IBF
Istituto di Biofisica
CNR – Consiglio Nazionale delle Ricerche

Rapporto Tecnico

Implementazione e sviluppo di una interfaccia grafica - GUI - per la gestione della temperatura del termostato Lauda Series E200 sfruttando l'interfaccia seriale RS232

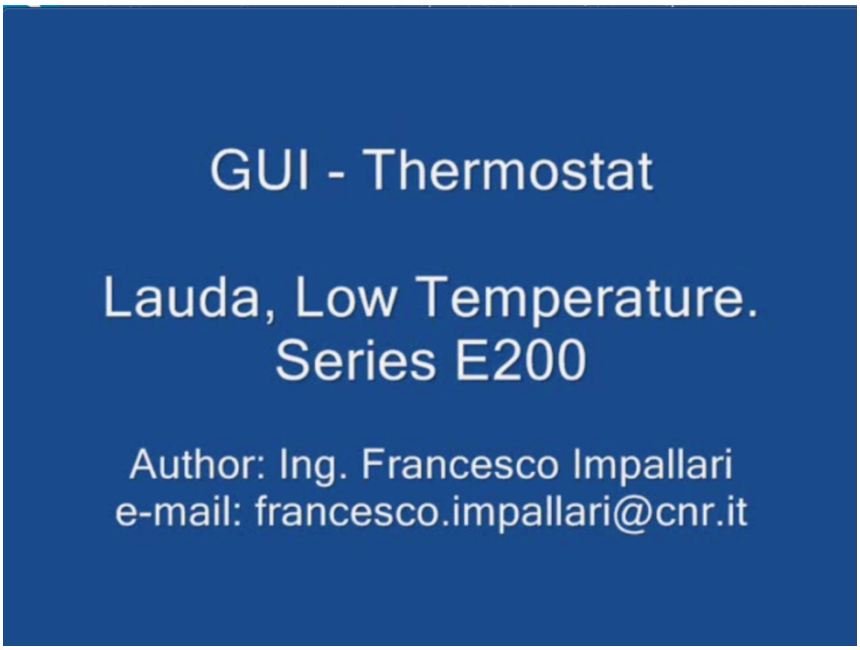
Ing. Francesco Impallari

Palermo, 14/09/2020

Sommario

1. Breve filmato di presentazione del software

Nel riquadro di sotto, è possibile visualizzare un breve filmato dell'interfaccia grafica (GUI) sviluppata; l'utente ha ampia possibilità di gestire e/o impostare le rampe di temperatura del termostato Lauda Series E200 grazie ad un collegamento seriale tramite cavo DB9 che sfrutta l'interfaccia RS232



N.B. Ho creato un link da dropbox (senza bisogno di essere registrati) per la visualizzazione del breve filmato:
https://www.dropbox.com/scl/fi/m5vwyau9evekomvtqpf8h/thermostat.mp4?rlkey=bdm82u4auidhbdfxljkap6kv9&st=fopuvsp7&dl=0

2. Introduzione

Il presente progetto, implementato e sviluppato presso l'Istituto di Biofisica di Palermo ha per oggetto quello di stabilire l'architettura hardware e software per ottenere una interfaccia grafica GUI, per la gestione, invio, visualizzazione dei dati e soprattutto per il setting delle rampe di temperatura al variare del tempo scelto.
Lo scambio delle informazioni avviene tramite un cavo seriale RS 232 collegata alla porta seriale del termostato Lauda Series E200 e a quella del PC in cui è presente il software sviluppato.

L'hardware necessario per il suo funzionamento consiste in:

- un PC con sistema operativo XP o superiore, dotato di porta COM o adattatore USB-RS232;
- un Cavo seriale DB9 per interfaccia RS 232;
- Termostato Lauda Series E200, dotato di porta seriale, presente nel mio Istituto di Biofisica.

E di un software open-source installato nel PC secondo le modalità previste:

- Eclipse IDE for Developers.

Fino a qualche anno fa si utilizzavano le linee seriali RS232 per collegare apparecchiature al PC. Ora questa linea di comunicazione è oramai obsoleta ed è stata sostituita dall'USB che ha in qualche modo monopolizzato la connessione di periferiche di ogni tipo con i PC. Esistono anche in commercio circuiti di conversione da USB a RS232 proprio per poter connettere un microcontrollore sprovvisto di USB ma munito di UART ad un PC per gli usi più disparati.

In questo progetto, la nostra apparecchiatura del termostato (presente nel laboratorio Light Scattering dell'Istituto di Biofisica di Palermo) è molto vecchiotta, di almeno 15 anni, e quindi ho creato una interfaccia di collegamento, tra apparecchiatura e PC, configurando un cavo seriale DB9 con due connettori RS 232 da 9 PIN seguendo lo schema del manuale di funzionamento del termostato stesso:

Computer					**Thermostat RS 232 Interface**		
Data	**9-pin sub-D socket**		**25-pin sub-D socket**		**9-pin sub-D socket**		**Data**
	①	②	①	②	①	②	
R x D	2	2	3	3	2	2	T x D
T x D	3	3	2	2	3	3	R x D
DTR	4		20		4		DSR
Signal Ground	5	5	7	7	5	5	Signal Ground
DSR	6		6		6		DTR
RTS	7		4		7	7	CTS
CTS	8		5		8	8	RTS

Per provare le comunicazioni seriali e il corretto funzionamento del cavo DB9 ho realizzato un programma di emulazione di terminale usando l'HyperTerminal del PC. Essere in grado di fare un programma di terminale vuol dire essere in grado di fare qualsiasi altro programma che scriva e legga in una linea seriale.

Per lo sviluppo di programmazione della nostra GUI la scelta è caduta sul linguaggio JAVA per una serie di motivi:

- è un bel linguaggio ad oggetti e quindi molto semplice da programmare per sviluppare il codice;
- è multi-piattaforma. Java funziona bene sui PC con Windows o sistema operativo superiore, Linux e su MAC;
- il sistema di sviluppo Eclipse è ottimo, potente, facile da utilizzare e completamente open-source;
- ci sono parecchi tutorial in rete che ne rendono facile l'apprendimento del linguaggio Java;
- le linee seriali si gestiscono con facilità tramite il pacchetto RxTx;

3. Installazione pacchetto RxTx

Il Java non è un linguaggio compilato ma utilizza un codice intermedio, quindi per poter fare funzionare programmi è necessario avere la macchina virtuale installata con la possibilità di cambiare da piattaforma a piattaforma.
La macchina virtuale è il cosiddetto JRE (acronimo che sta per Java Runtime Enviroment) ma per sviluppare programmi c'è bisogno del sistema di sviluppo chiamato JDK (Java developement Kit). Oggi la Oracle fornisce un pacchetto unico comprendente JRE, JDK ed il sistema di sviluppo Eclipse che si possono scaricare ed installare insieme in un sol colpo. Quindi si scarica il file, lo si apre ed il gioco è fatto.

Adesso, per poter utilizzare le porte seriali è necessario installare il pacchetto RxTx.

Inzialmente la Sun (oggi è tutto in mano alla Oracle) forniva un pacchetto chiamato javaxcomm. Gli utenti preferivano invece il pacchetto RxTx (sviluppato altrove sotto licenza GNU) fino al punto che Oracle non supporta più javaxcomm perchè non c' era abbastanza interesse da parte dell'utenza e quindi è finito nel dimenticatoio.

RxTx (rxtx-2.1-7-bins-r2.zip) è disponibile a questo link.

Una volta scaricato il pacchetto ed aperto, all' interno troviamo la cartella rxtx-2.1-7-bins-r2. Spostiamo quindi la cartella sul desktop e facciamo le seguenti operazioni supponendo che Java sia installato (come fa in automatico il programma d'installazione descritto prima) in c:\Programmi\Java\

i. Copiare rxtxSerial.dll

che si trova nella cartella rxtx-2.1-7-bins-r2\windows\i368-mingw32\

in c:\Programmi\Java\jre1.6.0_23\bin\

ii. Copiare RXTXcomm.jar

che si trova nella cartella rxtx-2.1-7-bins-r2\

in c:\Programmi\Java\jre1.6.0_23\lib\ext\

iii. Copiare rxtxSerial.dll

che si trova nella cartella rxtx-2.1-7-bins-r2\windows\i368-mingw32\

in c:\Programmi\Java\jdk1.6.0_23\jre\bin\

iv. Copiare RXTXcomm.jar

che si trova nella cartella rxtx-2.1-7-bins-r2\

in c:\Program Files\Java\jdk1.6.0_23\jre\lib\ext\

4. Requisiti di sviluppo

Nel caso specifico della progettazione della nostra GUI si devono avere i seguenti requisiti, per il corretto funzionamento del software:

1) Metodi Java:
    - portList: acquisisce la lista delle porte COM disponibili;
    - portInit: connessione alla porta COM selezionata;
    - outputWrite dell'OutputStream: per l'invio dei caratteri;
    - serialEvent: gestore dell'evento per la ricezione dei caratteri;
    - portClose: per chiudere la porta COM.

2) Parametri di configurazione della porta seriale del termostato e del PC:

CONFIGURAZIONE PORTA SERIALE:

Porta:	COM2
Baud Rate:	9600
Data Bits:	8
Stop Bits:	1
Parity:	None
Flow Controll:	Xon/Xoff

3) Comandi per l'invio dei caratteri dal software al termostato
    - OUT_SP_00_XXX.XX
      Setpoint transfer with up to 3 places before the decimal point and up to 2 places behind;
    - OUT_SP_01_XXX
      Pump output step 1, 2, 3, 4, or 5; 0 = Stop;
    - OUT_PAR_00_XXX.XX
      Setting of control parameter Xp for controller (0.1 – 10°C);
    - OUT_PAR_01_XXX
      Setting of control parameter Tn (5 - 60 sec);
    - OUT_MODE_00_X
      Keys: 0 = free / 1 = inhibited (corresponds to "KEY");
    - START

Switches thermostat on (from standby);

- STOP

  Switches thermostat to standby (pump, heating, refrigerator off).

I formati dei dati permessi sono:

-XXX.XX	-XXX.X	-XXX.	-XXX	XXX.XX	XXX.X	XXX.	XXX
-XX.XX	-XX.X	-XX.	-XX	XX.XX	XX.X	XX.	XX
-X.XX	-X.X	-X.	-X	X.XX	X.X	X.	X
-.XX	-.X	.XX	.X				

4) Comandi per la ricezione dei caratteri da termostato al software

- IN_PV_00

  Read bath temperature (outflow temperature);

- IN_PV_01

  Read external temperature TE;

- IN_SP_00

  Read temperature setpoint;

- IN_SP_01

  Read pump output step;

- IN_SP_03

  Read current overtemperature switch-off point;

- IN_SP_04

  Read current outflow temperature limit TiH;

- IN_SP_05

  Read current outflow temperature limit TiL;

- IN_PAR_00

  Read current value of Xp;

- IN_PAR_01

  Read current value of Tn (201 = OFF);

- IN_PAR_04

  Read current value of KPE;

- IN_PAR_05

  Read current value of TNE (201 = OFF);

- IN_MODE_00

Keys: 0 = free / 1 = inhibited;

- IN_MODE_01

  Control: 0 = internal / 1 = external;
- TYPE

  Read thermostat type;
- VERSION

  Read software version number;
- STATUS

  Read thermostat status 0 = OK, -1 = error;
- STAT

  Read error diagnosis response.

5) Messaggi di Errore del termostato

- ERR_2

  Wrong input (e.g. buffer overflow)
- ERR_3

  Wrong command
- ERR_5

  Syntax error in value
- ERR_6

  Illegal value
- ERR_8

  Channel (ext. temperature) not available
- ERR_30

  Programmer, all segments occupied

5. Sviluppo GUI (Graphical User Interface) con Java

Progettare una GUI è un'attività abbastanza complessa perché il progettista deve:

- Conoscere il bisogno degli utenti per il suo uso;
- Prevenire gli errori degli utenti e agire quando possibile;
- Snellire il più possibile l'accesso ai comandi per l'interazione con il sistema;
- Facile ed intuitivo per l'uso.

Per ottenere quanto sopra descritto si devono installare due plugin su eclipse

1) AWT – Abstract Window Toolkit

   La Abstract Window Toolkit (AWT) è la libreria Java contenente le classi e le interfacce fondamentali per il rendering grafico. Queste classi consentono di realizzare interfacce utente complesse e di definire l'interazione attraverso la specifica di elementi e di gestori degli stessi.

2) SWING

   Swing è un framework per Java, appartenente alle Java Foundation Classes (JFC) e orientato allo sviluppo di interfacce grafiche. Parte delle classi del framework Swing sono implementazioni di widget (oggetti grafici) come caselle di testo, pulsanti, pannelli e tabelle. La libreria Swing viene utilizzata come libreria ufficiale per la realizzazione di interfacce grafiche in Java. È un'estensione del precedente Abstract Window Toolkit.

Un framework orientato agli oggetti facilita lo sviluppo tramite due diversi tipi di riuso:

- Il riuso black-box di componenti già pronti. Classi Swing pronte da usare (es. JButton)
- Il riuso white-box di classi semi-lavorate, da completare. Interfacce o classi astratte Swing da specializzare (Action)

Le caratteristiche principali di una GUI sono:

1. Struttura (suddividere lo spazio con criterio)
2. Reattività (reagire a eventi con azioni)
3. Visualizzazione (disegnare con aspetto piacevole)

Un programma con una GUI deve:

1. Costruire tutti i vari oggetti Java che rappresentano i componenti grafici
2. Comporli in una struttura
    - Relazioni di contenimento
    - Gestione della disposizione (layout management)

Una GUI è composta da almeno tre tipi di oggetti
- Componenti, come bottoni o menù, che vengono visualizzati
- Eventi, che reificano le azioni dell'utente
- Listener, che rispondono agli eventi sui componenti

Una GUI è infatti un sistema reattivo, significa che intraprende una azione solo quando riceve uno stimolo esterno (evento).
Quello che bisogna fare, una volta costruita la struttura grafica, è specificare quale oggetto Java è il gestore di ciascun diverso stimolo e quindi in gergo Java, bisogna installare i listener.

Nel capitolo successivo si trova il codice sorgente, sviluppato secondo i requisiti del capitolo quattro e le caratteristiche principali che abbiamo descritto in questo capitolo usando le due librerie swing e awt.

Mostro in figura sotto, la GUI sviluppata per il funzionamento del termostato:

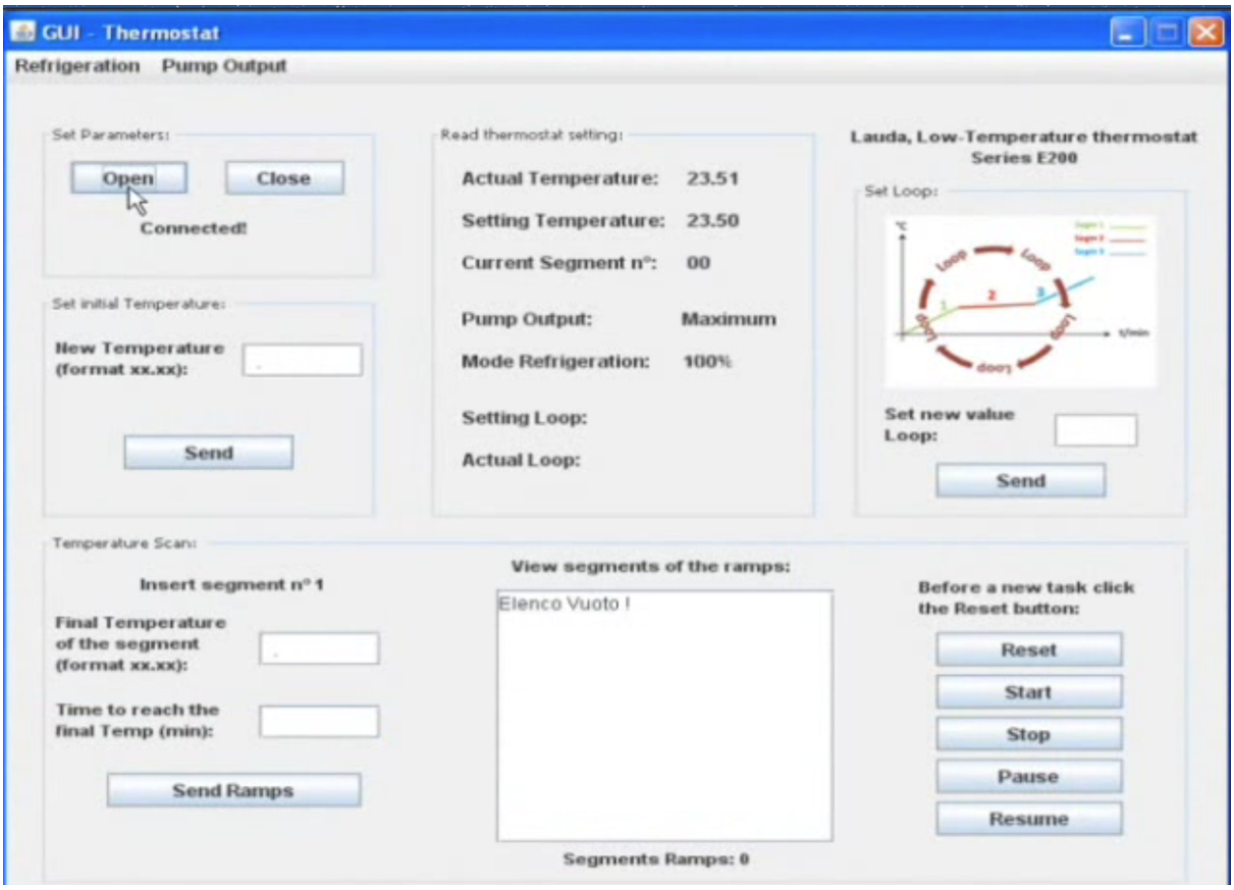


Figura: GUI

Nel pannello “Set Parameters” troviamo due pulsanti:

- Open: si apre il collegamento al termostato secondo i parametri impostati nella configurazione seriale (Porta COM, baud-rate, ecc);
- Close: disconnessione.

Pannello “Set Initial Temperature”:

- Si imposta il valore di temperatura che si desidera e il termostato provvederà ad ottenere la temperatura impostata dall’utente.

Pannello “Temperature Scan”:

- Si inseriscono le rampe di temperatura, chiamiamoli segmenti, impostando il valore della temperatura e del tempo che si desidera. Possono essere inseriti più segmenti in tempi diversi in modo da ottenere le varie temperature desiderate.
- Tali segmenti si potranno visualizzare nell’area: “View segments of the ramps”

Pannello “Set Loop”:

- In questo pannello si impostano i cicli delle operazioni che si vogliono ottenere dai segmenti impostati.
- Ad es. imposto tre segmenti del tipo:
    - Temperatura 25°C in due minuti;
    - Temperatura 26°C in un minuto;
    - Temperatura 30°C in cinque minuti;

  impostando il valore di Loop: “4”, il sistema ripete quattro volte le stesse operazioni dell’esempio di prima.

Pannello “Read Thermostat setting”:

- Actual Temperature: visualizziamo il valore della temperatura attuale del termostato;
- Setting temperature: visualizziamo il valore della temperatura impostata dall’utente nel pannello set initial temperature o nel pannello temperature scan.
- Current Segment: segna il numero del segmento corrente dell’operazione che sta effettuando del pannello view segment of the ramps;
- Pump output e Mode Refrigeration: valori impostati dall’utente dal menù in alto a sinistra della GUI;
- Setting Loop: visualizza quanti cicli ha impostato l’utente in modo da ripetere le stesse operazioni impostate in temperature scan;
- Actual Loop: mostra a quale numero di loop si trova e che sta eseguendo.

Menù “Refrigeration” contiene tre valori selezionabili dall’utente:

- Refrigeration OFF
- Refrigeration 100%
- Refrigeration Automatic

Menù “Pump Output” contiene tre valori selezionabili dall’utente:

- Low Pump Output
- Medium Pump Output
- Maximum Pump Output

Ci sono anche i pulsanti di:

- RESET, per cancellare i valori inseriti nei vari campi;
- START, per avviare lo scambio dati con il termostato;
- STOP, per fermare il termostato;
- PAUSE, mette in pausa il termostato;
- RESUME, riprende le operazioni precedentemente impostate.

6. Codice Sorgente

```
//Codice Sorgente GUI Termostato
import java.io.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.text.MaskFormatter;
import gnu.io.*;
import java.text.DateFormat;
import java.util.*;
import java.util.Timer;

@SuppressWarnings({ "serial", "unused" })
public class therm extends JFrame {

 gestioneporta g;
 JButton connetti;
 JButton disconnetti;
 JButton outflow_temperature;
 JButton read_temperature;
 JButton start;
 JButton stop;
 JButton reset;
 JButton pausa;
 JButton continua;
 JButton segmentocorrenterampa;
 JButton inviorampa;
 JButton invioloop;
 JButton Clear;
 JButton invionewtemp;
 JButton leggifile;
 JScrollPane labelsegmenti;
 JTextArea leggitesto;

 JLabel Label1;
 JLabel Label3;
 JLabel Label4;
 JLabel Label5;
 JLabel Label6;
 JLabel Label7;
 JLabel Label8;
 JLabel Label9;
 JLabel Label10;
 JLabel Lactualtemperature;
 JLabel Lsettingtemperature;
 JLabel Lmostravaloreactualtemp;
 JLabel Lmostravaloresettingtemp;
 JLabel Lcurrentsegment;
 JLabel Lmostravalorecurrentsegm;
 JLabel Lscrittanewtemperature;
 JLabel LPumpOutput;
 JLabel Lmostravalorepumpout;
 JLabel LRefrigeration;
 JLabel Lmostravalorerefrigeration;
 JLabel loopsetting;
 JLabel loopsettingmostra;
 JLabel loopactual;
 JLabel loopactualmostra;
 JLabel scrittaloop;
```

```
 JLabel sfondo;
 ImageIcon immagine;


 JMenuBar jMenuBar1;
 JMenu jmFile;
 JMenuItem jmiExit;
 JMenu jmRefrigeration;
 JMenuItem jmiRefrigeration0; // OFF
 JMenuItem jmiRefrigeration1; //ON  100 %
 JMenuItem jmiRefrigeration2; // 50 %
 JMenuItem jmiRefrigeration3; // automatic
 JMenu jmPumpOutput;
 JMenuItem jmiPumpOutput0; // pump Stopped
 JMenuItem jmiPumpOutput1; // low pump output
 JMenuItem jmiPumpOutput3; // medium pump output
 JMenuItem jmiPumpOutput5; // maximum pump output


 JPanel Panel1;   // Read thermostats setting
 JPanel Panel2;   // Set initial Temperature
 JPanel Panel3;   // Set Parameters
 JPanel Panel4;   // Temeperature Scan
 JPanel Panel5;  //  Set LOOP

 JFormattedTextField write_temperature;
 JFormattedTextField impostarampanew;
 //JFormattedTextField impostarampanew;
 JTextField impostarampatempo;
 JTextField impostaloop;
 //A convenience method for creating a MaskFormatter.
 protected MaskFormatter createFormatter(String s) {
     MaskFormatter formatter = null;
     try {
         formatter = new MaskFormatter(s);
     } catch (java.text.ParseException exc) {
         System.err.println("formatter is bad: " + exc.getMessage());
         System.exit(-1);
     }
     return formatter;
 }

 public therm()
 {
        termostatoGUILayout customLayout = new termostatoGUILayout();

     getContentPane().setFont(new Font("Helvetica", Font.PLAIN, 12));
     getContentPane().setLayout(customLayout);

     Label4 = new JLabel(); //Mostra Testo Tipologia Termostato Lauda
     getContentPane().add(Label4);
     Label4.setFont(new Font("sansserif", Font.BOLD, 11));

     Panel3 = new JPanel();
     Panel3.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Set
Parameters:", javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font("Tahoma",
0, 10)));
     Panel3.setFont(new Font("sansserif", Font.BOLD, 11));
     connetti = new JButton("Open");
     getContentPane().add(connetti);
```

```
      disconnetti = new JButton("Close");
      getContentPane().add(disconnetti);

      Label3 = new JLabel(); //Mostra Se On / OFF
      getContentPane().add(Label3);
      Label3.setFont(new Font("sansserif", Font.BOLD, 11));
      getContentPane().add(Panel3);

      outflow_temperature = new JButton("outflow temperature");
      getContentPane().add(outflow_temperature);

      read_temperature = new JButton("read temperature");
      getContentPane().add(read_temperature);

      start = new JButton("Start");
      getContentPane().add(start);

      stop = new JButton("Stop");
      getContentPane().add(stop);

      pausa = new JButton("Pause");
      getContentPane().add(pausa);

      continua = new JButton("Resume");
      getContentPane().add(continua);

      segmentocorrenterampa = new JButton("Current Segment");
      getContentPane().add(segmentocorrenterampa);

       Panel1 = new JPanel();
       Panel1.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Read
thermostat setting:", javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font("Tahoma",
0, 10)));
       Panel1.setFont(new Font("sansserif", Font.BOLD, 11));

      Lactualtemperature = new JLabel(); //Mostra scritta actual temperature
(outflow)
      getContentPane().add(Lactualtemperature);
      Lmostravaloreactualtemp = new JLabel(); //Mostra valore actual temperature
(outflow)
      getContentPane().add(Lmostravaloreactualtemp);
      Lmostravaloreactualtemp.setFont(new Font("sansserif", Font.BOLD, 12));
      Lsettingtemperature = new JLabel(); //Mostra scritta Setting temperature
(read)
      getContentPane().add(Lsettingtemperature);
      Lmostravaloresettingtemp = new JLabel(); //Mostra valore setting
temperature (read)
      getContentPane().add(Lmostravaloresettingtemp);
      Lcurrentsegment = new JLabel(); //Mostra scritta current segment
      getContentPane().add(Lcurrentsegment);
      Lmostravalorecurrentsegm = new JLabel(); //Mostra valore current segment
      getContentPane().add(Lmostravalorecurrentsegm);

      LPumpOutput = new JLabel(); //Mostra scritta Pump Output step
      getContentPane().add(LPumpOutput);
      Lmostravalorepumpout = new JLabel(); //Mostra valore della Pump Output
      getContentPane().add(Lmostravalorepumpout);
      LRefrigeration = new JLabel(); //Mostra scritta  mode Refrigeration
      getContentPane().add(LRefrigeration);
      Lmostravalorerefrigeration = new JLabel(); //Mostra valore della mode
Refrigeration
```

```
     getContentPane().add(Lmostravalorerefrigeration);
     loopsetting = new JLabel(); //Mostra scritta  Loop Setting
     getContentPane().add(loopsetting);
     loopsettingmostra = new JLabel(); //Mostra valore del loop setting
     getContentPane().add(loopsettingmostra);
     loopactual = new JLabel(); //Mostra scritta  loop actual
     getContentPane().add(loopactual);
     loopactualmostra = new JLabel(); //Mostra valore loop actual
     getContentPane().add(loopactualmostra);


     getContentPane().add(Panel1);

     Panel2 = new JPanel();
    // getContentPane().add(Panel2);
     Panel2.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Set
initial Temperature: ", javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font("Tahoma",
0, 10)));
   //  write_temperature = new JFormattedTextField(" "); //TextField
inserimento nuovo valore di lettura

     Lscrittanewtemperature = new JLabel(); //Mostra Testo temperatura rampa
     getContentPane().add(Lscrittanewtemperature);
     Lscrittanewtemperature.setFont(new Font("sansserif", Font.BOLD, 11));

     write_temperature = new JFormattedTextField(
             createFormatter("##.##"));

     getContentPane().add(write_temperature);

     invionewtemp = new JButton("Send");
     getContentPane().add(invionewtemp);

     getContentPane().add(Panel2);

     Panel4 = new JPanel();
     Panel4.setBorder(javax.swing.BorderFactory.createTitledBorder(null,
"Temperature Scan: ", javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font("Tahoma",
0, 10)));

     Label5 = new JLabel(); //Mostra Testo temperatura rampa
     getContentPane().add(Label5);
     Label5.setFont(new Font("sansserif", Font.BOLD, 11));

     impostarampanew = new JFormattedTextField(
             createFormatter("##.##"));
     getContentPane().add(impostarampanew);

     Label6 = new JLabel(); //Mostra Testo Tempo rampa
     getContentPane().add(Label6);
     Label6.setFont(new Font("sansserif", Font.BOLD, 11));

 //    impostarampatempo = new JFormattedTextField(
 //    createFormatter("###"));
 //    getContentPane().add(impostarampatempo);

     impostarampatempo = new JTextField();
     getContentPane().add(impostarampatempo);

     inviorampa = new JButton("Send Ramps");
```

```
     getContentPane().add(inviorampa);

     Label7 = new JLabel(); //Mostra Elenco segmenti da File della rampa,
temperatura tempo
     getContentPane().add(Label7);
     Label7.setFont(new Font("sansserif", Font.BOLD, 11));
     getContentPane().add(Label7);

     leggitesto = new JTextArea("");
     leggitesto.setEditable (false);
     labelsegmenti = new JScrollPane(leggitesto);
     getContentPane().add(labelsegmenti);

     Label8 = new JLabel(); //Mostra numero segmenti: count
     Label8.setFont(new Font("sansserif", Font.BOLD, 11));
     getContentPane().add(Label8);

     leggifile = new JButton("View Ramps");
     getContentPane().add(leggifile);

     Label10 = new JLabel(); //Mostra Elenco segmenti da File della rampa,
temperatura tempo
     Label10.setFont(new Font("sansserif", Font.BOLD, 11));
     getContentPane().add(Label10);

     reset = new JButton("Reset");
     getContentPane().add(reset);

     Label9 = new JLabel(); //Mostra scritta inserimento segmento, count +1
     Label9.setFont(new Font("sansserif", Font.BOLD, 11));
     getContentPane().add(Label9);

     getContentPane().add(Panel4);

      Clear = new JButton("Clear All");
      getContentPane().add(Clear);

     Label1 = new JLabel(); //Mostra Lettura da pulsanti e TextField
     getContentPane().add(Label1);

     Panel5 = new JPanel();
     Panel5.setBorder(javax.swing.BorderFactory.createTitledBorder(null, "Set
Loop: ", javax.swing.border.TitledBorder.DEFAULT_JUSTIFICATION,
javax.swing.border.TitledBorder.DEFAULT_POSITION, new java.awt.Font("Tahoma",
0, 10)));

     scrittaloop = new JLabel(); //Mostra Testo inserimento nuovo loop
     getContentPane().add(scrittaloop);
     scrittaloop.setFont(new Font("sansserif", Font.BOLD, 11));

     impostaloop = new JTextField();
     getContentPane().add(impostaloop);

     invioloop = new JButton("Send");
     getContentPane().add(invioloop);

     immagine = new ImageIcon("logo3.png");
     sfondo = new JLabel(immagine);
     getContentPane().add(sfondo);

     getContentPane().add(Panel5);
```

```
// Creo la barra dei men
  jMenuBar1 = new JMenuBar();

  jmRefrigeration = new JMenu("Refrigeration");
  jmiRefrigeration0 = new JMenuItem("Refrigeration OFF");
  jmiRefrigeration0.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
            jbrefrigeration0ActionPerformed(evt);
      }
  });
  jmRefrigeration.add(jmiRefrigeration0);
  jmiRefrigeration1 = new JMenuItem("Refrigeration 100%");
  jmiRefrigeration1.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
            jbrefrigeration1ActionPerformed(evt);
      }
  });
  jmRefrigeration.add(jmiRefrigeration1);
  jmiRefrigeration3 = new JMenuItem("Refrigeration Automatic");
  jmiRefrigeration3.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
            jbrefrigeration3ActionPerformed(evt);
      }
  });
  jmRefrigeration.add(jmiRefrigeration3);

  jmPumpOutput = new JMenu("Pump Output");
  jmiPumpOutput1 = new JMenuItem("Low Pump Output");
  jmiPumpOutput1.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
            jbpumpoutput1ActionPerformed(evt);
      }
  });
  jmPumpOutput.add(jmiPumpOutput1);
  jmiPumpOutput3 = new JMenuItem("Medium Pump Output");
  jmiPumpOutput3.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
            jbpumpoutput3ActionPerformed(evt);
      }
  });
  jmPumpOutput.add(jmiPumpOutput3);
  jmiPumpOutput5 = new JMenuItem("Maximum Pump Output");
  jmiPumpOutput5.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(java.awt.event.ActionEvent evt) {
            jbpumpoutput5ActionPerformed(evt);
      }
  });
  jmPumpOutput.add(jmiPumpOutput5);



  jMenuBar1.add(jmRefrigeration);
  jMenuBar1.add(jmPumpOutput);
  setJMenuBar(jMenuBar1);

  pack();

  setSize(getPreferredSize());

  connetti.addActionListener(new connetti());
  disconnetti.addActionListener(new disconnetti());
  outflow_temperature.addActionListener(new miolistener());
```

```
     read_temperature.addActionListener(new miolistener2());
     start.addActionListener(new miolistener3());
     stop.addActionListener(new miolistener4());
     reset.addActionListener(new miolistener5());
     pausa.addActionListener(new miolistener6());
     continua.addActionListener(new miolistener7());
     segmentocorrenterampa.addActionListener(new miolistener8());
     write_temperature.addKeyListener(new java.awt.event.KeyAdapter() {
     public void keyReleased(java.awt.event.KeyEvent e) {
             jtaSentKeyReleased(e);
         }
     });
    inviorampa.addActionListener(new miolistener9());
    Clear.addActionListener(new miolistener10());
    leggifile.addActionListener(new miolistener11());
    invionewtemp.addActionListener(new miolistener12());
    invioloop.addActionListener(new miolistener13());


    addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent e) {
             System.exit(0);
         }
     });

 }

public static void main(String args[]) {
     javax.swing.SwingUtilities.invokeLater(new Runnable() {
         public void run() {
               JFrame frame = new JFrame("");
               JOptionPane.showMessageDialog(frame ,  "Before carrying out any
operation, make sure that the thermostat is turned on.");

             //Turn off metal's use of bold fonts
                therm window = new therm();
               window.setTitle("GUI - Thermostat");
               window.setResizable(false);  // Per ingrandire o no la finestra
               window.pack();
               window.show();
         }
     });
   // End Refresh
 }

private void jtaSentKeyReleased(java.awt.event.KeyEvent e) {
         System.out.println("\n " + e.toString());
     if(e.getKeyCode() == KeyEvent.VK_ENTER) {
           System.out.println("\n da Text Area =
"+write_temperature.getText());
              // Scrivo temperatura da me
              String a = "OUT_SP_00_";
              String b ="\r\n";
          g.scrivi(a+write_temperature.getText()+b);
                      //g.scrivi("OUT_SP_00_25\r\n");
          write_temperature.setText("");

      }
  }
private void jbinviorampaActionPerformed(java.awt.event.ActionEvent e){
 try
{
```

```
     FileOutputStream temperatura_tempo = new
FileOutputStream("temperatura_tempo.txt", true);
     PrintStream scrivi = new PrintStream(temperatura_tempo);
     String a = "RMP_OUT_00_";
     String c = "_";
     String b ="\r\n";
     String tempo = impostarampatempo.getText().trim();
  //   if (tempo.length() <4){
     if (Integer.parseInt(tempo) < 255){
        g.scrivi(a+impostarampanew.getText()+c+impostarampatempo.getText()+b);

         //Scrivo sul file temperatura_tempo.txt su: "C:\Documents and
Settings\Utente\workspace\Software_Sviluppati"
         scrivi.println("Temperature: "+impostarampanew.getText()+ " - Time:
"+impostarampatempo.getText());
         impostarampanew.setText("");
         impostarampatempo.setText("");
         leggitesto.setText("");
         write_temperature.setText("");
         impostaloop.setText("");
         Thread.sleep(500);
         jbleggifileActionPerformed(e);
     }
      else {
        //System.out.println("Errore, non puoi inserire un num > di 999");
        impostarampatempo.setText("");
        write_temperature.setText("");
         impostaloop.setText("");
        JFrame frame = new JFrame("");
                        //JOptionPane.showMessageDialog(frame ,  "Error: The
Maximum number of minutes allowed is 999");
        JOptionPane.showMessageDialog(frame ,  "Error: If a segment time longer
than 255 min is required, this time must be distributed over several
consecutive segments.");
     }
     }
        catch (InterruptedException e1) {
                e1.printStackTrace();
        }
 catch (IOException e1)
 {
     System.out.println("Errore: " + e1);
     System.exit(1);
 }
}

private void jbinvionewtempActionPerformed(java.awt.event.ActionEvent e){
   // Scrivo temperatura da me
   String a = "OUT_SP_00_";
   String b ="\r\n";
          g.scrivi(a+write_temperature.getText()+b);
        write_temperature.setText("");
}

private void jbinvioloopActionPerformed(java.awt.event.ActionEvent e){

             String a = "RMP_OUT_02_";
             String b ="\r\n";
             String loop = impostaloop.getText().trim();

             if (loop.length() <4){
                g.scrivi(a+impostaloop.getText()+b);
```

```
                impostaloop.setText("");
                write_temperature.setText("");

            }
             else {
               //System.out.println("Errore, non puoi inserire un num > di
999");
                impostaloop.setText("");
                write_temperature.setText("");
              JFrame frame = new JFrame("");
                            JOptionPane.showMessageDialog(frame ,  "Error:
The Maximum number of loop allowed is 250");
            }

        }

private void jbrefrigeration0ActionPerformed(java.awt.event.ActionEvent e){
 // Mode Refrigerator 0 = OFF
 String a = "OUT_SP_02_000\r\n";
         g.scrivi(a);
}

private void jbrefrigeration1ActionPerformed(java.awt.event.ActionEvent e){
 // Mode Refrigerator 1 = ON, 100 %
 String a = "OUT_SP_02_001\r\n";
         g.scrivi(a);
}

private void jbrefrigeration3ActionPerformed(java.awt.event.ActionEvent e){
 // Mode Refrigerator 3 = automatic
 String a = "OUT_SP_02_003\r\n";
         g.scrivi(a);
}

private void jbpumpoutput1ActionPerformed(java.awt.event.ActionEvent e){
 // Pump output 1 = Low pump output
 String a = "OUT_SP_01_001\r\n";
         g.scrivi(a);
}
private void jbpumpoutput3ActionPerformed(java.awt.event.ActionEvent e){
 // Pump output 3 = medium pump output
 String a = "OUT_SP_01_003\r\n";
         g.scrivi(a);
}
private void jbpumpoutput5ActionPerformed(java.awt.event.ActionEvent e){
 // Pump output 5 = maximum pump output
 String a = "OUT_SP_01_005\r\n";
         g.scrivi(a);
}
private void jbleggifileActionPerformed(java.awt.event.ActionEvent e){
try
{
                File f;
                FileInputStream fs;
                InputStreamReader isr;
                BufferedReader fileInput;
                String line;

                f = new File("temperatura_tempo.txt");
                fs = new FileInputStream(f);
                isr = new InputStreamReader(fs);
                fileInput = new BufferedReader(isr);
```

```
                  if (f.exists()) {
                          leggitesto.setText("");
                         impostarampanew.setText("");
                         impostarampatempo.setText("");
                          FileReader fr = new FileReader(f);
                          LineNumberReader ln = new LineNumberReader(fr);
                         int count = 0;
                          while (ln.readLine() != null) {
                          count++;
                         }
                        //  System.out.println("Righe Totali: " + count);

                           if (count != 0)
                           {
                                  for(int i=0;i<count;i++)
                                  {
                                          line = fileInput.readLine();
                                          System.out.println(line);
                                          leggitesto.append(line + "\n"); // Mi
stampa sulla TexArea il contenuto del file
                                          Label8.setText("Segments Ramps: "
+count);
                                          Label9.setText("<html>Insert segment n
"+(count+1)+"<br>         (max
20)</html>");
                                  }
                           }
                           else {
                                       leggitesto.setText("Empty List !");
                                       Label8.setText("Segments Ramps: "
+count);
                                       Label9.setText("Insert segment n
"+(count+1));
                           }

                         ln.close();
                          }else {
                          System.out.println("Il file non esiste.");
                          }

}
catch (IOException e1)
{
  System.out.println("Errore: " + e1);
  System.exit(1);
}
}
public Timer timer;
public class connetti implements ActionListener {
        public void actionPerformed(ActionEvent e) {
                g=new gestioneporta();
                impostarampatempo.setText("");
                impostarampanew.setText("");
          write_temperature.setText("");
          impostaloop.setText("");
          jbleggifileActionPerformed(e);
                 timer=new Timer();
                 timer.schedule(new Task(),0,5000);//parti subito e itera ogni 5
secondi
     }
class Task extends TimerTask{
```

```
        public void run(){
                try {
                        //Mostro Outflow Temperature
                g.scrivi("IN_PV_00\r\n");
                Thread.sleep(500);
                //Mostro Segmento corrente della rampa
                g.scrivi("IN_SP_00\r\n");
                Thread.sleep(500);
                //Mostro Segmento corrente della rampa
                g.scrivi("RMP_IN_01\r\n");
                Thread.sleep(500);
                //Mostro pump output
                 g.scrivi("IN_SP_01\r\n");
                Thread.sleep(500);
                //Mostro mode refrigeration
                g.scrivi("IN_SP_02\r\n");
                Thread.sleep(500);
                //Mostro Setting Loop
                g.scrivi("RMP_IN_02\r\n");
                Thread.sleep(500);
                //Mostro Actual Loop - corrente
                g.scrivi("RMP_IN_03\r\n");
                }
     catch (InterruptedException e1) {
                e1.printStackTrace();
        }
        }
}
}

public class miolistener implements ActionListener {

        public void actionPerformed(ActionEvent e) {
                // outflow_temperature
                g.scrivi("IN_PV_00\r\n");
                impostarampatempo.setText("");
                impostarampanew.setText("");
     write_temperature.setText("");
 }
}

public class miolistener2 implements ActionListener {

        public void actionPerformed(ActionEvent e) {
                // read_temperature set point
                g.scrivi("IN_SP_00\r\n");
                impostarampatempo.setText("");
                impostarampanew.setText("");
     write_temperature.setText("");
 }
}
public class miolistener3 implements ActionListener {

        public void actionPerformed(ActionEvent e) {
                // start rampa
                g.scrivi("RMP_START\r\n");
                impostarampatempo.setText("");
                impostarampanew.setText("");
     write_temperature.setText("");
 }
}
public class miolistener4 implements ActionListener {
```

```
        public void actionPerformed(ActionEvent e) {
                // stop rampa
                g.scrivi("RMP_STOP\r\n");
                impostarampatempo.setText("");
                impostarampanew.setText("");
     write_temperature.setText("");
 }
}

public class miolistener5 implements ActionListener {

        public void actionPerformed(ActionEvent e) {
                // reset rampa
                try {
                        g.scrivi("RMP_RESET\r\n");
                        File temp = new File("temperatura_tempo.txt");
                        BufferedReader in = new BufferedReader(new
FileReader(temp));
                        PrintWriter out = new PrintWriter(new BufferedWriter(new
FileWriter(temp)));
                        impostarampatempo.setText("");
                        impostarampanew.setText("");
                 write_temperature.setText("");
                 leggitesto.setText("");
                        in.close();
                        out.close();
                        Thread.sleep(500);
                        temp.delete();
                        Thread.sleep(500);
                        temp.createNewFile();
                        Thread.sleep(500);
                        jbleggifileActionPerformed(e);
                } catch (IOException e1) {
                        e1.printStackTrace();
                 } catch (InterruptedException e1) {
                        e1.printStackTrace();
                }
 }
}
public class miolistener6 implements ActionListener {

        public void actionPerformed(ActionEvent e) {
                // PAUSA Rampa
                g.scrivi("RMP_PAUSE\r\n");
                impostarampatempo.setText("");
                impostarampanew.setText("");
     write_temperature.setText("");
 }
}
public class miolistener7 implements ActionListener {

        public void actionPerformed(ActionEvent e) {
                // CONTINUA RAMPA
                g.scrivi("RMP_CONT\r\n");
                impostarampatempo.setText("");
                impostarampanew.setText("");
     write_temperature.setText("");
 }
}

public class miolistener8 implements ActionListener {
```

```
        public void actionPerformed(ActionEvent e) {
                // Leggi Segmento corrente della rampa
                g.scrivi("RMP_IN_01\r\n");
                impostarampatempo.setText("");
                impostarampanew.setText("");
     write_temperature.setText("");
 }
}


public class miolistener9 implements ActionListener {

        public void actionPerformed(ActionEvent e) {

         jbinviorampaActionPerformed(e);
 }
}

public class miolistener10 implements ActionListener {

        public void actionPerformed(ActionEvent e) {

                impostarampatempo.setText("");
                //impostarampatempo.grabFocus();
                impostarampanew.setText("");
                //impostarampanew.grabFocus();
     write_temperature.setText("");
    // write_temperature.grabFocus();
     leggitesto.setText("");
    // Label1.setText("");
 }
}
public class miolistener11 implements ActionListener {

        public void actionPerformed(ActionEvent e) {

         jbleggifileActionPerformed(e);
 }
}

public class miolistener12 implements ActionListener {

        public void actionPerformed(ActionEvent e) {

         jbinvionewtempActionPerformed(e);
 }
}
public class miolistener13 implements ActionListener {

        public void actionPerformed(ActionEvent e) {

         jbinvioloopActionPerformed(e);
 }
}


public class disconnetti implements ActionListener {

        public void actionPerformed (ActionEvent e) {

                 g.disconnetti();
```

```
    }
}

class termostatoGUILayout implements LayoutManager {

 public termostatoGUILayout() {
 }

 public void addLayoutComponent(String name, Component comp) {
 }

 public void removeLayoutComponent(Component comp) {
 }

 public Dimension preferredLayoutSize(Container parent) {
     Dimension dim = new Dimension(0, 0);

     Insets insets = parent.getInsets();
     dim.width = 720 + insets.left + insets.right;
     dim.height = 600 + insets.top + insets.bottom;

     return dim;
 }

 public Dimension minimumLayoutSize(Container parent) {
     Dimension dim = new Dimension(0, 0);
     return dim;
 }

 public void layoutContainer(Container parent) {
     Insets insets = parent.getInsets();

     Component c;
     //Label con scritta valore temperatura
     c = parent.getComponent(0);
     if (c.isVisible()) Label4.setText("<html>Lauda, Low-Temperature
thermostat<br><center>Series E200</center></html>.");

        {c.setBounds(insets.left+500,insets.top+20,300,50);}

     // Pulsante connetti
     c = parent.getComponent(1);
     if (c.isVisible()) {c.setBounds(insets.left+38,insets.top+55,70,24);}

     //Pulsante disconnetti
     c = parent.getComponent(2);
     if (c.isVisible()) {c.setBounds(insets.left+130,insets.top+55,70,24);}

     //Label Visualizzazione scritta ON / OFF
     c = parent.getComponent(3);
     if (c.isVisible()) {c.setBounds(insets.left+80,insets.top+90,70,24);}

     //Panel3 con scritta Set Parameters, riquadro Panel3
     c = parent.getComponent(4);
     if (c.isVisible())
     {c.setBounds(insets.left+20,insets.top+30,200,110);}


     //Pulsante Overflow temperature
     c = parent.getComponent(5);
  //   if (c.isVisible()) {c.setBounds(insets.left+11,insets.top+150,200,24);}
```

```
     // Pulsante Read Temperature
     c = parent.getComponent(6);
  //  if (c.isVisible()) {c.setBounds(insets.left+11,insets.top+180,200,24);}

     // Pulsante start rampa
     c = parent.getComponent(7);
     if (c.isVisible()) {c.setBounds(insets.left+550,insets.top+420,110,24);}

     // Pulsante stop rampa
     c = parent.getComponent(8);
     if (c.isVisible()) {c.setBounds(insets.left+550,insets.top+450,110,24);}

     // Pulsante pausa rampa
     c = parent.getComponent(9);
     if (c.isVisible()) {c.setBounds(insets.left+550,insets.top+480,110,24);}

     // Pulsante continua/Riprendi rampa
     c = parent.getComponent(10);
     if (c.isVisible()) {c.setBounds(insets.left+550,insets.top+510,110,24);}


     // Pulsante leggi segmento corrente della rampa
     c = parent.getComponent(11);
 //    if (c.isVisible()) {c.setBounds(insets.left+11,insets.top+210,200,24);}

     //Label Lactualtemperature scritta Actual Temperature (outflow
temperature)
     c = parent.getComponent(12); Lactualtemperature.setText("Actual
Temperature:");
     if (c.isVisible()){c.setBounds(insets.left+270,insets.top+55,300,24);}

     //Label Lmostravaloreactualtemp mostra lettura della Actual Temperature
(outflow temperature)
     c = parent.getComponent(13);
     if (c.isVisible()){c.setBounds(insets.left+400,insets.top+55,300,24);}

     //Label Lsettingtemperature scritta setting Temperature (Read temperature)
     c = parent.getComponent(14); Lsettingtemperature.setText("Setting
Temperature:");
     if (c.isVisible()){c.setBounds(insets.left+270,insets.top+85,300,24);}

     //Label Lmostravaloresettingtemp mostra lettura della Setting Temperature
(Read temperature)
     c = parent.getComponent(15);
     if (c.isVisible()){c.setBounds(insets.left+400,insets.top+85,300,24);}

     //Label Lcurrenttemperature scritta Current Temperature (Current
temperature della rampa dei segmenti)
     c = parent.getComponent(16); Lcurrentsegment.setText("Current Segment
n:");
     if (c.isVisible()){c.setBounds(insets.left+270,insets.top+115,300,24);}

     //Label Lmostravalorecurrenttemp mostra lettura della Current Temperature
(Current temperature delle rampa dei segmenti)
     c = parent.getComponent(17);
     if (c.isVisible()){c.setBounds(insets.left+400,insets.top+115,300,24);}

     //Label LPumpOutupt scritta Pump Output
      c = parent.getComponent(18); LPumpOutput.setText("Pump Output:");
     if (c.isVisible()){c.setBounds(insets.left+270,insets.top+155,300,24);}
```

```
     //Label Lmostravalorepumpout mostra valore pump output
     c = parent.getComponent(19);
     if (c.isVisible()){c.setBounds(insets.left+400,insets.top+155,300,24);}

     //Label LRefrigeration scritta Mode Refrigeration
       c = parent.getComponent(20); LRefrigeration.setText("Mode
Refrigeration:");
     if (c.isVisible()){c.setBounds(insets.left+270,insets.top+185,300,24);}

     //Label Lmostravalorerefrigeration mostra valore refrigeration
     c = parent.getComponent(21);
     if (c.isVisible()){c.setBounds(insets.left+400,insets.top+185,300,24);}

     //Label loopsetting scritta Loop Setting
     c = parent.getComponent(22); loopsetting.setText("Setting Loop: ");
     if (c.isVisible()){c.setBounds(insets.left+270,insets.top+225,300,24);}

     //Label loopsettingmostra mostra valore Loop setting
     c = parent.getComponent(23);
     if (c.isVisible()){c.setBounds(insets.left+400,insets.top+225,300,24);}

     //Label loopactual scritta Loop Actual
     c = parent.getComponent(24); loopactual.setText("Actual Loop: ");
     if (c.isVisible()){c.setBounds(insets.left+270,insets.top+255,300,24);}

     //Label loopactualmostra mostra valore Loop actual
     c = parent.getComponent(25);
     if (c.isVisible()){c.setBounds(insets.left+400,insets.top+255,300,24);}


     //Panel1 con scritto READ thermostats setting nel riquadro del Panel1
     c = parent.getComponent(26);
     if (c.isVisible())
     {c.setBounds(insets.left+250,insets.top+30,230,280);}


     //Label Lscrittanewtemperature con scritta new temperature (format XX.XX)
     c = parent.getComponent(27);
     if (c.isVisible()) Lscrittanewtemperature.setText("<html>New
Temperature<br>(format xx.xx):</html>");

        {c.setBounds(insets.left+30,insets.top+180,130,30);}

     // campo vuoto di TextField per inserimento nuova temperatura
     c = parent.getComponent(28);
     if (c.isVisible())

        {c.setBounds(insets.left+140,insets.top+185,72,24);}

     // Pulsante invio new temperatura (set initial temperature)
     c = parent.getComponent(29);
    if (c.isVisible()){c.setBounds(insets.left+70,insets.top+250,100,24);}

     // Panel2 con testo" Set initial temperature, riquadro
     c = parent.getComponent(30);
     if (c.isVisible()){c.setBounds(insets.left+20,insets.top+150,200,160);}

///// Inizio Riquadro TEMPERATURE SCAN

     //Label con scritta valore temperatura della rampa
     c = parent.getComponent(31);
```

```
     if (c.isVisible()) Label5.setText("<html>Final Temperature<br>of the
segment<br>(format xx.xx):</html>");

        {c.setBounds(insets.left+30,insets.top+375,120,45);}

     // campo vuoto di TextField per inserimento segmento nuova temperatura
della rampa
     c = parent.getComponent(32);
     if (c.isVisible())

        {c.setBounds(insets.left+150,insets.top+390,72,24);}

     //Label con scritta valore tempo della rampa
     c = parent.getComponent(33);
     if (c.isVisible()) Label6.setText("<html>Time to reach the<br>final Temp
(min):</html>");

        {c.setBounds(insets.left+30,insets.top+435,120,35);}

     // campo vuoto di TextField per inserimento segmento del tempo (dopo la
temperatura) della rampa
     c = parent.getComponent(34);
     if (c.isVisible())

        {c.setBounds(insets.left+150,insets.top+442,72,24);}


     // Pulsante invio rampa dei parametri temperatura e tempo presi dalle
texfield
     c = parent.getComponent(35);
     if (c.isVisible()) {c.setBounds(insets.left+60,insets.top+490,150,24);}

     //Label di visualizzazione Elenco Segmenti della Rampa
     c = parent.getComponent(36);
     if (c.isVisible())  Label7.setText("View segments of the ramps:");
     {c.setBounds(insets.left+299,insets.top+330,180,24);}

     //TexTArea leggi Testo da File, elenco segmenti rampe
     c = parent.getComponent(37);
     if (c.isVisible())
     {c.setBounds(insets.left+290,insets.top+360,200,180);}

     //Label8 di visualizzazione numero segmenti rampa
     c = parent.getComponent(38);
     if (c.isVisible())  //
     {c.setBounds(insets.left+330,insets.top+330,200,440);}

     // Pulsante  view Ramps File, dal file
     c = parent.getComponent(39);
//     if (c.isVisible()) {c.setBounds(insets.left+550,insets.top+510,110,24);}

     // Label10, visualizzazione scritta reset prima di inizio nuova operazione
     c = parent.getComponent(40); Label10.setText("<html>Before a new task
click<br>the Reset button:</html>");
   if (c.isVisible()) {c.setBounds(insets.left+540,insets.top+340,150,50);}

     // Pulsante reset rampa
     c = parent.getComponent(41);
     if (c.isVisible()) {c.setBounds(insets.left+550,insets.top+390,110,24);}

     //Label9 di visualizzazione inserimento nuovo segmento di rampa, count +1
     c = parent.getComponent(42);
```

```
     if (c.isVisible()) {c.setBounds(insets.left+80,insets.top+340,150,30);}

     // Panel4 con testo" temperature scan, riquadro
     c = parent.getComponent(43);
     if (c.isVisible()){c.setBounds(insets.left+20,insets.top+320,680,250);}

//// END Riquadro Temperature SCAN

     // Pulsante Clear
     c = parent.getComponent(44);
//     if (c.isVisible()){c.setBounds(insets.left+20,insets.top+250,100,24);}

     //Label1 mostra lettura dalla TextField e dai pulsanti (UNIVERSALE)
     c = parent.getComponent(45); //Label1.setText("Ciao:");
     if (c.isVisible()){c.setBounds(insets.left+450,insets.top+250,300,24);}

  // Inizio Riquadro Panel 5 Set Loop
     // Mostro scritta inserimento nuovo loop
     c = parent.getComponent(46); scrittaloop.setText("<html>Set new value
Loop: </html>");
     if (c.isVisible()){c.setBounds(insets.left+520,insets.top+225,100,35);}
     //Mostro il campo vuoto per l'inserimento del nuovo valore
     c = parent.getComponent(47);
     if (c.isVisible()){c.setBounds(insets.left+620,insets.top+235,50,24);}
   // pulsante invio nuovo valore loop
     c = parent.getComponent(48);
     if (c.isVisible()){c.setBounds(insets.left+550,insets.top+270,100,24);}
  // Immagine Loop
     c = parent.getComponent(49);
     if (c.isVisible()){c.setBounds(insets.left+510,insets.top+65,180,180);}
     // Panel5 con testo" Set Loop, riquadro
     c = parent.getComponent(50);
     if (c.isVisible()){c.setBounds(insets.left+500,insets.top+70,200,240);}

    /**
 */
 }
}


public class gestioneporta implements SerialPortEventListener { // implements
Runnable, SerialPortEventListener

CommPortIdentifier comid;
Thread gestore;
InputStream in;
OutputStream out;
Enumeration e;
boolean trovata=false;
SerialPort portaseriale;
// String portadefault="COM2";
String portadefault;
int transfer_rate;

public gestioneporta() {
        //Parametri lettura dal file Configuration.txt
          InputStreamReader isr = null;
          InputStream is = null;
          StringBuffer sb = new StringBuffer();

          char[] buf = new char[1024];
          int len;
```

```
        try{
                is = new FileInputStream("Configuration.txt");
                isr = new InputStreamReader(is);
                while ((len = isr.read(buf))>0)
                        sb.append(buf, 0, len);
                //System.out.println("sb: "+sb.toString());
                String Testo = sb.toString().trim();
                //System.out.println("Testo: "+Testo);
                Boolean COM = Testo.contains("COM");
                //System.out.println(COM);
                if (COM.toString().equals("true")){
                        String str;
                        str = sb.toString().trim();
                                String[] strings= str.split("=");
                                String portaserialefile =
strings[1].substring(0,5).trim();
                                System.out.println(portaserialefile);
                                String transfer_ratefile =
strings[2].substring(0,5).trim();
                                System.out.println(transfer_ratefile);
                  // Adesso memorizzo la porta trovata dal file Configuration
all'interno di una variabile
                                transfer_rate =
Integer.parseInt(transfer_ratefile);
                                portadefault=portaserialefile;
                        //System.out.println("Porta COM trovata");
                     is.close();
                   } else {
                  System.out.println("Non hai specificato la porta seriale nel
file Configuration.txt");
                  is.close();
           }
                        // Apre la porta seriale passatagli come parametro
stringa
                        e=CommPortIdentifier.getPortIdentifiers();
                        while (e.hasMoreElements()) {
                        comid = (CommPortIdentifier)e.nextElement();

                        if(comid.getPortType()== CommPortIdentifier.PORT_SERIAL)
{
                            if(comid.getName().equals(portadefault)) {
                                        trovata=true;
                                        System.out.println("");
                                        System.out.println("Found serial
port "+comid.getName()+" waiting...");

                                        System.out.println("");

                        try {

        portaseriale=(SerialPort)comid.open("Gestioneporte",3000); // La
stringa indiva il nome dell'applicazione che sta usando la porta
                                //timeout, se la porta nn  disponibile prima
dello scadere dello stesso parte l'eccezione di porta in uso
                                try{
                                        gestore.sleep(1500);
                                }
                                catch(InterruptedException e) {
                                }

                                System.out.println("");
```

```
                                System.out.println("Connected...");
//comid.getName()
                                Label3.setText("Connected!");
                                System.out.println("");
                        }
                        catch (PortInUseException e) {
                        System.out.println("");
                        System.out.println("Port in use.");
                        System.out.println("");
                        }
                        // Lo stream in ingresso viene collegato alla seriale

                        try {
                                in = portaseriale.getInputStream();

                        }
                        catch (IOException e) {
                        }

                        try {
                                portaseriale.addEventListener(this);
                        }
                        catch (TooManyListenersException e) {

                                System.out.println("errore");
                        }

                        portaseriale.notifyOnDataAvailable(true);

                        try {
                                //portaseriale.setSerialPortParams(9600,
SerialPort.DATABITS_8,SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);
                                portaseriale.setSerialPortParams(transfer_rate,
SerialPort.DATABITS_8,SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);

        portaseriale.setFlowControlMode(SerialPort.FLOWCONTROL_RTSCTS_OUT);
                        }
                        catch (UnsupportedCommOperationException e) {
                        }

                            } // Connetti
                        }
                        }

                        if(!trovata) {

                                System.out.println("");
                                System.out.println("Port "+portadefault+" not
found.");
                                Label3.setText("Port "+portadefault+" not
found.");
                                System.out.println("");
                        }
        }// fine del try iniziale
        catch (Exception e) {
                System.out.println("");
                System.out.println("Connessione gia stabilita");
                Label3.setText("Connessione gia stabilita");
                System.out.println("");
        }
}
```

```
public void serialEvent(SerialPortEvent event) {

switch (event.getEventType()) {

case SerialPortEvent.BI:
case SerialPortEvent.OE:
case SerialPortEvent.FE:
case SerialPortEvent.PE:
case SerialPortEvent.CD:
case SerialPortEvent.CTS:
case SerialPortEvent.DSR:
case SerialPortEvent.RI:
case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
break;
case SerialPortEvent.DATA_AVAILABLE:

        byte[] readBuffer = new byte[1024];
        int numByte=0;
        try {
                //int c;
                //char car;
                String risposta=" ";
                //String risp="";

                while (in.available()>0 ) {

                        numByte=in.read(readBuffer); // Mi restituisce il numero
di byte che compongono la risposta
                }

                String result = new String(readBuffer,0,numByte);
                // System.out.println("lettura: " +result);
                // Label1.setText("Outflow Temperature: "+result.toString());

                String a = "IN_PV_00";
                String b = "RMP_IN_01";
                String c = "IN_SP_00";
                String d = "IN_SP_01"; //Pump output
                String e = "IN_SP_02"; //Refrigeration
                String f = "RMP_IN_02"; // Mostra Setting Loop
                String g = "RMP_IN_03"; // Mostra Actual Loop
                String var = valore.trim();

                if(var.equals(a))
                {
                //System.out.println("Outflow Temperature: " +result);
                // Lmostravaloreactualtemp.setText(result.toString());
                String res = result.toString().trim().substring(1,6);
                Lmostravaloreactualtemp.setText(" "+res);
                }
                else if (var.equals(c))
                {
                        //System.out.println("Setting Temperature (read):
"+result);
                        //Lmostravaloresettingtemp.setText(result.toString());
                        String res = result.toString().trim().substring(1,6);
                        Lmostravaloresettingtemp.setText(" "+res);
                }
                else if (var.equals(b))
                {
                        //System.out.println("Current Segment: "+result);
                        //Lmostravalorecurrentsegm.setText(result.toString());
```

```
                        String res = result.toString().trim().substring(1,3);
                        Lmostravalorecurrentsegm.setText(" "+res);
                }
                else if (var.equals(d))
                {
                        String pump = result.trim();
                        //System.out.println("Pump Output: "+result);
                        //      Lmostravalorepumpout.setText(result.toString());
                        if (pump.equals("000.00")){
                                Lmostravalorepumpout.setText("Stopped");
                        }
                        else if (pump.equals("001.00")){
                                Lmostravalorepumpout.setText("Low");
                        }
                        else if (pump.equals("003.00")){
                                Lmostravalorepumpout.setText("Medium");
                        }
                        else if (pump.equals("005.00")){
                                Lmostravalorepumpout.setText("Maximum");
                        }
                        else {
                                Lmostravalorepumpout.setText(result);
                        }
                }
                else if (var.equals(e))
                {
                        //System.out.println("Mode Refrigeration: "+result);
                        //
Lmostravalorerefrigeration.setText(result.toString());
                        String refrig = result.trim();
                        if (refrig.equals("000.00")){
                                Lmostravalorerefrigeration.setText("OFF");
                        }
                        else if (refrig.equals("001.00")){
                                Lmostravalorerefrigeration.setText("100%");
                        }
                        else if (refrig.equals("002.00")){
                                Lmostravalorerefrigeration.setText("50%");
                        }
                        else if (refrig.equals("003.00")){
                                Lmostravalorerefrigeration.setText("Automatic");
                        }
                        else {
                                Lmostravalorerefrigeration.setText(result);
                        }

                }
                else if (var.equals(f))
                {
                        //System.out.println("Current loop: "+result);
                        //loopsettingmostra.setText(result.toString());
                        String res = result.toString().trim().substring(1,3);
                        loopsettingmostra.setText(" "+res);
                }
                else if (var.equals(g))
                {
                        //System.out.println("Actual loop: "+result);
                        loopactualmostra.setText(result.toString());
                        String res = result.toString().trim().substring(1,3);
                        loopactualmostra.setText(" "+res);
                }
                else
```

```
                {
                        //System.out.println("lettura universale: "+result);
                        Label1.setText(result.toString());
                }


                /** Questo sotto per creare un file dove posso analizzare la
temperatura di overflow ogni 3 secondi
                 * e studiarmi l'oscillazione della rampa in base al tempo
trascorso
                        */
                    //    FileOutputStream temperatura = new
FileOutputStream("temperatura.txt", true);
                    //    PrintStream scrivi = new PrintStream(temperatura);
                        //      scrivi.println(result.toString());

        }

        catch (IOException e) {}
        }
}
//Serialevent


////////////////////////////////////////////////////////////////////////////
//////////////
//Scrivi()
////////////////////////////////////////////////////////////////////////////
//////////////

public String valore;
public void scrivi(String s) {

//Connettiamo lo stream d'uscita alla porta seriale passata come parametro

try {
                try {
                                out=portaseriale.getOutputStream();
                                //PrintStream pr=new PrintStream(out);
                        }
                catch (IOException e) {
                                System.out.println("");
                                System.out.println("Impossibile connettere lo
stream d'uscita");
                                System.out.println("");
                        }

                        // Settiamo i parametri per la scrittura

                try {

                                //portaseriale.setSerialPortParams(9600,
SerialPort.DATABITS_8,SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);
                                // Setto Hardware = Handshake
FLOWCONTROL_RTSCTS_OUT or FLOWCONTROL_RTSCTS_IN

                                portaseriale.setSerialPortParams(transfer_rate,
SerialPort.DATABITS_8,SerialPort.STOPBITS_1, SerialPort.PARITY_NONE);

        portaseriale.setFlowControlMode(SerialPort.FLOWCONTROL_RTSCTS_OUT);
                    }
                catch (UnsupportedCommOperationException e) {
```

```
                        }

                try {
                                portaseriale.notifyOnOutputEmpty(true);
                        }
                catch (Exception e) {
                                System.out.println("");
                                System.out.println("Error setting event
notification");
                                Label1.setText("Error setting event
notification");
                                System.out.println(e.toString());
                                System.exit(-1);
                        }

                try {
                                //out.write(comando.getBytes());
                             out.write(s.getBytes());
                             //System.out.println("Mostro quello che scrivo:"
+s);
                             valore = s;
                             //System.out.println("Mostro Valore:" +valore);
                               out.flush();
                        }
                catch (IOException e) {

                        System.out.println("");
                        System.out.println("Impossibile scrivere sull'out");
                        Label1.setText("Impossibile scrivere");
                        System.out.println("");
                        }

}
catch (Exception e) {
System.out.println("");
System.out.println("No connected...");
//System.out.println("Leggo File di Configurazione: "+line);
Label3.setText("No connected...");
System.out.println("");
}
}


////////////////////////////////////////////////////////////////////////////
//////////////
//Disconnetti()
////////////////////////////////////////////////////////////////////////////
//////////////

public void disconnetti() {

try {
            timer.cancel();

                leggitesto.setText("");
                Lmostravaloreactualtemp.setText("");
                Lmostravaloresettingtemp.setText("");
                Lmostravalorecurrentsegm.setText("");
                Lmostravalorepumpout.setText("");
                Lmostravalorerefrigeration.setText("");
                loopsettingmostra.setText("");
                loopactualmostra.setText("");
```

```
                portaseriale.close();
                System.out.println("");
                System.out.println("Disconnected...");
                Label3.setText("Disconnected!");
                System.out.println("");
                //System.exit(0);
        }

catch (Exception e) {

                System.out.println("");
                System.out.println("No connected...");
                Label3.setText("No connected!");
                System.out.println("");
        }
}
}
```

7. Conclusioni

Lo sviluppo della piattaforma software, in linguaggio Java, si è dimostrato altamente stimolante. Durante la sua evoluzione, come accade normalmente, si sono manifestati problemi di varia natura, ma che sono stati progressivamente risolti.
Nonostante ciò il sistema si è dimostrato funzionante, pratico, e soprattutto funzionale per i sistemi operativi Windows. Il software permette una gestione ad alto livello del sistema di scrittura e lettura tra PC e termostato tramite l'interfaccia RS232.

8. Bibliografia

1) Interfaccia RS 232 https://www.virtual-serial-port.org/it/article/what-is-serial-port/rs232-pinout/
2) Piedinatura DB9 RS 232 https://www.eltima.com/it/article/9-pin-serial-port.html
3) IDE Eclipse https://www.eclipse.org/downloads/
4) WindowBuilder https://www.eclipse.org/windowbuilder/
5) Installazione Plugins https://www.html.it/pag/48224/installazione-e-prima-panoramica-del-plugin/
6) GUI Java https://www.html.it/pag/15115/interfacce-grafiche-gui-e-awt/
7) Porte Seriali e libreria RxTX https://www.javastaff.com/2007/04/23/comunicazione-su-porta-serialeparallela-in-java/