PROGRAMMARE in JAVA

Continuiamo il nostro viaggio nel fantastico mondo del linguaggio Java. Questa volta vi proponiamo un esempio di come sia possibile gestire una porta USB sfruttando la nostra scheda LX.1734 per ottenere 1.000 strumenti diversi, che potrete utilizzare con i sistemi operativi più comuni: Windows, Linux, Mac.

Come abbiamo accennato nell'articolo "**Programmare in JAVA la porta seriale**" pubblicato nella rivista **N.233**, il linguaggio di programmazione Java nasce per la programmazione orientata agli oggetti con il grande pregio della facile **portabilità**.

È infatti possibile creare delle applicazioni scrivendo il software una volta sola, indipendentemente dal tipo di computer sul quale verranno eseguite.

Questa caratteristica è possibile grazie alla cosiddetta **J**ava **V**irtual **M**achine, nota anche come **JVM**, che mette in esecuzione il software su tutte le più note piattaforme: **Windows**, **Linux**, **Mac OS X** e **Solaris**.

Inoltre Java è un linguaggio di tipo **open source** e quindi risulta molto facile trovare applicazioni e librerie senza problemi di Copyright e licenze d'uso.

Difatti nel presente articolo verrà fatto uso della libreria **RXTXcomm.jar**, per la gestione delle porte seriali e parallele, che è liberamente scaricabile da Internet.

Verranno inoltre illustrati questi concetti in maniera operativa, grazie all'aiuto del nostro **amico esper-**

to progettista di sistemi Dott. Ing. Pier Alessandro Aisa che ci ha aiutato a sviluppare un paio di applicazioni Java, all'interno di un ambiente di sviluppo software integrato **IDE** (Integrated Development Environment), che prende il nome di **NetBeans**.

I due applicativi Java sono da utilizzare con la "**Scheda USB 1000 usi**" **LX.1734**, pubblicata nella rivista **N.239** e si tratta di un **Monitor dei dati** che transitano sulla porta USB e di un **DataLogger** per il monitoraggio dei dati forniti dai sensori collegati alla scheda USB (come ad esempio la temperatura ambiente, la misura del campo magnetico oppure la misura della temperatura della pelle tramite una termopila).

Come abbiamo detto, trattandosi di software open source i pacchetti necessari all'installazione dell'ambiente **NetBeans IDE** e le librerie per la gestione della porta USB sono tutti scaricabili gratuitamente da Internet.

Per praticità abbiamo inserito nel CDRom che vi forniremo i pacchetti di installazione per **Windows**, **Linux** e **Mac**.

DUE PAROLE sulla PROGRAMMAZIONE ad OGGETTI

Prima dell'avvento della programmazione ad oggetti, la programmazione classica si basava sul modello del cosiddetto **paradigma procedurale**, che consisteva nell'avere un approccio di scomposizione di un problema complesso in problemi più semplici, al fine di identificare e creare le funzioni necessarie all'applicazione software finale.

Questo approccio, sebbene ben strutturato soffre però di problemi di integrità e consistenza del software, nel senso che obbliga il programmatore a ricordarsi sempre come ha organizzato il software, dove ha dichiarato le variabili per evitare di incappare in spiacevoli **bugs**, difficilmente identificabili.

Difatti, se il progetto risulta particolarmente complesso si rischia di generare molti moduli software e di perdere visibilità sulle relazioni tra essi.

La **programmazione ad oggetti**, previene questi rischi affrontando il problema in maniera differente, adottando cioè un modo diverso di vedere le cose.

la PORTA USB

Fig.1 Programmazione orientata agli oggetti.

Per capire la differenza tra la programmazione tradizionale e la **programmazione orientata** agli **oggetti** possiamo ricorrere al seguente esempio.

Immaginiamo di dover costruire una casa. Il programmatore lo si può paragonare al muratore, la casa al programma software e gli attrezzi del muratore all'ambiente di sviluppo software (vedi fig.1).

Il "**muratore tradizionale**" per costruire una parete della casa si procura gli elementi primari (cemento, acqua, mattoni, vernice, pennello), prepara il cemento ed impilando i mattoni uno dietro l'altro comincia ad erigere un muro.

Quando ha terminato, prepara la vernice e comincia a colorare il muro ottenendo una parete colorata.

Per ogni nuova parete che deve costruire l'approccio è sempre lo stesso, ripetendo le stesse fasi dall'inizio.

Il "**muratore orientato agli oggetti**", invece, si trova a disposizione appunto degli oggetti già prefabbricati (che in gergo vengono chiamati **classi**), di cui può utilizzare le caratteristiche che più fanno al caso suo.

Nell'esempio specifico possiamo pensare a delle **pareti prefabbricate**, da personalizzare per ottenere dei pannelli di cui si può decidere la forma, lo spessore, il colore e la funzione (come ad esempio muro portante o muro di separazione).

Ogni nuovo pannello che il muratore orientato agli oggetti costruirà, sarà un'**istanza** della classe "pareti", cioè una particolare occorrenza, che eredita le caratteristiche della **classe** necessarie per ottenere lo scopo del muratore.

Da questo esempio si può dedurre che il muratore orientato agli oggetti è più facilitato nel suo compito, perché non deve ripartire dagli elementi di base ogni volta, ma può sfruttare dei modelli già esistenti (**oggetti** appunto) che offrono funzioni e hanno caratteristiche già pre-definite e impostare le caratteristiche specifiche di cui ha bisogno.

JAVA e l'AMBIENTE NETBEANS IDE

Le applicazioni Java possono essere organizzate in molti files di diverso tipo: ad esempio i sorgenti hanno estensione **.java**, i files compilati estensione **.class**, gli eseguibili, i files di libreria e gli archivi hanno estensione **.jar**.

Un'applicazione inoltre può essere composta da molti oggetti grafici ed utilizzare molte librerie a supporto.

Appare allora evidente che per gestire applicazioni di una certa complessità non è consigliabile lavorare da riga comando ed utilizzando semplici editor di testo quali il **blocco note** sotto Windows (oppure **nano** in ambiente Linux, o **TextEdit** in ambiente Mac), ma si rende necessario un ambiente di programmazione integrato, che dia l'accesso in maniera facile e veloce alle informazioni.

NetBeans IDE è un ambiente di sviluppo multi-linguaggio scritto interamente in Java e nato all'incirca negli anni 2000.

Sun Microsystems scelse NetBeans come IDE ufficiale, da contrapporre al più diffuso **Eclipse**, sviluppato da un consorzio di note società quali Intel, HP e IBM ed anch'esso scritto in Java.

È inoltre possibile corredare Netbeans di molteplici **plug-ins** che lo rendono completo ed appunto multilinguaggio, anche se richiede almeno **512 Megabytes** di Ram a causa dell'uso delle librerie grafiche standard di Java note come **Swing** e questo forse costituisce un punto di debolezza nei confronti del suo avversario Eclipse, che adotta invece librerie note come **SWT**.

Nel presente articolo si farà esclusivamente riferimento al linguaggio Java.
Citiamo per completezza alcuni dei linguaggi più noti supportati da NetBeans: **Ajax C/C++**, **Groovy**, **Grails**, **JavaScript Mobile**, **PHP**, **Python Ruby**, **XML**.

Un po' di storia

NetBeans nasce nel 1996 da un progetto dell'Università di Matematica e Fisica "**Charles**" di Praga, sotto il nome di "**Xelfi**".
Nel 1998 esce una prima versione del progetto che rimase commerciale, finché non venne acquisito nel 1999 da **Sun Microsystems**, che decise di rendere l'IDE open source, distribuendone i sorgenti.

Nel 2000 nacque il sito **www.netbeans.org**, a cui la comunità di sviluppatori fa riferimento tutt'oggi, per il proseguimento delle attività di sviluppo che hanno portato Netbeans alla attuale versione **6.9**.

Le caratteristiche principali dell'AMBIENTE NETBEANS

NetBeans IDE offre tutte le potenzialità di un ambiente di sviluppo software di nuova generazione.
Citiamo per semplicità alcune delle principali funzionalità, che verranno utilizzate in seguito per la realizzazione degli applicativi.

Una delle caratteristiche fondamentali di NetBeans è l'alto grado di **automazione**, che si esprime attraverso segnalazioni che sono state pensate per ridurre al minimo lo sforzo del programmatore nelle operazioni di scrittura del software.

Ad esempio molto utile è la funzione di **code-templates** che permette tramite abbreviativi denominati **keystrokes** la scrittura delle istruzioni più utilizzate. Digitando **sout** e premendo TAB, viene creata in automatico l'istruzione **System.out.println("")**, da utilizzare per mandare in output del testo.

Fig.2 La scheda dell'interfaccia USB KM1734K pubblicata nella rivista N.239 e i collegamenti ai vari accessori, vale a dire il modulo temperatura ambiente KM1734KT, la presa jack femmina, il diodo led e la sua resistenza.

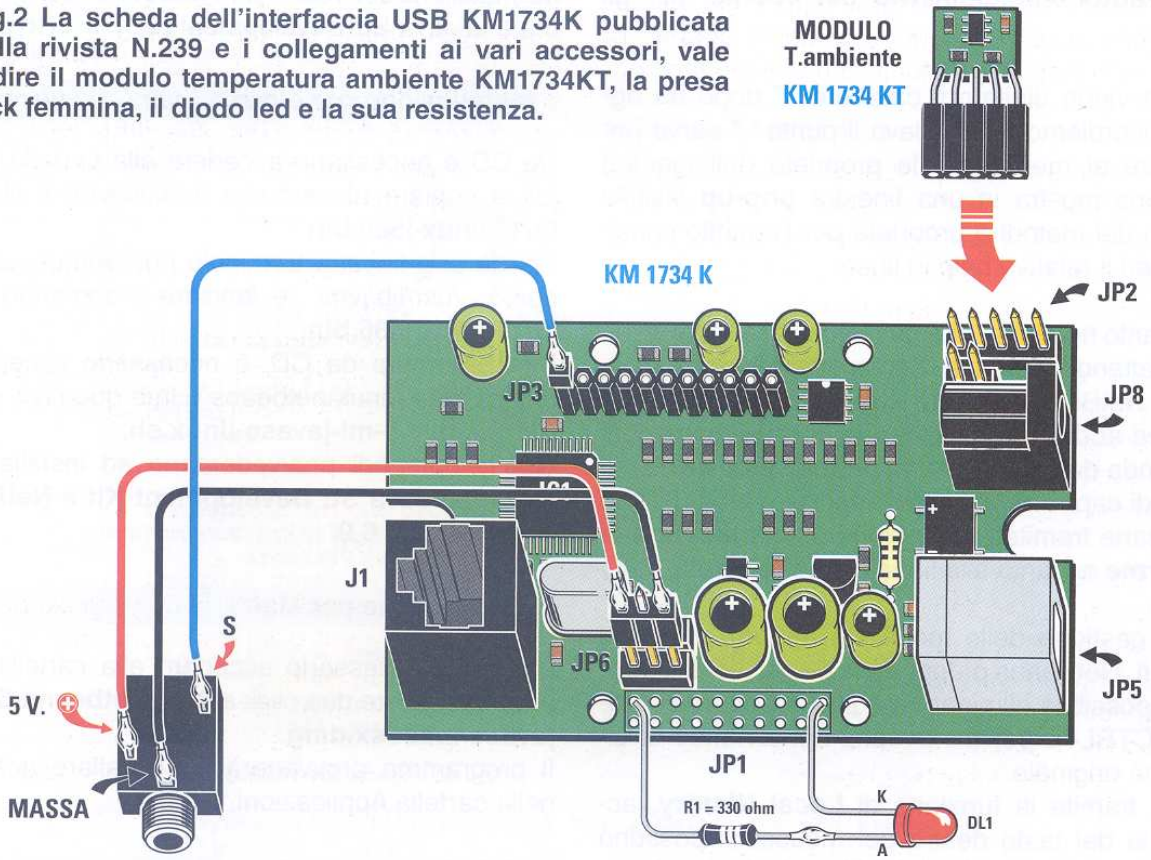
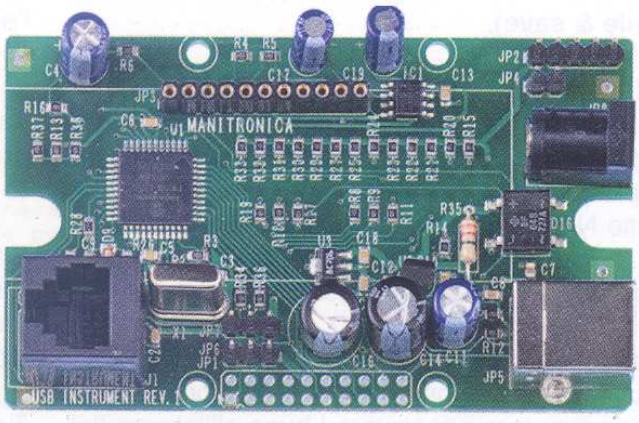


Fig.3 Foto della scheda dell'interfaccia USB che forniamo già montata con componenti SMD e collaudata.



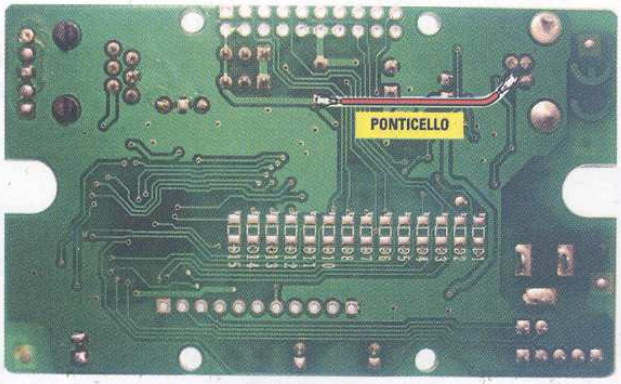


Fig.4 Foto della scheda USB vista dal retro, dal lato cioè sul quale dovrete saldare il ponticello necessario per trasferire a tutti i componenti l'alimentazione dei 5 Volt provenienti dalla presa USB del computer.

Sotto la stessa famiglia di funzioni avanzate si può citare l'**autocompletamento del codice**, per gli oggetti.

Appena viene digitato il carattere "." dopo un oggetto (ricordiamo che in Java il punto "." serve per accedere ai metodi o alle proprietà dell'oggetto) NetBeans mostra in una finestra **pop-up** visibile l'elenco dei metodi e proprietà per l'oggetto considerato ed il relativo help in linea.

Per quanto riguarda l'intercettazione di errori non bisogna attendere la fase di compilazione del codice, perché NetBeans durante la digitazione **notifica** gli errori ed addirittura **propone** le possibili correzioni a seconda del tipo di errore, permettendo all'utente anche di capire subito dove apportare le modifiche necessarie tramite l'accensione di una **lampadina di allarme** accanto alla linea di codice errata.

Per la gestione delle modifiche apportate ai files sorgenti, NetBeans propone un alto livello di **Undo**: è cioè possibile eliminare modifiche apportate battendo CTRL+Z per molte volte ripristinando la situazione originale.
Inoltre, tramite la funzione di **Local History**, accessibile dal tasto destro del mouse, si possono ottenere tutte le versioni del file salvate (ogni volta che si manda in esecuzione il codice NetBeans provvede a salvare i sorgenti, proprietà nota come **compile & save**).

Una funzione fondamentale è quella nota con il nome di **Refactor**, che garantisce la possibilità di ristrutturare il software in maniera automatica; ad esempio per rinominare un oggetto all'interno del progetto NetBeans si fa carico di eseguire l'operazione in maniera consistente, trovando tutte le occorrenze dell'oggetto e modificandole.

NetBeans è dotato inoltre di un potente **Debugger**, che rende disponibili tutte le funzioni in aiuto al programmatore per trovare i bugs all'interno del software.

INSTALLAZIONE di NETBEANS IDE 6.9

Al momento della redazione del presente articolo l'ultima versione disponibile di NetBeans è la 6.9, che potete installare a partire dal CD o da Internet dal sito **www.netbeans.org**.

Installazione per Windows

Da CD è necessario accedere alla cartella NetBeans e fare due click sul file **netbeans-6.9-ml-javase-windows.exe**. Questo programma provvederà ad installare sul vostro pc NetBeans IDE versione 6.9 e la Java SE Development Kit.

Installazione per Linux

Da CD è necessario accedere alla cartella Linux\jdk e copiare al percorso /usr/lib/jvm/ il file **jdk-6u17-linux-i586.bin**.
Poi da una finestra terminale posizionarsi sul percorso /usr/lib/jvm/ e lanciare il comando **./jdk-6u17-linux-i586.bin**.
Infine, sempre da CD, è necessario accedere e alla cartella Linux\netbeans e fare due click sul file **netbeans-6.9-ml-javase-linux.sh**.
Questi comandi provvederanno ad installare sul vostro pc **Java SE Development Kit** e **NetBeans IDE versione 6.9**.

Installazione per Mac

Da CD è necessario accedere alla cartella Mac\netbeans e fare due click sul file **Netbeans-6.9-ml-javase-macosx.dmg**.
Il programma provvederà ad installare netbeans nella cartella Applicazioni.

TUTORIAL e DEMO

Una volta terminata l'installazione potete lanciare l'ambiente facendo due click sull'icona "NetBeans IDE 6.9" presente sul Desktop.

La Start page che si presenta sulla destra del pannello all'avvio di IDE presenta due viste: **Welcome to NetBeans IDE** e **My NetBeans**.

Entrambe le viste offrono molti collegamenti interessanti, come ad esempio alcuni tutorials e demo, che spiegano passo passo come costruire applicazioni.

Tutti i link presenti all'interno della Start page richiedono la connessione Internet per essere visionati.
In particolare vi consigliamo di vedere il Quick Start Tutorial, che in pochi minuti vi permetterà di creare un semplice programma che stampa a video la tipica scritta "**Hello World !**".

Due APPLICATIVI per la GESTIONE della PORTA USB

Adesso divertiamoci a creare un paio di applicativi Java utilizzando NetBeans e la scheda "**USB 1.000 usi**" **LX.1734** progettata dal nostro carissimo

```
/* Nuova Elettronica - 2010
 * LeggiUSB.java 2.0
 * Date: 04/08/2010
 * Author: P.A. Aisa
 * Copyright (c) 2010 Nuova Elettronica. All Rights Reserved.
 */

import java.io.InputStream;
import java.io.IOException;
import gnu.io.*;

public class Main {
    public static void main(String[] args) {
        // Dichiarazione variabili
        int                     idx = 0;
        CommPortIdentifier      portId ;
        SerialPort              port = null;
        InputStream             in;
        int                     charx ;
        int                     charread = 0 ;
        // Stringa di inizio programma
        System.out.println("START Program USB: Linea comando " + args[0]+ " "
                + args[1]);
        // Apertura Porta
        try
            {
                portId = CommPortIdentifier.getPortIdentifier(args[1]);
                System.out.println("Apertura porta " + portId.getName());
                System.out.println("Porta "
                 + portId.getName() + " aperta con successo");
            }
        catch (NoSuchPortException e)
            {
                System.out.println("Porta " + args[1] + " non trovata !");
                 portId = null ;
            }
        try
            {
                port = (SerialPort) portId.open("TempLogger", 2000);
                if (port == null)
                    {
                        System.out.println("Errore nell'apertura della porta"
                      + portId.getName());
                    }
            }
        catch (PortInUseException e)
            {
                System.out.println("Attesa della coda per la porta  "
                + portId.getName() + ": porta utilizzata da " + e.currentOwner);
            }
        // Lettura e Stampa dati da USB
        System.out.println("**********************\nLettura dati da USB" +
                "\n**********************");
        try
            {
                in = port.getInputStream();
                while (charread <300)
                    {
                        charx =  in.read();
                        System.out.print((char)charx) ;
                        charread++ ;
                    }
            }
        catch (IOException e)
                {
                 System.out.println("Non riesco ad aprire la input stream");
                }
    // Chiusura Porta
        System.out.println("**********************\nChiusura porta " + port) ;
        port.close() ;
    }}
```

1 2 3 4 5 6

Figura 5

collaboratore **Dott. Ing. Alessandro Manigrassi** e pubblicata nella rivista N.239.

Per semplificare al massimo l'attività e per spiegare bene la differenza tra programmazione tradizionale e programmazione ad oggetti, procediamo con la creazione di due applicativi Java:

- **LeggiUSB**: applicativo Java senza interfaccia grafica, realizzato secondo l'approccio tradizionale per il monitoraggio del traffico di dati generato dalla scheda **LX.1734** sulla porta USB.

- **DataLogger**: applicativo con interfaccia grafica realizzato seguendo i canoni della programmazione ad oggetti per il monitoraggio di alcune grandezze fisiche quali la temperatura ambiente e il campo magnetico, acquisiti dalla scheda **LX.1734**.

APPLICATIVO LEGGIUSB.JAVA in ambiente WINDOWS

Le seguenti istruzioni fanno riferimento alla piattaforma per **Windows**.

Per l'installazione **Linux** e **Mac** fate riferimento al prossimo paragrafo.

L'applicativo **LeggiUSB** tramite la porta USB, legge i dati generati dalla scheda **LX.1734**, relativi ai canali di acquisizione analogico digitali e ne stampa il contenuto a video.

L'esempio è volutamente semplificato al massimo per fare capire come gestire la porta USB utilizzata come **porta seriale "virtuale"** (la porta seriale viene detta virtuale in quanto è il driver che si appoggia alla porta USB, che la fa vedere al sistema operativo come una porta seriale).

Il programma si compone di un unico file contenente la sola classe **Main**, necessaria alla **Java Virtual Machine** per mettere in esecuzione il programma (vedi i riferimenti ai blocchi in fig.5 dove viene riportato in codice, per comodità), così strutturata:

1 - importazione delle librerie necessarie
2 - dichiarazione delle variabili
3 - apertura della porta seriale
4 - lettura dalla porta dei dati
5 - stampa dei dati a video
6 - chiusura della porta seriale

Prima di utilizzare NetBeans è necessario installare i **driver USB-seriale**.

Per fare questo occorre connettere la scheda e quando viene richiesta la posizione dei files per il driver bisogna dare come percorso il CD sotto la cartella Windows\driverCCS, come illustrato nelle figg.6-7.

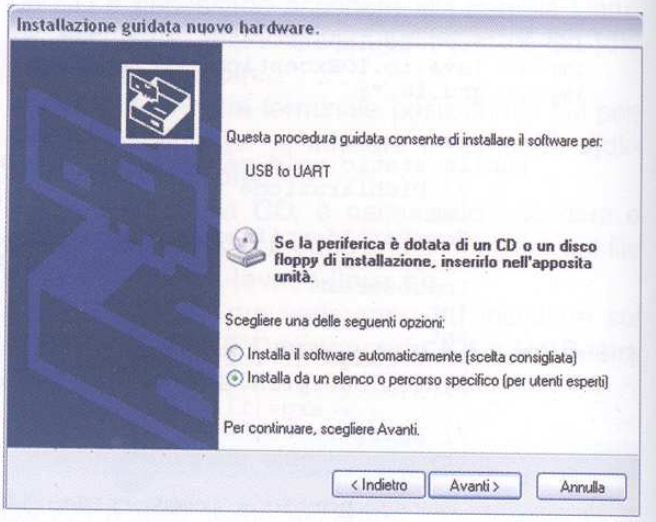


Figura 6

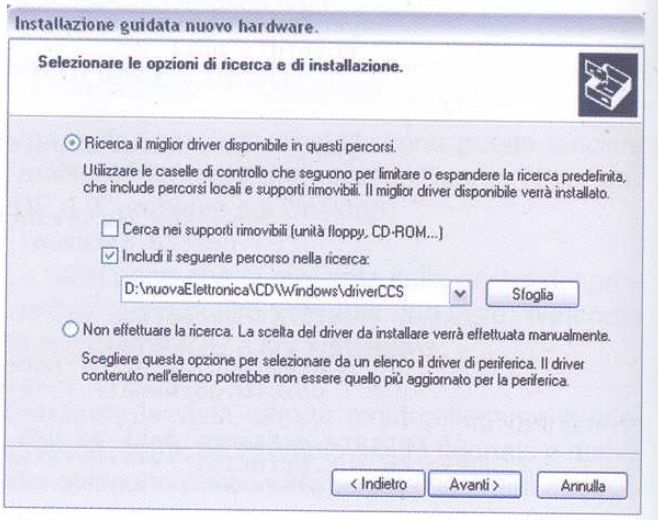


Figura 7

Poi è necessario copiare le cartelle **LeggiUSB** e **RXTX** da CD al percorso di lavoro (ad esempio nella cartella nE sotto il disco C, cioè al percorso C:\nE).

Per la creazione del vostro primo progetto NetBeans seguite le seguenti istruzioni:

1 - Aprite NetBeans dal menù Programmi di Windows e create un nuovo progetto, cliccando sul secondo tasto sotto la barra menù come evidenziato in fig.8: scegliete **Java Project with Existing Sources** e cliccate sul tasto **Next**.

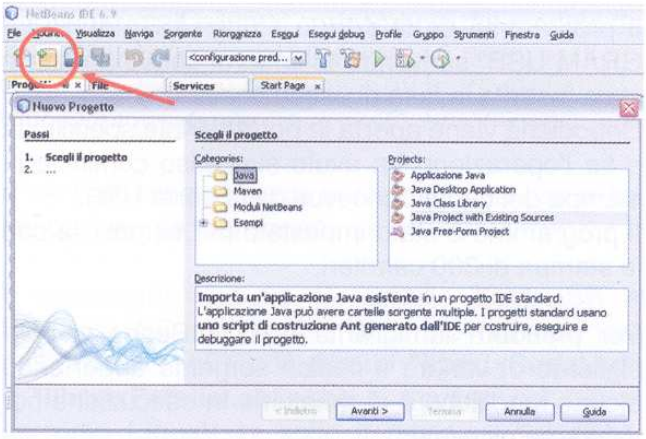


Figura 8

Ora digitate il nome **LeggiUSB** nel primo campo e **C:\nE\LeggiUSB** nel secondo campo (vedi fig.9) e cliccate sul tasto "Next".

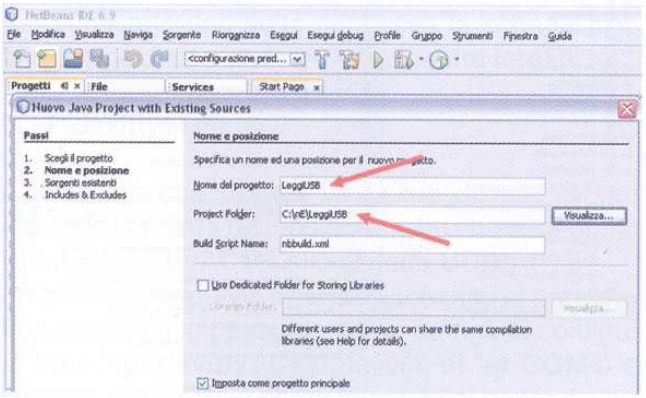


Figura 9

Ora cliccate sul tasto **Aggiungi cartella ...**, selezionate la cartella **C:\nE\LeggiUSB\sorgenti** (come indicato in fig.10) e cliccate sul tasto **Next** e sul tasto **Finish**.

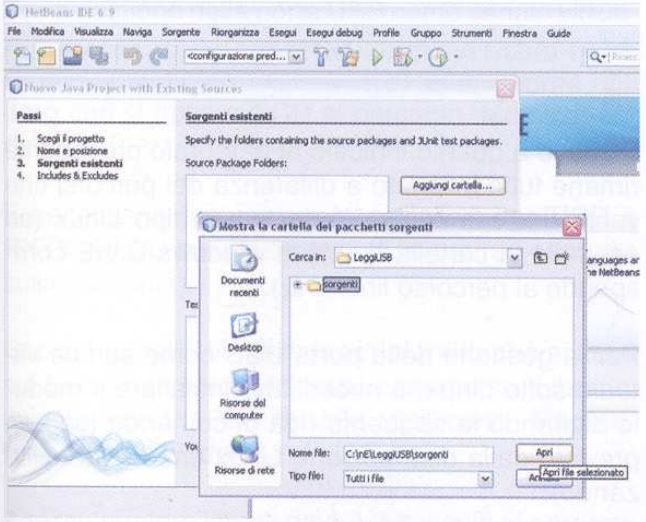


Figura 10

A questo punto nell'ambiente IDE compare il progetto **LeggiUSB** composto dai packages (equivalenti a delle cartelle) **<pacchetto predefinito>** e **Librerie**.

2 - Ora è necessario inserire nei files di progetto la libreria per la gestione della porta seriale: **RXTX-comm.jar**.
Fino a che questa operazione non verrà effettuata, infatti, IDE segnalerà errori a tutti i riferimenti agli oggetti di questa libreria.

Selezionate quindi il package **Librerie** premendo il tasto destro del mouse e selezionate la voce **Aggiungi cartella / JAR** e andate a selezionare nel percorso **C:\nE\RXTX** il file **RXTXcomm.jar**, e poi cliccate sul tasto **Apri**, come evidenziato in fig.11.

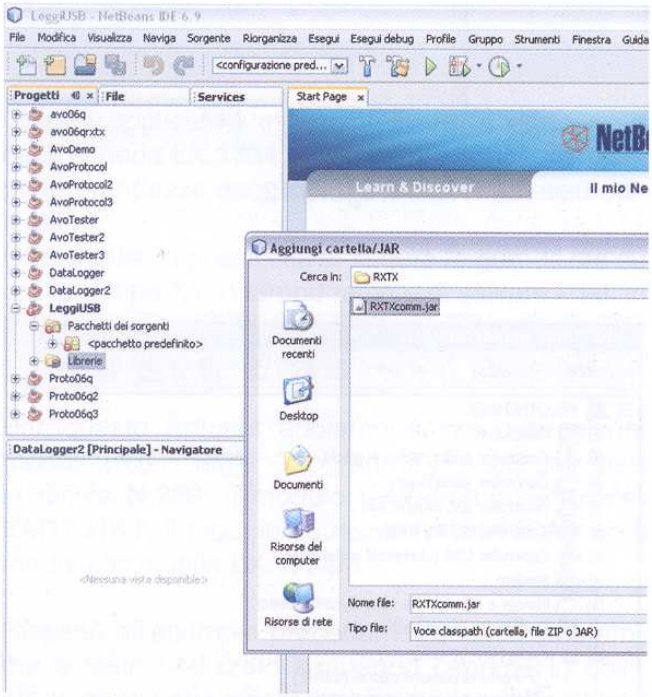


Figura 11

3 - Ora è necessario configurare l'esecuzione del programma inserendo la riga comandi per specificare la porta seriale su cui collegare la scheda USB.

Selezionate quindi dal menù di NetBeans **Esegui**, **Set Project Configuration ...**, **Personalizza ...** ed indicate nella riga comandi "**-p COM4**", sostituendo a COM4 la porta seriale a cui è collegata la scheda USB, come indicato in fig.12 e premete Ok.
Difatti nel nostro caso si tratta di COM4.

Per sapere su quale porta seriale del vostro pc viene vista la scheda, si può selezionare con il tasto destro su **Risorse del Computer** la voce **Proprietà** e poi dal pannello hardware, il tasto **Gestione**

Periferiche, poi **Porte** (COM e LPT) ed andare a trovare il numero della porta COM corrispondente alla voce "**USB to UART**", come indicato in fig.13.

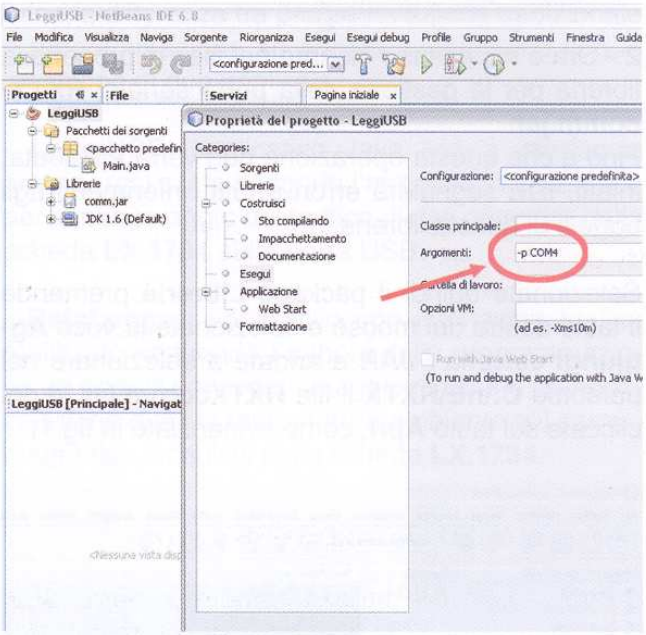


Figura 12

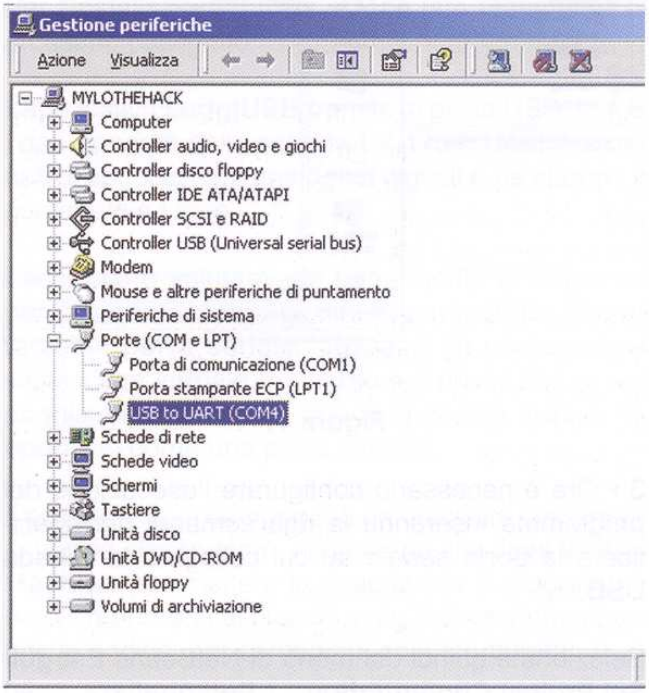


Figura 13

4 - A questo punto non vi resta che collegare la scheda **LX.1734** e mettere in esecuzione il codice, cliccando sul tasto con la "freccia verde" oppure premendo F6.
Il risultato dovrebbe apparire sulla finestra di Output dell'ambiente IDE come mostrato in fig.14.

Il programma parte con la stringa "START PROGRAM USB" scrivendo la linea di comando che è stata inserita in fase di esecuzione.
Dopodiché viene aperta la porta seriale specificata e se l'operazione ha avuto successo comincia la stampa dei caratteri ricevuti dalla porta USB.
Il programma è stato impostato per semplicità con la stampa di 300 caratteri.

Per prendere famigliarità con NetBeans vi consigliamo di variare il codice sorgente apportando alcune modifiche e di rimetterlo in esecuzione per valutarne gli effetti.
Ad esempio potreste variare il numero di caratteri acquisiti.

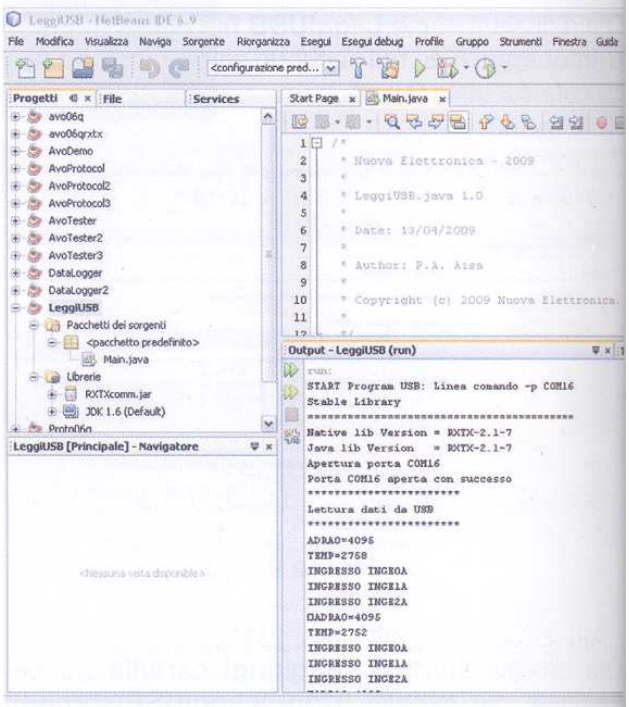


Figura 14

APPLICATIVO LEGGIUSB.JAVA in ambiente LINUX

Rispetto a quanto indicato al paragrafo precedente rimane tutto invariato a differenza dei percorsi che vanno indicati come file system di tipo Linux (ad esempio la cartella di lavoro windows C:\nE corrisponde al percorso linux /nE).

Per la gestione della porta USB come seriale virtuale sotto Linux, è necessario installare il modulo digitando la seguente riga di comando (se non previsto dalla distribuzione Linux che si sta utilizzando):

modprobe usbserial

Per verificare il corretto caricamento del modulo da riga di comando si può digitare:

lsmod | grep usb.

e verificare che compaia in risposta una riga di testo contenente "usbserial".

È necessario copiare le cartelle LeggiUSB e RXTX da CD al percorso di lavoro (ad esempio nella cartella \nE) e le librerie **"librxtxSerial.so"** e **"librxtxParallel.so"**, che si trovano nel CD al percorso **Linux\rxtx** sotto il percorso **"\usr\lib"**.

Adesso ripetete i punti da 1 a 2 descritti al paragrafo precedente.
Prima di procedere con il punto 3 è necessario verificare quale sia il nominativo della porta seriale assegnato da Linux, quando la scheda **LX.1734** viene connessa al pc.
Per verificarlo una volta connessa la scheda digitare la riga di comando da una finestra terminale:

ls –l /dev/tty*

e verificare che appaia la device di tipo tty (ad esempio la nostra distribuzione Linux ha assegnato il nominativo **ttyACM0** alla scheda **USB**).
Procedete quindi con il punto 3 descritto al paragrafo precedente impostando come stringa di configurazione "**-p /dev/ttyACM0**", invece di "**-p COM4**" e lanciate l'applicativo come descritto al punto 4.

APPLICATIVO LEGGIUSB.JAVA in ambiente Mac

Rispetto a quanto indicato ai paragrafi precedenti rimane tutto invariato a differenza dei percorsi che sono gestiti come cartelle Mac.
Per la gestione della porta USB come seriale virtuale sotto Mac è necessario installare il driver USB-seriale facendo due click sul file **PL2303_1.2.1r2.dmg** che si trova sul CD al percorso \Mac\usb-serial\.

È necessario poi copiare le cartelle **LeggiUSB** e **RXTX** dal CD su una cartella di lavoro (ad esempio sulla scrivania).

Copiate quindi il file **librxtxserial.jnilib** che si trova sul CD al percorso \Mac\rxtx\ al percorso \Libreria\ Java\Extensions accessibile dall'icona **Machintosh HD** del Finder.

Adesso ripetere i punti da 1 a 2 descritti al paragrafo precedente.

Prima di procedere con il punto 3 è necessario verificare quale sia il nominativo della porta seriale assegnato da Linux, quando la scheda **LX.1734** viene connessa al pc.
Per farlo, una volta connessa la scheda, digitate la riga di comando da una finestra terminale:

ls –l /dev/tty*

e verificate che appaia la device di tipo tty (ad esempio il nostro Mac ha assegnato il nominativo **tty.usbmodem431**).
Adesso procedete con il punto 3 descritto al paragrafo precedente impostando come stringa di configurazione "**-p /dev/tty.usbmodem431**", invece di "**-p COM4**" e lanciate l'applicativo come descritto al punto 4.

APPLICATIVO DATALOGGER

Questo applicativo utilizza sempre la porta USB della scheda **LX.1734** al fine di monitorare nel tempo le grandezze acquisite dai sensori ad essa collegati.
Offre inoltre la possibilità di vedere le grandezze su grafici di tipo XY in tempo reale e di salvare i dati in files, anche molto lunghi, per la registrazione delle variazioni delle grandezze.

Per questo articolo abbiamo selezionato alcuni moduli (n.d.r. leggi "programmi") pubblicati nella rivista **N.239**, il modulo temperatura ambiente **KM1734KT**, il modulo gaussmetro **LX.1734/2** ed il modulo termopila **LX.1734/4**.

Rispetto all'esempio precedente, questo programma è stato realizzato seguendo l'approccio della programmazione orientata agli oggetti, sfruttando gli oggetti di design grafico messi a disposizione da NetBeans.
Con riferimento al codice sorgente riportato in fig.15, a titolo di esempio notiamo la strutturazione del programma con una classe principale **DataLogger()**, che rende disponibili proprietà e metodi.
I principali metodi possono essere così riassunti:

OpenCOM, **ReadCOM**, **CloseCOM** = come nell'applicativo LeggiUSB.java servono per aprire, leggere e chiudere la porta seriale virtuale collegata alla porta USB.

WriteData, **PlotCOM** = servono per l'aggiornamento degli elementi grafici della finestra principale.

WriteFile = serve per la gestione della scrittura su file.

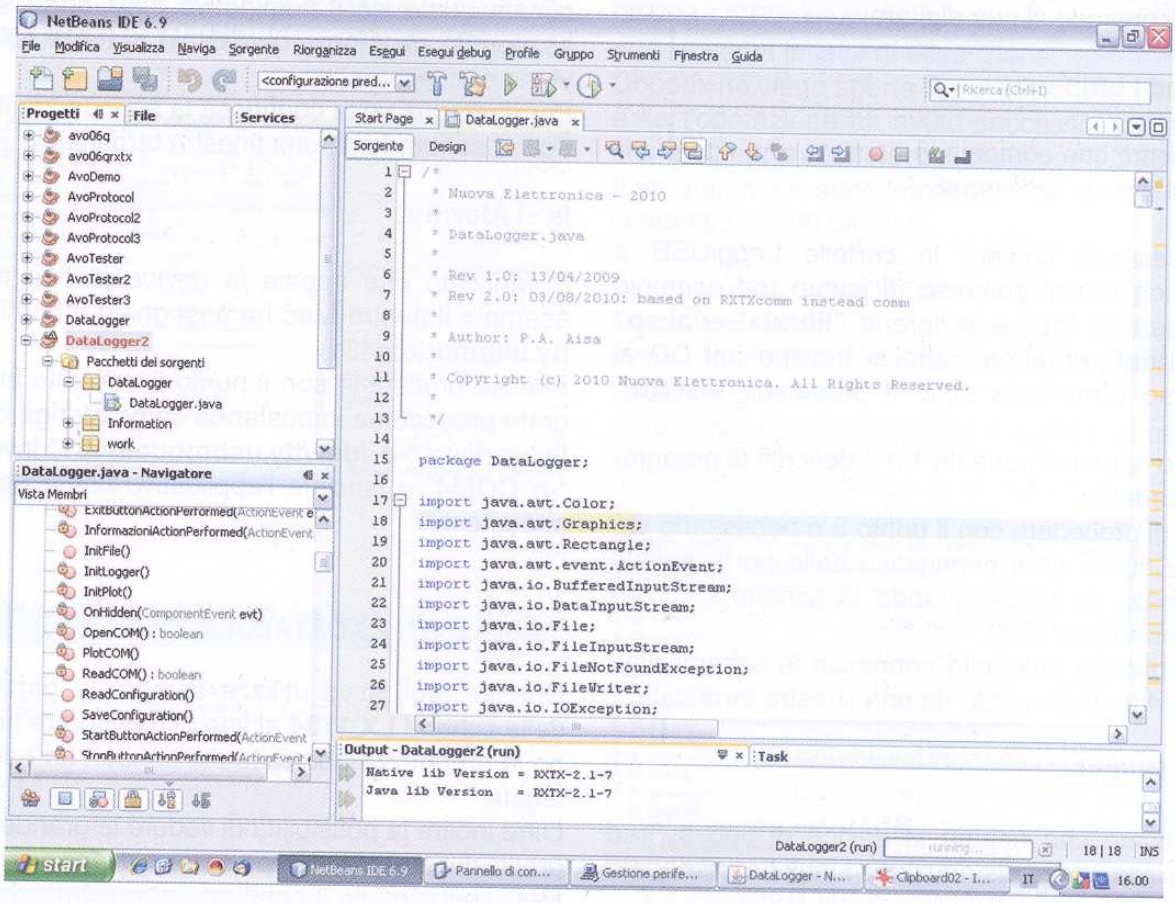


Figura 15

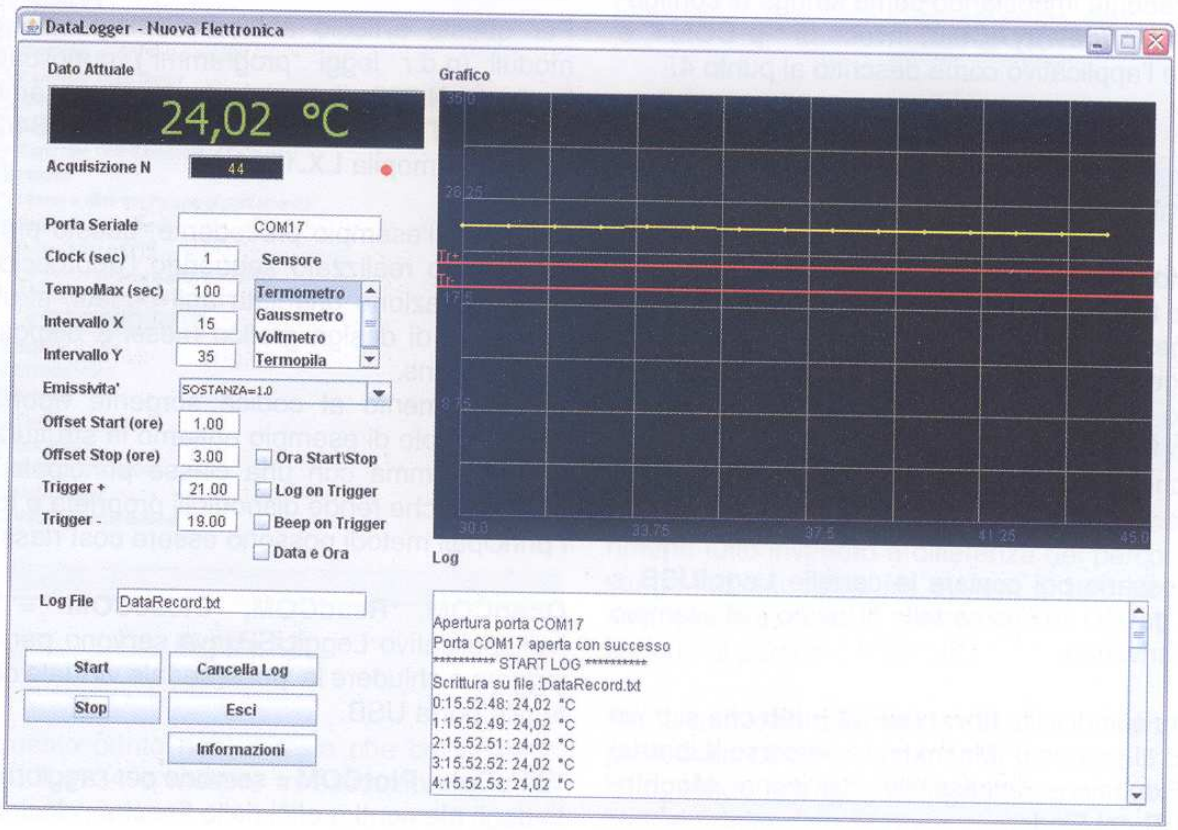


Figura 16

FUNZIONAMENTO dell'APPLICATIVO DATALOGGER

Per eseguire il programma è necessario copiare la cartella DataLogger presente sul CD in una cartella di lavoro (ad esempio C:\nE\DataLogger\) e poi fare due click sul file **DataLogger.jar**.

Come prima operazione è necessario valorizzare i campi della maschera rappresentata in fig.16.

Sensore = selezionare il sensore collegato alla scheda USB.

Porta Seriale = rappresenta il nome della porta seriale virtuale assegnata alla porta USB (vedi fig.13, per identificare su quale porta COM o tty è stata mappata la scheda USB).

Clock = rappresenta il tempo in secondi che intercorre tra due acquisizioni successive (deve essere un numero intero diverso da 0).

TempoMax = rappresenta il tempo di esecuzione della acquisizione in secondi a partire dal momento in cui si preme il tasto **Start**, se non è attivata la casella "Ora Start\Stop" (deve essere un numero intero diverso da 0).

Intervallo X = rappresenta la risoluzione orizzontale del grafico. Immettere un valore intero compreso tra 4 e 200.

Intervallo Y = rappresenta la risoluzione verticale del grafico. Immettere un valore maggiore di zero e minore del fondo scala che si intende utilizzare.

Emissività = questo menù a tendina viene usato quando è collegata una termopila e definisce il coefficiente di remissività per fornire la temperatura acquisita dalla termopila.

Offset Start (**ore**) = numero di ore rispetto all'ora corrente a cui il programma inizierà l'acquisizione, se la casella Ora Start\Stop è stata selezionata.

Offset Stop (**ore**) = numero di ore rispetto all'ora corrente a cui il programma terminerà l'acquisizione, se la casella Ora Start\Stop è stata selezionata.

Ora Start\Stop = se selezionata questa casella, il programma parte all'ora che si ha come somma dell'ora attuale e di quella indicata nel campo "Offset Start (ore)".

Trigger+ Trigger- = rappresentano il livello dell'asse Y sul quale sono posizionati i trigger positivo e negativo. Se la casella "Log Fuori Trigger" è selezionata, verranno acquisiti solo i valori che escono dall'intervallo definito da Trigger+ e Trigger-.

Log Fuori Trigger = se selezionato, il programma cattura solo i dati che risultano essere fuori dall'intervallo definito dai campi Trigger+ e Trigger-.

Beep on Trigger = se selezionato, il PC emette un suono per i dati che risultano essere fuori dall'intervallo definito dai campi Trigger+ e Trigger-.

Log File = in questo campo bisogna inserire il nome del file che verrà creato e che conterrà tutti i dati acquisiti.

Data e **Ora** = se selezionato, il formato dell'ora comprende anche la data.

Il tasto "Informazioni" fornisce le informazioni del programma.
A questo punto il DataLogger può essere avviato premendo il tasto **Start** e fermato premendo il tasto **Stop**.
Per cancellare la finestra di Log si può utilizzare il tasto **Cancella Log**.
Per uscire dal programma è possibile utilizzare il tasto **Esci** oppure chiudere la finestra con il tasto **X** in alto a destra.
Al termine dell'acquisizione potrete andare ad aprire il file di testo con titolo definito dal campo **Log File**, contenente i dati acquisiti nella cartella C:\nE\DataLogger.

COSTO di REALIZZAZIONE

L'**interfaccia USB** siglata **LX.1734** (vedi fig.2-3-4) pubblicata nella rivista **N.239**, compresi la scheda **KM.1734K** premontata in **SMD** e il **CDRom CDR1734J** contenente i sorgenti, i driver e gli applicativi da installare eseguibili, escluso il modulo temperatura **KM1734KT** **Euro 79,00**

Nota: *a richiesta forniamo anche il modulo temperatura ambiente* ***KM1734KT*** *(vedi fig.2)*
Euro 15,00

Attenzione: *a coloro che acquisteranno la scheda* ***LX.1734*** *corredata di tutti e 4 i kits applicativi (vedi rivista* ***N.239****) e cioè* ***conducimetro LX.1734/1, Gaussmetro LX.1734/2, rilevatore UVA/UVB LX.1734/3, Termopila LX.1734/4 + modulo KM1734KT****, praticheremo uno sconto eccezionale. Tutto soltanto a* **Euro 185,00**

Il costo è comprensivo di **IVA**, ma **non** delle spese postali di spedizione a domicilio.