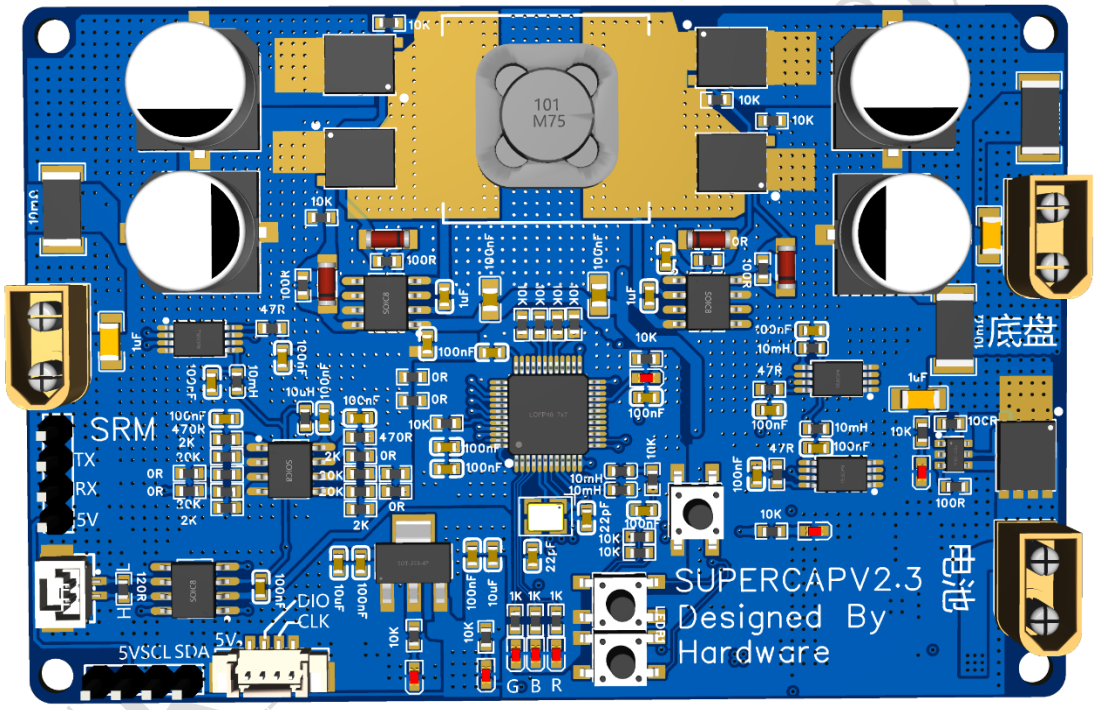
基于四开关 BUCK-BOOST 智能功率调节控制系统——超级电容



基于四开关 BUCK-BOOST 智能功率调节控制系统——超级电容

技术研发	唐源涛 胡时宇
报告撰写	柳怡冰 吕翰琪 吴昊泽
后期调试	柳怡冰 吕翰琪 吴昊泽
顾问	唐源涛 胡时宇 李璟怿

上海大学

目录

第一部分：选题说明

一、选题背景

随着科技的快速发展，能源利用效率和系统的稳定性成为了各个领域中关注的重点，特别是在需要高性能、高可靠性电源的系统中，如无人机、机器人、电动汽车等，电源管理系统的性能直接决定了设备的运行效率和安全性。因此，开发一种高效、稳定的电源控制系统显得尤为重要。

传统的电源控制系统在应对复杂多变的工作环境时，往往存在以下问题：（1）功率调节不精准，存在较大的误差，无法满足高精度应用的需求；（2）系统稳定性差，无法应对复杂的工作环境，影响设备的正常运行；（3）能源利用效率低，浪费宝贵的能源资源。

为了解决上述问题，提出一种基于四开关 BUCK-BOOST 电路和超级电容组的智能电源控制系统，该系统通过软件与硬件的结合，实现了对电源的高效、稳定管理，提高了设备的运行效率和安全性，为各种应用场景提供了可靠的电源方案。

二、方案关键功能

1. 智能功率调节。通过 STM32F334C8TX 的 ADC 模块实时检测系统的电压和电流，结合双串级 PID 控制算法，实现电流环、电压环和功率环的精细控制，确保电源的稳定输出和高效利用。
2. 高精度采样与滤波。采用两点拟合法计算误差参数，结合滑动窗口滤波器，

有效去除 ADC 采样中的噪声干扰，提高数据的准确性。

3. 状态自检与异常状况处理。实时监测供电端电压、负载端和功率控制板的状态，一旦发现异常情况，会通过灯光闪烁提示并采取相应措施，确保系统运行的可靠性。
4. CAN 通信与远程控制。利用 CAN 通信实现与负载端的可靠数据传输，同时支持远程控制功能，方便用户进行远程监控和管理。
5. 储能端管理。通过智能控制超级电容组的充电和放电过程，实现储能端的高效利用和长寿命运行。

第二部分：方案描述

一、硬件 PCB 设计

1 整体方案

本方案基于四开关 BUCK-BOOST 电路 设计了一款可以智能调节功率的控制系统。整个系统由供电端、功率控制端、储能端、负载端共四个部分组成，系统示意见图 1。

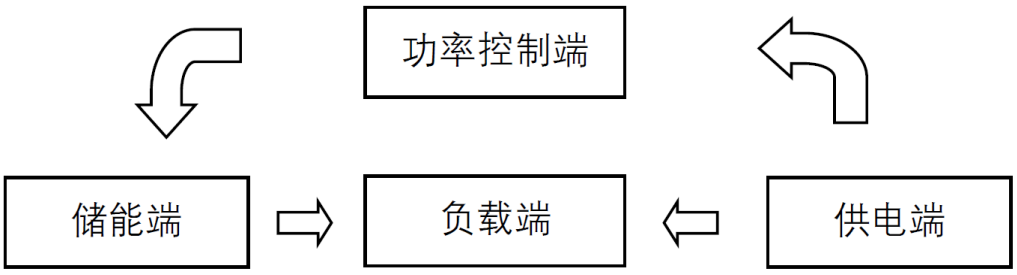


图 1 控制系统示意图

供电端由一块大疆 TB48 智能电池构成，电池参数见表 1。

表 1 大疆 TB48 智能电池主要参数

类别	参数
名称	Intelligent Flight Battery
型号	TB48
容量	5700 mAh
电压	22.8 V

电池类型	LiPo 6S
能量	129.96 Wh
电池整体重量	670 g
工作环境温度	-10℃ 至 40℃
充电环境温度	0℃ 至 40℃
最大充电功率	180 W

功率控制端由 MCU 控制电路、降压电路、采样电路、放大电路、BUCK-BOOST 数控模块及其外围电路构成。

储能端由 9 个 2.7V 60F 的法拉电容串联 及其外围电路构成。

负载端为用电器，损耗功率。

2 功率控制端电路设计

2.1 四开关 BUCK-BOOST 主拓扑设计

2.1.1 主拓扑电路器件选型

四开关 BUCK-BOOST 主拓扑电路主要由 TI 公司生产的 CSD19534Q5A N 沟道场效应管、栅极驱动芯片 、功率电感、电解电容，器件参数见表 2.1.1。

表 2.1.1 主拓扑电路器件参数

器件名称	数值/封装	器件型号	生产商
N 沟道场效应管	POWERVDFN-8_L6.0-W5.0-P1.27-BL	CSD19534Q5A	TEXAS INSTRUMENTS
半桥栅极驱动器	SOIC-8_L4.9-W3.9-P1.27-LS6.0-BL	UCC27211D	TEXAS INSTRUMENTS
功率电感	22 uH IND-SMD_L17.5-W17.5	CDRH104NP-221MC	Sumida
铝电解电容器	220 uF	EEETK1V221UP	PANASONIC

2.1.2 主拓扑电路设计说明

四开关 BUCK-BOOST 变换器原理图见图 2.1.1。

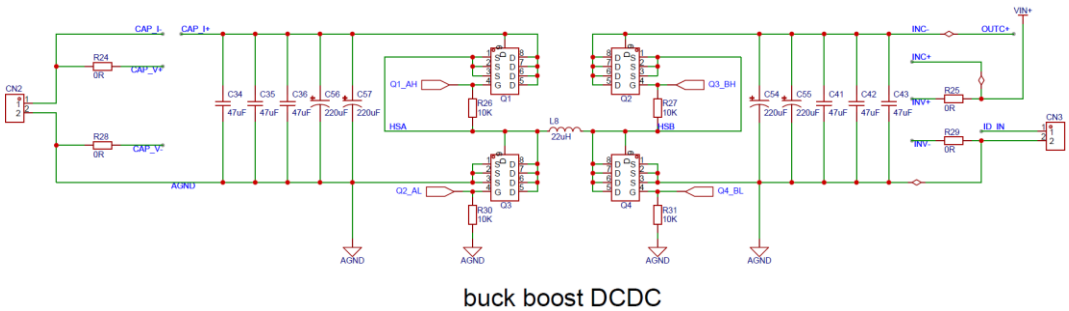


图 2.1.1 四开关 BUCK-BOOST 变换器原理图

四开关 Buck-Boost 变换器中只含一个功率电感，两个桥臂上有四个开关场

效应管，输入输出电压极性相同。输出电压与输入电压的关系如下：

$$Vout = \frac{d_1}{1 - d_2} Vin$$

通过代码控制占空比d_1和d_2大小，进而控制电路工作在升压（Boost 状态）、降压（Buck 状态）或升降压（输入输出电压相接近的Buck-Boost）的状态

Buck 模式下需要的电感大小计算如下：

$$L = \frac{(V_{inMax} - Vout)\, d_1}{0.5f \times IoutMax} = \frac{26-5}{100k \times 0.5 \times 8} \times \frac{5}{26} = 10\text{uH}$$

Boost 模式下需要的电感大小计算如下：

$$L = \frac{k \times Vout \times (1-d_1)^2 d_1}{2f \times IoutMax \times 10\%} = \frac{24}{100k \times 2 \times 1.4} \times \left(\frac{19}{24}\right)^2 \times \frac{5}{24} = 11.2\text{uH}$$

选择较大的电感值且留有余量，该功率电感取值 22uH。

输出最大纹波电压 $U_{ripple} = Vout_{Max} \times 0.5\% = 0.12V$

忽略直插式电解电容的等效串联电阻，只考虑电容充电引起的电容纹波，因此所需电容的容值理论计算值如下：

$$C = \frac{\delta I}{f \times U_{ripple} \times 2} = \frac{0.25 \times 8}{100k \times 0.12 \times 2} = 83.3uF$$

由此得到，主拓扑电路的输入输出电容需要大于 83.3uF，电路设计时留有余量，单端电容采用两个 220uF 铝电解电容，和三个 47uF 贴片电容并联，主拓扑电路完全对称。

由于 PWM 脉冲频率较高，数字电路上的脉冲干扰电压会影响模拟电路，因此采用 0Ω 电阻将数字地和模拟地连接起来，减小干扰。

公式参数说明见表 2.1.2。

表 2.1.2 公式参数说明

参数	意义
$Vout$	输出电压
Vin	输入电压
d_1	左桥臂占空比
d_2	右桥臂占空比
V_{inMax}	最大输入电压
$Iout_{Max}$	最大输出电流
f	主拓扑工作频率
K	纹波系数

2.1.3 栅极驱动电路设计

栅极驱动电路见图 2.1.2。

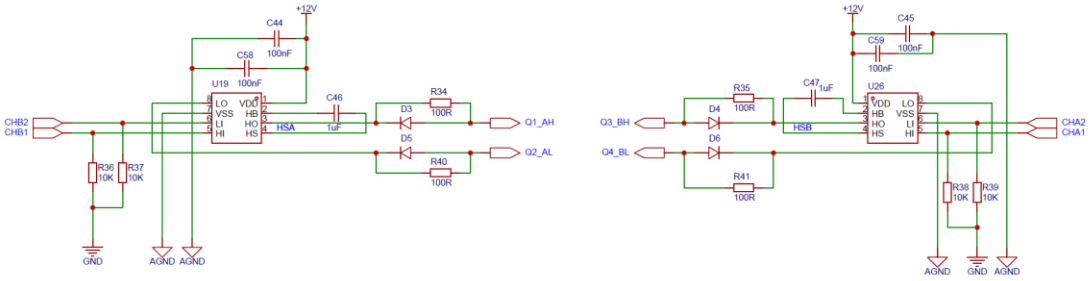


图 2.1.2 栅极驱动电路

MOSFET 驱动器采用 TEXAS INSTRUMENTS 生产的 UCC2721x 芯片，该芯片开关节点（HS 引脚）最高可处理-18V 电压，从而保护高侧通道不受寄生电感和杂散电容所固有的负电压影响。UCC27210（伪 CMOS 输入）和 UCC27211（TTL 输入）已经增加了滞后特性，从而使得到模拟或数字脉宽调制 (PWM) 控制器接口的抗扰度得到了增强。

主拓扑中使用 TEXAS INSTRUMENTS 生产的 SON 5mm x 6mmNexFET™ 功率 MOSFET，耐压值达 100V，最小漏源导通电阻为 12.6mΩ，最大持续漏极电流（受芯片限制，T=25°C 时测得）为 44A; 而本设计中最大电压为 26V，远低于 MOSFET 耐压；最大峰值电流为 12A，远小于 MOSFET 最大持续漏极电流。

CHB1、CHB2 分别是主控芯片 STM32F334C8TX 的 PA10、PA11 引脚，这两个引脚使能为 HRTIM1 TimerB，TimerB 具有两个输出通道，通过相应配置，

使 CHB1 和 CHB2 输出一对互补的 PWM 信号。Q1_AH、Q2_AL 分别连接左桥臂高边驱动管 Q1、左桥臂低边驱动管 Q2 的门极。

在 PWM 信号驱动 MOSFET 时，为避免高低边驱动管同时导通，需要在高低边驱动管之间设置一个短暂的信号延迟，即为死区时间，保证至少有一个 MOSFET 是关闭的，从而防止穿通现象的发生。

2.2 信号采样与运放电路设计

电池输入电压、电容组输出电压通过 TEXAS INSTRUMENTS 生产的运算放大器 OPA2350，采用差分电路将电压按一定比例缩小至 ADC 能够采样的范围，再使用 STM32F334C8TX 的 ADC 采样，通过软件计算出真实电池输入电压、电容组输出电压值。电压采样电路如图 2.2.1 所示。

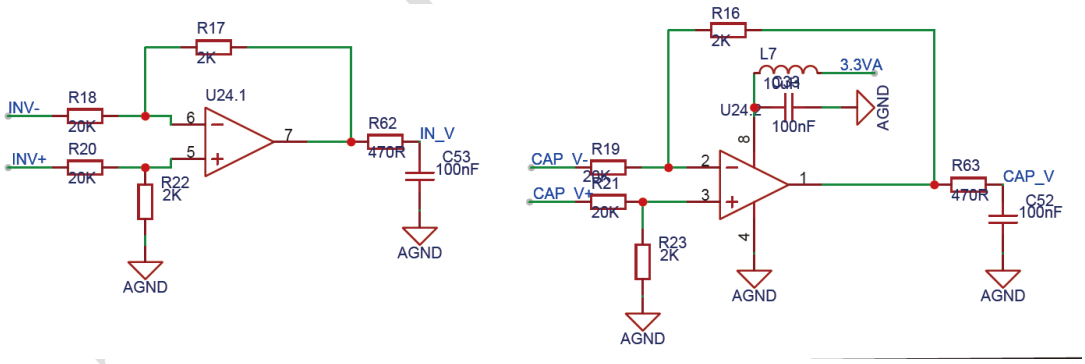


图 2.2.1 电压采样电路

以下图为例，INV-和 INV+分别为 0V 和 24V（供电端提供的电压值为 24V），设定基准电压为 3.3V，则通过该运算放大电路后，IN_V 为 3.28V（由 Multisim 仿真得出），在 STM32F334C8TX 的 ADC 采样范围内，放大倍数约为 0.1375, 仿真见图 2.2.2。

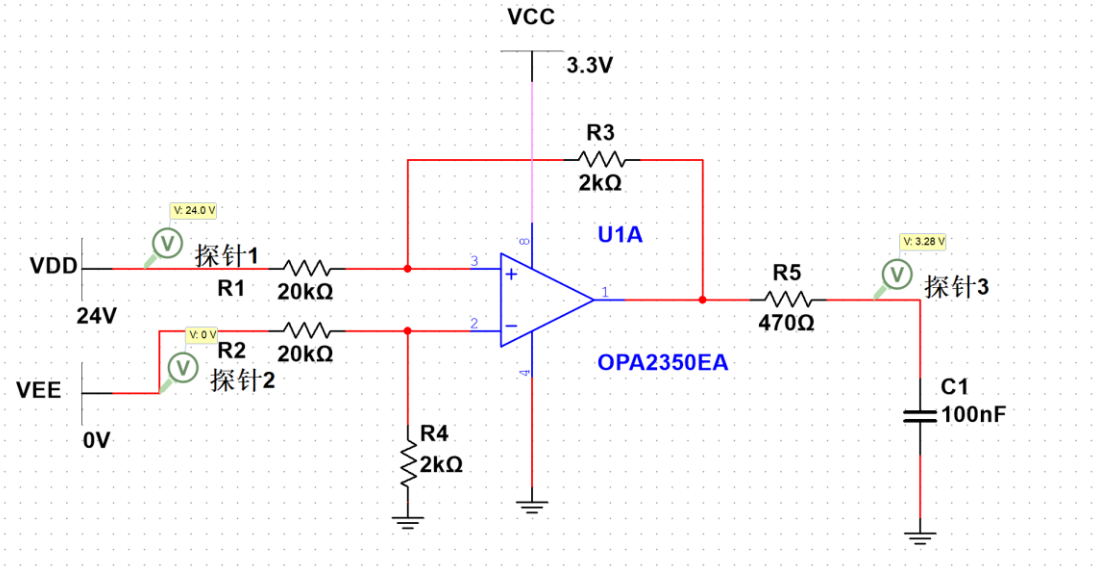


图 2.2.2 Multisim 仿真电压采样电路

电池输入电流、电容组电流、输出电流检测电路通过 TEXAS INSTRUMENTS 生产的 INA240A2 采样差分放大电路实现：采样电阻为 10mΩ，考虑采样电阻两端电压降较小，不利于直接采样，且本设计中电流双向流动，STM32F334 无法对负电压进行 ADC 检测，因此需要设定一个基准电压，将放大后的负电压抬升至正电压供 STM32F334 采样。通过软件解算，得到电池输入电流、电容组电流、输出电流的实际值。输出电压计算公式，电压反解电流公式如下：

INA240A2-Q1 增益：50V/V

V_{out} = VCC/2+I×R×50V/V

I=（V_{out} -VCC/2）/ R×50

电流检测电路见图 2.2.3。

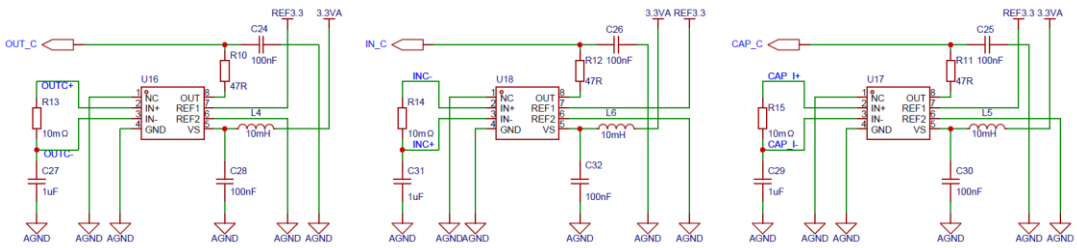


图 2.2.3 电流检测电路

2.3 主拓扑外围保护电路设计

对于本设计负载端，存在电池和电容组两个电源同时供电的情况，因此需要设计相应电路防止电流倒灌。采用 TEXAS INSTRUMENTS 生产的 LM5050，与外部 MOSFET 配合工作，实现理想二极管的功能，从而在一定程度上防止了电流倒灌，当检测到 V_{DS}（漏源电压）上的电压异常时，LM5050 会控制 V_{GS}（栅源电压）上的电压，使 MOSFET 工作在合适的状态，从而防止电流倒灌。保护电路见图 2.3.1。

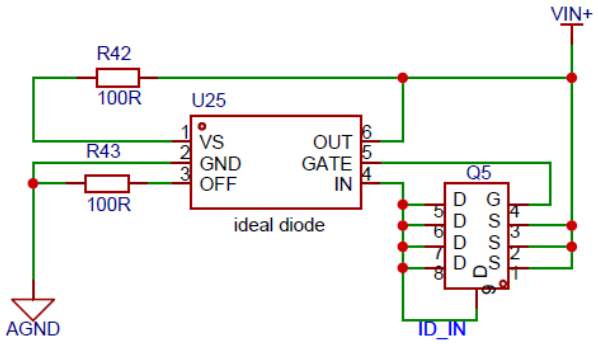


图 2.3.1 外围保护电路

2.4 降压电路设计

降压电路的设计原则是将电池提供的 24V 电压经过相应的 DC-DC 变换，降至 3.3V，给 STM32F334、OPA2350、INA240 提供工作电压。

采用 MPS 生产的 MP4462 将输入电压（电压范围为 22V-26V）降压至 12V，采用 MPS 生产的 MP1584 将 12V 电压降压至 5V，降压电路原理图见图 2.4.1。

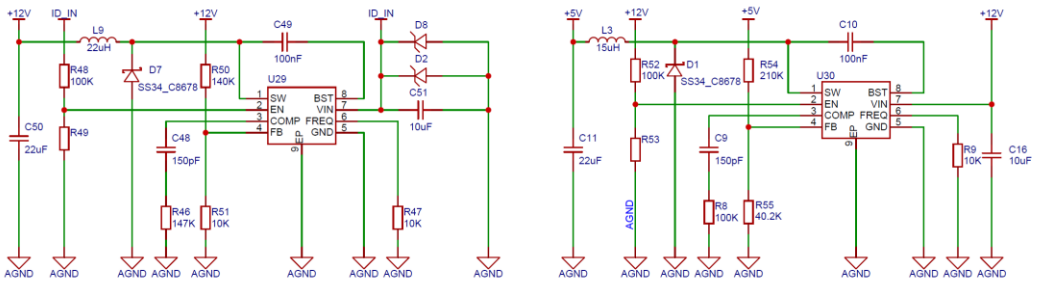


图 2.4.1 输入电压降至 12V，12V 降至 5V 原理图

采用艾迈斯半导体生产的 AMS1117_3.3 将 5V 降压至 3.3V，一次降压为 REF3.3，目的是给电流检测电路提供基准电压；另一次降压为 3.3VA，目的是给 STM32F334、OPA2350、INA240 提供工作电压。

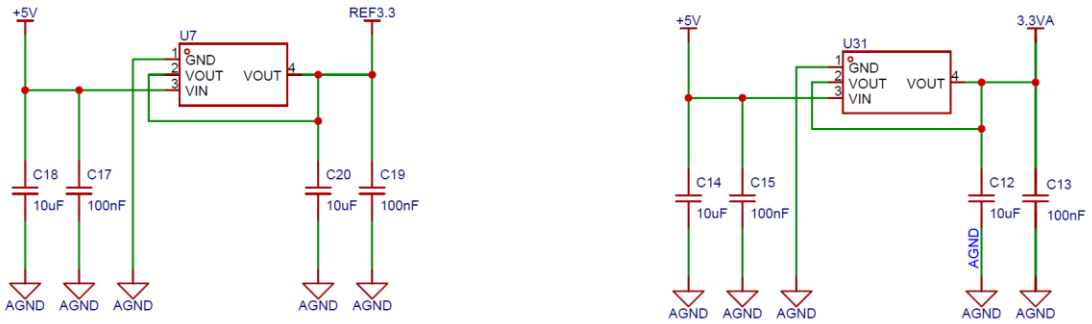


图 2.4.2 5V 降至 3.3V 原理图

设置四组 LED 灯，通过观察 LED 灯亮灭，检测上电、降压是否成功。见图 2.4.3。

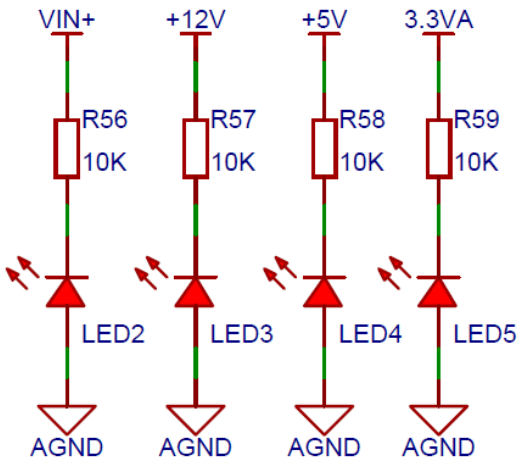


图 2.4.3 LED 上电检测

2.5 STM32F334 主控电路设计

主控部分以 STM32F334C8TX 为主控芯片，预留串口通信接口、CAN 通信接口、程序下载接口。主控芯片通过 CAN 通信与负载端通讯。设计原理图见图 2.5.1。

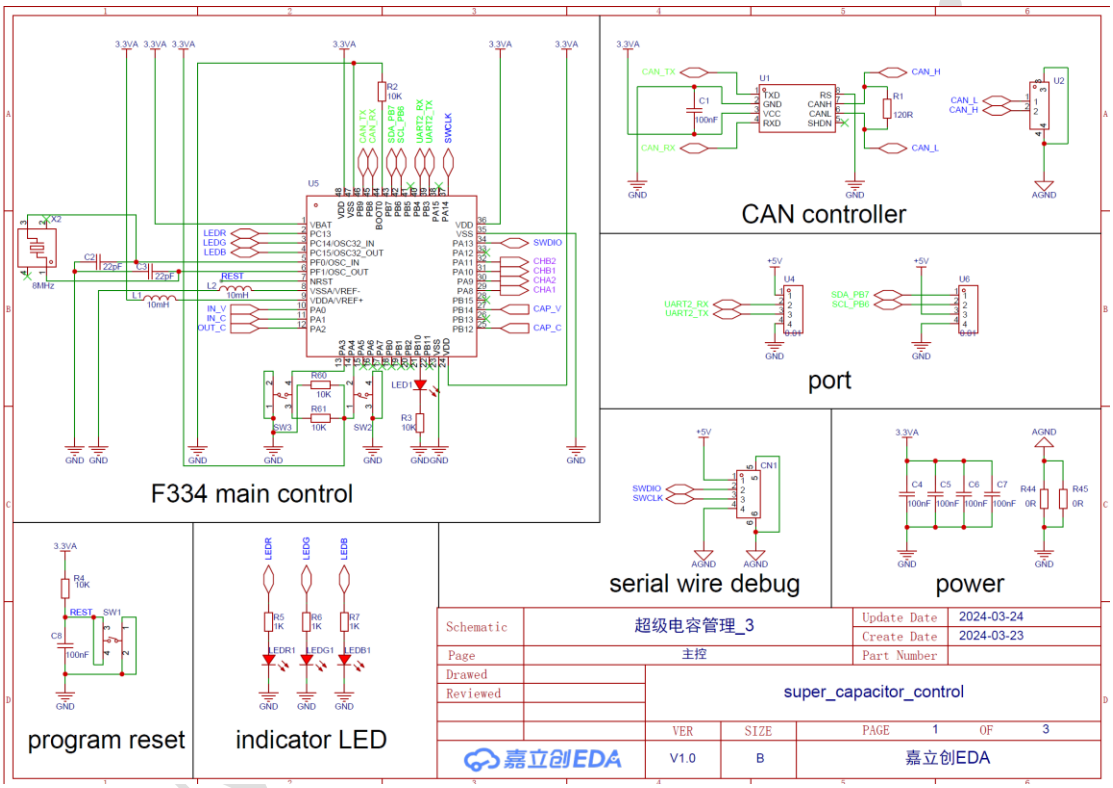


图 2.5.1 STM32F334 主控电路

STM32F334C8TX 引脚分配及功能见表 2.5.1，表 2.5.2，表 2.5.3。

表 2.5.1 ADC 信号对应引脚分配

信号分类	引脚名称	信号	属性
ADC 信号	PB12	CAP_C	电容组电流
	PB14	CAP_V	电容组电压
	PA0	IN_V	电池电压
	PA1	IN_C	电池电流
	PA2	OUT_C	负载电压

表 2.5.2 PWM 信号对应引脚分配

信号分类	引脚名称	信号	属性
PWM 信号	PA8	CHA1	上桥臂驱动信号
	PA9	CHA2	下桥臂驱动信号
	PA10	CHB1	上桥臂驱动信号
	PA11	CHB2	下桥臂驱动信号

表 2.5.3 其他信号对应引脚分配

信号分类	引脚名称	信号	属性
串口通信	PB3	UART2_TX	UART2 发送
	PB4	UART2_RX	UART2 接收
CAN 通信	PB8	CAN_RX	CAN 接收
	PB9	CAN_TX	CAN 发送
SWD 下载	PA13	SWDIO	SWD 调试接口
	PA14	SWCLK	
LED	PC13	LEDR	对应模块工作状态
	PC14	LEDG	
	PC15	LEDB	

3 储能端电路设计

3.1 器件选型

储能端电路器件型号见表 3.1.1。

表 3.1.1 储能端电路器件

器件名称	数值/封装	器件型号	生产商
法拉电容	2.7V 60F	CXHP2R7606R-TW 2.7V60F	CDA(智烽维)
法拉电容保护芯片	SOT-23	BW6101	必威尔科技
MOSFET	SOT-23	IRLML2502	UMW(友台半导体)
厚膜电阻	10Ω	R2010	UNI-ROYAL

3.2 设计说明

储能端电路主要由 9 个 2.7V、60F 的法拉电容串接及其外围保护电路构成，原理图见图 3.2.1。

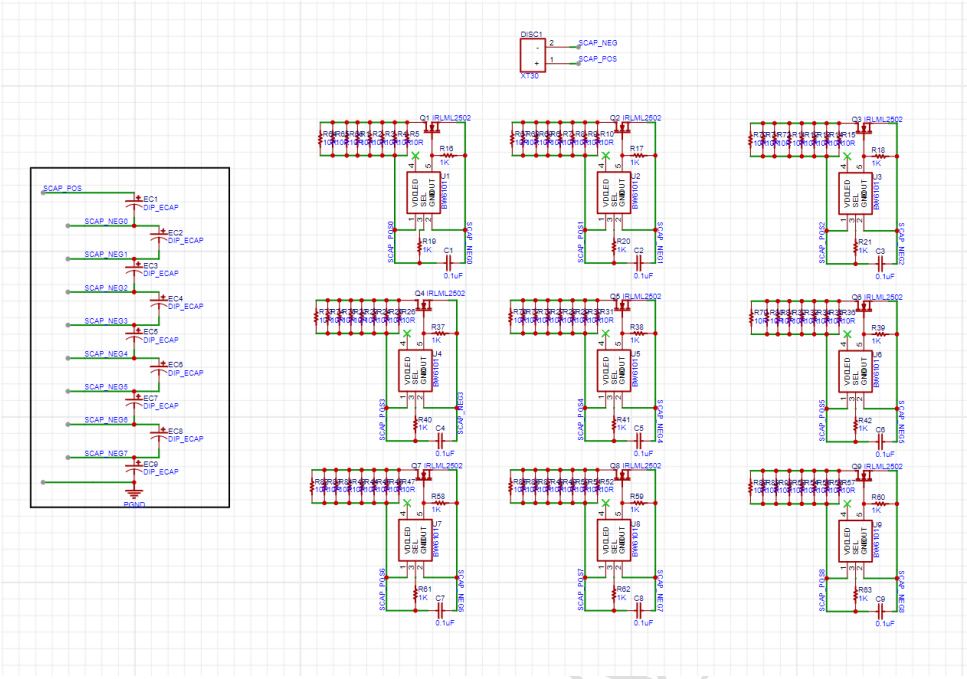


图 3.2.1 超级电容组及其外围电路

储能端最大电压、理论储存能量计算如下：

$$U_{max} = 9 \times 2.7V = 23.6V$$

$$E_{ideal} = 9 \times \frac{1}{2} C U^2 = 1968.3\,J$$

由于 2.7V 60F 法拉电容容量较大，且同时需要大电流进行充放电，因此需要更大功率的泄放电路使得单体电容不过压，进而保护电容组的工作安全。

采用 BW6101、外部扩流 MOSFET、10Ω 泄放电阻，每个单体法拉电容的保护电压为 2.65V，当电容两端的电压大于 2.65V 时，开启内部泄放开关，再通过泄放电阻对下一级电容进行放电，从而保证电容两端电压不会过压。

二、软件设计

1 控制算法综述

整个系统由供电端、功率控制端、储能端、负载端，共四个部分组成。代码设计思路如下：初始化程序和循环执行程序。

初始化程序（1）开启 STM32F334 的 ADC 模式，用于检测供电端的输入电压电流，以及输出给负载的电压电流；（2）由于负载端和功率控制端的主控模块通过 CAN 通信进行板间通讯，为了防止通讯数据帧异常导致程序初始化失败，编写通讯错误检测函数，并且代码将标志位置零；（3）初始化功率控制端相应的设置（即 Supercap_init 函数），并且配置控制板主控的定时器生成 PWM，编写代码使控制板输出 PWM。初始化代码见图 1.1.1。

```
  /* USER CODE BEGIN 2 */
//  HAL_UART_Receive_IT(&huart2, (uint8_t *)&Rx_buff, 1);
  /*开启adcdma===用于ADC测输入输出电压电流*/
  HAL_ADC_Start_DMA(&hadc1,(uint32_t *)adc1_buff,3);
  HAL_ADC_Start_DMA(&hadc2,(uint32_t *)adc2_buff,2);

   uint8_t sendcant=1;
   err_det.start=0;        //错误检测信号，手动置0，说明使得刚开始没有错误信号
   err_det.flag_can=0;     //can错误检测标志 置零
    HAL_Delay(100);

    Supercap_init();//初始化超电 看我函数本体
    HAL_Delay(100);
    //定时器配置
    HAL_HRTIM_WaveformCounterStart(&hhrtim1,HRTIM_TIMERID_TIMER_B|HRTIM_TIMERID_TIMER_A|HRTIM_TIMERID_MASTER);
    HAL_HRTIM_WaveformOutputStart(&hhrtim1,HRTIM_OUTPUT_TB1|HRTIM_OUTPUT_TB2|HRTIM_OUTPUT_TA1|HRTIM_OUTPUT_TA2);
    err_det.condition=1;//板子处于开启状态
```

图 1.1.1 初始化程序

循环执行程序：在功率控制端和负载端通讯正常情况下，执行功率控制函数（即 Supercap_control），通过设置 PWM 占空比来控制 MOSFET 开通关断，从而按需利用超级电容组的能量;同时，功率控制端的主控模块大约每隔 5ms 向

负载端的主控模块发送一次超级电容组电压数据，由负载端主控判断超级电容组电压是否正常，随后进入错误检测函数监测系统的工作状态，错误检测函数中设置了 6 种系统异常条件判断状态，当系统处于异常状态时，功率端主控关闭定时器输出 PWM 信号。循环执行代码见图 1.1.2。

```
while (1)
{
     //Supercap_pwm_updata(vration);
     Supercap_control();    //超电control函数
    /*oled显示*/
    /*
      OLED_Showfloat(60,0,Supercap.input_voltage.windows_filter.output,2,4,4);  //输入电压
      OLED_Showfloat(60,1,Supercap.input_current.windows_filter.output,2,4,4);  //输入电流
      OLED_Showfloat(60,2,Supercap.output_current.windows_filter.output,2,4,4); //输出电流
      OLED_Showfloat(60,3,Supercap.supercap_voltage.windows_filter.output,2,4,4); //电容电压
      OLED_Showfloat(60,4,Supercap.supercap_current.windows_filter.output,2,4,4); //电容电流
      OLED_Showfloat(60,5,Supercap.in_p,2,4,4);  //输入功率
      OLED_Showfloat(60,6,Supercap.out_p,2,4,4); //输出功率
      OLED_Showfloat(60,7,Supercap.cap_p,2,4,4); //电容功率
/
    sendcant++;
    if( sendcant==100)
    {
      //大约5ms发送一次
      supercap_sendmessage(Supercap.cap_v * 1000);
      sendcant=0;
    }
   /* 串口显示*/
   // senddebug();
   //状态检查
  Error_dete();   //错误判断
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
}
```

图 1.1.2 循环执行程序

执行完整的控制程序可以实现 3 种功率控制模式，见表 1.1.1。

表 1.1.1 功率控制模式

	条件	电路状态
1	负载功耗较低	电池全部功率提供给负载
2	负载功耗较低	电池供给负载功率后， 剩余功率给超级电容组充电
3	负载功耗较高，电池功率供给不足	超级电容组放电， 给负载提供能量

2 主要程序说明

2.1 Supercap_init 初始化函数

2.1.1 采样校准算法

由于采样芯片 STM32F334C8TX 和 ADC 检测的误差，构建线性回归函数减小误差。ADC 检测测量的数据名称见表 2.1.1。

表 2.1.1 ADC 检测数据

变量名称	意义
Supercap.input_voltage.linear_x[i]	电池端输入电压
Supercap.input_current.linear_x[i]	电池端输入电流
Supercap.output_current.linear_x[i]	输出至负载电流
Supercap.supercap_voltage.linear_x[i]	超级电容组电压
Supercap.supercap_current.linear_x[i]	超级电容组电流

线性回归模型公式为：

$$Y = aX + b$$

误差参数定义为比例因子 a，和零偏 b，采用两点拟合法计算出五组变量的误差参数。构建误差拟合模型，重新计算出实际值。实际值变量名称见表 2.1.2。

表 2.1.2 实际值变量名称

变量名称	意义
Supercap.input_voltage.linear_y[i]	电池端输入电压
Supercap.input_current.linear_y[i]	电池端输入电流
Supercap.output_current.linear_y[i]	输出至负载电流
Supercap.supercap_voltage.linear_y[i]	超级电容组电压
Supercap.supercap_current.linear_y[i]	超级电容组电流

主要实现代码见图 2.1.1。

```
//最小二乘函数
LinearFitResult linear_fit(float x[], float y[], int n)
{
  float sum_x = 0, sum_y = 0, sum_xy = 0, sum_x2 = 0;
  int i;

  for (i = 0; i < n; i++) {
    sum_x += x[i];
    sum_y += y[i];
    sum_xy += x[i] * y[i];
    sum_x2 += x[i] * x[i];
  }


  LinearFitResult result;
  //斜率
  result.slope = (n * sum_xy - sum_x * sum_y) / (n * sum_x2 - sum_x * sum_x);
  //截距
  result.intercept = (sum_y - result.slope * sum_x) / n;

  return result;
}
```

```
    {2.000, 1.80},
    {2.0431, 2.0},
    {2.1688, 2.60},
    {2.210, 2.80}
};
#endif
//计算截距和斜率
void Calculate_a_b(adc_to_real *p)
{
    p->intercept = linear_fit(&p->linear_data_x[0],&p->linear_data_y[0] , ROWS).intercept;
    p->slope = linear_fit(&p->linear_data_x[0],&p->linear_data_y[0] , ROWS).slope;
}
//计算实际值
void ADC_conversion(adc_to_real *p,int input)
{
    p->windows_filter.input=(double)input/4096*3.3;
    p->measured_value=p->windows_filter.input;
    Window_Filter_Calc(&p->windows_filter);
    //y=kx+b
    p->actual_value=p->windows_filter.output*p->slope  +  p->intercept;
}
```

图 2.1.1 采样校准算法代码

2.1.2 窗口滤波算法

在数字 ADC 采样中，如果直接将读取到的信号参与后续的处理，可能存在脉冲干扰导致逻辑判断错误，因此工程应用中需要使用软件滤波。

本项目中构建滑动窗口滤波器，经过滤波处理后的数据，消除了噪声干扰，数据波动稳定平滑。具体实现代码见图 2.1.2。

滑动窗口滤波器原理如下：

1. 建立采样窗口和滤波窗口，自定义上述窗口的长度。
2. 当数据样本点数未填满采样窗口，对采样窗口内的数据累加做平均值计算。
3. 当数据样本点数已填满采样窗口，进行冒泡排序，去除 n 个最大值和最小值后，再对滤波窗口内剩余的数据累加后做平均值计算。
4. 新的数据样本到来后，先进的数据先出 ，后进的数据后出（FIFO），移除采样窗口中时间最早的数据。重复上述第三点操作。

```
//窗口滤波初始化
//把ADC 读取的电压 当中可能会存在奇怪的信号，若无软件滤波，会导致动态采样时信号曲线不是很平滑，以及存在脉冲干扰导致逻辑判断错误 用这个滤波函数把这些滤
//解释：ADC测的数据很多，这个算法是十个窗口。
//参考知乎【滑动窗口滤波器原理分析及代码讲https://zhuanlan.zhihu.com/p/600867023#%E6%BB%91%E5%8A%A8%E7%AA%97%E5%8F%A3%E6%BB%A4%E6%B3%A2%E5%99%A8%E7%9A
void Window_Filter_Init(window_filter_struct *p) {
    p->ptr = 0;
    p->input = 0;
    p->output = 0;
    for (unsigned char i = 0; i < 10; i++) {
        p->buffer[i] = 0;
    }
}
//窗口滤波计算函数
void Window_Filter_Calc(window_filter_struct *p) {

    //窗口填充

    p->buffer[p->ptr] = p->input;

    float sum = 0;
    //十个窗口求和
    for (unsigned char i = 0; i < 10; i++) {
        sum += p->buffer[i];
    }

    p->output = sum / 10;
    p->ptr++;
    //填满后
    if (p->ptr == 10) p->ptr = 0;
}
```

图 2.1.2 窗口滤波算法代码

2.2 Error_detection 状态自检函数

在系统供电端上电时，需要对所涉及的电流电压采样，供电端、功率控制端、负载端工作状态进行自检，避免在异常状态开始工作对系统造成不可逆的损害。状态自检函数中定义了 6 种系统在工作种可能出现的状态，说明如下。

1. 负载端处于断电状态，功率控制板 PWM 开启，则关闭控制程序使板子 PWM 关闭，PCB 上的红灯、绿灯闪烁提示。代码见图 2.2.1。

```
void Error_dete(void)
{
  //判断底盘和板子状态状态
    if(Supercap.output_state == 0  &&  err_det.condition == 1 )//如果底盘处于断电状态而板子pwm仍开启，则关闭板子pwm
  {
    RED_ON();
    GREEN_ON();
    HAL_HRTIM_WaveformOutputStop(&hhrtim1,HRTIM_OUTPUT_TB1|HRTIM_OUTPUT_TB2|HRTIM_OUTPUT_TA1|HRTIM_OUTPUT_TA2); //关闭pwm输出
    err_det.start=1;//底盘处于断电状态，关闭控制程序
    err_det.condition=0;//板子处于关闭状态
```

图 2.2.1 第一种工作状态代码

2. 负载端处于上电状态，功率控制板 PWM 未开启，则开启控制程序使板子输出 PWM，PCB 上的红灯、黄灯闪烁提示，代码见图 2.2.2。

```
    else  if(Supercap.output_state == 1 &&  err_det.condition == 0 )//如果底盘处于上电状态而板子pwm未开启，则打开板子pwm
  {
    RED_ON () ;
    YELLOW_ON();
    HAL_HRTIM_WaveformOutputStart(&hhrtim1,HRTIM_OUTPUT_TB1|HRTIM_OUTPUT_TB2|HRTIM_OUTPUT_TA1|HRTIM_OUTPUT_TA2);//打开pwm输出
    err_det.start=2;//底盘处于上电状态，打开控制程序
    err_det.condition=1;//板子处于开启状态
  }
```

图 2.2.2 第二种工作状态代码

3. 负载端正常上电状态，功率控制板 PWM 正常开启，如果负载端突然断电，则关闭控制程序使板子 PWM 关闭，PCB 上绿灯、黄灯闪烁提示，代码见图 2.2.3。

```
if(Supercap.output_state == 1 &&  err_det.condition == 1 )//如果底盘处于上电状态且板子pwm开启

//底盘断电后，板子停止输出PWM
    if(Supercap.in_v<20.0f)

   err_det.time_inv++;
   //关闭输出
   if(err_det.time_inv==1000)
   {
      GREEN_ON();
      YELLOW_ON();
      HAL_HRTIM_WaveformOutputStop(&hhrtim1,HRTIM_OUTPUT_TB1|HRTIM_OUTPUT_TB2|HRTIM_OUTPUT_TA1|HRTIM_OUTPUT_TA2); //关闭pwm输出
      err_det.start=5;//关闭控制程序

   }
```

图 2.2.3 第三种工作状态代码

4. 在系统正常工作状态下，检测供电端电压，如果电压大于 40V 或小于 18V，则关闭控制程序使板子 PWM 关闭，PCB 上绿灯、黄灯闪烁提示，代码见图 2.2.4。

```
//判断电压是否正常
if(Supercap.in_v>40||Supercap.in_v<18) //输入电压不正常，控制板，防止击穿mos管
{                                  //电压不正常关闭板子，否则开启板子

      GREEN_ON();
      YELLOW_ON();
      HAL_HRTIM_WaveformOutputStop(&hhrtim1,HRTIM_OUTPUT_TB1|HRTIM_OUTPUT_TB2|HRTIM_OUTPUT_TA1|HRTIM_OUTPUT_TA2);  //关闭输出
      err_det.start=3;//关闭控制程序
      //err_det.condition=0;//板子处于关闭状态

}
else
{
    if(err_det.DCDC_mode==loopc)   { GREEN_ON(); YELLOW_OFF(); }
    if(err_det.DCDC_mode==loopv)   { GREEN_OFF(); YELLOW_ON(); }
    if(err_det.Supercap_mode==charge)          RED_ON();
    if(err_det.Supercap_mode==discharge)       RED_OFF();
}
```

图 2.2.4 第四种工作状态代码

5. 在超级电容组电压大于 12V 的前提下，如果检测到电容组的输入输出电流不正常，计数，当计数到一定值，关闭控制程序使板子 PWM 关闭，PCB 上红灯、绿灯闪烁提示，计数清零，代码见图 2.2.5。

```
//判断电容的电量
if(Supercap.cap_v>=12.0f)   //电容组电压大于12V
{
GPIOB->BSRR =GPIO_PIN_10;
}
//判断输入输出电流是否正常
if(Supercap.out_c>=OUTC_MAX+5||Supercap.in_c>=INC_MAX+5)       //如果输入输出电流不正常
{
     err_det.time_inc++;
     //关闭输出
     if(err_det.time_inc==9000)   //计数器记错，错到一定时间后，板子关闭控制程序
     {
        RED_ON () ;
        GREEN_ON();
        HAL_HRTIM_WaveformOutputStop(&hhrtim1,HRTIM_OUTPUT_TB1|HRTIM_OUTPUT_TB2|HRTIM_OUTPUT_TA1|HRTIM_OUTPUT_TA2);  //关闭输出
        err_det.start=4;//关闭控制程序

     }

}
else
{

     err_det.time_inc=0;  //清零，重新计数
}
```

图 2.2.5 第五种工作状态代码

6. 在功率控制端和负载主控端的 CAN 通信正常开启的前提下，如果功率控制端在 4s 之内没有收到负载端的数据，则关闭输出 PWM，PCB 上红、绿、黄灯闪烁提示；当功率控制端和负载主控端的 CAN 通信异常后恢复正常，开启控制程序，输出 PWM，代码见图 2.2.6。

```
#if  can |
  if(HAL_GetTick()-err_det.gettime_can>=1000)
    {//4秒没有收到can 关闭输出
         err_det.flag_can=1;
         HAL_HRTIM_WaveformOutputStop(&hhrtim1,HRTIM_OUTPUT_TB1|HRTIM_OUTPUT_TB2|HRTIM_OUTPUT_TA1|HRTIM_OUTPUT_TA2);
         RED_ON () ;
         GREEN_ON();
         YELLOW_ON();
         err_det.start=4;
         Supercap.referee_buff=0;
         Supercap.referee_power=0;
  }
    //c板和控制板之间can通信异常后恢复正常，把输入输出重新打开的处理方式
  if(err_det.flag_can==1&&(HAL_GetTick()-err_det.gettime_can)<500)
  {
      HAL_HRTIM_WaveformOutputStart(&hhrtim1,HRTIM_OUTPUT_TB1|HRTIM_OUTPUT_TB2|HRTIM_OUTPUT_TA1|HRTIM_OUTPUT_TA2);
      err_det.flag_can=0;
      err_det.start=0;
   }
  #endif
```

图 2.2.6 第六种工作状态代码

2.3 Supercap_control 控制函数

2.3.1 实际功率计算函数

使用 ADC 测出来的真实电压、电流值，计算供电端输入功率、负载端输出功率，功率控制端需要设置的功率由用户自定义。功率计算代码见图 2.3.1。

```
//2、计算功率
      //IN
      Supercap.in_v=Supercap.input_voltage.actual_value;      //测出来的真实值--数据迁移，从结构体
      Supercap.in_c=Supercap.input_current.actual_value;
      Supercap.in_p=Supercap.in_v*Supercap.in_c;
      //OUT
      Supercap.out_v=Supercap.input_voltage.actual_value;
      Supercap.out_c=Supercap.output_current.actual_value;
      Supercap.out_p=Supercap.out_v*Supercap.out_c;
     //DCDC
      Supercap.dcdc_v=Supercap.input_voltage.actual_value;
      Supercap.dcdc_c=Supercap.in_c-Supercap.out_c;
      Supercap.dcdc_p=Supercap.dcdc_v*Supercap.dcdc_c;
      //CAP
      Supercap.cap_v=Supercap.supercap_voltage.actual_value;
    //  Supercap.cap_c=Supercap.supercap_current.actual_value;
    //  Supercap.cap_p=Supercap.cap_v*Supercap.cap_c;
```

图 2.3.1 功率计算代码

2.3.2 功率设置函数

通过检测负载端的功率情况，灵活调节超级电容组储存的能量。调节代码见图 2.3.2。

```
//3、获取需要设置的功率
 #if mode
  DCDC_control.power_set=Supercap.referee_power; //从裁判系统发过来的chassis功率和buffer能量
  DCDC_control.buffer_power=Supercap.referee_buff;

  if(Supercap.referee_power==0) //底盘功率为0 电池全部给SCAP充电
    {
        DCDC_control.power_set=50;
    }


  if(Supercap.referee_power==0||Supercap.referee_buff==0)//提高容错率
  {
        DCDC_control.buffer_power=0;
  }
  if(Supercap.referee_power>250||Supercap.referee_power<0)//电管给底盘最大输出功率是250W，大于250W电管硬件保护，自动关闭电源
    {
        DCDC_control.power_set=30;  //用于保护官方电池的代码，防止过功率，电流倒灌
    }
```

图 2.3.2 功率设置函数

2.3.3 PID 反馈函数

整个控制闭环由双串级 PID 构成，为内环并行计算的电流环和电压环，以及外环功率环。

PID 控制的基本原理是基于负反馈控制理论，根据输入的偏差值，按照比例、积分、微分的函数关系进行运算，运算结果用以控制输出。PID 控制器是一种基于比例（proportion）、积分（integration）和微分（derivative）的控制理论的数字控制算法，通过调整这三个参数来调节系统的响应，从而使系统具有良好的动态特性和稳态性能。

比例控制（P 控制）根据系统的误差信号来调整控制输出，使输出与误差信号成正比，即误差信号越大，控制输出也越大。它可以快速地对系统进行调节，但存在稳定性差和超调量大的缺点。

积分控制（I 控制）用于消除系统的静态误差，对误差信号进行积分运算，将历史误差的累积值作为控制输出。这样可以消除系统的稳态误差，提高系统的稳定性和精度，但可能导致系统的超调和振荡。

微分控制（D 控制）则通过对误差信号进行微分运算，预测系统未来的变化趋势，从而提前调整控制输出。这有助于抑制系统的振荡和超调，提高动态响应速度，但对噪声信号敏感，可能会引发抖动。

PID 控制的思想是通过结合比例、积分和微分三部分，综合考虑系统的静态特性、动态特性和稳定性，实现对系统的精确控制[27]。通过不断测量系统

的实际输出和期望输出之间的误差，并根据误差的大小来调整控制器的输出，使系统最终达到期望的稳态。PID 控制的核心是误差控制，有误差才能有控制输出。

PID 参数定义见图 2.3.3。

```
//PID_init(pid_type_def *pid, unsigned char mode, const float p, const float i, const float d, float max_out, float max_iout, float min_out, float min_iout);
//增量式pid只用限制输出
//定义4个环的kp ki kd，以便后续以功率为衡量要求，监测底盘所需功率，电容组能量，电池能给的功率，后续一共三种状态要求反馈
PID_init(&Supercap.PID_buffer,1,0.8f,0.01f,1,30,0,20,-20);   //缓冲能量环
PID_init(&Supercap.PID_power,1,0.3f,0.035f,3.0f,400,25,-500,-25);   //功率环
PID_init(&Supercap.PID_loopv,1,1.3f,0.06f,3,100,25,-200,-25);   //电压环
PID_init(&Supercap.PID_loopc,1,0.15f,0.025f,0.11f,150,25,-150,-25);       //电流环
```

图 2.3.3 PID 参数

负载端由于应用的情景不同，有不同的功率消耗，因此供电端，电容组需要给负载端提供的功率是灵活的。原则上，由电池给负载端供电；当负载需要大功率输出且电池供给能量不足时，电容组作为补充为负载提供功率；如果负载只损耗由电池提供的较少的功率，则多余的功率给电容组充电。

电容组充电时电流保持不变，先用 PID 电流环，电压线性缓慢上升；充满电后，使用 PID 电压环，使电容组电压保持在满电压状态；当电容组需要放电，为负载端提供功率时，使用 PID 电流环恒流放电。代码见图 2.3.4。

```
//充电模式
//限制充电电流
if(DCDC_control.dcdc_curr>4)  DCDC_control.dcdc_curr=4;

err_det.Supercap_mode=charge;
//调试电压--输入输出电压比
DCDC_control.vloop_ratio= PID_calc(&Supercap.PID_loopv,Supercap.cap_v,23.00)/ Supercap.dcdc_v;
//调试电流 --输入输出电流比
DCDC_control.cloop_ratio= PID_calc(&Supercap.PID_loopc,Supercap.dcdc_c,DCDC_control.dcdc_curr)/Supercap.dcdc_v;
     ////充电的时候电流保持不变 电压慢慢上升 电流变化较小 所以先用电流环 当他充满了之后 则用电压是保持其电压不变
    if(DCDC_control.cloop_ratio<=DCDC_control.vloop_ratio)//电流环 恒流充电 电压线性缓慢上升
     {
       err_det.DCDC_mode=loopc;
       Supercap_pwm_updata(DCDC_control.cloop_ratio);
     }
     else//电压环 充满电后，保持电压在满电电压
     {
       err_det.DCDC_mode=loopv;
       Supercap_pwm_updata(DCDC_control.vloop_ratio);
     }
```

图 2.3.4 PID 充电模式

使用 PID 电流环恒流放电，代码见图 2.3.5。

```
if(DCDC_control.dcdc_power<0)//放电模式，只用计算电流环
{
  //放电模式 放电，亮个灯
   HAL_GPIO_WritePin(GPIOB, GPIO_PIN_10, GPIO_PIN_RESET);
  //限制放电电流 恒流放电即可
  if(DCDC_control.dcdc_curr<-8)
  DCDC_control.dcdc_curr=-8;
  err_det.Supercap_mode=discharge;
//  DCDC_control.vloop_ratio= PID_calc(&Supercap.PID_loopv,Supercap.cap_v,CAPV_MAX)/ Supercap.dcdc_v;
  DCDC_control.cloop_ratio=PID_calc(&Supercap.PID_loopc,Supercap.dcdc_c,DCDC_control.dcdc_curr)/Supercap.dcdc_v;
  err_det.DCDC_mode=loopc;
  Supercap_pwm_updata(DCDC_control.cloop_ratio);
}
```

图 2.3.5 PID 放电模式

同时，计算调试电压的输入输出电压比、调试电流的输入输出电流。代码见图 2.3.6。

```
//调试电压--输入输出电压比
DCDC_control.vloop_ratio= PID_calc(&Supercap.PID_loopv,Supercap.cap_v,23.00)/ Supercap.dcdc_v;
//调试电流 --输入输出电流比
DCDC_control.cloop_ratio= PID_calc(&Supercap.PID_loopc,Supercap.dcdc_c,DCDC_control.dcdc_curr)/Supercap.dcdc_v;
```

图 2.3.6 PID 调试电压相关参数

PID 初始化和计算代码见图 2.3.7，图 2.3.8。

```
void PID_init(pid_type_def *pid, unsigned char mode, const float p, const float i, const float d, float max_out, float max_iout,
              float min_out, float min_iout) {
    if (pid == 0) {
        return;
    }
    pid->mode = mode;
    pid->Kp = p;
    pid->Ki = i;
    pid->Kd = d;
    pid->max_out = max_out;
    pid->max_iout = max_iout;
    pid->min_out = min_out;
    pid->min_iout = min_iout;
    pid->Dbuf[0] = pid->Dbuf[1] = pid->Dbuf[2] = 0.0f;
    pid->error[0] = pid->error[1] = pid->error[2] = pid->Pout = pid->Iout = pid->Dout = pid->out = 0.0f;
}
```

图 2.3.7 PID 初始化

```
float PID_calc(pid_type_def *pid, float ref, float set) {
    if (pid == 0) {
        return 0.0f;
    }

    pid->error[2] = pid->error[1];
    pid->error[1] = pid->error[0];
    pid->set = set;
    pid->fdb = ref;
    pid->error[0] = set - ref;
    if (pid->mode == PID_POSITION) {
        pid->Pout = pid->Kp * pid->error[0];
        pid->Iout += pid->Ki * pid->error[0];
        pid->Dbuf[2] = pid->Dbuf[1];
        pid->Dbuf[1] = pid->Dbuf[0];
        pid->Dbuf[0] = (pid->error[0] - pid->error[1]);
        pid->Dout = pid->Kd * pid->Dbuf[0];
        LimitMax(pid->Iout, pid->max_iout);
        pid->out = pid->Pout + pid->Iout + pid->Dout;
        LimitMax(pid->out, pid->max_out);
    } else if (pid->mode == PID_DELTA) {
        pid->Pout = pid->Kp * (pid->error[0] - pid->error[1]);
        pid->Iout = pid->Ki * pid->error[0];
        pid->Dbuf[2] = pid->Dbuf[1];
        pid->Dbuf[1] = pid->Dbuf[0];
        pid->Dbuf[0] = (pid->error[0] - 2.0f * pid->error[1] + pid->error[2]);
        pid->Dout = pid->Kd * pid->Dbuf[0];
        pid->out += pid->Pout + pid->Iout + pid->Dout;
        LimitMax(pid->out, pid->max_out);
    }
    return pid->out;
}
```

图 2.3.8 PID 参数计算

2.3.4 Supercap_pwm_updata 函数

由于 MOSFET 采用自举电容驱动方式，上管不能 100%占空比导通，则设置为 95%占空比导通。降压工作时，BOOST 的半桥的占空比为固定 95%，通过改变 BUCK 半桥的占空比来实现稳压；升压或等压工作时，BUCK 的半桥的占空比为固定 95%，通过改变 BOOST 半桥占空比实现稳压。

代码中主要定义的参数和取值见表 2.1.3。

表 2.1.3 主要定义的参数和取值

变量名称	意义
shift_duty 0.96f	最大占空比 96%
shift_duty_2	最大占空比*2
MAX_PWM_CMP	PWM 最大比较值
MIN_PWM_CMP	PWM 最小比较值
DP_PWM_PER 1440	定时器 HRTIM 周期设置为 1440
max_volt_ratio 1.2f	最大电压比较值 1.2
min_volt_ratio 0.1f	最小电压比较值 0.1
CompareValue	比较值

电路 BUCK-BOOST 两种状态的切换由比较值（CompareValue）决定。当实际输入电压大于输出电压时，供电端应向超级电容组充电，求出此时电路中高电平持续时间；同理，当实际输入电压小于输出电压时，超级电容组应放电给负载端供给能量，求出此时电路中高电平持续时间。具体代码见图 2.3.8。

```
if (V_ratio > 1.0f)
    CompareValue = DP_PWM_PER * (shift_duty_2 - shift_duty / V_ratio);
else
    CompareValue = shift_duty * V_ratio * DP_PWM_PER;
```

图 2.3.8 电路中高电平持续时间计算

通过比较“CompareValue“ 和“shift_duty * DP_PWM_PER”数值的大小，判断 DC-DC 电路应工作在 BUCK 模式，还是 BOOST 模式，（shift_duty * DP_PWM_PER 的意义是，用设定的最大占空比乘以定时器周期，即求得设定的 PWM 高电平持续时间）。

当 CompareValue > shift_duty * DP_PWM_PER 时，DC-DC 电路工作在 BOOST 模式下，供电端给超级电容组充电，分别计算对应 BOOST 半桥占空比和 BUCK 半桥占空比。

当 CompareValue < shift_duty * DP_PWM_PER 时,DC-DC 电路工作在 BUCK 模式下，超级电容组放电，给负载端供给能量，分别计算对应 BOOST 半桥占空比和 BUCK 半桥占空比。

具体实现代码见图 2.3.9。

```
//计算boost半桥占空比
if ((float) CompareValue > shift_duty * DP_PWM_PER)
    //电路在BOOST模式下，BOOST半桥占空比，Dboost = D总 - Dbuck
    boost_duty = shift_duty_2 * DP_PWM_PER - (float) CompareValue;
else
    //电路在BUCK模式下，BOOST半桥占空比，固定占空比
    boost_duty = shift_duty * DP_PWM_PER;
////////////////////////////////////////////////////////////////////////
//计算buck半桥占空比
if ((float) CompareValue > shift_duty * DP_PWM_PER)
  //电路在BOOST模式下，BUCK半桥占空比，固定占空比
    buck_duty = shift_duty * DP_PWM_PER;
else
  //电路在BUCK模式下，BUCK半桥占空比，计算
    buck_duty = CompareValue;
```

图 2.3.9 BUCK-BOOST 半桥占空比计算

为了防止软件计算出的电压值不符合实际情况，对 BUCK 半桥和 BOOST 半桥的占空比进行限制，见图 2.3.10。

```
    //占空比限制
if (boost_duty > MAX_PWM_CMP)
    boost_duty = MAX_PWM_CMP;
if (boost_duty < MIN_PWM_CMP)
    boost_duty = MIN_PWM_CMP;
if (buck_duty > MAX_PWM_CMP)
    buck_duty = MAX_PWM_CMP;
if (buck_duty < MIN_PWM_CMP)
    buck_duty = MIN_PWM_CMP;
   //更新占空比
   //解释：boost为1300，buck为1300
```

图 2.3.10 半桥占空比限制

2.4 supercap_sendmessage 通信函数

超级电容组应避免过度放电，如果电容组过度放电，可能会损害电容内部结构导致性能下降，降低电容组寿命。

因此，功率控制端主控芯片通过 CAN 通信，每隔大约 5ms 将超级电容组的电压发送给负载端主控芯片。设定超级电容组电压最低值为 12V，当负载端主控检测到超级电容组电压低于 12V 时，负载端与超级电容组断开连接，电容组停止放电。通信代码见图 2.4.1。

```
//发送超级电容电压给c板
void supercap_sendmessage(uint16_t Vcap)
{

    TxlMessage.StdId=0x211;
    TxlMessage.IDE =CAN_ID_STD;
    TxlMessage.RTR=CAN_RTR_DATA;
    TxlMessage.DLC =0x08;
    memcpy (supercap_tx_data, &Vcap, sizeof(Vcap));
    supercap_tx_data[2]=err_det.start;
    HAL_CAN_AddTxMessage(&hcan, &TxlMessage, supercap_tx_data ,&pTxMailbox);
}
```

图 2.4.1 控制与负载通信代码

第三部分：创新点

1 整体说明

本方案基于四开关 Buck-Boost 电路，设计了一款可以智能调节功率的控制系统。整个系统由供电端、功率控制端、储能端、负载端,共四个部分组成。

项目研发的创新点主要围绕四点展开，分别是功率控制、储能、通信、监测，以下是详细说明。

2 硬件创新点

2.1 高效能的功率控制模块

四开关 Buck-Boost 变换器，利用开关电路的开关周期性地将输入电源与输出负载连接或断开，通过调节开关状态和开关周期，控制输出电压的大小。

由于本方案中，通过设置合适的 PWM 占空比，使主拓扑工作在电感电流连续状态（CCM: Continuous Conduction Mode），而非电感电流“伪断续”状态（DCM：Discontinuous Conduction Mode）。

CCM，开关电源连续导通模式，在此模式下，功率电感电流在整个开关周期内不会降至零，能够实现较高的效率，产生较低的输出电压纹波，广泛应用于需要恒定输出电压、电流的应用中。

DCM，开关电源不连续导通模式，在此模式下，功率电感电流在开关周期内会下降到零，电感“复位”，在低功率应用下具有较高的效率，但产生较大的

输出电压纹波，通常应用于反激式转换器、推挽式转换器等类型的开关电源中。

2.2 先进的储能系统

储能端由九个 2.7V 60F 的法拉电容串联，及其外围电路构成。

超级电容组较普通的电池而言，能够在更短的时间内提供更大的电流，实现快速充放电，从而满足高功率的需求；并且，其充放电循环次数远高于普通电池，在长期使用中具有更高的可靠性和稳定性，从而降低维护成本和更换频率。

考虑本方案中需要高功率脉冲、快速充放电、安全可靠性，采用超级电容组作为储能端。

2.3 可靠的通信模块

由于超级电容组不能过度放电，否则会对电容组造成不可逆的损害。于是，在功率控制端设计 CAN 通信电路，其主控芯片通过 CAN 通信，将超级电容组电压数据发送给负载端主控，由负载端主控判断超级电容组的电压是否低于限定电压，低于则断开负载与超级电容组的连接。

2.4 智能监测与保护系统

根据采样电路，设计 Error_detection 状态自检函数，对供电端、功率控制端、负载端工作状态进行自检，避免在异常状态开始工作对系统造成不可逆的

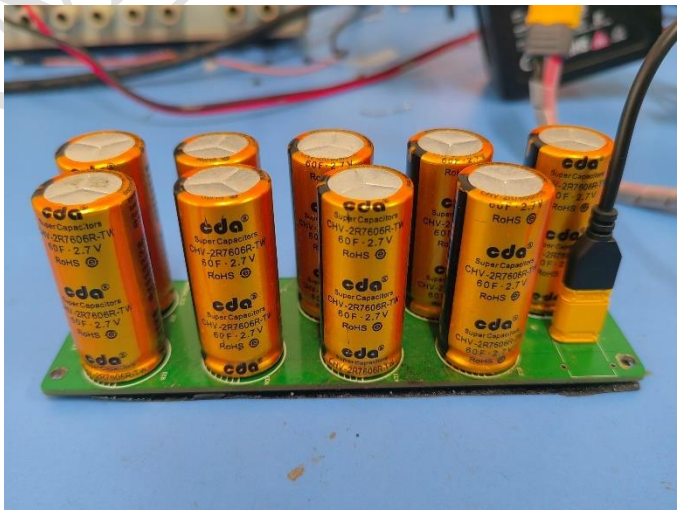
损害。结合供电端、功率控制端、负载端的工作状态，共编写了六种在系统工作时异常状态的处理方案代码，对系统中各部分电压、电流数据持续监测，一旦出现工作异常，将会立即执行对应异常状态的处理方案，从而保护各个模块不受损害。

第四部分：成果展示

基于四开关 BUCK-BOOST 智能功率调节控制系统成果展示，分为实物展示，和使用视频展示，展示内容见表 4.1，图片展示见下图，视频展示见附件压缩包。

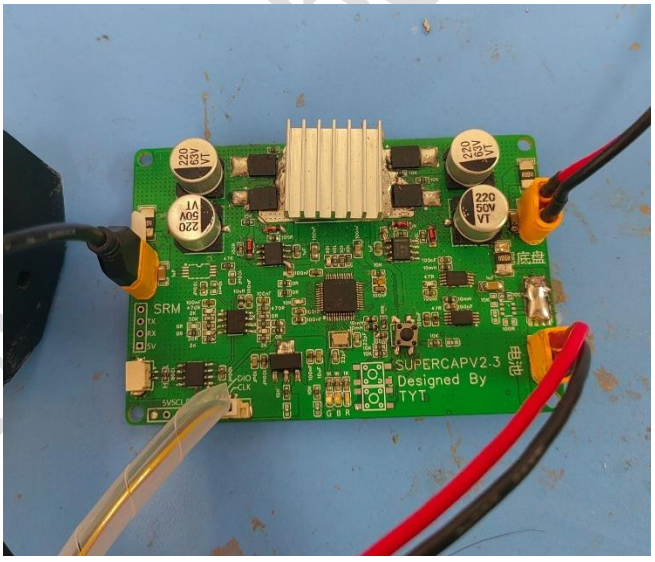
表 4.1 附件展示清单

序号	附件编号	展示内容
1	附件 1	储能端：超级电容组
2	附件 2	供电端：TB48 电池
3	附件 3	功率控制端：主控板
4	附件 4	负载端：多模式可调节负载
5	附件 5	智能功率调节控制系统
6	视频 1	智能功率调节控制系统使用示范

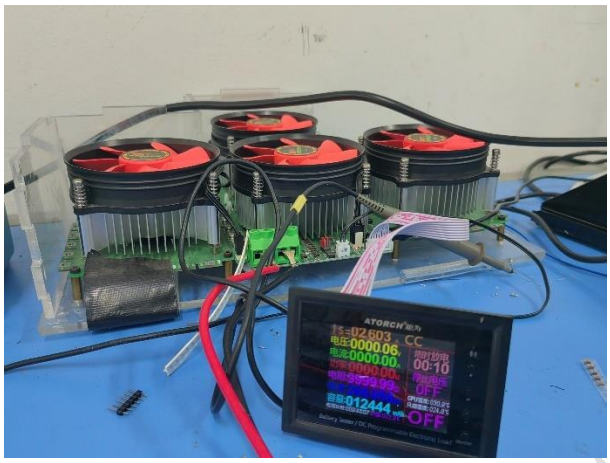


附件 1 超级电容组

附件 2 TB48 电池



附件 3 功率控制主控板



附件 4 多模式可调节负载

附件 5 智能功率调节控制系统

第五部分：团队介绍

2022-2024 赛季上海大学 SRM 战队使用的超级电容控制系统，由 2023 赛季硬件组组长唐源涛研发，2024 赛季硬件组组长胡时宇改进，2024 赛季硬件组成员柳怡冰、吕翰琪、吴昊泽完成报告撰写。

唐源涛来自上海大学机电工程与自动化学院，自动化专业，为上海大学 SRM 战队 2023 赛季硬件组组长。

胡时宇来自上海大学机电工程与自动化学院，机器人工程专业，为上海大学 SRM 战队 2024 赛季硬件组组长。

柳怡冰来自上海大学机电工程与自动化学院，电气工程及其自动化专业，上海大学 SRM 战队 2024 赛季副队长、硬件组成员。

吕翰琪来自上海大学通信与信息工程学院，通信工程专业，上海大学 SRM 战队 2024 赛季硬件组组成员。

吴昊泽来自上海大学机电工程与自动化学院，自动化专业，上海大学 SRM 战队 2024 赛季硬件组组成员。

李璟怿来自上海大学机电工程与自动化学院，电气工程及其自动化专业，上海大学 SRM 战队 2023 赛季工程电控、2024 赛季硬件组成员。

一个项目的研发离不开团队的努力与配合，“山重水复疑无路，柳暗花明又一村”，我们不仅是一起合作的队友，更是相互鼓励、相互支持的朋友，学无止境，we will never walk alone!