设计**

这里经过考虑，决定使用模块化的结构设计，使用各个模块，然后设计小车主板，通过PCB主板将小车的各个模块连接起来，然后将小车主板固定在小车底盘上，就形成了完整的小车结构。

完成效果如图：



**2.2 小车底板和电机的选型**

这里使用常见的亚克力底板和小尺寸的有刷直流减速电机，选择带有编码器的电机，便于后期获取更多数据。



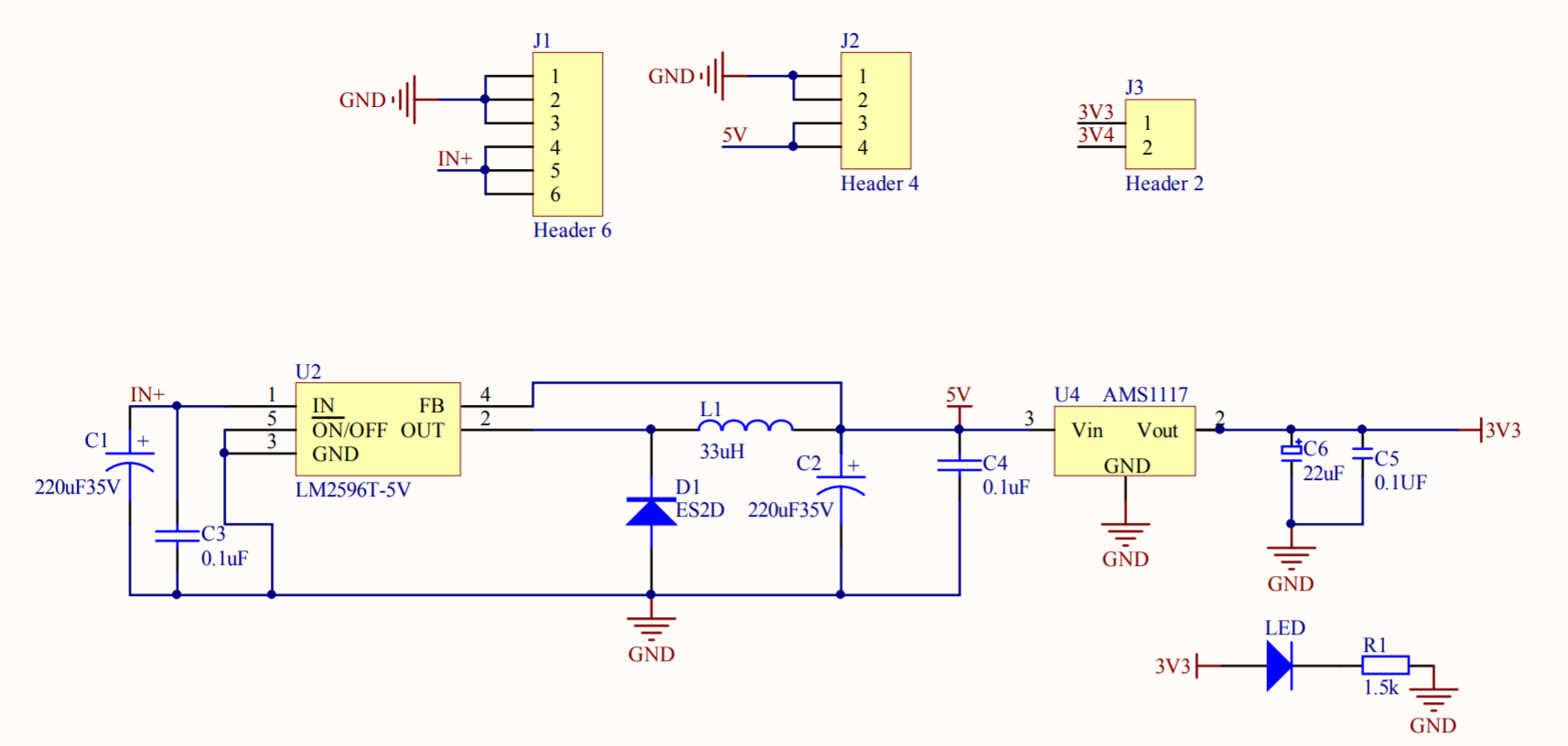
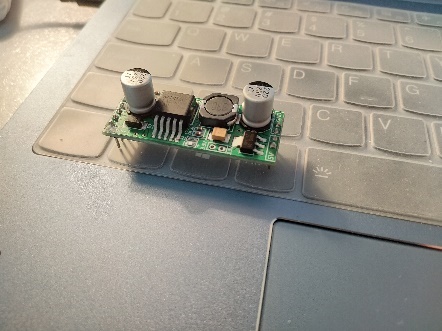
**2.3 电机驱动的选型**

这里使用自带H桥电路的集成双电机驱动模块：TB6612。避免使用体积较大的L298N模块：



**2.4 电源模块的选型**

这里使用LM2596S作为降压模块，使用方便，输入航模电池电压8V或12V，输出恒定5V：



**2.5 平衡所需传感器选型**

这里使用最常见的集成的MPU6050陀螺仪模块，可以利用移植官方DMP库的形式，方便的获取3个轴的角度以及角速度，为小车平衡和遥控转向提供数据。

**2.6 避障所需传感器选型**

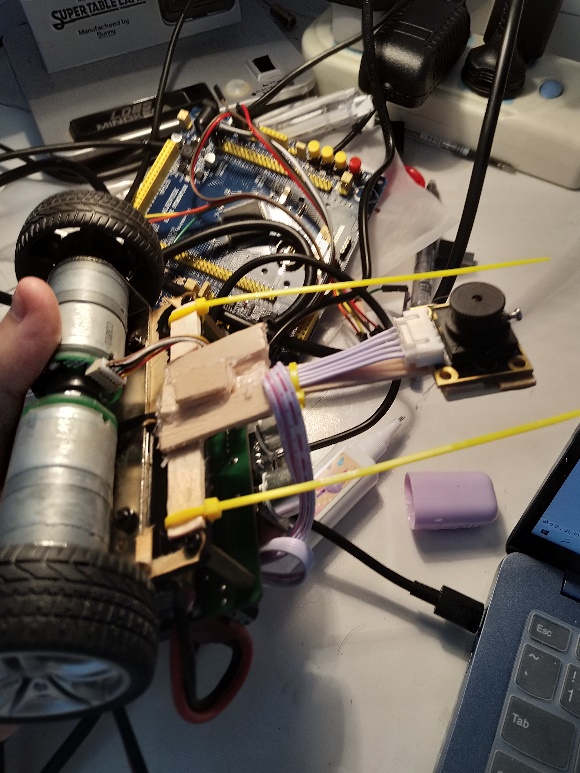
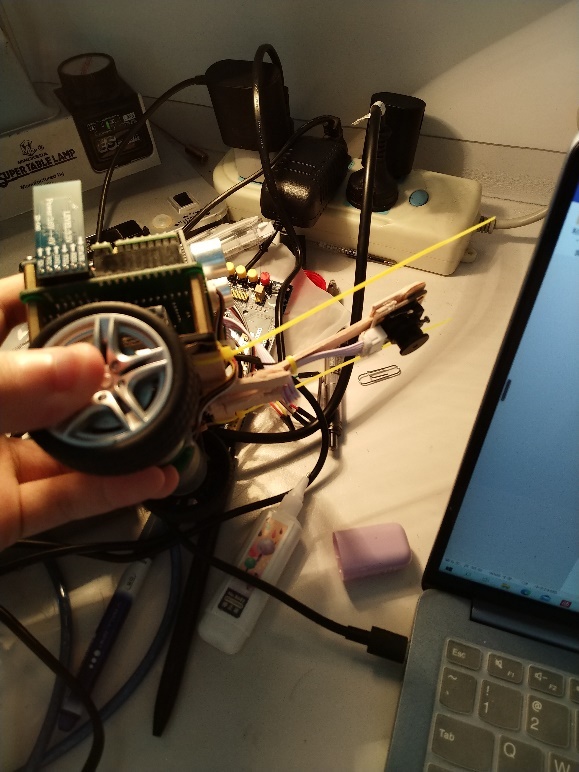
这里使用US015超声波模块，使用方便，简化了编程的难度。

**2.7 巡线所需传感器选型**

巡线一般有多种方式：红外对管，电磁巡线，摄像头巡线等等。但是考虑到我使用平衡车，红外对管由于车体俯仰，误差较大；电磁巡线不适用于普通的电气胶带赛道；摄像头过于复杂了，不适用与本次项目。

因此，使用线性CCD，既能够获得较高的精度，又不至于很难使用。

我通过雪糕棒和热熔胶，做了一个简易支架，然后用轧带把模块绑在了小车地板上，考虑到CCD的实际视角，做了一个大约15°的角度，加上小车重心前移带来的机械平衡位置改变，有20读的上仰，较为合适，实际效果如图：

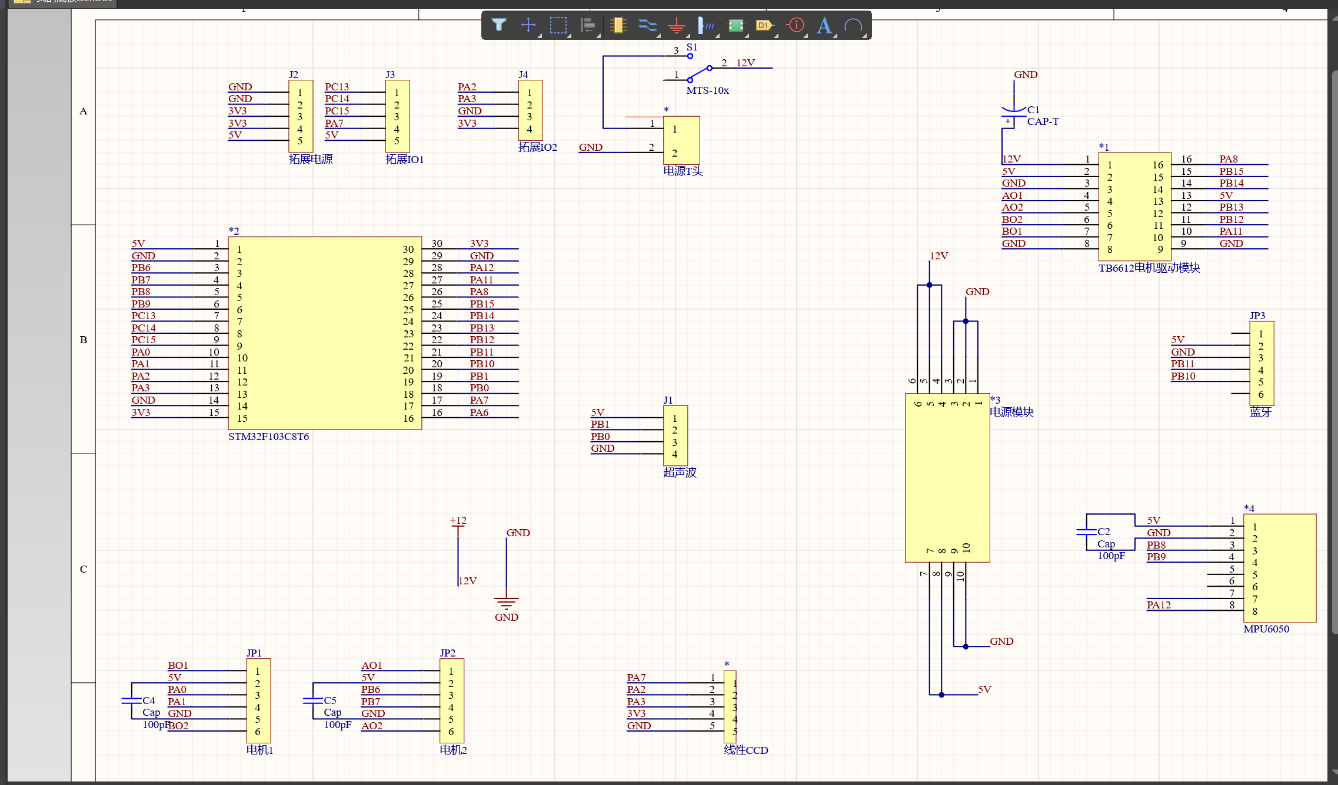


**第三章 硬件部分的设计**

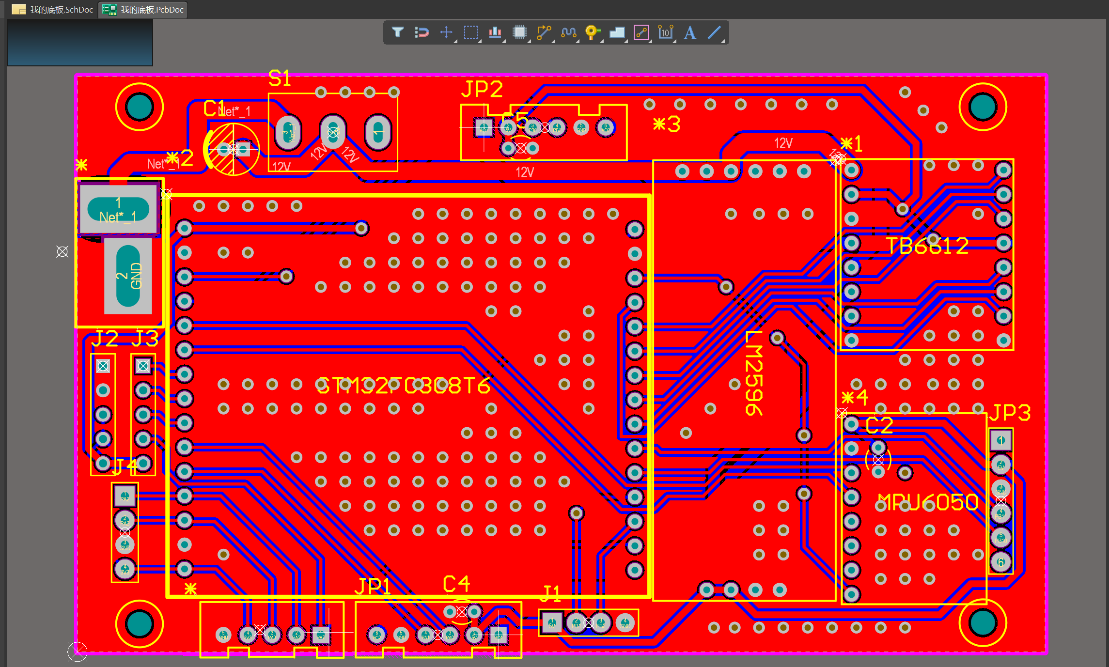
**3.1 小车主板硬件电路设计**

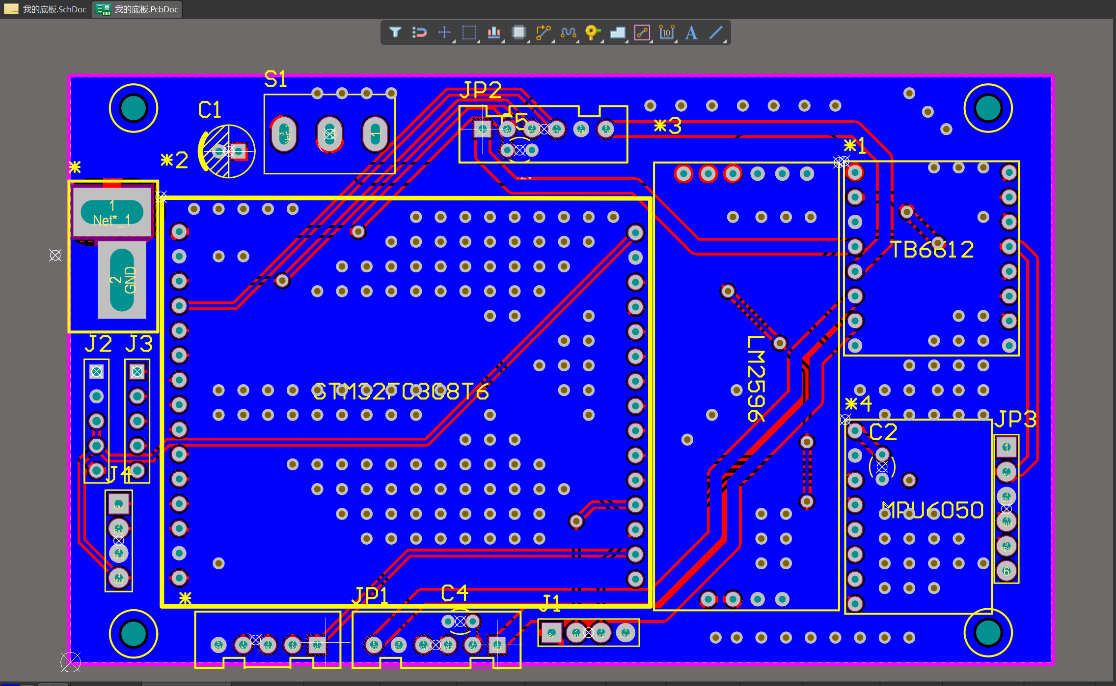
硬件电路在整个智能车系统占据非常重要地位，是系统运行的基础，所以每一个电容每一个电阻都需要去考究。硬件电路的设计是一个非常深的学问，为了提高硬件电路的稳定性和可靠性，需要对其不断的改进和考究。我们也经过不断的设计和更改，最终设计了一快简洁、稳定、集成度高的主板。

主板的原理图：



主板的PCB顶层以及底层：



****

**第四章 理论分析及控制器算法设计与实现**

高效适用的软件程序是智能车实现稳定快速运行及智能化的基础。该小车通过串口蓝牙模块实现了小车端和遥控端的两个STM32之间的通信，通过超声波模块检测距离，实现避障功能，通过线性CCD传感器，实现了巡线功能。

该作品使用三个PID环，进行简单的线性叠加，较好的实现了控制功能。通过PD控制，用MPU6050读取到俯仰角，实现了直立环；通过PI控制， 读取编码器，实现了速度环；在遥控模式下，通过PD控制，读取偏航角，实现了转向环；在巡线模式下，通过PD控制，处理CCD传感器的数据得到偏差，实现了转向环。

**4.1 系统软件流程图**

系统软件设计部分主要包括：MPU6050，编码器，超声波，线性CCD传感器的数据采集，蓝牙遥控，处理器处理数据，输出信号驱动电机，其系统流程图如图所示：

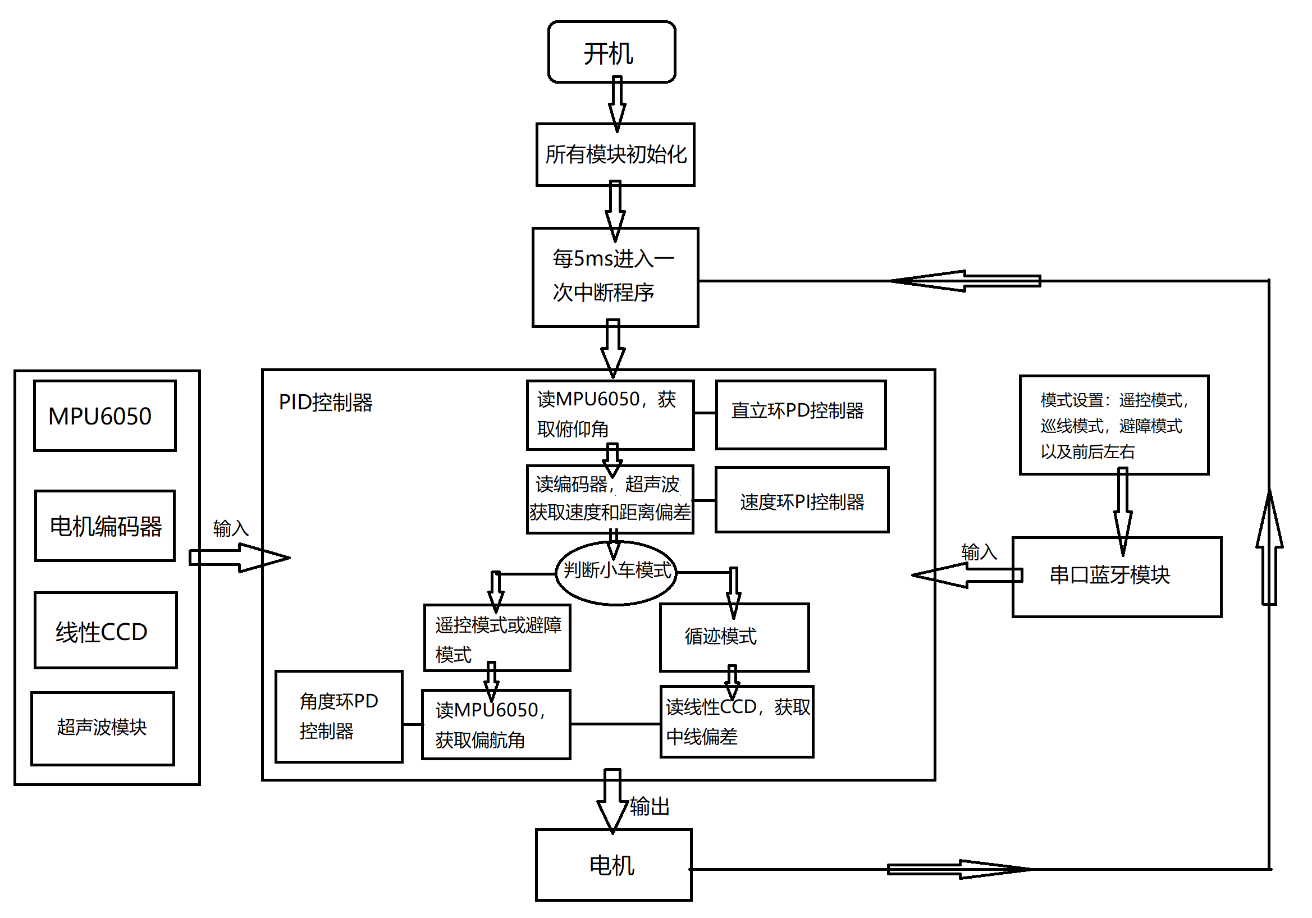


图4.1 系统软件流程图

**4.2 直立环控制算法**

**比例控制：把握当下**

那么，究竟什么时候用PI，什么时候用PD呢？这就要从原理说起。

比如我做一个平衡车，希望通过一种控制算法来实现平衡，最容易想到的方法就是比例控制：

1、我希望平衡车稳定达到平衡状态（用户期望值）是平衡车的机械平衡角度（目标物理量），以及受控物体的角度，MPU6050传感器的DMP度数，

2、两个数据的偏差，作为输入控制器的值，在控制器中乘上一个增益，经过输出限幅，然后输出给执行器件：电机，

3、电机的工作状态，反过来又会影响平衡车的角度，角度又会被传感器收集，重新计算偏差，输入控制器。

于是，这样一个偏差控制器就形成了一个闭环。

理论上，一个理想的P控制器，只能使受控物理量无限接近目标值，因为当偏差逐渐减小，趋于零时，控制器的输入趋于零，那么无论有多大增益，输出也会趋于零，这样就会停在一个固定值。

但是在实际中，我们要考虑多种因素，第一，输出趋于零，小车可不会静止在一个角度，而是迅速倒下，第二，实际上第一种情况也是不存在的，因为从控制器输出PWM给电机，到平衡车角度改变，本身就是有很长时间的延时，会有惯性，这种控制影响总是延后造成。因此，真正的情况是，平衡车的角度已经很接近机械零度了，电机的输出也已经快要停止了，然而，平衡车却有一个角速度，方向正是最初纠正平衡车的方向，这样，就算停掉电机，小车依然会越过平衡点，越过平衡点之后，偏差为负数，但是数值却很小，偏差经过控制器增益、限幅、输出给电机，电机却无法产生足够及时纠正偏离的转动，于是到小车反向倒下足够大的角度时，电机输出的转速才足够纠正，最终的结果就是，小车开始出现大幅度的震荡，根本无法平衡。

**比例+微分控制：预测未来，控制平衡**

如何改善这种震荡呢，经过前面的分析，我们知道，出现震荡的根本原因是机械系统的惯性，导致输出对受控物体的影响总是延后的。问题就出在，当角度接近机械零度时，我们的P控制器不工作了，控制器认为完成了任务，然而控制器忽略了一种对下一个时刻的预测，那么这个对下一个时刻的预测，指的就是角度偏差的导数，角度变化的速率，就是角速度。

有了这个认识，我们继续分析，角度偏差接近零度，但是系统存在角速度，那么下一刻就一定会有角度偏差，并且方向与角速度方向相反，通过数据，获取到角速度，就能够通过角速度，来修正这种状态，有预见性的输出给电机，让电机在下一个时刻就开始纠正偏差的倾向。

于是，我们加入D控制，微分项，相当于一个阻尼力：在偏差项P占主导时，微分项D的影响相对来说非常小，然而当偏差趋于零时，微分项的作用就是，减慢受控物理量的变化速率。

这样做的核心原理就是，既然控制存在滞后，那我就提前控制，就达到了抑制震荡，促进系统在平衡点稳定的目的。

**4.3 速度环控制算法**

**比例+积分控制：总结过去，控制速度**

上面的PD控制系统，基本上可以让平衡小车处于一个短期稳定平衡的状态，但是肯定不能长时间平衡，原因呢，要从具体的平衡控制结构来分析。

我们的传感器是单一的MPU6050，通过DMP库，输出的是角度以及角速度信号，而我们控制的是电机，也就是说，我们不考虑电机的感受，只要求平衡车的角度能够保持在机械零度，那么一旦平衡车收到扰动，或者控制闭环的任何一个环节，出现了微小的干扰，平衡车就会向一个方向加速，角度还是平衡的，但是平移的速度却没有限制，这样一旦驱动电机的PWM信号达到了限幅的数值，电机不再加速旋转，小车的执行器件就失去了执行能力，就会向这个方向倒下。

经过分析，我们需要再参考一个物理量，就是编码器的读数，既然使用编码电机，编码器的读数就需要纳入测量范围。然后问题又来了，如何处理读到的编码器数据呢？我们设置定时器模式为AB相编码器模式，那么每次读编码器，得到的是0-65535这个区间的一个值，电机转动，编码器的读数就会变化，我们希望小车在一个位置静止，就需要在平衡PD控制的基础上，加上一个位置控制，两种信号经过处理，输入控制器，控制器经过融合，输出给电机。

现在的问题是，位置控制使用什么样的方法？显然，位置控制并不像角度的平衡控制，电机转动就能够较快地达到效果，因此，我们需要一段时间，一步一步地接近我们设定的位置，也就是说，需要使用速度控制，对速度的一步步累积，使小车的位置（编码器的读数）逐渐趋于某一个值，这就要用到PI算法，即比例+积分控制来实现速度控制。

继续分析，我们假设用户输入的目标值是放下小车时刻，编码器的读数就是小车（车轮）的速度，这里可能就有点问题了，明明编码器的读数是脉冲信号，是位置啊，为什么说是速度呢？这就涉及到对编码器的数据处理，我们每隔一段时间，我们读取CNT寄存器，然后软件清零，意味着我们每次读取的数据，都是一段时间的计数值变化，不就是速度吗。弄清楚这个，我们开始分析怎么实现比例+积分的控制。

每隔一定周期，我们获取一次编码器的速度，然后与用户期望的速度，得到一个偏差，这个偏差就是P（比例），很显然偏差越大，我们需要的回复力就越大。到这里，我们吧前面的平衡PD控制，加上速度P控制，经过调整三个增益的大小以及比例，在控制器中对信号进行简单的叠加（叠加式PID，另外还有串联式，这里没有用到），就能够完美的实现小车的长时间稳定了。完美的状态是，推动小车，小车会前进一段距离，然后重新进入平衡。

那么，我们如果想要实现小车能够记住自己的位置，在受到外部干扰后自动回复到初始位置，该如何操作呢？注意，这里我提到了记住这么一个概念，就不难想到，这种控制，需要历史的全部速度信息，对离散的速度进行积分，得到从初始时刻，到当前时刻的位置信息。为了做到这一点，我们需要引入速度PI控制，比例+积分。具体的操作就是：从开始时刻，固定周期将速度偏差进行累加（具体可能需要滤波，这里讲解原理，略去），这个累加的值，就是偏离的距离，把这个距离，经过一个增益，也叠加进前面的3个参数，一共4个参数，共同作为输入，经过控制器输出给电机，这样就实现了在一个位置的PD+PI控制算法。

然后，将速度的比例项持续加偏置，就可以实现小车的匀速前进或者后退。

**4.4 转向环控制算法**

这个小车具有具有两种角度的控制算法，具体根据小车的模式进行选择。

如果是遥控模式，那么转向环的控制算法就和直立环的算法形式相同，都属于PD控制，不同的是，转向环的比例项是MPU6050读取的偏航角，而微分项是Z轴的角速度，就可以实现匀速原地转动。

如果是巡线模式，那么转向环的控制算法，就是以线性CCD读取的数据，经过处理得到了中线的位置，这个位置和128/2=64进行比较，就得到了偏差项，微分项仍然采用MPU6050的Z轴角速度，就实现了循迹功能。

**4.5 线性CCD的二值化以及中线提取**

线性CCD的工作原理是，内置128个呈线性排列的光电耦合管，各自有独立的电容，通过特定的通信协议，控制CCD开始曝光，这些电容就会进行放电，最后依次输出128个独立光电管的模拟量，而单片机做的就是用ADC外设，收集这些模拟量，转换为数字量，然后对这个数组进行数据处理。

我使用了二值化的算法，遍历128个数，提取最大和最小，求平均，那么高于平均的就是白色部分，低于平均的就是黑色部分。

然后从左到右寻找跳变，再反向寻找跳变，就得到了黑线的位置以及宽度，也就得到了线的中间位置。

**第五章 开发平台**

由于在开始前，我并没有STM32基础，因此使用KEIL进行开发，后期的编码器模式，输入捕获模式等等，使用CubeMX生成代码，结合KEIL进行开发，利用J-Link进行在线调试，取得了较好的效果。

**第六章 总 结**

**6.1 总结**

通过短短几周的学习，我从对STM32一无所知，对PID算法也是从未了解，到现在可以熟练运用HAL库进行较为系统的开发，设计并调试了三个平衡车相关的PID算法，对控制有了一个更为直观的认识，不再满足于用51单片机点灯，更加向往高级的算法和功能实现。

**6.2特色设计**

1.机械设计上， 我使用自己设计的小车底板，结合各种成熟的模块，对于硬件的容错率大大提升，可以专注于软件

2.硬件电路上，我采用自己构建原理图库，元器件库，集成库，自己画原理图和PCB的方式开发小车主板，借此机会学习了双层板的绘制以及各个元素的规则设置，对AD的操作更加熟练。

3.车模的选择上，由于本题目并未要求采用平衡车进行功能实现，而平衡车在多方面都会比困难。

4.巡线功能的实现上，我采用其他人都没有使用的线性CCD模块，对于赛道的适应性比红外对管要好很多。

**6.3 结束语**

通过这一个阶段的考核，我不仅仅是学习了STM32这一种单片机，通过踩各种坑，也对单片机的底层有了深刻的认识，知道了各种文件的作用，编译器的工作原理。同时，我也对C程序的开发，模块化程序的编写，各种软件驱动的使用更加熟悉。另外，每周一总结的考核方式，督促我不断地学习，避免了三天打鱼两天晒网的情况，让我在假期的时间里养成了良好的学习习惯。

同时感谢SIRB给我这么一个系统的学习机会，让我能和其他人一起学习新的知识，新的技术，让我在家也能感受到实验室的紧张刺激