一、实验目的

1. 掌握使用Pt_{100}测温方法，掌握使用电桥电路进行电阻电压转换的方法。
2. 复习 PID 控制原理，掌握 PID 参数调试方法。
3. 掌握 LCD1602 的初始化和使用方法。
4. 掌握 ADC TLC1549 的使用方法。

二、实验任务

1. 使用Pt_{100}热敏电阻及相关电路和单片机实现$0°C\sim100°C$范围内的温度测量，测量精度要求±1°C。
2. 利用功率为 400W 的加热器、继电器和单片机相关电路，利用 PID 方法实现温度控制，要求控制精度为±2°C。
3. 要求设置温度和测量温度显示在 LCD1602 屏幕上。

三、实验原理

3.1 系统原理

3.1.1 系统框图

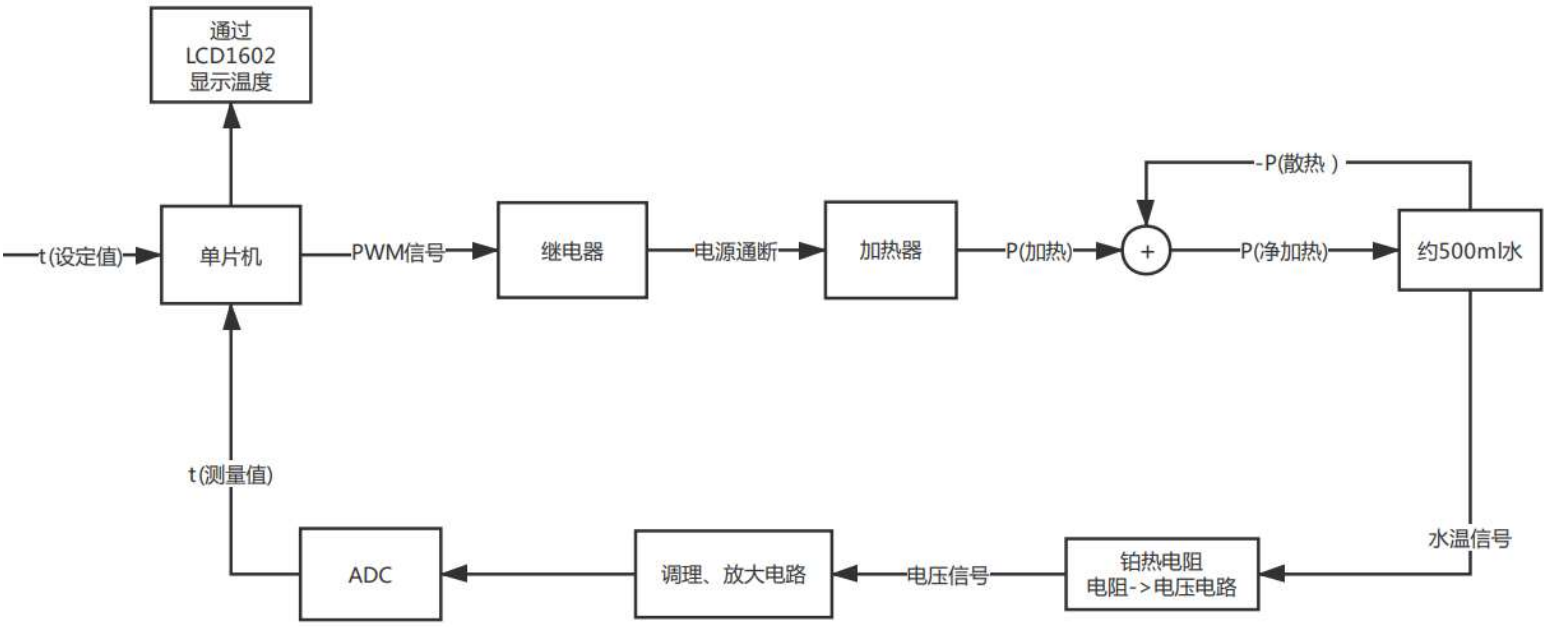

图 1 系统框图

3.1.2.系统原理简介

3.1.2.1 被测对象：电热杯，约 500mL 水

3.1.2.2 执行元件：继电器

3.1.2.3 测温元件：铂热电阻Pt_{100}

3.1.2.4 系统工作原理：

单片机通过比较设定温度和测量温度数据，输出占空比为 W 的 PWM 信号给继电器，继电器在高电平时控制加热器电源接通，在低电平时控制加热器电源断开，由此调整加热功率$P_{加热} = W * 400$ 瓦。当$P_{加热} - P_{散热} > 0$时，水温上升，< 0时水温下降。但由于加热器的惯性过大，即$P_{加热}$对占空比 W 的反应有延迟，因此需要使用 PID 算法调整 W。

Pt_{100}热敏电阻在 0℃~100℃内阻值 R 与温度 t 近似成线性正相关，因此可利用铂热电阻、电阻电压转换电路加放大电路通过 ADC 采样量化获得水温测量值。

3.2 外设简介

3.2.1 LCD1602

LCD1602 是一款一行可以显示 16 个字符，一共有两行的液晶显示屏。

3.2.1.1 LCD1602 引脚

表一 LCD1602 引脚

引脚号	符号	引脚说明	引脚号	符号	引脚说明
1	VSS	电源地	9	D2	数据端口
2	VDD	电源正极	10	D3	数据端口
3	VO	偏压信号	11	D4	数据端口
4	RS	命令/数据	12	D5	数据端口
5	RW	读/写	13	D6	数据端口
6	E	使能	14	D7	数据端口
7	D0	数据端口	15	A	背光正极
8	D1	数据端口	16	K	背光负极

1）RS 是命令/数据选择引脚，接单片机的一个 I/O，当 RS 为低电平时，选择命令；当 RS 为高电平时，选择数据。

2）RW 是读/写选择引脚，接单片机的一个 I/O，当 RW 为低电平时，向 LCD1602 写入命令或数据；当 RW 为高电平时，从 LCD1602 读取状态或数据。如果不需要进行读取操作，可以直接将其接 VSS。

3）E 是执行命令的使能引脚.

4）D0—D7，并行数据输入/输出引脚，可接单片机的 P0—P3 任意的 8 个 I/O 口。如果接 P0 口，P0 口应该接 4.7K—10K 的上拉电阻。如果是 4 线并行驱动，只须接 4 个 I/O 口。

3.2.1.2 LCD1602 基本操作

LCD1602 的操作分为 4 种

1）读状态：输入 RS=0，RW=1，E=高电平。输出：D0—D7 为状态字。

2）读数据：输入 RS=1，RW=1，E=高电平。输出：D0—D7 为数据。

3）写命令：输入 RS=0，RW=0，E=高电平。

4）写数据：输入 RS=1，RW=0，E=高电平。

3.2.1.3 LCD1602 控制字

指令	主要功能	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0
1	清显示	0	0	0	0	0	0	0	0	0	1
2	光标复位	0	0	0	0	0	0	0	0	1	*
3	设定输入模式	0	0	0	0	0	0	0	1	I/D	S
4	开关控制的显示方式	0	0	0	0	0	0	0	D	B	C
5	光标或画面滚动	0	0	0	0	0	1	S/C	R/L	*	*
6	数据和字符的工作模式	0	0	0	0	1	DL	N	F	*	*
7	设置字符发生器的地址	0	0	0	1	字符发生器的地址 CGRAM					
8	设置数据存储器的地址	0	0	1	数据存储器地址 DDRAM						
9	忙标志位和数据指针	0	1	BF	数据指针						
10	写数据	1	0	需读的数据							
11	读数据	1	1	需写的数据							

http://www.go-gddq.com

图 2 LCD1602 控制字

鉴于此部分在每一份实验报告可能都会出现，且“正统”内容在网上十分容易搜到，因此在此处仅介绍一下本次实验使用到的几个控制字：

0x38：DL=1，设置使用八位数据口；N=1，设置 16 字符×2 行显示；F=0，使用 5*7 点阵显示。

0x0E：D=1 开启显示，C=0 设置无光标，B=1 设置光标闪烁。

0x06：I/D=1 设置光标右移当读或写一个字符后，地址指针加 1；S=0，当写一个字符时，整屏的显示都不移动。

0x81、0xC0：设置数据指针为第一行第一列和第二行第一列。

3.2.2 TLC1549

TLC1549 是一个 10 位串行 AD 转换芯片。

3.2.2.1 TLC1549 引脚

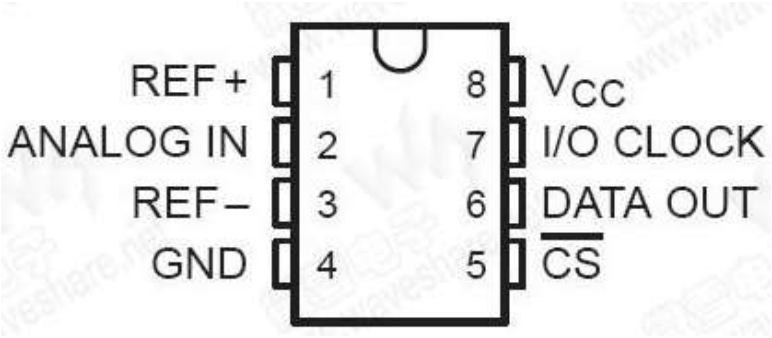

图 3 TCL1549 引脚图

3.2.2.2 TLC1549 使用方法

REF+/REF-为参考高/低电平，电路中分别接 Vcc 和 GND 即可。

模拟信号从 ANALOG IN 输入，采样量化后的信号在 CS 使能时，在 I/O CLOCK 的上升沿后从 DATA OUT 输出，高位先输出。编程时可软件模拟 I/O CLOCK，然后读取 DATA OUT，重复十次即可获取芯片上一次采样得到的值。因为读的是上一次采样的值，因此芯片第一次读取的数无意义，其需要通过一次读取来初始化芯片。

四、电路和程序设计：

4.1 电阻电压转换电路

4.1.1 电路图

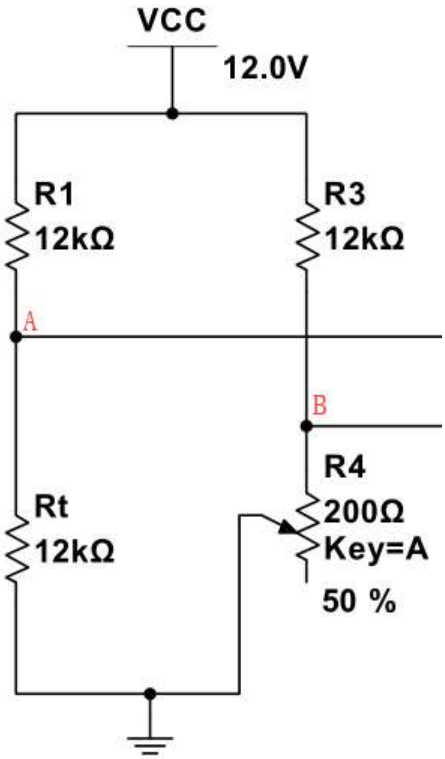

图 4 电阻电压转换电路

4.1.2 工作原理

设Pt_{100}的电阻

$$\mathrm{R_t} = R_0 + \Delta R$$

则有

$$V_A = \frac{R_t}{R_1 + R_t} \times V_{cc} = \frac{R_0 + \Delta R}{R_1 + R_0 + \Delta R} \times V_{cc}$$

$$\mathrm{V_B} = \frac{R_4}{R_3 + R_4} \times V_{cc}$$

若取$R_1 = R_3, R_4 = R_0$.并假设$\Delta R \ll R_0$，则有

$$V_o = V_A - V_B = \frac{R_0 + \Delta R}{R_1 + R_0 + \Delta R} - \frac{R_0}{R_1 + R_0}$$

$$\approx \frac{\Delta R}{R_1 + R_0}$$

因此$\mathrm{V_o}$近似正比与ΔR。但考虑到正比关系是基于$\Delta R \ll R_0$的假设的，而实际上，$R_0 \approx 100\Omega$，在 100℃时$\Delta R \approx 40\Omega$，在实验中可用分段近似或查表法求得精确关系，但本次试验中依然当成线性关系考虑。

4.1.3 注意事项

1）注意极性$\mathrm{V_A} > \mathrm{V_B}$.

2）由于电流通过铂热电阻也会使铂热电阻自身产生热量，为了减小电阻自热对测量温度和被测物体温度场的影响，要尽量使用小电流，因此在$V_{cc}=12V$时，取$R_1=R_3\approx 12k\Omega$，$I_{P_t}\approx 1mA$.

3）电路中为了调零偏，要将R_4选为电位器，其阻值包含 100Ω即可。

4.2 调理、放大电路

4.2.1 电路图

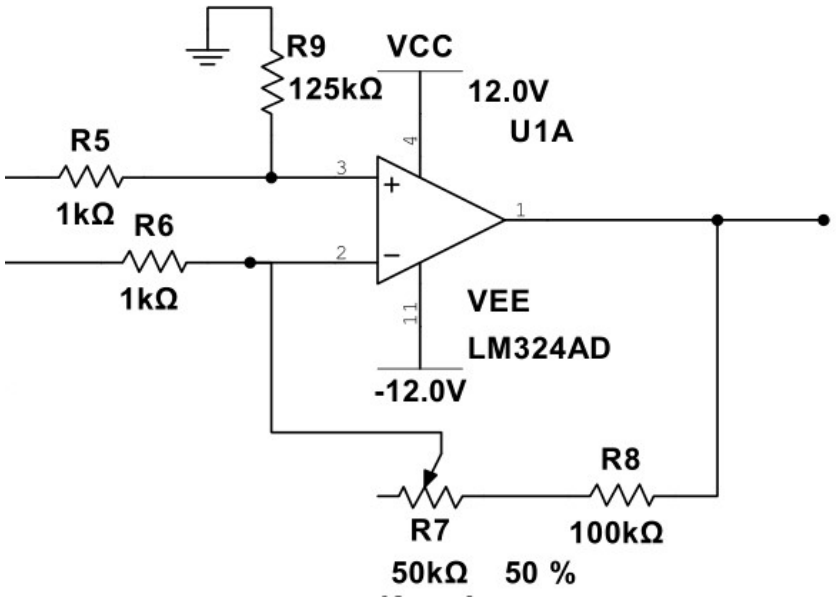

图 5 放大电路

4.2.2 注意事项

1）R_5和R_6作为电桥电路的负载，其阻值不能太小，从 A 点看电桥电路的电阻仅有约 100Ω，因此R_5至少要 1kΩ，但也不能阻值过大，否则要达到相同放大倍数，会使反馈电路上的电阻过大。

2）当$R_5=1k\Omega$时，为使放大倍数约为 125，反馈回路电阻应为 125kΩ，但之后需要测试调满，因此选反馈电阻R_f为定值电阻和电位器的串联，以提高调节精度，其可调范围应包含 125kΩ。

4.3 程序设计

程序设计中 LCD1602 和 TLC1549 的使用方法已在前文叙述，此处不再赘述。程序的核心是 PID 算法。考虑到 51 单片机的速度，程序中的温度使用 int 型表示，而非 float 型，并在实际温度的基础上扩大 100 倍，也即相当于

对实际温度保留两位小数的精度。但在下列公式中，并未体现该程序设计技巧。

设

$$Error(k) = T_{set}(k) - T_{measure}(k)(℃)$$

则占空比

$$DutyCycle'(k) = K_p \times Error(k) + K_I \times \sum_{0}^{n} Error(k) + K_D \times [Error(k) - Error(k-1)]$$

注意到占空比取值在 0~1，因此实际占空比

$$DutyCycle(k) = \begin{cases} 1, DutyCycle'(k) > 1 \\ DutyCycle'(k), 0 < DutyCycle'(k) < 1 \\ 0, DutyCycle'(k) < 0 \end{cases}$$

在积分（求和）环节中，可以选用从 n-N+1 到 n 的 Error 求和，也即可取积分窗口为 N，但由于这样做需要额外 N 个变量储存这些 Error，因而程序中依然使用了从 0 到 n 的求和。

五、实验结果

通过对电位器 R4、R7 调零调满、调节 PID 的参数，使系统将水从室温（约 20℃）加热到 70℃，超调量<1℃，并可在 70℃时保温，保温时占空比约 10%~15%，保温功率约 40W~60W。

5.1 模拟电路部分电路图：

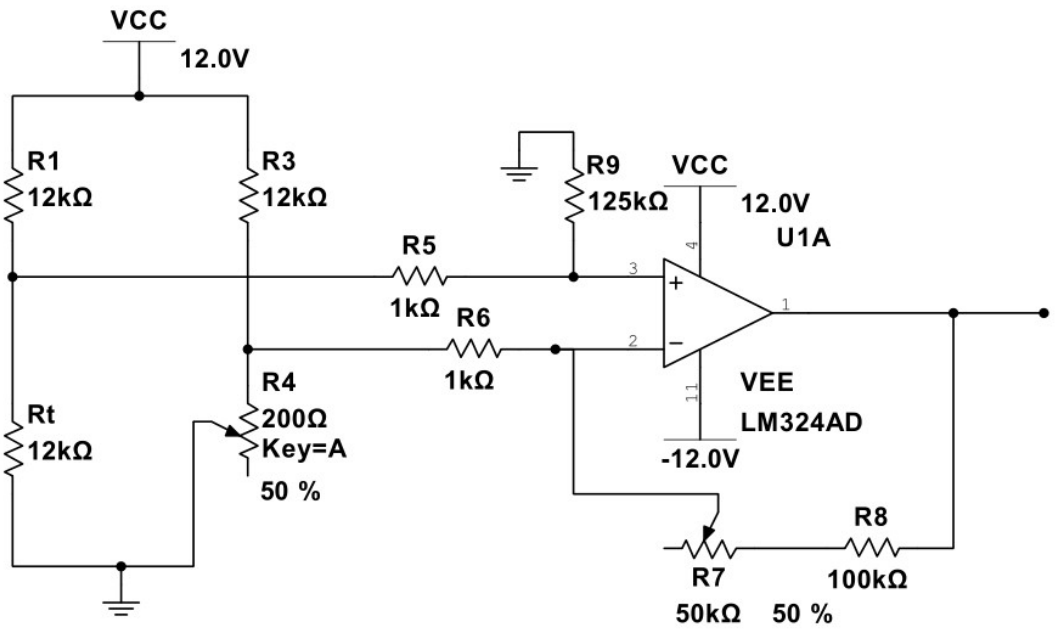

图 6 模拟电路部分电路图

调零调满后电阻参数：

$$R4: 100.8\Omega,\ R7 = 48.2k\Omega$$

5.2 PID 参数

在测温周期为 1s 时，PID 参数如下：

$$\mathrm{K_p} = 3, K_I = 0.01, K_D = 0.1.$$

六、附录

6.A 代码

完整代码已开源：https://github.com/cantjie/EELC530705-Electronic-System-Design1/blob/master/exp3_temperature

6.A.1 LCD1602 初始化

```
void LCDInit()
{
    LCDWriteCommand(0x38); // 16*2 显示，5*7 点阵，8 位数据口
    LCDWriteCommand(0x0E); // 设置开启显示，不显示光标
    LCDWriteCommand(0x06); // 写一个字符后地址指针+1
    LCDWriteCommand(0x01); // 显示清零，数据指针清零 0
    LCDWriteCommand(0x80 + 0x01); // 设置数据地址指针从第一个开始
    LCDWriteString(G_LCD_table[0], 16);
    LCDWriteCommand(0x80 + 0x40);
    LCDWriteString(G_LCD_table[1], 16);
}
```

6.A.2 读 TLC1549

```
unsigned int ADCRead()
{
    unsigned int ADC_data = 0;
    unsigned char i = 0;
    pin_ADC_cs = 0;
    for (i = 0; i < 10; i++)
    {
        pin_sclk = 0;
        pin_sclk = 1; // 上升沿读数据
        ADC_data |= (pin_ADC_dataout & 0x01);
        ADC_data <<= 1;
        delay(1);
    }
    ADC_data >>= 1;
    pin_ADC_cs = 1;//片选禁止
    return ADC_data;
}
```

6.A.3 PID 算法

```
void PIDupdate()
{
    int temperature_difference = 0;
    temperature_difference = G_setting_temperature -
  G_measured_temperature;
    G_pid_sum += temperature_difference;
    G_pid_difference = temperature_difference -
  G_pid_last_temperature_difference;
    G_pid_last_temperature_difference = temperature_difference;
}


unsigned char getDutyCycle()
{
    float duty_cycle = 0;
    PIDupdate();
    duty_cycle = PID_K_P * G_pid_last_temperature_difference +
  PID_K_I * G_pid_sum + PID_K_D * G_pid_difference;

    if (duty_cycle > 100) {
        return G_duty_cycle = 100;
    }
    else {
        if (duty_cycle < 0) {
            return G_duty_cycle = 0;
        }
        else {
            return G_duty_cycle = (unsigned int)duty_cycle;
        }
    }
}
```