PROJECTS | PUZZLERS | PRODUCT REVIEW | LEARNING CENTER | JAMECO @ WORK

Tech Tips ▾

Thermistors/Temperature Measurement with NTC Thermistors

By Philip Kane

Thermistors (thermal resistors) are temperature dependent variable resistors. There are two types of thermistors, **Positive Temperature Coefficient** (PTC) and **Negative Temperature Coefficient** (NTC). When the temperature increases, PTC thermistor resistance will increase and NTC thermistor resistance will decrease. They exhibit the opposite response when the temperature decreases.

Both types of thermistors are used in a variety of application areas. However, here the focus will be on using NTC thermistors to measure temperature in microcontroller based applications.

Thermistor Specifications
The following NTC thermistor parameters can be found in the manufacturer's data sheet.

- Resistance
  This is the thermistor resistance at the temperature specified by the manufacturer, often 25°C.
- Tolerance
  Indicates how much the resistance can vary from the specified value. Usually expressed in percent (e.g. 1%, 10%, etc). For example, if the specified resistance at 25°C for a thermistor with 10% tolerance is 10,000 ohms then the measured resistance at that temperature can range from 9,000 ohms to 11000 ohms.
- B (or Beta) constant
  A value that represents the relationship between the resistance and temperature over a specified temperature range. For example, "3380 25/50" indicates a beta constant of 3380 over a temperature range from 25°C to 50°C.
- Tolerance on Beta constants
  Beta constant tolerance in percent.
- Operating Temperature Range
  Minimum and maximum thermistor operating temperature.
- Thermal Time Constant
  When the temperature changes, the time it takes to reach 63% of the difference between the old and new temperatures.
- Thermal Dissipation Constant
  Thermistors are subject to self-heating as they pass current. This is the amount of power required to raise the thermistor temperature by 1°C. It is specified in milliwatts per degree centigrade (mW/°C). Normally, power dissipation should be kept low to prevent self-heating.
- Maximum Allowable Power
  Maximum power dissipation. It is specified in Watts (W). Exceeding this specification will cause damage to the thermistor.
- Resistance Temperature Table
  Table of resistance values and associated temperatures over the thermistors operating temperature range. Thermistors operate over a relatively limited temperature range, typically -50 to 300°C depending on type of construction and coating.

Thermistor Response to Temperature

As with any resistor, you can use the ohmmeter setting on your multimeter to measure thermistor resistance. The resistance value displayed on your multimeter should correspond to the ambient temperature near the thermistor. The resistance will change in response to temperature change.

Part List Full kit with Arduino

Qty.	Description	Mfr. Part No.
1	MCU, Arduino Uno R3	**A000066**
1	28AWG Shielded USB Cable	**10U2-02206W**
1	Thermistor 10kΩ	**NTC-103-R**
10	Resistor Carbon Film 10kΩ	**CF1/4W103JRC**
1	Breadboard 170 points 1.9" x 1.3"	**WBP-317**
1	Jumper Wire Male to Male, 10-pack	**WJW004**
1	9V Battery Snap	**1X9V-2.1 SNAP**
1	9V Alkaline Battery	**ALK 9V 522**

Part List without Arduino

Qty.	Description	Mfr. Part No.
1	Thermistor 10kΩ	**NTC-103-R**
10	Resistor Carbon Film 10kΩ	**CF1/4W103JRC**
1	Breadboard 170 points 1.9" x 1.3"	**WBP-317**
1	Jumper Wire Male to Male, 10-pack	**WJW004**
1	9V Battery Snap	**1X9V-2.1 SNAP**
1	9V Alkaline Battery	**ALK 9V 522**

Figure 1: Thermistor resistance changes with temperature.

Figure 2 shows the response of a NTC thermistor between -40°C and 60°C. From the figure you can see that thermistors have high sensitivity. A small change in temperature causes a large change in resistance. Also note that the response of this thermistor is not linear. That is, the change in resistance for a given change in temperature is not constant over the thermistor's temperature range.

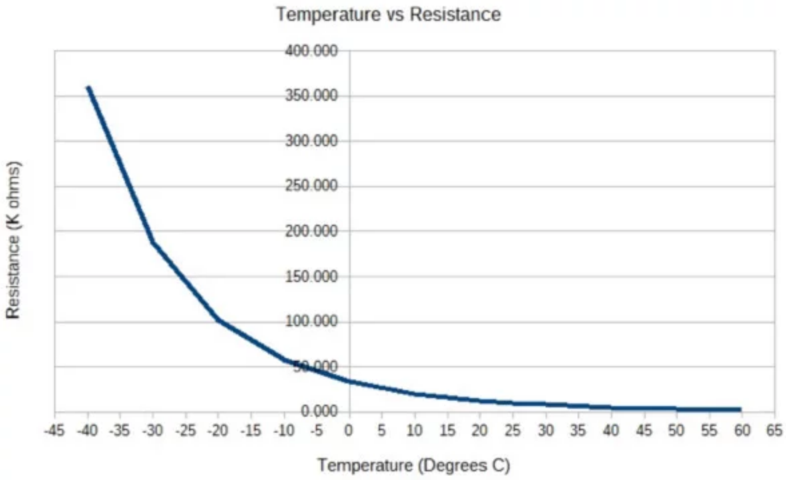


Figure 2: Thermistor temperature-resistance curve -40°C to 60°C

The manufacturer's data sheet includes a list of thermistor resistance values and corresponding temperatures over its range. One solution for dealing with this non-linear response is to include a look-up table containing this temperature-resistance data in your code. After calculating the resistance (to be described later) your code searches the table for the corresponding temperature.

Linearizing Thermistor Response

On the hardware side you can linearize thermistor response by placing a fixed resistor in parallel or in series with it. This improvement will come at the cost of some accuracy. The value of the resistor should be equal to the thermistor resistance at the midpoint of the temperature range of interest.

Thermistor – parallel resistor combination

Figure 3 shows the S shaped temperature-resistance curve produced by placing a 10K resistor in parallel with a thermistor whose resistance is 10K at 25°C. This makes the region of the curve between 0°C and 50°C fairly linear. Note that maximum linearity is around the midpoint, which is at 25°C.

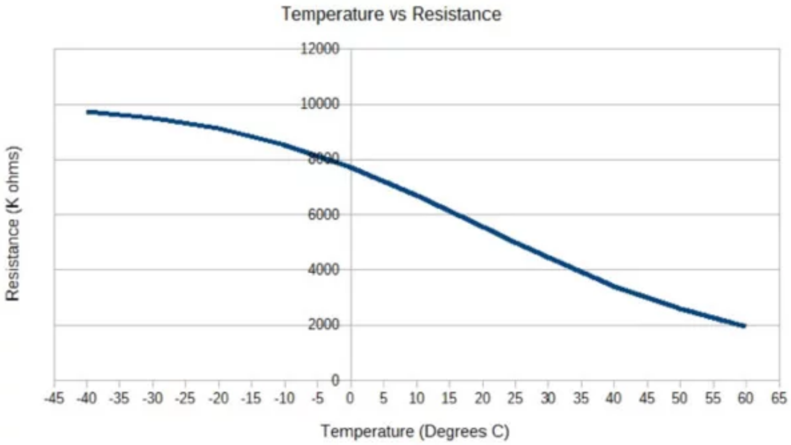


Figure 3: Temperature-resistance curve of thermistor and parallel resistor combination.

Thermistor - series resistor combination (voltage divider)

A common way for microcontrollers to capture analog data is via an analog to digital converter (ADC). You can't directly read the thermistors resistance with an ADC. The series thermistor-resistor combination, shown in figure 4, provides a simple solution in the form of a voltage divider.

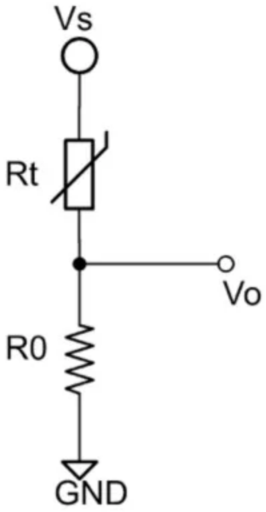


Figure 4: Thermistor voltage divider.

You use the following formula to calculate the voltage divider output voltage:

Vo = Vs * (R0 / (Rt + R0))

The linearized temperature-voltage curve in figure 5 shows the change in voltage divider output voltage Vo in response to temperature change. The source voltage Vs is 5 volts, the thermistor resistance Rt is 10K ohms at 25°C, and series resistor R0 is 10K ohms. Similar to the parallel resistor-thermistor combination above, this combination has maximum linearity around the mid point of the curve, which is at 25°C.

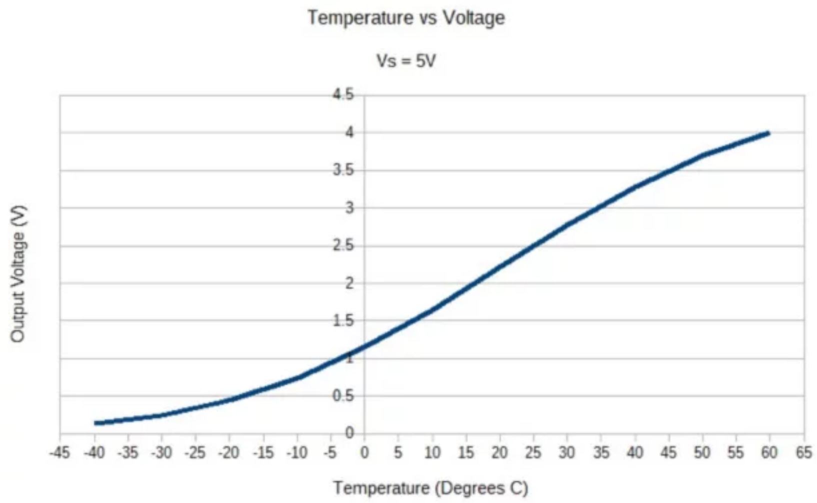


Figure 5: Temperature-voltage curve.

Note that, since Vs and R0 are constant, the output voltage is determined by Rt. In other words, the voltage divider converts thermistor resistance (and thus temperature) to voltage. Perfect for input to a microcontroller ADC.

Converting ADC Data to Temperature by first finding the thermistor resistance

To convert ADC data to temperature you first find the thermistor resistance and then use it to find the temperature.

You can rearrange the above voltage divider equation to solve for the thermistor resistance Rt:

Rt = R0 * ((Vs / Vo) - 1)

If the ADC reference voltage (Vref) and voltage divider source voltage (Vs) are the same then the following is true:

adcMax / adcVal = Vs / Vo

That is, the ratio of voltage divider input voltage to output voltage is the same as the ratio of the ADC full range value (adcMax) to the value returned by the ADC (adcVal). If you are using a 10 bit ADC then adcMax is 1023.

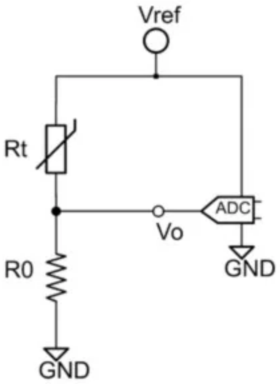


Figure 6: Voltage divider circuit and ADC with common reference voltage.

Now you can replace the ratio of voltages with the ratio of ADC values in the equation to solve for Rt:

Rt = R0 * ((adcMax / adcVal) - 1)

For example, assume a thermistor with a resistance of 10K ohms at 25°C, a 10 bit ADC, and adcVal = 366.

Rt = 10,000 * ((1023 / 366) – 1)
= 10,000 * (2.03)
= 17,951 ohms

Once you calculate the value for Rt, you can use a look-up table containing temperature-resistance data for your thermistor to find the corresponding temperature. The calculated resistance for the thermistor in the above example corresponds to a temperature of approximately 10°C.

9	18,670
10	**17,926**
11	17,214

The manufacturer's data sheet might not include all temperature-resistance values for the thermistor or you might not have sufficient memory to include all of the values in your look-up table. In either case you will need to include code to interpolate between the listed values.

By Calculating The Temperature Directly

Alternatively, you can use an equation that approximates the thermistors temperature response curve to calculate the temperature. For example, the widely used Steinhart-Hart equation shown below. It is not as exact as the manufacturer's resistance-temperature data. However, compared to other methods, it provides a much closer approximation of the thermistor's response curve over its operational range.

$1/T = A + B*\ln(R) + C*(\ln(R))^3$

The manufacturer may or may not supply values for the coefficients A, B, and C. If not, they can be derived using measured temperature-resistance data. However, that is beyond the scope of this article. Instead, we will use the simpler Beta (or B) parameter equation shown below. Though not as accurate as the Steinhart-Hart equation, it still provides good results over a narrower temperature range.

$1/T = 1/T0 + 1/B * \ln(R/R0)$

The variable T is the ambient temperature in Kelvin, T0 is usually room temperature, also in Kelvin (25°C = 298.15K), B is the beta constant, R is the thermistor resistance at the ambient temperature (same as Rt above), and R0 is the thermistor resistance at temperature T0. The values for T0, B, and R0 can be found in the manufacturer's data sheet. You can calculate the value for R as described previously for Rt.

If the voltage divider source voltage and Vref are the same you don't need to know R0 or find R to calculate the temperature. Remember you can write the equation for the thermistor resistance in terms of the ratio of ADC values:

$R = R0 * ((adcMax / adcVal) - 1)$

then:

$1/T = 1/T0 + 1/B * \ln(R0 * ((adcMax / adcVal) - 1) / R0)$

R0 cancels out, which leaves:

$1/T = 1/T0 + 1/B * \ln((adcMax / adcVal) - 1)$

Take the reciprocal of the result to get the temperature in Kelvin.

For example, assume a thermistor voltage divider circuit is connected to a 10 bit ADC. The beta constant for the thermistor is 3380, the thermistor resistance (R0) at 25°C is 10K ohms, and the ADC returns a value 366.

$1/T = 1/298.15 + 1/3380 * \ln((1023 / 366) - 1)$
$1/T = 0.003527$
$T = 283.52K - 273.15K = 10.37°C$

Example: A Simple Arduino Based Temperature Logger

Figure 7 shows a simple temperature logger consisting of an Arduino Uno SBC and a thermistor voltage divider (right). The voltage divider output is connected to Arduino's internal 10 bit ADC via one of the analog pins. The Arduino gets the ADC value, calculates the temperature, and sends it to the serial monitor for display.

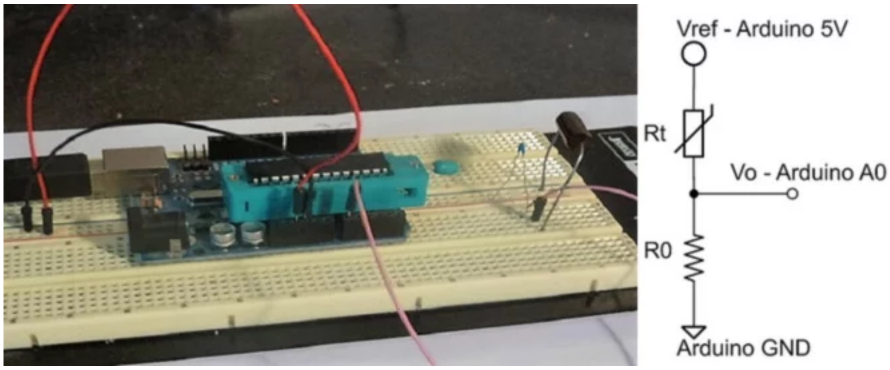
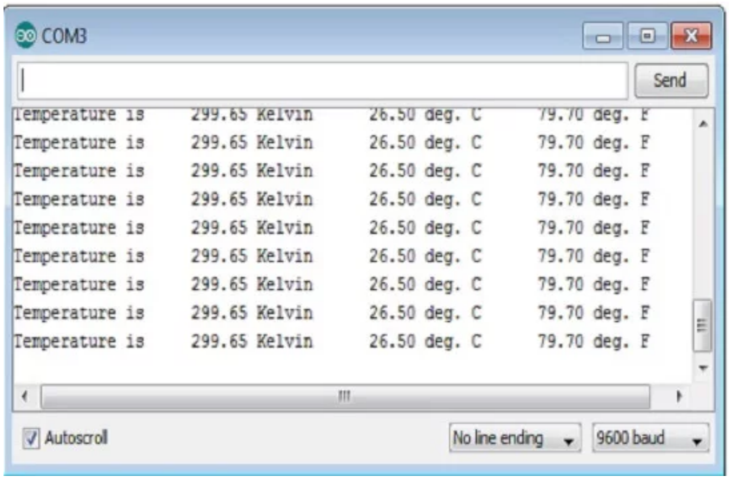


Figure 7: Arduino temperature logger circuit.

The following Arduino sketch uses the B parameter equation to calculate the temperature. The function getTemp does most of the work. It reads the analog pin multiple times and averages the ADC values. It then calculates the temperature in Kelvin, converts it to Celsius and Fahrenheit and returns all three values to the main loop. The main loop repeatedly calls getTemp, with a 2 second delay between calls. It sends the temperature values returned by getTemp to the serial monitor.



```
Temperature is    299.65 Kelvin     26.50 deg. C     79.70 deg. F
Temperature is    299.65 Kelvin     26.50 deg. C     79.70 deg. F
Temperature is    299.65 Kelvin     26.50 deg. C     79.70 deg. F
Temperature is    299.65 Kelvin     26.50 deg. C     79.70 deg. F
Temperature is    299.65 Kelvin     26.50 deg. C     79.70 deg. F
Temperature is    299.65 Kelvin     26.50 deg. C     79.70 deg. F
Temperature is    299.65 Kelvin     26.50 deg. C     79.70 deg. F
Temperature is    299.65 Kelvin     26.50 deg. C     79.70 deg. F
Temperature is    299.65 Kelvin     26.50 deg. C     79.70 deg. F
```

Figure 8: Screenshot of temperature logger output.

Download example code here.

```
void getTemp(float * t)
{

    // Converts input from a thermistor voltage divider to a temperature value.
    // The voltage divider consists of thermistor Rt and series resistor R0.
    // The value of R0 is equal to the thermistor resistance at T0.
    // You must set the following constants:
    //                  adcMax  ( ADC full range value )
    //                  analogPin (Arduino analog input pin)
    //                  invBeta  (inverse of the thermistor Beta value supplied by manufacturer).
    // Use Arduino's default reference voltage (5V or 3.3V) with this module.
    //

  const int analogPin = 0; // replace 0 with analog pin
  const float invBeta = 1.00 / 3380.00;   // replace "Beta" with beta of thermistor

  const  float adcMax = 1023.00;
  const float invT0 = 1.00 / 298.15;   // room temp in Kelvin

  int adcVal, i, numSamples = 5;
  float  K, C, F;

  adcVal = 0;
  for (i = 0; i < numSamples; i++)
   {
     adcVal = adcVal + analogRead(analogPin);
     delay(100);
   }
  adcVal = adcVal/5;
  K = 1.00 / (invT0 + invBeta*(log ( adcMax / (float) adcVal - 1.00)));
  C = K - 273.15;                      // convert to Celsius
  F = ((9.0*C)/5.00) + 32.00;   // convert to Fahrenheit
  t[0] = K; t[1] = C; t[2] = F;
  return;
}

void setup()
{
  analogReference(DEFAULT);
  Serial.begin(9600);
}

void loop()
{
  float temp[3];
  getTemp(temp);

  Serial.print("Temperature is     ");
  Serial.print(temp[0]); Serial.print(" Kelvin      ");
  Serial.print(temp[1]); Serial.print(" deg. C      ");
  Serial.print(temp[2]); Serial.print(" deg. F      ");
  Serial.println();
  delay(2000);
  return;
}
```

Measurement Error and ADC Resolution

There are a number of factors that can contribute to measurement error. For example, the thermistor and series resistors may vary from their rated values (within specified tolerance limits) or there can be error due to thermistor self-heating or a noisy electrical environment can result in fluctuations in input to the ADC[6].

Below are a few suggestions for reducing measurement error. This assumes you are using the B parameter equation.

- Measure series resistor (R) to get the actual resistance and use that value in your temperature calculation.
- Similarly, if possible, measure the actual resistance (R0) for your thermistor at T0 and use that value in your calculations. You will need an accurate thermometer, an accurate resistance meter and a way to produce the desired temperature.
- Alternatively, you can select a thermistor and a resistor with tighter tolerances to reduce the error to a level acceptable for your application.
- It turns out that the Beta constant isn't really constant. The value depends on temperature and, as mentioned earlier, is usually given for a specified temperature range. If you can get accurate thermistor resistance values at two temperatures in the range of interest (preferably at the endpoints) you can use the following formula [2] to find the actual beta constant for your thermistor.

  B = ln(R1/R2) * (t1*t2)/(t2-t1)
- Keep power dissipation as low as possible to avoid error due to self-heating.
- Take a number of successive ADC readings and use the average of these readings in your temperature calculation.
- Use a filtered voltage source for Vref.

ADC Resolution

At best, the temperature in the above example is accurate to the nearest .1°C. This is because of the limitation due to ADC resolution.

The ADC is not sensitive to voltage changes between steps. For a 10 bit ADC the smallest voltage change that can be measured is Vref/1023. This is the voltage resolution of the ADC. If Vref is 5V the voltage resolution is 4.89 mV. Assuming T0 is 25°C the smallest temperature change that can be detected at 25°C is ±0.1°C. This is the temperature resolution at 25°C. What this means is that a change in the least significant bit will cause the displayed temperature to jump by 0.1°C. This jump is due to ADC resolution not measurement error.

ADC	Output	Temp
511	0111111111	24.95°C
512	1000000000	25.05°C
513	1000000001	25.15°C

If you need better resolution there are techniques (e.g. oversampling [1]) that you can use to increase the effective resolution of your microcontroller's ADC or you can use an external ADC with higher resolution.

References

1. AVR121: Enhancing ADC Resolution by Oversampling
   http://www.atmel.com/Images/doc8003.pdf
2. How To Find An Expression For Beta
   http://www.zen22142.zen.co.uk/ronj/tyf.html
3. Temperature Measurement with a Thermistor and an Arduino
   http://web.cecs.pdx.edu/~eas199/B/howto/thermistorArduino/thermistorArduino.pdf
4. Thermistor
   https://en.wikipedia.org/wiki/Thermistor
5. Thermistor Tutorial
   http://www.radio-electronics.com/info/data/resistor/thermistor/thermistor.php
6. Understanding and Minimising ADC Conversion Errors
   http://www.st.com/content/ccc/resource/technical/document/application_note/9d/56/66/74/4e/97/48/93/CD00004444.pdf/files/CD00004444.pdf/jcr:content/transla

If you have an electronics story you'd like to share, please send it to **MyStory@Jameco.com**.

For almost two decades, Phil Kane has been a technical writer in the software industry and occasionally authored articles for electronics enthusiast magazines. He has a bachelor's in Electronics Engineering Technology with a minor in Computer Science. Phil has had a life-long interest in science, electronics and space exploration. He enjoys designing and building electronic gadgets, and would very much like to see at least one of those gadgets on its way to the moon or Mars one day.

Jameco Electronics
1355 Shoreway Road
Belmont, CA 94002

CUSTOMER CARE

1-800-831-4242

Privacy Policy
Careers
Terms & Conditions
Get Directions

Home
About Us
FAQ/Help
Contact Us
Site Map

Power Resource Center
Resource Center
Gift Center
Clearance Center
Product Index
Product Catalog Locator
Proposition 65 Warning