武漢紡織大學
WUHAN TEXTILE UNIVERSITY

电力电子综合设计报告书

设计题目：基于单片机的开关电源设计与制作

姓　　名：叶盛廷

班　　级：电气 12201

学　　号：2203240107

指导教师：韩谷静

成绩评定：

2025 年 6 月 20 日

目录

第1章 设计任务与要求

1.1 设计任务

1. 题目：基于单片机的开关电源设计与制作

1）设计并制作如图 1 所示的直流开关电源。

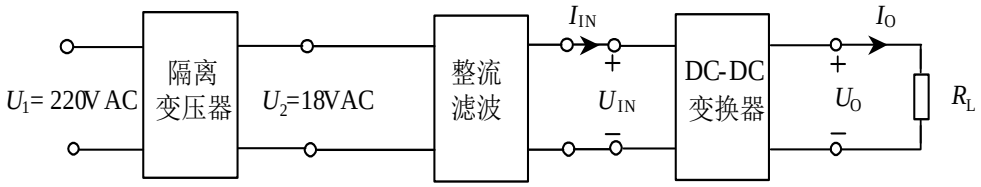


图 1 直流开关电源框图

2）设计基本要求

（1）输出电压 U_O：35-40VDC（按 36VDC 额定电压设计）。

（2）最大输出电流 I_{Omax}：2A（实验室提供 100Ω 滑动变阻器）。

（3）具有电路保护功能（通过 AD 模块采样软件保护）。

3）设计加分要求

（1）I_O 从 0 变到 1A 时，输出电压 U_O 波动不超过±10%。

（2）电源效率 $\eta \geq 80\%$（U_2=18VAC,U_O=36VDC, I_O=1A）。

1.2 设计要求

2. 基本要求

1）采用单片机进行开环 PWM 升压控制。

2）根据题目要求进行方案论证，理论分析与计算，可重新或修改参考设计。

3）用 Matlab 等软件进行主电路原理仿真（参考：电气 22 级电力电子综合设计 Matlab 参考仿真模型 word 版）。

4）原理图及 PCB 采用立创 EDA 软件进行设计，然后申请免费打板，PCB 上至少需丝印“武汉纺织大学+电气 1220x 班+“本人姓名”。

5）电源电路焊接、调试，并达到参数要求。

6）撰写设计报告。

第 2 章 系统硬件设计

2.1 电源板设计

2.1.1 Boost 电路设计

本次设计要求需要设计的电源板中升压斩波电路(boost)，采样电路和栅极驱动电路。

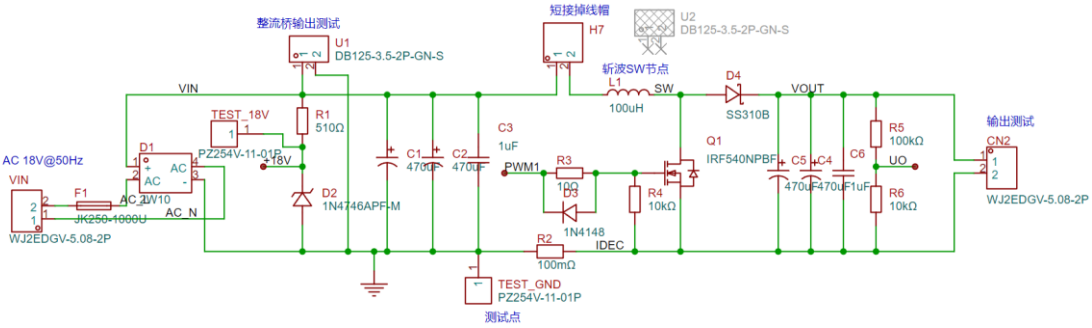


图 1 Boost 电路设计

输入 18V 的交流电后经过整流桥整流，然后经过典型的升压斩波电路拓扑进行升压，并配备有电流采样电阻，滤波电容以及各种测试点。

2.1.2 采样电路设计

采样电路首先经过一个 ADC 采样前级的电压跟随器（Buffer），使用了一个运算放大器 LM358，输入信号首先经过了一个$10k\Omega$的限流电阻，它同时可以抑制输入突变对后级的影响，与电容形成 RC 低通滤波器。经过计算，可以抑制输入噪声

$$f_c = \frac{1}{2\pi RC} \approx 159.155Hz$$

的噪声。并且被配有一个快速二极管，用于钳位，当电压超过 5V 的时候保护信号输入端。随后信号到达运放，运放在这里配置为电压跟随器，输出电压紧跟随输入电压，提供低输出阻抗。

前级具体功能汇总由下表给出

表格 1 前级设计

原因	描述
降低信号源阻抗	ADC 输入通常具有较高的输入阻抗和较低的采样电容。如果信号源阻抗太高，会导致 ADC 采样电容无法及时充电，影响精度。电压跟随器具有极高的输入阻抗和极低的输出阻抗，能驱动 ADC 而不引起压降或响应延迟。
隔离功能	运放起到隔离作用，避免 ADC 的采样瞬间拉低信号源电压，防止干扰信号源稳定性。
稳定电压采样	运放输出能力强，能快速跟随输入变化并驱动 ADC，因此提高采样精度和响应速度。
简单滤波配合	前级 RC 滤波器（R7+C7）配合电压跟随器，可有效滤除高频干扰，而不会因电阻导致压降。
过压保护（D5）	当 UO 意外高于+5V（如电感反冲或外部错误信号），D5 导通将电压钳位保护在+5V 附近，保护 LM358 和 ADC 输入。

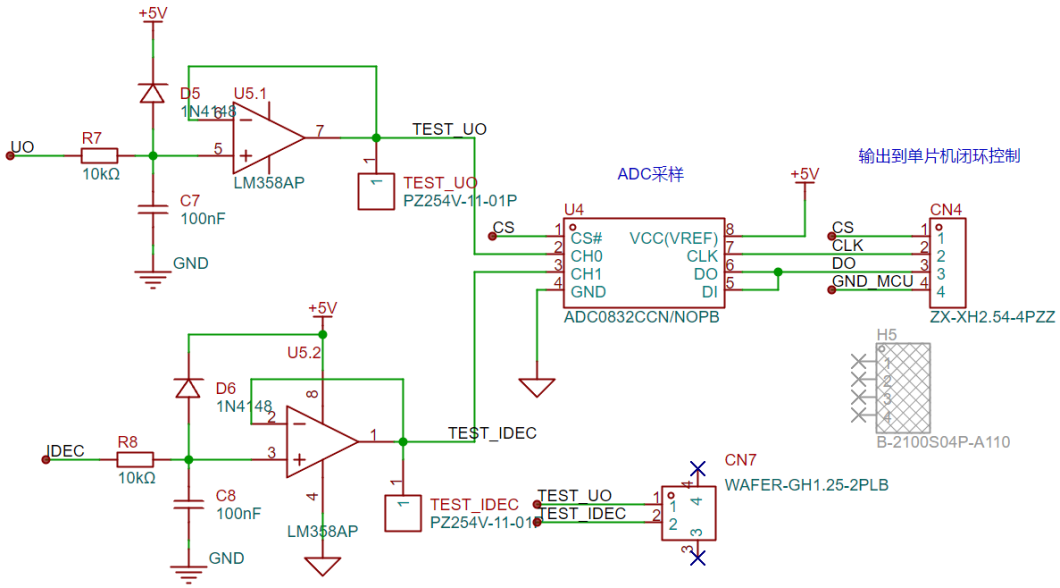


图 2 ADC 采样及其前级

由于 MCU 的 I/O 口推挽能力不够，因此本次采样一颗带有高低侧驱动的 IR2104 栅极驱动

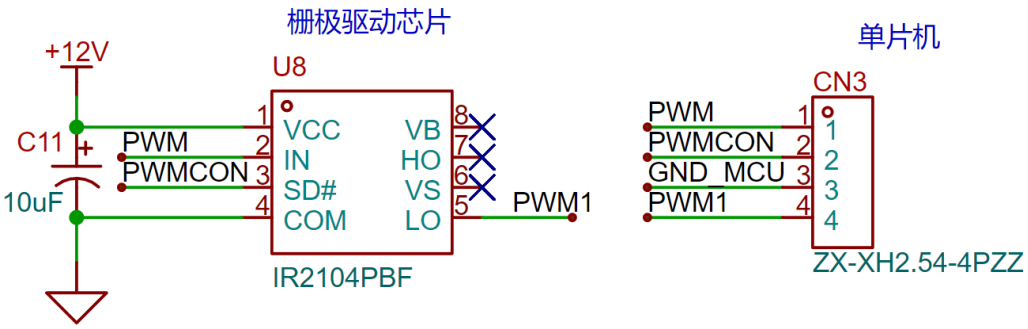


图 3 栅极驱动

该栅极驱动具有 UVLO 功能，因此在电感的 DCM 区会导致 PWM 占空比陡然减小，该现象将在第四章做详细说明，该栅极驱动的 t_{on/off}(typ.)为 680ns 和 150ns，加上死区时间 520ns，预期的 PWM 频率上限可以设定为

$$f = \frac{1}{t_{on} + t_{off} + t_{dead}} \approx 740.74\ kHz$$

为了流出足够的裕度以及根据实际波形质量测试结果最终采用 100kHz 的 PWM 频率。

2.1.3 接地与隔离

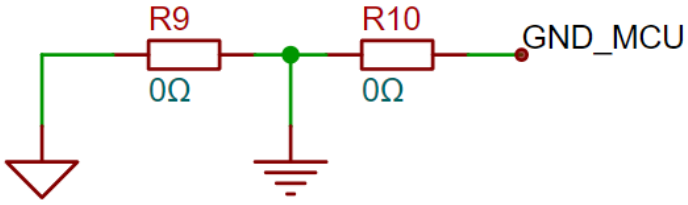


图 4 单端接地

采用 0Ohm 的电阻对功率地和数字地进行单端桥架，以减小回流路径，将高频噪声，高电流回流的地电流和精密信号回流路径物理隔离，用于改善 EMC 性能。

有的设计此处采用磁珠进行隔离，可以提供更优秀的高频阻抗，抑制高频噪声，改善 EMI，此处因为成本以及方便调试采用 0Ohm 进行单端接地。

2.2 控制板设计

本次设计由于涉及 ADC 采样，各种硬件防护以及闭环控制等，因此需要配备有一个控制单元，采用 MCU 作为控制，具体型号为 STM32F411CEU6，以下为具体的详细设计:

2.2.1 电源设计

Boost 端为 18V 整流，首先需要将电压等级降至 MCU 系统常用的 5V 供电，由于压差较大，因此采用两级 LDO 降压，以防止压差过大造成的发热问题

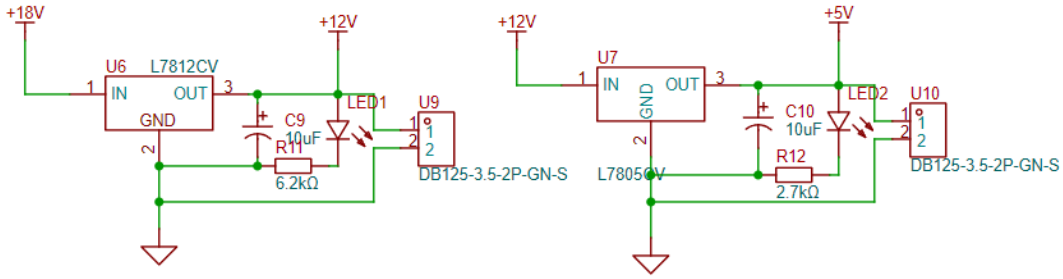


图 5 两级 LDO 降压

随后 5V 系统输入控制板，经过 TLV76733 单元进行欠压保护，过温/过流保护，并且考虑上位由整流桥整流输入可能具有较大的涌流采用一颗 TVS 管防止涌流输入，为 MCU 系统提供稳定的电平。

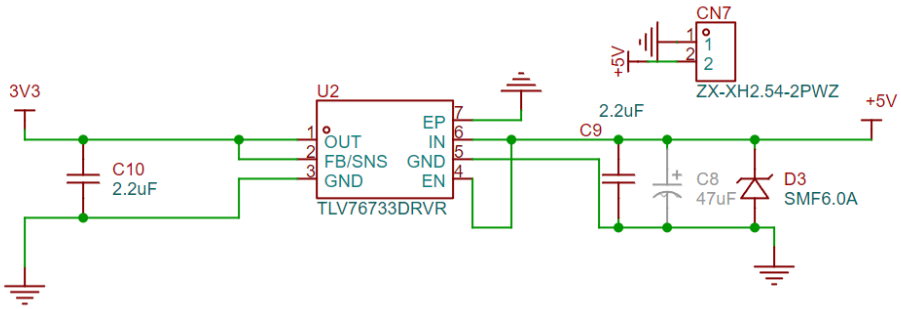


图 6 MCU 供电

2.2.2 复位与时钟

采用 25MHz 无源晶振，拨码开关用于配置 BOOT 模式

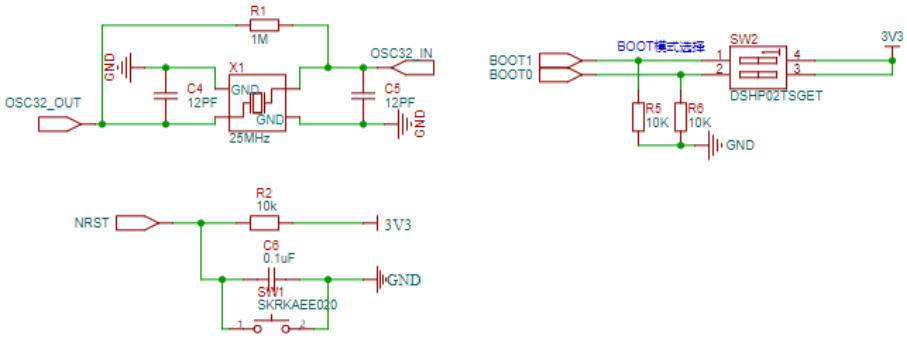
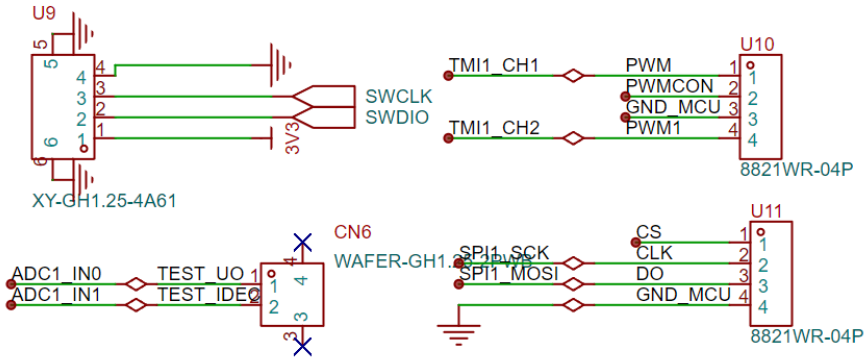


图 7 复位与时钟电路

2.2.3 外部接口

该部分和电源板的接口一一对应，使用 XH 和 GH 系列的冷压端子进行压接接线，并灌入 UV 胶增加稳定性，接口是 ADC 采样以及通讯可靠性的前提。



2.2.4 MCU 系统设计

将上述的电源输入进来，以及时钟.复位电路，连接上外部接口即可完成 MCU 的系统设计。

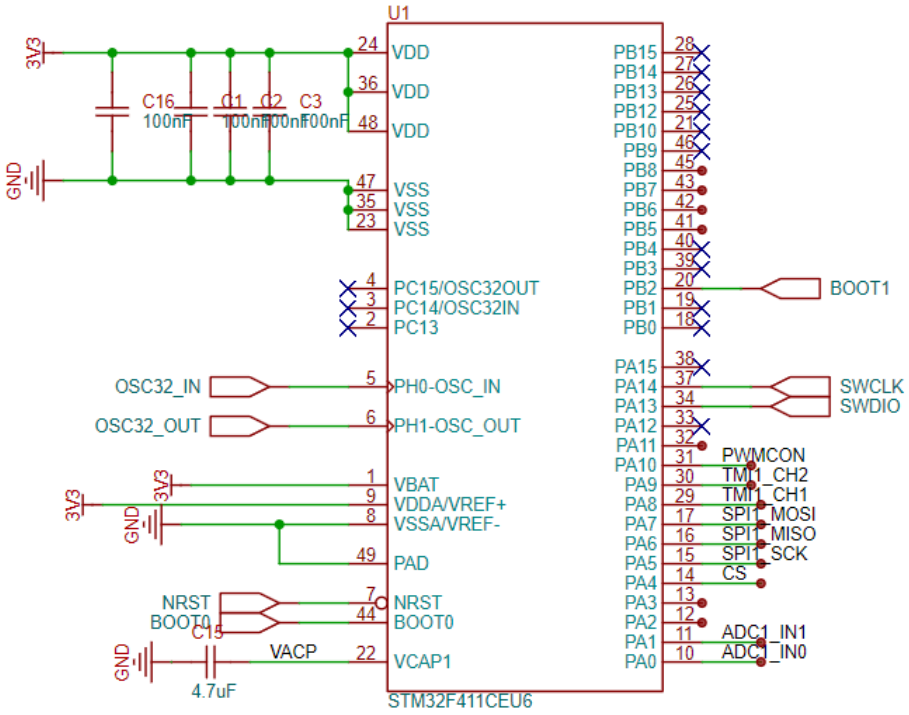


图 8 MCU 系统设计

2.3 PCB-Layout

2.3.1 电源板布局

电源板 Boost 部分采用 U 型布局，栅极驱动尽量靠近 SW 节点，并且 SW 节点回路尽量短，数字部分注意和功率部分分开布局，减小功率部分的各种串扰，功率部分注意表面开窗并走有厚锡，以增大载流能力以及更好的散热效果。

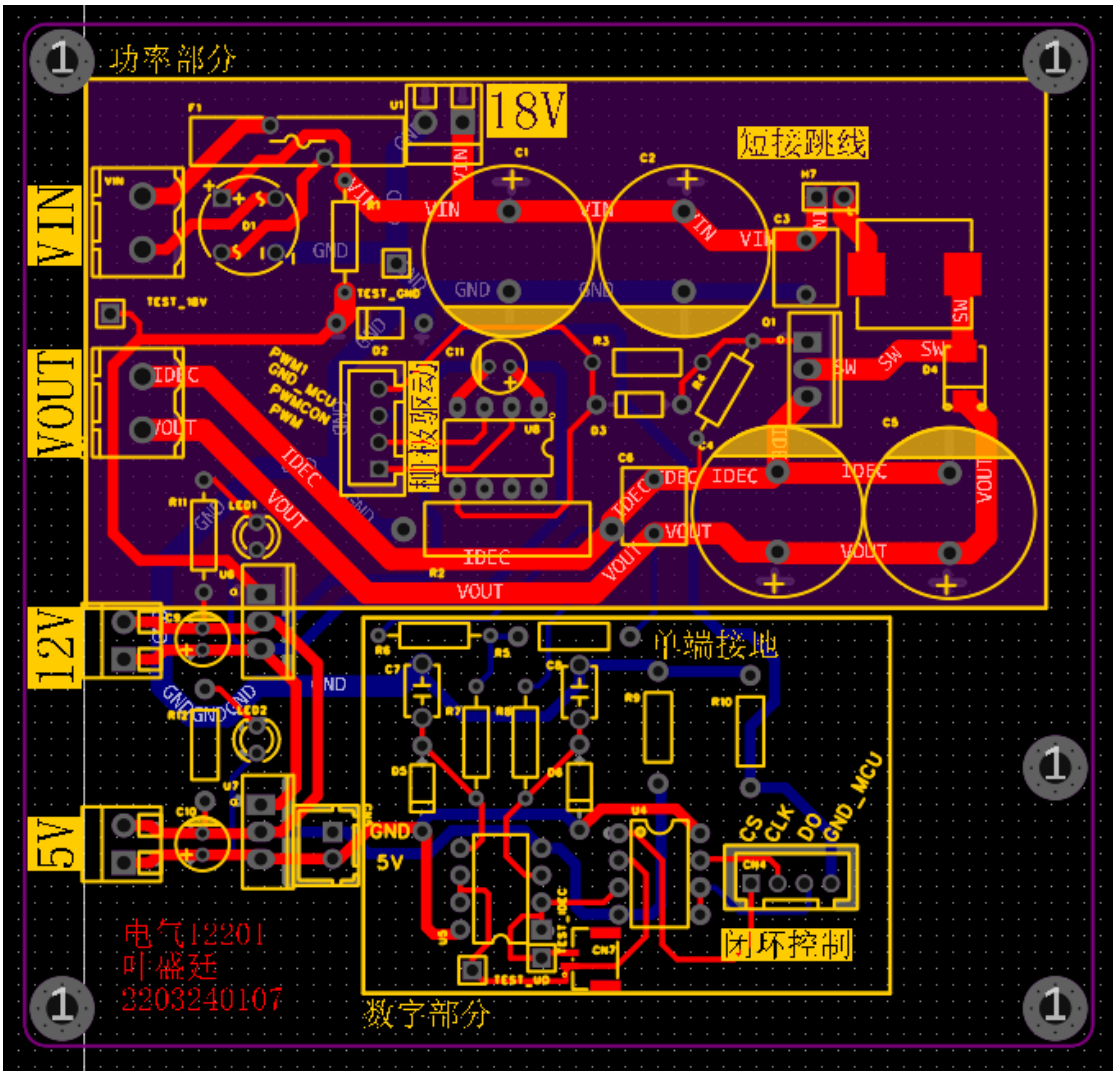

功率部分
18V
短接跳线
VIN
VOUT
12V
5V
驱动模块
单端接地
GND
5V
CS
CLK
DO
GND_MCU
闭环控制
数字部分
电气12201
叶盛廷
2203240107

图 9 电源板布局

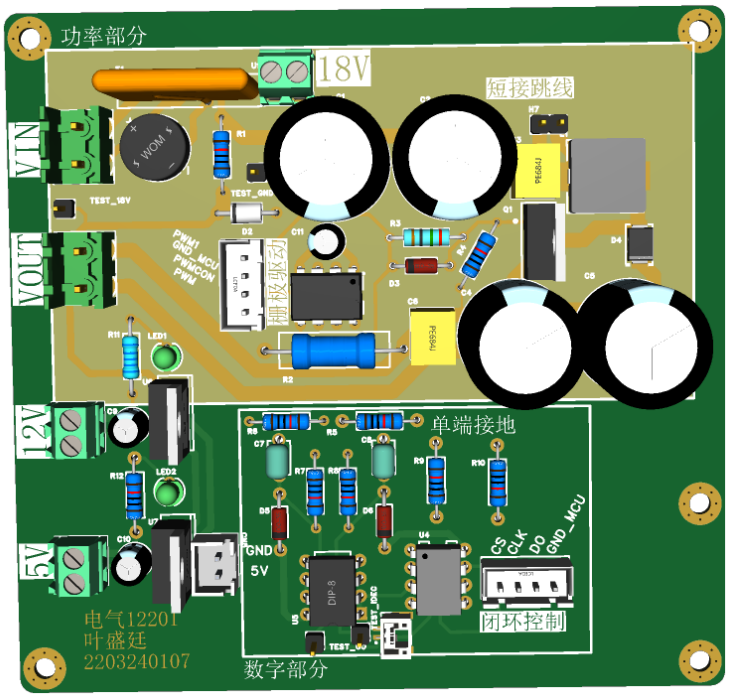


图 10 仿真图

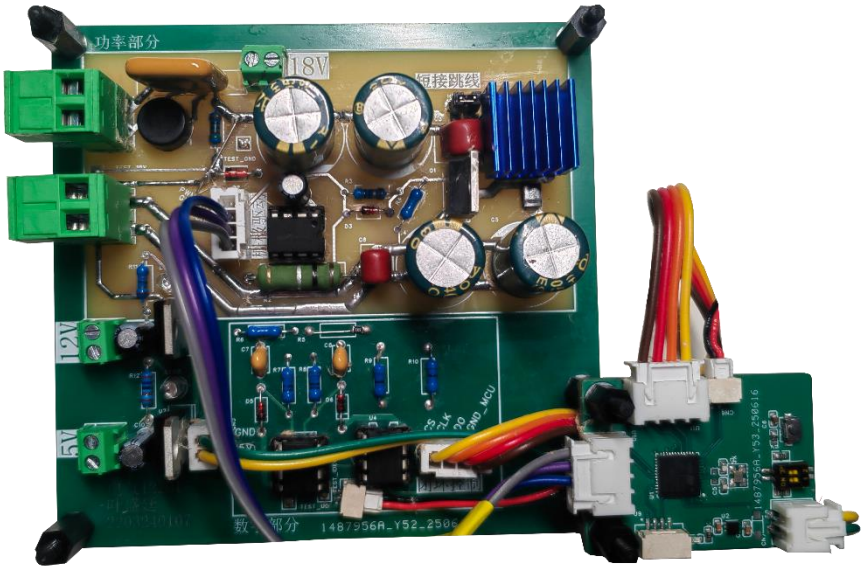


图 11 实物图

2.3.2 控制板布局

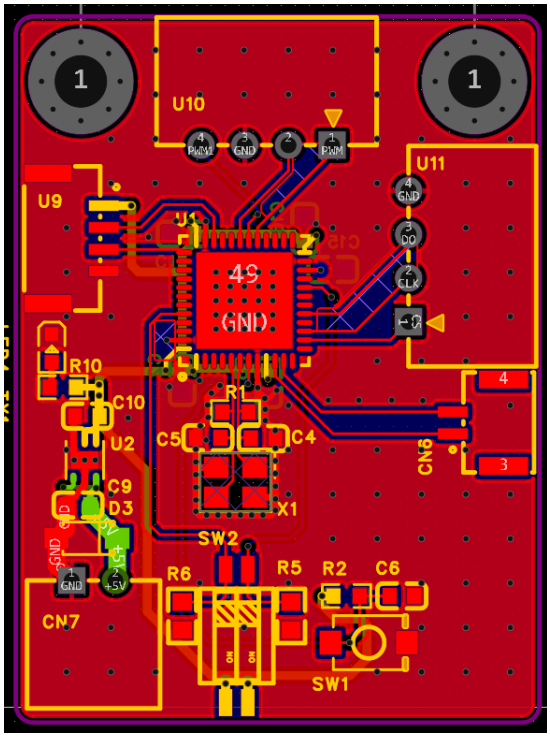


图 12 正面布局

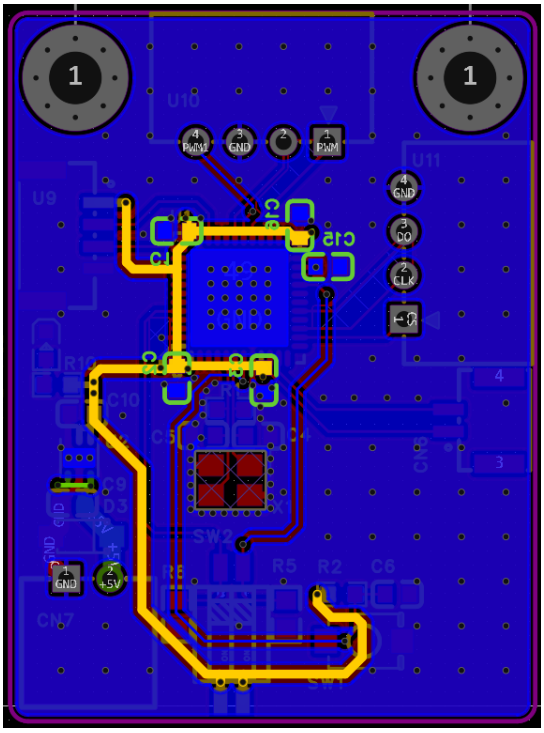


图 13 背面局部

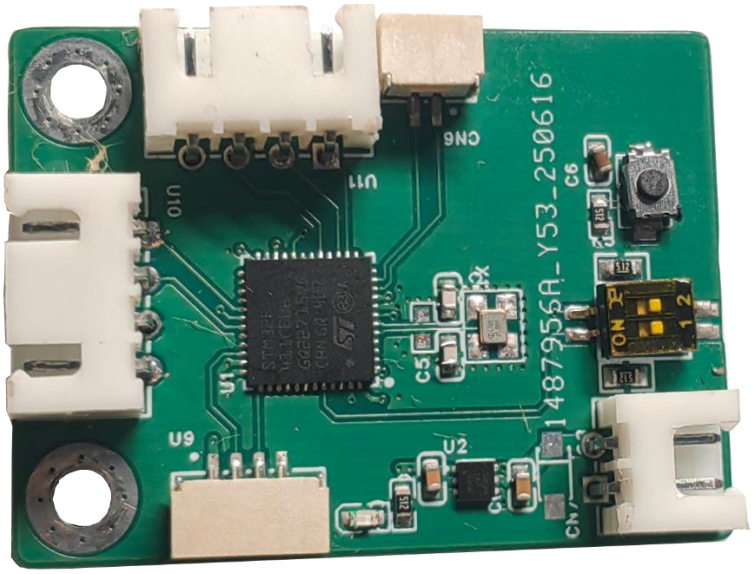


图 14 实物图

第3章 系统软件设计

3.1 仿真建模分析

根据题目参数搭建 BOOST 电路拓扑

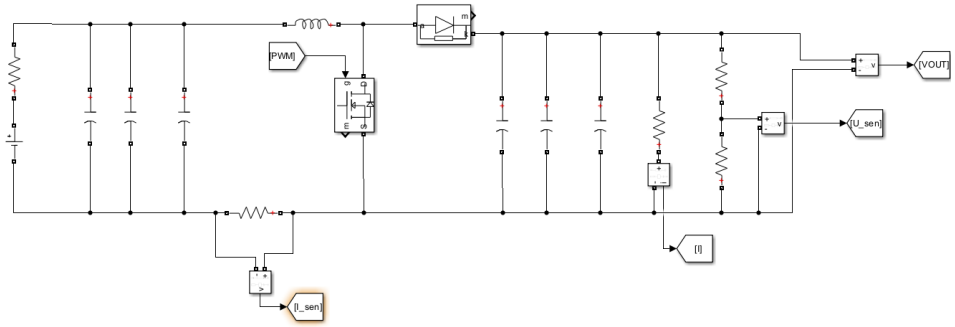


图 15 BOOST 电路拓扑

分别设定开环控制和闭环控制进行不同的分析:

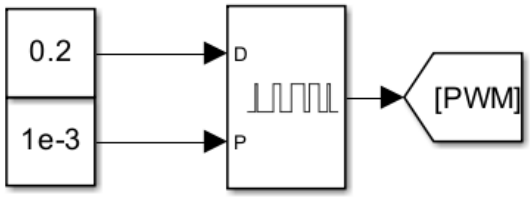


图 16 开环控制

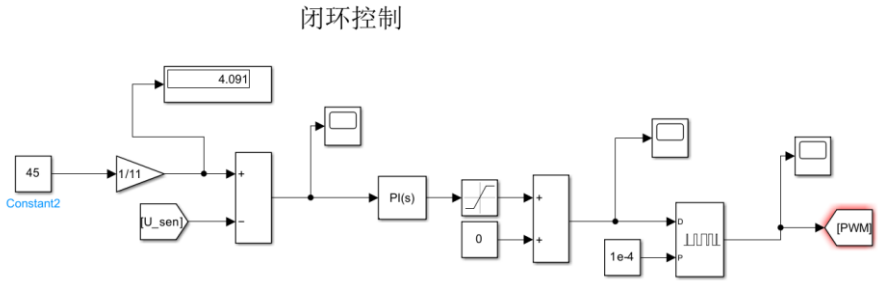


图 17 闭环控制

并设立有各种监测模块用于分析

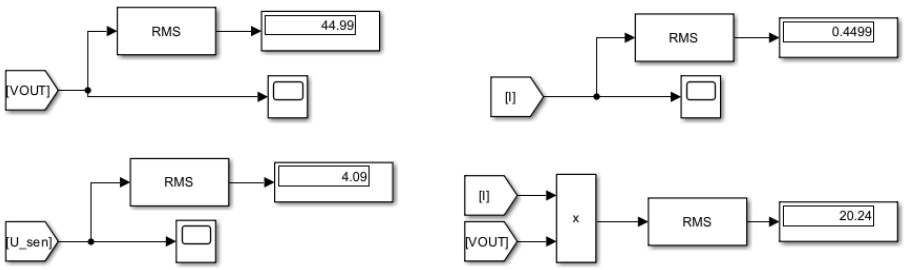


图 18 监测模块

3.1.1 开环分析

开分分析，我们的预期电压是 36V，输入电压为 18V，则根据稳态电压计算公式

$$V_{out}=\frac{V_{in}}{1-D}$$

则计算的到占空比应该为 50%

当考虑负载的影响时

$$\frac{V_{out}}{V_{\text{in}}}=\frac{1}{1-D}\cdot\frac{1}{1+\frac{1}{(1-D)^{2}}\cdot\frac{R_L}{R}}$$

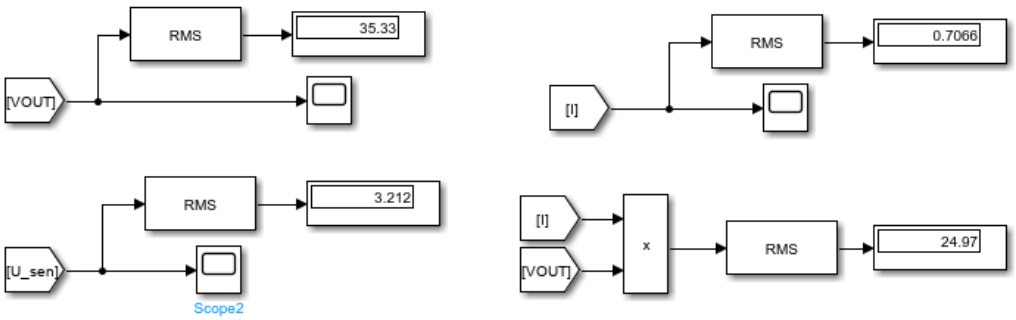


图 19 开环结果

3.1.2 闭环分析

通过调整 PI 参数，使偏差迅速趋于 0

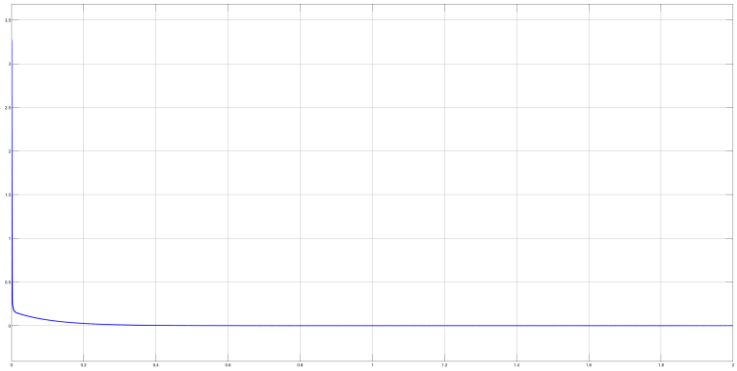


图 20 ERROR 数值监测

这使得我们的输出电压迅速达到我们给定的设定值:

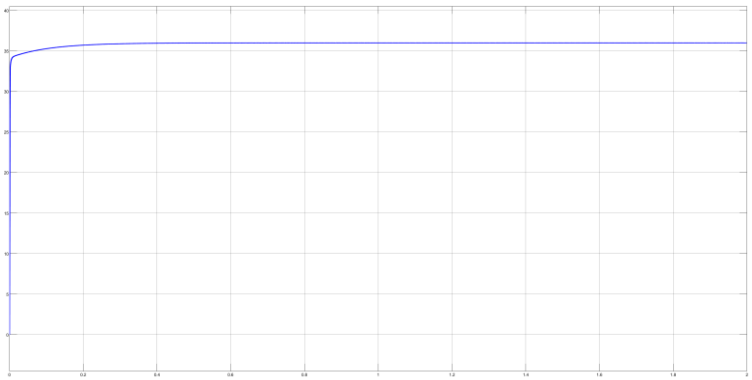


图 21 输出电压波形

3.2 开环特性分析

硬件设计完毕后，首先需要在开环条件下测定物理量的关系，用于拟合出 ADC 采样电压，占空比，以及输出电压的数据关系，随后拟合的结果可以用于闭环控制。

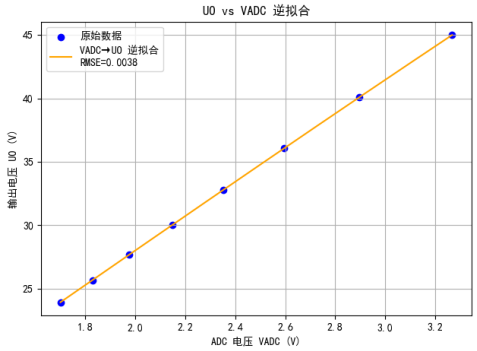
在开环条件下，100Ohm 额定负载，100kHzPWM 脉冲情况下测得如下数据用于拟合分析

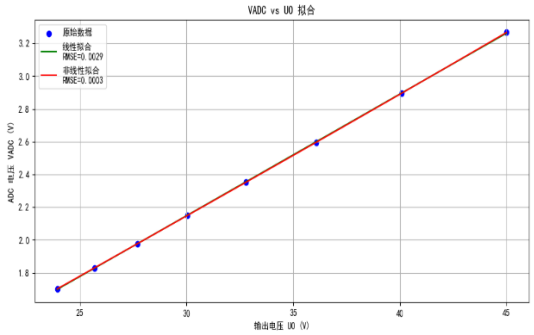
表格 2 开环参数测定

Duty (%)	UO (V)	VADC (V)
35%	45.000	3.267
40%	40.088	2.896
45%	36.080	2.596
50%	32.789	2.352
55%	30.030	2.149
60%	27.691	1.976
65%	25.679	1.829
70%	23.936	1.701

根据该开环参数可以得到拟合曲线，我们需要的到 ADC 电压和输出电压的关系用于设定目标值，我们还需要得到 ADC 电压和 PWM 波的关系，用于设定基础 PWM。

为了得到比较准确的数据我们建立了 ADC 电压模型，使用三次多项式尝试拟合





非线性拟合关系(三次多项式)

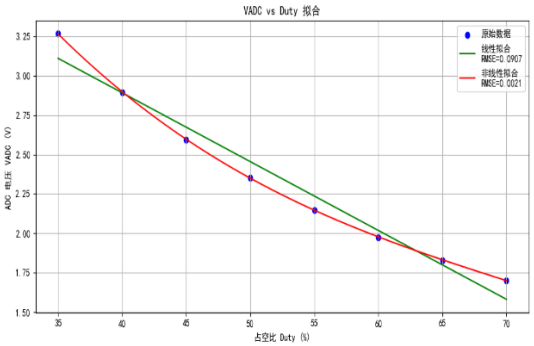
$$V_{ADC} = -0.000001U_O^3 - 0.000086U_O^2 + 0.074785U_O - 0.06280$$

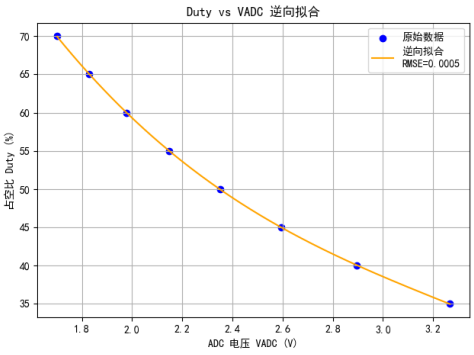
由于高次项误差极小，优化为一次最小二乘拟合:

$$V_{ADC} = 0.0743U_O - 0.0802$$

验证 RMSE 数值，其中线性拟合的 RMSE: 0.0029 V，多项式拟合的 RMSE: 0.0003 V，为了减小开销，并且线性拟合的 RMSE 值也不大，因此采用线性拟合的方式。

同样根据上述经验建立 PWM 和 VADC 的拟合关系，以提供基础占空比





$$V_{ADC} = -13.696969D^3 + 29.382250D^2 - 23.580367D + 8.506454$$

$$V_{ADC} = -4.371429D + 4.640750$$

其中最小而成得到的 RMSE: 0.0907 V，三次多项式的 RMSE: 0.0021 V，此时应当采用三次拟合以减小误差。根据该拟合关系球的逆拟合关系，通过 VADC 给出 Duty

$$D = -0.035971V_{ADC}^3 + 0.355110V_{ADC}^2 + -1.299556V_{ADC} + 2.059551$$

计算得到 RMSE: 0.0005，在合理的精度范围以内。

综上所述得到如下关系

$$\begin{cases} D = -0.035971V_{ADC}^3 + 0.355110V_{ADC}^2 - 1.299556V_{ADC} + 2.059551 \\ V_{ADC} = 0.0743U_O - 0.0802 \end{cases}$$

计算代码

```
/**
 * @brief 电压拟合：根据 vadc_target 目标电压返回基础占空比
 */
static float BaseDuty_FromFit(float vadc_target)
{
  // 方法一：三次多项式拟合（更精确）
  float v2 = vadc_target * vadc_target;
  float v3 = v2 * vadc_target;
  return 100.0f * (-0.035971f * v3 + 0.355110f * v2 - 1.299556f
* vadc_target + 2.059551f);

  // // 方法二：线性拟合（更高性能）
  // return 100.0f * (-0.221493f * vadc_target + 1.044567f);
}

/**
 * @brief 将输出电压转换为 ADC 目标电压（线性拟合）
 */
float calculate_vadc(float v_target)
{
  return 0.0743f * v_target - 0.0802f;
}
```

3. 3 闭环算法设计

闭环控制的软件主要由如下部分构成 0. 系统初始化 1. 双路 ADC 采样 2. DMA 循环传输 3. TIM1 产生 PWM 4. TIM2 产生特定频率的回调用于计算和控制频率 4. ADC 数据滤波 5. PI 闭环控制算法 。

软件平台选择 Cmake 构建工程， 工具链选择 GCC-ARM，IDE 使用 VSC，debug 基于 SWD 接口调用 ARM-GDB。

3. 3. 1 系统初始化

系统初始化主要包含如下部分 1. 全局 debug 变量声明，各种句柄声明 2. 各种 BSP 初始化 3. 等待 ADC 数据稳定 4. 滤波器初始化

3.3.2 采样滤波以及控制

该部分代码主要包含 ADC 采样，滤波以及定时器回调闭环控制，软件主要流程如下图所示:

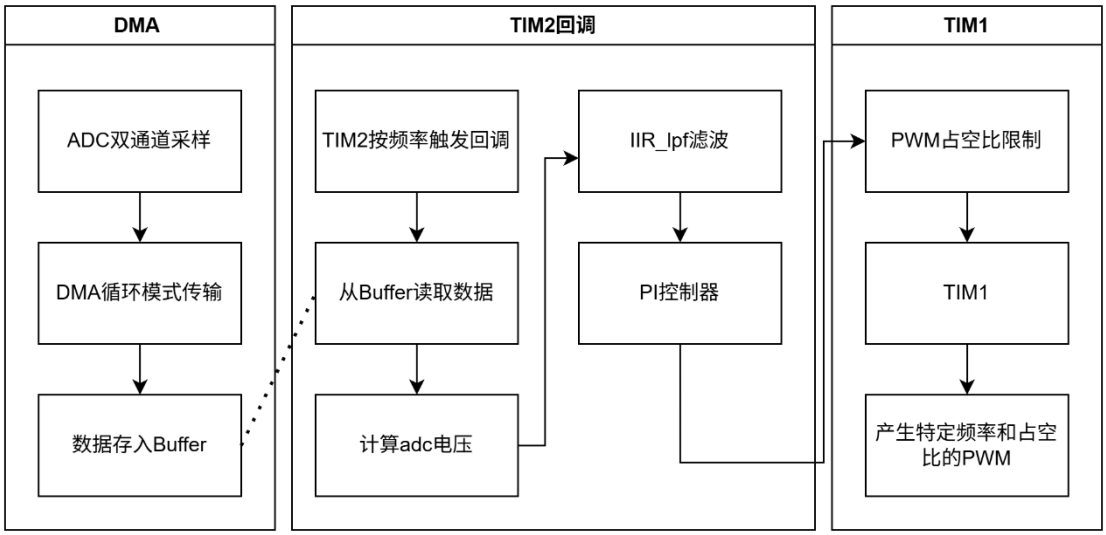


图 22 采样以及控制流程

ADC 部分采用双通道，因此使用 Scan Conversion Mode，由于开关频率在 100kHz 所以这里采样频率不用过高，从 PCLK2 时钟 6 分频即可。使能 Continuous 模式，一面再回调中重复触发中断，使能 DMA Continuous 并搭配 DMA 的 Circular 模式连续搬运数据，实现 ADC 的数据高效传输。

表格 3 ADC 详细配置

配置项	值
Clock Prescaler	PCLK2 divided by 6
Resolution	12 bits (15 ADC Clock cycles)
Data Alignment	Right alignment
Scan Conversion Mode	Enabled
Continuous Conversion Mode	Enabled
Discontinuous Conversion Mode	Disabled
DMA Continuous Requests	Enabled
End Of Conversion Selection	EOC flag at the end of single channel conversion

完成采样后通过设定 TIM2 定时器，稳定触发回调函数从内存中读取 ADC 存在 Buffer 中的数据，紧跟着计算 ADC 的电压，然后输入 IIR 低通滤波器，抑制高频噪声、平滑信号，然后输入 PI 控制器用于闭环控制。

回调计算流程

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  if (htim->Instance == TIM2)
  {
    // ADC 原始采样
    float v_raw = adc_buffer[0] * 3.3f / 4096.0f;
    float i_raw = adc_buffer[1] * 3.3f / 4096.0f;

    // 滤波处理
    v_adc = IIR_LPF_Update(&lpf_vadc, v_raw);
    current = IIR_LPF_Update(&lpf_current, i_raw);

    // PID 控制输出
    delta = PI_Update(&voltage_pi, vadc_target, v_adc,
CONTROL_PERIOD_S);
    duty = base_duty + delta;

    // 限制输出占空比
    if (duty < 0.0f)
      duty = 0.0f;
    if (duty > 100.0f)
      duty = 100.0f;

    // 应用 PWM
    Set_PWM_Duty(&htim1, TIM_CHANNEL_1, duty);
  }
}
```

闭环控制器采用 PI 控制，经过积分限幅后输入 TIM1 模块，产生控制所需要的 PWM 频率和占空比。

PI 控制器流程

```
float PI_Update(PI_Controller_t *pi, float target, float
feedback, float dt)
{
    //UPdate 使用全局变量
    pi_error = -(target - feedback);
```

```
    // 更新积分项
    pi->integral += pi_error * dt;
    // 防止积分项过大，限幅在 [-output_limit/ki, +output_limit/ki]
    if (pi->ki > 0.0f)
    {
        float int_max = pi->output_limit / pi->ki;
        if (pi->integral > int_max)
            pi->integral = int_max;
        else if (pi->integral < -int_max)
            pi->integral = -int_max;
    }

    // PI 计算
    float output = pi->kp * pi_error + pi->ki * pi->integral;

    // 输出限幅
    if (output > pi->output_limit)
        output = pi->output_limit;
    else if (output < -pi->output_limit)
        output = -pi->output_limit;

    return output;
}
```

详细的控制算法和代码见附录

第 4 章 系统调试与测试

4. 1 调试

调试平台为了保证实时性和最简化处理，通过 SWD 接口，以及从全局变量空间直接读取内存数据的方式进行调试分析，PI 参数的标定，搭配 CubeMonitor 等实时检测数据变化和响应。

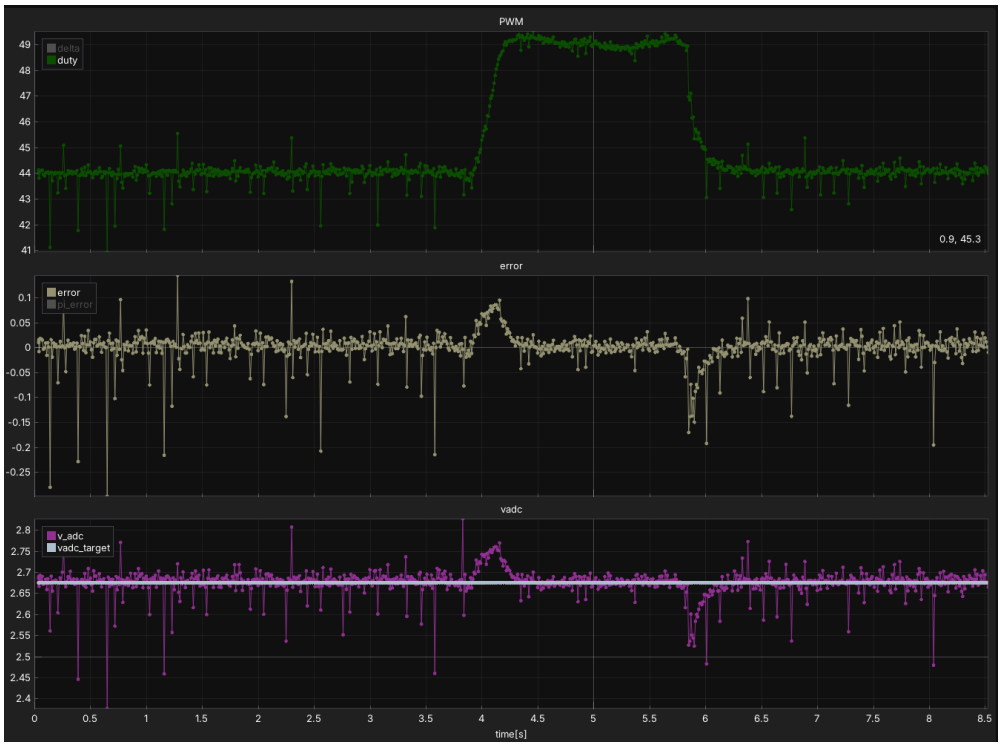


图 23 MCU Viewer 波形

根据 MCUviewer 波形调整 PI 参数

表格 4 MCU 系统常见 Debug 方式

方法	实时性	输出速度	使用复杂度	占用资源	优点	缺点
printf 串口重定向	中等	慢(几百 B/s)	低	占用 UART 资源	易用、格式化输出、通用	阻塞性强、影响实时性、速度慢
SWD+变量暂停观察	低(暂停)	低	中	占用调试器和 IDE	支持断点、单步调试,适合静态分析	程序必须暂停,不能动态跟踪变量
SWD + Live Watch	中高	中 (~1-10Hz)	中	仅占用 SWD	程序运行时查看变量,适合中低速控制调试	不适合高速信号、变量受限于编译优化

SEGGER RTT	高	快(>100KB/s)	中	J-Link + SRAM	非阻塞、无需串口、可视化好、速度极快	需 SEGGER J-Link、嵌入 RTT 库,配置略复杂
SWO	高	快 (10~100KB/s)	中	占用 SWO 引脚	低资源、高速、可视化输出变量、兼容 STM Studio 等工具	配置复杂、需 MCU 和调试器支持 SWO

这里由于回调频率并不高，而且无需 RTOS 等队列复杂 debug，因此采用 SWD 结合内存 LiveWatch 是本次任务的最佳调试方案。

书写 debug 规则:

Debug 规则(Launch.json)

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Build & Debug Microcontroller - ST-Link",
            "cwd": "${workspaceFolder}",
            "type": "cortex-debug",
            "executable": "${command:cmake.launchTargetPath}",
            // Let CMake extension decide executable:
"${command:cmake.launchTargetPath}"
            // Or fixed file path:
"${workspaceFolder}/path/to/filename.elf"
            "request": "launch",
            "servertype": "stlink",
            "device": "STM32F411CEUx", //MCU used
            "interface": "swd",
            "serialNumber": "", //Set ST-Link ID if you use
multiple at the same time
            "runToEntryPoint": "main",
            "svdFile":
"${config:STM32VSCodeExtension.cubeCLT.path}/STMicroelectronics_
CMSIS_SVD/STM32F411.svd",
            "v1": false, //Change it depending on ST Link version
```

```
            "serverpath":
"${config:STM32VSCodeExtension.cubeCLT.path}/STLink-gdb-
server/bin/ST-LINK_gdbserver",
            "stm32cubeprogrammer":
"${config:STM32VSCodeExtension.cubeCLT.path}/STM32CubeProgrammer
/bin",
            "stlinkPath":
"${config:STM32VSCodeExtension.cubeCLT.path}/STLink-gdb-
server/bin/ST-LINK_gdbserver",
            "armToolchainPath":
"${config:STM32VSCodeExtension.cubeCLT.path}/GNU-tools-for-
STM32/bin",
            "gdbPath":
"${config:STM32VSCodeExtension.cubeCLT.path}/GNU-tools-for-
STM32/bin/arm-none-eabi-gdb",
            "serverArgs": [
                "-m",
                "0",
            ],
            "liveWatch": {
                "enabled": true,
                "samplesPerSecond": 4
            }
            //"preLaunchTask": "Build + Flash"
            /* If you use external loader, add additional
arguments */
            //"serverArgs": ["--extload",
"path/to/ext/loader.stldr"],
        },
        {
            "name": "Attach to Microcontroller - ST-Link",
            "cwd": "${workspaceFolder}",
            "type": "cortex-debug",
            "executable": "${command:cmake.launchTargetPath}",
            // Let CMake extension decide executable:
"${command:cmake.launchTargetPath}"
            // Or fixed file path:
"${workspaceFolder}/path/to/filename.elf"
            "request": "attach",
            "servertype": "stlink",
            "device": "STM32F411CEUx", //MCU used
            "interface": "swd",
            "serialNumber": "", //Set ST-Link ID if you use
multiple at the same time
```

```
            "runToEntryPoint": "main",
            "svdFile":
"${config:STM32VSCodeExtension.cubeCLT.path}/STMicroelectronics_
CMSIS_SVD/STM32F411.svd",
            "v1": false, //Change it depending on ST Link version
            "serverpath":
"${config:STM32VSCodeExtension.cubeCLT.path}/STLink-gdb-
server/bin/ST-LINK_gdbserver",
            "stm32cubeprogrammer":
"${config:STM32VSCodeExtension.cubeCLT.path}/STM32CubeProgrammer
/bin",
            "stlinkPath":
"${config:STM32VSCodeExtension.cubeCLT.path}/STLink-gdb-
server/bin/ST-LINK_gdbserver",
            "armToolchainPath":
"${config:STM32VSCodeExtension.cubeCLT.path}/GNU-tools-for-
STM32/bin",
            "gdbPath":
"${config:STM32VSCodeExtension.cubeCLT.path}/GNU-tools-for-
STM32/bin/arm-none-eabi-gdb",
            "serverArgs": [
                "-m",
                "0",
            ],
            /* If you use external loader, add additional
arguments */
            //"serverArgs": ["--extload",
"path/to/ext/loader.stldr"],
        }
    ]
}
```

绘制调试数据流:

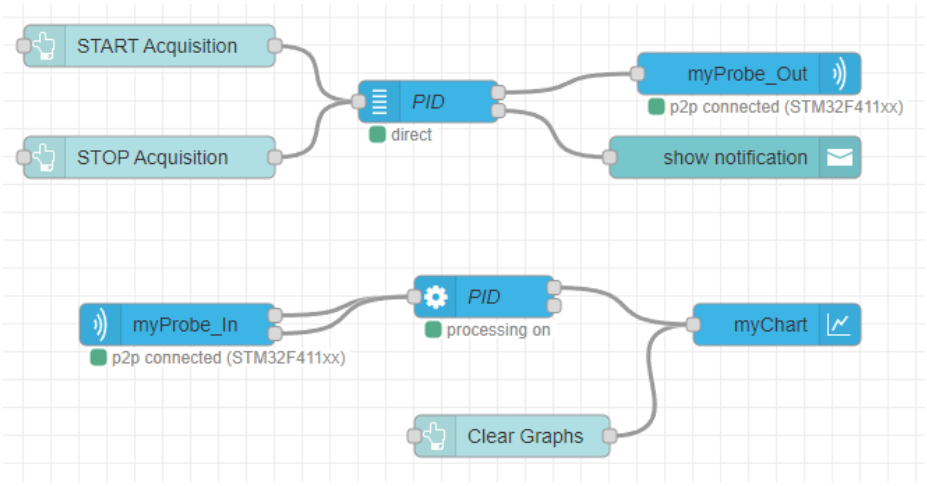


图 24 Debug 数据流

建立完数据流之可以根据 Chart 进行误差分析

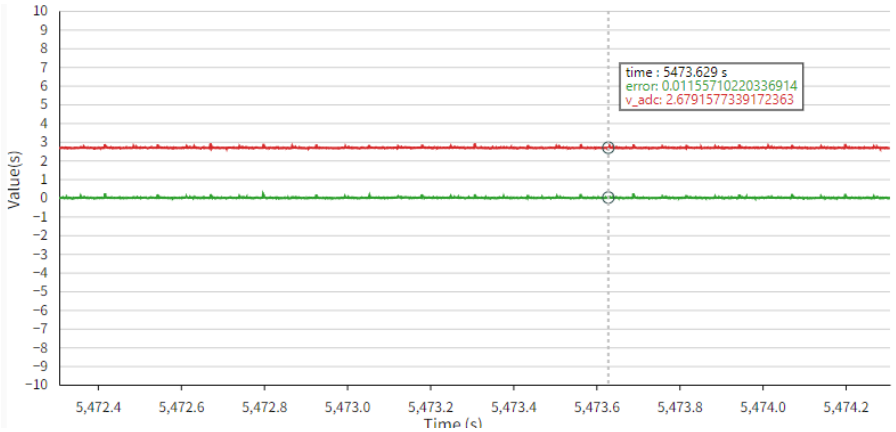


图 25 误差分析

开启 MCUgragh 可视化调整 PI 参数

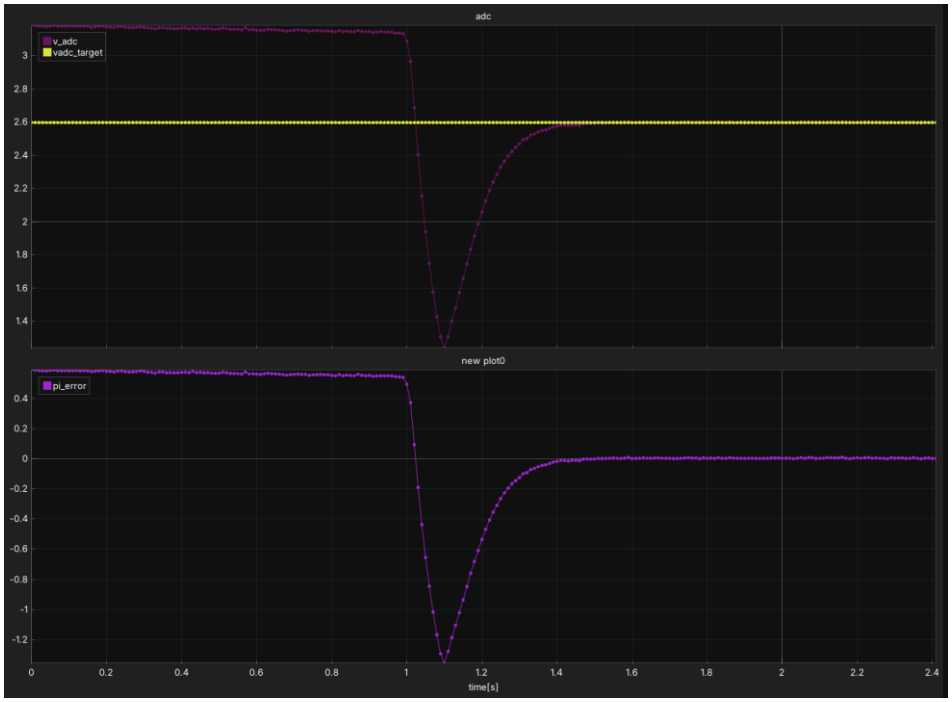


图 26 Error 偏差图

4.2 测试

表格 5 课程设计设计要求

设计要求
输出电压 U_O：35-40VDC（按 36VDC 额定电压设计）。
最大输出电流 I_{Omax}：2A（实验室提供 100Ω 滑动变阻器）。
具有电路保护功能（通过 AD 模块采样软件保护）。
I_O 从 0 变到 1A 时，输出电压 U_O 波动不超过±10%。
电源效率 $\eta \geq 80\%$（U_2=18VAC,U_O=36VDC, I_O=1A）。

开关电源的测试指标如下表所示：

表格 6 电源常见测试项目

测试类别	测试指标
输入输出参数	输入电压范围 (VBBIN)
	输出电压范围 (VOUT)
	输出电压精度(VOUT_ACC)
	最大输出电流(IOMAX)
动态性能	负载调整率(AV_BOOST_LOAD)

	电压调整率 (AV_BUS)
	Phase 点 Jitter (jitter)
	动态响应时间(T_LOAD)
保护机制	过电压保护(VBUSOVP)
	欠压保护(V_BOOSTUV)
	热保护(T_SHUTDOWN)
	电流限制(I_LIM)
效率与损耗	总功率损耗(P_tot)
	MOSFET 导通损耗(R_DS(ON))
	输出电压纹波(Vpp)
其他关键指标	启动时间(TSTART)
	静态电流(ISHUT)
	栅极驱动峰值电流(IGC1 Peak)

本次主要测试动态指标，效率，纹波以及温度

4.2.1 动态带载

负载跳变 0.2A~1.2A,电压超调 4.75V，恢复时间 180ms

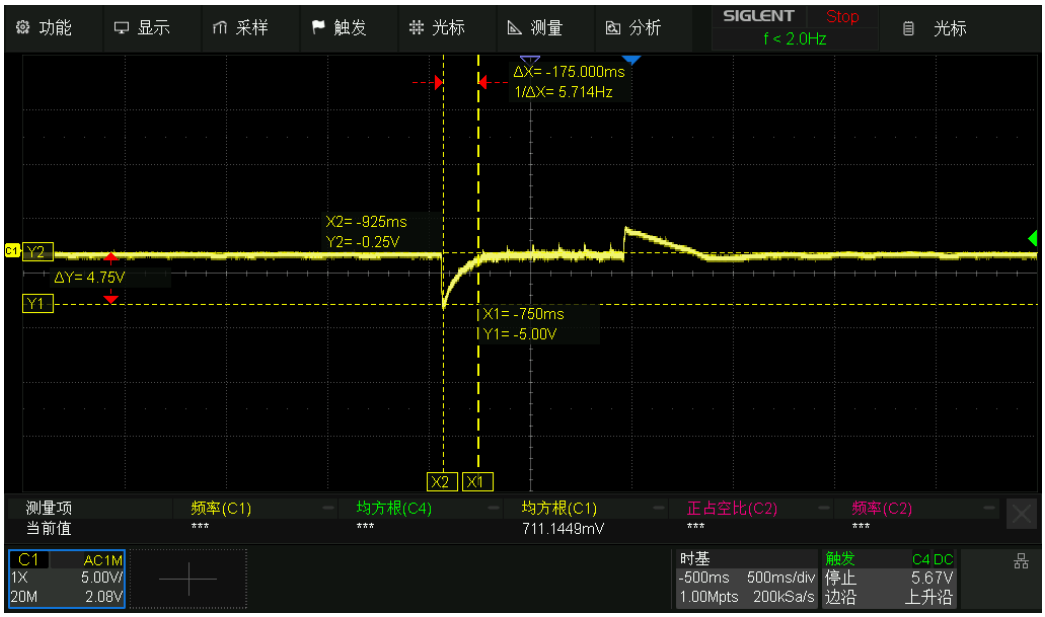


图 27 动态带载测试

4. 2. 2 效率分析

Pi	Po
8.657	7.58
12.70	11.28
16.7	14.93
20.5	18.5
24.5	22.0
28.3	25.5
32.2	28.9
36.0	32.23
39.9	35.5

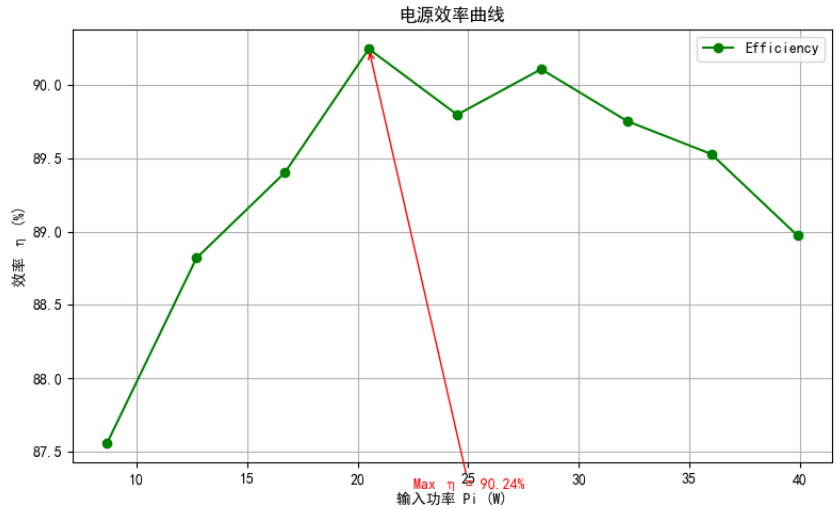


图 28 电源效率分析

效率区间为 87.5%~90%

4. 2. 3 纹波

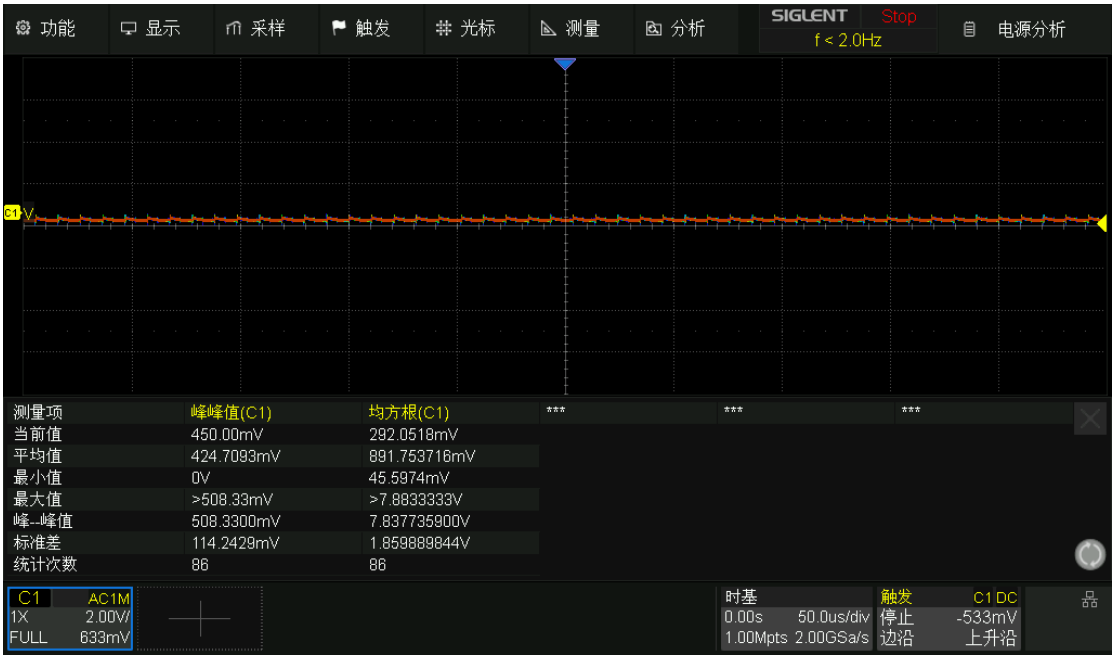


图 29 纹波分析

4.2.4 温度分析

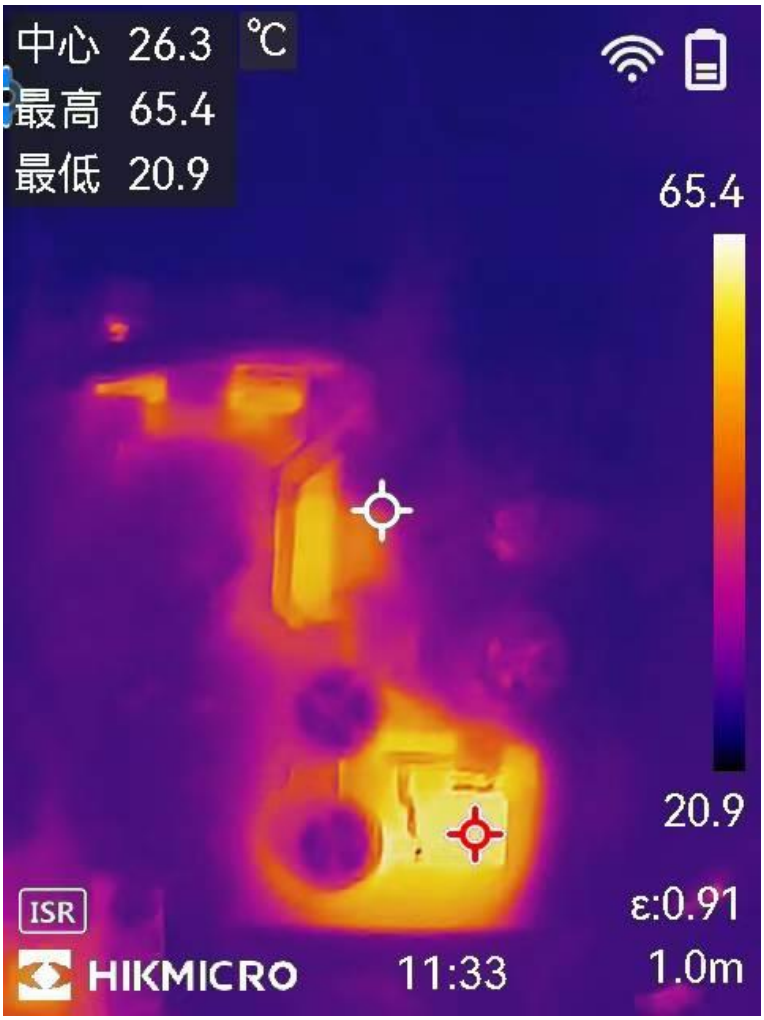


图 30 温度分析

4.2.5 EMI 测试

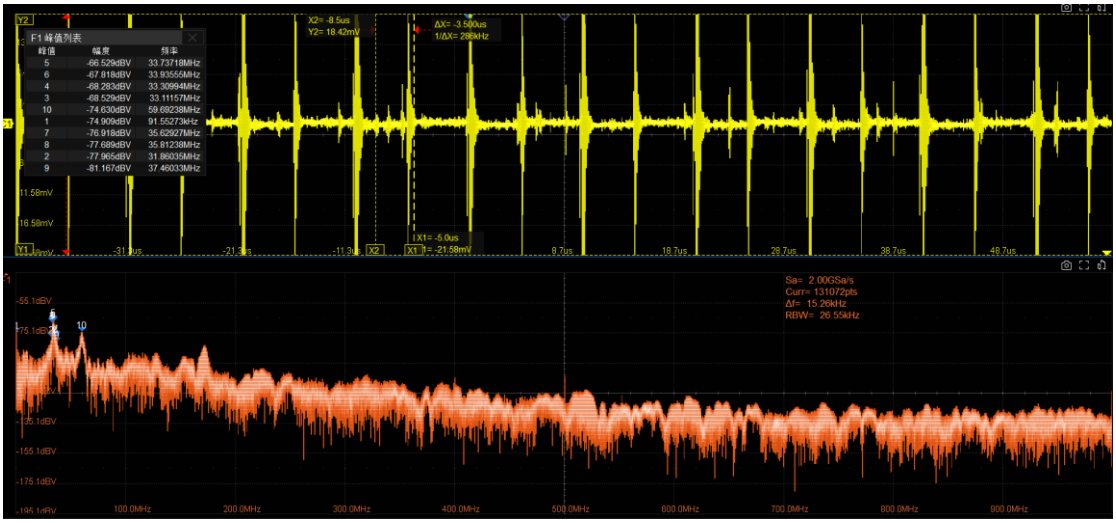


图 31 全频段 EMI

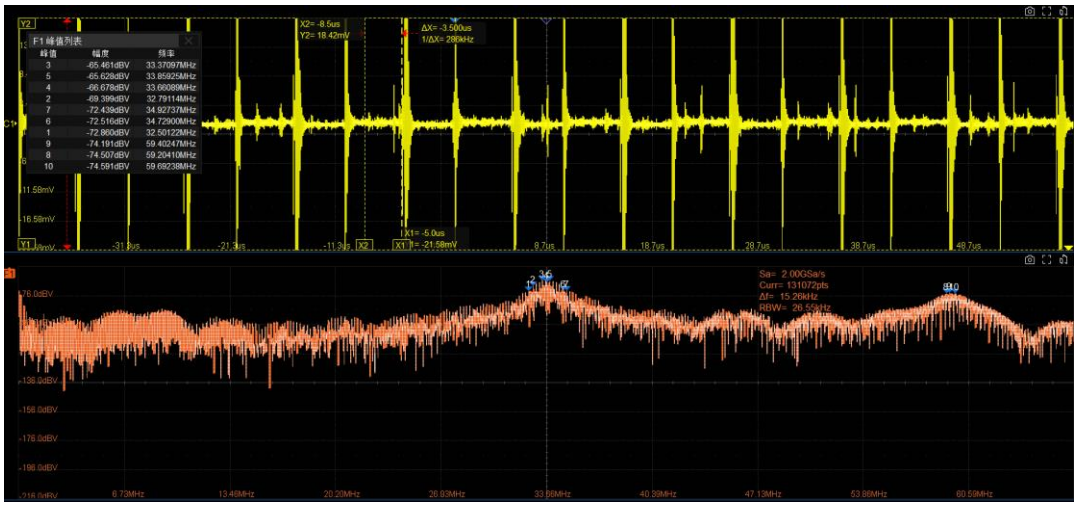


图 32 峰值

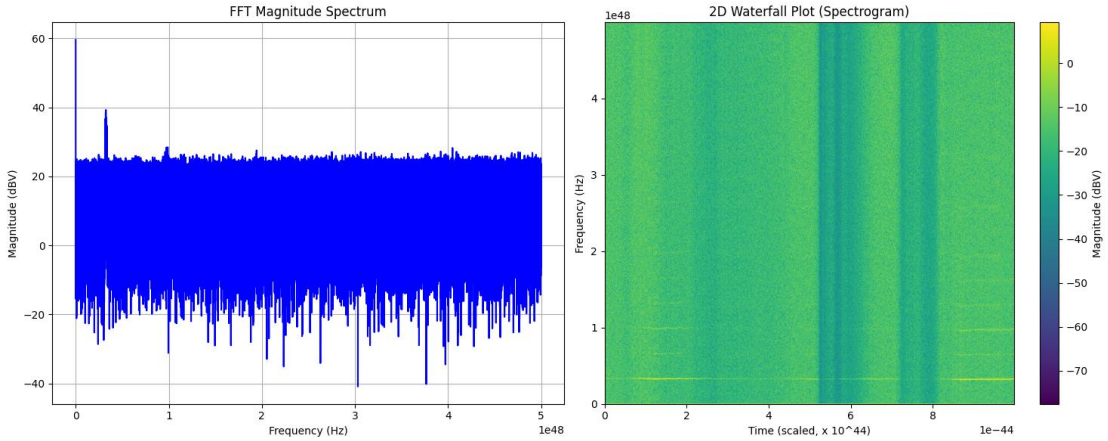


图 33 瀑布图

第 5 章 设计总结

此次设计，即使 ERROR 偏差非常接近 0，让 ADC 的采样值稳定不变，但是输出电压依然会随着负载的变化而产生波动，这是因为输出电压为 40V 的高压，因此 ADC 需要使用 FB 分压电阻(14 变比)到 ADC，次数输出电压的波动在 FB 上会变得非常小，ADC 的精度不足以跟踪。尝试引入电流前馈补偿，但是收效仍然甚微，因此后续可以考虑提高 ADC 精度，调整运放参数等优化电路输出电压精度。

第 6 章 参考文献

无参考，自行设计

6. 1 附录

Main.c

```
/* USER CODE BEGIN Header */
/**
 ******************************************************************************
 * @file           : main.c
 * @brief          : Main program body
 ******************************************************************************
 * @attention
 *
 * Copyright (c) 2025 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 *
 ******************************************************************************
 */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "adc.h"
#include "dma.h"
#include "tim.h"
#include "gpio.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
```

```c
/* USER CODE END PTD */

/* Private define ------------------------------------------------
--------------*/
/* USER CODE BEGIN PD */
#define CONTROL_PERIOD_S (0.001f) // 控制周期 1 ms
/* USER CODE END PD */

/* Private macro -------------------------------------------------
--------------*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------
--------------*/

/* USER CODE BEGIN PV */
__attribute__((aligned(4))) uint16_t adc_buffer[2];

float v_adc = 0.0f;
float current = 0.0f;

float v_target = 36.00f;
float vadc_target = 0.0f;
float delta = 0.0f;
float base_duty = 0.0f;
float duty = 0.0f;

PI_Controller_t voltage_pi;
PIController v_pi;
/* 定义滤波器句柄 */
static IIR_LPF_HandleTypeDef lpf_vadc;
static IIR_LPF_HandleTypeDef lpf_current;
/* USER CODE END PV */

/* Private function prototypes -----------------------------------
--------------*/
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */
```

```c
/* Private user code ---------------------------------------------
--------------*/
/* USER CODE BEGIN 0 */

/**
 * @brief 电压拟合：根据 vadc_target 目标电压返回基础占空比
 */
static float BaseDuty_FromFit(float vadc_target)
{
  // 方法一：三次多项式拟合（更精确）
  float v2 = vadc_target * vadc_target;
  float v3 = v2 * vadc_target;
  return 100.0f * (-0.035971f * v3 + 0.355110f * v2 - 1.299556f
* vadc_target + 2.059551f);

  // // 方法二：线性拟合（更高性能）
  // return 100.0f * (-0.221493f * vadc_target + 1.044567f);
}

/**
 * @brief 将输出电压转换为 ADC 目标电压（线性拟合）
 */
float calculate_vadc(float v_target)
{
  // 直接拟合@1000hm Load
  // return 0.0743f * v_target - 0.0802f;

  // 电流补偿
  return 0.0743f * v_target - 1.0553 * current;
}

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  if (htim->Instance == TIM2)
  {
    // ADC 原始采样
    float v_raw = adc_buffer[0] * 3.3f / 4096.0f;
    float i_raw = adc_buffer[1] * 3.3f / 4096.0f;

    // 滤波处理
    v_adc = IIR_LPF_Update(&lpf_vadc, v_raw);
    current = IIR_LPF_Update(&lpf_current, i_raw);

    // 不滤波
```

```
    //  v_adc = v_raw;
    //  current = i_raw;

    // PID 控制输出
    // delta = PI_Update(&voltage_pi, vadc_target, v_adc,
CONTROL_PERIOD_S);
    delta = PIController_Update(&v_pi, vadc_target, v_adc);
    duty = base_duty + delta;

    // 限制输出占空比
    if (duty < 0.0f)
      duty = 0.0f;
    if (duty > 100.0f)
      duty = 100.0f;

    // 应用 PWM
    Set_PWM_Duty(&htim1, TIM_CHANNEL_1, duty);
  }
}

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{

  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------
--------------*/

  /* Reset of all peripherals, Initializes the Flash interface
and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */
```

```c
  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_DMA_Init();
  MX_ADC1_Init();
  MX_TIM1_Init();
  MX_TIM2_Init();
  /* USER CODE BEGIN 2 */
  // PI_Init(&voltage_pi, 2.0f, 120.0f, 500.0f);
  PIController_Init(&v_pi, 10.0f, 300.0f, 0.001f, -50.0f,
50.0f); // kp, ki, dt(1ms), min, max
  vadc_target = calculate_vadc(v_target);
  base_duty = BaseDuty_FromFit(vadc_target);

  HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
  Set_PWM_Duty(&htim1, TIM_CHANNEL_1, 45); // 初始占空比
  /* 首次读取 ADC，获得初始值 */
  HAL_ADC_Start_DMA(&hadc1, (uint32_t *)adc_buffer, 2);
  HAL_Delay(10); // 等待 DMA 首次采样完成
  float v_init = adc_buffer[0] * 3.3f / 4096.0f;
  float curr_init = adc_buffer[1] * 3.3f / 4096.0f;
  /* 初始化滤波器，α 推荐 0.6 */
  IIR_LPF_Init(&lpf_vadc, 0.6f, v_init);
  IIR_LPF_Init(&lpf_current, 0.60f, curr_init);

  HAL_TIM_Base_Start_IT(&htim2); // 启动中断模式定时器
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
    /* USER CODE END WHILE */

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

```c
/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

  /** Initializes the RCC Oscillators according to the specified
parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue =
RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 8;
  RCC_OscInitStruct.PLL.PLLN = 100;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType =
RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_P
CLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;
```

```
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct,
FLASH_LATENCY_3) != HAL_OK)
  {
    Error_Handler();
  }
}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
  * @brief  This function is executed in case of error
occurrence.
  * @retval None
  */
void Error_Handler(void)
{
  /* USER CODE BEGIN Error_Handler_Debug */
  __disable_irq();
  while (1)
  {
  }
  /* USER CODE END Error_Handler_Debug */
}

#ifdef  USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source
line number
  *         where the assert_param error has occurred.
  * @param  file: pointer to the source file name
  * @param  line: assert_param error line source number
  * @retval None
  */
void assert_failed(uint8_t *file, uint32_t line)
{
  /* USER CODE BEGIN 6 */
  // printf("Wrong parameters value: file %s on line %d\r\n",
file, line);
  /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */
```

Adc.c

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file    adc.c
  * @brief   This file provides code for the configuration
  *          of the ADC instances.
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2025 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
  ******************************************************************************
  */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "adc.h"

/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

ADC_HandleTypeDef hadc1;
DMA_HandleTypeDef hdma_adc1;

/* ADC1 init function */
void MX_ADC1_Init(void)
{
```

```
  /* USER CODE BEGIN ADC1_Init 0 */

  /* USER CODE END ADC1_Init 0 */

  ADC_ChannelConfTypeDef sConfig = {0};

  /* USER CODE BEGIN ADC1_Init 1 */

  /* USER CODE END ADC1_Init 1 */

  /** Configure the global features of the ADC (Clock,
Resolution, Data Alignment and number of conversion)
  */
  hadc1.Instance = ADC1;
  hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV4;
  hadc1.Init.Resolution = ADC_RESOLUTION_12B;
  hadc1.Init.ScanConvMode = ENABLE;
  hadc1.Init.ContinuousConvMode = ENABLE;
  hadc1.Init.DiscontinuousConvMode = DISABLE;
  hadc1.Init.ExternalTrigConvEdge =
ADC_EXTERNALTRIGCONVEDGE_NONE;
  hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
  hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
  hadc1.Init.NbrOfConversion = 2;
  hadc1.Init.DMAContinuousRequests = ENABLE;
  hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
  if (HAL_ADC_Init(&hadc1) != HAL_OK)
  {
    Error_Handler();
  }

  /** Configure for the selected ADC regular channel its
corresponding rank in the sequencer and its sample time.
  */
  sConfig.Channel = ADC_CHANNEL_0;
  sConfig.Rank = 1;
  sConfig.SamplingTime = ADC_SAMPLETIME_480CYCLES;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }

  /** Configure for the selected ADC regular channel its
corresponding rank in the sequencer and its sample time.
```

```
  */
  sConfig.Channel = ADC_CHANNEL_1;
  sConfig.Rank = 2;
  if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN ADC1_Init 2 */

  /* USER CODE END ADC1_Init 2 */

}

void HAL_ADC_MspInit(ADC_HandleTypeDef* adcHandle)
{

  GPIO_InitTypeDef GPIO_InitStruct = {0};
  if(adcHandle->Instance==ADC1)
  {
  /* USER CODE BEGIN ADC1_MspInit 0 */

  /* USER CODE END ADC1_MspInit 0 */
    /* ADC1 clock enable */
    __HAL_RCC_ADC1_CLK_ENABLE();

    __HAL_RCC_GPIOA_CLK_ENABLE();
    /**ADC1 GPIO Configuration
    PA0-WKUP     ------> ADC1_IN0
    PA1     ------> ADC1_IN1
    */
    GPIO_InitStruct.Pin = GPIO_PIN_0|GPIO_PIN_1;
    GPIO_InitStruct.Mode = GPIO_MODE_ANALOG;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

    /* ADC1 DMA Init */
    /* ADC1 Init */
    hdma_adc1.Instance = DMA2_Stream0;
    hdma_adc1.Init.Channel = DMA_CHANNEL_0;
    hdma_adc1.Init.Direction = DMA_PERIPH_TO_MEMORY;
    hdma_adc1.Init.PeriphInc = DMA_PINC_DISABLE;
    hdma_adc1.Init.MemInc = DMA_MINC_ENABLE;
    hdma_adc1.Init.PeriphDataAlignment =
DMA_PDATAALIGN_HALFWORD;
```

```
    hdma_adc1.Init.MemDataAlignment = DMA_MDATAALIGN_HALFWORD;
    hdma_adc1.Init.Mode = DMA_CIRCULAR;
    hdma_adc1.Init.Priority = DMA_PRIORITY_LOW;
    hdma_adc1.Init.FIFOMode = DMA_FIFOMODE_DISABLE;
    if (HAL_DMA_Init(&hdma_adc1) != HAL_OK)
    {
      Error_Handler();
    }

    __HAL_LINKDMA(adcHandle,DMA_Handle,hdma_adc1);

  /* USER CODE BEGIN ADC1_MspInit 1 */

  /* USER CODE END ADC1_MspInit 1 */
  }
}

void HAL_ADC_MspDeInit(ADC_HandleTypeDef* adcHandle)
{

  if(adcHandle->Instance==ADC1)
  {
  /* USER CODE BEGIN ADC1_MspDeInit 0 */

  /* USER CODE END ADC1_MspDeInit 0 */
    /* Peripheral clock disable */
    __HAL_RCC_ADC1_CLK_DISABLE();

    /**ADC1 GPIO Configuration
    PA0-WKUP     ------> ADC1_IN0
    PA1     ------> ADC1_IN1
    */
    HAL_GPIO_DeInit(GPIOA, GPIO_PIN_0|GPIO_PIN_1);

    /* ADC1 DMA DeInit */
    HAL_DMA_DeInit(adcHandle->DMA_Handle);
  /* USER CODE BEGIN ADC1_MspDeInit 1 */

  /* USER CODE END ADC1_MspDeInit 1 */
  }
}

/* USER CODE BEGIN 1 */
```

```
/* USER CODE END 1 */
```

Dma.c

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file    dma.c
  * @brief   This file provides code for the configuration
  *          of all the requested memory to memory DMA transfers.
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2025 STMicroelectronics.
  * All rights reserved.
  *
  * This software is licensed under terms that can be found in the LICENSE file
  * in the root directory of this software component.
  * If no LICENSE file comes with this software, it is provided AS-IS.
  *
  ******************************************************************************
  */
/* USER CODE END Header */

/* Includes ------------------------------------------------------------------*/
#include "dma.h"

/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/*----------------------------------------------------------------------------*/
/* Configure
DMA                                                                        */
```

```
/*----------------------------------------------------------------
--------------*/

/* USER CODE BEGIN 1 */

/* USER CODE END 1 */

/**
  * Enable DMA controller clock
  */
void MX_DMA_Init(void)
{

  /* DMA controller clock enable */
  __HAL_RCC_DMA2_CLK_ENABLE();

  /* DMA interrupt init */
  /* DMA2_Stream0_IRQn interrupt configuration */
  HAL_NVIC_SetPriority(DMA2_Stream0_IRQn, 0, 0);
  HAL_NVIC_EnableIRQ(DMA2_Stream0_IRQn);

}

/* USER CODE BEGIN 2 */

/* USER CODE END 2 */
```

Iir_lpf.c

```
/* iir_lpf.c */
#include "iir_lpf.h"

void IIR_LPF_Init(IIR_LPF_HandleTypeDef *h, float alpha, float
init)
{
    if (alpha < 0.0f)
        alpha = 0.0f;
    if (alpha > 1.0f)
        alpha = 1.0f;
    h->alpha = alpha;
    h->prev_out = init;
}
```

```c
float IIR_LPF_Update(IIR_LPF_HandleTypeDef *h, float sample)
{
    // y[n] = α·y[n-1] + (1-α)·x[n]
    h->prev_out = h->alpha * h->prev_out + (1.0f - h->alpha) *
sample;
    return h->prev_out;
}
```