CALIBRAÇÃO DE UM TERMISTOR DE NTC 10K ATRAVÉS DE UMA PLACA DE ARDUÍNO

Felipe Abdalla Tiradentes de Souza,
Nilo Sérgio Souza de Almeida

Instituto de Física/Universidade de São Paulo

felipe.abdalla.souza@usp.br
nilo.almeida@usp.br

1. Introdução

Um termistor é um tipo de resistor em que a resistência depende da temperatura do meio em que se encontra.

$$R = R(T)$$

O NTC (*Negative Temperature Coefficient*) se caracteriza por ter o coeficiente negativo, ou seja, quando a variação da temperatura é positiva, a variação da resistência é negativa[1].

$$\frac{dR}{dT} < 0$$

Existem diversos sistemas que utilizam termistores como uma válvula para ativar circuitos internos como:

1. **Ar condicionado**: controle de temperatura;
2. **Motores e geradores**: o termistor monitora um possível superaquecimento que, caso ocorra, ativa um circuito de proteção;
3. **Chips eletrônicos**: todo chip possui um dissipador de calor para evitar o superaquecimento do sistema, porém dificilmente a temperatura ambiente é a prevista pelo projetista, logo um termistor é necessário para uma melhor regulação da temperatura interna do sistema.

2. Objetivos

O principal objetivo deste projeto é a calibração de um termistor de NTC através de uma coleta de dados utilizando uma placa de Arduíno e um outro termistor, LM35, que possui uma leitura mais acurada a procura de uma alternativa econômica em sistemas que utilizam vários termistores.

Figura 1: NTC 10k

Figura 2: LM35DZ

3. Métodos e procedimentos

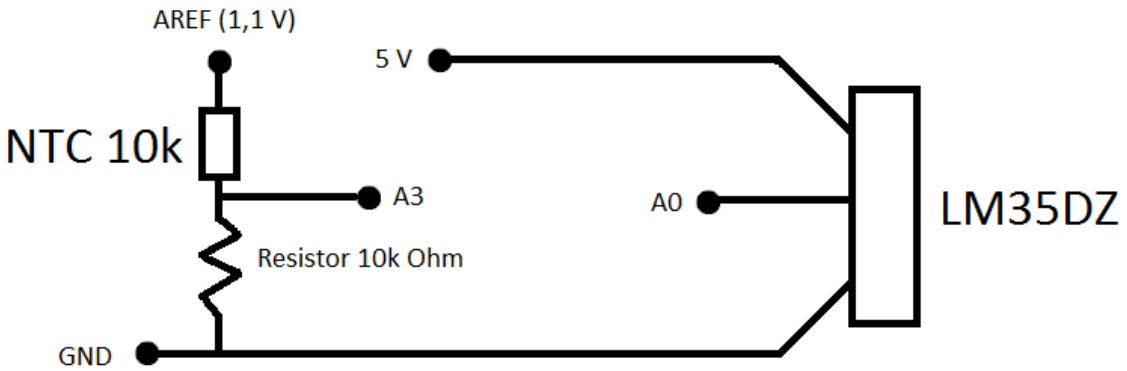
Para determinar a temperatura lida pelo NTC devemos utilizar a Equação de Steinhart-Hart, que é geral para qualquer termistor

$$\frac{1}{T} = A + B\ln(R) + C(\ln(R))^3$$

A ideia da calibração é determinar as constantes A, B e C tais que a leitura da temperatura do NTC seja igual à do LM35. Na prática, substituímos T pela temperatura lida pelo LM35 e R pela resistência medida do NTC.

Existe uma forma simplificada da equação de Steinhart-Hart que assume o coeficiente C nulo[2], porém vamos utilizar a forma padrão e verificar se tal hipótese é válida.

Foi montado um circuito como ilustrado abaixo



Utilizamos como meio para variação térmica a água, mas como os termistores incluídos no kit básico de Arduíno não permite a imersão dos mesmos em um outro meio por não alcançarem qualquer recipiente, nós utilizamos termistores NTC e LM35 com terminais soldados e envoltos em um material isolante.

Para preparar o meio, aquecemos por volta de 230 ml de água da torneira utilizando um fogão elétrico e uma panela. Ao atingir o ponto de ebulição, transferimos a água da panela

a um copo de vidro e colocamos ambos termistores a mesma altura dentro de copo e ligamos a placa de Arduíno para a coleta de dados. Todos os conjuntos de dados foram obtidos utilizando o mesmo procedimento.

4. Resultados

Utilizando um *script* compatível com Matlab e Octave nós transformamos as informações em bits obtidas pela placa de Arduíno em dados de resistência e temperatura. Como a quantidade de dados é da ordem de trezentos mil, utilizamos um filtro que utiliza um determinado intervalo (janela) e substitui tais pontos pela média dos mesmos. Superpormos os dados de temperatura por resistência de cada conjunto de dados.

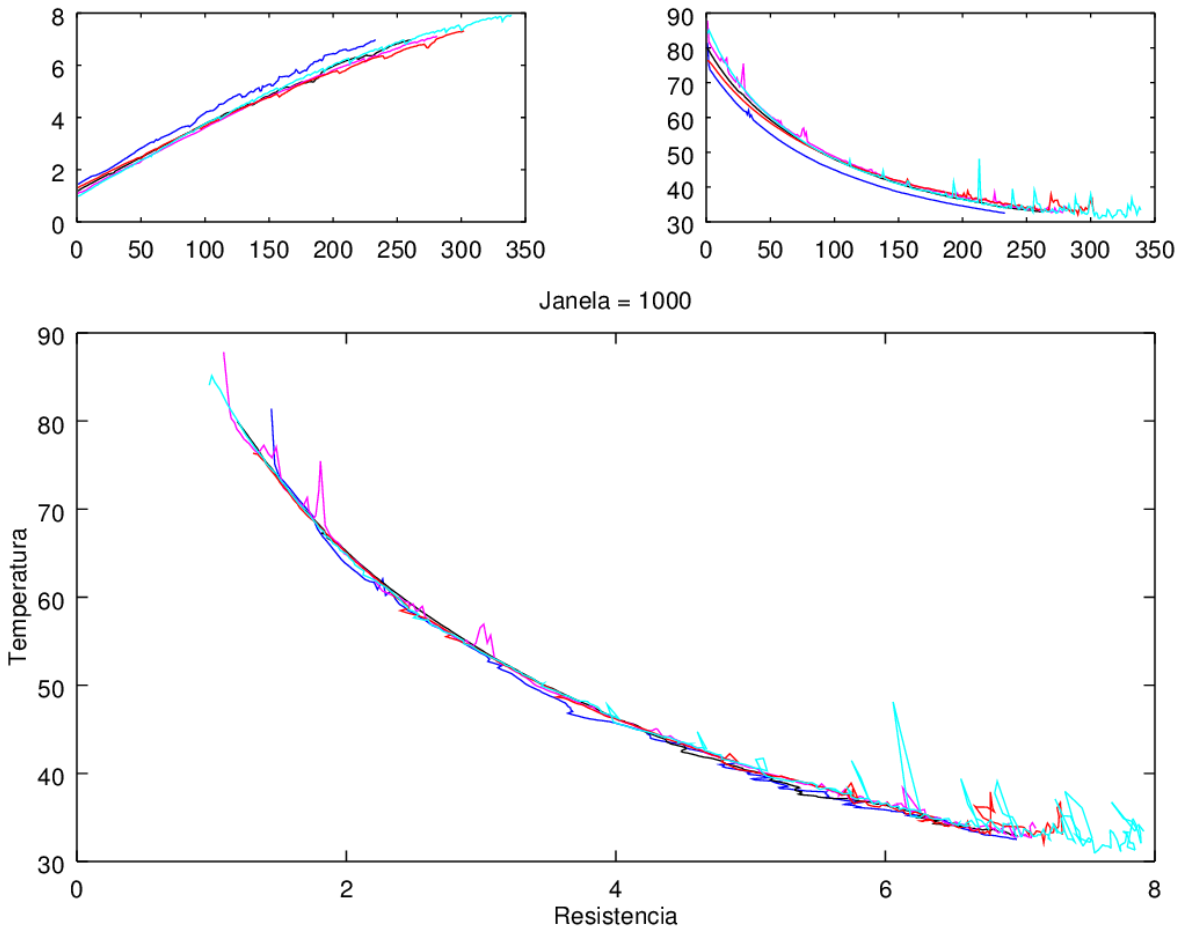


Figura 3: Exemplo de uma janela de 1000 dados

É visível que uma quantidade considerável de dados possuía erros grosseiros, provavelmente por causa de uma possível falha no isolamento dos terminais dos termistores, então nossa solução foi ir em cada arquivo de dados e excluir dados cuja variação é característica de erros grosseiros e refizemos a superposição de dados.

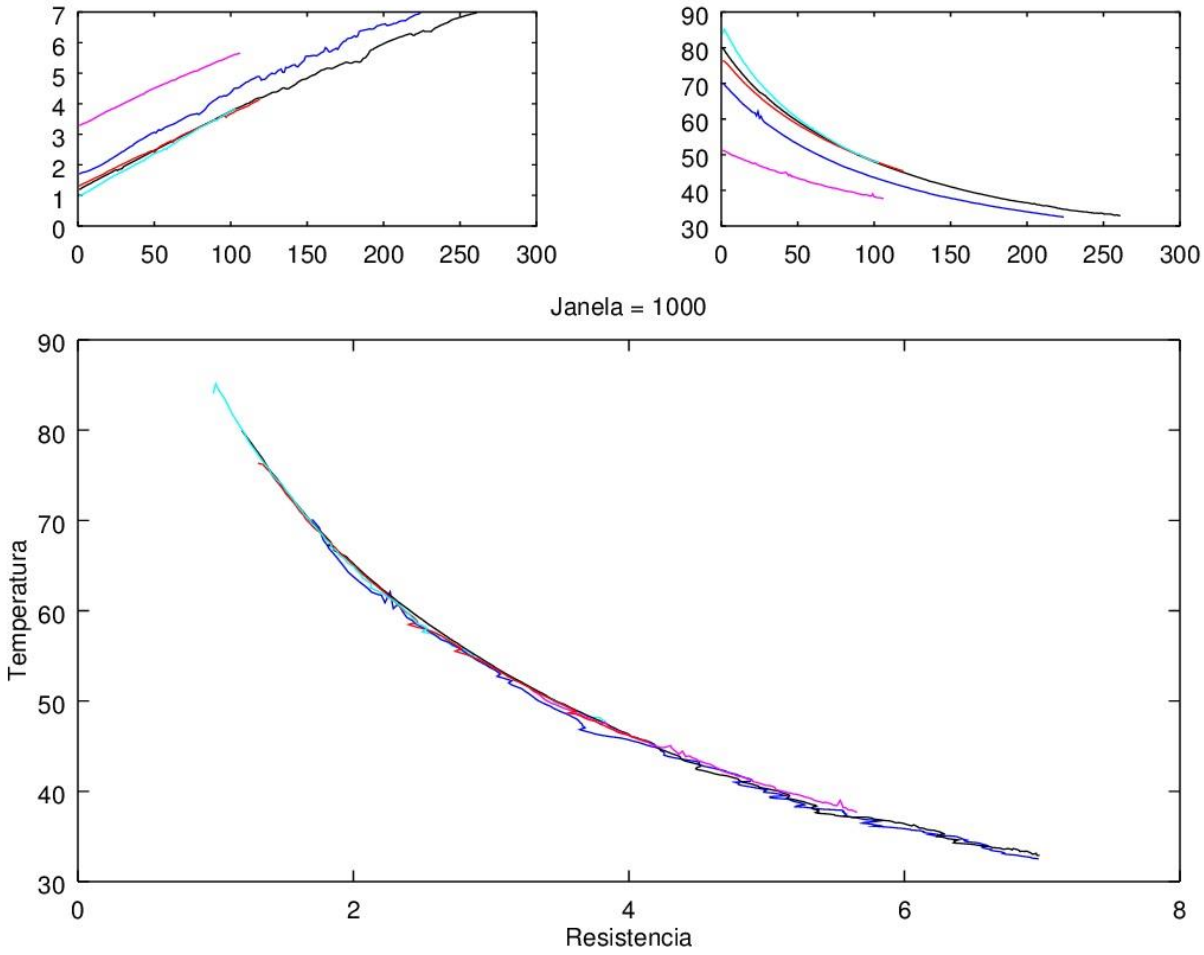


Figura 4: Exemplo de uma janela de 1000 dados

Além de dados coerentes, devemos escolher qual a melhor janela de dados a se trabalhar e para isso fizemos análises do valor do χ^2_{red} e da curtose da distribuição dos dados em janelas diferentes em função do número de dados da janela.

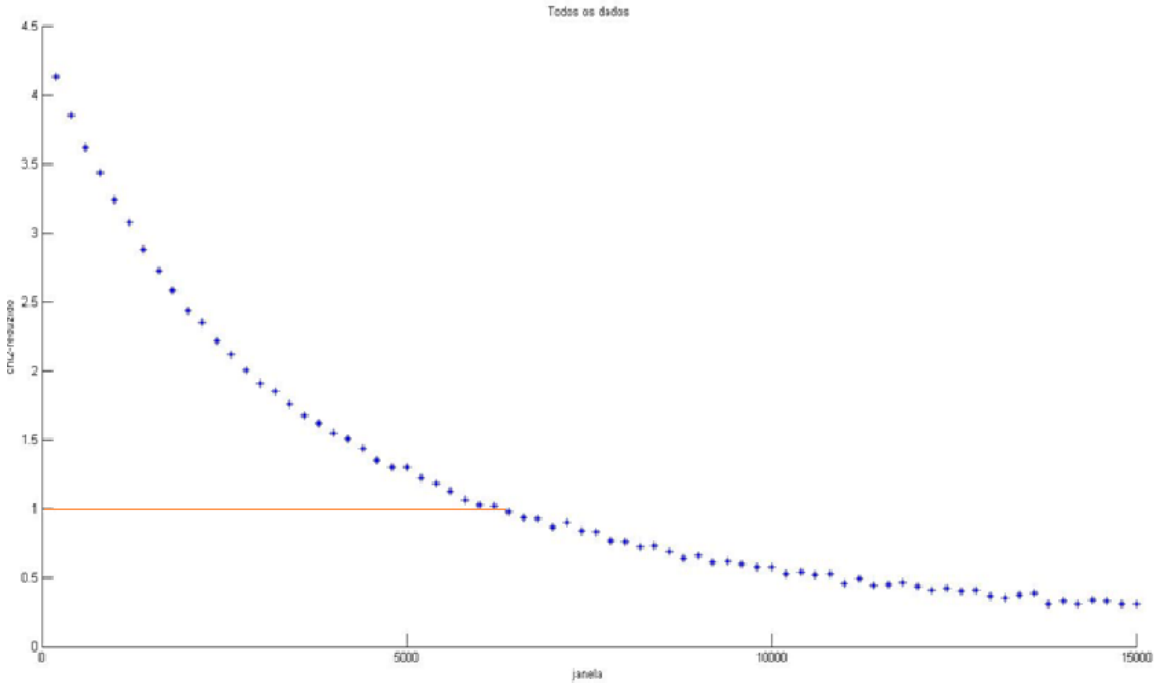


Figura 5: χ^2_{red} por janela

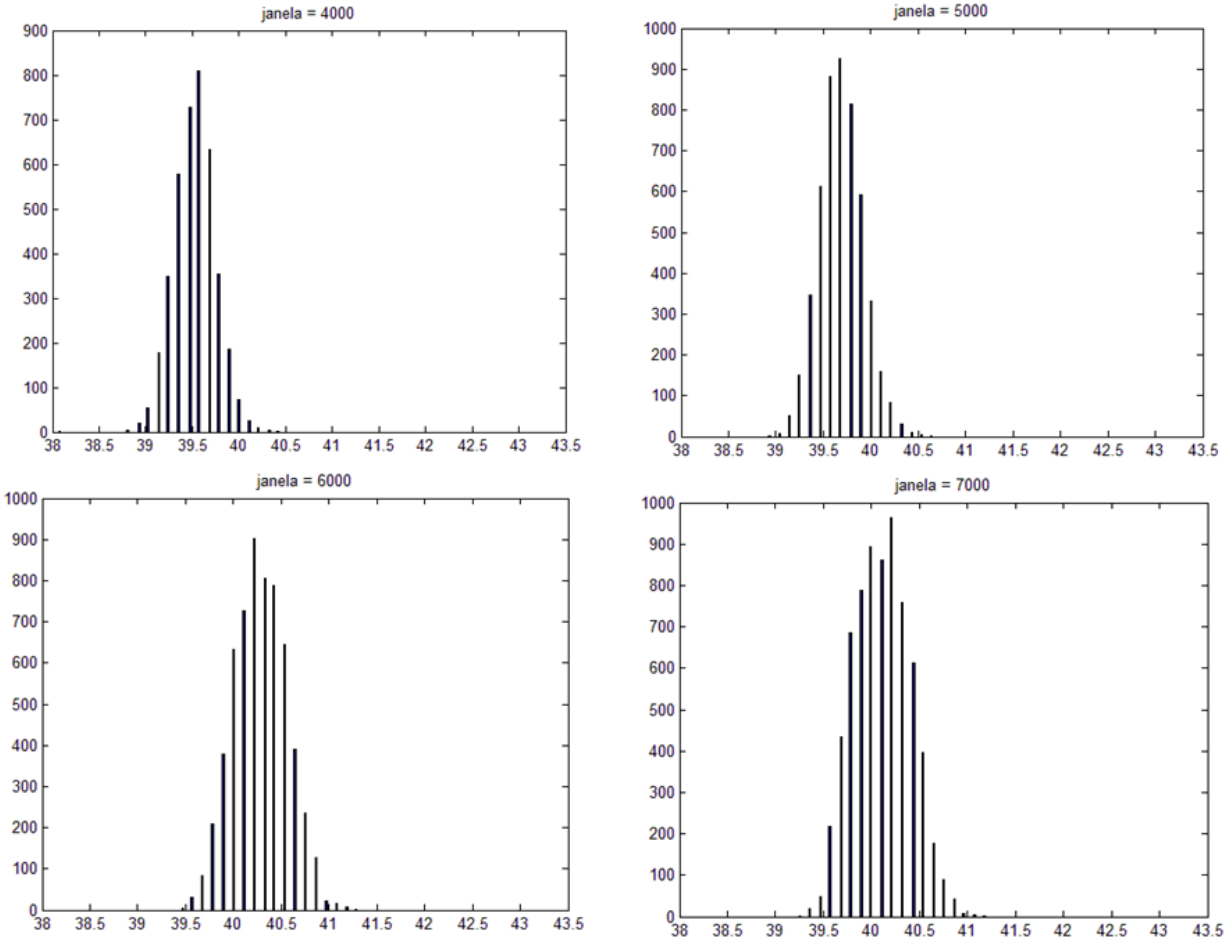


Figura 6: Exemplos de distribuição de dados em janelas diferentes

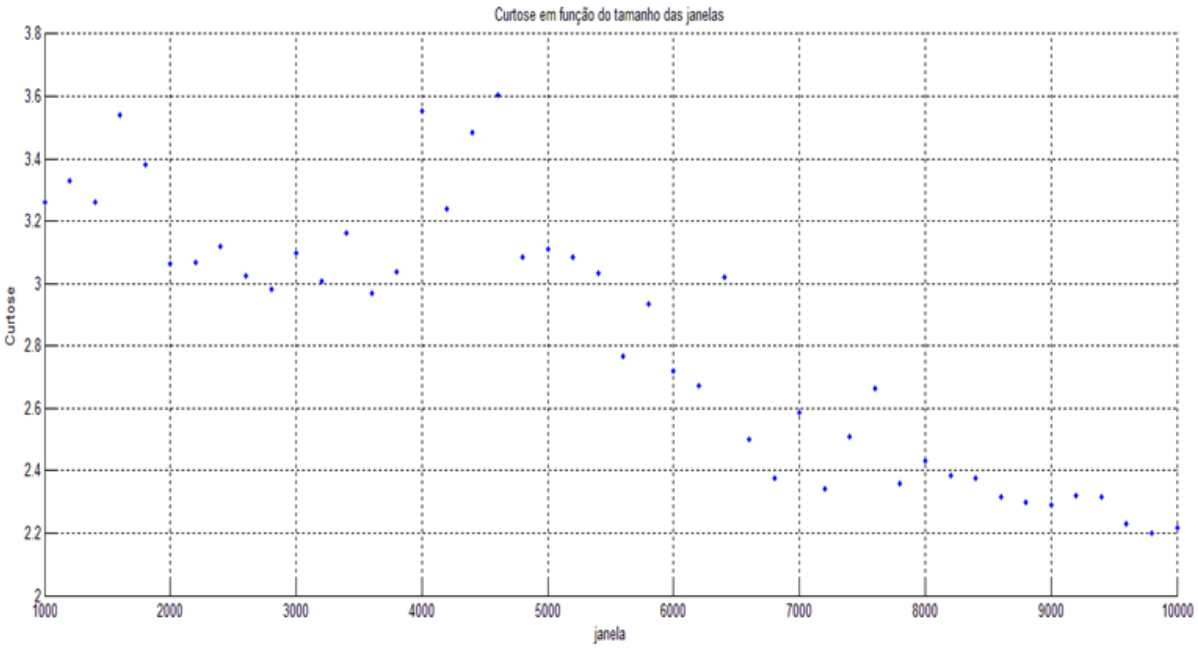
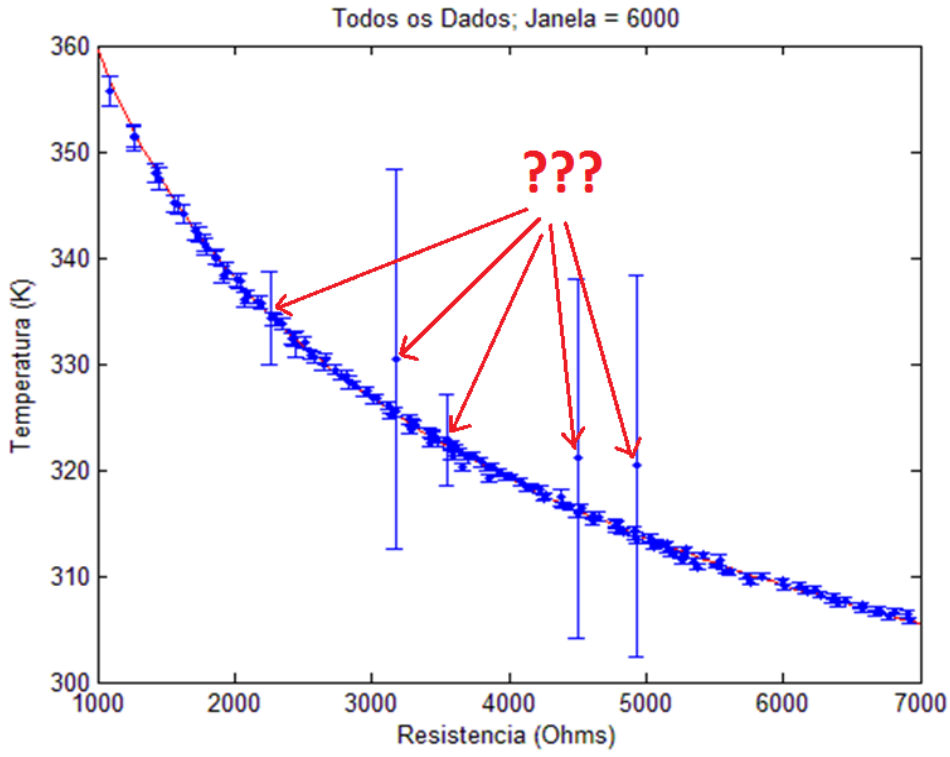
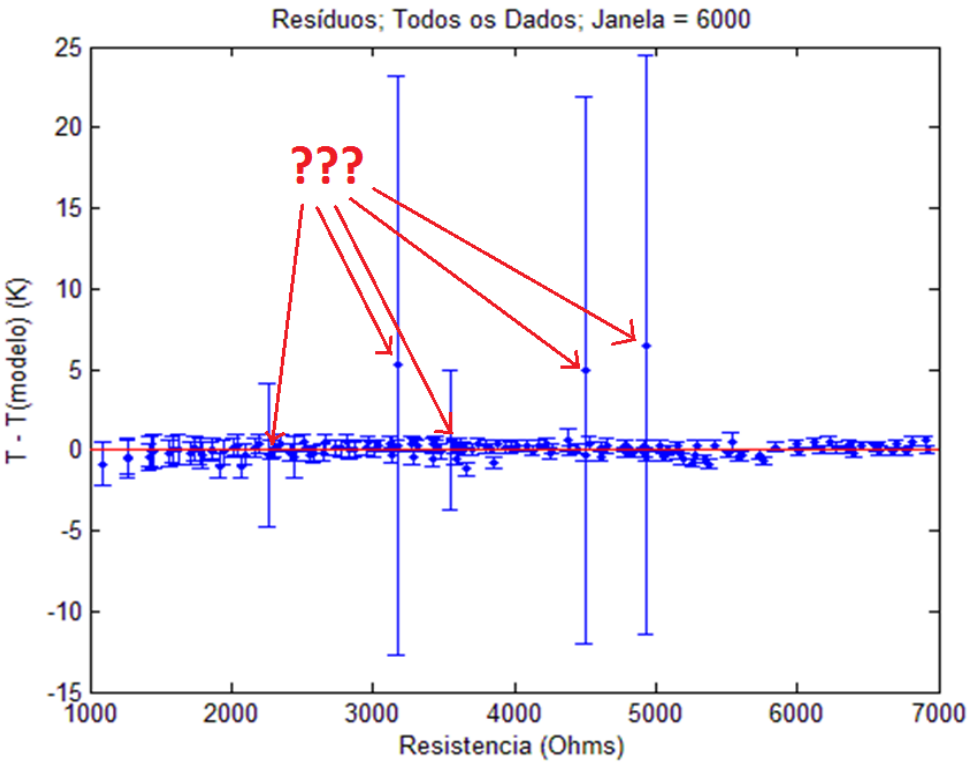


Figura 7: Curtose das distribuições anteriores em função da janela

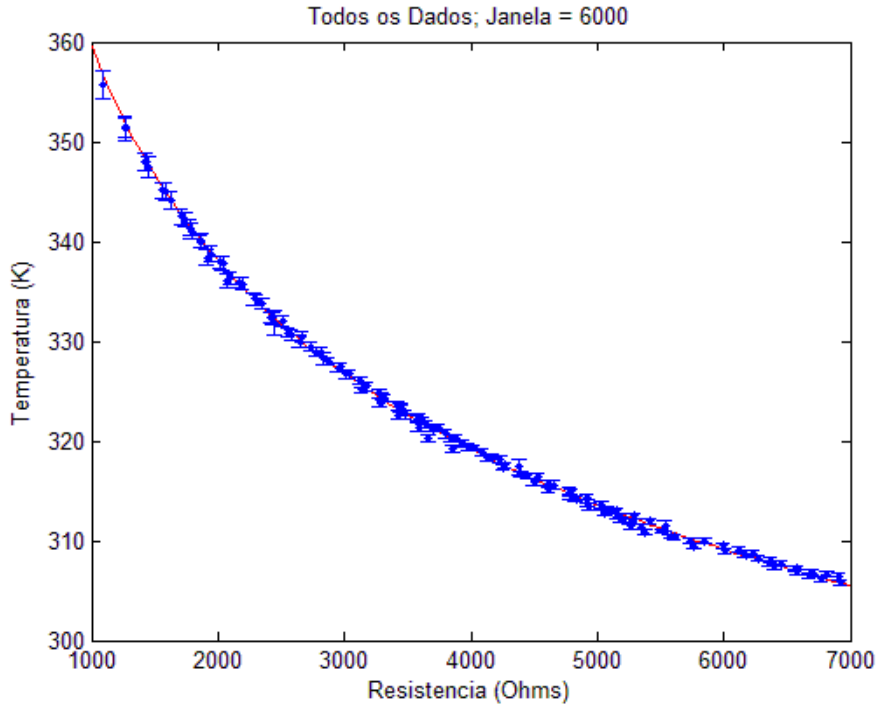
Como o modelo apresentado é geral para qualquer termistor, optamos por usar a janela que melhor enquadra os dados no modelo, ou seja, a janela de 6000 dados que nos dá um valor de χ^2_{red} de 1,02. A análise da curtose nos mostra que a partir da janela de 6000 dados a curtose das distribuições passa a decrescer do valor da curtose gaussiana que é 3, passando a se parecer mais com uma distribuição linear. Concluímos assim que nossa janela de escolha está dentro do intervalo em que a curtose é maior ou igual a 3 e nos dá um χ^2_{red} satisfatório.

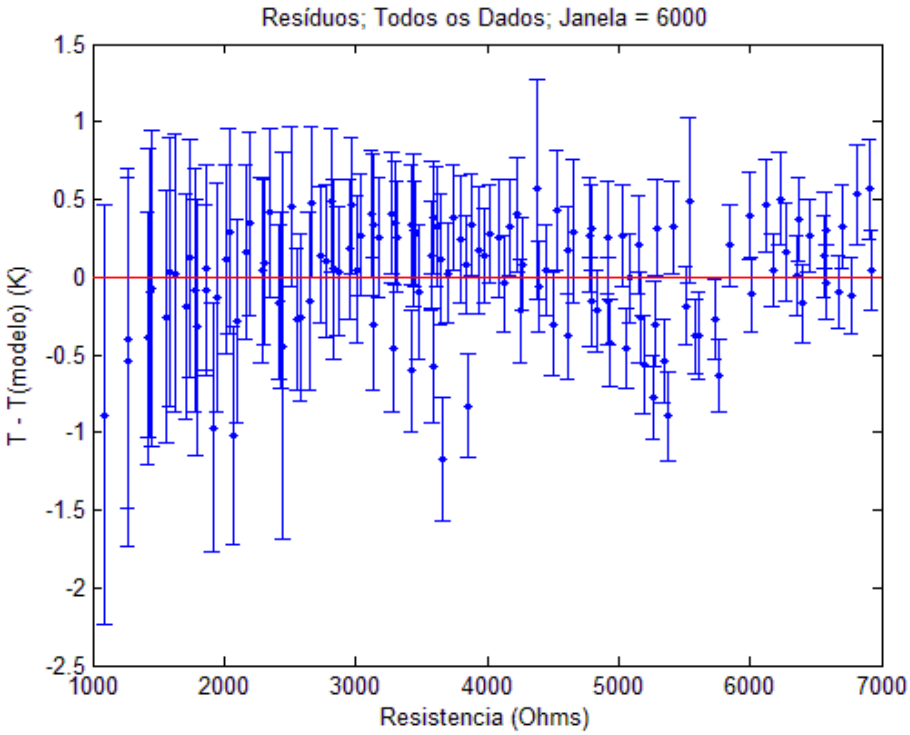
Fixando a janela como sendo a de 6000 dados refizemos as análises, mas agora juntamos todos os arquivos de dados em um só, para assim fazer um ajuste de todos os dados.



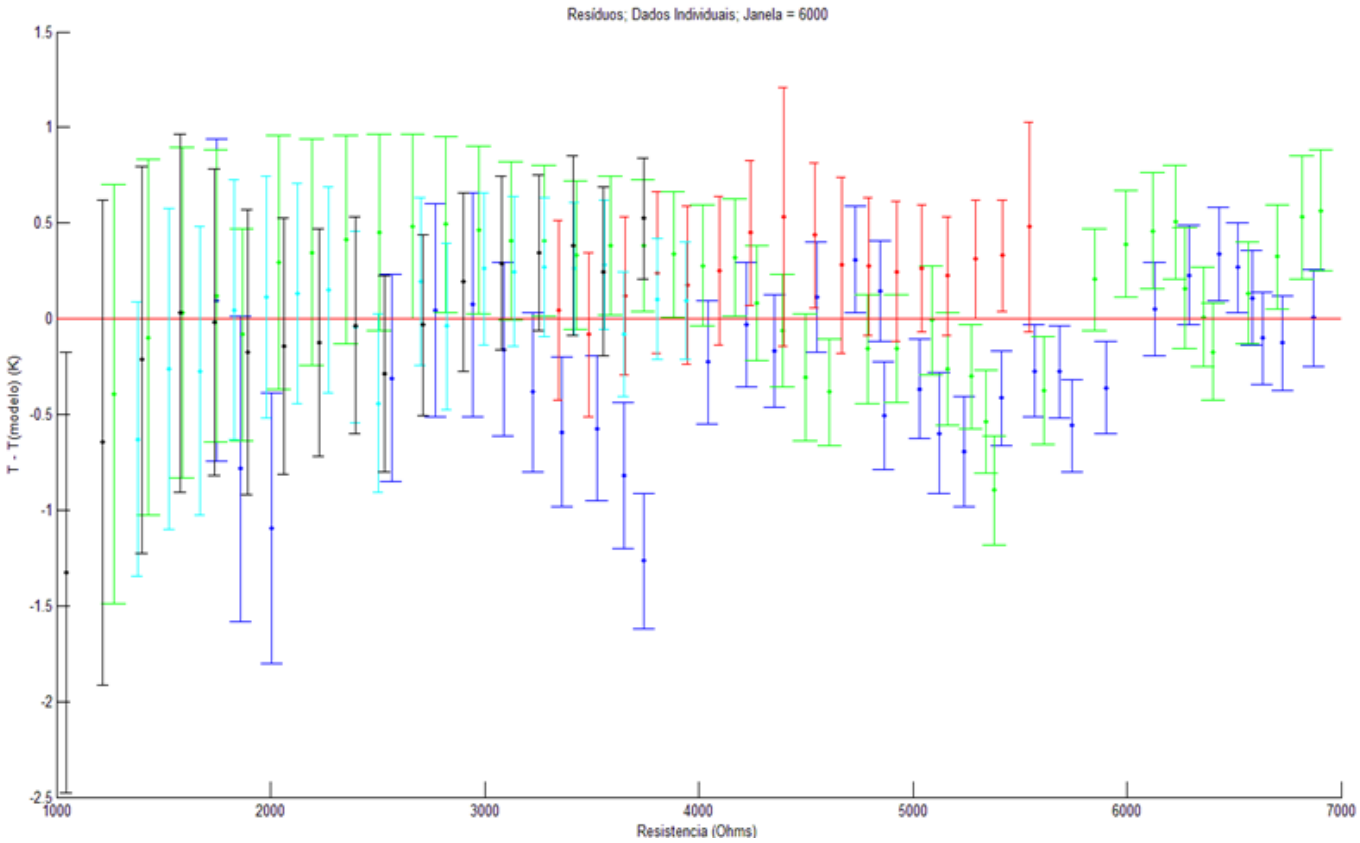


Como evidenciado pela imagem, encontramos um erro visível em nossa análise pois os pontos em destaque, tanto no ajuste quanto nos resíduos, não apareceram durante a análise individual dos dados e chegamos à conclusão que tais pontos aparecem devido a nós simplesmente juntarmos os dados em um arquivo sem uma formatação apropriada, logo uma janela de dados cobre o início e o fim de uma coleta de dados na mesma janela, levando a uma média irreal de dados. Com uma pequena modificação no *script* foi possível simplesmente eliminar tais pontos dos resultados finais.





Ao analisarmos os resíduos suspeitamos de uma tendência nos dados, porém não estava explícita o suficiente, então identificamos os diferentes conjuntos de dados por cores



Um padrão ondulatório agora é visível, porém não soubemos explicar tal padrão, então é apenas como uma observação.

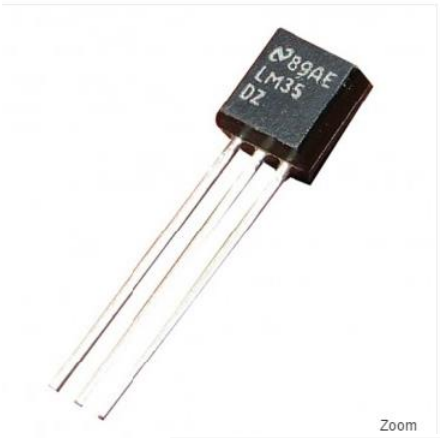
A curva de cada termistor é característica do mesmo, mas para termos algum tipo de referência para nossos resultados utilizamos os parâmetros dados em um programa que faz a leitura de NTC direto do software do Arduino[3]. Do ajuste obtemos os seguintes parâmetros

	MEDIDO	REFERÊNCIA
A	$9{,}50 \times 10^4 \pm 0{,}93 \times 10^4$	$11{,}29 \times 10^4$
B	$2{,}69 \times 10^4 \pm 0{,}17 \times 10^4$	$2{,}34 \times 10^4$
C	$-8{,}5 \times 10^8 \pm 8{,}8 \times 10^8$	$8{,}8 \times 10^8$

5. Conclusão

A suposição do parâmetro C ser nulo é válida pelo fato de sua incerteza ter a mesma ordem de grandeza de seu valor. Observamos também que ambos parâmetros A e B estão distantes cerca de 2σ, porém também é notável que A e B se distanciam dos valores de referência em sentidos diferentes e isso é explicado através de uma análise de correlação entre ambos pois o coeficiente de correlação é negativo e muito próximo a 1. Em suma, a confiança no modelo, ajuste de uma cura satisfatória aos dados e a análise de covariância indicam que nossos parâmetros são uma boa aproximação dos desejados.

Através de uma pesquisa em sites de compra concluímos que ao utilizar um termistor NTC calibrado se economiza 92,2% no valor dos termistores de um sistema.



Sensor de Temperatura LM35

★★★★★ 1 Avaliação

Sensor de temperatura LM35

R$8,50

Comprar Quantidade: 1

Cep: Calcular Frete

Sensor de Temperatura NTC 10K 5mm

☆☆☆☆☆ Escrever uma avaliação

Sensor de temperatura NTC 10K 5mm

R$0,60

Comprar Quantidade: 1

Cep: Calcular Frete

Bibliografia

[1] https://en.wikipedia.org/wiki/Thermistor

[2] http://thermistor.sourceforge.net/

[3] http://labdegaragem.com/profiles/blogs/tutorial-como-utilizar-o-termistor-ntc-com-arduino

https://www.arduino.cc/

Apêndice

Rotina de obtenção de dados:

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

void loop() {
   int sensorValue1 = analogRead(A0);
//  sensorValue1 se refere ao LM35
  Serial.print(sensorValue1);
  Serial.print("\t");
  delay(10);
  int sensorValue2 = analogRead(A3);
//  sensorValue2 se refere ao NTC
  Serial.println(sensorValue2);
  delay(10);
```

***Script* de análise de dados:**

```
clear

NOME_ARQUIVO = 'Todos-juntos.txt';

JANELA = 6000;

A = load(NOME_ARQUIVO);

% fix = arredonda para baixo

A = A(1:fix(size(A,1)/JANELA)*JANELA,:);

% 1.1 = voltagem do AREF;

T = 100*(1.1/1023)*reshape(A(:, 1),JANELA, []);

B = reshape(A(:, 2),JANELA, []);

R = 10*(1023./B -1);
```

```
Tm = mean( T );
Rm = mean( R );
sT = std( T );
sR = std( R );

Rm = Rm*1000;
Tm = Tm + 273.15;

%deleta pontos contaminados
k = find (sT>3);
 for i= 1 : length(k)
     sT(k(i)-(i-1)) = [];
     Tm(k(i)-(i-1)) = [];
    Rm(k(i)-(i-1)) = [];
end

%ajuste do modelo
model = @(b,Rm) (1./(b(1) + b(2)*log(Rm) + b(3)*(log(Rm)).^3));
[ beta, Res ] = mmqGM( Rm, Tm, sT, model, [1.8987861143e-3; 1.818009621e-4; -
5.760160042e-7]);

%plota dados+modelo c/ barra de erro
figure (1)
x = linspace (1000,7000,100);
y = (1./(beta(1) + beta(2)*log(x) + beta(3)*(log(x)).^3));
plot( x, y, '-r');

title('Todos os Dados; Janela = 6000')
xlabel('Resistencia (Ohms)')
ylabel('Temperatura (K)')
xlim('auto')
ylim('auto')
```

```
figure (1)
hold on
errorbar( Rm, Tm, sT,'.b');

%plot residuos
TmAjustado = (1./(beta(1) + beta(2)*log(Rm) + beta(3)*(log(Rm)).^3));
figure (2)
errorbar (Rm,(Tm-TmAjustado),sT, '.b');
ylim('auto')
xlim('auto')

%disp(Res)

chi2Red = (Res.qui2)/(Res.ngl)
```

Função “mmqGM” disponibilizada no site da disciplina