

目录

第一部分 系统概览与技术要求	3
1 系统概览	3
1.1 系统简要介绍	3
2 技术要求	3
2.1 系统的动力学原理	3
2.2 系统的主要结构件及功能	3
第二部分 系统硬件选型及设计	4
3 主控制器硬件设计	4
3.1 主控制器硬件介绍	4
3.2 MCU 芯片选型	4
3.3 电源管理电路	5
4 电子调速器硬件设计	5
4.1 电子调速器硬件介绍	5
4.2 元器件选型	6
4.2.1 MCU 芯片选型	6
4.2.2 MOSFET 选型	6
4.2.3 栅极驱动芯片选型	6
4.3 外围电路设计	6
4.3.1 三相逆变电路	6
4.3.2 栅极驱动电路	7
4.3.3 反电动势过零检测电路	8
第三部分 系统软件设计	11
5 主控制器软件系统	11
5.1 软件系统介绍	11
5.2 电机控制算法	11
5.2.1 PID算法介绍	11
5.2.2 行进电机控制	12

5.2.3 橫向电机控制	15
5.3 软件濾波器	15
5.3.1 加权滑动平均濾波函数	15
5.3.2 卡尔曼濾波函数	16
6 电子调速器软件系统	18
6.1 软件系统介绍	18
6.2 系统初始化	19
6.3 状态控制与指示	20
6.4 PPM 编解码	21
6.5 六步换相法	22
第四部分 总结	24
7 总结与展望	24
7.1 感想总结	24
7.2 改进展望	24
A 主控制器硬件原理图	25
B 电子调速器硬件原理图	26

第一部分 系统概览与技术要求

1 系统概览

1.1 系统简要介绍

气垫船通过其底部安装的风扇或喷嘴产生高压空气，形成一个气垫，将船体抬升离开水面或地面一定的高度。这种空气垫层大大减少了船体与水面或地面的接触，从而显著降低了摩擦阻力，使得气垫船能够以更高的速度前进，甚至能够在泥滩、浅滩、沼泽、冰面等传统船只无法行驶的复杂地形上自由移动。在军事、救援、旅游等领域，气垫船因其全地形通过能力和高效的行驶性能得到了广泛应用。

2 技术要求

2.1 系统的动力学原理

在设计气垫船时，需考虑的几项关键动力学原理包括：

- i) 推力与阻力的平衡：气垫船需要有足够的推力克服阻力，推力主要由风扇或喷嘴产生，阻力则包括空气阻力、水面或地面摩擦阻力和波浪阻力等。
- ii) 空气动力学效应：船体设计应尽量减少空气阻力，这包括优化船体形状以降低风阻，以及确保风扇喷嘴的设计能够有效地将空气转化为向下的推力。
- iii) 浮力和稳定性的控制：通过气垫来保持浮力，确保船体稳定是设计时的重要考量。这通常涉及调整风扇的位置、数量和输出功率来精确控制浮力和稳定性。

2.2 系统的主要结构件及功能

气垫船的核心组件包括风扇、船体结构、气垫密封系统和推进系统。下面分别介绍这些组件的功能：

- i) 风扇：负责产生高速气流，推动空气进入船底形成气垫。
- ii) 船体结构：需要足够的强度和刚度以承载整个船体，。
- iii) 气垫密封系统：确保形成的气垫不被外部空气迅速泄漏，维持气垫的压力和形状。
- iv) 推进系统：通常由舵机和螺旋桨组成，负责控制气垫船的方向和提供额外的推进力。

第二部分 系统硬件选型及设计

3 主控制器硬件设计

3.1 主控制器硬件介绍

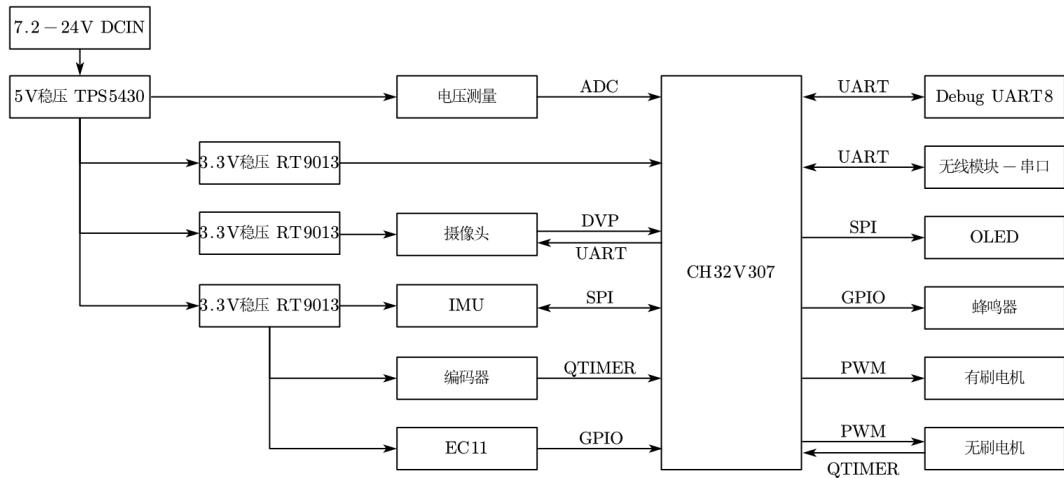
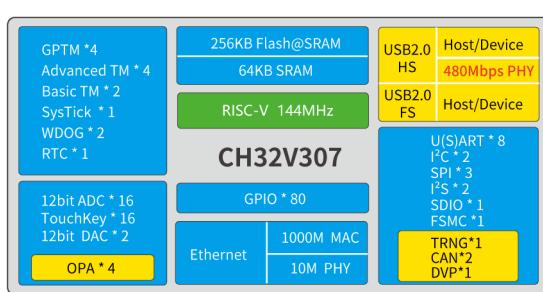


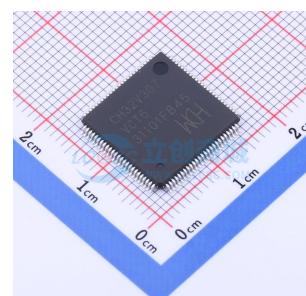
图 1: 主控制器硬件框图

3.2 MCU 芯片选型

CH32V307 是一款基于32位 RISC-V 架构的高性能互联型微控制器，具备 144MHz 主频、硬件浮点运算和增强的中断响应能力。其特点包括64KB SRAM、256KB Flash、多种低功耗模式、丰富的外设接口（如8个串口、USB 2.0高速接口、千兆以太网）、高精度 ADC/DAC 转换以及支持 CAN、SDIO、IIC、SPI 等通信协议。该系列还内置了随机数发生器、安全调试接口，并提供LQFP64M、LQFP100等封装形式，适合物联网、工业控制等领域。



(a) CH32V307 系统框图



(b) CH32V307 实物图

图 2: CH32V307 微控制器

3.3 电源管理电路

电源管理电路是负责控制和调节电子设备中电源的分配与使用的关键部分，确保设备能够在各种工作环境下高效、稳定地运行。它通过精确调节电压、电流、功率等参数，不仅维持设备的正常工作状态，还能优化能耗、延长电池寿命，并保障系统在过压、过流或短路等极端情况下的安全。这种灵活而高效的电源管理在现代电子系统设计中不可或缺，是提升系统整体性能和用户体验的核心环节。为满足需要，本系统上存在3种供电电压：

- i) 电机供电使用航模锂电池，充满时电压在11.1V左右。
- ii) 部分数字器件使用直流5V，选用降压转换器 TPS5430。
- iii) 部分数字器件使用直流3.3V，选用线性稳压器 RT9013-33GB。

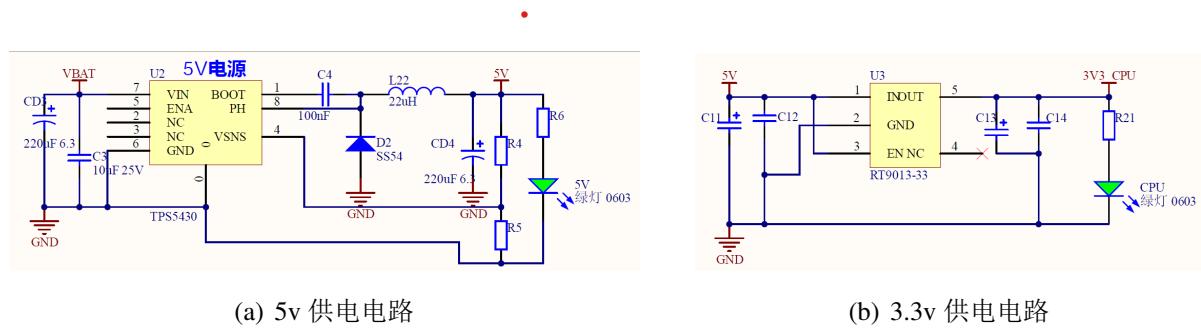


图 3: 电源管理电路原理图

其中，降压转换器 TPS5430 的输出电压 V_{OUT} 与反馈电阻满足如下关系式

$$R_5 = \frac{R_4 \times 1.221}{V_{OUT} - 1.221}$$

4 电子调速器硬件设计

4.1 电子调速器硬件介绍

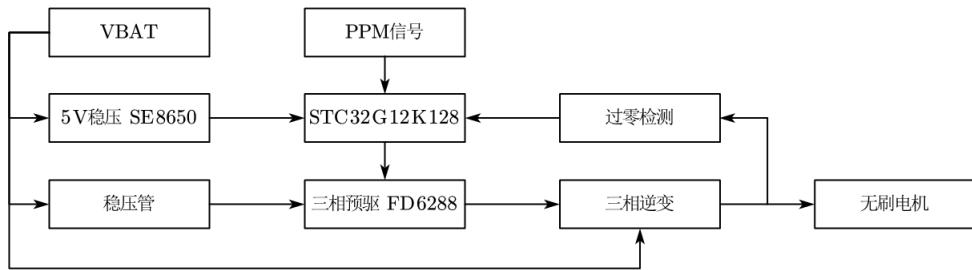


图 4: 电子调速器硬件框图

4.2 元器件选型

4.2.1 MCU 芯片选型

主控选用 STC32G12K128-LQFP48 驱动无刷电机，这款单片机特点是主频最大可达 35MHz、两组8通道 PWM 通道、一个比较器，比较器输入端可以连接到ADC通道或者IO、宽范围工作电压 1.9 - 5.5v、UART 数量有4个。

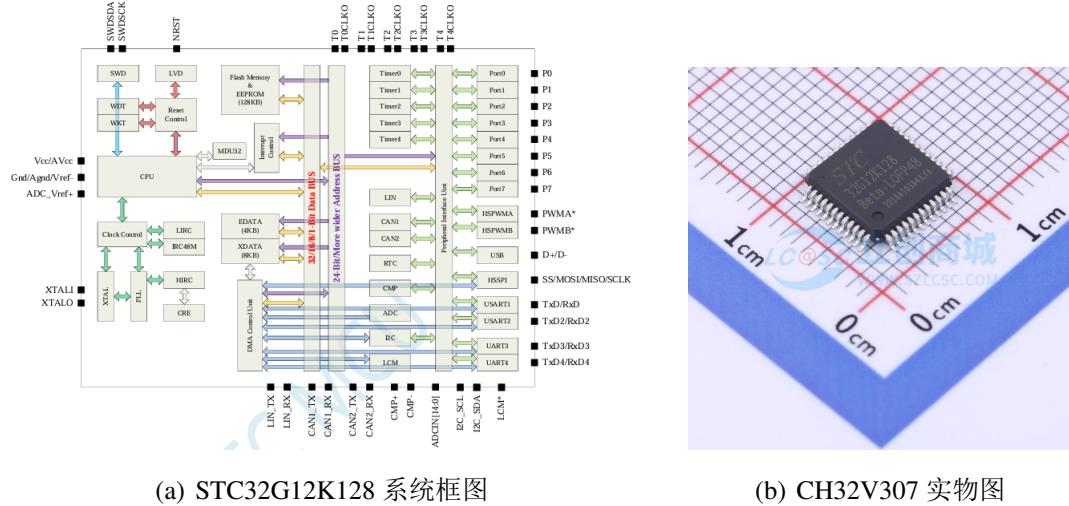


图 5: STC32G12K128 微控制器

4.2.2 MOSFET 选型

由于单片机内部已经集成了比较器，因此外围器件选择主要就集中在了栅极驱动芯片、MOS管型号的选择上，MOS管我们选择的型号是 TPN2R703NL，这款 MOS 电流高达45A，10V的时候内阻低至2.2毫欧，开启电压低至2.3V左右，性价比较高。

4.2.3 栅极驱动芯片选型

栅极驱动芯片选用 FD6288Q，主要因为该芯片集成有三个独立半桥栅极驱动器，这样的高集成度减少了外部组件的数量，从而降低了故障风险。此外，FD6288Q支持高电压操作，具备高速开关特性，并且内置了过流保护、欠压锁定等保护机制，增强了系统的整体稳定性。

4.3 外围电路设计

4.3.1 三相逆变电路

三相逆变器是一种将直流电转化为三相交流电的电力电子设备，广泛应用于电动机驱动、风力发电、光伏发电等场合。其核心工作原理是利用功率半导体器件（如IGBT、

MOSFET等) 的开关动作, 通过脉宽调制 (PWM) 技术来控制输出电压和频率。三相逆变器输出的三相交流电具有相互错开120度的相位, 能够为三相电动机或其他负载提供平稳的动力。

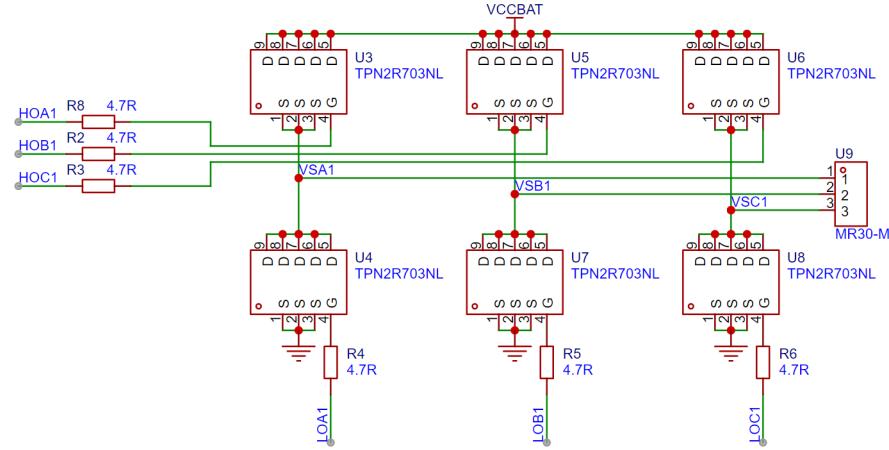


图 6: 三相逆变电路原理图

4.3.2 棚极驱动电路

在无刷直流电机 (BLDC) 驱动器中, 棚极驱动电路用于控制功率 MOSFET, 驱动电机绕组电流。由于 MOSFET 棚极的电容较大, 单靠 MCU 的输出驱动能力往往不足以在短时间内快速充电和放电。因此, 棚极驱动电路能够提供足够的电流来快速切换功率器件, 确保开关的快速导通和关断。另一方面, 棚极驱动电路通常包括死区时间控制功能, 在半桥或全桥电路中, 确保了上下桥臂的开关不会同时导通, 以避免直通现象导致的短路。

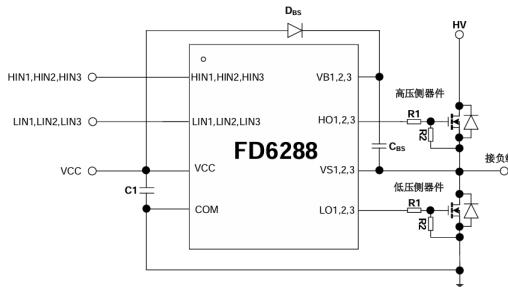


图 7: FD6288 典型应用电路

自举二极管 D_{BS} 的作用是为自举电容提供充电路径, 确保其在每个开关周期内都能够被有效充电, 维持所需的电压水平。当低侧开关导通时, 电流通过自举二极管流入电容, 使其充电。选择自举二极管时, 应考虑其反向击穿电压和反向恢复时间这两个关键参数。高反向击穿电压能够保证自举二极管在高侧开关导通时承受较高的电压应力, 而短的反向恢复时间则能确保二极管在从正向导通到反向截止的过程中迅速切换。

基于这些要求，设计中选择了B5819WS肖特基二极管，其40V的反向击穿电压和10ns的反向恢复时间能够在高频开关操作中提供良好的稳定性和可靠性。

自举电容 C_{BS} 则用于储存能量，并在高侧开关导通时为栅极驱动电路提供所需的电压。在半桥驱动电路中，自举电容通过自举二极管在低侧开关导通时充电，当高侧开关导通时，自举电容释放储存的电压，驱动高侧MOSFET的栅极。因此，自举电容的选择和设计对系统的正常运行也至关重要。

依据FD6288Q数据手册，选择陶瓷电容并使用如下公式计算其容量。

$$C_{BS} \geq 10 \times \frac{Q_g}{V_{cc} - V_F}$$

式中， C_{BS} 为自举电容的容值， Q_g 为MOSFET的栅极电荷， V_{cc} 为FD6288Q的供电电压， V_F 为自举二极管的正向压降值。

根据计算的结果，考虑到自举电容如果选用的容值远大于计算结果，会增加电路的充放电时间，影响系统的响应速度和动态性能。如果选用的容值小于计算结果，则没有容量来存储足够的电荷，无法提供足够的瞬态响应，导致栅极驱动电压不足。最后选用了具有 $4.7\mu F$ 容值、耐压50V的陶瓷电容，以确保在高频开关操作中的高可靠性和稳定性。

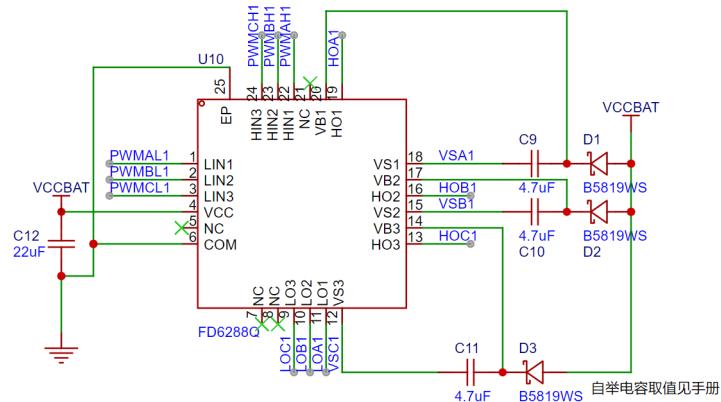


图 8: 栅极驱动电路原理图

4.3.3 反电动势过零检测电路

无刷直流电机通过电子控制器对定子绕组进行顺序激励，生成旋转磁场。这个旋转磁场与转子上的永磁体相互作用，产生电磁转矩，从而驱动转子旋转。为了确保电机能够平稳运行，电子控制器需要精确掌握转子的实时位置。只有在确认转子到达指定位置后，才能执行换相操作，以保持电机的平滑运转。转子位置检测常用有三种方式，基于电压的过零比较、霍尔传感器、磁编码器。

我们采用低成本的过零比较检测，如图9所示，电路通常由分压电路和比较电路组成。

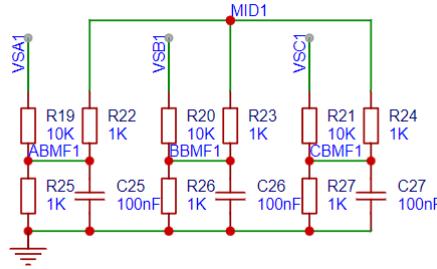


图 9: 反电动势过零检测电路原理图

如图10所示, 是 BLDC 的定子三相绕组星形连接的等效电路图。因为BLDC三相线圈的公共点, 即中性点 N 在电机内部没有引出, 不能直接测量得到中性点的电势 U_N , 但是也可以构造虚拟中性点 N' 来模拟真实中性点的电势.

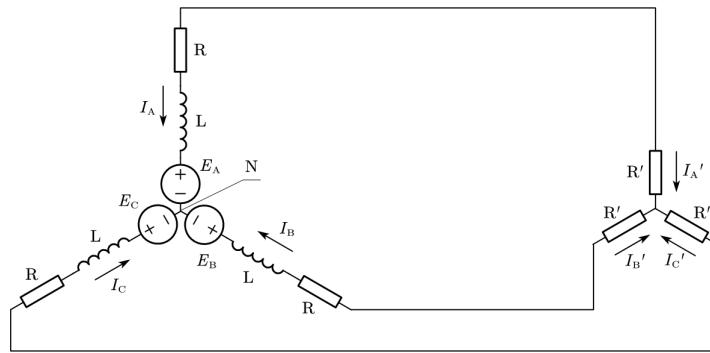


图 10: 虚拟中性点示意图

根据KVL、KCL对 BLDC 三相绕组建立数学模型

$$\left\{ \begin{array}{l} U_A = I_A R + L \frac{dI_A}{dt} + E_A + U_N \\ U_B = I_B R + L \frac{dI_B}{dt} + E_B + U_N \\ U_C = I_C R + L \frac{dI_C}{dt} + E_C + U_N \\ I_A + I_B + I_C = 0 \\ E_A + E_B + E_C = 0 \end{array} \right. \quad (1)$$

式中, U_A 、 U_B 、 U_C 为三相绕组A、B、C三个极上的电势; I_A 、 I_B 、 I_C 为三相绕组内的电流; E_A 、 E_B 、 E_C 为三相绕组上的反电动势; U_N 为三相绕组的公共点N的电动势。

因为B相悬空的缘故有 $I_B = 0$, 易知

$$E_B = U_B - U_N \quad (2)$$

也就是说, 悬空相上的反电动势就是悬空相对应极(端)的电势减去中性点的电势。

式(1)中第 5 个式子, 在悬空相反电动势过零时刻成立, 即此时反电动势之和为零。将式(1)中前三行式子两侧各自相加, 并考虑后两个式子, 得到悬空相反电动势过零时刻实际三相绕组中性点 N 的电势为

$$U_A + U_B + U_C = 3U_N \quad (3)$$

再对虚拟中性点用KCL、KVL列得

$$\begin{cases} U_A = I'_A R + U'_N \\ U_B = I'_B R + U'_N \\ U_C = I'_C R + U'_N \\ I'_A + I'_B + I'_C = 0 \end{cases} \quad (4)$$

对式(4)中前三式两侧相加, 得到虚拟中性点 N' 的电势

$$U_A + U_B + U_C = 3U'_{N'} \quad (5)$$

结合式(3)和式(5)得到

$$U_N = U'_{N'} \quad (6)$$

即反电动势过零点时, 虚拟中性点与三相绕组中性点电压一致。

结合式(2)和式(6), 有 $E_B = U_B - U'_{N'}$ 。反电动势过零点时刻 $E_B = 0$, 则只需要检测并验证下式是否成立即可。

$$U_B \stackrel{?}{=} U'_{N'}$$

实际应用中, 虚拟中性点的检测电路如图11所示。电路中, 分压以获得合适检测范围; 滤波滤除尖峰、高次谐波。

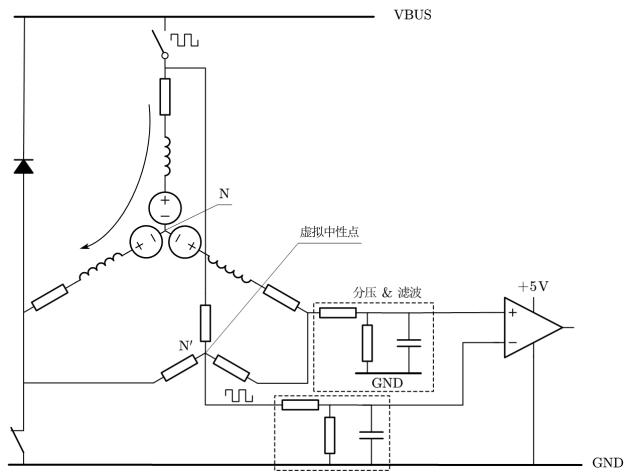


图 11: 过零检测电路示意图

第三部分 系统软件设计

5 主控制器软件系统

5.1 软件系统介绍

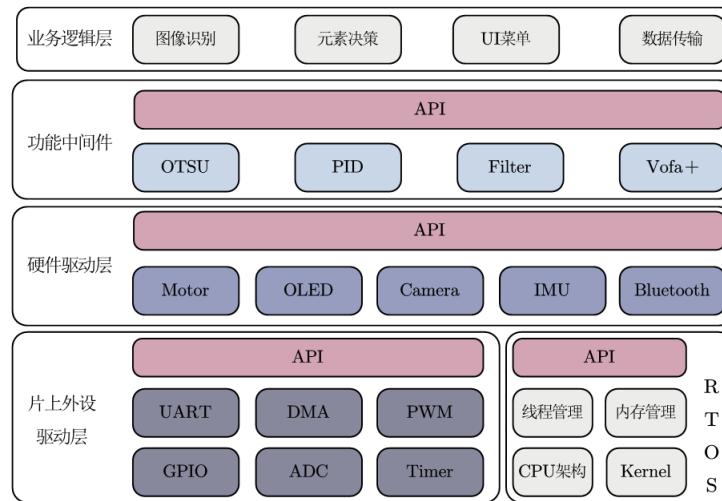


图 12: 系统软件架构图

5.2 电机控制算法

5.2.1 PID算法介绍

PID控制是工程实际中应用最为广泛的调节器控制方法。问世至今70多年来，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。PID控制原理框图如图13。

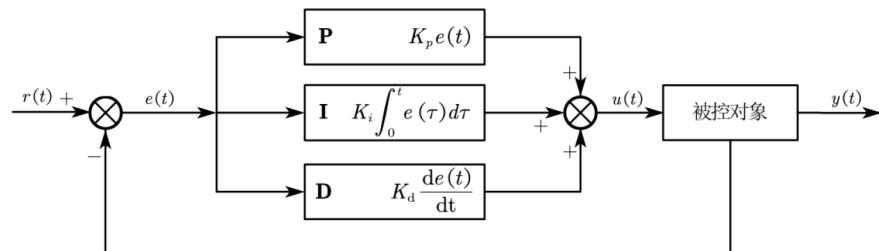


图 13: PID控制流图

反馈 $e(t)$ 代表理想输入与实际输出的误差，这个误差信号被送到控制器，控制器算出误差信号的积分值和微分值，并将它们与原误差信号进行线性组合，得到输出量 $u(t)$ 。

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt}$$

其中， K_p 、 K_i 、 K_d 分别称为比例系数、积分系数、微分系数。 $u(t)$ 接着被送到了执行机构，这样就获得了新的输出信号，这个新的输出信号被再次送到感应器以发现新的误差信号，这个过程就这样周而复始地进行。数字控制系统中，PID控制器是通过计算机PID控制算法程序实现的。计算机直接数字控制系统大多数是采样-数据控制系统。进入计算机的连续-时间信号，必须经过采样和整量化后，变成数字量，方能进入计算机的存贮器和寄存器，而在数字计算机中的计算和处理，不论是积分还是微分，只能用数值计算去逼近。

5.2.2 行进电机控制

在本系统中，方向环使用了串级PID将角度环和角速度环串联起来，从而达到更高的响应速度和稳定性。其具体结构如图14。

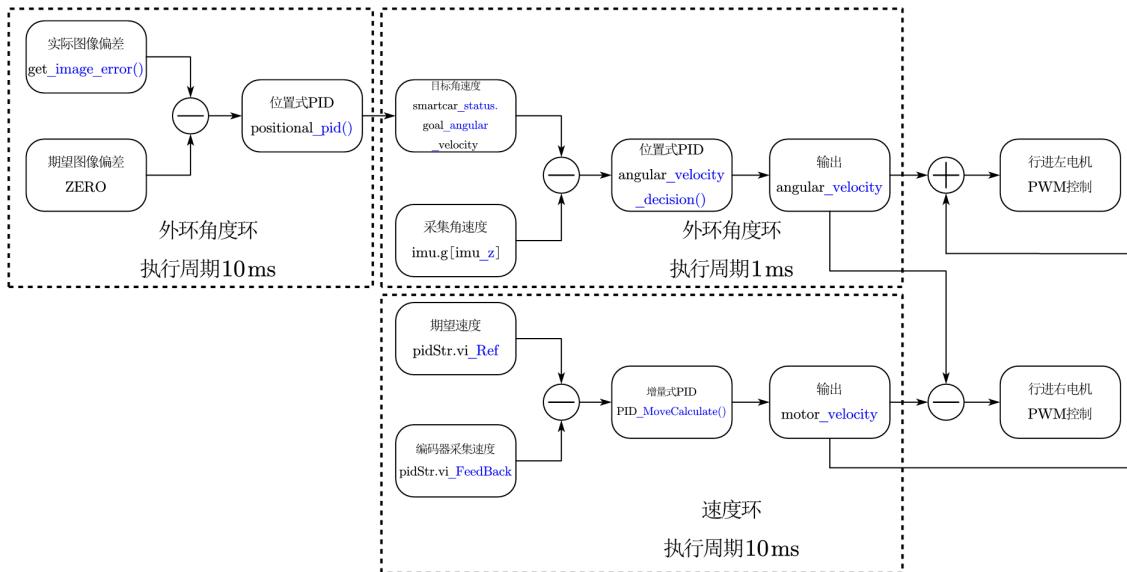


图 14: 行进电机PID控制方案

根据框图可以看出，外环角度环期望的图像偏差为0，实际的图像偏差是通过摄像头采集到的图像经过 MCU 处理和识别，寻线而得出的赛道中线。在直道的时候，角度环的输出接近于0，而此时角速度环的输入量仅有陀螺仪角速度值。在直道的时候，内环的作用是抑制系统的转向。在弯道的时候，角度环的输出，当作角速度环的输入传入位置式 PID 系统，并与陀螺仪采集到的角速度值进行对比，经过计算达到期望角度环输出的角速度值，从而让系统更快，更稳的转向。另一方面，速度环对于角速度环来说，它是一个干扰量，会干扰系统的转向，所以对于干扰量就需要将其周期中断间隔设置较大。

在位置式 PID 结构体 Pid 中, set_point 为设定的期望目标, K_p、K_i、K_d 分别为系统的比例、积分、微分系数, sum_error 为累计误差和, last_error 为上次误差, last_2_error 为上上次误差。

在增量式 PID 结构体 PIDStruct 中, vi_ref 为设定的期望目标速度, vi_FeedBack 为编码器采集到的速度实际值, vi_PreError 为速度误差值, vi_PreDerror 为上一次速度误差之差, v_Kp、v_Ki、v_Kd 分别为系统的比例、积分、微分系数, vi_PreU 为系统输出值。

```

4 //位置式PID模型
5 @typedef struct
6 {
7     float set_point;
8     float K_p;
9     float K_i;
10    float K_d;
11    float sum_error;
12    float last_error;
13    float last_2_error;
14 }Pid;

```

(a) 位置式PID结构体

```

17 //PID速控模型
18 @typedef struct
19 {
20     float vi_Ref;
21     float vi_FeedBack;
22     float vi_PreError;
23     float vi_PreDerror;
24     float v_Kp;
25     float v_Ki;
26     float v_Kd;
27     float vi_PreU;
28 }PIDStruct;

```

(b) 速控（增量式）PID结构体

图 15: 离散化PID模型参量

如图16这个函数主要实现了位置式PID算法, 用传入的目标值减去实际值得到误差值得到比例项, 在对误差值进行累加得到积分项, 用本次误差减去上次的误差得到微分项, 然后通过前面章节介绍的PID公式实现PID算法, 并返回实际控制值。

$$u(k) = K_p \text{err}(k) + K_i \sum \text{err}(k) + K_d (\text{err}(k) - \text{err}(k-1))$$

这个公式就是代码第57行中的公式形式, 公式和代码的计算方式基本一致, 只不过在公式中第二项的Ki是使用的对误差积分, 在代码中变成了对误差的累加, 虽然表达形式不一样, 但是达到的效果和目的是一样的。计算过后将误差传递用于下一次使用, 并将实际值返回。

如图17这个函数主要实现了增量式PID算法, 用传入速度的误差值减去上一次误差值得到比例项, 在对误差值进行累加得到积分项, 用上一次速度的误差之差得到微分项, 然后通过前面章节介绍的PID公式实现PID算法, 并返回实际控制值。

$$\Delta u(k) = K_p [\text{err}(k) - \text{err}(k-1)] + K_i \text{err}(k) + K_d [\text{err}(k) - 2\text{err}(k-1) + \text{err}(k-2)]$$

$$u(k) = u(k-1) + \Delta u(k)$$

这两个公式就是代码第193行中的公式形式, 公式和代码的计算方式基本一致, 可以看出增量式的PID是与近三次的误差有关; 虽然代码与公式的表达形式不一样, 但是达到的效果和目的是一样的。计算过后将误差传递用于下一次使用, 并将实际值返回。

```

39@float positional_pid(Pid *pid, float sensor_val)
40 {
41     float output;
42     float error = 0.0f;
43
44     float p_error = 0.0f;//当前误差项
45     float i_error = 0.0f;//积分误差项
46     float d_error = 0.0f;//微分误差项
47
48     error = pid->set_point - sensor_val;
49
50     p_error = error;
51     i_error = pid->sum_error;
52     d_error = error - pid->last_error;
53
54     //积分限幅
55     range_protect(&pid->sum_error, 5000.0f, -5000.0f, 3);
56
57     output = pid->K_p * p_error + pid->K_i * i_error + pid->K_d * d_error;
58
59     //输出限幅
60     if(output > 5000.0f) output = 5000.0f;
61     else if(output < -5000.0f) output = -5000.0f;
62
63     pid->last_error = error;
64     pid->sum_error += error;
65
66     return output;
67 }

```

图 16: 位置式PID函数实现

```

175@signed int PID_MoveCalculate(PIDStruct *pp)
176 {
177     float error,d_error,dd_error;
178     float I_error;
179
180     error = pp->vi_Ref - pp->vi_FeedBack;
181     d_error = error - pp->vi_PreError;
182     dd_error = d_error - pp->vi_PreDerror;
183
184     pp->vi_PreError = error;
185     pp->vi_PreDerror = d_error;
186
187@    if( ( error < VV_DEADLINE ) && ( error > -VV_DEADLINE ) )
188    {
189        ;
190    }
191    else
192    {
193        pp->vl_PreU += (pp -> v_Kp * d_error + pp -> v_Ki * error + pp->v_Kd*dd_error);
194    }
195
196@    if( pp->vl_PreU >= MOTOR_PWM_MAX )
197    {
198        pp->vl_PreU = MOTOR_PWM_MAX;
199    }
200@    else if( pp->vl_PreU <= MOTOR_PWM_MIN )
201    {
202        pp->vl_PreU = MOTOR_PWM_MIN;
203    }
204
205    return (pp->vl_PreU);
206 }

```

图 17: 速控（增量式）PID函数实现

5.2.3 横向电机控制

横向电机则通过图像的偏差进行位置式 PID 的计算，用来提供系统转向时所需要的向心力，就能实现更快更稳定的转向。

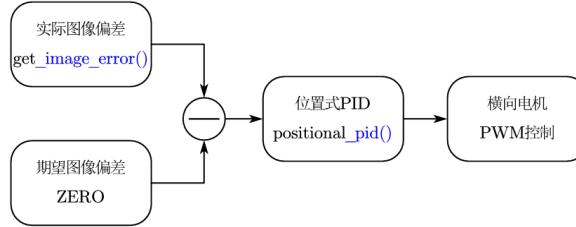


图 18: 横向电机PID控制方案

5.3 软件滤波器

5.3.1 加权滑动平均滤波函数

加权滑动平均滤波算法是一种经典的滤波方法。其主要思路是对信号进行滑动窗口处理，窗口内的数据进行加权平均化，以得到平滑后的信号。这样可以有效地去除周期性噪声和高频噪声，同时保留信号的整体趋势。滑动平均滤波算法的实现如图19。

$$\text{WMA}_t = \frac{\sum_{i=1}^n w_i X_{t+1-i}}{\sum_{i=1}^n w_i}$$

式中， X_t 为 t 时刻的观测值， w_i 为第*i*个数据点的权重， n 为滑动窗口的长度。

```

10 const float filter_weight[WIN_SIZE] = { 10, 15, 20, 25, 30, 40, 50, 60, 70, 80};
11
12 float SlidingFilter(float array[WIN_SIZE], float value)
13 {
14     //窗口滑动
15     for(int i = 0; i < WIN_SIZE - 1; i++)
16     {
17         array[i] = array[i + 1];
18     }
19     array[WIN_SIZE - 1] = value;
20
21     //加权
22     float result = 0.0f;
23     float weight_sum = 0.0f;
24     for(uint8 i = 0; i < WIN_SIZE; i++)
25     {
26         result += filter_weight[i] * array[i];
27         weight_sum += filter_weight[i];
28     }
29     if(weight_sum != 0)
30     {
31         result = result / weight_sum;
32     }
33     return result;
34 }
  
```

图 19: 加权滑动平均滤波函数实现

5.3.2 卡尔曼滤波函数

卡尔曼滤波（Kalman Filter）是一种用于估计动态系统状态的递归算法，广泛应用于信号处理、控制系统等领域。其基本思想是基于对系统状态的预测和观测数据进行最佳估计，并通过不断更新估计值来降低噪声的影响，从而得到更精确的系统状态。

卡尔曼滤波主要分为预测和更新两部分，需要构建系统的状态方程和观测方程，且已知系统的初始状态。在预测阶段，滤波器使用上一状态的估计，做出对当前状态的估计。在更新阶段，滤波器利用对当前状态的观测值优化在预测阶段获得的预测值，以获得一个更精确的新估计值。

其预测和更新过程如下：

$$\hat{X}_{k-1|k-1} \xrightarrow{\text{Prediction}} \hat{X}_{k|k-1} \xrightarrow{\text{Correction}} \hat{X}_{k|k}$$

式中， $\hat{X}_{k|k-1} = E(X_k|Y_1, Y_2, \dots, Y_{k-1})$ 表示已知过去 $k-1$ 个时刻状态，在时刻 k 时的状态估计。

在预测 $\hat{X}_{k-1|k-1} \Rightarrow \hat{X}_{k|k-1}$ 步骤中，根据上一时刻的状态和控制量，预测当前时刻的状态。这个预测值是一个估计值，因为它还没有考虑当前时刻的观测值。预测值的误差协方差矩阵是通过上一时刻的误差协方差矩阵和系统噪声协方差矩阵计算得到的。

$$\text{预测步骤: } \begin{cases} \hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1} \\ P_k^- = AP_{k-1}A^T + Q \end{cases}$$

式中，上标“-”代表先验； \hat{x}_k^- 为 k 时刻的先验状态估计； \hat{x}_{k-1} 代表 $k-1$ 时刻的后验状态估计； u_{k-1} 为 $k-1$ 时刻的输入向量； A 为状态矩阵； B 为输入矩阵； P_k^- 为 k 时刻的先验状态估计误差协方差矩阵； P_{k-1} 为 $k-1$ 时刻的后验状态估计误差协方差矩阵； Q 为过程噪声协方差矩阵。

在更新 $\hat{X}_{k|k-1} \Rightarrow \hat{X}_{k|k}$ 步骤中，根据当前时刻的观测值和预测值，计算出当前时刻的状态估计值。这个估计值是一个更加准确的估计值，因为它已经考虑了当前时刻的观测值。状态估计值的误差协方差矩阵是通过预测步骤中计算得到的误差协方差矩阵、观测噪声协方差矩阵和卡尔曼增益计算得到的。

$$\text{更新步骤: } \begin{cases} K_k = P_k^- H_m^T (H_m P_k^- H_m^T + R)^{-1} \\ \hat{x}_k = \hat{x}_k^- + K_k (z_k - H_m \hat{x}_k^-) \\ P_k = (I - K_k H_m) P_k^- \end{cases}$$

式中， K_k 为 k 时刻的卡尔曼增益； H_m 为观测矩阵，下标 m 代表 measure； R 为测量噪声协方差矩阵； z_k 为 k 时刻的观测向量； I 为单位矩阵。

现在，我们只关注解算 X 轴的角度，那么系统状态矩阵为

$$X = \begin{bmatrix} \theta \\ \omega_b \end{bmatrix}$$

式中， θ 为横滚角度； ω_b 为角速度漂移量，下标 b 即为 bias。

第一步，计算先验状态估计：

在这个系统中，由于 IMU 不可避免的漂移，所以就必须从 IMU 的测量值 X_{Gyro} 上减去漂移量 ω_b 得到实际的角速度值，再进行积分求角度 θ 。对于漂移量 ω_b ，我们认为同一个 IMU 的漂移每次都是一致的。于是有

$$\theta_k = \theta_{k-1} + (X_{Gyro} - \omega_b) dt$$

式中， X_{Gyro} 为 IMU 采集到的测量角速度。

将状态矩阵代入上式，进而得到系统先验状态估计方程

$$\hat{x}_k^- = \begin{bmatrix} \theta^- \\ \omega_b^- \end{bmatrix} = \begin{bmatrix} 1 & -dt \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \theta \\ \omega_b \end{bmatrix}_{[k-1]} + \begin{bmatrix} dt \\ 0 \end{bmatrix} X_{Gyro}$$

式中，下标 $[k-1]$ 代表第 $k-1$ 次的系统后验状态估计方程。

第二步，计算先验状态估计误差协方差矩阵：

由第一步先验状态估计，有状态向量

$$A = \begin{bmatrix} 1 & -dt \\ 0 & 1 \end{bmatrix}$$

由于 IMU 的角度漂移噪声和角速度漂移噪声相互独立，则测量噪声协方差矩阵 Q 有

$$Q = \begin{bmatrix} \text{Cov}(\theta, \theta) & \text{Cov}(\omega_b, \theta) \\ \text{Cov}(\theta, \omega_b) & \text{Cov}(\omega_b, \omega_b) \end{bmatrix} = \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_b \end{bmatrix}$$

将 A 和 Q 代入式中有

$$P_k^- = \begin{bmatrix} P_{00}^- & P_{01}^- \\ P_{10}^- & P_{11}^- \end{bmatrix} = \begin{bmatrix} 1 & -dt \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P_{00[k-1]} & P_{01[k-1]} \\ P_{10[k-1]} & P_{11[k-1]} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -dt & 1 \end{bmatrix} + \begin{bmatrix} Q_\theta & 0 \\ 0 & Q_b \end{bmatrix}$$

展开上式，有

$$\begin{cases} P_{00}^- = P_{00[k-1]} + Q_\theta - (P_{01[k-1]} - P_{10[k-1]}) dt + \cancel{P_{11[k-1]} dt^2} \\ P_{01}^- = P_{00[k-1]} - P_{11[k-1]} dt \\ P_{10}^- = P_{10[k-1]} - P_{11[k-1]} dt \\ P_{11}^- = P_{11[k-1]} + Q_b \end{cases}$$

由于 dt^2 是高阶小量，故可以忽略 $P_{11[k-1]} dt^2$ 项以减少计算时间。

第三步，计算卡尔曼增益矩阵：

将 P_k^- 和 $H_m = [1 \ 0]^T$ 代入式中，有

$$K_k = \begin{bmatrix} K_0 \\ K_1 \end{bmatrix} = \frac{\begin{bmatrix} P_{00}^- & P_{01}^- \\ P_{10}^- & P_{11}^- \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}}{\begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} P_{00}^- & P_{01}^- \\ P_{10}^- & P_{11}^- \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + R_\theta}$$

式中 R_θ 为角度测量噪声，一般为常数。

展开上式，有

$$\begin{cases} K_0 = P_{00}^{-1}(P_{00}^{-1} + R_\theta)^{-1} \\ K_1 = P_{10}^{-1}(P_{00}^{-1} + R_\theta)^{-1} \end{cases}$$

第四步，更新后验状态估计：

将 \mathbf{x}_k^- 、 \mathbf{H}_m 、 \mathbf{K}_k 和 $z_k = \theta_{new}$ 代入式中，有

$$\hat{\mathbf{x}}_k = \begin{bmatrix} \theta \\ \omega_b \end{bmatrix} = \begin{bmatrix} \theta^- \\ \omega_b^- \end{bmatrix} + \begin{bmatrix} K_0 \\ K_1 \end{bmatrix} \left(\theta_{new} - \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} \theta^- \\ \omega_b^- \end{bmatrix} \right)$$

展开上式，有

$$\begin{cases} \theta = \theta^- + K_0(\theta_{new} - \theta^-) \\ \omega_b = \omega_b^- + K_1(\theta_{new} - \theta^-) \end{cases}$$

第五步，更新后验状态估计误差协方差矩阵：

将 \mathbf{P}_k^- 、 \mathbf{H}_m 和 \mathbf{K}_k 代入式中，有

$$\mathbf{P}_k = \begin{bmatrix} P_{00} & P_{01} \\ P_{10} & P_{11} \end{bmatrix} = \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} K_0 \\ K_1 \end{bmatrix} \begin{bmatrix} 1 & 0 \end{bmatrix} \right) \begin{bmatrix} P_{00}^- & P_{01}^- \\ P_{10}^- & P_{11}^- \end{bmatrix}$$

展开上式，有

$$\begin{cases} P_{00} = P_{00}^- - K_0 P_{00}^- \\ P_{01} = P_{01}^- - K_0 P_{01}^- \\ P_{10} = P_{10}^- - K_1 P_{10}^- \\ P_{11} = P_{11}^- - K_1 P_{11}^- \end{cases}$$

到这里为止，最后计算得出的后验状态估计误差协方差矩阵 \mathbf{P}_k 为下一循环的卡尔曼滤波做准备。

6 电子调速器软件系统

6.1 软件系统介绍

无感无刷电调软件系统的功能是通过预定的算法实现对电机转子位置和速度的实时估算，从而在没有物理传感器的情况下进行高精度的电机控制，能够在保持电机运行稳定性的同时，实现高效、低能耗的控制。该控制通常依赖于反电势过零检测技术。通过检测绕组上的反电势信号的零点位置来确定转子的位置，并生成相应的 PWM 信号来调节电机的转速、方向以及输出扭矩。

为了实现无感无刷电机的平稳启动，尤其是在低速或静止状态下，软件系统也集成了启动算法。启动阶段结束后，系统会逐渐切换到基于反电势的控制模式，实现更高效的运行。此外，该系统也配备了通信接口模块，PPM、UART，用于接收外部控制命令。

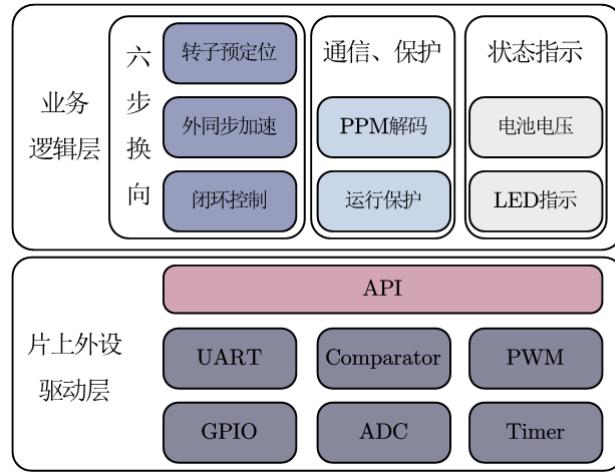


图 20: 系统软件架构图

6.2 系统初始化

系统启动初期，主函数先对 STC32G 芯片的各个外设模块，如定时器、比较器、PWM、ADC 等进行初始化配置，以确保系统的正常运行。同时，对软件资源，如变量初始化、PWM 占空比默认值、电机死区设置等进行默认处理，确保系统运行时的数据一致性，以避免在 H 桥驱动电机时出现上下桥臂同时导通的情况，确保电机的安全运行。此外，其他如中断优先级的设置、串口通信的初始化等，都在系统启动的初期得到合理配置，为后续系统的稳定运行奠定基础。

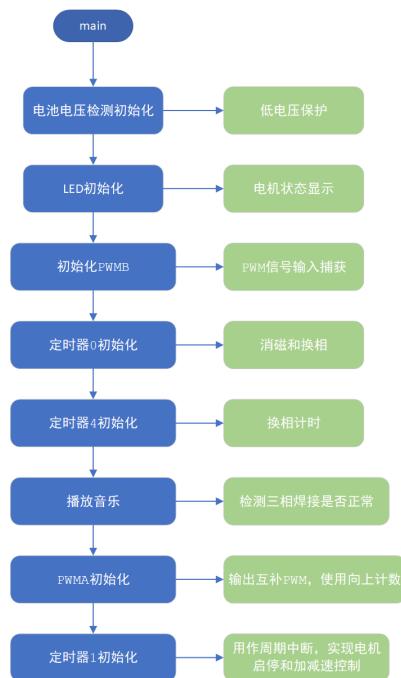


图 21: 电调初始化流程图

6.3 状态控制与指示

周期中断以每 50us 的固定频率运行，主要负责电机的启动运行、停止、故障等信息，方便用户实时了解系统的运行情况；另一方面，周期中断承担着电机启动、停止以及堵转检测等关键任务。在电机启动过程中，它按照预定的启动流程和时序，协调各个模块之间的关系，确保电机能够平稳、顺利地启动。同时，通过对电机运行状态的实时监测，一旦检测到电机出现堵转情况，能够迅速停止电机运转，并发出故障报警信号。此外，周期中断还肩负着电池低电压保护的重要职责。通过对电池电压的实时监测和分析，当检测到电池电压低于设定的安全阈值时，立即停止电机运行，以保护电池免受过度放电的损害。

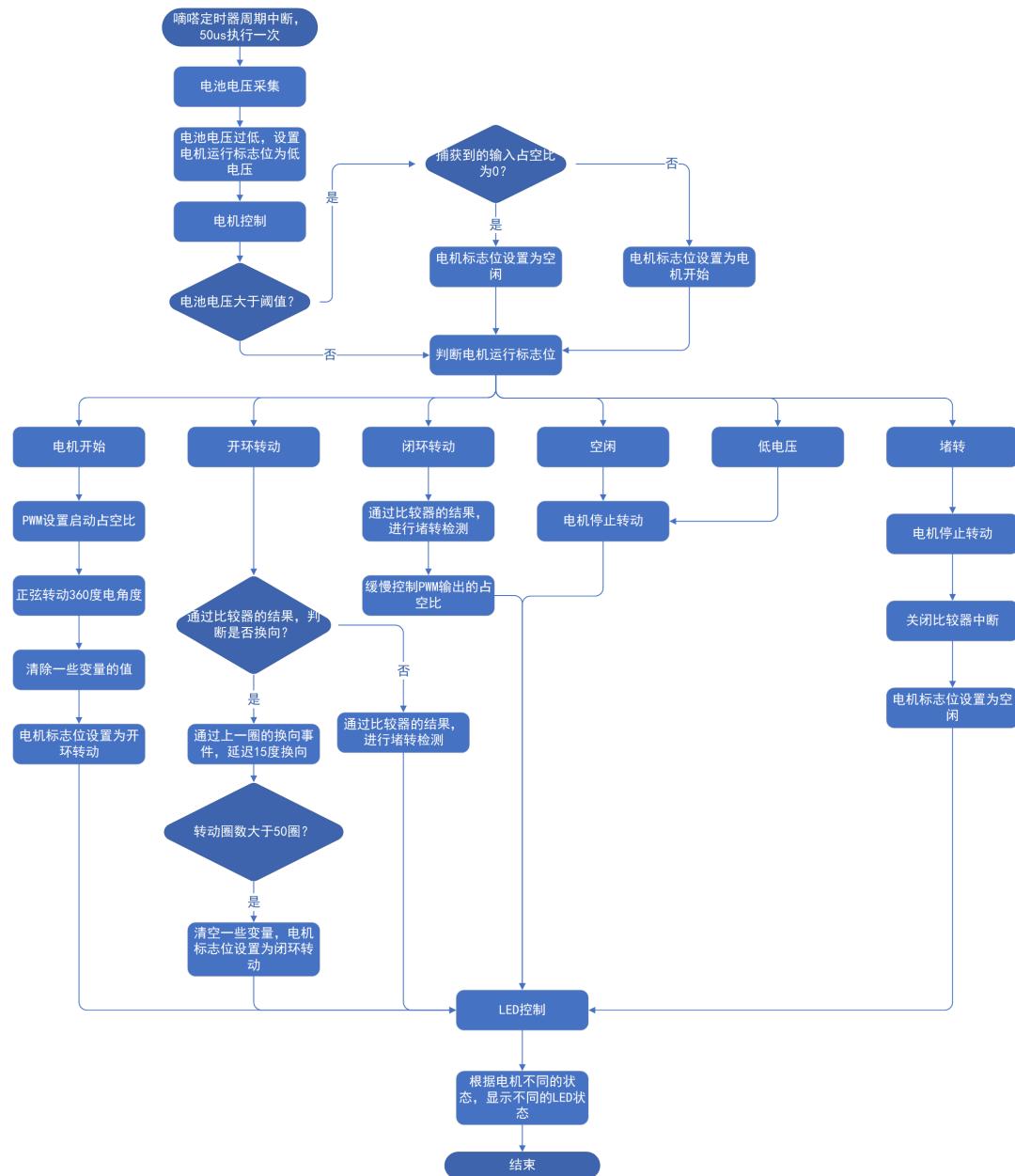


图 22: 电调状态控制流程图

6.4 PPM 编解码

脉冲位置调制（Pulse Position Modulation, PPM）的原理是通过编码产生脉冲信号，然后对基带信号进行调制，也就是将信号信息附加到 PPM 脉冲信号上，调制信号决定载波脉冲的时间。这是遥控模型中比较通用的一种信号格式，PPM 解码原理是通过检测给定频率的 PWM 信号的占空比来获取指令信号。其原理见图23，信号频率为50Hz，一个周期为20ms。对于电调来讲，脉宽为1ms表示停转，脉宽为2ms表示满油门运转，其间的各点按比例换算：比如脉宽为1.5ms就表示50%油门等等。

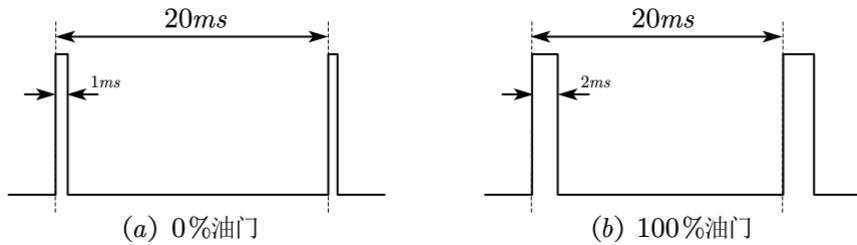


图 23: PPM信号图解

在如图24的 PWMB 中断服务程序中，系统专注于高效处理电调输入信号，确保对电机控制信号的实时响应。通过精确测量输入信号的高电平持续时间以及整个周期时间，能够准确地计算出油门信号的变化情况，进而获取油门大小信息。具体来说，当接收到PWM信号时，中断服务程序会首先记录信号的上升沿和下降沿，计算高电平时间和整个周期时间。根据这些数据，系统能够确定油门信号的占空比，从而准确反映电调输入的油门大小。

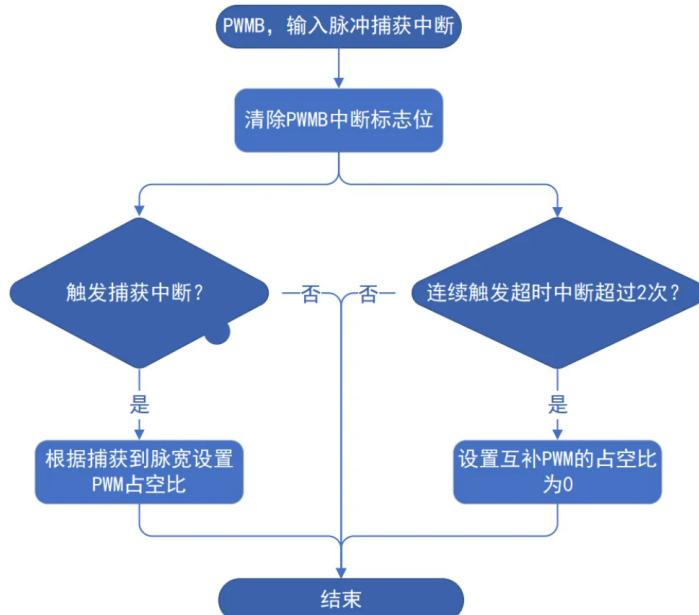


图 24: 电调油门控制流程图

6.5 六步换相法

六步换相法的基本思想是通过顺序地给电机的三相绕组施加电压，使得电流方向在电机的三个绕组中不断变化，从而产生旋转磁场，驱动电机转动。具体来说，电机的转动是通过控制绕组的电流变化来实现的。每当一个绕组的电流达到最大值后，电流方向被切换到下一个绕组，这样就能形成一个不断旋转的磁场，使得电机的转子随着外部的磁场变化而转动。

其将每个电机周期分为六个步骤，如图25。每个步骤通过激活两个绕组（一个绕组为高电平，另一个为低电平）并关闭第三个绕组来产生磁场。在这些步骤中，电流的变化顺序按照特定的逻辑进行，以确保电流在各个绕组之间的顺序变化，保持电机的平稳转动。通过这种方法，电机每转一圈，换相就会进行六次，从而实现无刷电机的平稳运行。

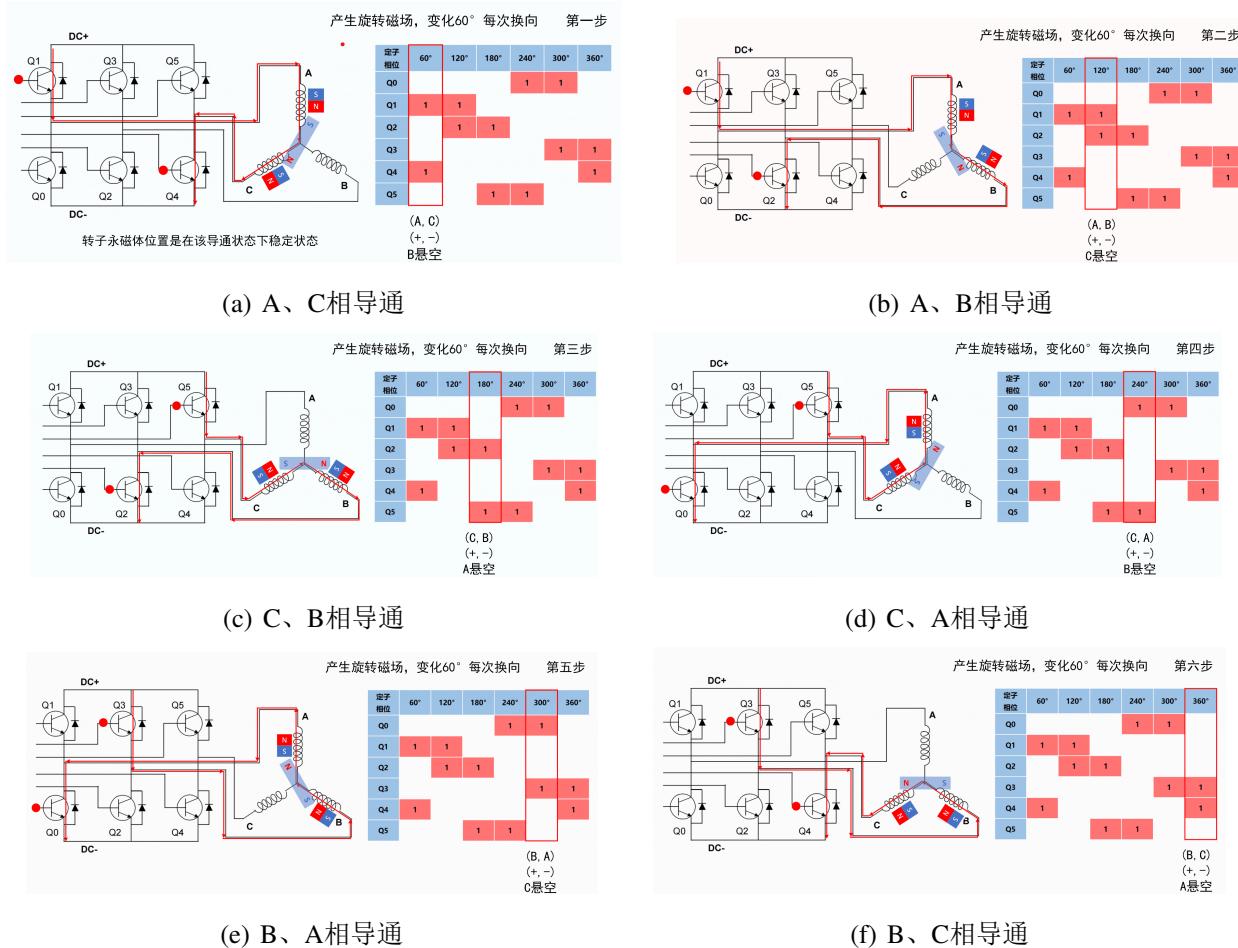


图 25: 六步换相法图解

电机的换相过程是无刷直流电机驱动控制的核心，定时器 0 中断通过定时触发，按照电机运行的时序，根据=转子位置和预设的换相策略，按顺序切换定子的电流方向，精确控制每次换相的时机，从而驱动无刷电机旋转。

此外，定时器 0 中断还负责电机的消磁处理，如图??。消磁过程是为了防止电机在运

行过程中出现过剩的磁场，减少电磁损耗和热量积累，保护电机的运行效率与寿命。通常，在每次换相完成后，为了消除定子绕组中的剩余磁场，需要进行一定的延时处理。

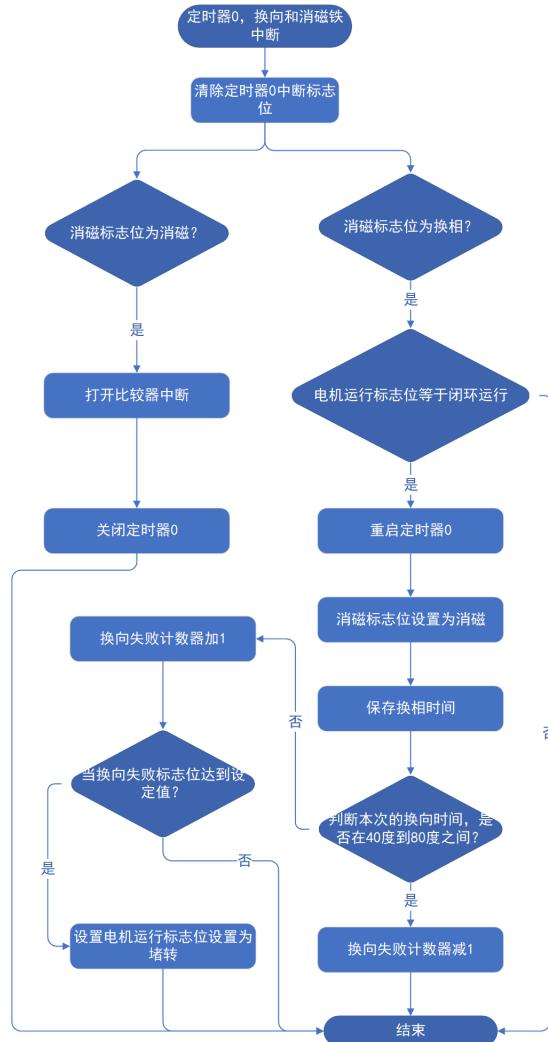


图 26: 电调六步换相流程图

第四部分 总结

7 总结与展望

7.1 感想总结

毕业要求	课程对应毕业要求指标点
3.设计/ 开发解决方案	3-2 能够针对特定指标需求，设计并实现功能完整的硬件和软件系统，包括整体架构设计、各模块交互和数据通信等（系统设计）
4. 研究	4-2 能够根据方案，运用实验工具、仪器，开展实验，对实验结果进行分析与解释，并通过信息综合得到合理有效的结论
10.沟通	10-1能够就电子信息工程领域的复杂工程问题撰写报告，设计文稿，并具有较好的语言表达和沟通能力，能够清晰陈述观点和回答问题

在参与整个项目的过程中，我深刻体会到了团队合作、技术创新以及工程实践的重要性。作为一项综合性极强的嵌入式系统设计扩充，该项目不仅需要扎实的理论基础，还需要高效的实践和调试能力。在设计与制作过程中，我学到了如何将传感器、嵌入式控制、电机驱动等技术结合在一起，解决实际问题。

在构建项目的过程中，我们面临了不少挑战。首先是平衡系统的稳定和速度。为了实现这一目标，我们反复调试气垫系统的气压分布以及行进电机的角度和功率，通过不断的试验找到了最优配置。其次，为了让气垫船能够在赛道上平稳前进并快速完成任务，我们采用了多级 PID 控制算法来调整船体姿态，同时通过卡尔曼滤波算法对传感器实时采集系统的位置信息进行数据处理，确保路径规划和导航的准确性。

该系统的设计和调试不仅让我提升了动手实践和编程调试的能力，也增强了我对多学科融合技术的理解与应用。竞赛和课程中的每一次挑战和收获，都是宝贵的经验。最重要的是，这次经历让我更加坚定了继续探索智能控制技术和创新设计的决心。

7.2 改进展望

在硬件层面，可以对无刷电机过零检测电路中的 C25、C26、C27 三颗滤波电容作移除处理。这三颗电容是用来滤除采集三点的高频分量的，但可能导致过零信号滞后，进而引发电机换相问题。

在软件层面，无刷电机启动直接采用了强拉换相方式。未来可以引入更为先进的正弦启动模式。通过使电机柔和地转动 360 度电角度后再切换至开环转动检测，可以极大地优化了启动过程，不仅能够降低启动冲击，还能显著提高了反电动势的检测精度，为电机的平稳运行奠定良好开端。

A 主控制器硬件原理图

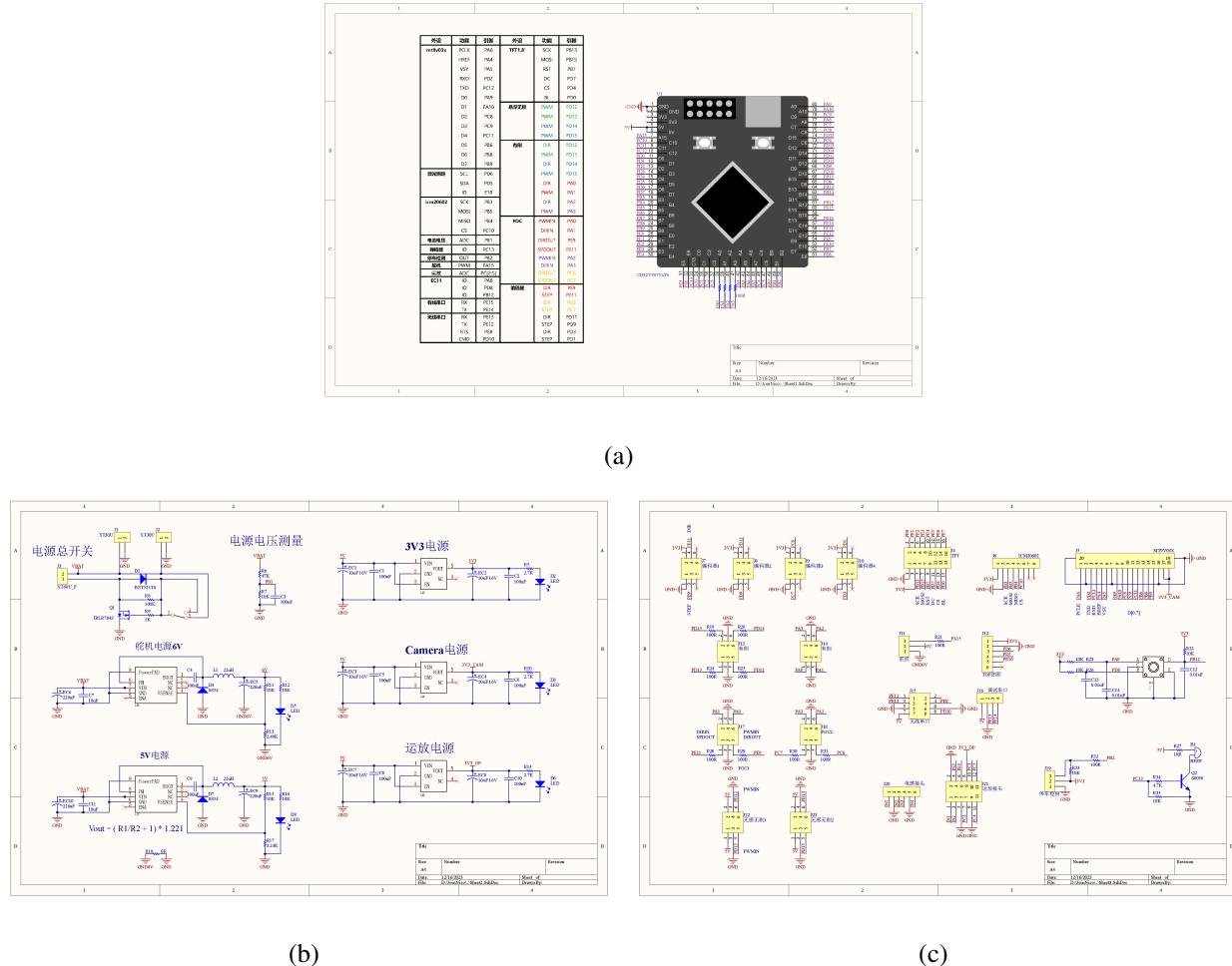
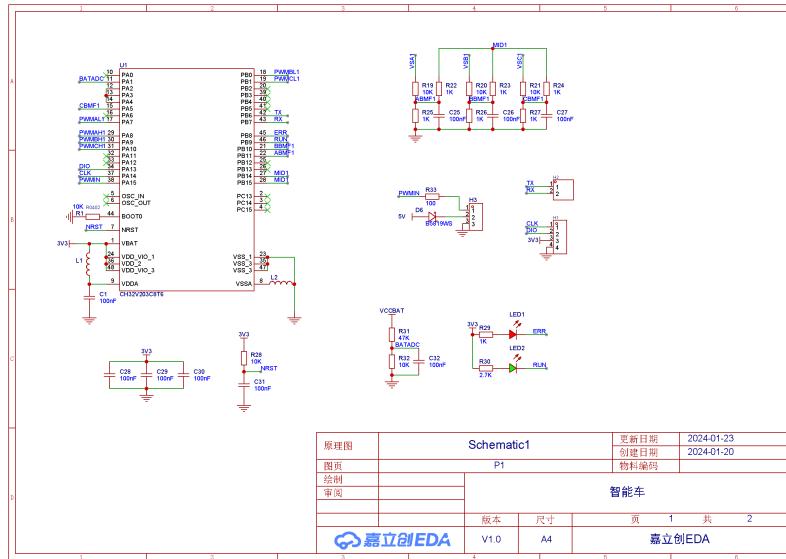
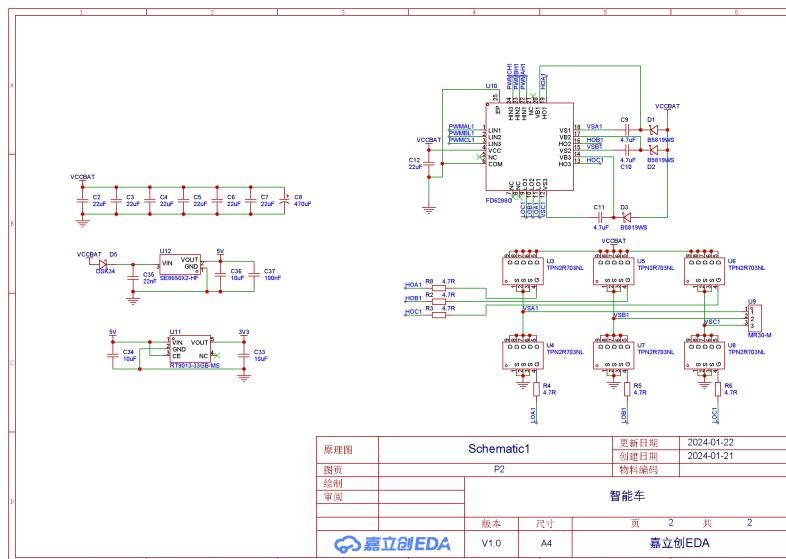


图 27: 主控制器硬件原理图

B 电子调速器硬件原理图



(a)



(b)

图 28: 电子调速器硬件原理图