PT100
铂金属温度传感器使用说明书

产品介绍

Pt100 温度传感器为正温度系数热敏电阻传感器，主要技术参数如下：

- 测量范围：-200℃～+850℃；
- 允许偏差值△℃：A 级±（0.15＋0.002｜t｜），B 级±（0.30＋0.005｜t｜）；
- 最小置入深度：热电阻的最小置入深度≥200mm；
- 允通电流 ≤ 5mA。

另外，Pt100 温度传感器还具有抗振动、稳定性好、准确度高、耐高压等优点。铂热电阻的线性较好，在 0~100 摄氏度之间变化时，最大非线性偏差小于 0.5 摄氏度。

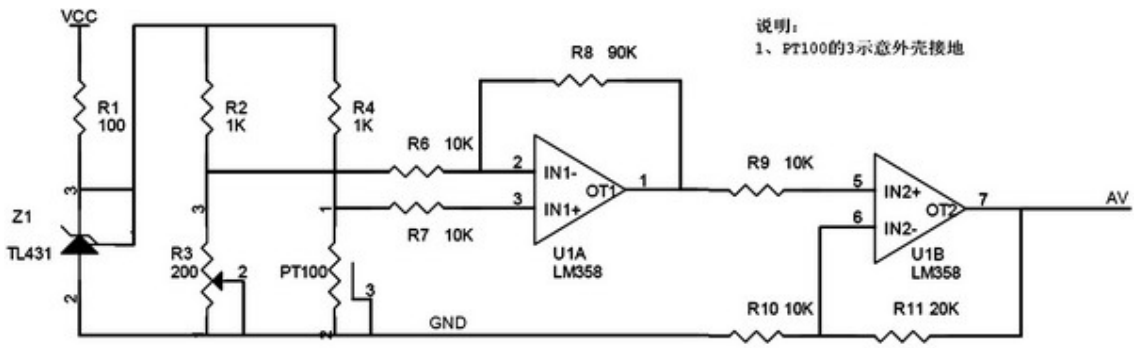
图 2 支 PT100 传感器封装图

应用领域

宽范围、高精度温度测量领域。如：

- 轴瓦，缸体，油管，水管，汽管，纺机，空调，热水器等狭小空间工业设备测温和控制。
- 汽车空调、冰箱、冷柜、饮水机、咖啡机，烘干机以及中低温干燥箱、恒温箱等。
- 供热/制冷管道热量计量，中央空调分户热能计量和工业领域测温和控制

常用电路图



R2、R3、R4 和 Pt100 组成传感器测量电桥，为了保证电桥输出电压信号的稳定性，电桥的输入电压通过 TL431 稳至 2.5V。从电桥获取的差分信号通过两级运放放大后输入单片机。电桥的一个桥臂采用可调电阻 R3，通过调节 R3 可以调整输入到运放的差分电压信号大小，通常用于调整零点。

放大电路采用 LM358 集成运算放大器，为了防止单级放大倍数过高带来的非线性误差，放大电路采用两级放大，如图 5.1 所示，前一级约为 10 倍，后一级约为 3 倍。温度在 0~100 度变化，当温度上升时，Pt100 阻值变大，输入放大电路的差分信号变大，放大电路的输出电压 Av 对应升高。

注意：虽然电桥部分已经经过 TL431 稳压，但是整个模块的电压 VCC 一定要稳定，否则随着 VCC 的波动，运放 LM358 的工作电压波动，输出电压 Av 随之波动，最后导致 A/D 转换的结果波动，测量结果上下跳变。

温度变化

铂热电阻阻值与温度关系为：

（1）-200℃<t<0℃时，$RPt100 = 100*[1 + At + B*t^2 + C*t^3*(t-100)]$

（2）0℃≤t≤850℃时，$RPt100 = 100*(1 + At + B*t^2)$

式中，A=0.00390802；B=-0.000000580；C=0.0000000000042735。可见 Pt100 在常温 0~100 摄氏度之间变化时线性度非常好，其阻值表达式可近似简化为：RPt=100（1+At），当温度变化 1 摄氏度，Pt100 阻值近似变化 0.39 欧。

Tel：010-62669059
Website：www.61mcu.com
E-mail：fae_61mcu@163.com

℃	0	1	2	3	4	5	6	7	8	9
0	100	100.39	100.78	101.17	101.56	101.95	102.34	102.73	103.12	103.51
10	103.9	104.29	104.68	105.07	105.46	105.85	106.24	106.63	107.02	107.4
20	107.79	108.18	108.75	108.96	109.35	109.73	110.12	110.51	110.9	111.28
30	111.67	112.06	112.45	112.83	113.22	113.61	114.99	114.38	114.77	115.15
40	115.54	115.93	116.31	116.7	117.08	117.47	117.85	118.24	118.62	119.01
50	119.4	119.78	120.16	120.55	120.93	121.32	121.7	122.09	122.47	122.86
60	123.24	123.62	124.01	124.39	124.77	125.16	125.54	125.92	126.31	126.69
70	127.07	127.45	127.84	128.22	128.6	128.98	129.37	129.75	130.13	130.51
80	130.89	131.27	131.66	132.04	132.42	132.8	133.18	133.56	133.94	134.32
90	134.7	135.08	135.46	135.84	136.22	136.6	136.98	137.36	137.74	138.12
100	138.5	138.88	139.26	139.64	140.02	140.39	140.77	141.15	141.53	141.91

Pt100 的分度表（0℃~100℃）

程序处理

一般在使用 PT100 的温度采集方案中，都会对放大器 LM358 采集来的模拟信号 AV 进行温度采样，即进行 A/D 转换。

A/D 处理包括两方面内容，一是 A/D 值的滤波处理，二是 A/D 值向实际温度转换。由于干扰或者电路噪声的存在，在采样过程当中会出现采样信号与实际信号存在偏差的现象，甚至会出现信号的高低波动，为了减小这方面原因造成的测量误差，在实际采样时采样 18 个点，然后再除去其中偏差较大的两个点，即一个最大值和一个最小值，再对剩余的 16 个点取均值，这样得到的 A/D 转换结果比较接近实际值。

在对数值进行滤波操作之后，还要将 A/D 值转换为温度，常用的两种方法为查表法和公式法：查表法比较麻烦，而且精度也不高，适合于线性化较差的 NTC 温度传感器；公式法比较简单，只需要确定比例系数 K 和基准偏差 B 即可，适合于线性化较好的传感器

温度转换的 C 语言实现过程为：

```
fT = (ADC_data * K)  -  B;              //换算成温度值。
```

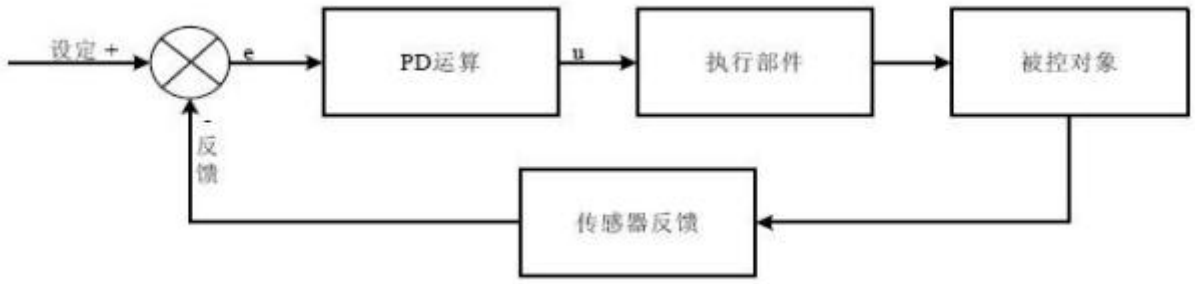
得到温度后，一般还会对被控对象根据实际温度和目标温度进行实时的控制，要又要设计到控制算法，如：模糊控制、PID 调节等。这里简单介绍一下 PID 控制原理，更多内容请察看相关书籍。

PID 算法介绍

PID（Proportional Integral Derivative）控制是控制工程中技术成熟、应用广泛的一种控制策略，经过长期的工程实践，已形成了一套完整的控制方法和典型的结构。它不仅适用于数学模型已知的控制系统中，而且对于大多数数学模型难以确定的工业过程也可应用，在众多工业过程控制中取得了满意的应用效果。

PID 工作基理

由于来自外界的各种扰动不断产生，要想达到现场控制对象值保持恒定的目的，控制作用就必须不断的进行。若扰动出现使得现场控制对象值(以下简称被控参数)发生变化，现场检测元件就会将这种变化采集后经变送器送至 PID 控制器的输入端，并与其给定值(以下简称 SP 值)进行比较得到偏差值(以下简称 e 值)，调节器按此偏差并以我们预先设定的整定参数控制规律发出控制信号，去改变调节器的开度，使调节器的开度增加或减少，从而使现场控制对象值发生改变，并趋向于给定值(SP 值)，以达到控制目的 ，如图所示，其实 PID 的实质就是对偏差（e 值）进行比例、积分、微分运算，根据运算结果控制执行部件的过程。



温度控制 PID 算法设计

利用了上面所介绍的位置式 PID 算法，将温度传感器采样输入作为当前输入，然后与设定值进行相减得偏差，然后再对之进行 PID 运算产生输出结果 fOut，然后让 fOut 控制定时器的时间进而控制加热

Tel：010-62669059
Website：www.61mcu.com
E-mail：fae_61mcu@163.com

器。为了方便 PID 运算，首先建立一个 PID 的结构体数据类型，该数据类型用于保存 PID 运算所需要的 P、I、D 系数，以及设定值，历史误差的累加和等信息：

```
typedef struct PID
{
    float SetPoint;                 // 设定目标 Desired Value
    float Proportion;               // 比例系数 Proportional Const
    float Integral;                 // 积分系数 Integral Const
    float Derivative;               // 微分系数 Derivative Const
    int LastError;                  // 上次偏差
    int SumError;                   // 历史误差累计值
} PID;
PID stPID;                          // 定义一个 stPID 变量
```

PID 运算的 C 实现代码

```
float PIDCalc( PID *pp, int NextPoint )
{
    int dError,Error;
    Error = pp->SetPoint*10 － NextPoint;    // 偏差，设定值减去当前采样值
    pp->SumError += Error;                   // 积分，历史偏差累加
    dError = Error－pp->LastError;           // 当前微分，偏差相减
    pp->PrevError = pp->LastError;           // 保存
    pp->LastError = Error;
    return (pp->Proportion * Error+ pp->Integral * pp->SumError－ pp->Derivative * dError);
}
```

其中(pp->Proportion * Error)是比例项；(pp->Integral * pp->SumError)是积分项;、(pp->Derivative * dError)是微分。