技术文件

完成时间：2012-12-16

工程实践与科技创新[3A]
设计报告

2012 年

项目名称：DC-DC 开关电源模块及其控制系统

设计小组编号：87

设计小组名单：强杰(组长)、郝宇、熊小凡

上海交通大学 电子信息与电气工程学院

姓名	班级	学号	具体负责的工作	联系方式
强杰	F1003008	5100309366	焊接，调试	18801969330
郝宇	F1003008	5100309368	程序，调试	18801969495
熊小凡	F1003008	5100309382	调试，焊接	18801970638

上海交通大学 电子信息与电气工程学院

地　址：东川路 800 号

邮　编：200240

摘　要：本报告是上海交通大学 2010 级工程实践与科技创新（3A）的理论研究与实验报告部分，详细阐述制作降压型 DC-DC 开关电源及其控制系统的设计思路、实现方式、制作过程及最后测试结果。降压型 DC-DC 开关电源及其控制系统包含开环控制和闭环控制两种控制模式，可实现电压 20~30V 输入 5~10V 输出。

系统以 TL494 为核心，单片机通过改变输出的 PWM 信号的占空比来控制开关三极管的通断，从而达到控制输出电压的目的，而 ADC 则对输出信号进行测量并反馈给单片机分析处理，形成闭环系统。

关键词：MCU AVR ATmega16单片机，降压型DC-DC开关电源，开环控制，闭环控制，MSP430单片机

ABSTRACT

This article is a theoretical part of engineering practice and technological innovation(3A), in SJTU 2010. This article presents the design, implementation, debugging and the performance of the DC-DC switch power supply. The dropping DC-DC switch power supply has two different control mode: open loop control system and closed loop control system.This system can convert input voltage of 20~30V to output of 5~10V.

This type of power supply system uses TL494 as its core element. The MCS dominates the connection and disconnection of the switch audion through changing the duty cycle of its output PWM signal, in order to control the output voltage. Moreover, the ADC converts the output signal and feeds back to MCS, which can then analyse and then deal with it.

KEYWORDS

Switch mode DC-DC power supply, MCU AVR ATmega16, open-loop control, closed-loop control，MPS430 power supply

---

上海交通大学 电子信息与电气工程学院

地　址：东川路 800 号

邮　编：200240

目　　录

1. 概述

1.1 编写说明

本篇报告为上海交通大学电子信息与电气工程学院学生大三第一学期科技创新(3A)课程的设计报告，详细阐述了基于单片机控制 DC－DC 电压转换器的设计原理，硬件结构，软件结构以及使用说明。旨在全面记录本实验小组的设计思路和操作过程，总结经验与心得体会，供指导老师在检查评分时参考，亦可作为与同学交流沟通的书面材料。本文适合电子相关专业人士以及有一定理论基础的业余电子设计爱好者阅读。

1.2 名词定义

单片机小系统：把中央处理器、存储器、输入输出接口都集成在一块集成电路芯片上，这样的微型计算机叫做单片机。它的最大优点是体积小，可放在仪表内部，但存储量小，输入输出接口简单，功能较低。

降压型 DC-DC 开关电源：将 20V~30V 的电源输入转化为 5V~10V 的电压输出的电路系统。

开环控制系统：利用单片机小系统控制其输出的 PWM 波占空比来改变电压输出的控制系统。

闭环控制系统：对输出电压进行测量并反馈给单片机小系统从而调节单片机的输出 PWM 波占空比来精确控制电压输出的控制系统。

过流保护：设定系统电流上限，防止功耗过大，导致元件损坏。

电压调整率：输入电压变化时，输出电压变化幅度与输入电压变化幅度的比值。

纹波：指输出直流电压中含有的工频交流成分。

中断机制：处理突发事件(CASE)的方式之一。

Keil C：电脑上的单片机高级语言编译系统。允许用户用 C 语言的格式书写单片机应用程序，并用自带的编译器将其编译为十六进制的 hex 文件，能够直接被单片机所接受运行。

ISPsoftware：专用软件，将编译好的十六进制应用程序文件写入单片机的存储区 。

1.3 硬件开发环境

表 1.1 硬件开发环境

开发工具	开发环境	作用
直流稳压电源	实验室，室温	为系统供电
数字万用表	实验室，室温	测量直流电压
示波器	实验室，室温	测量纹波
QBG-3B 高频 Q 表	实验室，室温	测量电感
电源引线及示波器探头	实验室，室温	提供电器连接

1.4 软件开发环境

表 1.2 软件开发工具及运行环境

软件名称	运行环境
Keil uVision 2	Windows XP
Proteus6.7	Windows XP
ISPLAY	Windows XP
Protel 99 SE	Windows XP
Origin 7.0	Windows XP
MATLAB 7.1	Windows XP

1.5 缩略语

DC	(direct current)	直流
PWM	(pulse width modulation)	脉冲宽度调制信号
LPF	(low pass filter)	低通滤波器
A	(analog)	模拟
D	(digital)	数字
ADC	(analog-digital convert)	模拟-数字转换

2. 系统总述

2.1 系统组成

2.1.1 系统概述

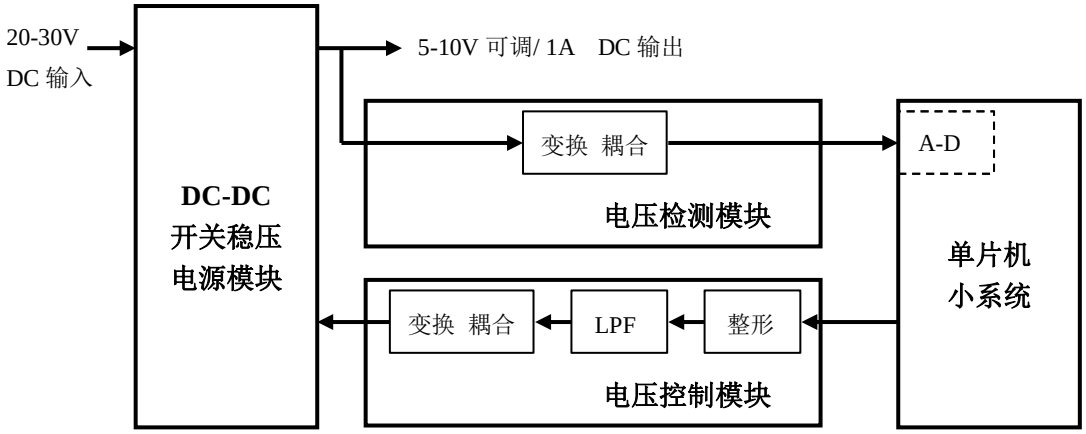


图 2.1 系统组成示意图[1]

降压型 DC-DC 开关电源及其控制系统是由单片机控制的 DC-DC 电压转换系统，可以将 20V~30V 的直流输入电压转换成 5V~10V 的稳定的直流电压输出。该系统主要由 DC-DC 开关电源子系统及外部控制电路组成。外部控制电路由单片机小系统、电压控制子系统和电压检测子系统三个部分组成。单片机子系统输出 PWM 波，经电压控制子系统处理，通过光电耦合器件 4n25 控制开关电源子系统的输出电压；电压测量子系统对开关电源子系统的输出电压进行测量转化并反馈给单片机形成闭环控制系统。图 1 为降压型 DC-DC 开关电源系统框架图。

2.1.2 DC-DC 开关电源子系统

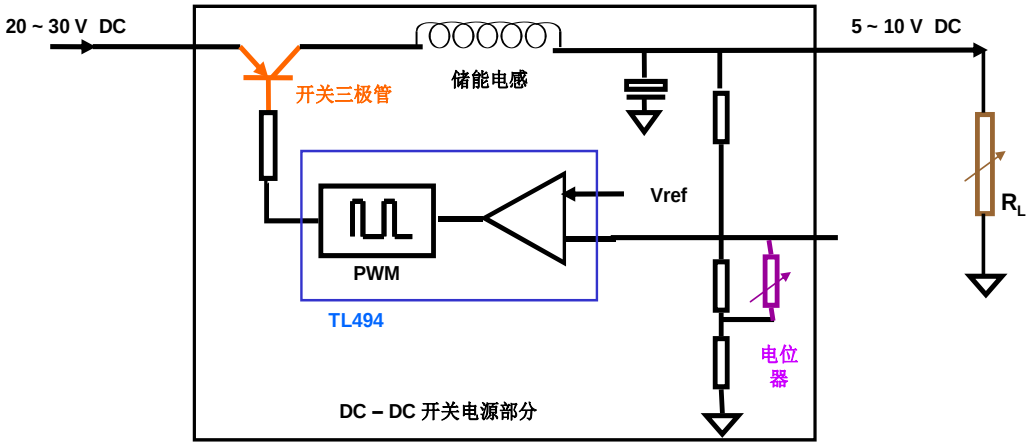


图 2.2 DC-DC 开关电源部分结构图[1]

本系统是以 TL494 为 PWM 控制器的 DC-DC 开关电源部分，在一块印刷电路板上实现 DC－DC 变换器的功能，对于 20V~30V 之间波动的电压输入，经过变换器得到一个稳定的输出电压。此稳定的输出电压值可以在 5V～10V 的范围内手动调节。

2.1.3 电压控制子系统

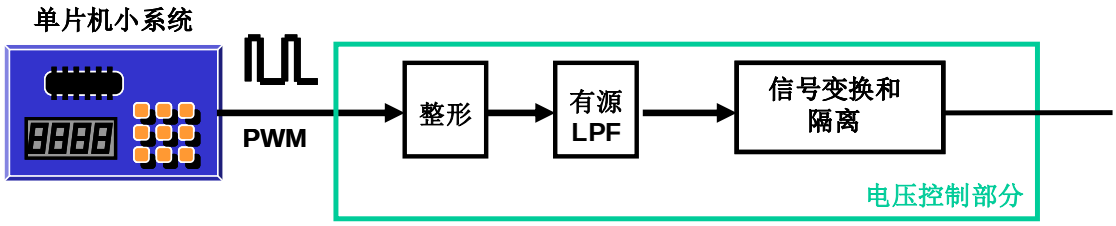


图 2.3 电压控制部分结构图[1]

电压控制部分主要通过单片机改变 PWM 信号的占空比来改变输出电压，使输出能达到 5V~10V，从而达到开环控制的效果，其间要经过整形和低通滤波的整形和滤波，然后再经过信号变换输入到 4N 25 光耦合器件。

2.1.4 电压检测子系统

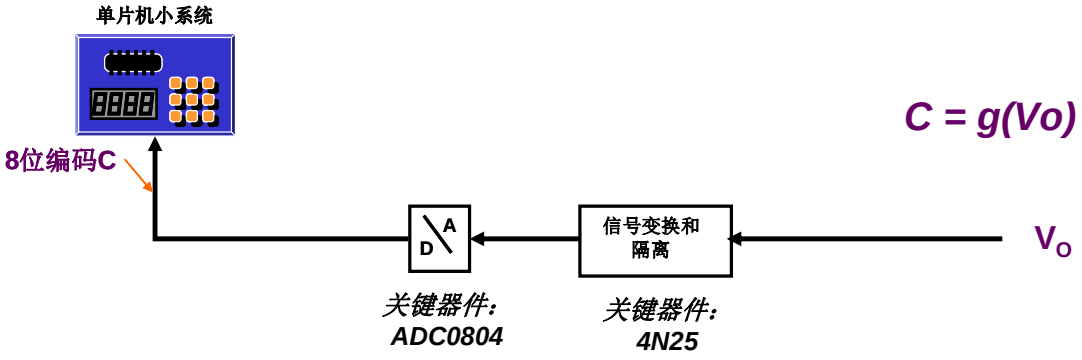


图 2.4 电压检测部分结构图[1]

这部分系统采集开环输出电压，然后输入给 ADC0804 进行编码，然后将 8 位二进制编码输入给单片机并有其控制闭环输出，从而实现整个系统的闭环控制和监测。

2.1.5 单片机子系统

单片机子系统提供 PWM 信号，利用输出 PWM 占空比的变化来控制输出的电压值，并且同时在其数码管显示部分显示输出电压值。

2.2 系统的主要功能

本次实验目标是实现降压型 DC-DC 开关电源，其主要功能如下：

1．降压型 DC-DC 开关电源：以 TL494 芯片为核心为核心的电路，将输入为 20-30V 直流电源，输出为 5-10V 可调直流输出，手动调节电压输出：降压型开关电源子系统在单独工作的方式下，可通过手动调节可调电阻阻值，实现在输入电压 20~30V 范围内，输出 5~10V 稳定的直流电压。

2．开环控制：在输入电压 20~30V 范围内，人通过键盘给单片机信号，单片机直接从程序内部读取对应的参数，输出控制信号（PWM）以得到所需要的 5~10V 输出电压。

3．闭环控制：在输入电压 20~30V 范围内，人通过键盘给单片机信号，单片机不断读入输出电压的采样信号并与给定值比较，再不断改变输出控制信号 PWM 的占空比，开关电源的输出电压更为准确于稳定，输出用户通过按键指定的 5~10V 电压。

4. 键盘与显示：键盘是人机界面的输入部分，即人通过按动键盘来调整输出电压的大小。

3. DC-DC 开关稳压电源模块的设计

3.1 主要功能与设计指标

3.1.1 主要功能

降压型 DC-DC 开关电源子系统的主要功能为将输入的不稳定 20~30V 直流电压变换为 5~10V 稳定可调的直流电压输出。其设计指标如表 3.1 所示。

表 3.1 降压型 DC-DC 开关电源子系统设计指标[1]

项目	指标
输入直流电压	20V~30V
输出直流电压	5V~10V
限流值	1.1A
电压调整率	＜0.5 %
电流调整率	1%
输出电压纹波	≤100mVp-p
效率	≥65%

3.1.2 基本设计原理

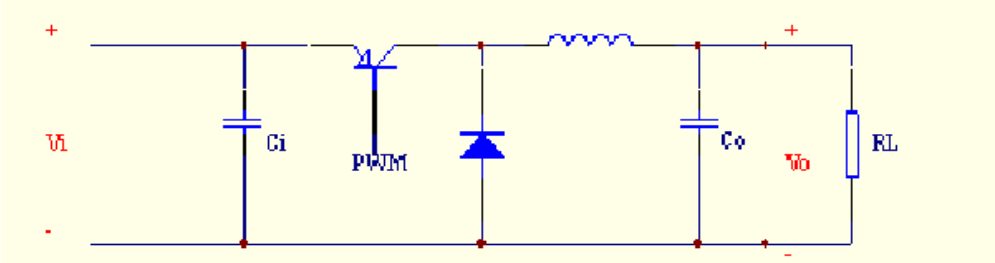


图 3.1 降压型 DC-DC 开关电源原理电路图

该子系统的等效电路如图 3.1 所示。系统通过 TL494 由比较基准参考电压和输出反馈电压，产生的 PWM 波来控制开关三极管 TIP42 的通断，使电路处于导通和断路两种工作状态。其波形如图 3.2 所示。

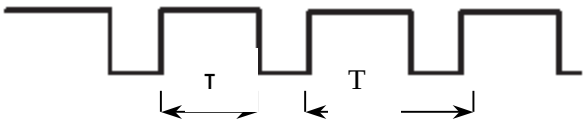


图 3.2 TL494 输出的 PWM 波形

图中 TL494 芯片、开关三极管（TIP42），基极电阻构成一个可控关闭与断开时间的开关电路。后级充放电路则由续流二极管 VD，储能电感 L，滤波电容 C 等构成。TL494 芯片通过 1 号管脚对 Vs，即输出电压 Vo 的线性分压，进行采样，通过和其内部的参考电压 Vref1进行比较后，使得开关三极管 TIP42 处于截止或导通状态。PWM 为低电平时，三极管导通，对电感进行充电，负载电

注：[1] TL494 提供了一个 5V 基准电压可用

压变大。当 PWM 为高电平时，三极管截止，电感放电，负载电压变小。TL494 芯片的 1 号管脚在不断地进行电压采样，当其采样电压低于 Vref 时，导通开关三极管。当其采样电压高于 Vref 时，断开三极管。因为 Vref 是一个固定不变的比较电压，那么我们只要调节分压电阻就可达到在某个范围内改变及稳定 Vo 的目的。这就是降压型开关电源的工作原理。由于充放电过程需要一定的时间，所以输出端 Vo 处的波形为锯齿波，它的峰峰值的大小取决于 TL494 的开关频率。占空比 $\eta=\tau/T$ 和频率由 TL494 及其外围电路的接法控制。

3.1.3 主要电路和参数设计

3.1.3.1 TL494 简介

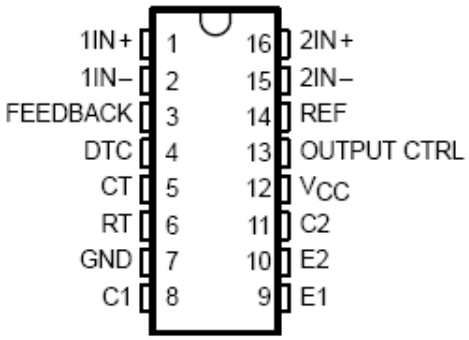


图 3.3 TL494 管脚图[2]

1. [1IN+] 第一个运放的正输入端 12
2. [1IN-] 第一个运放的负输入端
3. [FEEDBACK] 运放的反馈端
4. [DTC] 死区时间控制端
5. [CT] 内部振荡器接电容端
6. [RT] 内部振荡器接电阻端
7. [GND] 接地端
8. [C1] 第一个输出三极管的集电极
9. [E1] 第一个输出三极管的发射极
10. [E2] 第二个输出三极管的发射极
11. [C2] 第二个输出三极管的集电极
12. [Vcc] 工作电源输入端
13. [OUTPUT CTRL] 输出控制端
14. [REF] 基准电压输出端
15. [2IN-] 第二个运放的正输入端
16. [2IN+] 第二个运放的负输入端

TL494 的内部结构与工作原理如图 3.4 所示。V_{ref} 经过分压后从 1IN-端输入误差放大器 1（Error Amplifier 1），输出电压的采样值从 1IN+端输入。误差放大器 1 将两者进行比较，当采样电压大于基准电压时，其输出端对 V_A 进行充电，使 V_A 的电压不断升高；当采样电压小于基准电压时，V_A 通过恒流源放电，从而 V_A 的电压不断下降。而 V_A 的变化会引起 PWM 比较器（PWM Comparator）判决门限的改变，使得 PWM 比较器输出占空比动态改变的 PWM 波。

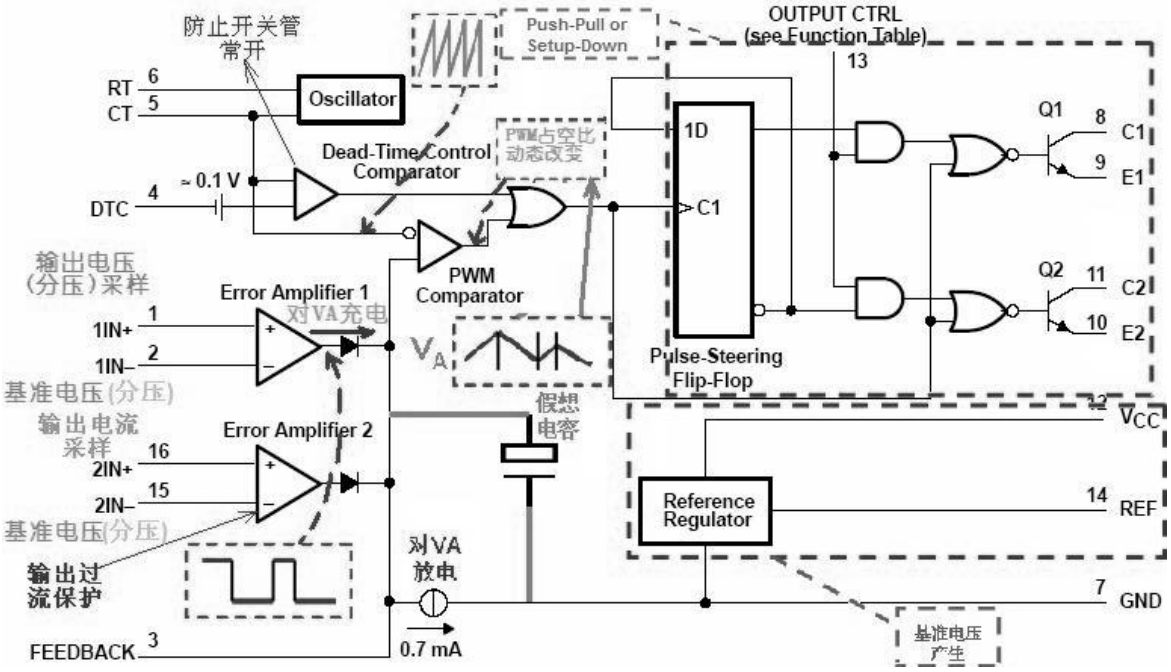


图 3.4 TL494 内部结构图[1]

TL494 还另外引入了两个关断条件。一是 DTC 端口连接的 Dead-Time Control Comparator，可防止开关管常开，保证在 1 个开关周期里至少有 Dead-Time 时间是关断的。另一个是误差放大器 2，它能起到输出过流保护的作用，输出电流采样值从 2IN+端输入，原理与误差放大器 1 类似。

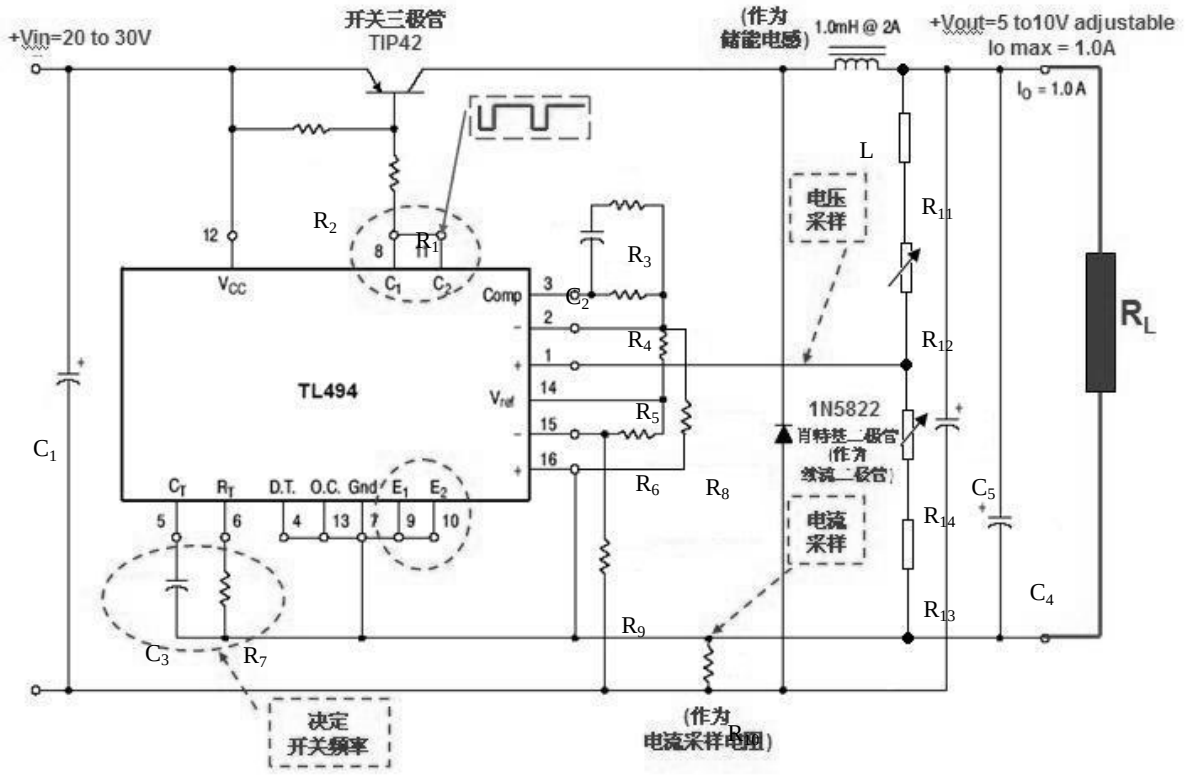


图 3.5 降压型 DC-DC 开关电源子系统电路图[1]

3.1.3.3 元件参数列表

表 3.2 降压型开关子系统元件参数表

元件	元件参数	元件	元件参数	元件	元件参数
C_1	100μF	R_3	47kΩ	R_{10}	0.1Ω
C_2	0.1μF	R_4	1MΩ	R_{11}	5.1kΩ
C_3	1000pF	R_5	6.8kΩ	R_{12}	0~22kΩ
C_4	470μF	R_6	5.1kΩ	R_{13}	5.1kΩ
C_5	100μF	R_7	6.8kΩ	R_{14}	0~1kΩ
R_1	390Ω	R_8	5.1kΩ	R_L	10Ω
R_2	51Ω	R_9	120Ω	L	1mH

3.1.3.4 元件参数设计

(1) 工作频率的确定（C_3、R_7）

开关电源的工作频率（PWM 波的频率，即三极管开、关的频率）是由 TL494 芯片 5 号管脚上的电容 C_3 和 6 号管脚上的电阻 R_7 决定的，其关系为：

$$f = \frac{1}{C_3 \cdot R_7}$$

增大工作频率使充放电时间变短可减小纹波幅度，然而，频率高了之后三极管消耗功率变大，会降低效率。经过反复试验，最终确定工作频率为

$$f=\frac{1}{C_3\cdot R_7}=\frac{1}{1000\,pF\times 6.8k\Omega}=147.059kHz$$

(2) 三极管基极电阻 R_1、R_2 的选取

适当增加与三极管基极相连的电阻 R_1、R_2 的阻值，可降低开关管饱和导通深度，降低开关状态切换速率，可减小在开关瞬间电感漏感产生的涡流引起的开关瞬间噪声，即减小纹波的毛刺，但是矛盾的是 R_1、R_2 的增大会导致系统效率的下降。因此应该综合考虑毛刺以及效率两方面因素来确定 R_1、R_2 的取值。试验中，只能通过多次尝试决定。

(3) 输出电压采样网络的设计

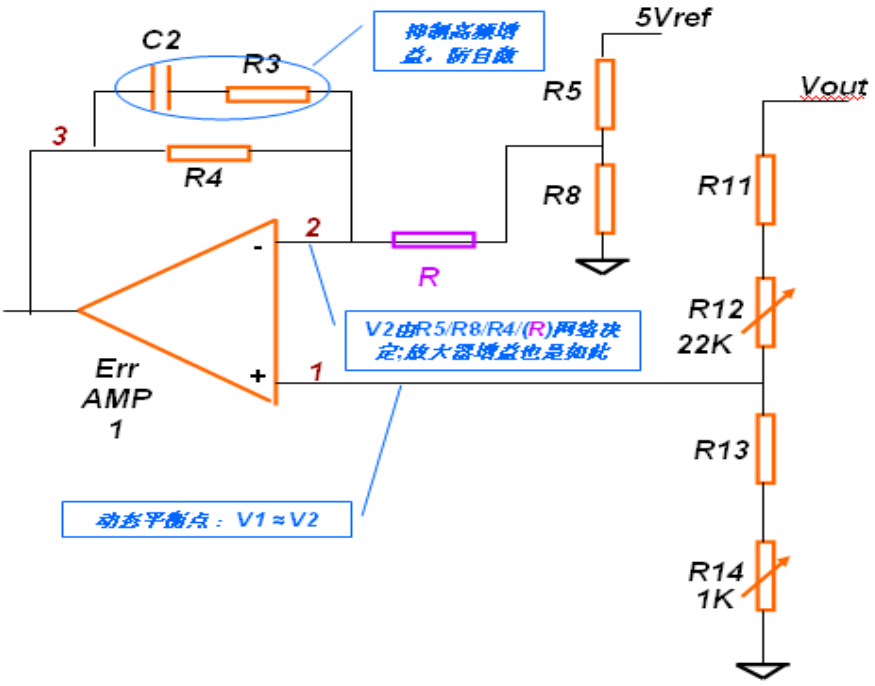


图3.6 误差放大器1外围电路分析[1]

由图 3.6 可以看到输出电压 V_{out} 经 R_{11}、R_{12}、R_{13} 以及 R_{14} 组成的串联分压网络分压后，从误差放大器 1 同相输入端输入，并与 V_{ref} 经 R_5、R_8 的分压进行比较，由于放大器两输入端电势差约等于 0，所以得到如下等式：

$$V_{ref}\times\frac{R_8}{R_5+R_8}=V_{out}\times\frac{R_{13}+R_{14}}{R_{11}+R_{12}+R_{13}+R_{14}}$$

本次实验中电阻 R_{11} 和 R_{13} 给定为 5.1kΩ。

(4) 误差放大器反馈网络的原理

误差放大器反馈网络的参数中，R_4 决定误差放大器的增益倍数，从而对电压调整率造成影响，而 C_2 和 R_3 组成的串联支路可抑制高频增益，防止误差放大器产生自激振荡。关于参数的计算并没有较好的方法，只能在实验中不断尝试。

(5) 电流采样电阻网络的选取：

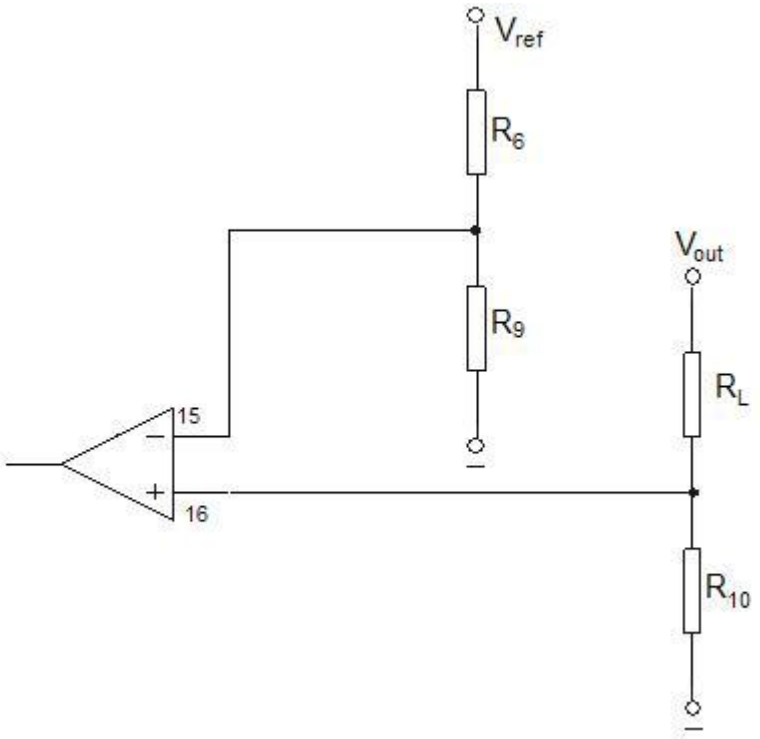


图3.7 限流保护电路原理图

图3.7说明了系统限流保护的原理，若误差放大器的反相输入端电压值大于或等于正相输入端电压值，系统将进行输出过流保护。设限流值为I，则电阻网络应满足：

$$I \times R_{10} = V_{ref} \times \frac{R_9}{R_6 + R_9}$$

其中I=1.1A， R_{10}=0.1Ω，计算可得R_6，R_9的电阻值。

3.1.4 纹波讨论

3.1.4.1 纹波产生主要原因

（1）50Hz频电残留的波动，开关的频率越小，残留的越小，又由于实验中使用稳压源提供，可以忽略。另外在输入端接有C1滤波电容，可进一步减小。这时候的C1为电解电容，取值为470uF。

（2）开关噪声，在开与关的瞬间，由于电感的漏感所产生的磁涡流和开关响应的延迟造成尖峰装的衰减振荡，开关速度愈大，磁涡流强度越大，即纹波中观察到的毛刺现象。

3.1.4.2 纹波的抑制

抑制纹波可采取一下几种措施：

(1) 由于储能电感量 L 越大，纹波越小。所以为了降低纹波,设计选用饱和电流比较大的电感。因为当磁芯接近饱和时损耗增大,会降低转换效率,所以电感的饱和电流应大于回路中的峰值电流。

(2) 增大开关频率。此方法在一定程度上可以使三极管开关切换更加频繁，从而显著降低纹波峰峰值。但若三极管开关过于频繁，就会使得效率显著下降，TL494 放出大量热量。

(3) 减小开关管饱和导通深度。适当增大R_1，可以降低饱和导通深度，但也会降低效率。

(4) 增大滤波电容C_5，但是效果不明显，不太适合。

3.1.4.3 效率的提高

由上述分析可知，效率与频率是一对相互制约的矛盾量，因此抑制纹波的元件参数在对效率的调节也起着至关重要的作用。将 3.1.4.1 节所述措施(1)、(2)、(3)的相关参数向相反的方向调节就可提高效率。

4. 用单片机控制 DC-DC 开关电源输出电压

4.1 单片机控制系统整体方案

4.1.1 ATmega16 简介

本实验所采用的单片机是 ATmega16。这款 8 位 AVR 微处理器的特点是高性能、低功耗。我们用到了输入输出端口，ADC 采样模块，定时/计数/PWM 控制器资源。

4.1.2 单片机资源配置

4.1.2.1 输入输出口资源配置

AVR ATmega16 单片机共有 4 个 8 位可编程 I/O 端口，分别为 PA0-7，PB0-7，PC0-7，PD0-7。对于每个端口都有三个 I/O 存储器地址：数据寄存器-PORTx、数据方向寄存器-DDRx 和端口输入引脚-PINx。PORTx 和-DDRx 为读/写寄存器，-PINx 为只读寄存器。每个端口可以作为独立的输入输出来使用，也可根据不同的设计要求，进行管脚复用或作为特殊模块的接口。

在本次的课程中，我们使用 PA0-3 作为 ADC 采样模块的输入，PD4，PD5 作为 PWM 波的输出端口，PC4-7 为按键的检测输入口。PA5-7 作为串行输出的数据输出和控制信号来控制数码管的显示。

4.1.2.2 ADC 模块配置

ATmega16 有一个 10 位的逐次逼近型 ADC。ADC 与一个 8 通道的模拟多路复用器连接，能对来自端口 A 的 8 路单端输入电压进行采样。

在每次 ADC 采样转换结束后，转换结果被存入 ADC 结果寄存器(ADCL, ADCH)。单次转换的结果如下：

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}}$$

式中，VIN 为被选中引脚的输入电压，VREF 为参考电压(参见 P203Table 83 与 P204Table84)。 0x000 代表模拟地电平， 0x3FF 代表所选参考电压的数值减去 1LSB。如果使用差分通道，结果是：

$$ADC = \frac{(V_{POS} - V_{NEG}) \cdot GAIN \cdot 512}{V_{REF}}$$

式中，VPOS 为输入引脚正电压，VNEG 为输入引脚负电压，GAIN 为选定的增益因子，且 VREF 为参考电压。结果用 2 的补码形式表示，从 0x200 (-512d) 到 0x1FF (+511d) 。ADC 模块的控制可由控制寄存器 ADMUX 和 ADCSRA 实现，寄存器如下图：

Bit	7	6	5	4	3	2	1	0	
	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
读 / 写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
读 / 写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
初始值	0	0	0	0	0	0	0	0	

图 4.1 控制寄存器 ADMUX 与 ADCSRA 示意图[1]

在本次课程中，对电压采样采用了差分输入，PA0 口为电压差分输入端，PA1 口为比较电压输入端，大小为 3.75V。因为端口个数的限制，无法满足采样的需求，所以我们采用单端输入，输入为经过运放放大的小电压信号。输入端口分别为 AD2，AD3。

4.1.2.3 定时/计数/PWM 控制器资源配置

定时器/计数器 0 产生 5ms 中断，供程序调用中断服务子程序。定时器/计数器 1 产生占空比可调的 PWM 信号，由 I/O 端口 PD4，PD5 输出，并分别控制模块 B，和模块 A。

4.1.2.4 其他资源配置

按键 1： 电压值+1；

按键 2：电压值+0.1；

按键 3：电压值-0.1；

按键 4：开闭环切换；

4.1.3 数码管显示

第一位：开闭换标志，后三位：显示数值，电压或占空比，或内部某个变量，可认为设定。

4. 2 开环控制系统的硬件电路

4.2.1 功能与指标说明

电压控制模块主要实现的功能是将单片机输出的可调占空比 PWM 波进行整形、滤波的方式将占空比的变化转化为电压的变化，最后通过光耦芯片 4N25 和 DC-DC 开关稳压电源模块连接。该模块的设计指标是输出电压误差绝对值≤0.05V。

4.2.2 电路选型说明：

方案一：直接利用单片机输出的PWM波信号控制4N25，进而控制调控电阻的阻值以实现电压调节的目的。这种方法属于直接连接，电路复杂性基本为零，实现较为简单。但缺点也显而易见，由于单片机系统提供的PWM信号约为5V的信号，会随着供电系统的变化而变化，所以控制信号不够稳定。

方案二：利用无源滤波电路，针对直接连接单片机输出信号与调控电阻的元件4N25之间的电路，进行滤波，这种方法电路只需要电阻电容即可实现滤波的目的。但也存在并没有解决PWM高电平不恒定，且与电源电压有关的缺陷。

方案三：使用有源滤波电路。利用CD4011反相器，并利用TL431提供基准电源电路，对单片机输出的PWM信号先用CD4011进行整形，将高电平不稳定且与电源电压有关的PWM波，整形为，高电平稳定且与电源电压无关的信号，之后再利用ua741进行有源滤波。由于滤波相对于RC无源滤波有如下优点：输出阻抗小，可以带比较大的后继负载，别把负载看成高阻；需要的话还可以进行放大。其次，由于滤波器阻带衰减在60dB左右，因此一阶滤波器不在选择范围之内；另一方面二阶以上的滤波器实现难度较大，而且通频带变窄。综上考虑选择二阶滤波器。而Sallen-Key二阶低通滤波器的优点如下：电路结构简单、通带增益、极点角频率和品质因数表达式简介，且品质因数可调范围大，调节方便。故实验中采用sallen-Key二阶低通滤波器。

综上所述，我们决定采用方案三整形电路加有源滤波电路来实现电压控制模块控制目的。

4.2.3 主要组成部分

1）有源低通滤波器：单片机输出占空比经过有源低通滤波器，滤除高频成分，只剩下直流成分，并加到光耦一端。

2）光耦电路：主要起到光电隔离的作用。

4.2.4 系统原理图设计

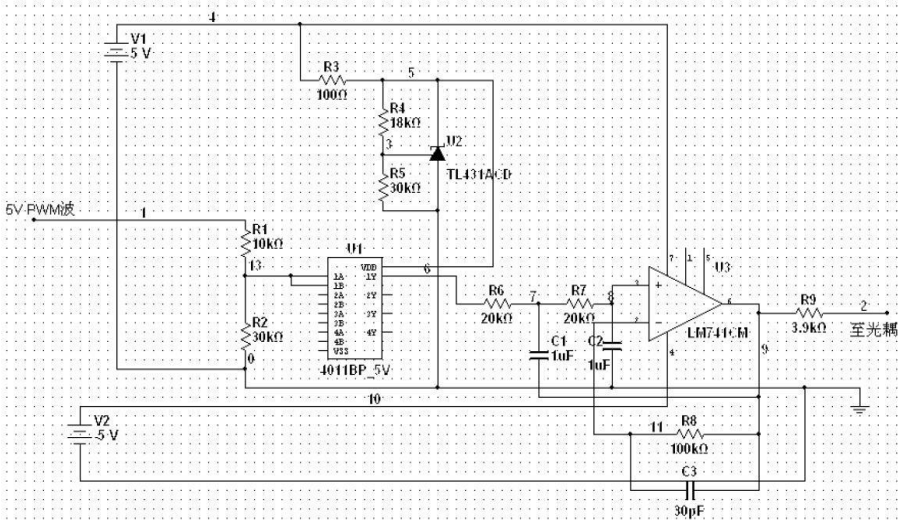


图 4.2 电压控制模块电路图

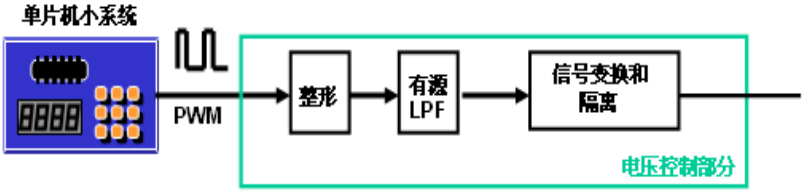


图4.3用电信号控制输出电压的原理[1]

4.2.5 基本设计原理

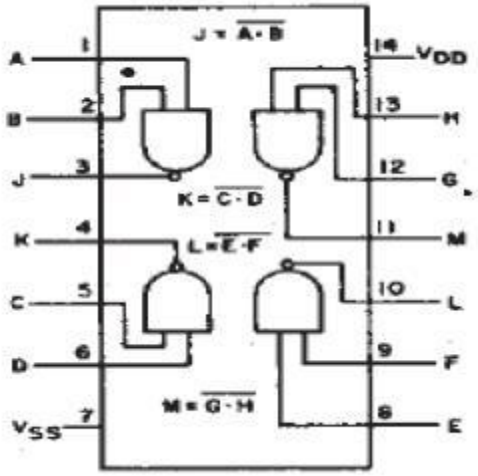
该子系统由基准电源、整形、有源低通滤波与信号隔离变换 4 个模块组成。其关系如图 4.1 所示。基准电源模块对 5V 电源进行稳压并分压，输出 4V 稳定电压作为整型电路的稳压源，整型电路将单片机输出的不稳定的 PWM 信号整型，有源低通滤波模块将起转化为与占空比成正比的直流电流，通过信号隔离变换模块与 DC-DC 开关电源子系统连接，达到控制的目的。

4.2.6 专用芯片介绍

4.2.6.1 CD4011 简介

该芯片的管脚定义以及功能结构如图所示。CD4011B 是一块与非门电路芯片，当 VDD 一定时候，只要输入的电压高于芯片的阈值电压，则输出高电平为稳定的 VDD 电压。单片机输出的 PWM 波高电平不稳定，经过 CD4011B 之后，可以形成高电平稳定的 PWM 波，达到整波的目的。值得注意的是 PWM 波输入的电压高电平为 5V，而 TL431 电路提供的 VDD 为 4V，而根据 CD4011B 的

Datasheet，CD4011B 的极限输入电压上限是 VDD + 0.5，故需要对输入的 PWM 波进行分压后再输入到 CD4011B 芯片当中。



4.4 CD4011B 功能结构图[3]

4.2.6.2 UA741 简介

uA741（单运放）是高增益运算放大器，用于军事，工业和商业应用.这类单片硅集成电路器件提供输出短路保护和闭锁自由运作。 这些类型还具有广泛的共同模式，差模信号范围和低失调电压调零能力与使用适当的电位。从 UA741 的 datasheet 上可知，其正常工作电压为 15V ，最高工作电压为 22V ，但实际电路应用中提供工作电压为 5V ，所以其输出的电压绝对值应小于 5V，实际测量发现为近 4V 的位置。

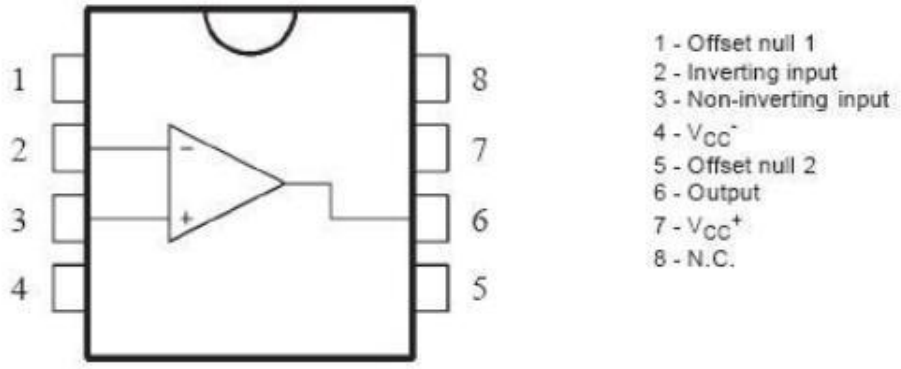


图 4.5UA741 管脚说明图[4]

4.2.6.3 TL431 简介

TL431 是一个有良好的热稳定性能的三端可调分流基准源。它的输出电压用两个电阻就可以任意地设置到从 Vref（2.5V）到 36V 范围内的任何值。该器件的典型动态阻抗为 0.2Ω，在很多应用中可以用它代替齐纳二极管，例如，数字电压表，运放电路、可调压电源，开关电源等等。图所示

的是 TL431 的管脚图、功能结构图以及实际结构图。TL431 的三个引脚分别为参考（Reference），阳极（Anode）,阴极（Cathode）。Vref 是一个内部的 2.5V 基准源，接在运放的反相输入端。由运放的特性可知，只有当 REF 端（同相端）的电压非常接近 Vref（2.5V）时，三极管中才会有一个稳定的非饱和电流通过，而且随着 REF 端电压的微小变化，通过三极管的电流将从 1 到 100mA 变化。

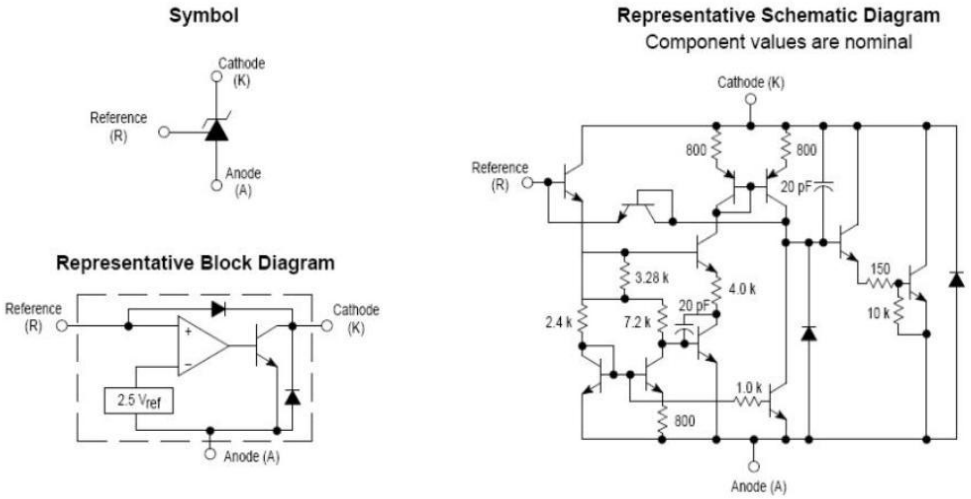


图 4.6　TL431 管脚、功能及结构图[1]

4.2.7 具体参数计算与说明

4.2.7.1 基准源电路模块：

外围电路参数计算

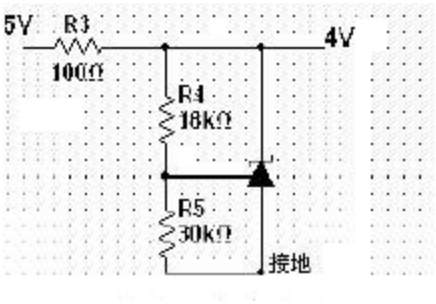


图 4.7 基准电压模块设计图[1]

电路参数的选择：

由于稳压管反向电流等有最低要求不能过小，查看 datasheet 可知大致为 3-7mA，而基准源需要提供的电压为 4V，所以整形模块 R_3<(5V-4V)/7mA=150 欧。所以在此处我们选取 R3=100Ω的电阻。考虑到电流大小、实验是提供的电阻等因素，我们选用 R4，R5 的取值分别取 18KΩ，30KΩ。

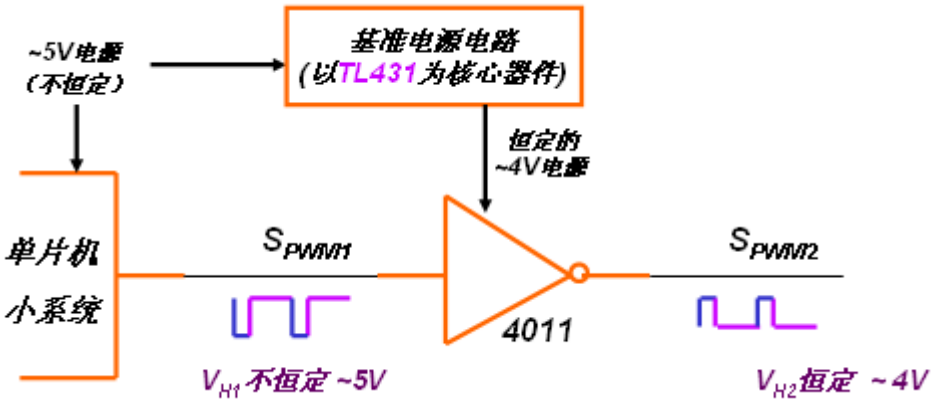


图 4.8 整型模块设计框图[1]

CD4011 是与非门元件，以稳定的 4V 电压为工作电源，对输入信号进行整型。R_1、R_2 电阻网络对单片机子系统 P1.6 引脚输出的 PWM 信号进行分压，使高电平电压降至 4V，故满足关系式：

$$\frac{R_2}{R_1+R_2}=\frac{4}{5}$$

4.2.7.2 有源低通滤波模块

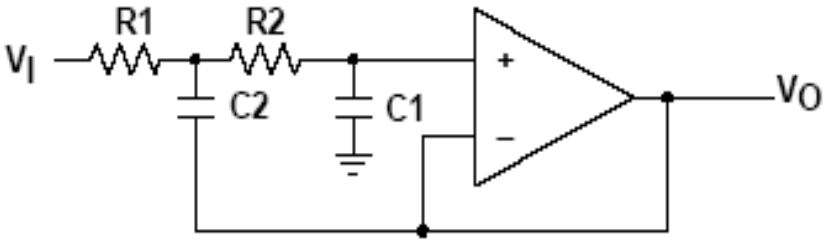


图 4.9 有源低通滤波模块

由于无源滤波器的通带放大倍数及其截止频率都随负载的变化而变化，不能满足实验的要求，因此本次实验采用了有源滤波器。图 4.6 为 Sallen Key 二阶滤波器的典型电路，其截止频率为 50Hz。因为单片机输出的 PWM 信号频率为 500Hz，远大于截止频率，故 CD4011 输出的稳定 PWM 波通过该滤波器之后只剩下直流分量，其值约为 PWM 波高电平和占空比的乘积。

如图 4.6 R_1、R_2、C_1 及 C_2 决定了滤波器的截至频率，一般取 $R_1=R_2$，$C_1=C_2$，可由下式求得：

$$f_c=\frac{1}{\sqrt{R_1R_2C_1C_2}}=50Hz$$

4.2.7.3 信号隔离变换模块

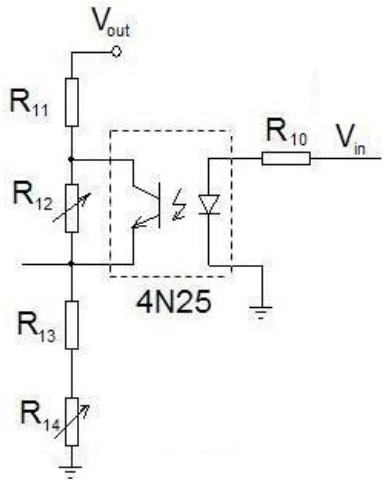


图 4.10 信号隔离变换模块[1]

该模块以光电耦合器 4N25 为核心元件，它相当于一个流控电流源，实现了电压控制子系统与 DC-DC 开关电源子系统的信号变换传递。如图 4.7，4N25 将右侧的电压信号 V_{in} 传递到左侧负载电路，对 R12 上流过的电流进行了分流，从而改变输出电压，同时，右侧电压控制子系统的工作不受左侧负载阻抗变化的影响，实现了电气隔离。

4.3 开环控制系统的软件控制

开环的控制主要通过每过 5ms 的中断程序来实现，首先重装计数器计数值。然后进行对四个按键的检测，并执行相应操作。并根据之前的用户控制信息，决定是否操作闭环控制模块，并适当调整 PWM 波占空比。最后根据相应数值进行数码管扫描。流程图如下：

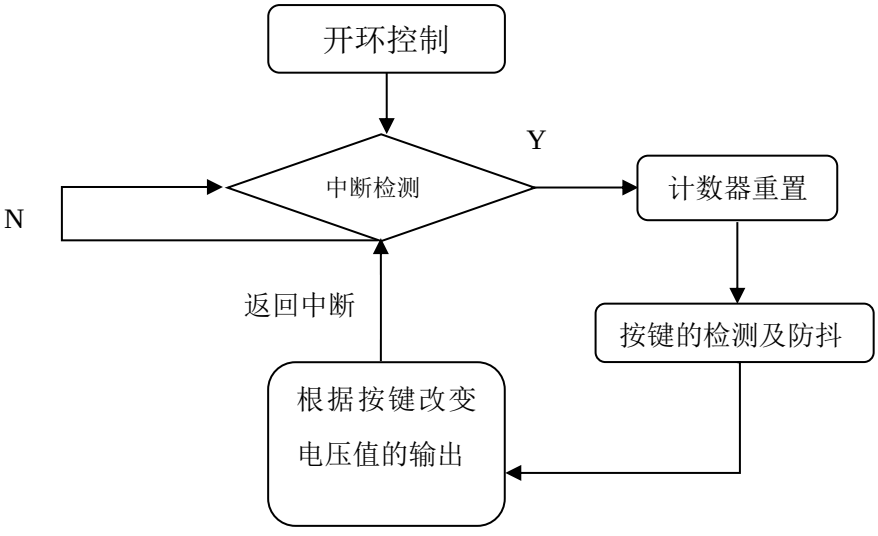


图 4.11 开环控制软件流程图

4.4 闭环控制系统的硬件电路

4.4.1 总述

闭环即在开环的基础上增加反馈环节。系统通过对 DC-DC 开关电源子系统的输出电压进行采样，并转化为数字信号，与既定值比较，自动调节控制信号，使系统输出用户通过按键所指定的 5~10V 电压，实现系统的闭环控制。同时，对电压的采样有光耦合管进行光电隔离，光耦合管工作在近似线性的区域。与开环控制相比，闭环控制可适应 DC-DC 开关电源子系统工作状态在一定程度内的变化，如可调电阻阻值变化，工作温度变化等。

4.4.2 电路主要功能

1) 对 DC-DC 开关电源的输出进行 A/D 转换。
2) 对 DC-DC 开关电源的输出进行测量。
3） 将测量结果反馈给单片机系统。
4）输出电压误差绝对值≤0.05V。

4.4.3 基本设计原理

电压测量子系统分为基准电源、信号隔离变换及 ADC 三个模块，分别以 TL431、4N25 和 ADC0804 为核心。电压信号通过信号隔离变换模块的处理，输出给 ADC 模块进行模数转换，而基准电源模块则提供给 ADC 模块稳定的参考电压，使其能保持固定的转换关系。其中基准电源以及信号隔离变换模块与 4.3.2 节与 4.3.5 节相同，在此不再赘述。

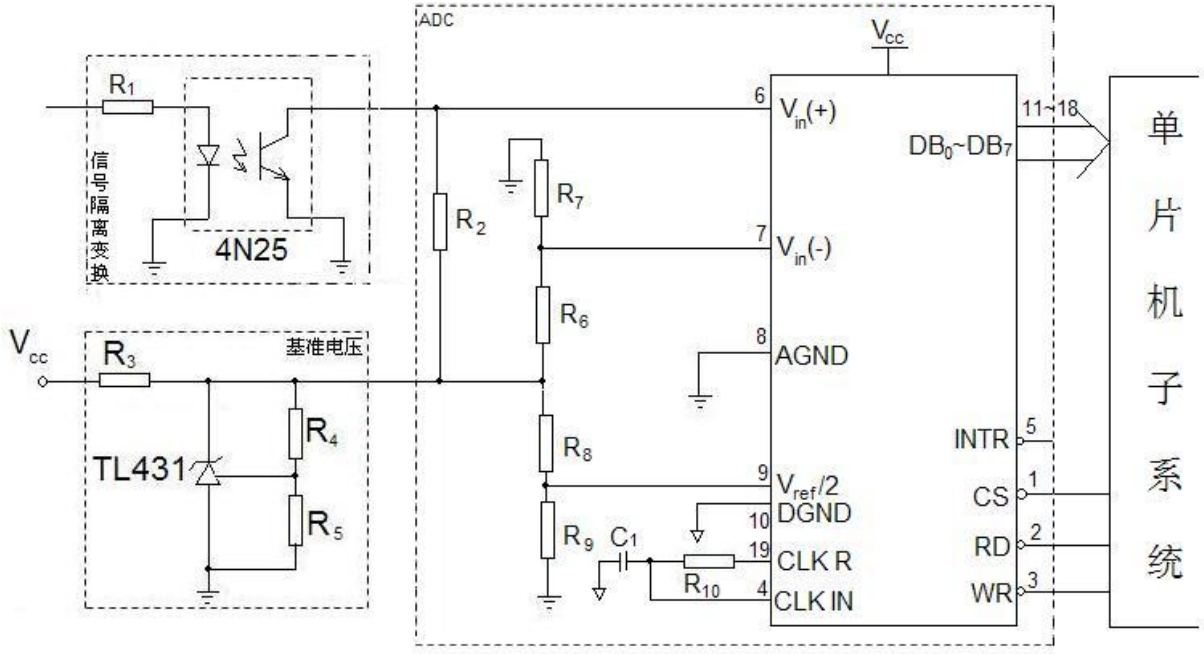


图 4.12 电压测量子系统设计图[5]

本系统信号变换仍然用的是 4N25 光耦合合器件，AD 转换使用的是 ADC0804 逐次比较型 AD 转换芯片。DC-DC 开关电源子系统的输出电压经过 4N25 的信号变换，得，ADC0804 对于 VA 进行量化、编码并输入到单片机，单片机根据程序中已有的输出电压对应到 VA 的编码值，对于输出占空比进行微调，实现对于电压的更精确的控制。

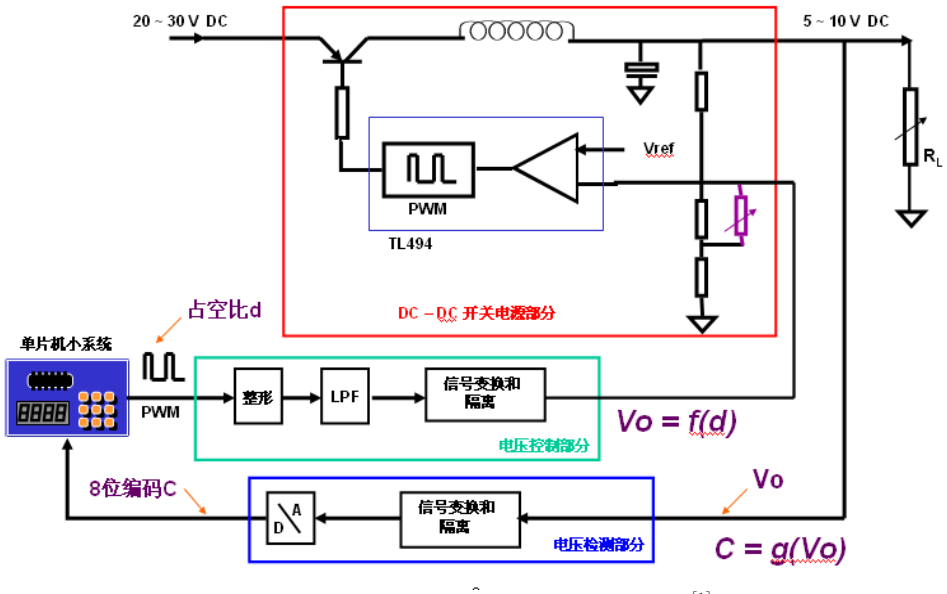


图4. 13电压检测子系统系统框图[1]

4. 5 闭环控制系统的软件控制

在闭环控制模块中，首先读取 ADC 采样值，采样时采取滑动平均计数，并且再开启一次 ADC 采样，对采样值与设定值进行比较，然后对实际 PWM 波进行调整。实现闭环负反馈，使精度进一步提高。下图，为闭环控制流程图。

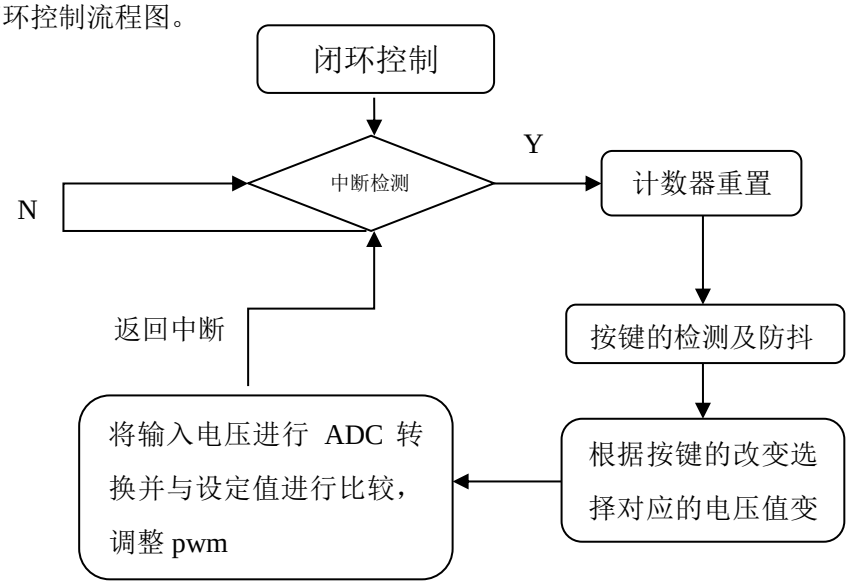


图 4.14 开环控制软件流程图

5. 用 MSP430 闭环控制 DC-DC 开关电源输出电压（高级拓展）

本实验中的高级拓展部分所使用的单片机的型号为 MSP430G2553。它是超低功耗混合信号微控制器　具有内置的 16 位定时器、多达 24 个支持触摸感测的 I/O 引脚、一个通用型模拟比较器以及采用通用串行通信接口的内置通信能力。此外 MSP430G2553 还具有一个 10 位模数(A/D)转换器。

5.1 MSP430 闭环控制系统整体方案

与基础部分的闭环控制系统基本相同。对输出电压进行检测并与以输入的设定值进行比较，从而对单片机输出的 PWM 波进行调整来实现闭环控制 DC-DC 开关电源输出电压。

5.1.1 资源分配情况

本实验中利用到了 MSP430 中的 10 位 ADC、TIMER 计时器以及 UART 串口通信模块。其设计思想与基本部分大致相同。区别在于，由于 MSP430G2553 所配置的按键有限，我们小组采用通过 URAT 串口与电脑进行通信的方式来控制电压值的改变。

5.1.2 单片机的管脚配置为：

P1.4:用于 ADC 单端输入，即负载电压分压后的电压；

P2.1:输出 PWM 波；

GND：接负载的负端；

5.2 MSP430 闭环控制系统的硬件电路

本小组的高级拓展所采用的硬件电路是在基础部分之外添加了一个分压电路。该分压电路的功能是对输出电压 U_o 进行分压，得到单片机的输入电压 U_i （$U_i = U_o/4$ ）。

该做法的原因在于 msp430G2553 的 10 位 ADC 的输入电压须在 3V 以下，故添加该分压电路。

5.3 MSP430 闭环控制系统的软件控制

5.3.1 软件设计

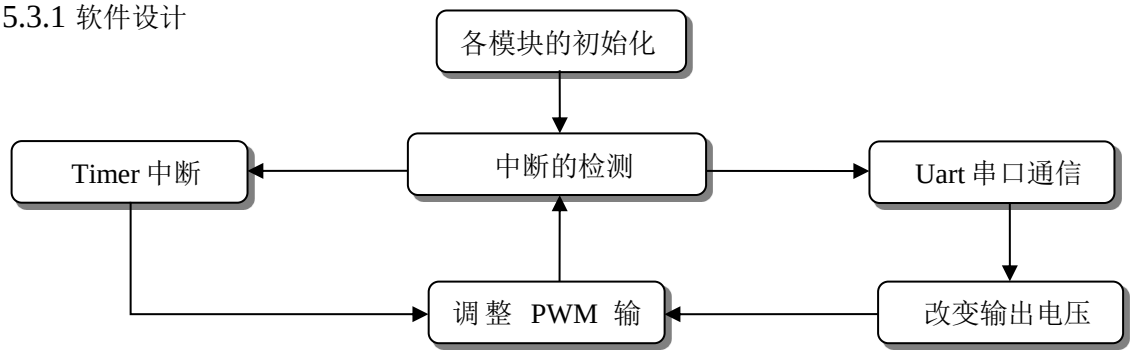


图 5.1 MSP430 程序流程图

5.3.2 两个重要的数组

int pwmv[53]：用于存储 4.9V ~ 10.1V 该 53 个电压对应的 pwm 波的调整值；

int ad[53]：用于存储 4.9V ~ 10.1V 该 53 个电压对应的 ADC 输出的标准对比值。通过将输入的电压进行 AD 转换并与该标准值对比从而改变输出的 pwm 波。

5.3.3 程序中所用到的函数及其作用

IOInit( )：IO 口的初始化；

Timer0Init( )：Timer0 的初始化；

Timer1Init( )：Timer1 的初始化；

uart_init( )：uart 的初始化；

ADCInit( )：ADC 的初始化；

__interrupt void USCI0RX_ISR( )：uart 串口 RX 接收到一个字符,产生中断处理；

valueInitial( )：程序中所用到的几个变量的初始化;

init_devices( ): 通过调用该函数将各个元件的初始化执行完毕；

__interrupt void Timer_A (void)：Timer 的中断函数；

Main（）：程序的主函数。

程序的设计思想与基本部分相同，并添加了人机交互界面。我们组的控制输入使用的是超级终端程序。数字键 1、2、3、4 对应的电压变化为：

表 5.1 数字输入与电压变化的对应关系

输入	电压变化/V
1	+0.1
2	-0.1
3	+1
4	-1

6. 致谢

感谢在本次实验中予以我们小组帮助的袁焱等指导老师！

感谢在本次课程中对我们提供过帮助的同学和学长！

感谢在中期和终期检测时为我们检测的老师以及检测时提供的建议和意见！

7. 参考文献

[1] 上海交大电子工程系. 工程实践与科技创新 3A 课程系列 讲义[EB/OL].ftp://202.120.39.248.

[2] TL494 datasheet.

[3] CD4011B datasheet.

[4] UA741 datasheet

[5] 百度 www.baidu.com

8. 附录 A　系统操作说明书

8.1 基本部分

8.1.1 系统按键说明

按键一：电压设定值增加键，按下设定值加 1V ,调整范围为 4.9~10.1V。当大于 9.1 时，按此键，则会返回最初的低电压值，如图 8.1 所示

图 8.1 按键一说明图

按键二：电压设定值增加键，按下电压设定值加 0.1，调整范围为 4.9~10.1V。当加到 10.1 时，继续加 0.1 会返回 4.9。如图 8.2 所示

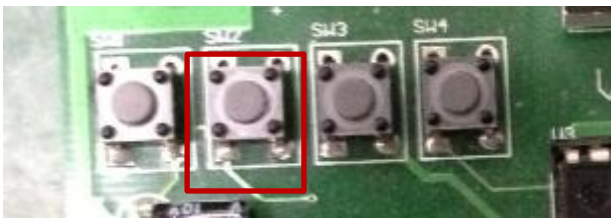


图 8.2 按键二说明图

按键三：电压设定值减小键，按下电压设定值减 0.1，调整范围为 4.9~10.1V。当减到 4.9 时，继续减 0.1 会返回 10.1。如图 8.3 所示

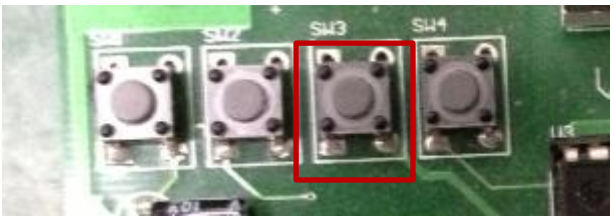


图 8.3 按键三说明图

按键四：开环闭环控制键，按下进行开环闭环模式切换 。如图 8.4 所示

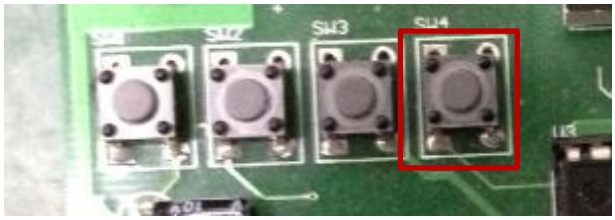


图 8.4 按键四说明图

8.1.2 系统显示说明

系统显示由四个数码管组成。

最左面的数码管是状态位，开环控制时显示”R”，闭环控制时显示”b”。 如图 8.5 所示。

图 8.5 开闭环标示数码管示意图

后三位数码管显示相应的数值，显示电压时，用后两位显示，并在中间显示小数点，显示占空比时，用全部的三位显示转换为百分比的占空比。如图8. 6所示。

图8. 6数值显示数码管示意图

8. 2 高级拓展

如表 5.1 中所述，数字键用于电压的控制。下图即为超级终端操作界面。

图 8.7 高级拓展人机交互界面

8. 3 系统整体展示

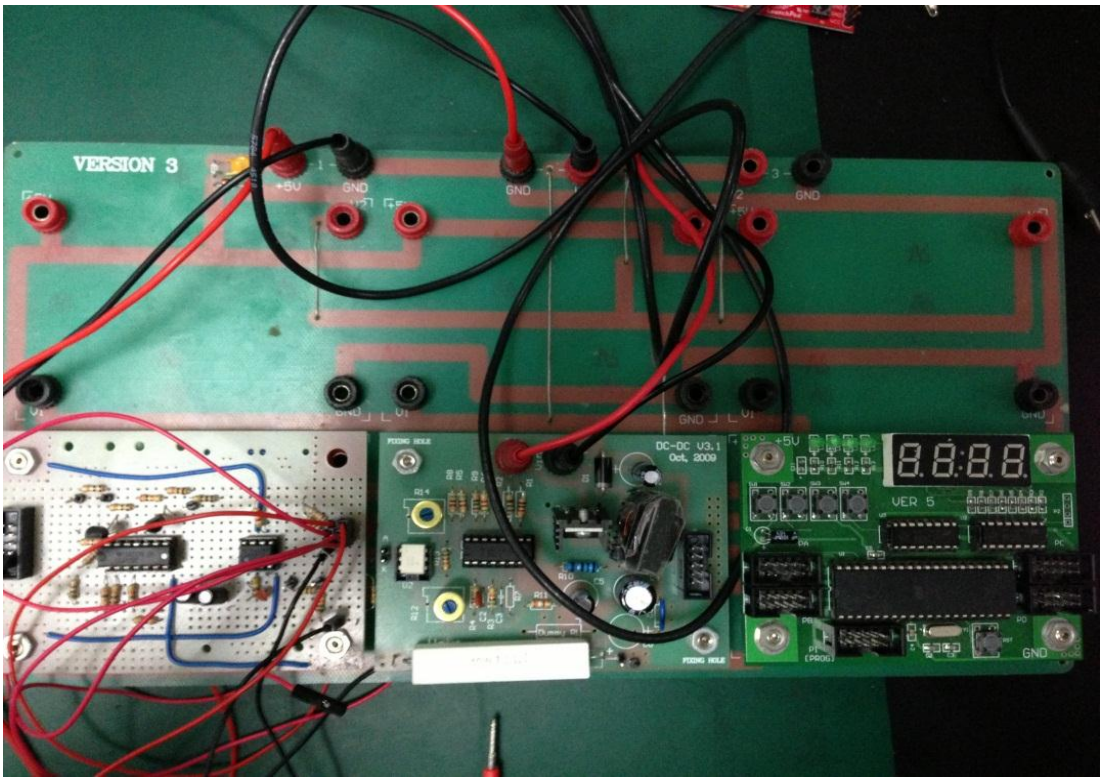


图8.8 系统整体示意图

9. 附录 B　测试和分析

9.1 测试项目和方法

1．DC-DC 模块独立测试

测试目的：检验开关电源电路的功能和达标情况。

表 9.1 DC-DC 模块独立测试和评分[1]

项 目		评分说明	测试方法或条件
输出电压可调范围（5 分）	最低输出 ≤ 5.0V； 最高输出 ≥ 10.0V	上下限之一不达标，扣 2 分；上下限均每不达标，扣 5 分	输入 30.0V±0.1V 人工调整相应电位器
输出纹波（5 分）	小于等于 100mV	超过限值扣 2 分；每高过限值 20mV，多扣 1 分	输入 30.0V±0.1V；输出 10.0V±0.1V；TDS 系列 60 或 100MHz 带宽示波器，探头 X10，Y 向 20 或 50mV 档
效率（5 分）	大于 65%	低于限值扣 2 分；每低过限值 5 个百分点，多扣 1 分	输入 30.0V±0.1V；输出 5.0V±0.1V；测量输入电压、输入电流、输出电压，负载以 10Ω计。
电压调整率（5 分）	小于 1%	超过限值扣 2 分；每高过限值 0.1 个百分点，多扣 1 分	输出 10.0V±0.1V；输入 20.0V±0.1V 到 30.0V±0.1V；
工艺（5 分）		满分要求：在各项指标良好的前提下，焊点匀称可靠，元件装列整齐，电感绕制做工良好	主观评价
特色记录（指标特别优异，或其他自创设计）		可自行提出，由老师验证并记录。	
注[1]：输出电压过大波动等异常情况，酌情扣分。对各种必须记录的状况，测试时不打分，事后综合评价。			

2．电压测控

测试目的：检验电路的功能和达标情况。

表 9.2 电压测控功能的测试和评分[1]

项 目		评分说明	测试方法或条件
开环控制精度（10 分）	输出=设点电压 1[注 2] ±0.05V	超过限值扣 2 分；每高过限值 0.05V，多扣 1 分	断开电压测量回路； 输入 30.0V±0.1V； 开始测试时，任意指定 3 个电压值分别作为设点电压
	输出=设点电压 2[注 2] ±0.05V	超过限值扣 2 分；每高过限值 0.05V，多扣 1 分	
	输出=设点电压 3[注 2] ±0.05V	超过限值扣 2 分；每高过限值 0.05V，多扣 1 分	
闭环控制绝对精度（10 分）	输出=设点电压 A[注 2] ±0.05V (实际输出记作 V1)	超过限值扣 2 分；每高过限值 0.05V，多扣 1 分	接入电压测量回路； 输入 30.0V±0.1V； 开始测试时，任意指定 3 个电压分别作为设点电压； 记录实际输出的 3 个电压值。
	输出=设点电压 B[注 2] ±0.05V (实际输出记作 V2)	超过限值扣 2 分；每高过限值 0.05V，多扣 1 分	
	输出=设点电压 C[注 2] ±0.05V (实际输出记作 V3)	超过限值扣 2 分；每高过限值 0.05V，多扣 1 分	

项 目		评分说明	测试方法或条件
电压自动调整的精度（10 分）	设点电压 A 输出=V1 ±0.05V	超过限值扣 2 分；每高过限值 0.05V，多扣 1 分	调偏 R14，重做指定电压的测试； 调偏后系统输出应在 30 秒内达到稳定。
	设点电压 B 输出=V2 ±0.05V	超过限值扣 2 分；每高过限值 0.05V，多扣 1 分	
	设点电压 C 输出=V3±0.05V	超过限值扣 2 分；每高过限值 0.05V，多扣 1 分	
工艺（5 分）		满分要求：在各项指标良好的前提下，焊点匀称可靠，元件装列整齐，走线清晰不杂乱	主观评价
特色记录（指标特别优异，或其他自创设计）		可自行提出，由老师验证并记录。	
注[1]：输出电压过大波动等异常情况，酌情扣分。对各种必须记录的状况，测试时不打分，事后综合评价。 注[2]：由老师当场指定			

9.2 测试的资源

测试设备：

1、 电压表

2、 示波器及探头

3、 稳压电源及电源引线 3 根，稳压电源要有 5V、 20V~30V 的输出

9.3 测试结果及分析

最终测试记录(以下结果均在老师监督下测得并填写)

表 9.3 最终测试记录表

DC-DC 模块独立测试			
测试条件		**项 目**	**记录**
输入电压	**输出电压**		
30.0V±0.1V	——	（5 分）输出电压可调范围 5.0V 至 10.0V	4.31~11.02V
20. 0V±0.1V 变化 30.0V±0.1V	设定 10.0V	（5 分）电压调整率 电压变化应小于 0.05V，记录输出电压	10.12V(In:30V) 10.108V
30.0V±0.1V	10.0V±0.1V	（5 分）输出纹波小于等于 100mV	100mV
30.0V±0.1V	5.0V±0.1V	（5 分）效率大于等于 65%，输入电流不大于 0.13A，记录输入电流	0.13A
——	——	（5 分）工艺	√
		其他情况记录（如选测限流项目等）	
用 ATMega16 单片机控制电源模块输出电压（测试前给予准备时间，允许进行修改程序等操作）			
测试条件		**项 目**	**记录**
输入电压	**输出电压设点**		
30.0V±0.1V 电路为开环控	（ 5 ）	（10 分）开环控制下输出电压绝对偏差，记录输出电压	5.001
	（ 7.5 ）		7.526/7.546

制状态	（ 10 ）		10.023
30.0V±0.1V 电路为闭环控制状态	V1=（ 5.1 ）	（10 分）闭环控制下输出电压绝对偏差，老师指定 3 个电压，记录输出电压	5.105
	V2=（ 7.3 ）		7.297
	V3=（ 9.9 ）		9.921
30.0V±0.1V 电路为闭环控制状态, R14 调偏	V1	（10 分）闭环控制下输出电压相对偏差，重做 3 个指定电压的测试，记录输出电压	5.119
	V2		7.296
	V3		9.906
——	——	（5 分）工艺	√
		其他情况记录： 开环、闭环可键控	

用 MSP430 单片机闭环控制电源模块输出电压（测试前给予准备时间，允许进行修改程序等操作）

测试条件		项 目	记录
输入电压	输出电压设点		
30.0V±0.1V 电路为闭环控制状态	V1=（ 5.1 ）	（5 分）闭环控制下输出电压绝对偏差，老师指定 3 个电压，记录输出电压	5.1264/5.08421
	V2=（ 7.3 ）		7.2742/7.3137
	V3=（ 9.9 ）		9.8981/9.9165
30.0V±0.1V 电路为闭环控制状态, R14 调偏	V1	（5 分）闭环控制下输出电压相对偏差，重做 3 个指定电压的测试，记录输出电压	5.08-5.13
	V2		7.285
	V3		9.897
其他情况或拓展功能记录 直接使用超级终端，用数字键控制电压变化			

10. 附录 C　课程学习心得和意见建议

工程实践与科技创新3A是我们进行的第三门科技创新的系列课程，在刚开始做本次科创实验的时候我们就明显感到它与前两次不太相同，以前科创自主设计的部分占的比重较小，但是本次的自主设计的部分却有了较大的提升。

在本学期中，我们组三人的学习压力都很大，在本学期同时进行几门实验，但是我们组三人还是克服万难、同心协力地顺利完成了科创3A这门课程的基本任务和高级拓展，当然中间过程中也出现了较多的问题，比如系统的稳定性我们就保持的不太好，在做的过程中我们自己测量的纹波和输入电流均在要求指标的范围之内，但是在最终检测的时候却出现了一些小问题。

本次科创的难点我们认为主要在纹波与效率的取舍方面，我们在设计好电路后，发现纹波和效率的实际值都略大指标一些，这才意识到规定这些指标的用意，在不断的计算和更改后电路后，二者总是矛盾出现，或纹波达到要求，牺牲了效率；或效率达到了指标，纹波却又大了，几经更改，终于使二者较好的达到了指标。

在完成本次的科创后，我们的动手和实践能力得到了提升，更进一步的明白了理论和实践的区别，期待通过本次的科创以后的相关课程提供借鉴和经验。

11. 附录 D　软件程序清单

11.1 基础部分的程序

```
//AVR application builder : 2010-04-06 12:01:38
// Target : M16
// Crystal: 8.0000Mhz
#include <avr/io.h>
#include <avr/interrupt.h>
//////////////////////////////
// 常量定义 //
//////////////////////////////
// 1s 软件定时器溢出值，200 个 5ms
#define V_T1s 200
//////////////////////////////
// 变量定义 //
//////////////////////////////
// 数码管位和指示灯显示数据变量
unsigned char output_sel;
// 数码管段显示数据变量
unsigned char output_8seg;
// 1s 软件定时器计数
unsigned char clock1s;
// 1s 软件定时器溢出标志
unsigned char volatile clock1s_flag;
// 指示灯驱动信号输出缓存
unsigned char led1,led2,led3,led4;
// 数码管扫描驱动指针
unsigned char digi_scaner;
// 测试用计数器
unsigned int test_counter;
// 测试用计数值十进制表示
unsigned char volatile digi[4];
unsigned int workmode;
unsigned int keypress1,keypress2,keypress3,keypress4;
unsigned int last_k1,last_k2,last_k3,last_k4;
unsigned int voltage,duty_cycle;
unsigned int trig1,trig2,trig3,trig4;
unsigned int display_type;
unsigned int tempvoltage,tempduty_cycle;
unsigned int temp;
unsigned int settime;
signed int digitalvoltage;
signed int totaldigitalvoltage;
unsigned int tempadc;
signed int tempADC;
unsigned int pwmv[53]=
{373,379,
382,388,391,395,400,//5.5
404,408,414,418,422,//6
428,432,438,442,447,//6.5
451,457,457,463,469,//7
475,481,487,493,500,//7.5
507,513,520,523,527,//8
531,534,541,549,556,//8.5
564,572,581,589,598,//9
608,613,617,628,639,//9.5
645,651,664,669,675,//10
680};

signed int ad[53]=
{-260       ,
-255        ,
-245        ,
-229        ,
-218        ,
-208        ,
-198        ,
-189        ,
```

```
-177        ,
-165        ,
-155        ,
-143        ,
-130        ,
-123        ,
-110        ,
-99         ,
-90         ,
-77         ,
-66         ,
-60         ,
-50         ,
-38         ,
-25         ,
-15         ,
-4          ,
4           ,
16          ,
25          ,
35          ,
49          ,
58          ,
68          ,
77          ,
90          ,
100         ,
112         ,
121         ,
128         ,
138         ,
153         ,
161         ,
173         ,
186         ,
193         ,
205         ,
219         ,
232         ,
249         ,
254         ,
258         ,
268         ,
280         ,
290


};
// 函数定义 //
//////////////////////////////
// 7 段数码显示译码
// 参数：
// DATA： 需要显示的数字或符号；
// 返回值： 7 段译码结果 ( D7~0 = PGFEDCBA )
unsigned char NUMTOSEG7(unsigned char DATA)
{ unsigned char AA;
switch (DATA)
{ case 0: AA=0xc0;break; /* '0'*/
case 1: AA=0xf9;break; /* '1'*/
case 2: AA=0xa4;break; /* '2'*/
case 3: AA=0xb0;break; /* '3'*/
case 4: AA=0x99;break; /* '4'*/
case 5: AA=0x92;break; /* '5'*/
case 6: AA=0x82;break; /* '6'*/
case 7: AA=0xf8;break; /* '7'*/
case 8: AA=0x80;break; /* '8'*/
case 9: AA=0x90;break; /* '9'*/
case 10: AA=0x88;break; /* 'A'*/
case 11: AA=0x83;break; /* 'B'*/
case 12: AA=0xc6;break; /* 'C'*/
case 13: AA=0xa1;break; /* 'D'*/
```

```
case 14: AA=0x86;break; /* 'E'*/
case 15: AA=0x8e;break; /* 'F'*/
case 20: AA=0x40;break; /*'0.'*/
case 21: AA=0x79;break; /*'1.'*/
case 22: AA=0x24;break; /*'2.'*/
case 23: AA=0x30;break; /*'3.'*/
case 24: AA=0x19;break; /*'4.'*/
case 25: AA=0x12;break; /*'5.'*/
case 26: AA=0x02;break; /*'6.'*/
case 27: AA=0x78;break; /*'7.'*/
case 28: AA=0x00;break; /*'8.'*/
case 29: AA=0x10;break; /*'9.'*/
case '-':AA=0xbf;break; /* 破折号*/
case '_':AA=0xf7;break; /* 下划线*/
case ' ':AA=0xff;break; /* 消隐*/
default: AA=0xff;
}
return(AA);
}


void display_led(unsigned char seg,unsigned char sel)
{
        unsigned char i;
        //先将 sel 数据送 74hc595
        PORTA &= ~(1<<PA7); // PA7=0; rclk=0
        for (i=0;i<8;i++)
        {
                if ((sel & 0x80) == 0) //最高位送 U2 SER 端
                PORTA &= ~(1<<PA5); // PA5=0
                else
                PORTA |= (1<<PA5); // PA5=1
                PORTA &= ~(1<<PA6); //PA6=0
                PORTA |= (1<<PA6); //PA6=1 srclk=1，产生移位时钟信号
                sel <<= 1; //sel 左移一位
        }
        //再将 seg 数据送 74hc595
        for (i=0;i<8;i++)
        {
                if ((seg & 0x80) == 0) //最高位送 U2 SER 端
                PORTA &= ~(1<<PA5); //PA5=0
                else
                PORTA |= (1<<PA5); //PA5=1
                PORTA &= ~(1<<PA6); //PA6=0
                PORTA |= (1<<PA6); //PA6=1 srclk=1，产生移位时钟信号
                seg <<= 1; //seg 左移一位
        }
                PORTA |= (1<<PA7); // PA7=1; rclk=1
                PORTA &= ~(1<<PA7); // PA7=0; rclk=0，产生锁存输出信号
}


void port_init(void)
{
                PORTA = 0x00;
                DDRA = 0xE0; //PA 口 PA7、PA6、PA5 为输出
                PORTB = 0x00;
                DDRB = 0x00;
                PORTC = 0xf0; //PC 口 PC7、PC6、PC5、PC4 接上拉电阻
                DDRC = 0x00; //PC 口 为输入
                PORTD = 0x00;
                DDRD = 0x00;
}
//TIMER1 initialize - prescale:256
// WGM: Normal
// desired value: 200Hz
// actual value: 200.321Hz (0.2%)
void timer0_init(void)
{
TCCR0 = 0x00; //stop
```

```
TCNT0 = 0x64; //set count
OCR0 = 0x9C; //set compare
TCCR0 = 0x04; //start timer
}
void timer1_init(void)
{
DDRD |= (1<<PD4) | (1<<PD5); //OC1A 、OC1B 设置输出
TCCR1B = 0x00;
OCR1A = 0x1fff;
OCR1B = 0x03ff;
ICR1 = 0xffff;
TCCR1A = 0xb2;
TCCR1B = 0x19;
}



ISR(TIMER0_OVF_vect) //定时器 1 5ms 溢出中断
{
        TCNT0 = 0x64; //reload counter value
        // 1 秒钟软定时器计数
        if (++clock1s>=V_T1s)
        {
                clock1s_flag = 1; //当 1 秒到时，溢出标志置 1
                clock1s = 0;
        }
        output_sel = 0xf0; /*初值，令数码管驱动位无效，指示灯全灭*/
        last_k1=keypress1;
        last_k2=keypress2;
        last_k3=keypress3;
        last_k4=keypress4;
// *** 检测到按键被按下（0）时，相应的指示灯亮（0）*****//
if ((PINC&(1<<PC4))==0) {led1 = 0;keypress1=1;}else{led1 = 1;keypress1=0;}
if ((PINC&(1<<PC5))==0) {led2 = 0;keypress2=1;}else{led2 = 1;keypress2=0;}
if ((PINC&(1<<PC6))==0) {led3 = 0;keypress3=1;}else{led3 = 1;keypress3=0;}
if ((PINC&(1<<PC7))==0) {led4 = 0;keypress4=1;}else{led4 = 1;keypress4=0;}
if((!last_k1)&&keypress1){trig1=1;}else{trig1=0;} //按键防抖
if((!last_k2)&&keypress2){trig2=1;}else{trig2=0;}
if((!last_k3)&&keypress3){trig3=1;}else{trig3=0;}
if((!last_k4)&&keypress4){trig4=1;}else{trig4=0;}
if (workmode==0)
{

                if(trig1==1)
                {
                        voltage+=10;
                        if (voltage>101)
                        voltage=49;
                }
                if(trig2==1)
                {
                        voltage++;
                        if (voltage>101)
                        voltage=49;
                }
                if(trig3==1)
                {
                        voltage--;
                if (voltage<49)
                        voltage=101;
                }

        if(trig4==1)
        {
                workmode=1;
        }
}

else if(workmode==1)
{
        ADMUX=0xD0;
```

```
		ADCSRA=0xC0;
		for(int i=0;i<100;i++) {}
		if(trig1==1)
		{
			voltage+=10;
			if (voltage>110)
			voltage=49;
		}
		if(trig2==1)
		{
			voltage++;
			if (voltage>110)
			voltage=49;
		}
			if(trig3==1)
			{
				voltage--;
			if (voltage<49)
				voltage=101;
			}

		if(trig4==1)
		{
			workmode=0;
		}

		settime++;
		if(settime>20)
		{
			settime=0;
			digitalvoltage=totaldigitalvoltage/20;
			totaldigitalvoltage=0;
			if(digitalvoltage>ad[voltage-49])
			{
				if(digitalvoltage-ad[voltage-49]>50) pwmv[voltage-49]-=15;
				else if(digitalvoltage-ad[voltage-49]>20)	pwmv[voltage-49]-=7;
				else if(digitalvoltage-ad[voltage-49]>10)	pwmv[voltage-49]-=3;
				else if(digitalvoltage-ad[voltage-49]>2)	pwmv[voltage-49]-=1;
			}
			else if(digitalvoltage<ad[voltage-49])
			{
				if(digitalvoltage-ad[voltage-49]<-50)
				pwmv[voltage-49]+=15;
				else if(digitalvoltage-ad[voltage-49]<-20)
				pwmv[voltage-49]+=7;
				else if(digitalvoltage-ad[voltage-49]<-10)
				pwmv[voltage-49]+=3;
				else if(digitalvoltage-ad[voltage-49]<-2)
				pwmv[voltage-49]+=1;
			}
		}

		else if(settime<=20)
		{
			if (ADC>511) tempADC=ADC-1024;
			else tempADC=ADC;
			totaldigitalvoltage+=tempADC;
		}
}

duty_cycle=pwmv[voltage-49]*65536/1000;
OCR1A=duty_cycle;
temp=pwmv[voltage-49];
// 数码管扫描驱动指针值从 1 到 4 重复变换，每 5ms 间隔对一个数码管进行驱动，20ms 一个轮回

if (++digi_scaner>=5) digi_scaner = 1;
output_sel=led1*16+led2*32+led3*64+led4*128; //四个发光管送高四位
switch (digi_scaner)
{
case 1: // 取第一个数码管显示数据
output_sel += 1;
```

```
output_8seg = NUMTOSEG7(10+workmode);
break;
case 2: // 取第二个数码管显示数据
output_sel += 2;
tempvoltage=voltage/100;
tempduty_cycle=temp/100;
tempadc=digitalvoltage/100;
if(display_type==0)
output_8seg = NUMTOSEG7(tempvoltage);
else
output_8seg = NUMTOSEG7(tempduty_cycle);
break;
case 3: // 取第三个数码管显示数据
output_sel += 4;
tempvoltage=(voltage-(voltage/100)*100)/10;
tempduty_cycle=(temp-temp/100*100)/10;
tempadc=(digitalvoltage-digitalvoltage/100*100)/10;
if(display_type==0)
output_8seg = NUMTOSEG7(tempvoltage+20);
else
output_8seg = NUMTOSEG7(tempduty_cycle+20);
break;
case 4: // 取第四个数码管显示数据
output_sel += 8;
tempvoltage=voltage-(voltage/10)*10;
tempduty_cycle=temp-(temp/10)*10;
tempadc=digitalvoltage-(digitalvoltage/10)*10;
if(display_type==0)
output_8seg = NUMTOSEG7(tempvoltage);
else
output_8seg = NUMTOSEG7(tempduty_cycle);
break;
}
display_led(output_8seg,output_sel); //串转并输出
}
//call this routine to initialize all peripherals
void init_devices(void)
{
//stop errant interrupts until set up
cli(); //disable all interrupts
port_init();
timer0_init();
timer1_init();
MCUCR = 0x00;
GICR = 0x00;
TIMSK = 0x01; //timer interrupt sources
sei(); //re-enable interrupts
//all peripherals are now initialized
}
//
int main()
{
init_devices();
workmode=0;
display_type=0;
voltage=49;
keypress1=keypress2=keypress3=keypress4=0;
last_k1=last_k2=last_k3=last_k4=0;
trig1=trig2=trig3=trig4=0;
settime=0;
totaldigitalvoltage=0;
// 主循环，本例中，在 T1 中断服务程序未被执行的空余时间里，处理机在以下程序中不断循环
while(1)
{ } }
```

11.2 高级拓展的程序

```
#include <msp430g2553.h>
// 1s 软件定时器溢出值，200 个 5ms
#define V_T1s  200
```

```
//////////////////////////////
//                          变量定义       //
//////////////////////////////
// 1s 软件定时器计数
unsigned char clock1s=0;
// 1s 软件定时器溢出标志
unsigned char clock1s_flag=0;
unsigned int voltage;
unsigned long dutyCycle;
unsigned int tempvoltage,tempdutyCycle;
unsigned int settime;
unsigned int digitalVoltage;
unsigned int totalVoltage;
unsigned int tempadc;
unsigned int now,last,trig;
unsigned int i;
unsigned int pwmv[53]={100,
100, 100, 100, 100, 100, 100, 100, 100, 100, 100,
157, 157, 157, 157, 157, 157, 157, 157, 157, 157,
200, 200, 200, 200, 200, 200, 200, 200, 200, 200,
235, 235, 235, 235, 235, 235, 235, 235, 235, 235,
260, 260, 260, 260, 260, 260, 260, 260, 260, 260,
260, 260};
signed int ad[53]={ 436, 454, 463, 471, 480, 488 , 497, 506,515, 524, 533,541,551 ,559,568,578,586,594 ,603,612    ,621,630,639
      ,648,658,666,675,683 ,697,702,710,719,728 ,737,746,755,764,773 ,782,791,800,809,817 ,826,835,844,853,862 ,871,880,889
      ,897};
int t_last,new;

void IOInit()
{
      WDTCTL = WDTPW + WDTHOLD;
      P1DIR |= BIT0;     // P1.0 output
      P1DIR &=~BIT3;
      P1REN |= BIT3;
      P1OUT &=~BIT0;
}

void Timer0Init()
{
      TACCR0=7894;
      TACTL=TASSEL_2 + MC_1;
      TACCTL0 |= CCIE;
}

void Timer1Init()
{
       P2DIR |= BIT1;                        // P2.1output
       P2SEL |= BIT1;
       TA1CTL = TASSEL_2 + MC_1+ID_3 + TACLR;       //upmode SMCLK
       TA1CCR0 = 511;
       TA1CCTL1 = OUTMOD_7;                       // CCR1 reset/set
       TA1CCR1 = 128;                          // CCR1 PWM duty cycle(TA1CCR1/TA1CCR0)

}

void uart_init(void)
{
      P1SEL |= BIT1 + BIT2 ;                  // P1.1 = RXD, P1.2=TXD
      P1SEL2 |= BIT1 + BIT2 ;                   // P1.1 = RXD, P1.2=TXD
      UCA0CTL1 |= UCSSEL_2;                     // uart 时钟: SMCLK=1MHz
      UCA0BR0 = 104;                        // 设置波特率:9600
      UCA0BR1 = 0;                          // 设置波特率:9600
      UCA0MCTL = UCBRS0;                        // Modulation UCBRSx = 1
      UCA0CTL1 &= ~UCSWRST;                     // **Initialize USCI state machine**
      IE2 |= UCA0RXIE;                      // Enable USCI_A0 RX interrupt
}

//  uart 串口 RX 接收到一个字符,产生中断处理
#pragma vector=USCIAB0RX_VECTOR
__interrupt void USCI0RX_ISR(void)
{
```

```
    switch (UCA0RXBUF)
                        {
                                case '1': voltage++;
                                        break;

                                case '2': voltage+=10;
                                        break;

                                case '3': voltage--;
                                        break;

                                case '4': voltage-=10;
                                        break;
                                default:
                                    break;
                        }
     if(voltage>101) voltage=49;
     if(voltage<49) voltage=101;
}


void ADCInit()
{
    ADC10CTL0|=SREF_1+ADC10SHT_2+REFON+REF2_5V+ADC10ON;  //VR+ = VREF+ and VR- = AVSS; 64 x ADC10CLKs;using Vref=2.5
    ADC10CTL1|=INCH_4; //A1  SMCLK
   ADC10AE0 |=BIT4;                 // PA.4 ADC option select


}

void valueInitial()
{
    voltage=49;
    settime=0;
    totalVoltage=0;
    last=0;
    now=0;
    trig=0;
    _EINT();

}

void init_devices(void)
{
    WDTCTL = WDTPW + WDTHOLD;           // Stop watchdog timer

            if (CALBC1_8MHZ ==0xFF || CALDCO_8MHZ == 0xFF)
            {
                    while(1);               // If calibration constants erased, trap CPU!!
            }

            //振荡源为片内 RC 振荡器，DCO=8MHz,供 CPU;    SMCLK=1MHz,供定时器 A0、串行口 UART
            BCSCTL1 = CALBC1_8MHZ;                  // Set range
            DCOCTL = CALDCO_8MHZ;                   // Set DCO step + modulation，DCO=8MHz
            BCSCTL3 |= LFXT1S_2;                // LFXT1 = VLO
            IFG1 &= ~OFIFG;                     // Clear OSCFault flag
            BCSCTL2 |= DIVS_3;     //  SMCLK = DCO/8 = 1MHz


            IOInit();
            Timer0Init();
            Timer1Init();
            ADCInit();
            valueInitial();
    uart_init();
 _BIS_SR(GIE); //开全局中断
 //all peripherals are now initialized
}

// Timer A0 interrupt service routine
```

```
#pragma vector=TIMER0_A0_VECTOR
__interrupt void Timer_A (void)
{
    last=now;
    if(!(P1IN & 0X08))
                                                {P1OUT |= BIT0;now=1;}
                                    else    {P1OUT &=  ~BIT0;now=0;}
    if((!last)&&now)trig=1;else trig=0;

    ADC10CTL0|=+ADC10SC+ENC;//Start convertion
    for(i=0;i<100;i++){}
    if(trig==1){voltage++;if(voltage>101)voltage=49;}

    settime++;
    if(settime<=20)         totalVoltage+=ADC10MEM;
    else
            {
                    settime=0;
                    digitalVoltage=totalVoltage/20;
                    totalVoltage=0;
                    if(digitalVoltage>ad[voltage-49])
                    {
                            if(digitalVoltage-ad[voltage-49]>50) pwmv[voltage-49]-=15;
                            else if(digitalVoltage-ad[voltage-49]>20)     pwmv[voltage-49]-=7;
                            else if(digitalVoltage-ad[voltage-49]>10)     pwmv[voltage-49]-=3;
                            else if(digitalVoltage-ad[voltage-49]>2)      pwmv[voltage-49]-=1;
                    }
                    else if(digitalVoltage<ad[voltage-49])
                    {
                            if(ad[voltage-49]-digitalVoltage>50)
                            pwmv[voltage-49]+=15;
                            else if(ad[voltage-49]-digitalVoltage>20)
                            pwmv[voltage-49]+=7;
                            else if(ad[voltage-49]-digitalVoltage>10)
                            pwmv[voltage-49]+=3;
                            else if(ad[voltage-49]-digitalVoltage>2)
                            pwmv[voltage-49]+=1;
                    }

            }
    dutyCycle=pwmv[voltage-49];
    TA1CCR1=dutyCycle;
}

void main(void)
{

    init_devices( );


             while(1)
     {new=voltage;
     if(t_last!=new)
     {t_last= new;
     while (!(IFG2&UCA0TXIFG));              // USCI_A0 TX buffer ready?
              UCA0TXBUF = 0x0D;                  //输出“回车”
        int tmp,k[10],i,q=1,j;
              tmp=new;
            while (!(IFG2&UCA0TXIFG));              // USCI_A0 TX buffer ready?
                                    UCA0TXBUF = 'V';
            while (!(IFG2&UCA0TXIFG));              // USCI_A0 TX buffer ready?
                                    UCA0TXBUF = 'o';
            while (!(IFG2&UCA0TXIFG));              // USCI_A0 TX buffer ready?
                                    UCA0TXBUF = 'l';
            while (!(IFG2&UCA0TXIFG));              // USCI_A0 TX buffer ready?
                                    UCA0TXBUF = 't';
            while (!(IFG2&UCA0TXIFG));              // USCI_A0 TX buffer ready?
                                    UCA0TXBUF = 'a';
            while (!(IFG2&UCA0TXIFG));              // USCI_A0 TX buffer ready?
                                    UCA0TXBUF = 'g';
            while (!(IFG2&UCA0TXIFG));              // USCI_A0 TX buffer ready?
```

```
                                        UCA0TXBUF = 'e';
                while (!(IFG2&UCA0TXIFG));              // USCI_A0 TX buffer ready?
                                        UCA0TXBUF = ':';
            for(i=1; q>0 ;i++)
            {q=tmp/10;
            k[i]=tmp%10;tmp/=10;}
                    k[i]=tmp;
                    i--;
                    for(j=i;j>0;j--){
                    if(j==1){while (!(IFG2&UCA0TXIFG)); UCA0TXBUF = '.';}
            while (!(IFG2&UCA0TXIFG)); UCA0TXBUF = k[j]+0x30;}

        while (!(IFG2&UCA0TXIFG));              // USCI_A0 TX buffer ready?
           UCA0TXBUF = '\n';
        }

        }
}
```