Presented here is a spreadsheet to help you generate (C) code for easily converting an ADC (Analog to Digital Converter) value, as for example returned by the analogRead() function of Arduino, into a temperature in °C.

When you give it just a few simple parameters it will produce:

- A table listing temperature, resistance and the ADC (analogRead) result for a range of values determined by you.
- Three C (language) functions for converting an ADC result into a temperature in °C, suitable for Arduino use
- A "standard" Beta approximation producing good results across the whole range
- A floating point linear approximation producing good results across a narrower (configurable) range
- An integer-only linear approximation producing reasonable results across a narrower (configurable) range
- Charts showing the relative error-margin for the three different C functions to help you tune your values
- Wiring "diagram" is included in the comments of the generated C (language) code

Additionally it incorporates the following helpful functions:

- If you don't know, or wish to calibrate yourself, the Beta value of your thermistor, it can calculate it from just one measurement (measure the resistance at a known temperature)
- It can select an appropriate value for the paired resistor in the wiring for you to minimise the errors
- The ability to test your settings by converting a measured temperature into a resistance which you can check with your multimeter.

The spreadsheet is in Google Docs, create your own copy here to use the NTC spreadsheet.

Comments/Feedback/Discussion/Help

Head over to this Reddit thread and have at it.

https://www.reddit.com/r/electronics/comments/3z8rpq/ntc_thermistors_and_you_converting_an_adc/

Some brief screenshots and instructions (there's lots of info in the spreadsheet)

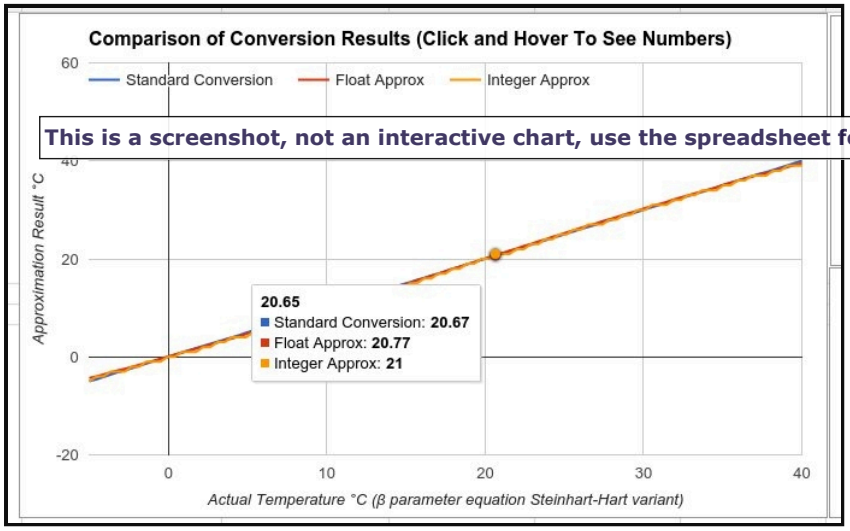
When you open the spreadsheet you will see the settings on the right side of the sheet (horizontal scroll across if you don't see them!), note that there are lots of comments there in the green next to the settings, and if you scroll even further right there is even more information in blue.

The settings you will most likely want to adjust are "Beta Value To Use", "Nominal Resistance", Low °C and High °C. The Beta Value and Nominal Resistance you obtain from your thermistor's datasheet or vendor, the Low and High °C are your choice and help the calculations provide results that are of good accuracy in the range you particularly are interested in.

If you don't know the Beta value, or want to re-calibrate it yourself, simply type =O7 into the "Beta Value To Use" field, and then enter a Measured Resistance and Measured Temperature (usually done at 50°C), the Beta will be computed then.

Settings	See More Info >>>	
Thermistor Characteristics		**See More Info ==>**
Calculated Beta Value	3949.049047	
Beta Value To Use	3435	Set to a known Beta, or set to =O7 to calculate the Beta
Nominal Resistance	10000	10k thermistor has 10000 (Ω) Nominal Resistance
Nominal Temperature	25	Usually 25 (°C) if not specified by thermistor datasheet
Measured Resistance	5461	For calculating beta, measure the thermistor's resistance...
Measured Temperature	50	... when at a known measured temperature (usually 50°C)
Thermistor Connection and Verification		**See Also Wiring Diagram and Tips ==>**
Thermistor Vcc/Gnd	Vcc	Is Thermistor connected to "Gnd" or "Vcc"
ADC Max Value	1023	Max value returned by ADC (analogRead), 1023 typically
Other Resistor	13462	The value of the resistor paired with the thermistor.. **SEE >>**
Measured Temperature	0	Test your settings with a known measured temperature,
Calculated Resistance	28704.29039	
ADC (analogRead) Result	327	
Table Generation Parameters		
Starting Temperature	-23	Controls the range of the values in the lookup table
In Steps Of	1	generated (<-- over there,)
Linear Approximation of ADC Result to °C		**See More Info ==>**
Low °C	-10	Enter the band of temperatures you are interested in, simple
High °C	45	formulae for approximating temperature from ADC values are
Slope	0.1014983864	shown.
Intercept	-33.63307818	
Float Math Approximation		The value of the "Other Resistor" (set it above) has a large
°C = Slope * analogRead(x) + Intercept		effect on the error, set it to `=B105*1000` to get a good
Max Error	2.306 °C	resistance for your range.
Mean Error	0.6231188119	
Median Error	0.541	The C code generated underneath the lookup table includes
Integer Math Approximation		functions for using these approximations -
`Multiplier`	32	approximateTemperatureFloat() and
°C = ((((analogRead(x)*32) / 315) * 32) + -1076) / 32		approximateTemperatureInt().
Max Error	3.45 °C	
Mean Error	0.7168316832	The float approximation is closer, the int approximation
Median Error	0.5	doesn't use floating point math so it's much lighter-weight.

Charts are shown below the settings to give you an idea of the error-margins for the three methods.

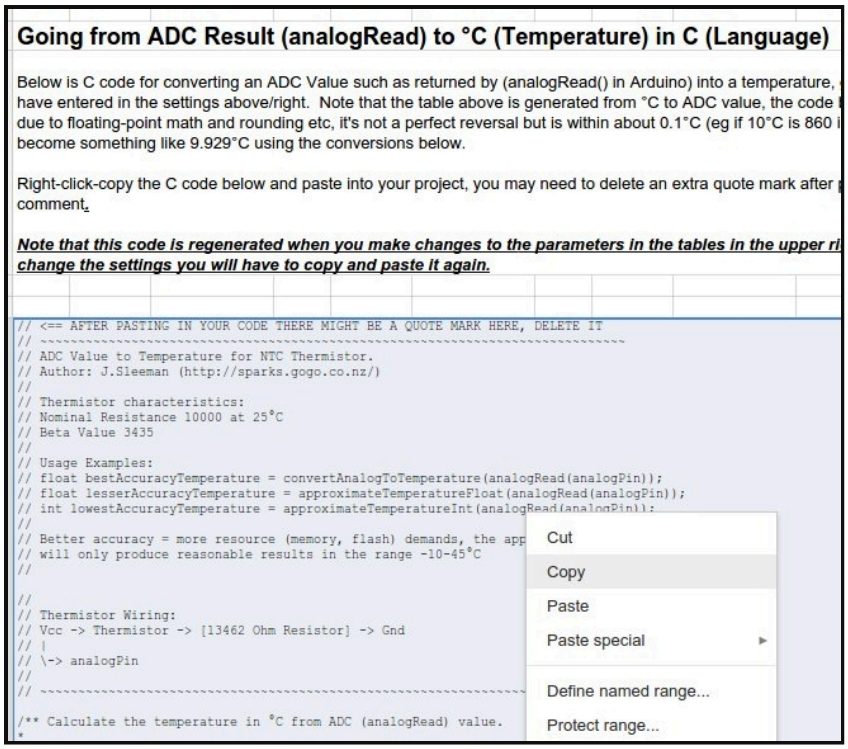


;

A table is generated on the left of the spreadsheet from the settings you provide.

°C	kΩ	ADC Result (analogRead)	°C	kΩ	ADC Result (analogRead)	°C	kΩ	ADC Result (analogRead)	°C	kΩ	ADC Result (analogRead)
-23	91.222	132	14	15.548	475	51	3.969	790	88	1.340	930
-22	86.369	138	15	14.916	485	52	3.842	796	89	1.305	933
-21	81.809	145	16	14.313	496	53	3.719	802	90	1.272	935
-20	77.523	151	17	13.739	506	54	3.601	807	91	1.239	937
-19	73.492	158	18	13.192	517	55	3.488	812	92	1.208	939
-18	69.701	166	19	12.669	527	56	3.379	818	93	1.177	941
-17	66.132	173	20	12.171	537	57	3.274	823	94	1.147	943
-16	62.771	181	21	11.696	547	58	3.172	828	95	1.118	945
-15	59.606	188	22	11.242	557	59	3.075	833	96	1.091	946
-14	56.623	196	23	10.809	567	60	2.981	838	97	1.063	948
-13	53.810	205	24	10.395	577	61	2.890	842	98	1.037	950
-12	51.157	213	25	10.000	587	62	2.803	847	99	1.012	951
-11	48.654	222	26	9.622	597	63	2.719	851	100	0.987	953
-10	46.290	230	27	9.261	606	64	2.638	855	101	0.963	955
-9	44.058	239	28	8.916	615	65	2.559	860	102	0.940	956
-8	41.950	249	29	8.585	625	66	2.484	864	103	0.917	958
-7	39.957	258	30	8.269	634	67	2.411	868	104	0.895	959
-6	38.072	267	31	7.967	643	68	2.341	871	105	0.874	961
-5	36.290	277	32	7.678	651	69	2.273	875	106	0.853	962
-4	34.603	287	33	7.400	660	70	2.207	879	107	0.833	963
-3	33.006	296	34	7.135	669	71	2.144	882	108	0.814	965
-2	31.494	306	35	6.881	677	72	2.083	886	109	0.795	966
-1	30.062	316	36	6.637	685	73	2.024	889	110	0.776	967
0	28.704	327	37	6.403	693	74	1.967	893	111	0.758	968
1	27.417	337	38	6.179	701	75	1.912	896	112	0.741	970
2	26.197	347	39	5.965	709	76	1.858	899	113	0.724	971
3	25.039	358	40	5.759	716	77	1.807	902	114	0.708	972
4	23.940	368	41	5.561	724	78	1.757	905	115	0.692	973
5	22.897	379	42	5.372	731	79	1.709	908	116	0.676	974
6	21.906	389	43	5.189	738	80	1.662	911	117	0.661	975
7	20.964	400	44	5.015	745	81	1.617	913	118	0.646	976
8	20.070	411	45	4.847	752	82	1.574	916	119	0.632	977
9	19.219	421	46	4.686	759	83	1.532	918	120	0.618	978
10	18.410	432	47	4.531	765	84	1.491	921	121	0.604	979
11	17.641	443	48	4.382	772	85	1.451	923	122	0.591	980
12	16.909	453	49	4.239	778	86	1.413	926	123	0.578	981
13	16.212	464	50	4.101	784	87	1.376	928	124	0.566	982

C (language) code is produced from your settings, simply right-click the cell, select copy and then paste into your program (or sketch), make sure to delete the extraneous quote mark in your program after pasting, as is noted in the comment :-)



Theory of Operation, Credits Go To

This stuff is extra learning for those who are interested (nobody), and really just notes for myself when I come back to look at this in a couple years and forgot how it all works, You don't need to know any of this to use the spreadsheet or generated C (language) code.

The formula for calculating (closely approximating) temperature to/from a resistance is a well known equation…

https://en.wikipedia.org/wiki/Thermistor#B_or_.CE.B2_parameter_equation

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{B} \ln\left(\frac{R}{R_0}\right)$$

Where T is temperature, B is Beta, To is nominal temperature and Ro is nominal resistance, temperatures are in Kelvin (K = C + 273.15)

Beta is usually a value provided in the NTC datasheet or by the NTC vendor, but in some case you might not know the Beta (reliably) and want to calculate it, we can rewrite the above formula to solve for B when it is unknown and we measure a specific resistance (R) at a specific known temperature (T) (in addition to the nominal To (nominal temperature, usually 25°C) and Ro (resistance of thermistor at that temperature, eg 10000 for a 10k thermistor). Note that usually the measurement for R and T is taken at 50°C, but you can do so at a different point if you like, it just affects the Beta value result slightly.

1/T = 1/To + 1/B * ln(R/Ro)

1/T - 1/To = 1/B * ln(R/Ro)

(1/T - 1/To) / ln(R/Ro) = 1/B

1/ (1/T - 1/To) / ln(R/Ro)) = B

You can see this implemented in cell O7
Once we have a Beta (given or calculated) then we again use this equation, this time solved for "R" the resistance of the thermistor given a known temperature in cell O25 of the spreadsheet (and in the resistance columns of the table), the algebraic manipulation being…

1/T = 1/To + 1/B * ln(R/Ro)

1/T = 1/B * ln(R/Ro) + 1/To [Rearrange for clarity]

1/T − 1/To = 1/B * ln(R/Ro)

1/T − 1/To = ln(R/Ro) / B [1/B * N == N / B]

B * (1/T − 1/To) = ln(R/Ro)

exp(B * (1/T − 1/To)) = R/Ro [exp is inverse of natural log]

Ro * exp(B * (1/T − 1/To)) = R

[2017-06-08 : there was an error in the working for the notes above and these have been corrected, the spreadsheet was correct]

Going from a calculated resistance to an ADC result is pretty straight forward (if you know the max adc result, and you know both resistances (resistor and thermistor) and know know which order they are in (resistor-thermistor or thermistor-resistor) then you know the ratio between them and get the ADC result, no need to care about what the voltage/current is), this is my own concoction…

[for Thermistor connected to Gnd] adcVal = Rth /(Rother + Rth) * adcMax

[for Thermistor connected to Vcc] adcVal = adcMax - (Rth /(Rother + Rth) * adcMax)

For which you can see the implementation in the ADC Result cells of the results table, note that the values are rounded in implementation, because that's how an ADC works.

So at this point we have formulae to get us from known temperature to resistance to ADC value, but in practice we need to go the opposite direction.

Reversing the adc equations to solve for resistance with a known adcVal required the help of Wolfram Alpha's solve-for-x ability with the following equivalent equation (because my algebra was too rusty for this…)

[for Thermistor connected to Gnd] solve for x, v = (((x)/(u + (x)) * m))

[for Thermistor connected to Vcc] solve for x, v = m - (((x)/(u + (x)) * m))

which Wolframp Alpha helpfully solved to

[for Thermistor connected to Gnd] x = (uv) / (m-v)

[for Thermistor connected to Vcc] x = (u (m - v)) / v

which more readibly is

[for Thermistor connected to Gnd] Rth = ((Rother * adcVal) / (adcMax - adcVal))

[for Thermistor connected to Vcc] Rth = ((Rother * (adcMax - adcVal)) / adcVal)

So now if we have an adcVal we can get a resistance, and we have the same standard beta-approximation-of-temperature equation above

$$\frac{1}{T} = \frac{1}{T_0} + \frac{1}{B} \ln\left(\frac{R}{R_0}\right)$$

which we can rewrite once again to solve for T (and rename some variables for clarity) to

T = 1/((1/Tnominal) + ln(Rth/Rnominal)/B)

Where: T = Temperature (K), Tnominal = Nominal Temperature, Rth = Thermistor Resistance, Rnominal = Nominal Resistance, B = Beta

and insert our Rth equation that Wolfram Alpha made for us to bring us to

[for Thermistor connected to Gnd] T = 1/((1/Tnominal) + ln(((Rother * adcVal) / (adcMax - adcVal))/Rnominal)/B)
[for Thermistor connected to Vcc] T = 1/((1/Tnominal) + ln(((Rother * (adcMax - adcVal)) / adcVal)/Rnominal)/B)

note these temperatures are in Kelvin, in the spreadsheet and code we convert to/from C in these equations by adding/subtracting 273.15 as appropriate.

Those last two formulae are the crux of converting from an ADC result (analogRead) into a temperature, you'll see you need to know the nominal resistance (eg, 10k thermistor), the nominal temperature (at which the thermistor has that nominal resistance, usually 25 degrees C), the size of the "other" resistor the thermistor is connected in series with, the maximum adc value (adcMax) possible (1023 for a 10bit adc for example), the adc result (adcVal) you want to convert, and the beta value. All the values can be hard coded except for the adc result.

However due the need to use log functions, floating point, division and so forth with that, which might not be possible on a small microcontroller, linear approximations are also calculated. This is done by creating a list (A55:C155) of temperature and calculated ADC value pairs, then using the "SLOPE" and "INTERCEPT" spreadsheet functions to create those two values which can be used in the formula shown in cell N:O33 - with thanks to RHETT ALLAIN from Wired.Com this remarkably closely approximates the "proper" beta-approximation (approximation of an approximation of an approximation…)

That however still requires floating point operations and just truncating them into integer operations wouldn't work at all, so the formula for the float approximation is adjusted to use only pre-calculated integer values, a multiplier factor is included which has the tendency to influence the error margin of the integer approximation in useful ways, this was a happy accidental discovery :-).

This formula is displayed in N:O39. It is then applied to all the ADC Values C55:C155 with results in the cells K55:K155, from this we calculate the error for each result in N:55:N155. This error is averaged in V53, it is preferable that this average is zero, indicating that the errors are evenly distributed above and below the "no-error" line, if there are sufficiently more positive errors than negative errors, or vice-versa, then this value in V53 will be non-zero (it's rounded, this is an integer approximation remember). To counteract such imbalance this imbalance is subtracted, which you can see in the last term of the formula in N:O39, this has the effect of shifting the approximation and reducing the error (significantly) when there is imbalance.

The charts are produced from the various data in the calculations found in rows 53 onwards.

The "Other Resistor" suggestions for 1% and 5% are made possible from the spreadsheet formulae found at

http://electronicdesign.com/components/excel-formula-calculates-standard-1-resistor-values

which convert a given value into the closest E96/48/24 series resistor.