UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO

INSTITUTO POLITÉCNICO

GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

Rennan Cockles Cardoso Fontes de Araujo

Circuito para elaboração gráfica de temperaturas e automação

Nova Friburgo

2017

UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO

INSTITUTO POLITÉCNICO

GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

Rennan Cockles Cardoso Fontes de Araujo

Circuito para elaboração gráfica de temperaturas e automação

Trabalho de Conclusão de Curso apresentado como pré-requisito para obtenção do título de Engenheiro de Computação, ao Departamento de Modelagem Computacional do Instituto Politécnico, da Universidade do Estado do Rio de Janeiro.

Orientador: Prof. Dr. José Humberto Zani

Nova Friburgo

2017

UNIVERSIDADE DO ESTADO DO RIO DE JANEIRO
INSTITUTO POLITÉCNICO DO RIO DE JANEIRO
CURSO DE ENGENHARIA DE COMPUTAÇÃO

Reitor: Ruy Garcia Marques
Vice-Reitor: Maria Georgina Muniz Washington
Diretor do Instituto Politécnico: Ricardo Carvalho de Barros
Coordenador de Curso: José Humberto Zani

Banca Avaliadora Composta por: Prof. Dr. José Humberto Zani (Orientador)
Prof. Dr. Joaquim Teixeira de Assis
Prof. Dr. Roberto Pinheiro Domingos

CATALOGAÇÃO NA FONTE
UERJ/REDE SIRIUS/BIBLIOTECA CTC/E

A663 Araujo, Rennan Cockles Cardoso Fontes de.
Circuito para elaboração gráfica de temperaturas e automação / Rennan Cockles Cardoso Fontes de Araujo. – 2017.
58f. : il.

Orientador: José Humberto Zani.
Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) - Universidade do Estado do Rio de Janeiro, Instituto Politécnico.

1. Temperatura – Medição - TCC. 2. Arduino - TCC. 3. Scilab (Linguagem de programação de computador) - TCC. I. Zani, José Humberto. II. Universidade do Estado do Rio de Janeiro. Instituto Politécnico. III. Título.

CDU 536.5:004.4

Endereço: UERJ - IPRJ
Caixa Postal 97282
CEP 28614-090 - Nova Friburgo – RJ – Brasil.

_____
Rennan Cockles Cardoso Fontes de Araujo

Rennan Cockles Cardoso Fontes de Araujo

Circuito para elaboração gráfica de temperaturas e automação

Trabalho de Conclusão de Curso apresentado como pré-requisito para obtenção do título de Engenheiro de Computação, ao Departamento de Modelagem Computacional, do Instituto Politécnico, da Universidade do Estado do Rio de Janeiro.

Aprovado em 07 de Junho de 2017.

Banca Examinadora:

_____

Prof. Dr. José Humberto Zani (Orientador)

Instituto Politécnico - UERJ

_____

Prof. Dr. Joaquim Teixeira de Assis

Instituto Politécnico - UERJ

_____

Prof. Dr. Roberto Pinheiro Domingos

Instituto Politécnico - UERJ

Nova Friburgo

2017

RESUMO

ARAUJO, Rennan Cockles Cardoso Fontes de. *Circuito para elaboração gráfica de temperaturas e automação,* 2017. 58 f. Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) - Instituto Politécnico, Universidade do Estado do Rio de Janeiro, Nova Friburgo, 2017.

O projeto foi desenvolvido com o intuito de facilitar o trabalho de medição e monitoramento de temperaturas em experimentos realizados no laboratório de física do Instituto Politécnico da Universidade do Estado do Rio de Janeiro. Foram utilizados dois termistores conectados à uma plataforma de prototipagem eletrônica chamada Arduino[1] para fazer medições de temperaturas. Para a exibição dos dados obtidos, foi desenvolvida uma aplicação utilizando o software SCILAB[2]. Este projeto traz grandes benefícios para aplicações onde se é necessário observar e/ou comparar variações de temperaturas com o tempo, pois conta com gráficos detalhados do comportamento da mesma. O software foi desenvolvido para a plataforma Windows e se comunica com o Arduino através de uma porta USB. Ele monitora a temperatura exibindo os dados dos sensores instantaneamente em um gráfico de temperatura versus tempo e em um histograma de barra única. O software foi implementado para receber dados de um ou dois sensores do Arduino e salvá-los em um arquivo para análise posterior dos resultados.

Palavras-chave: Arduino. Scilab. Termistor. Automação. Elaboração gráfica de temperaturas.

ABSTRACT

ARAUJO, Rennan Cockles Cardoso Fontes de. *Circuit for graphic elaboration of temperatures and automation,* 2017. 58 f. Trabalho de Conclusão de Curso (Graduação em Engenharia de Computação) - Instituto Politécnico, Universidade do Estado do Rio de Janeiro, Nova Friburgo, 2017.

The project was developed with the purpose of facilitating the measurement and monitoring of temperatures in experiments carried out in the physics laboratory of the Polytechnic Institute of the State University of Rio de Janeiro. In this work was used two thermistors connected to an electronic prototyping platform called Arduino[1] to make temperature measurements. For displaying the data obtained, it was developed an application using the SCILAB software. This project brings great benefits for applications where it is necessary to observe and / or compare variations in temperature with time, because of the detailed graphs of its behavior. The software was developed for the Windows platform and communicates with the Arduino through a USB port. It monitors the temperature displaying the sensor data instantly on a temperature versus time graphic and in a single bar histogram. The software was implemented to receive data from one or two Arduino sensors and save them in a file for later analysis of the results.

Keywords: Arduino. Scilab. Thermistor. Automation. Graphic elaboration of temperatures.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

INTRODUÇÃO

O projeto foi desenvolvido com o intuito de facilitar o trabalho de medição e monitoramento de temperaturas em experimentos realizados no laboratório de física do Instituto Politécnico da Universidade do Estado do Rio de Janeiro. Utilizando sensores com uma precisão de ± 0,5°C entre -10°C e +85°C e funcionando na faixa entre -55°C e +125°C, o projeto é indicado para pequenas medições e comparações de temperaturas e variações da mesma, não sendo indicado para indústrias e projetos de grande porte.

Foi percebida a dificuldade em monitorar e registrar a variação da temperatura em experimentos como avaliação de calor específico e cálculo do equivalente em água do calorímetro. Com a utilização de sensores digitais, torna-se muito mais fácil o registro dos valores de temperatura por tempo em comparação com termômetros de mercúrio e álcool. Outro fator positivo na escolha de um sensor digital é a possibilidade de manipulação dos dados obtidos na medição.

A parte física do projeto conta com dois sensores de temperatura à prova d'água conectados em um Arduino Uno. O código do Arduino foi totalmente desenvolvido para este projeto e necessita de duas bibliotecas para seu funcionamento, a DallasTemperature[7] e a OneWire[8].

O software foi desenvolvido utilizando a linguagem Scilab por conta da sua facilidade na manipulação de dados em aplicações científicas. Ele também necessita de duas bibliotecas para a sua utilização, a Serial Communication[9] e a Scilab JSON[10], e conta com um gráfico com os valores da temperatura x tempo, um histograma de barra única para controle da temperatura e os últimos valores obtidos de tempo e temperatura exibidos em um painel.

1 **HARDWARE**

1.1 **Arduino**

O Arduino é uma interface eletrônica com um microcontrolador programável de 8bits com portas de entrada e saída, digitais e analógicas. Para este projeto foi utilizado o modelo de placa Arduino Uno que é um modelo mais barato, porém oferece recurso suficiente para o desenvolvimento deste projeto.

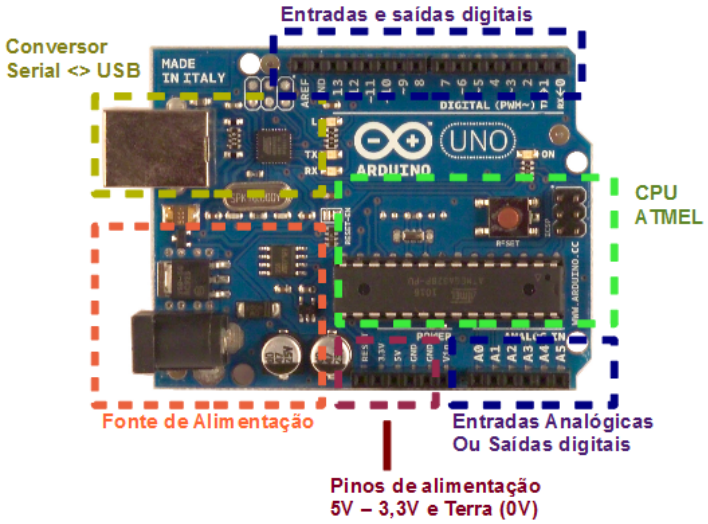
A Figura 1 mostra o esquema de entradas e saídas e alguns componentes da placa. Suas características podem ser observadas na Tabela 1.

Tabela 1 — Características da placa Arduino Uno

Microcontrolador	ATmega328
Tensão de funcionamento	5V
Tensão de entrada	6V - 20V
E/S Digitais	14
Entradas analógicas	6
Flash Memory	32KB
Clock	16MHz

Fonte: EMBARCADOS [11], 2013.

Figura 1 - Esquema de entradas e saídas do Arduino Uno



Fonte: ROBOTIZANDO [12], 2011.

1.2 **Sensor**

O sensor utilizado foi um sensor de temperatura DS18B20[3] à prova d'água. Ele foi escolhido por ser de simples utilização, impermeável, o que possibilita experimentos onde o sensor precisa ficar imerso, e por conter um cabo de 1 metro, o que facilita bastante seu manuseio. Outros sensores, como o LM35[4], possuem uma precisão melhor do que o sensor utilizado (±0,3°C), porém o conjunto de características do DS18B20 fez com que ele fosse escolhido.

A Tabela 2 mostra as especificações do sensor e a Figura 2 uma imagem do mesmo.

Tabela 2 — Especificações do sensor DS18B20

Chip	DS18B20
Tensão de operação	3,0V - 5,5V
Faixa de medição	-55°C a +125°C
Precisão	±0,5°C entre -10°C e +85°C
Dimensão da ponta de aço	6 x 50mm
Dimensão do cabo	100cm

Fonte: FILIPEFLOP [13], 2015.

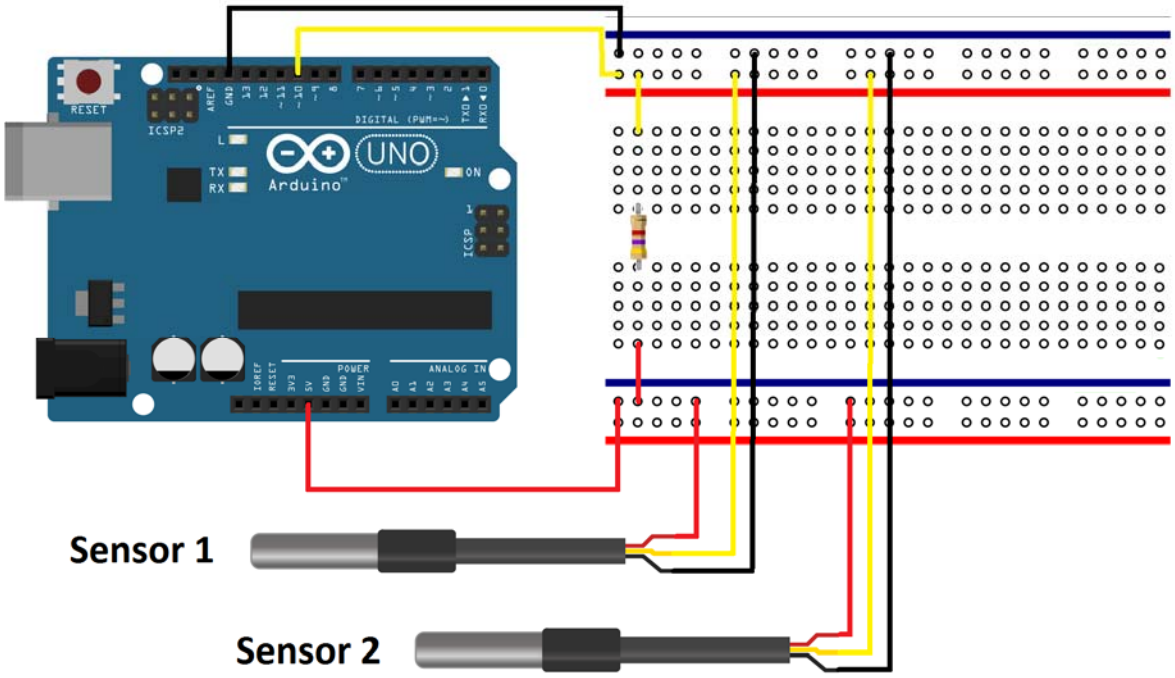
Figura 2 — Sensor DS18B20

Fonte: ARDUINOBR [14], 2014.

1.3 **Montagem**

O sensor DS18B20 contém três fios: um vermelho para a tensão (VCC), um preto que é o terra (GND) e um amarelo para os dados. Para fazer a ligação basta conectar o fio vermelho no pino de 5V do Arduino, o fio preto no GND, o fio amarelo no pino digital 10 e um resistor de pull-up de 4,7kΩ entre o fio de dados e o VCC. Para ligar o segundo sensor, basta conectá-lo em paralelo com o primeiro. Um esquema da montegem, utilizando uma matriz de contato (protoboard), pode ser visto na Figura 3.

Figura 3 – Montagem do hardware



Fonte: O autor, 2015.

A utilização da protoboard não é necessária. A conexão pode ser feita conectando os fios do sensor diretamente nas entradas do Arduino ou soldando em uma placa externa (PCI / PCB).

1.4 **Orçamento**

O preço dos equipamentos é bem variado, mas pode-se encontrar em lojas online a placa Arduino Uno por aproximadamente R$60,00 e os sensores de temperatura por aproximadamente R$25,00 cada. O resistor pode ser facilmente encontrado em qualquer loja de materiais eletrônicos por menos de R$1,00 cada. Caso queira utilizar a protoboard para fazer as conexões, pode-se adquirir uma com 400 pontos por aproximadamente R$20,00.

Com menos de R$115,00 (R$135,00 com a protoboard) pode-se adquirir todos os componentes do projeto, um custo relativamente baixo para suas aplicações.

2 **SOFTWARE**

2.1 **Scilab**

O Scilab é um software científico para computação numérica que fornece um poderoso ambiente computacional aberto para aplicações científicas. É uma linguagem de programação de alto nível, orientada à análise numérica. A linguagem provê um ambiente para interpretação, com diversas ferramentas numéricas. O Scilab inclui centenas de funções matemáticas com a possibilidade de adicionar interativamente programas de várias linguagens (FORTRAN, C, C++, Java).

2.2 **Arduino IDE**

O Arduino IDE[5] é uma aplicação multiplataforma escrita em Java derivada dos projetos Processing[2] e Wiring[3]. Inclui um editor de código com recursos de realce de sintaxe, parênteses correspondentes e identação automática, sendo capaz de compilar e carregar programas para a placa com um único clique.

Tendo uma biblioteca "Wiring", ele possui a capacidade de programar em C/C++. Isto permite criar com facilidade muitas operações de entrada e saída, tendo que definir apenas duas funções para fazer um programa funcional:

- Setup() – Inserida no início, no qual pode ser usada para inicializar a configuração;
- Loop() – Chamada para repetir um bloco de comando ou esperar até que seja desligada.

2.3 **O software**

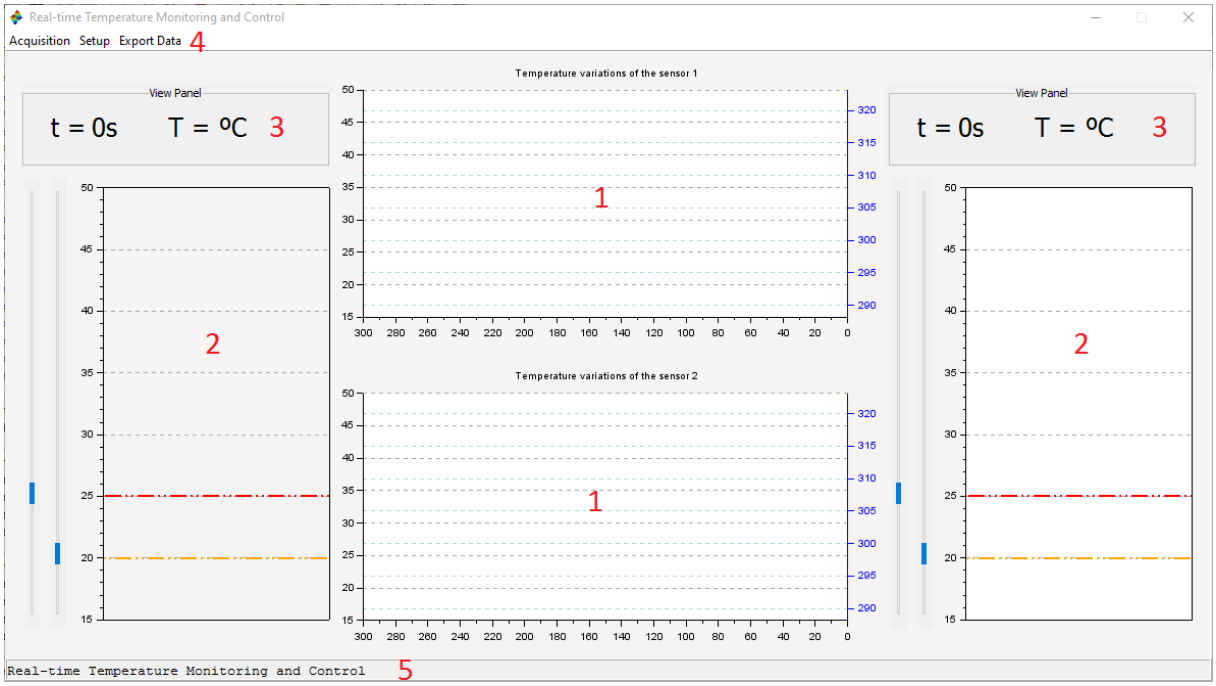
Implementado na linguagem Scilab, o software foi desenvolvido para auxiliar na visualização e controle dos dados obtidos dos sensores.

O código do Arduino foi desenvolvido para obter os dados do sensor a cada 1 segundo, pois o sensor DS18B20 possui um tempo de leitura de no máximo 750 ms.

[2] Para saber mais acesse: http://processing.org.
[3] Para saber mais acesse: http://wiring.org.co.

O software não possui nenhum tempo de espera entre as leituras, sendo o maior tempo de processamento definido pelo próprio tempo de leitura do sensor. Todos os dados armazenados no *buffer* do arduino são inseridos no gráfico instantaneamente.

Figura 4 – Tela inicial do software



Fonte: O autor, 2015.

A tela inicial do software, Figura 4, é constituída de:

1. Duas janelas de plot, uma para cada sensor, onde exibe o valor da temperatura em grau Celsius e Kelvin para os últimos 5 minutos (este valor pode ser alterado no menu setup);
2. Dois histogramas de barra única onde exibe o valor em grau Celsius da última temperatura lida. Cada histograma possui duas barras do tipo slide, uma vermelha e outra amarela. O slide vermelho controla o valor máximo da temperatura e o amarelo o valor mínimo;
3. Dois frames exibindo os valores do tempo e da temperatura atuais;
4. Um menu Acquisition onde se pode iniciar, parar e resetar a leitura dos sensores; um menu Setup onde se pode configurar os parâmetros do software (tamanho do buffer e valores de máximo e mínimo para as janelas de plot); e

um menu Export Data onde se pode exportar os dados obtidos para um arquivo com a extensão ‘csv’;

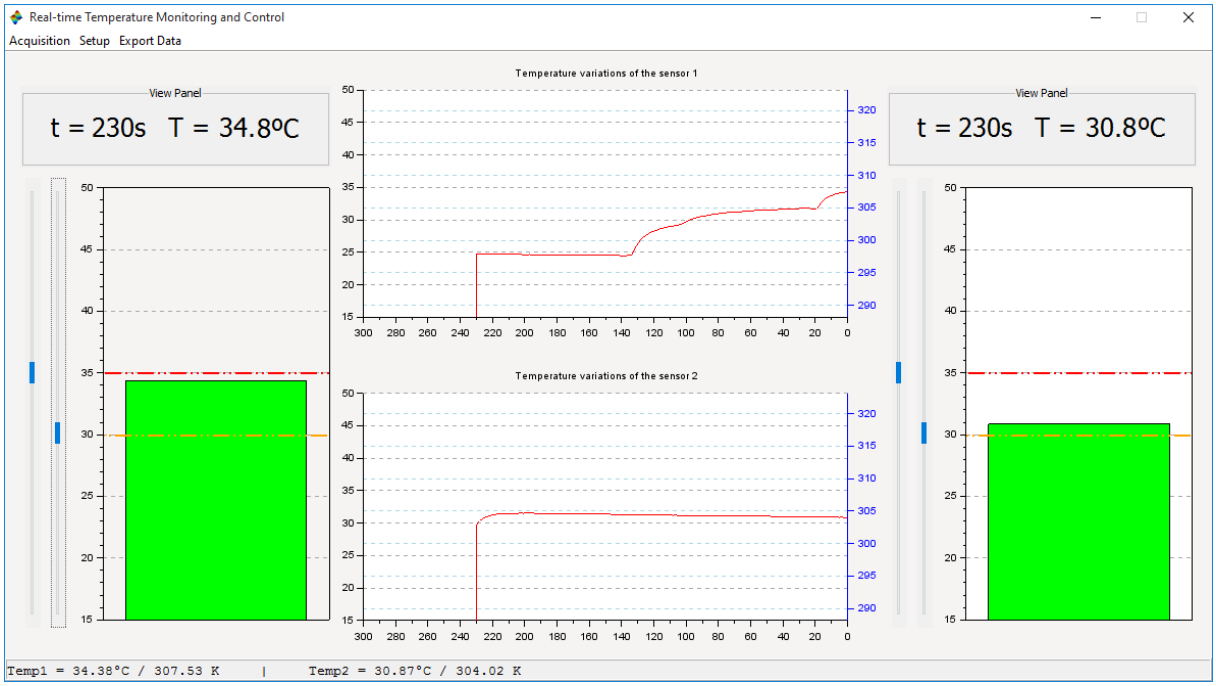
5. Uma barra inferior onde é exibido o valor da temperatura atual obtida pelos sensores tanto em grau Celsius quanto em Kelvin.

2.3.1 Funcionamento

Os dados obtidos pelos sensores no arduíno são exibidos em duas janelas de plot que vai sendo preenchida conforme os dados são recebidos. Ao mesmo tempo, o último valor obtido dos sensores é exibido nos histogramas que são coloridos de acordo com a posição dos *sliders* de controle.

Se o valor obtido estiver entre a temperatura máxima e a mínima definida pelo *slider*, o histograma ficará com a cor verde significando que a temperatura está dentro do padrão estabelecido, Figura 5.

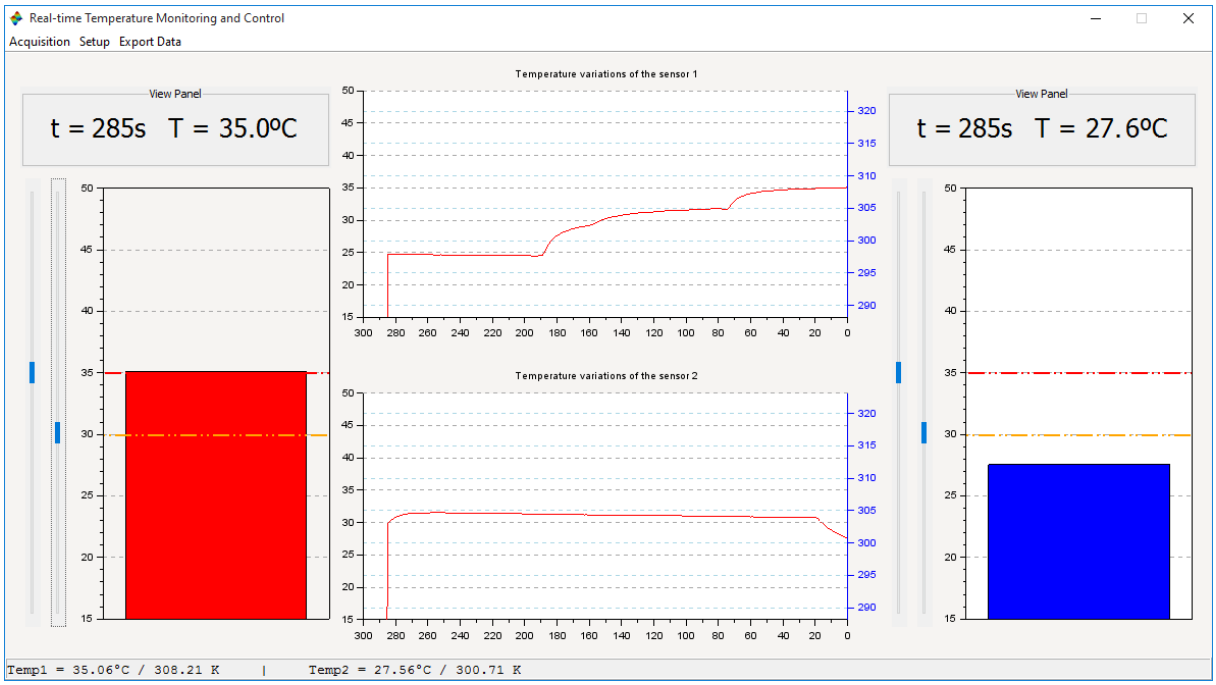
Figura 5 – Funcionamento com temperatura dentro do padrão



Fonte: O autor, 2015.

Se o valor obtido estiver acima da temperatura máxima definida, o histograma ficará com a cor vermelha significando que a temperatura está mais quente que o padrão estabelecido. Se o valor estiver abaixo da temperatura mínima definida, o histograma ficará com a cor azul significando que a temperatura está mais fria que o padrão estabelecido, veja a Figura 6.

Figura 6 – Funcionamento com temperaturas acima e abaixo do padrão

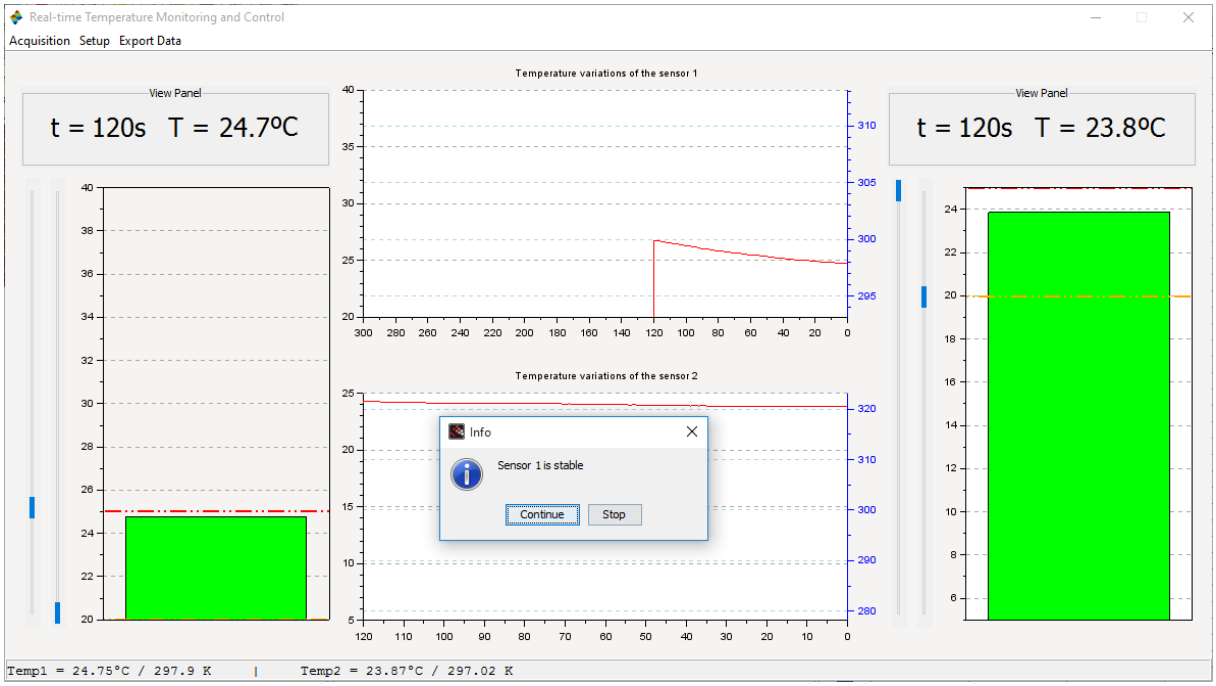


Fonte: O autor, 2015.

O software também sinaliza quando a temperatura fica estável. Por padrão, se a temperatura variar menos de 0,3 °C em 30 segundos, valores que podem ser alterados no menu Setup, um alert será exibido informando que a temperatura está estável dando as opções de parar ou continuar efetuando a leitura como mostra a Figura 7.

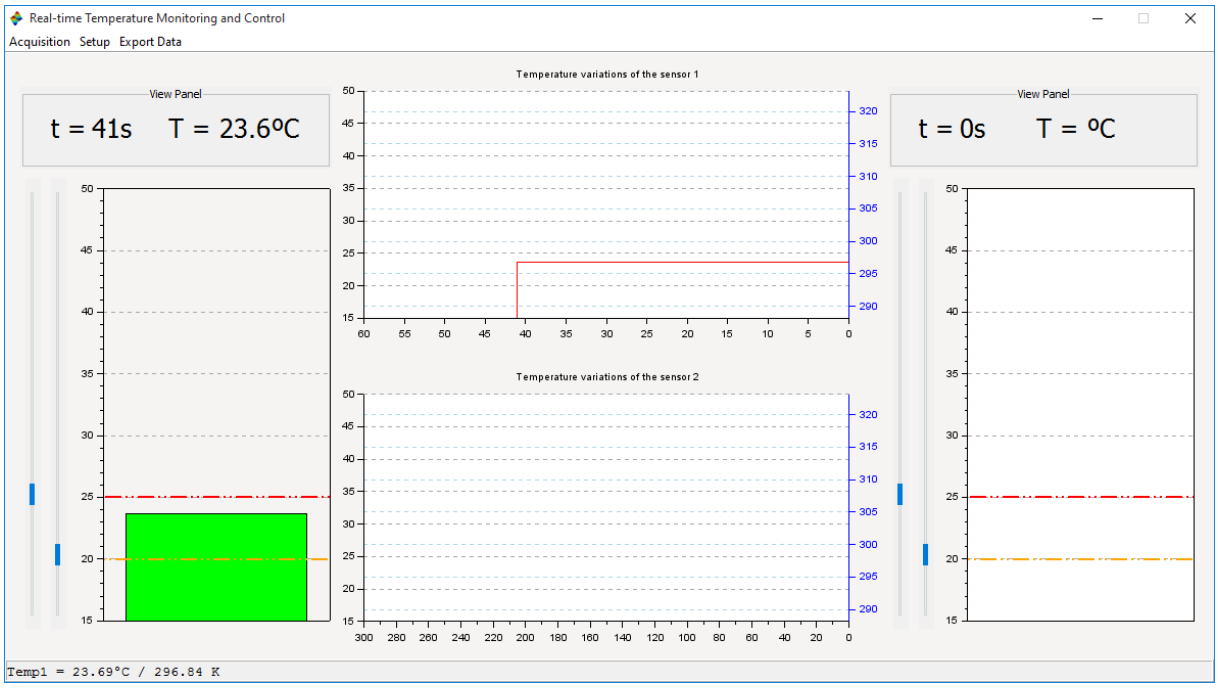
O projeto pode ser utilizado tanto com dois sensores conectados quanto com um. O funcionamento utilizando apenas um sensor é da mesma forma, tendo as mesmas funcionalidades. Basta desconectar um dos sensores e quando iniciar a leitura o sensor que estiver conectado será o sensor 1. A Figura 8 mostra o funcionamento do software com apenas um sensor.

Figura 7 – Sensor com temperatura estável



Fonte: O autor, 2015.

Figura 8 – Funcionamento com apenas um sensor conectado



Fonte: O autor, 2015.

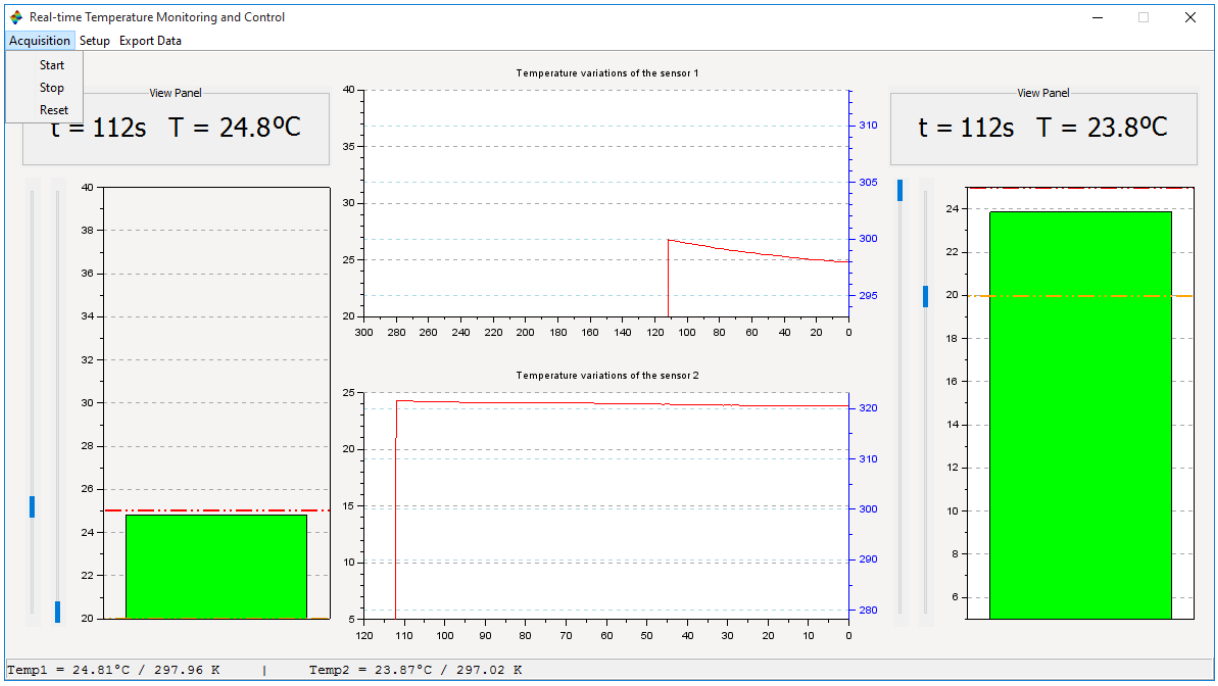
2.3.2 Menu e configuração

O barra superior possui 3 (três) menus, cada um com uma função distinta. O primeiro menu é o Acquisition, Figura 9, que é responsável por iniciar, pausar e resetar a leitura dos sensores.

O segundo menu é o Setup, responsável por configurar os parâmetros do software. Ele possui 3 submenus: Sensor 1, Sensor 2 e Stability.

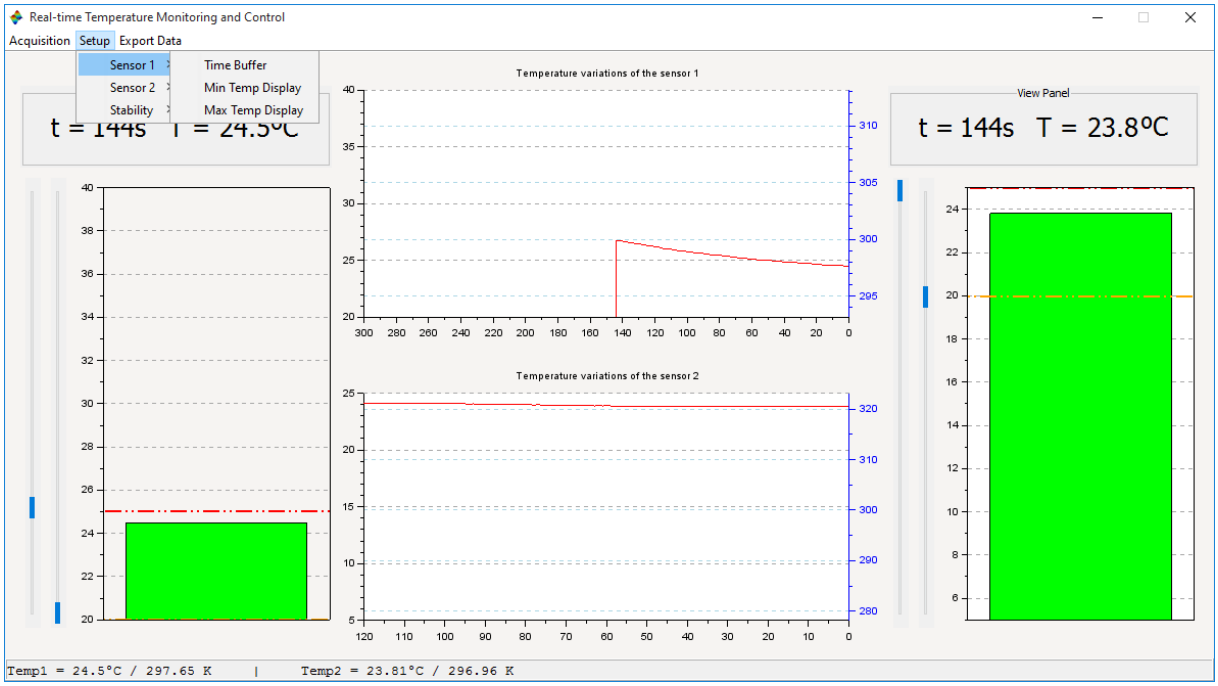
Os submenus Sensor 1 e Sensor 2, como mostra a Figura 10, são responsáveis por configurar os parâmetros dos sensores tais como temperaturas mínima e máxima das janelas de plot e histograma e o tempo, em segundos, que será exibido na janela de plot (time buffer).

Figura 9 – Menu Acquisition



Fonte: O autor, 2015.

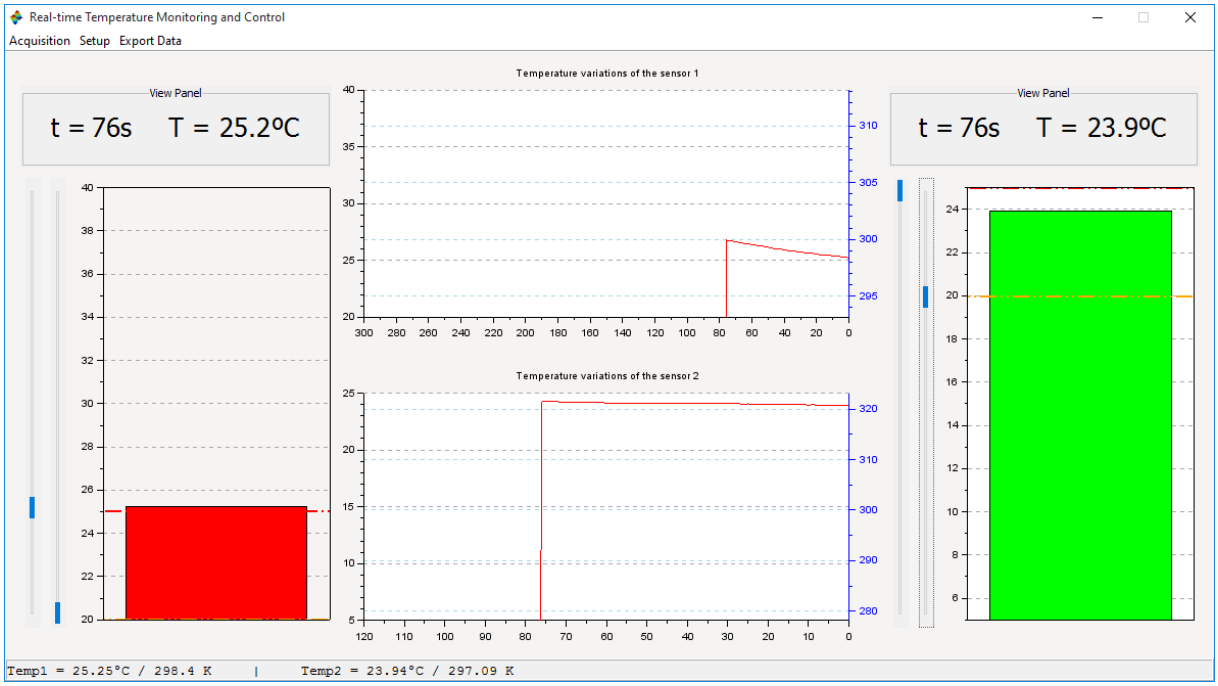
Figura 10 – Submenu Sensor 1 do menu Setup



Fonte: O autor, 2015.

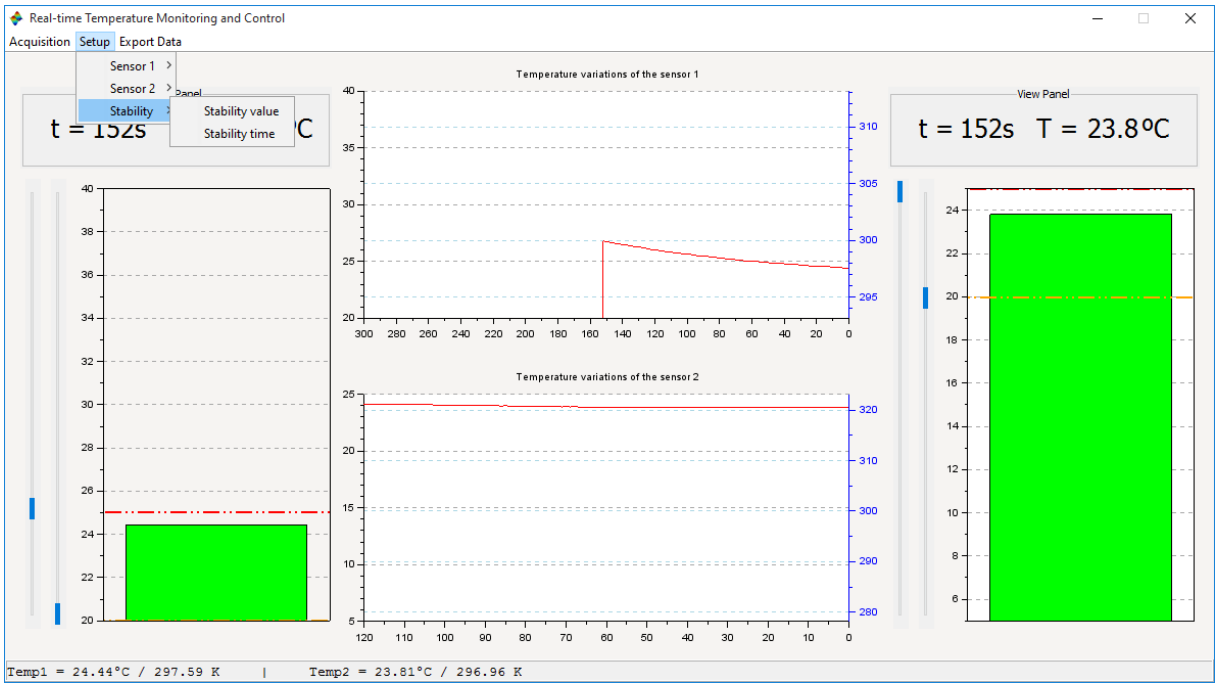
Ainda na Figura 10 pode-se observar os valores dos parâmetros alterados. Como por exemplo o valor da temperatura mínima da janela de plot e do histograma do sensor 1 é de 20°C, enquanto que para o sensor 2 é de 5°C. Pode-se observar também que o valor da temperatura máxima da janela de plot e do histograma do sensor 1 é de 40°C, enquanto que para o sensor 2 é de 25°C. Já o time buffer para a janela de plot do sensor 1 é 300s enquanto que para o sensor 2 é 120s. Outro exemplo de parâmetros alterados pode ser visto na Figura 11.

Figura 11 – Software com parâmetros alterados



Fonte: O autor, 2015.

Figura 12 – Submenu Stability



Fonte: O autor, 2015.

O submenu Stability, mostrado na Figura 12, é responsável por alterar os parâmetros que definem a estabilidade do sensor. Nele é possível configurar qual a variação máxima que a temperatura deve ter e em quanto tempo ela deve ser avaliada para que o sensor seja considerado estável.

O terceiro e último menu, Export Data, é responsável por exportar os dados obtidos pelo sensor desde o início da leitura até o momento atual. Os dados são exportados em um arquivo com a extensão ‘csv’ (comma-separated values).

3 **ANÁLISE DOS DADOS**

Os dados são exportados para o arquivo de três formas: ao fechar o software, ao resetar a leitura dos sensores e ao clicar no menu Export Data. Eles são salvos em arquivos com a extensão CSV, que pode ser aberto em softwares do tipo office para visualização de planilhas.

Abrindo o arquivo em um editor de planilhas, ele contará com três colunas sendo a primeira os valores do tempo de cada leitura do sensor, a segunda os valores obtidos pelo sensor 1 e a terceira os valores do sensor 2. Caso esteja utilizando apenas um sensor, a terceira coluna será preenchida com -1 (menos um).

O arquivo gerado também pode ser aberto pelo SCILAB a fim de fazer novos cálculos e processamentos com os dados exportados.

A Tabela 3 mostra exatamente como os dados são exibidos quando abertos em um editor de planilhas, como por exemplo, o Microsoft Office Excel[4].

Tabela 3 — Visualização dos dados

Tempo (s)	**(Sensor 1 ± 0,5) ºC**	**(Sensor 2 ± 0,5) ºC**
0	21,1	30,5
1	21,3	30,2
2	21,5	30,1
3	21,7	29,9
4	21,9	29,8
5	22,0	29,6
6	22,4	29,4
7	22,9	29,3
8	23,4	29,2
9	23,9	29,0
10	24,4	28,9
11	24,8	28,8
12	25,1	28,8
13	25,5	28,6
14	25.8	28,5

Fonte: O autor, 2015.

4 **APLICAÇÂO PRÁTICA**

Foi realizado um experimento utilizando ambos os sensores em uma geladeira como a exibida na Figura 13.

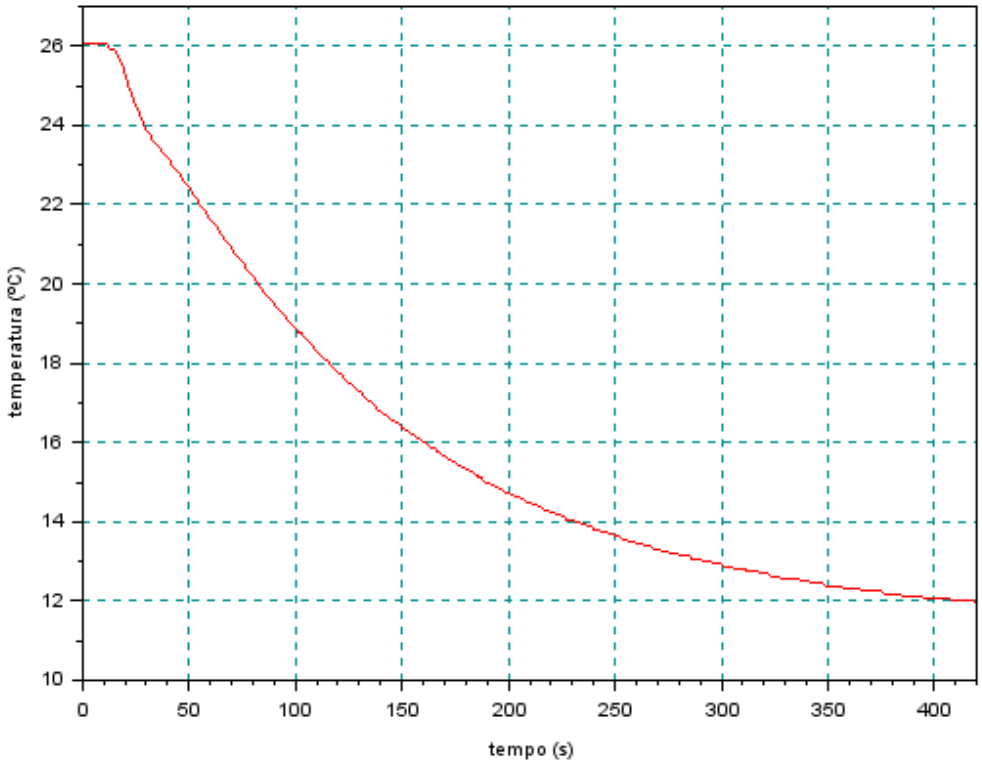
Figura 13 – Geladeira utilizada no experimento

Fonte: ZOOM [15], 2016.

Foi colocado um sensor na primeira prateleira da geladeira e o outro dentro do compartimento de freezer. Em seguida a medição foi iniciada e as portas do freezer e da geladeira foram fechadas.

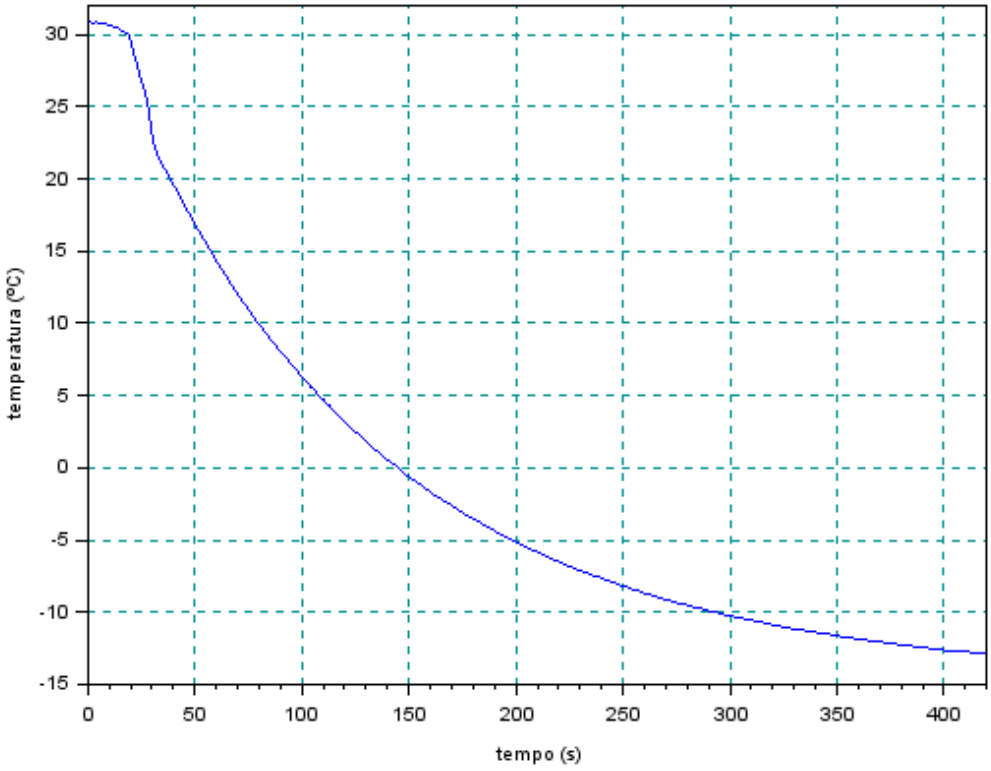
Foram necessários 7 minutos (420 segundos) para que as temperaturas tanto do freezer quanto da geladeira se estabilizassem em, respectivamente, -13°C e 12ºC. As Figuras 14 e 15 mostram o gráfico das temperaturas decaindo enquanto se estabilizavam.

Figura 14 – Estabilização da temperatura da geladeira



Fonte: O autor, 2016.

Figura 15 – Estabilização da temperatura do freezer



Fonte: O autor, 2016.

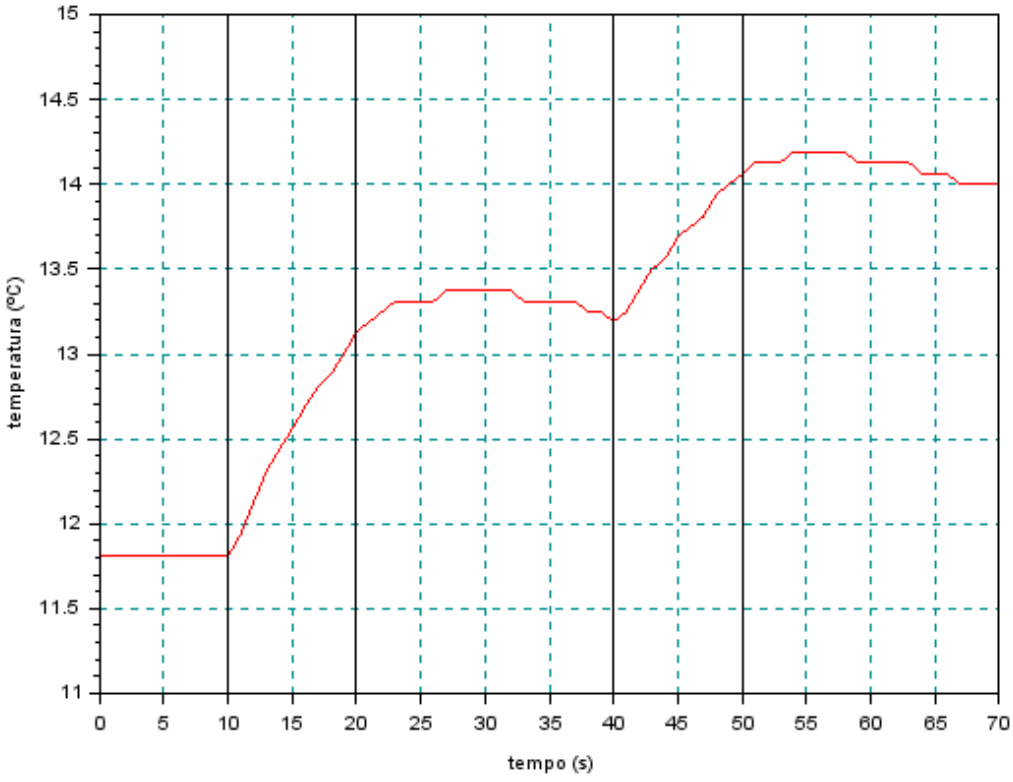
O experimento foi realizado para verificar e efeito que a abertura da porta da geladeira promove na temperatura da mesma e do compartimento de freezer. O experimento se iniciou quando ambos os sensores se estabilizaram.

O experimento levou 70 segundos no total e se deu da seguinte maneira:

1. Durante os primeiros 10 segundos as portas foram mantidas fechadas;
2. Nos 10 segundos seguintes a porta da geladeira foi mantida aberta em 90º com relação à sua posição fechada enquanto a do freezer permaneceu fechada;
3. Em seguida a porta foi mantida fechada por 20 segundos;
4. A porta da geladeira foi aberta novamente em 45º com relação à sua posição fechada enquanto o freezer permaneceu fechado por mais 10 segundos;
5. Para finalizar, as portas foram mantidas fechadas por mais 20 segundos.

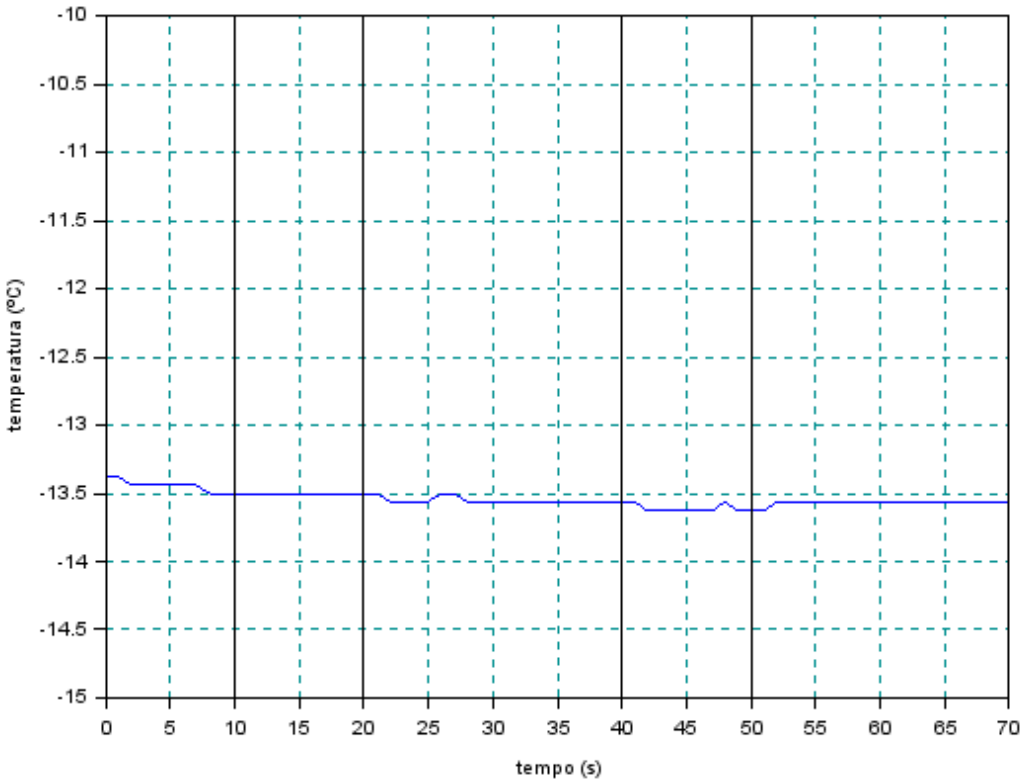
As Figuras 16 e 17 mostram o comportamento das temperaturas da geladeira e do freezer durante todo o experimento. As barras verticais mostram as transições de estados (porta aberta, porta fechada).

Figura 16 – Temperatura da geladeira



Fonte: O autor, 2016.

Figura 17 –Temperatura do freezer



Fonte: O autor, 2016.

Pode-se notar que a temperatura do freezer, Figura 17, permaneceu quase constante durante todo o experimento. Variou menos de 0,5ºC durante o experimento, não sofrendo muitas alterações com a abertura e o fechamento da porta de geladeira.

Já a temperatura da geladeira, Figura 16, sofreu grandes alterações.

1. Iniciou constante em 11,8ºC;
2. Subiu 1,3ºC em 10 segundos com a porta aberta em um ângulo de 90º;
3. Manteve-se equilibrada em 13,2ºC durante os 20 segundos com a porta fechada;
4. Subiu 0,8ºC com a porta aberta em 45º por 10 segundos;
5. Terminou equilibrada em 14ºC nos últimos 20 segundos com a porta fechada.

CONCLUSÃO

Com o experimento realizado, podemos comprovar que o projeto, tanto hardware quanto software, se comportou muito bem cumprindo sua tarefa principal, permitir a verificação e a comparação de medições de temperatura de uma forma simples e eficaz.

Com os dados obtidos no experimento, podemos notar que a variação da temperatura com a porta aberta em 90º (entre 10s e 20s) foi maior que a variação da temperatura com a porta aberta em 45º (entre 40s e 50s), mostrando que o ângulo de abertura da porta é diretamente proporcional à variação da temperatura. Também podemos notar que, mesmo a geladeira permanecendo o dobro do tempo com a porta fechada após a sua abertura em 90º, não foi suficiente para que sua temperatura voltasse ao seu valor anterior, 11,8ºC. Para a geladeira recuperar a temperatura perdida durante os 10 segundos com a porta aberta em 90º e retornar de 13,2ºC para 11,8ºC, ela levaria aproximadamente 120 segundos, um valor 12 vezes maior.

Um ponto à ser melhorado em projetos futuros é a montagem dos sensores. A utilização de muitos fios conectados à protoboard acaba dificultando o manuzeio. Fica a proposta de implantação de conectores nos fios dos sensores, construção de uma case para facilitar o tranporte e manuzeio e a transferência do projeto para uma placa de circuito impresso para redução do tamanho do hardware. Pode-se também acoplar outros sensores, como um sensor de controle de pressão e vazão, para que o projeto possa ser utilizado em outras aplicações. Muito importante também é calibrar o sistema para melhorar sua precisão.

.

REFERÊNCIAS

[1]. **ARDUINO**. Disponível em: http://www.arduino.cc. Acesso em: out. 2015.

[2]. **SCILAB**, versão 5.5.2. [S.l.]: Scilab Enterprises. Disponível em: http://www.scilab.org/download/5.5.2. Acesso em: out. 2015.

[3]. **DS18B20 Programmable Resolution 1-Wire Digital Thermometer**. [S.l.]: Maxim Integrated$_{TM}$. Disponível em: http://datasheets.maximintegrated.com/en/ds/DS18B20.pdf. Acesso em: out. 2015.

[4]. **LM35 Precision Centigrade Temperature Sensors**. [S.l.]: All Datasheet. Disponível em: http://html.alldatasheet.com/html-pdf/517588/TI1/LM35/51/1/LM35.html. Acesso em: jun. 2017

[5]. **ARDUINO**. Wikipedia, a enciclopédia livre. Disponível em: http://pt.wikipedia.org/wiki/Arduino#Software. Acesso em: out. 2015.

[6]. **JSON**. Wikipedia, a enciclopédia livre. Disponível em: https://pt.wikipedia.org/wiki/JSON. Acesso em: out. 2015.

[7]. BURTON, Miles. **Dallas Temperature Control Library**. Disponível em: http://www.hacktronics.com/code/DallasTemperature.zip. Acesso em: set. 2015.

[8]. STOFFREGEN, Paul. **One Wire Library**. Disponível em: http://www.pjrc.com/teensy/arduino_libraries/OneWire.zip. Acesso em: set. 2015.

[9]. SEGRE, Enrico; SENGUPTA, Aditya. **Serial Communication Toolbox**. Disponível em: https://atoms.scilab.org/toolboxes/serial. Acesso em: set. 2015.

[10]. SENGUPTA, Aditya. **Scilab JSON Toolbox**. Disponível em: https://atoms.scilab.org/toolboxes/JSON. Acesso em: set. 2015.

[11]. EMBARCADOS. Disponível em: https://www.embarcados.com.br/arduino-uno/. Acesso em: jun. 2017.

[12]. ROBOTIZANDO. Disponível em: http://www.robotizando.com.br/curso_arduino_hardware_pg1.php. Acesso em: jun. 2017.

[13]. FILIPEFLOP. Disponível em: http://www.filipeflop.com/pd-1e7d0e-sensor-de-temperatura-ds18b20-a-prova-d-agua.html. Acesso em: jun. 2017.

[14]. ARDUINOBR. Disponível em: http://www.arduinobr.com/arduino/arduino-sensor/como-medir-temperatura-com-um-ds18b20/. Acesso em: jun. 2017.

[15]. ZOOM. Disponível em: https://www.zoom.com.br/geladeira/consul-frost-free-bem-estar-crb36ab-300-litros-branco. Acesso em: jun. 2017.

APÊNDICE A - Código Arduino

```
/*-----( Import needed libraries )-----*/
#include <OneWire.h>
#include <DallasTemperature.h>

// Porta do pino de sinal do DS18B20
#define ONE_WIRE_BUS 10

// Define uma instancia do oneWire para comunicacao com o sensor
OneWire oneWire(ONE_WIRE_BUS);

DallasTemperature sensors(&oneWire);
DeviceAddress sensor0;
DeviceAddress sensor1;

int adress0;
int adress1;

void setup(void)
{
  Serial.begin(9600);
  sensors.begin();
}

void loop()
{
  // Get sensor adress
  adress0 = sensors.getAddress(sensor0, 0);
  adress1 = sensors.getAddress(sensor1, 1);

  // Read sensors
  sensors.requestTemperatures();
```

```
  float temp_0 = sensors.getTempC(sensor0);
  float temp_1 = sensors.getTempC(sensor1);

  //Print JSON format
  Serial.println("{");

  Serial.print("'sensors':[");
  Serial.print(adress0);
  Serial.print(",");
  Serial.print(adress1);
  Serial.println("],");

  Serial.print("'temp':[");
  Serial.print(temp_0);
  Serial.print(",");
  Serial.print(temp_1);
  Serial.println("]");

  Serial.println("}");
  delay(1000);
}
```

APÊNDICE B - Código Scilab

```
Close; clear; clearglobal; clc;


global %time

%time = 0;

global %temp

%temp = [];

global %temp2

%temp2 = [];

global %data

%data = [];

global %data2

%data2 = [];

global %MaxTemp

%MaxTemp = 25;

global %MinTemp

%MinTemp = 20;

global %MaxTemp2

%MaxTemp2 = 25;

global %MinTemp2

%MinTemp2 = 20;

global %Export

%Export = 0;

global %stability_time

%stability_time = 30;

global %stability_value
```

```
%stability_value = .3;
global %warning
%warning = [%t, %t];
//
global timeBuffer
global timeBuffer2
global minTempDisplay
global minTempDisplay2
global maxTempDisplay
global maxTempDisplay2
global minRegulationDisplay
global minRegulationDisplay2
global maxRegulationDisplay
global maxRegulationDisplay2
//
top_axes_bounds = [0.230 0 0.535 0.5];
bottom_axes_bounds = [0.230 0.5 0.535 0.5];
//
timeBuffer = 300;
minTempDisplay = 15;
maxTempDisplay = 50;
minRegulationDisplay = minTempDisplay + 273.15;
maxRegulationDisplay = maxTempDisplay + 273.15;
//
timeBuffer2 = 300;
minTempDisplay2 = 15;
maxTempDisplay2 = 50;
```

```
minRegulationDisplay2 = minTempDisplay2 + 273.15;
maxRegulationDisplay2 = maxTempDisplay2 + 273.15;
//
exec("..\etc\ScilabLib\Serial_Communication_Toolbox\0.4.1-2\loader.sce", -1)
exec("..\etc\ScilabLib\JSON\loader.sce", -1)
//
com =-1;
error_number = 999;
while (error_number <> 0)
   com = com+1;
   try
      %serial_port=openserial(com,"9600,n,8,1");
   end
   [error_message,error_number]=lasterror(%t)
   if com == 10 then
      disp("Arduino not found or permission denied")
      abort
   end
end
//
f=figure("dockable","off", "menubar", "none");
f.figure_position = [25, 58];
f.figure_name="Real-time Temperature Monitoring and Control";
f.figure_size = [1200 700];
f.background = color(246,244,242);
f.resize="off";
f.menubar_visible="on";
```

```
f.toolbar_visible="off";
f.info_message=f.figure_name
f.tag="mainWindow";
f.closerequestfcn="closeFigure";
//
bar(.5,0,'blue');
ge = gce();
ge = ge.children(1);
ge.tag = "instantSensor";
//
plot([0, 1], [%MinTemp, %MinTemp]);
e = gce();
e = e.children(1);
e.tag = "instantMinTemp";
e.line_style = 5;
e.thickness = 2;
e.foreground = color("orange");
//
plot([0, 3], [%MaxTemp, %MaxTemp]);
e = gce();
e = e.children(1);
e.tag = "instantMaxTemp";
e.line_style = 5;
e.thickness = 2;
e.foreground = color("red");
//
a = gca();
```

```
a.data_bounds = [0, minTempDisplay; 1, maxTempDisplay];

a.grid = [-1, color("darkgrey")];

a.axes_bounds = [0.05, 0.105, 0.25, .95];

a.axes_visible(1) = "off";

a.tag = "liveAxes";

a.tight_limits="on";

//

a = newaxes();

bar(.5,0,'blue');

e = gce();

e = e.children(1);

e.tag = "instantSensor2";

//

plot([0, 1], [%MinTemp2, %MinTemp2]);

e = gce();

e = e.children(1);

e.tag = "instantMinTemp2";

e.line_style = 5;

e.thickness = 2;

e.foreground = color("orange");

//

plot([0, 3], [%MaxTemp2, %MaxTemp2]);

e = gce();

e = e.children(1);

e.tag = "instantMaxTemp2";

e.line_style = 5;

e.thickness = 2;
```

```
e.foreground = color("red");
//
a.data_bounds = [0, minTempDisplay2; 1, maxTempDisplay2];
a.grid = [-1, color("darkgrey")];
a.axes_bounds = [.765, 0.105, 0.25, .95];
a.axes_visible(1) = "off";
a.tag = "liveAxes2";
a.tight_limits="on";
//
subplot(222);
a = gca();
a.axes_bounds = top_axes_bounds;
a.tight_limits="on";
a.axes_reverse = ['on', 'off', 'off'];
a.grid=[-1, color("darkgrey")]
a.tag = "sensor1Axes";
plot2d(0:timeBuffer, zeros(1,timeBuffer + 1)-50, color("red"));
a.title.text="Temperature variations of the sensor 1";
a.data_bounds = [0, minTempDisplay; timeBuffer, maxTempDisplay];
e = gce();
e = e.children(1);
e.tag = "Sensor1";
//
a = newaxes();
a.y_location = "right";
a.filled = "off"
a.grid=[-1, color("lightblue")]
```

```
a.tag = "sensor1NewAxes";
a.tight_limits="on"
a.axes_bounds = top_axes_bounds;
plot2d(0:timeBuffer, zeros(1,timeBuffer + 1)-50, color("blue"));
a.data_bounds = [0, minRegulationDisplay; timeBuffer, maxRegulationDisplay];
a.axes_visible(1) = "off";
a.foreground=color("blue");
a.font_color=color("blue");
e = gce();
e = e.children(1);
e.tag = "minute1Regulation";
//
subplot(224);
a = gca();
a.axes_bounds = bottom_axes_bounds;
a.tight_limits="on";
a.axes_reverse = ['on', 'off', 'off'];
a.grid=[-1, color("darkgrey")]
a.tag = "sensor2Axes";
plot2d(0:timeBuffer2, zeros(1,timeBuffer2 + 1)-50, color("red"));
a.title.text="Temperature variations of the sensor 2";
a.data_bounds = [0, minTempDisplay2; timeBuffer2, maxTempDisplay2];
e = gce();
e = e.children(1);
e.tag = "Sensor2";
//
a = newaxes();
```

```
a.y_location = "right";
a.filled = "off"
a.grid=[-1, color("lightblue")]
a.tag = "sensor2NewAxes";
a.tight_limits="on"
a.axes_bounds = bottom_axes_bounds;
plot2d(0:timeBuffer2, zeros(1,timeBuffer2 + 1)-50, color("blue"));
a.data_bounds = [0, minRegulationDisplay2; timeBuffer2, maxRegulationDisplay2];
a.axes_visible(1) = "off";
a.foreground=color("blue");
a.font_color=color("blue");
e = gce();
e = e.children(1);
e.tag = "minute2Regulation";
//
funcprot(0)
//
function launchSensor()
   global %MaxTemp
   global %MaxTemp2
   global %serial_port
   global %Acquisition
   global %time
   global %temp
   global %temp2
   global %data
   global %data2
```

```
global %stability_time
global %stability_value
global %warning
%Acquisition = %t;
readserial(%serial_port);
//
while(%Acquisition)
   //
   values = [];
   value = ascii(0);
   ln = 1;
   v="";
   v2="";
   //
   while (value ~= "{") do
      value = readserial(%serial_port,1);
   end
   //
   if (value == '{') then
      values(ln,1) = value;
   end
   //
   while (value ~= '}') do
      value =  readserial(%serial_port,1);
      //
      if (ascii(value) < 33 | ascii(value) > 125) then
         ln = ln+1;
```

```
      while(ascii(value) < 33 | ascii(value) > 125)
         value = readserial(%serial_port,1);
      end
      //
      values(ln,1) = value;
      continue
   end
   //
   values(ln,1) = values(ln,1) + value;
end
//
json = JSONParse(values);
//
if (json.sensors(1) == 1) then
   v=json.temp(1);
end
//
if (json.sensors(2) == 1) then
   v2=json.temp(2);
end
//
if (v~="" & v2~="") then
   xinfo("Temp1 = "+string(v)+"°C / "+string(v+273.15)+" K" + ...
   "    |    Temp2 = "+string(v2)+"°C / "+string(v2+273.15)+" K");
   %data = [%data v]
   %data2 = [%data2 v2]
   %time = length(%data)-1
```

```
    %temp = v;
    %temp2 = v2;
    updateSensorValue(v, v2);
  elseif (v~="" & v2=="") then
    xinfo("Temp1 = "+string(v)+"°C / "+string(v+273.15)+" K");
    %data = [%data v]
    %data2 = [%data2 -1]
    %time = length(%data)-1
    %temp = v;
    %temp2 = -1;
    updateSensorValue(v, -1);
  elseif (v=="" & v2~="") then
    xinfo("Temp2 = "+string(v2)+"°C / "+string(v2+273.15)+" K");
    %data = [%data -1]
    %data2 = [%data2 v2]
    %time = length(%data)-1
    %temp = -1;
    %temp2 = v2;
    updateSensorValue(-1, v2);
  end
  //
  if (%time>%stability_time) then
    resp1=0;
    resp2=0;
    if (sum(abs(diff(%data($-%stability_time+1 : $)))) < %stability_value) & (%warning(1) == %t)
then
      resp1 = messagebox("Sensor 1 is stable", "Info", "info", ["Continue" "Stop"], "modal")
```

```
        elseif  (sum(abs(diff(%data2($-%stability_time+1 : $)))) < %stability_value) & (%warning(2) ==
%t) then
            resp2 = messagebox("Sensor 2 is stable", "Info", "info", ["Continue" "Stop"], "modal")
        end
        //
        if resp1 == 1 then
            %warning(1) = %f
        end
        if resp2 == 1 then
            %warning(2) = %f
        end
        if resp1 == 2 | resp2 == 2 then
            stopSensor();
        end
      end
   end
endfunction
//
function stopSensor()
   global %Acquisition
   %Acquisition = %f;
endfunction
//
function closeFigure()
   stopSensor();
   exportValues();
   global %serial_port
   closeserial(%serial_port);
```

```
    f = findobj("tag", "mainWindow");
    delete(f);
endfunction
//
function updateSensorValue(data, data2)
    global %MaxTemp
    global %MaxTemp2
    global %MinTemp
    global %MinTemp2
    //
    if data ~= -1 then
        e = findobj("tag", "instantSensor");
        e.data(2) = data;
        if data > %MaxTemp then
            e.background = color("red");
        elseif data < %MinTemp  then
            e.background = color("blue");
        else
            e.background = color("green");
        end
        //
        e = findobj("tag", "Sensor1");
        lastPoints = e.data(:, 2);
        e.data(:, 2) = [data ; lastPoints(1:$-1)];
    end
    //
    if data2 ~= -1 then
```

```
        e = findobj("tag", "instantSensor2");
        e.data(2) = data2;
        if data2 > %MaxTemp2 then
            e.background = color("red");
        elseif data2 < %MinTemp2  then
            e.background = color("blue");
        else
            e.background = color("green");
        end
        //
        e = findobj("tag", "Sensor2");
        lastPoints = e.data(:, 2);
        e.data(:, 2) = [data2 ; lastPoints(1:$-1)];
    end
    //
    update();
endfunction
//
function update()
    global %temp
    global %temp2
    global %time
    //
    if %temp ~= -1 then
        set(tempValue, "string", string(%temp)+"ºC", "fontsize", 25);
        //
        set(timeValue, "string", string(%time)+"s", "fontsize", 25);
```

```
    end
    //
    if %temp2 ~= -1 then
        set(tempValue2, "string", string(%temp2)+"ºC", "fontsize", 25);
        //
        set(timeValue2, "string", string(%time)+"s", "fontsize", 25);
    end
endfunction
//
function changeMinTemp()
    global %MinTemp
    e = findobj("tag", "minTempSlider");
    %MinTemp = e.value;
    e = findobj("tag", "instantMinTemp");
    e.data(:,2) = %MinTemp;
endfunction
//
function changeMaxTemp()
    global %MaxTemp
    e = findobj("tag", "maxTempSlider");
    %MaxTemp = e.value;
    e = findobj("tag", "instantMaxTemp");
    e.data(:,2) = %MaxTemp;
endfunction
//
function changeMinTemp2()
    global %MinTemp2
```

```
    e = findobj("tag", "minTempSlider2");
    %MinTemp2 = e.value;
    e = findobj("tag", "instantMinTemp2");
    e.data(:,2) = %MinTemp2;
endfunction
//
function changeMaxTemp2()
    global %MaxTemp2
    e = findobj("tag", "maxTempSlider2");
    %MaxTemp2 = e.value;
    e = findobj("tag", "instantMaxTemp2");
    e.data(:,2) = %MaxTemp2;
endfunction
//
function resetDisplay()
    exportValues()
    global %time
    %time = 0;
    global %temp
    %temp = [];
    global %temp2
    %temp2 = [];
    global %data
    %data = [];
    global %data2
    %data2 = [];
    global %serial_port
```

```
    readserial(%serial_port);
    //
    update();
    xinfo(f.figure_name);
    //
    e = findobj("tag", "instantSensor");
    e.data(:, 2) = 0;
    e = findobj("tag", "Sensor1");
    e.data(:, 2) = 0;
    e = findobj("tag", "minute1Regulation");
    e.data(:, 2) = 0;
    //
    e = findobj("tag", "instantSensor2");
    e.data(:, 2) = 0;
    e = findobj("tag", "Sensor2");
    e.data(:, 2) = 0;
    e = findobj("tag", "minute2Regulation");
    e.data(:, 2) = 0;
endfunction
//
function exportValues()
    global %data
    global %data2
    global %Export
    if %data == [] & %data2 == [] then
        return
    end
```

```
    tempo = 0:size(%data,2)-1;
    M = [tempo' %data' %data2'];


    csvWrite(M, "../data/tempData_" + string(%Export) + ".csv", ";", [], "%g");
    %Export = %Export+1;
endfunction
//
function popupCallback ()
    obj = findobj("tag", "acqButton");
    val = get ( obj, 'value' );


    if val == 1 then
        launchSensor();
    else
        stopSensor();
    end
endfunction
//
ValueFrame = uicontrol(f, "style", "frame", "position", [15 480 305 80]...
,"tag", "valueFrame", "ForegroundColor", [0/255 0/255 0/255],...
"border", createBorder("titled", createBorder("line", "lightGray", 1), ...
_("View Panel"), "center", "top", createBorderFont("", 11, "normal"),...
 "black"));
//
timeLabel = uicontrol(f, "style", "text", "position", ...
[45 505 40 30], "string", "t = ", "fontsize", 25);
//
```

```
timeValue = uicontrol(f, "style", "text", "position", ...
[85 505 60 30], "string", string(%time)+"s", "fontsize", 25)
//
tempLabel = uicontrol(f, "style", "text", "position", ...
[160 505 50 30], "string", "T = ", "fontsize", 25);
//
tempValue = uicontrol(f, "style", "text", "position", ...
[210 505 105 30], "string", string(%temp)+"ยบC", "fontsize", 25)
//
minTempSlider = uicontrol("style", "slider", "position", [45 30 15 440], ...
"min", minTempDisplay, "max", maxTempDisplay, "sliderstep", [1 5], "value" , %MinTemp, ...
"callback", "changeMinTemp", "tag", "minTempSlider");
//
maxTempSlider = uicontrol("style", "slider", "position", [20 30 15 440], ...
"min", minTempDisplay, "max", maxTempDisplay, "sliderstep", [1 5], "value" , %MaxTemp, ...
"callback", "changeMaxTemp", "tag", "maxTempSlider");
//
ValueFrame2 = uicontrol(f, "style", "frame", "position", [865 480 305 80]...
,"tag", "valueFrame", "ForegroundColor", [0/255 0/255 0/255],...
"border", createBorder("titled", createBorder("line", "lightGray", 1), ...
_("View Panel"), "center", "top", createBorderFont("", 11, "normal"),...
 "black"));
//
timeLabel2 = uicontrol(f, "style", "text", "position", ...
[895 505 40 30], "string", "t = ", "fontsize", 25);
//
timeValue2 = uicontrol(f, "style", "text", "position", ...
```

```
[935 505 60 30], "string", string(%time)+"s", "fontsize", 25)
//
tempLabel2 = uicontrol(f, "style", "text", "position", ...
[1010 505 50 30], "string", "T = ", "fontsize", 25);
//
tempValue2 = uicontrol(f, "style", "text", "position", ...
[1060 505 105 30], "string", string(%temp2)+"องC", "fontsize", 25)
//
minTempSlider2 = uicontrol("style", "slider", "position", [895 30 15 440], ...
"min", minTempDisplay2, "max", maxTempDisplay2, "sliderstep", [1 5], "value" , %MinTemp2, ...
"callback", "changeMinTemp2", "tag", "minTempSlider2");
//
maxTempSlider2 = uicontrol("style", "slider", "position", [870 30 15 440], ...
"min", minTempDisplay2, "max", maxTempDisplay2, "sliderstep", [1 5], "value" , %MaxTemp2, ...
"callback", "changeMaxTemp2", "tag", "maxTempSlider2");
//
mAcq=uimenu(f,'label', 'Acquisition');
mSetup=uimenu(f,'label', "Setup");
mExport=uimenu(f,'label', "Export Data", "callback", "exportValues");
//
mAcq1=uimenu(mAcq,'label', "Start", "callback", "launchSensor");
mAcq2=uimenu(mAcq,'label', "Stop", "callback", "stopSensor");
mAcq3=uimenu(mAcq,'label', "Reset", "callback", "resetDisplay");
//
mSensor1=uimenu(mSetup,'label', "Sensor 1");
mSensor2=uimenu(mSetup,'label', "Sensor 2");
mStability=uimenu(mSetup,'label', "Stability");
```

```
//
mBuffer1=uimenu(mSensor1,'label', "Time Buffer", "callback", "changeBuffer(1)");
mMinTemp1=uimenu(mSensor1,'label', "Min Temp Display", "callback", "setupMinTemp(1)");
mMaxTemp1=uimenu(mSensor1,'label', "Max Temp Display", "callback", "setupMaxTemp(1)");
//
mBuffer2=uimenu(mSensor2,'label', "Time Buffer", "callback", "changeBuffer(2)");
mMinTemp2=uimenu(mSensor2,'label', "Min Temp Display", "callback", "setupMinTemp(2)");
mMaxTemp2=uimenu(mSensor2,'label', "Max Temp Display", "callback", "setupMaxTemp(2)");
//
mStabilityValue=uimenu(mStability,'label', "Stability value", "callback", "setupStability(1)");
mStabilityTime=uimenu(mStability,'label', "Stability time", "callback", "setupStability(2)");
//
function changeBuffer(id)
  newBuffer=evstr(x_dialog('Set new time buffer value (seconds): ','300'))
  //
  if newBuffer == [] then
    return
  elseif id == 1 then
    global timeBuffer
    timeBuffer = newBuffer;
    a = findobj("tag", "sensor1Axes");
    e = findobj("tag", "sensor1NewAxes");
    //
    a.data_bounds = [0, minTempDisplay; timeBuffer, maxTempDisplay];
    e.data_bounds = [0, minRegulationDisplay; timeBuffer, maxRegulationDisplay];
  elseif id == 2 then
    global timeBuffer2
```

```
        timeBuffer2 = newBuffer;
        a = findobj("tag", "sensor2Axes");
        e = findobj("tag", "sensor2NewAxes");
        //
        a.data_bounds = [0, minTempDisplay2; timeBuffer2, maxTempDisplay2];
        e.data_bounds = [0, minRegulationDisplay2; timeBuffer2, maxRegulationDisplay2];
    end
endfunction
//
function setupMinTemp(id)
    newMinTemp=evstr(x_dialog('Set new min temperature value: ','15'))
    //
    if newMinTemp == [] then
        return
    elseif id == 1 then
        global minTempDisplay
        global minRegulationDisplay
        minTempDisplay = newMinTemp;
        minRegulationDisplay = minTempDisplay + 273.15;
        a = findobj("tag", "sensor1Axes");
        e = findobj("tag", "sensor1NewAxes");
        l = findobj("tag", "liveAxes");
        minS = findobj("tag", "minTempSlider");
        maxS = findobj("tag", "maxTempSlider");
        //
        a.data_bounds = [0, minTempDisplay; timeBuffer, maxTempDisplay];
        e.data_bounds = [0, minRegulationDisplay; timeBuffer, maxRegulationDisplay];
```

```
        l.data_bounds = [0, minTempDisplay; 1, maxTempDisplay];
        minS.min = minTempDisplay;
        maxS.min = minTempDisplay;
    elseif id == 2 then
        global minTempDisplay2
        global minRegulationDisplay2
        minTempDisplay2 = newMinTemp;
        minRegulationDisplay2 = minTempDisplay2 + 273.15;
        a = findobj("tag", "sensor2Axes");
        e = findobj("tag", "sensor2NewAxes");
        l = findobj("tag", "liveAxes2");
        minS = findobj("tag", "minTempSlider2");
        maxS = findobj("tag", "maxTempSlider2");
        //
        a.data_bounds = [0, minTempDisplay2; timeBuffer2, maxTempDisplay2];
        e.data_bounds = [0, minRegulationDisplay2; timeBuffer2, maxRegulationDisplay2];
        l.data_bounds = [0, minTempDisplay2; 1, maxTempDisplay2];
        minS.min = minTempDisplay2;
        maxS.min = minTempDisplay2;
    end
endfunction
//
function setupMaxTemp(id)
    newMaxTemp=evstr(x_dialog('Set new max temperature value: ','50'))
    //
    if newMaxTemp == [] then
        return
```

```
elseif id == 1 then
    global maxTempDisplay
    global maxRegulationDisplay
    maxTempDisplay = newMaxTemp;
    maxRegulationDisplay = maxTempDisplay + 273.15;
    a = findobj("tag", "sensor1Axes");
    e = findobj("tag", "sensor1NewAxes");
    l = findobj("tag", "liveAxes");
    minS = findobj("tag", "minTempSlider");
    maxS = findobj("tag", "maxTempSlider");
    //
    a.data_bounds = [0, minTempDisplay; timeBuffer, maxTempDisplay];
    e.data_bounds = [0, minRegulationDisplay; timeBuffer, maxRegulationDisplay];
    l.data_bounds = [0, minTempDisplay; 1, maxTempDisplay];
    minS.max = maxTempDisplay;
    maxS.max = maxTempDisplay;
elseif id == 2 then
    global maxTempDisplay2
    global maxRegulationDisplay
    maxTempDisplay2 = newMaxTemp;
    maxRegulationDisplay2 = maxTempDisplay2 + 273.15;
    a = findobj("tag", "sensor2Axes");
    e = findobj("tag", "sensor2NewAxes");
    l = findobj("tag", "liveAxes2");
    minS = findobj("tag", "minTempSlider2");
    maxS = findobj("tag", "maxTempSlider2");
    //
```

```
        a.data_bounds = [0, minTempDisplay2; timeBuffer2, maxTempDisplay2];
        e.data_bounds = [0, minRegulationDisplay2; timeBuffer2, maxRegulationDisplay2];
        l.data_bounds = [0, minTempDisplay2; 1, maxTempDisplay2];
        minS.max = maxTempDisplay2;
        maxS.max = maxTempDisplay2;
    end
endfunction
//
function setupStability(opt)
    if opt == 1 then
        newStability=evstr(x_dialog('Set new stability value: ','0.3'))
        //
        if newStability == [] then
            return
        else
            global %stability_value
            %stability_value = newStability;
        end
    elseif opt == 2 then
        newStability=evstr(x_dialog('Set new stability time value (seconds): ','30'))
        //
        if newStability == [] then
            return
        else
            global %stability_time
            global %warning
            //
```

```
        %warning = [%t, %t];
        %stability_time = newStability;
      end
    end
endfunction
//
update();
```