Temperature monitoring Tutorial with Scilab/Xcos and Arduino

Document version 1 – Yann Debray - Scilab Enterprises © - 08/11/2015

This tutorial aims at acquiring a temperature signal through a Arduino board.

Configuration/Arduino Setup

In order to follow this tutorial you need the following configuration:

Software:

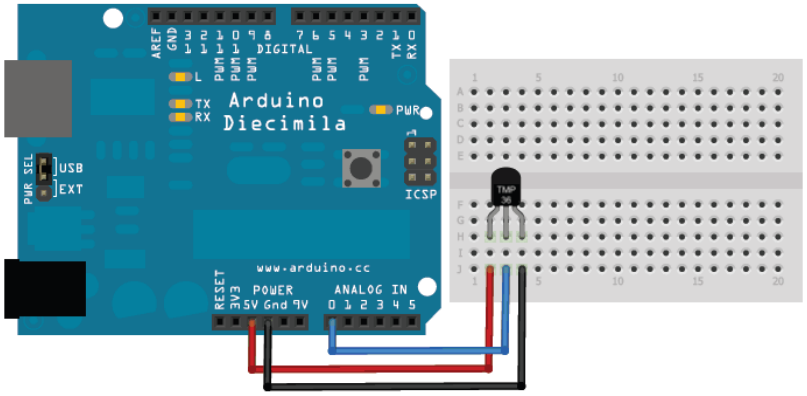
- Scilab on Windows 32 or 64 bits (Version >= 5.5.2)
- Arduino IDE http://arduino.cc/en/Main/Software
- Serial toolbox http://atoms.scilab.org/toolboxes/serial
- Arduino toolbox http://atoms.scilab.org/toolboxes/arduino
  Help on the installation of the module:
  https://www.scilab.org/en/community/education/si/install
- Analog displays https://fileexchange.scilab.org/toolboxes/312000

Hardware:

- Arduino Board (driver installation on
  http://www.arduino.cc/en/Guide/Windows#toc4)
- Breadboard, wires
- B & B Thermotechnik CON-TF-LM35DZ Temperature Sensor LM 35 DZ For Relative Humidity Detectors. -55 - +150 °C
  http://www.conrad.com/ce/en/product/156600/

Hardware Set-up

Set up the following hardware configuration:



Flash the firmware in the Arduino board

Plug your Arduino Board to your PC, open the Arduino IDE and flash the file *scilab_temp_reading.ino* on the Arduino Board.

```
1. //TMP36 Pin Variables
2. int sensorPin = 0; //the analog pin the TMP36's Vout (sense) pin is connected to
3.                        //the resolution is 10 mV / degree centigrade with a
4.                        //500 mV offset to allow for negative temperatures
5.
6. /*
7.  * setup() - this function runs once when you turn your Arduino on
8.  * We initialize the serial connection with the computer
9.  */
10. void setup()
11. {
12.   Serial.begin(9600);  //Start the serial connection with the computer
13.                        //to view the result open the serial monitor
14. }
15.
16. void loop()                     // run over and over again
17. {
18.  //getting the voltage reading from the temperature sensor
19.  int reading = analogRead(sensorPin);
20.
21.  // converting that reading to voltage, for 3.3v arduino use 3.3
22.  float voltage = reading * 5.0;
23.  voltage /= 1024.0;
24.
25.  // print out the voltage
26.  Serial.print(voltage); Serial.println(" volts");
27.
28.  // now print out the temperature
29.  float temperatureC = (voltage - 0.5) * 100 ;  //converting from 10 mv per degree wit
    500 mV offset
30.                                                 //to degrees ((voltage - 500mV) times
    100)
31.  Serial.print(temperatureC); Serial.println(" degrees C");
32.
33.  // now convert to Fahrenheit
34.  float temperatureF = (temperatureC * 9.0 / 5.0) + 32.0;
35.  Serial.print(temperatureF); Serial.println(" degrees F");
36.
37.  delay(1000);                                     //waiting a second
38. }
```

Scilab-side script for temperature acquisition

The temperature acquisition is directed through the serial communication from the Arduino board to the pc. The serial communication toolbox enables to get the data wired through this protocol in Scilab.

The way to exchange data with the serial communication is the following:

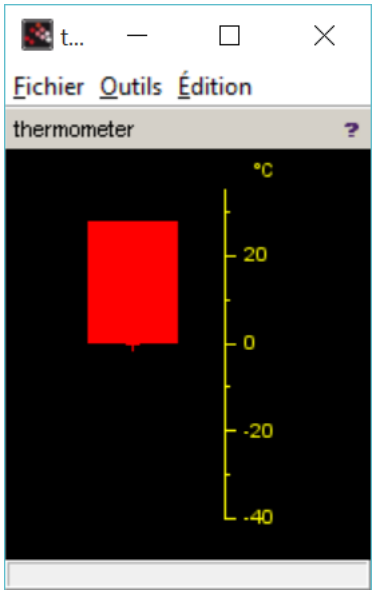
- Open serial communication
- Read/write on the serial communication
- Close the serial communication

First we open the serial communication on the COM port 1 (example running on Windows OS). We have to enter the communication parameters, in the form "baud,parity,data_bits,stop_bits":
-->h=openserial(1,"9600,n,8,1")

Second, we will in this case read the data on the serial port, coming from the temperature sensor on the Arduino board:
-->readserial(h)

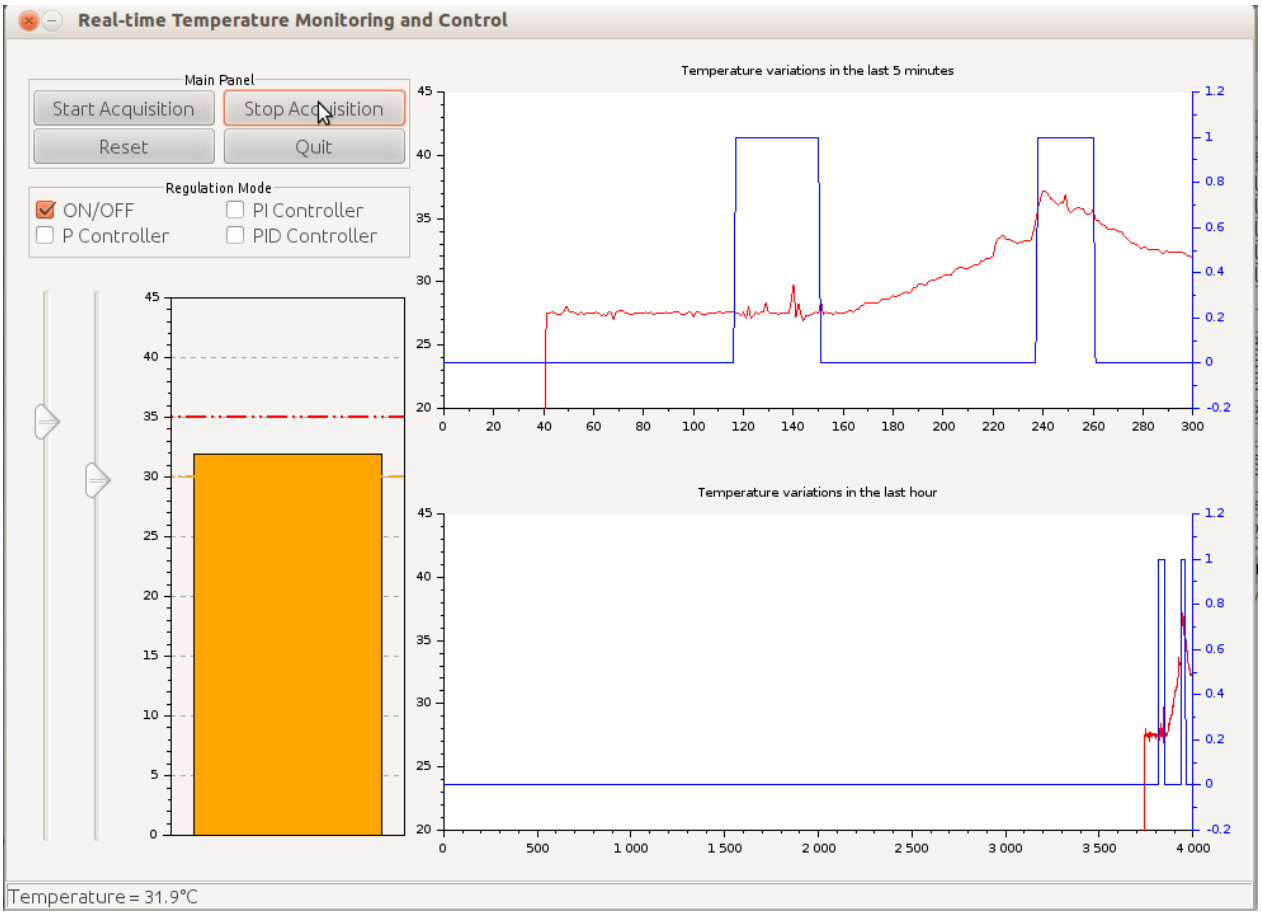
At last, as we have received the data, we have to close the serial communication:
-->closeserial(h)

Graphical User Interface 1: instant temperature value visualization



Cf. annexe 1 for the scilab code generating this graphical user interface

Graphical User Interface 2: historical temperature values visualization



Cf. annexe 2 for the scilab code generating this graphical user interface (with the serial communication)

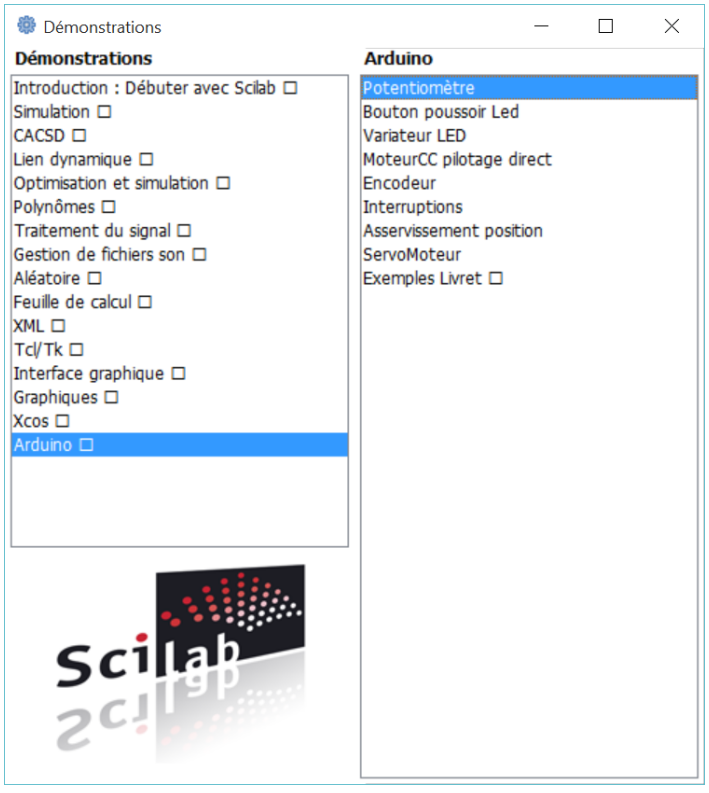
Alternative procedure with the Arduino toolbox for Xcos

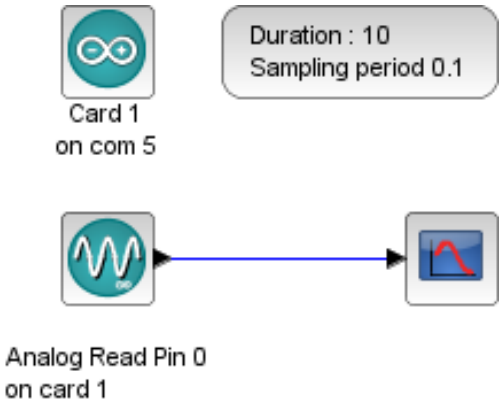
Go on the following website:
http://www.demosciences.fr/projets/scilab-arduino

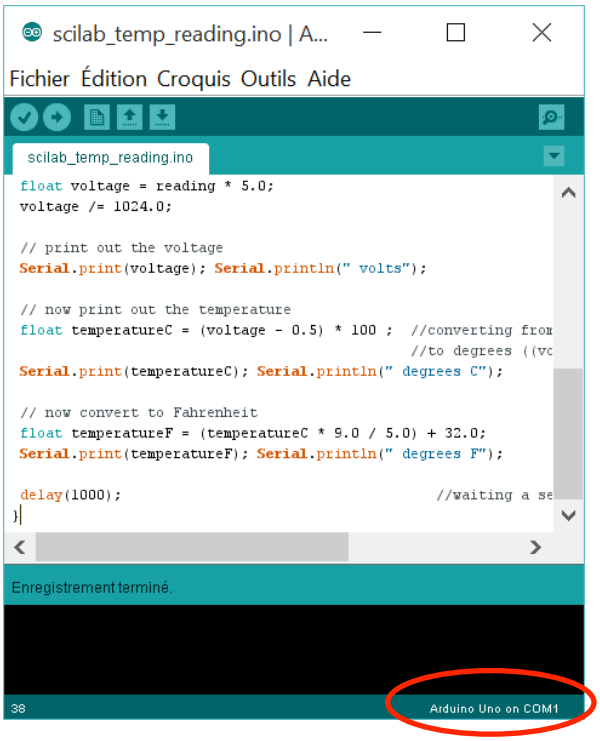
And download the following file:
toolbox_arduino_v3.ino

Plug your Arduino Board to your PC, open the Arduino IDE and flash the file toolbox_arduino_v3.ino on the Arduino Board.

This sketch is based on the demo “potentiometre” of the Arduino toolbox:



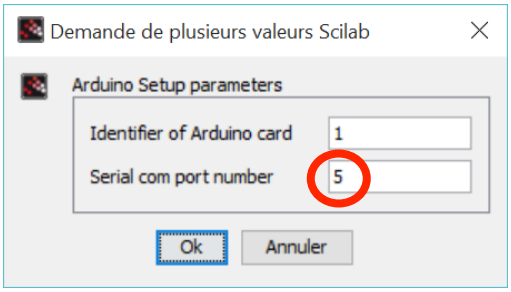




Start to build the Xcos schema, with the configuration blocks:

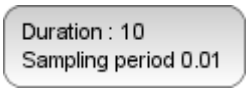
This allows a serial communication between Arduino and Scilab.

Double click on the block to let the following dialog box appear:

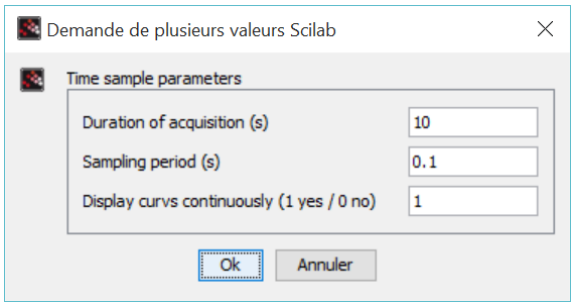


Set the Serial com port number with the information acquired in the previous step.

The sampling of the signal for the blocks of the model and the time of acquisition are configured by this block:



The sampling period can be specified and has to be at least twice smaller than the period of evolution of the model (Nyquist-Shannon sampling theorem)



Sources

Adafruit - Using a Temp Sensor
https://learn.adafruit.com/tmp36-temperature-sensor/using-a-temp-sensor

Real-time Temperature Monitoring and Control
https://fileexchange.scilab.org/toolboxes/311000/1.0

More readings

TP3 : Acquérir et piloter des systèmes à l'aide de cartes Arduino et d'une Toolbox Xcos dédiée - TP3 Démosciences 2012.pdf http://www.demosciences.fr/projets/scilab-arduino

Scilab / Xcos pour l'enseignement des sciences de l'ingénieur – © 2013 Scilab Enterprises chapitre « 4-acquisition et pilotage de moteur (module arduino) » - livret_Xcos.pdf
www.scilab.org/fr/content/download/1017/9485/file/livret_Xcos.pdf

Annexe 1: GUI 1 Scilab script

```
//THIS FUNCTION WILL CREATE A THERMOMETER LIKE DISPLAY OF THE INPUT DATA
//
//var1: 1xn vector          --> data to be displayd
//var2: string              --> data unit
//var3: string              --> figure name
//var4: [1x2] matrix        --> figure position
//var5: integer             --> figure color
//var6: integer             --> display color
//var7: integer             --> the time pause [in milliseconds] after that the
//                          next data point will be displayed.
function thermometer(data, unit, figname, figpos, figcolor, dispcolor, tstep)

CENTER = [0 0];
END    = max(size(data));

fignr = 1002;      // use big numbers, so normal figures won't be affected

f = figure(fignr);
delmenu(fignr,'Datei');
delmenu(fignr,'Zusatzprogramme');
delmenu(fignr,'Editieren');
delmenu(fignr,'?');
toolbar(fignr,'off');
f.background       = figcolor;
f.figure_size      = [5 300];
f.figure_name      = figname;
f.figure_position  = figpos;

// define the frame for the thermometer
rect=[-1,min(data),1,max(data)];
plot2d(0,0,0,rect=rect);
a = gca();
a.visible = "on";
a.box = "off";
a.margins = [0.1,0.4,0.1,0.1];
a.axes_visible = ["off","on","off"];
a.x_location = "middle";
a.y_location = "right";
y_label = a.y_label;
y_label.text = unit;
```

```
a.y_label.font_angle = 0;
a.y_label.position = [1.25 max(data)];

a.foreground               = dispcolor;
a.font_foreground          = dispcolor;
a.y_label.font_foreground  = dispcolor;

plot(CENTER(1), CENTER(2),'ro');
e = gce();
p = e.children(1);  // get the point as a handle
p.mark_style = 1;
p.mark_size = 6;

// display actual temperature

for i=1:END;
   drawlater();
   if data(i) == 0 then;
      rect=[-0.5,data(i),1,abs(data(i))];
      xrect(rect);
      a = gca();
      a.axes_visible = ["off","on","off"];
      e = a.children(1);
      e.background = 1;
      e.thickness = 1;
   elseif data(i) > 0;
      rect=[-0.5,data(i),1,abs(data(i))];
      xfrect(rect);
      a = gca();
      a.axes_visible = ["off","on","off"];
      e = a.children(1);
      e.foreground = 5;
      e.background = 5;
      e.thickness = 3;
   else
      rect=[-0.5,0,1,abs(data(i))];
      xfrect(rect);
      a = gca();
      a.axes_visible = ["off","on","off"];
      e = a.children(1);
      e.foreground = 2;
      e.background = 2;
      e.thickness = 3;
   end

   drawnow();
   xpause(tstep*1000);  // xpause counts in microseconds, tstep is for milliseconds, thatswhy factor 1000

   if  i < END
      delete(e);
   end
end

endfunction

clc;
temp = [0 -5 -10 -15 -25 -40 -10 -5 0 5 10 15 18 20 22 24 26 28 30 35 20 10 5 4 3 2 1 0];
thermometer (temp,'°C','thermometer',[500 300],1,7,50);
```

Annexe 2: GUI 2 Scilab script + serial communication

```
//
// Copyright (C) 2014 - A. Khorshidi <akhorshidi@live.com>
//
// This file is distributed in the hope that it will be useful;
// It must be used under the terms of the CeCILL.
// http://www.cecill.info/licences/Licence_CeCILL_V2.1-en.txt
//
//

// The following work provided the inspiration for this challenge.
// https://www.scilab.org/content/view/full/847
//
// I owe thanks to Bruno Jofret, the author of the original GUI.
// https://fileexchange.scilab.org/toolboxes/270000
//

ind = x_choose(["RS-232" ;"USB"; "Ethernet" ;"Wireless"],["Please select the type of communication interface: ";"Just double-click on
its name. "],"Cancel");
if ind==0 then
   msg=_("ERORR: No types of communication interfaces has been chosen. ");
   messagebox(msg, "ERROR", "error");
   error(msg);
   return;
elseif ind==2
   if (getos() == "Windows") then
      if ~(atomsIsInstalled('serial')) then
         msg=_("ERROR: A serial communication toolbox must be installed.");
         messagebox(msg, "Error", "error");
         error(msg);
         return;
      else
         flag=1;
      end
   elseif  (getos() == "Linux") then
      if ~(atomsIsInstalled('serialport')) & ~(atomsIsInstalled('serial')) then
         msg=_("ERROR: A serial communication toolbox must be installed.");
         messagebox(msg, "Error", "error");
         error(msg);
         return;
      elseif (atomsIsInstalled('serialport')) & (atomsIsInstalled('serial')) then
         stoolbx = x_choose(['serialport';'serial' ],"Which serial ...
         commiunication toolbox you prefer to use? "," Cancel ")
         if  stoolbx==1 then
            flag=2;
         elseif stoolbx==2 then
            flag=3;
         else
            msg=_("ERROR: No serial toolbox has been chosen. ");
            messagebox(msg, "Error", "error");
            error(msg);
            return;
         end
      elseif (atomsIsInstalled('serialport')) then
         flag=2;
      elseif  (atomsIsInstalled('serial')) then
         flag=3;
      end
   else
      msg=_(["WARNING: This program has been tested and works under Gnu/Linux ...
      and Windows."; "On other platforms you may need modify this script. "])
      messagebox(msg, "WARNING", "warning");
      warning(msg);
      return;
   end
```

```
else
   error("Not possible yet.");
   return;
end
//
if (getos() == "Linux") then
   [rep,stat,stderr]=unix_g("ls /dev/ttyACM*");
   if stderr ~= emptystr() then
      msg=_(["No USB device found. ";"Check your USB connection or try ...
         another port. "])
      messagebox(msg, "ERROR", "error");
      error(msg);
      return;
   end
   ind = x_choose(rep,["Please specify which USB port you wanna use for ...
      communication. ";"Just double-click on its name. "],"Cancel");
   if ind==0 then
      msg=_("ERORR: No serial port has been chosen. ");
      messagebox(msg, "ERROR", "error");
      error(msg);
      return;
   end
   port_name = rep(ind);
end
if (getos() == "Windows") then
   port_name=evstr(x_dialog('Please enter COM port number: ','13'))
   if port_name==[] then
      msg=_("ERORR: No serial port has been chosen. ");
      messagebox(msg, "ERROR", "error");
      error(msg);
      return;
   end
end
//
global %serial_port
if flag==2 then
   %serial_port = serialopen(port_name, 9600, 'N', 8, 1);
   while %serial_port == -1
      btn=messagebox(["Please check your USB connection, and then click on ...
      Try again. "; "To choose another port click on Change. "], "Error", ...
      "error", [" Try again " " Change "], "modal");
      if ~btn==1 then
         [rep,stat,stderr]=unix_g("ls /dev/ttyACM*");
         ind = x_choose(rep,["Please specify which USB port you wanna use...
         for communication. ";"Just double-click on its name. "],"Cancel");
      if ind==0 then
         msg=_("ERORR: No serial port has been chosen. ");
         messagebox(msg, "ERROR", "error");
         error(msg);
         return;
      end
      port_name = rep(ind);
      end
      %serial_port = serialopen(port_name, 9600, 'N', 8, 1);
   end
elseif flag==1 | flag==3
   %serial_port=openserial(port_name,"9600,n,8,1");
   //error(999)
else
   msg=_("ERROR: Could not specify which serial toolbox to use. ");
   messagebox(msg, "Error", "error");
   error(msg);
   return;
end
//
// * Monitoring Phase:
```

```
//
global %MaxTemp
%MaxTemp = 35;
global %MinTemp
%MinTemp = 30;
f=figure("dockable","off");
f.resize="off";
f.menubar_visible="off";
f.toolbar_visible="off";
f.figure_name="Real-time Temperature Monitoring and Control";
f.tag="mainWindow";
bar(.5,0,'blue');
e = gce();
e = e.children(1);
e.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, 0; 1, 45];
a.grid = [-1, color("darkgrey")];
a.axes_bounds = [0.1, 0.2, 0.25, 0.85];
a.axes_visible(1) = "off";
a.tag = "liveAxes";
//a.title.text="Current Temperature";
//
f.figure_position = [0 0];
f.figure_size = [1000 700];
f.background = color(246,244,242) //color("darkgrey")
//
minTempSlider = uicontrol("style", "slider", "position", [60 30 30 440], ...
"min", 0, "max", 45, "sliderstep", [1 5], "value" , %MinTemp, ...
"callback", "changeMinTemp", "tag", "minTempSlider");
maxTempSlider = uicontrol("style", "slider", "position", [20 30 30 440], ...
"min", 0, "max", 45, "sliderstep", [1 5], "value" , %MaxTemp, ...
"callback", "changeMaxTemp", "tag", "maxTempSlider");
//
// Functions:
function changeMinTemp()
    global %MinTemp
    e = findobj("tag", "minTempSlider");
    %MinTemp = e.value //45 - e.value;
    e = findobj("tag", "instantMinTemp");
    e.data(:,2) = %MinTemp;
endfunction
//
function changeMaxTemp()
    global %MaxTemp
    e = findobj("tag", "maxTempSlider");
    %MaxTemp = e.value //45 - e.value;
    e = findobj("tag", "instantMaxTemp");
    e.data(:,2) = %MaxTemp;
endfunction
```

```
//
function closeFigure()

    stopSensor();
    global %serial_port
    if flag == 2 then
      serialclose(%serial_port);
    elseif flag == 1 | flag == 3 then
       closeserial(%serial_port);
    end
    f = findobj("tag", "mainWindow");
    delete(f);
endfunction
//
function stopSensor()
    global %Acquisition
    %Acquisition = %f;
endfunction
//
function launchSensor()
    global %MaxTemp
    global %serial_port
    global %Acquisition
    %Acquisition = %t;
    global %fanStatus
    %fanStatus = 0;
    // Arduino toolbox
    values=[];
    value=ascii(0);
    while %Acquisition
      while(value~=ascii(13)) then
         if  flag == 2 then

             value=serialread(%serial_port,1);
         elseif flag == 1 | flag == 3 then
             value=readserial(%serial_port,1);
         end
       values=values+value;
       v=strsubst(values,string(ascii(10)),'')
       v=strsubst(v,string(ascii(13)),'')
       data=evstr(v)
       end
    //
    xinfo("Temperature = "+v+"°C");
    values=[]
    value=ascii(0);
    updateSensorValue(data);
    //
     global %RegulationEnable
    if  %RegulationEnable == 1 then
       if data > %MaxTemp then
          enableFan();
       else
          disableFan();
       end
    end
    updateFanValue(%fanStatus);
end
endfunction
//
function updateSensorValue(data)
    global %MaxTemp
    global %MinTemp
    e = findobj("tag", "instantSensor");
    e.data(2) = data;
    if data > %MaxTemp then
```

```
        e.background = color("red");
    else
        if data > %MinTemp  then
            e.background = color("orange");
        else
            e.background = color("green");
        end
    end
    //
    e = findobj("tag", "minuteSensor");
    lastPoints = e.data(:, 2);
    e.data(:, 2) = [lastPoints(2:$) ; data];
    e = findobj("tag", "hourSensor");
    lastPoints = e.data(:, 2);
    e.data(:, 2) = [lastPoints(2:$) ; data];
endfunction
//
// * Regulation Phase:
//
global %RegulationEnable
%RegulationEnable = 1;
global %PController
%PController = 0;
global %PIController
%PIController = 0;
global %PIDController
%PIDController = 0;
//
top_axes_bounds = [0.25 0 0.8 0.5];
bottom_axes_bounds = [0.25 0.5 0.8 0.5];
minTempDisplay = 20;
maxTempDisplay = 45;
minRegulationDisplay = -0.2;
maxRegulationDisplay = 1.2;
// Temperature variations in the last 5 minutes
timeBuffer = 300;
subplot(222);
a = gca();
a.axes_bounds = top_axes_bounds;
a.tag = "minuteAxes";
plot2d(0:timeBuffer, zeros(1,timeBuffer + 1), color("red"));
a.title.text="Temperature variations in the last 5 minutes";
a.data_bounds = [0, minTempDisplay; timeBuffer, maxTempDisplay];
e = gce();
e = e.children(1);
e.tag = "minuteSensor";
// adding a second vertical axis on the right side ...
// to show the On/Off status of the DC Fan.
a = newaxes();
a.y_location = "right";
a.filled = "off"
a.axes_bounds = top_axes_bounds;
plot2d(0:timeBuffer, zeros(1,timeBuffer + 1), 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 = "minuteRegulation";
// Temperature variations in the last hour
timeBuffer = 4000;
subplot(224);
a = gca();
a.axes_bounds = bottom_axes_bounds;
a.tag = "hourAxes";
```

```
plot2d(0:timeBuffer, zeros(1,timeBuffer + 1), color("red"));
a.title.text="Temperature variations in the last hour";
a.data_bounds = [0, minTempDisplay; timeBuffer, maxTempDisplay];
e = gce();
e = e.children(1);
e.tag = "hourSensor";
// 2nd vertical axis
a = newaxes();
a.y_location = "right";
a.filled = "off"
a.axes_bounds = bottom_axes_bounds;
a.axes_visible = "off";
plot2d(0:timeBuffer, zeros(1,timeBuffer + 1), 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 = "hourRegulation";
//
// Functions:
function resetDisplay()
    e = findobj("tag", "instantSensor");
    e.data(:, 2) = 0;
    e = findobj("tag", "minuteSensor");
    e.data(:, 2) = 0;
    e = findobj("tag", "hourSensor");
    e.data(:, 2) = 0;
    e = findobj("tag", "minuteRegulation");
    e.data(:, 2) = 0;
    e = findobj("tag", "hourRegulation");
    e.data(:, 2) = 0;
endfunction
//
function changeRegulationStatus()
    global %RegulationEnable
    e = findobj("tag", "enableRegulationCBO");
    %RegulationEnable = e.value;
    if %RegulationEnable == 0 then
        disableFan();
    end
endfunction
//
function updateFanValue(data)
    e = findobj("tag", "minuteRegulation");
    lastPoints = e.data(:, 2);
    e.data(:, 2) = [lastPoints(2:$) ; data];
    e = findobj("tag", "hourRegulation");
    lastPoints = e.data(:, 2);
    e.data(:, 2) = [lastPoints(2:$) ; data];
endfunction
//
function enableFan()
    global %serial_port
    if  flag == 2 then
        serialwrite(%serial_port,'H');
    elseif flag == 1 | flag == 3 then
        writeserial(%serial_port,ascii(72));
    end
    global %fanStatus
    %fanStatus = 1;
endfunction
//
function disableFan()
    global %serial_port
```

```
    if  flag == 2 then
        serialwrite(%serial_port,ascii(76));
    elseif flag == 1 | flag == 3 then
        writeserial(%serial_port,"L");
    end
    global %fanStatus
    %fanStatus = 0;
endfunction
//
// Buttons:
// * Main Panel
mainFrame = uicontrol(f, "style", "frame", "position", [15 560 305 80], ...
"tag", "mainFrame", "ForegroundColor", [0/255 0/255 0/255],...
"border", createBorder("titled", createBorder("line", "lightGray", 1)...
, _("Main Panel"), "center", "top", createBorderFont("", 11, "normal"), ...
"black"));
//
startButton = uicontrol(f, "style", "pushbutton", "position", ...
[20 595 145 30], "callback", "launchSensor", "string", "Start Acquisition", ...
"tag", "startButton");
//
stopButton = uicontrol(f, "style", "pushbutton", "position", ...
[170 595 145 30], "callback", "stopSensor", "string", "Stop Acquisition", ...
"tag", "stopButton");
//
resetButton = uicontrol(f, "style", "pushbutton", "position", ...
[20 565 145 30], "callback", "resetDisplay", "string", "Reset", ...
"tag", "resetButton");
//
quitButton = uicontrol(f, "style", "pushbutton", "position", ...
[170 565 145 30], "callback", "closeFigure", "string", "Quit", ...
"tag", "quitButton");
//
RegulationFrame = uicontrol(f, "style", "frame", "position", [15 490 305 65]...
,"tag", "mainFrame", "ForegroundColor", [0/255 0/255 0/255],...
"border", createBorder("titled", createBorder("line", "lightGray", 1), ...
_("Regulation Mode"), "center", "top", createBorderFont("", 11, "normal"),...
 "black"));
//
// * Regulation Mode
enableRegulation = uicontrol(f, "style", "checkbox", "position", ...
[20 520 140 20],"string", "ON/OFF", "value", %RegulationEnable, ...
"callback", "changeRegulationStatus", "tag", "enableRegulationCBO");
//
enableP = uicontrol(f, "style", "checkbox", "position", [20 500 140 20], ...
"string", "P Controller", "value", %PController, ...
"callback", "", "tag", "");
//
enablePI = uicontrol(f, "style", "checkbox", "position", [170 520 140 20], ...
"string", "PI Controller", "value", %PIController, ...
"callback", "", "tag", "");
//
enablePID = uicontrol(f, "style", "checkbox", "position", [170 500 140 20], ...
"string", "PID Controller", "value", %PIDController, ...
"callback", "", "tag", "");
//
```