EEE026 - Projeto de Sistemas Embutidos
Projeto - Sensor de Temperatura

José Geraldo Fernandes
Escola de Engenharia
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
josegeraldof@ufmg.br

Arthur de Oliveira Lima
Escola de Engenharia
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
arthurdeolima@gmail.com

João Víctor de Melo
Escola de Engenharia
Universidade Federal de Minas Gerais
Belo Horizonte, Brasil
joaovictordemelo@ufmg.br

I. Configuração

A. Sensor

O sensor utilizado é um termistor NTC 103, *negative temperatura coefficient component*, um componente eletrônico, no qual, a temperatura varia inversamente com à temperatura (coeficiente negativo), cuja curva de temperatura é logarítmica e inversamente proporcional a temperatura. Isso significa que, a medida em que a temperatura aumenta, sua resistência diminui. O coeficiente que dita esse comportamento é chamado β, que pode ser estimado utilizando informações do *datasheet* do termistor, como na Equação 1).

$$\beta = \frac{\log_{10}(\frac{R_{0C}}{R_{100C}})}{\frac{1}{273.15} - \frac{1}{373.15}} \tag{1}$$

Usando o beta, é possível calcular uma resistência de referência, que será utilizada para estimar a temperatura do ambiente com base na medida de resistência do componente. Essa referência é definida como na Equação 2

$$R_{inf} = R_{0°\mathrm{C}} * e^{-\beta/273.15} \tag{2}$$

Por fim, para de fato calcular a temperatura, usa-se um divisor de tensão simples, medindo-se a tensão sobre o termistor e usando isso para calcular a resistência que ele apresenta. Para isso, basta colocar o circuito sob uma tensão conhecida (Vcc) e saber o valor do resistor de divisão (Rdiv). Para nosso experimento, vamos usar 5V e 10kΩ, dessa forma o sinal já está condicionado. Nessa configuração, a resistência e, consequentemente, a temperatura podem ser calculadas como nas Equações 3 e 4:

$$R_{NTC} = \frac{R_{div} * V_{med}}{V_{cc} - V_{med}} \tag{3}$$

$$T_{celsius} = \frac{\beta}{\log_{10}(\frac{R_{NTC}}{R_{inf}})} - 273.15 \tag{4}$$

B. Interface

Para visualização da temperatura pelo usuário, utilizamos um LED LCD 16x2 com interface I2C. Este dispositivo é capaz de se comunicar com o microcontrolador por meio de protocolo I2C e exibir informações enviadas. Para sua manipulação, lançamos mão de uma biblioteca chamada LiquidCrystal I2C, colocada nas referências. Essa biblioteca possibilita a criação de um objeto LCD, que já possui métodos prontos para impressão de informações. Ela gerencia toda a parte de I2C utilizando a biblioteca Wire.h, do próprio Arduino, e as portas analógicas 4 e 5 do Arduino Uno, que já são preparadas para comunicação I2C (SDA e SDK). A única particularidade foi a necessidade de se rodar um script anterior para determinar o endereço dos slaves do LCD no I2C (esse parâmetro é necessário para o construtor do objeto LCD.

II. Observações Experimentais

A priori, sabemos que fixa-se 100 Graus centígrados aos 5V da fonte de tensão, e temperatura ambiente de 25 graus em 2,5V, então usa-se a regra que associa temperatura referencial do condutor e a sua resistência, com valores gerais deles. Ora, durante o experimento, notou-se uma dificuldade em se chegar ao 0 grau no LCD por medição. Além disso, tivemos outro problema, o mesmo ocorreu para 100 graus Celsius. Para que não ocorresse um curto-circuito, é evidente que mesmo nessas condições, colocou-se as pontas de prova ou partes do circuito dentro do líquido variante em temperatura.

III. Rotina

O código 1 mostra o programa em C utilizado. Trata-se, basicamente, de duas partes, a configuração e operação do conversor AD e do *display*.

Sobre o conversor, deve-se configurar os registradores ADMUX e ADCSRA. Definiu-se a referência do nível lógico alto como a entrada VCC, configuração de 10bits e entrada A0 são padrão. Para operação, o BIT7 é acionado e o BIT6 ligado para conversão única. Esse mesmo bit, BIT6, é monitorado para esperar o resultado da conversão. Na função leituraAnalogica, faz-se a conversão de dez amostras e o resultado é a média dessas.

A comunicação com o *display* utiliza as funções da biblioteca LiquidCrystal, e as funções de temperatura são definidas como nas Equações 1, 2, 3 e 4.

Listing 1. Rotina Implementada

```
#include <LiquidCrystal_I2C.h>      // usando a biblioteca LiquidCrystal

#define BIT0 0x01
#define BIT1 0x02
#define BIT2 0x04
#define BIT3 0x08
#define BIT4 0x10
#define BIT5 0x20
#define BIT6 0x40
#define BIT7 0x80

//Constantes do NTC
#define T0 298.15 // define constante igual a 298.15 Kelvin
#define R_div 10000 // Resistor do divisor de tensao
#define R0 10000 // Valor da resistencia inicial do NTC
#define T1 273.15 // [K] in datasheet 0   C
#define T2 373.15 // [K] in datasheet 100   C
#define RT1 35563 // [ohms] resistencia in T1
#define RT2 549 // [ohms] resistencia in T2

// Defini  es do conversor AD
#define sensorPin A0  // define entrada anal gica A0
#define CHECK_BIT(reg,n) ((reg) & (1<<(n))) // verificar estado do bit n no registrador

// Confirgura  es do LCD
const int rs = 8, en = 9, d4 = 4, d5 = 5, d6 = 6, d7 = 7; // definicao dos pinos do Display
LiquidCrystal_I2C lcd(0x27, 16, 2);                // configurando os pinos

// Variaveis globais
float sensorValue = 0;
float tensao;
float temperatura;
float beta = 0.0; // parametros iniciais [K]
float R_inferior = 0.0; // parametros iniciais [ohm]
float R_NTC = 0.0; // Rout in A0

float leituraAnalogica(){
  // talvez poderia passar o pino como argumento e fazer config de MUX[3:0] adequada
  unsigned short x;

  ADCSRA |= BIT7; // ativa adc, ADEN = 1

  float media = 0;
  for (int i = 0; i < 10; i++){
    ADCSRA |= BIT6; // single conversion, ADSC = 1
    while(CHECK_BIT(ADCSRA, 6)){ // espera at  ADSC = 0, q significa q a conversao terminou
      asm("");
    }
    x = ADC;
    media += (float)x;
  }
  media /= 10.0;
  ADCSRA &= ~(BIT7); // desliga adc
  return media;
}
```

```
void mostrar_temp(){
  R_NTC = (R_div * tensao / (5 - tensao)); // calculo de Rout
  temperatura = (beta / log(R_NTC / R_inferior)) - 273.15; // calculo da temp. em Celsius

  lcd.setCursor(0, 0);            // selecionando coluna 0 e linha 0
  lcd.print("Temperatura:");      // print da mensagem
  lcd.setCursor(2, 1);            // selecionando coluna 2 e linha 1
  lcd.print(temperatura));
}

void setup() {
  // configuracao adc
  ADMUX |= BIT6; // REFS[1:0] = 01, high reference como avcc
  // config 10bits default (ADMUX), ADLAR = 0
  // entrada A0 default (ADMUX), MUX[3:0] = 0000

  beta = (log(RT1 / RT2)) / ((1 / T1) - (1 / T2)); // calculo de beta
  R_inferior = R0 * exp(-beta / T0); // calculo de Rinf

  lcd.init();                      // inicializa LCD
  lcd.backlight();                 // ativa led de backlight

}

void loop() {
  sensorValue = leituraAnalogica();
  tensao = sensorValue * (5.0 / 1024); // c lculo da tens o
  mostrar_temp();
}
```