哈尔滨理工大学

自动化单片机应用系统设计与实践

-II 报告

姓　　名_____

班级学号_____

教　　师_____赵　阳_____

实验名称_____实验十一　数字式温度计实验_____

日　　期_____2024.11.02_____

一、实验名称：实验十一　数字式温度计实验

二、实验内容及分析：

1、 **实验内容：（实验指导书）**

用 proteus 软件。制作一个数字温度计，通过 DS18B20 测量温度，用共阴数码管动态显示温度。

2、分析：

1） **附硬件原理图（实验指导书）；**

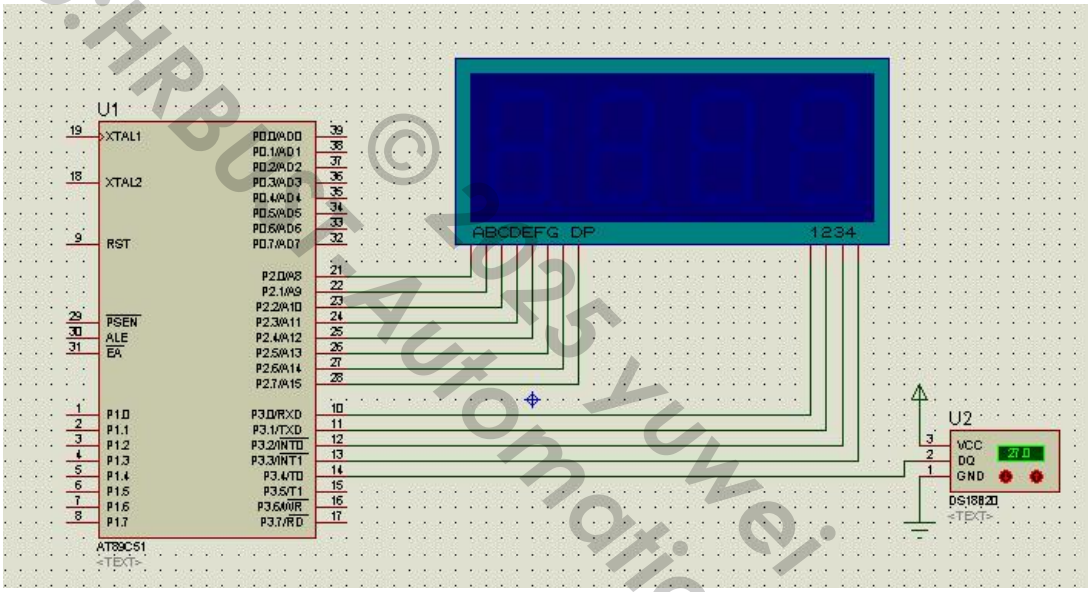

图 1 数字式温度计硬件原理图

2） **硬件原理图的功能及硬件电路中所涉及芯片的原理；**

- AT89C51 单片机 (U1)

AT89C51 是一款经典的 8 位单片机，带有 4KB 的 ROM 和 128 字节的 RAM。在该电路中，AT89C51 负责控制整个系统的运行，包括从 DS18B20 获取温度数据、处理数据并将结果显示在七段数码管上。P2 口用于连接数码管的各个段（A、B、C、D、E、F、G 和小数点 DP），以控制数码管的显示内容。P1 口的引脚用于控制数码管的位选信号，以实现动态显示。

- DS18B20 温度传感器 (U2）

DS18B20 是一种单总线数字温度传感器，具有测温范围广、精度高的特点。它采用单总线通信方式，仅需一根数据线与 AT89C51 连接（图中的 DQ 引脚与 AT89C51 相连），可以实现温度数据的传输。AT89C51 单片机通过向 DS18B20 发送指令来读取温度数据，DS18B20 会将测得的温度以数字信号形式返回给单片机。DS18B20 的数据输出以 9 至 12 位的数字格式表示，单片机会对读取的数值进行处理以显示在数码管上。

- 共阴极七段数码管

共阴极七段数码管用于显示温度数据，包含多个数码管，可以显示多个位数的数字。数码管的各段（A-G 和小数点 DP）通过 P2 口连接到 AT89C51，控制每段的点亮与否。数码管的位选信号通过 P1 口控制，实现动态显示功能。动态显示是指在时间上分时扫描各

个数码管，让不同位数的数码管依次显示不同的数字，从而在人眼视觉上形成一个完整的数字显示效果。

- 硬件工作原理总结

AT89C51 通过单总线与 DS18B20 通信，获取当前温度数据。获取的数据经过 AT89C51 处理后，转换成相应的数码管显示格式。AT89C51 通过控制 P1 口的位选信号，实现数码管的动态扫描，依次显示温度的每一位。这种电路可以实现实时温度监控，并通过七段数码管显示温度值。

3） **软件 C51 程序分析，要求对程序中每条语句加以注释。**

```
// 读写 DS18B20 温度传感器的函数
void ds18b20()
{
  int temp1, temp2;       // 定义两个整型变量 temp1 和 temp2，用于存储温度数据的高 8 位和低 8 位
  dq = 1;                 // 将数据线 dq 设置为 1，释放总线
  initial();              // 初始化 DS18B20，复位并准备通信
  write(0xcc);            // 发送指令 0xCC，忽略 ROM 指令，选择跳过 ROM 匹配，直接通信
  write(0x44);            // 发送指令 0x44，启动温度转换
  dq = 1;                 // 将数据线 dq 设置为 1，等待 DS18B20 完成转换
  initial();              // 再次初始化 DS18B20，复位并准备读取温度数据
  write(0xcc);            // 发送指令 0xCC，忽略 ROM 指令
  write(0xbe);            // 发送指令 0xBE，读取暂存器中的温度值
  temp1 = read();         // 读取温度值的低 8 位并存入 temp1
  temp2 = read();         // 读取温度值的高 8 位并存入 temp2

  if(temp2 < 8)           // 判断温度是否为正温度（温度数据的高 4 位小于 8 表示正温度）
  {
    n = (temp1) | (temp2 << 8);  // 将高 8 位和低 8 位组合成完整的温度值
    n = n * 0.0625;       // 转换成实际温度值，DS18B20 的温度精度为 0.0625°C
    flag = 0;             // 标记 flag = 0 表示正温度
  }
  if(temp2 > 8)           // 判断温度是否为负温度（温度数据的高 4 位大于 8 表示负温度）
  {
    n = (temp1) | (temp2 << 8);  // 将高 8 位和低 8 位组合成完整的温度值
    n = ~n;               // 取反操作，将负温度值转换为补码形式
    n++;                  // 加 1，完成补码转换
    n = n * 0.0625;       // 转换成实际温度值，精度为 0.0625°C
    flag = 1;             // 标记 flag = 1 表示负温度
  }
}
// 毫秒级延时函数void delayms(int i)
{
  int j;
  while(i--)              // 递减 i，直到 i 为 0，形成延时的循环
  {
```

```
    for(j = 255; j > 0; j--); // 内层循环，从 255 到 0，进一步增加延时
  }
}
```

- 个人理解和分析

这段代码用于通过 DS18B20 温度传感器读取温度值。首先，程序声明了两个整数变量 temp1 和 temp2，分别用于存储从传感器读取的低八位和高八位的温度数据。

在代码开始时，dq 引脚被设置为高电平，准备与 DS18B20 进行通信。接着调用 initial() 函数进行必要的初始化，以确保传感器可以正常工作。

随后，程序发送 0xCC 命令，这个命令是跳过 ROM 的命令，使得 DS18B20 能够与所有连接的设备进行通信。接着发送 0x44 命令以启动温度转换过程，这样 DS18B20 会开始测量温度。

温度转换完成后，程序再次将 dq 设置为高电平，并调用 initial() 函数，以准备读取温度数据。再次发送 0xCC 命令来选择 DS18B20，并发送 0xBE 命令以指示传感器读取温度值。

程序随后通过调用 read() 函数两次，分别读取温度的低八位和高八位，分别存储在 temp1 和 temp2 中。

接下来，程序根据 temp2 的值来判断温度是正值还是负值。当 temp2 小于 8 时，表示温度为正，程序将 temp1 和 temp2 组合成完整的温度值 n，然后将其除以 16（即 n / 2 / 2 / 2），相当于将值转换为摄氏度，单位为 0.0625° C，并设置 flag 为 0，表示当前温度为正。

如果 temp2 大于 8，则表示温度为负值，程序同样组合 temp1 和 temp2 得到完整的温度值 n。此时，程序对 n 进行取反，并加 1，以计算补码形式的负温度值，最后也将其除以 16，将其转换为摄氏度，并将 flag 设置为 1，表示当前温度为负。

通过这种方式，代码能够有效读取 DS18B20 传感器的温度数据，并正确处理正负温度情况。这一过程确保了温度值的准确获取，使得后续的温度控制或监测系统可以依赖于该数据进行操作。

3、 仿真作业：

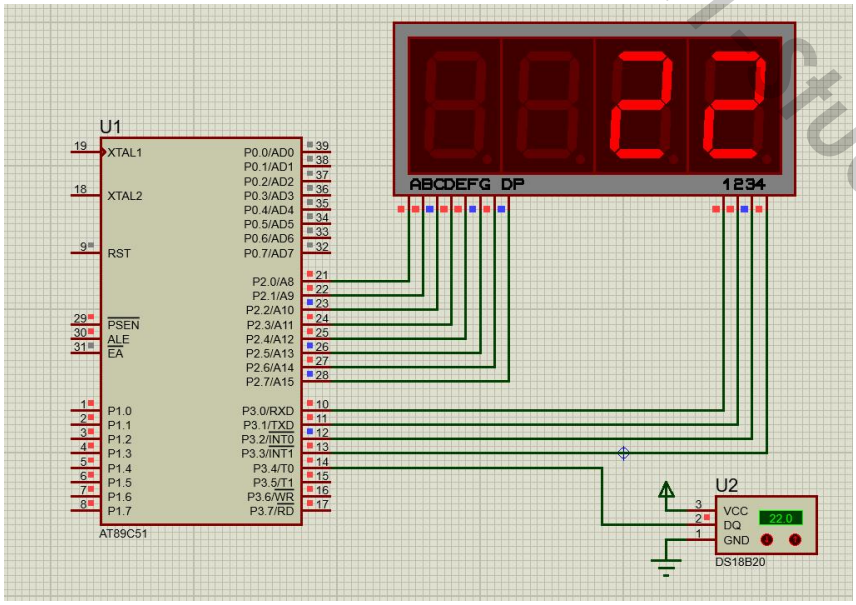

图 2 数字式温度计仿真实验图

三、小组成员在内容及分析中所承担的任务（如表 1）：

表 1

小组成员姓名	所承担的任务
	软件部分 DS18B20 读写函数

四、成绩评定（如表 2）：

表 2

姓名	过程考核成绩（30%）		实践考核成绩（70%）				合计
	课堂讨论 50 分	回答问题 50 分	仿真视频 20 分	所承担的任务（硬件、软件）分析 50 分	实验报告完成 20 分	分析思考 10 分	