**Reves apra programação do Datalloger de Temperatura, Questoes DE Programação, Instrínsecas do Processo!**

**ntendendo o Software**

**Lendo o valor de temperatura**

Primeiramente, em nosso programa usamos o comando de leitura analógica, já estudado no tutorial [Entradas e Saídas Analógicas](https://portal.vidadesilicio.com.br/entradas-e-saidas-digitais/), para fazer a leitura do valor em A0. Além disso, usamos a comunicação serial, também discutida em outro tutorial, [Comunicação Serial Arduino](https://portal.vidadesilicio.com.br/comunicacao-serial-arduino/) . É importante que o leitor entenda como eles funcionam. Experimente ler nossos tutoriais anteriores.

Em resumo, nosso programa lerá qual é o valor do sinal no pino A0, que varia de 0 a 1023, onde 0 corresponde a 0Volts e 1023 corresponde a 5Volts. Como sabemos, 1ºC é igual a 10mV. Sendo assim, temos:

*Tensão em A0 = (Valor lido em A0)\*(5/1023)*

*Temperatura = Tensão em A0/10mV*

Logo:

Temperatura =  [(Valor lido em A0)\*(5/1023)]/10mV

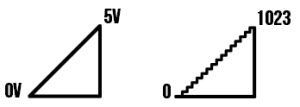
Em linguagem de programação, ficará:

temperatura = (**float**(analogRead(LM35))\*5/(1023))/0.01;

**Transformando o tipo da variável**

Perceba que colocamos o comando de leitura do valor analógico, analogRead, dentro de float(). Você saberia me dizer o motivo?

Quando o Arduino faz uma leitura analógica, ele converte o valor lido, que pode ser um valor de tensão entre 0V e 5V, em um número entre 0 e 1023. Ou seja, o Arduino divide 5Volts, que é o maior valor que ele é capaz de ler, em 1023 partes iguais e lhe informa quantas partes tem o valor que ele está medindo.



Pense que temos uma rampa que vai de 0 à 5V e dividimos essa rampa em 1024 degraus. Então, quando estamos no degrau 0, estamos no que equivale a 0V, quando subimos o primeiro degrau vamos para o que equivale a 5V/1023 (5 Volts dividido pelos 1023 degraus restantes), que é aproximadamente igual à 0,00487V.

O número que o Arduino nos informa é do tipo inteiro, contudo, o valor de temperatura é um numero racional, que pode assumir valores decimais. Por conta disso, no nosso programa, declaramos a temperatura como uma float.

Em programação, quando multiplicamos uma variável inteira por uma variável racional, o programa considera que o resultado deve ser inteiro, eliminando a parte decimal da variável. Dessa forma, para que tenhamos um resultado racional, devemos transformar o número inteiro em um número racional.

Em virtude disso, em nosso código foi necessário colocar o comando de leitura do valor analógico, analogRead, dentro de float().

numeroracional = **float**(numero);

temperatura = (**float**(analogRead(LM35))\*5/(1023))/0.01;

Sempre que for necessário fazer um calculo com o valor analógico, precisamos convertê-lo para uma variável do tipo float.

Em alguns casos, precisamos transformar uma variável qualquer para o tipo inteiro. O procedimento é o mesmo, ou seja, basta colocar o valor ou variável dentro dos parenteses de int();

numerointeiro= **int**(numero);

temperatura = **int**((**float**(analogRead(LM35))\*5/(1023))/0.01);

Se usamos o int() no calculo de temperatura, tal como mostrado acima, teremos um resultado sem os números decimais. Faça o teste.

//Sensor de temperatura usando o LM35

const int LM35 = A0; // Define o pino que lera a saída do LM35

float temperatura; // Variável que armazenará a temperatura medida

//Função que será executada uma vez quando ligar ou resetar o Arduino

void setup() {

Serial.begin(9600); // inicializa a comunicação serial

}

//Função que será executada continuamente

void loop() {

temperatura = (float(analogRead(LM35))\*5/(1023))/0.01;

Serial.print("Temperatura: ");

Serial.println(temperatura);

delay(2000);

}

**FIM PART 1**

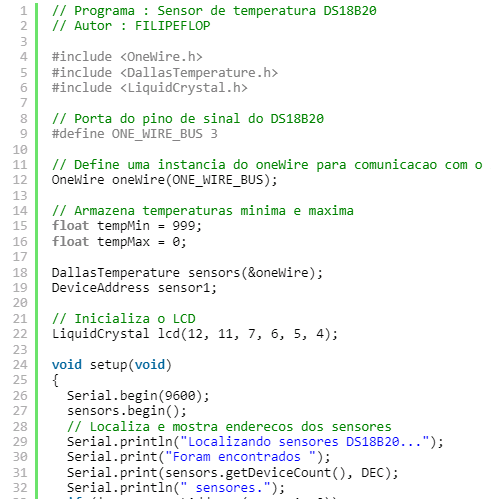
**FIM PART 1**

**Reves apra programação do Datalloger de Temperatura, Questoes DE Programação, Instrínsecas do Processo!**

**Parte 2**

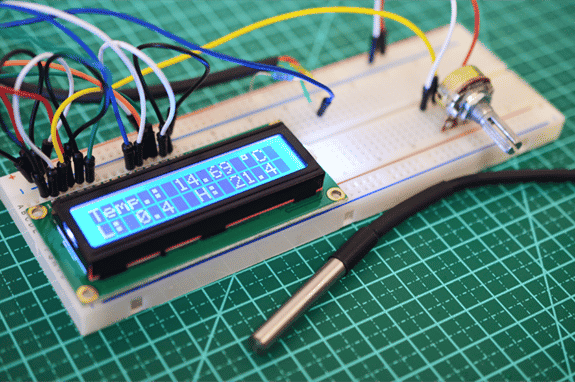
Para o programa precisamos de duas bibliotecas, a **Dallas Temperature** ([download](http://www.hacktronics.com/code/DallasTemperature.zip)) e também a **OneWire** ([download](https://www.pjrc.com/teensy/arduino_libraries/OneWire.zip)). Essas duas bibliotecas trabalham em conjunto para extrair os dados do sensor. Descompacte as bibliotecas e copie as pastas DallasTemperature e OneWire para dentro da pasta **LIBRARIES** da IDE do Arduino.

O programa mostra no display 16×2 as informações da temperatura atual e também a temperatura **mínima (L/Low)** e **máxima (H/High)**. A atualização é feita a cada 3 segundos.

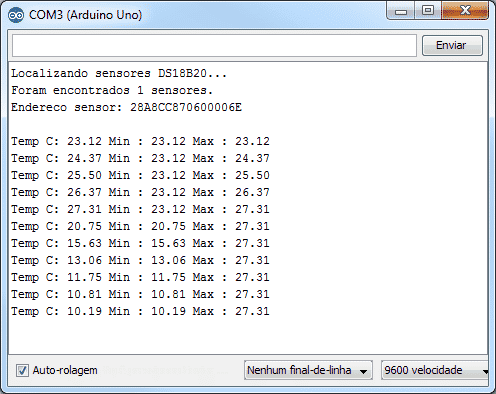






[](https://www.filipeflop.com/wp-content/uploads/2015/06/DSC_0762.png)

Se você quiser, pode usar este circuito sem o display, já que as informações também são enviadas para o serial monitor. Neste caso, além das informações de temperatura atual, mínima e máxima, temos também a informação de quantos sensores foram encontrados no barramento e qual o endereço do sensor:



**Fim da parte 2**

**Reves apra programação do Datalloger de Temperatura, Questoes DE Programação, Instrínsecas do Processo!**

**Parte 3**

/ this example is public domain. enjoy! https://learn.adafruit.com/thermocouple/

#include "max6675.h"

int thermoDO = 19;

int thermoCS = 23;

int thermoCLK = 5;

MAX6675 thermocouple(thermoCLK, thermoCS, thermoDO);

void setup() {

Serial.begin(9600);

Serial.println("MAX6675 test");

// wait for MAX chip to stabilize

delay(500);

}

void loop() {

// basic readout test, just print the current temp

Serial.print("C = ");

Serial.println(thermocouple.readCelsius());

Serial.print("F = ");

Serial.println(thermocouple.readFahrenheit());

// For the MAX6675 to update, you must delay AT LEAST 250ms between reads!

delay(1000);

}

**Reves apra programação do Datalloger de Temperatura, Questoes DE Programação, Instrínsecas do Processo!**

**Parte 4**

// Projeto Termômetro Digital com Arduino.

// Display I2C 20x4 + Kit Termopar Tipo K.

// Carrega as bibliotecas necessárias.

#include "Wire.h"

#include "LiquidCrystal\_I2C.h"

#include "max6675.h"

int pinSO = 8; // Atribui o pino 8 como SO

int pinCS = 9; // Atribui o pino 9 como CS

int pinSCK = 10; // Atribui o pino 10 como SCK

// Cria uma instância da biblioteca MAX6675.

MAX6675 termoclanek(pinSCK, pinCS, pinSO);

// Define o endereço utilizado pelo Adaptador I2C.

LiquidCrystal\_I2C lcd(0x27, 20, 4);

void setup() {

//Inicializa o LCD e o backlight.

lcd.init();

lcd.backlight();

}

void loop() {

// Lê a temperatura to termopar em ºC

float teplotaC = termoclanek.readCelsius();

// Imprime os valores correspondentes à temperatura.

lcd.setCursor(0, 0);

lcd.print("Temp. em Celsius:");

lcd.setCursor(14, 1);

lcd.print(teplotaC);

lcd.setCursor(0, 2);

lcd.print("Temp. em Fahrenheit:");

lcd.setCursor(14, 3);

lcd.print(termoclanek.readFahrenheit()); // Converte para ºF.

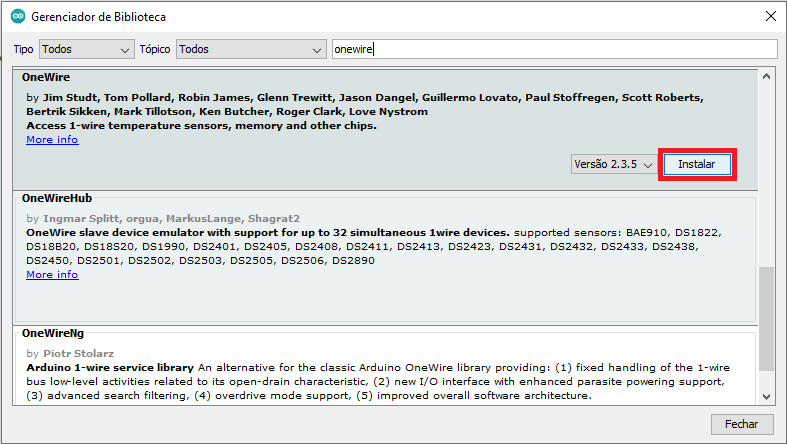
// Delay (tempo) para a próxima leitura.

delay(1000);

**Reves apra programação do Datalloger de Temperatura, Questoes DE Programação, Instrínsecas do Processo!**

**Parte 5**

Em seguida, a biblioteca OneWire deve ser instalada. Para isso, escreva “onewire” na barra de pesquisa do Gerenciador de Biblioteca e escolha a opção mostrada abaixo.



Por fim, pesquise “ds18b20” e instale a biblioteca DallasTemperature, como ilustrado.



Tudo pronto? Vamos para o código!

### Código

O código a seguir representa a implementação do projeto do sensor de temperatura:

#include <OneWire.h>

#include <DallasTemperature.h>

// O fio de dados é conectado no pino digital 2 no Arduino

#define ONE\_WIRE\_BUS 2

// Prepara uma instância oneWire para comunicar com qualquer outro dispositivo oneWire

OneWire oneWire(ONE\_WIRE\_BUS);

// Passa uma referência oneWire para a biblioteca DallasTemperature

DallasTemperature sensors(&oneWire);

void setup(void)

{

sensors.begin(); // Inicia a biblioteca

Serial.begin(9600);

}

void loop(void)

{

// Manda comando para ler temperaturas

sensors.requestTemperatures();

// Escreve a temperatura em Celsius

Serial.print("Temperatura: ");

Serial.print(sensors.getTempCByIndex(0));

Serial.println(" graus");

delay(500);

}

**Reves apra programação do Datalloger de Temperatura, Questoes DE Programação, Instrínsecas do Processo!**

**Parte 6**