Dipartimento di Elettronica
Università di Pavia

Introduzione a LabVIEW

1 Acquisizione dati e controllo remoto

L'acquisizione dati riveste grande importanza in tutti i campi dove sia necessario il controllo di sistemi remoti o la memorizzazione delle informazioni provenienti da strumentazione di misura. Alcuni esempi di situazioni in cui il ricorso a queste tecniche è indispensabile o comunque vantaggioso sono:

- il controllo remoto di strumenti localizzati in aree pericolose o inaccessibili;
- l'esecuzione automatica di misure ripetitive;
- l'acquisizione, l'elaborazione e la presentazione dei risultati di una misura;
- il controllo di processi on-line;

I processi descritti necessitano di hardware e software specifici che possono essere riassunti nei seguenti punti:

- interfaccia di comunicazione sullo strumento e relativo firmware;
- interfaccia di comunicazione sul calcolatore;
- bus di comunicazione (per esempio ethernet, GPIB, RS232);
- software di controllo per il processo di acquisizione dal lato del calcolatore.

Il software *L*abVIEW™ di National Instruments consente di sviluppare in modo relativamente semplice programmi mirati al controllo automatico di processi remoti ed alla acquisizione ed elaborazione di dati e misure.

2 Che cos'è LabVIEW

*L*abVIEW™ è un **ambiente per lo sviluppo di programmi**, simile, negli effetti, ad altri ambienti di programmazione che si basano su linguaggi più noti, quali il C o il BASIC.

La tecnica di programmazione costituisce la principale differenza rispetto agli ambienti di sviluppo classici: laddove questi ultimi impiegano linguaggi testuali (**text-based**), *L*abVIEW™ ricorre invece ad un **linguaggio di programmazione grafico (G)** che consente la creazione di programmi in forma di diagrammi a blocchi. Dunque, la descrizione del compito svolto dal programma è affidato a simboli grafici piuttosto che a linee di codice.

*L*abVIEW™ esiste nelle versioni per Windows 95/NT, Macintosh, SunOS e Linux.

L'ambiente di sviluppo offerto da *L*abVIEW™ possiede:

- estese librerie di funzioni per la maggior parte delle necessità di programmazione;
- librerie specificamente realizzate per le diverse piattaforme software sopra elencate;
- librerie specificamente progettate per il controllo remoto di strumenti (per esempio via GPIB o porta seriale);
- librerie per il salvataggio, l'analisi e la presentazione dei dati;
- strumenti per il debugging e lo sviluppo dei programmi: è possibile fissare punti di interruzione (breakpoints), animare l'esecuzione di un programma per seguire il flusso dei dati e muoversi al suo interno un passo per volta.

3 Strumenti virtuali (Virtual Instruments)

I programmi sviluppati in ambiente *L*abVIEW™ sono definiti **Virtual Instruments** (o VIs, strumenti virtuali) in quanto sono in grado di emulare lo strumento reale non solo nelle operazioni eseguite ma anche nell'aspetto.

Fondamentalmente ogni VI possiede:

- un'interfaccia interattiva per l'utente;
- uno schema a blocchi che costituisce l'equivalente del codice sorgente nei linguaggi di programmazione testuali.

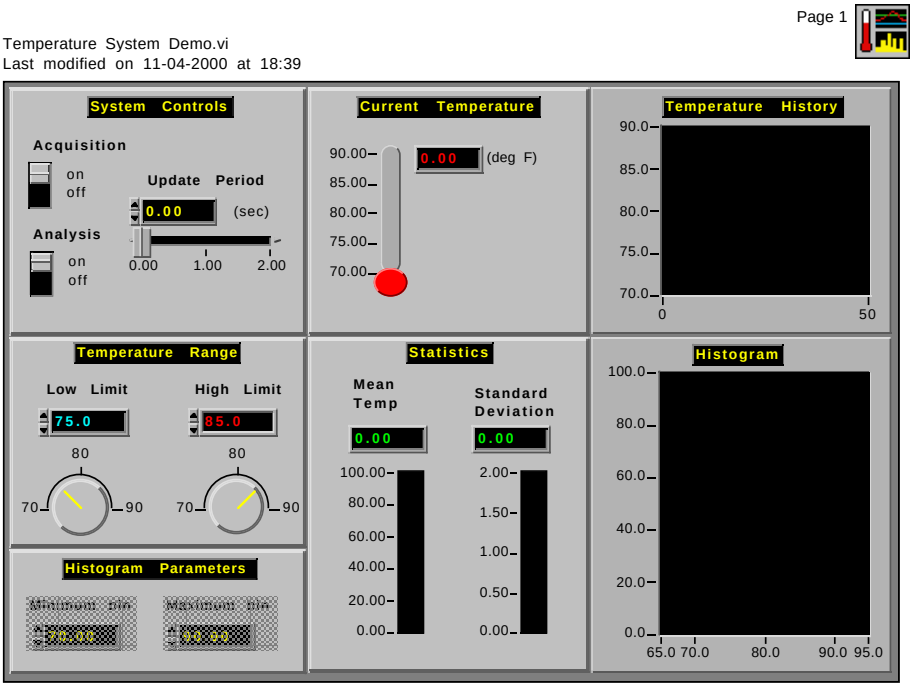

Figura 1: esempio di pannello frontale di un programma sviluppato in ambiente *L*abVIEW™ .

3.1 Pannello frontale (Front panel)

L'interfaccia interattiva per l'utente è chiamata **pannello frontale**, in quanto riproduce il pannello di uno strumento reale, sul quale si possono trovare pulsanti, manopole, display grafici, controlli ed indicatori. Mediante il pannello frontale, l'utente ha la possibilità di inserire dati secondo varie modalità e di osservare il risultato dell'operazione, più o meno complessa, compiuta dal programma sui medesimi dati. Un esempio di pannello frontale di un programma sviluppato in ambiente *L*abVIEW™ è rappresentato in figura 1.

3.2 Diagramma a blocchi (Block Diagram)

Il diagramma a blocchi costituisce il codice sorgente del VI e rappresenta una soluzione grafica ad un problema di programmazione. Esso è costituito da blocchi variamente interconnessi, con l'aggiunta di strutture (cicli for o while, strutture condizionali, sequenze) che governano il flusso dei dati. Un esempio di Diagramma a blocchi di un programma sviluppato in ambiente *L*abVIEW™ è rappresentato in figura 2.

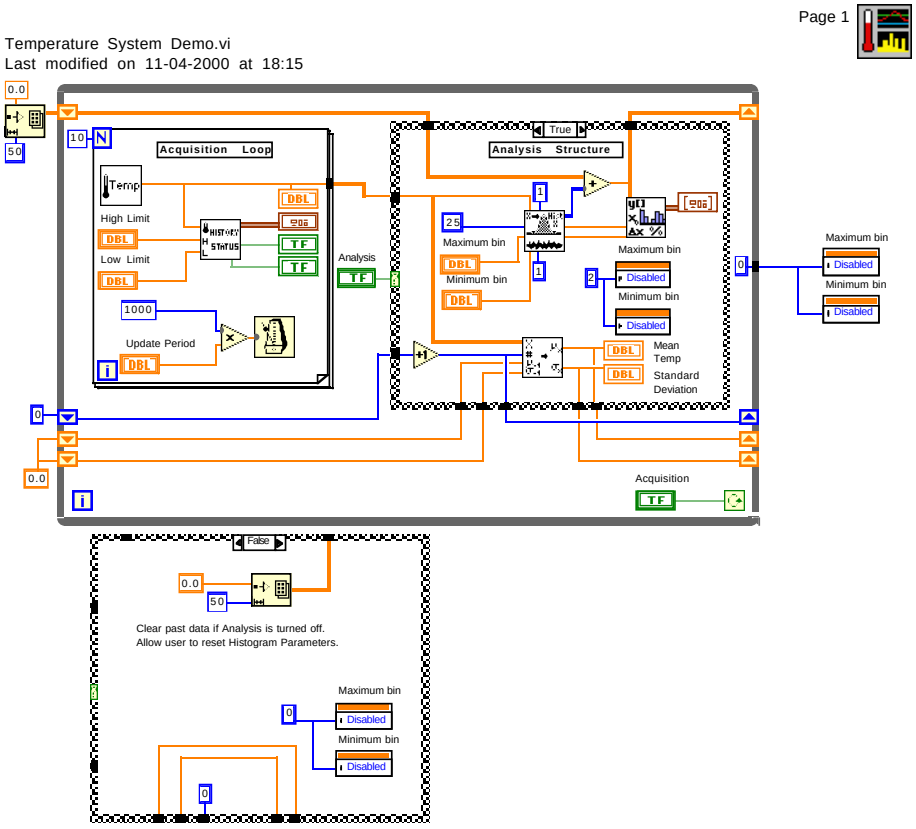

Figura 2: esempio di diagramma a blocchi di un programma sviluppato in ambiente *L*abVIEW™ .

4 Programmazione in LabVIEW

La programmazione in ambiente *L*abVIEW™ offre notevoli vantaggi in termini di strutturazione e sviluppo delle applicazioni.

4.1 Strutturazione gerarchica e modulare dei programmi

Ciascun VI può essere inserito all'interno di una struttura gerarchica o modulare. Un VI può essere utilizzato come programma di massimo livello (top-level VI) o come sottoprogramma (subVI) all'interno di un altro programma o sottoprogramma (**strutturazione gerarchica**). Inoltre ciascun VI possiede un **pannello dei connettori (connector panel)** che opera come una lista grafica di parametri in modo tale che i dati prodotti da uno strumento virtuale possano essere passati ad un altro (**strutturazione modulare**).

Sulla base di queste proprietà *L*abVIEW™ consente di suddividere un'ap-

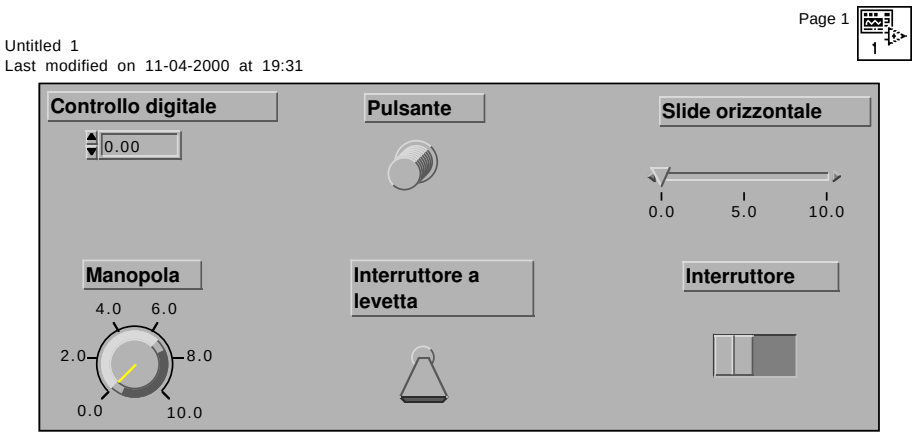

Figura 3: esempi di controlli.

plicazione in una serie di programmi più semplici, ciascuno dei quali può essere realizzato con uno strumento virtuale appropriato. La National Instruments rende disponibili sul proprio sito web (`http://www.ni.com`) programmi per l'acquisizione dati ed il controllo remoto dei più diffusi strumenti di misura.

4.2 Debugging

Dal momento che ciascun VI può essere eseguito singolarmente, la ricerca e l'eliminazione degli errori all'interno del programma **(debugging)** si presenta molto più semplice. E' inoltre possibile effettuare l'esecuzione di un programma un passo per volta, seguendo il percorso dei dati. *L*abVIEW^{TM} fornisce altri utili strumenti per il debugging, quali la possibilità di inserire breakpoints e sonde lungo i cammini che uniscono i diversi blocchi.

5 Creare uno strumento virtuale

La realizzazione di uno strumento virtuale richiede il posizionamento ed il collegamento di vari blocchi e strutture all'interno del diagramma di flusso.

5.1 Controlli ed indicatori

Un **controllo** è un oggetto che viene posizionato sul pannello frontale ed utilizzato per introdurre interattivamente un dato in uno strumento virtuale. Un **indicatore** è un oggetto che viene piazzato sul pannello frontale per visualizzare un dato in uscita da uno strumento virtuale. Controlli ed indicatori possono essere selezionati dalla tavolozza dei controlli (**Controls**) che può essere visualizzata selezionando **Windows≫Show Controls Palette**

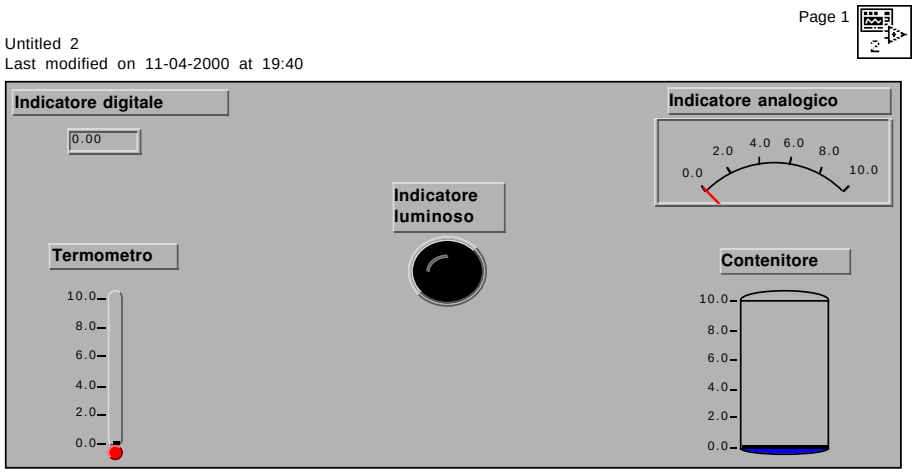

Figura 4: esempi di indicatori.

ad applicazione avviata. Esempi di controlli ed indicatori sono rappresentati rispettivamente nelle figure 3 e 4.

5.2 Funzioni

Le funzioni sono blocchi elementari nel linguaggio di programmazione G e sono equivalenti agli operatori ed alle funzioni di libreria dei linguaggi di programmazione tradizionali. Le cateorie di funzioni disponibili sono le seguenti:

- strutture di controllo (Structures): cicli for, while, strutture case, sequenze, strutture di tipo formula;
- funzioni numeriche (Numeric Functions);
- funzioni logiche (Boolean Functions);
- funzioni che operano su stringhe (String Functions);
- funzioni che operano su vettori (Array Functions);
- funzioni di raggruppamento (Cluster Functions)
- funzioni di confronto (Comparison Functions);
- funzioni di temporizzazione e gestione degli errori e delle finestre di dialogo (Time and Dialog Functions);
- funzioni per la gestione dei file (File I/O Functions);
- funzioni avanzate (Advanced Functions);

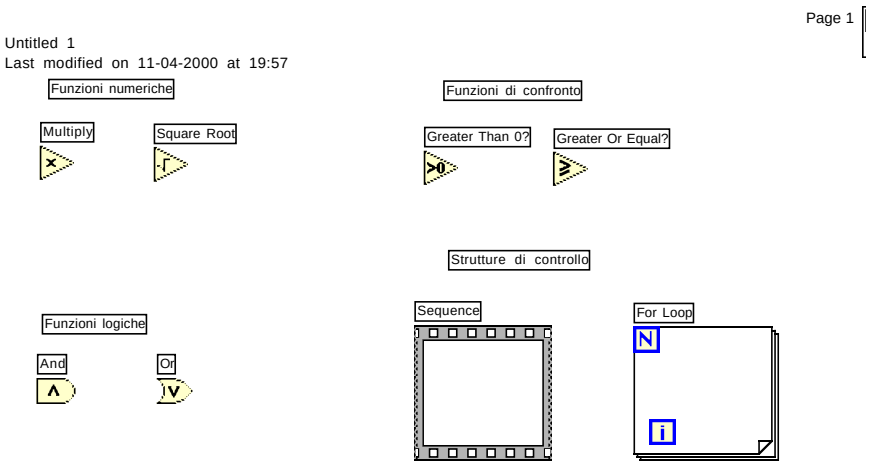

Figura 5: esempi di funzioni e strutture elementari.

- funzioni per l'acquisizione dati (DAQ);
- funzioni per la comunicazione remota con strumenti (Instrument I/O);
- funzioni per la comunicazione remota con calcolatori (Communication);
- funzioni per l'analisi dei dati (Analysis VIs).

I blocchi che rappresentano tali funzioni possono essere collegati tra loro e con controlli ed indicatori mediante opportune connessioni. Alcuni simboli di funzioni e strutture sono riportate in figura 5.

5.3 Connessioni (Wires)

Una connesione è il percorso seguito dai dati tra due nodi diversi. Il colore dei fili di connessione dipende dal tipo di dato trasportato: il colore blu è associato agli interi, il colore arancione ai numeri in virgola mobile, il verde viene utilizzato per i dati booleani ed il rosa per le stringhe. Per collegare tra loro due blocchi si utilizza il **wiring tool** selezionabile sulla tavolozza degli strumenti (**Tools**).

6 Esempio di programmazione in ambiente LabVIEW

In questa sezione viene proposta la realizzazione di un programma che calcoli le radici dell'equazione di secondo grado

$$y = ax^2 + bx + c$$

una volta che l'utente abbia introdotto i valori dei coefficienti a, b, c dell'equazione. Il programma deve informare l'utente nel caso in cui il discriminante

$$\Delta = b^2 - 4ac$$

sia minore di 0 (soluzioni complesse e coniugate).

6.1 Pannello frontale

Il pannello frontale dovrà ospitare tre terminali di controllo numerici, mediante i quali verranno introdotti i coefficienti dell'equazione Per posizionare un terminale di controllo sul pannello frontale

1. selezionare **Numeric≫Digital Control** sulla tavolozza dei controlli;
2. trascinare la relativa icona sul pannello.

In corrispondenza di questa operazione sulla finestra riservata al diagramma a blocchi comparirà un oggetto che rapresenta la variabile di ingresso associata al controllo, per default di tipo floating point a doppia precisione (double). L'operazione deve essere ripetuta per altre due volte; sarà inoltre opportuno contrassegnare ciascun terminale con il simbolo del coefficiente corrispondente. Per fare questo:

1. clickare col pulsante destro del mouse sul terminale di controllo;
2. selezionare **Show≫Label**;
3. digitare il testo voluto (in questo caso a, b o c).

Sul pannello frontale dovranno essere presenti anche due indicatori numerici (**Numeric≫Digital Indicator**, sempre sulla tavolozza dei controlli) per la visualizzazione delle soluzioni dell'equazione ed un indicatore booleano (che potrà dunque assumere solo due valori, ottenuto per esempio selezionando **Boolean≫Round LED**) che segnalerà all'utente l'assenza di soluzioni in campo reale per l'equazione in esame. Come in precedenza, queste operazioni faranno comparire nella finestra relativa al diagramma a blocchi tre oggetti che rappresentano altrettante variabili di uscita, di tipo double nel caso degli indicatori numerici, di tipo booleano nel caso del terzo indicatore. Ora il pannello di controllo è completo ed avrà l'aspetto rappresentato in figura 6.

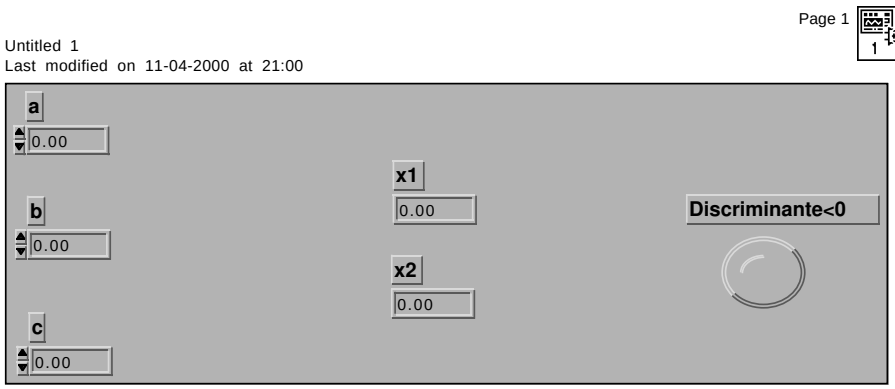

Figura 6: Pannello frontale del VI per il calcolo delle radici dell'equazione di secondo grado.

6.2 Diagramma a blocchi

I simboli già presenti sul diagramma a blocchi devono essere ora opportunamente utilizzati per produrre il risultato desiderato. A tale scopo saranno necessari blocchi che siano in grado di compiere operazioni matematiche e logiche e strutture più complesse che controllino il flusso dei dati e delle istruzioni. Il programma che si intende realizzare dovrà produrre lo stesso risultato delle seguenti linee di pseudocodice in linguaggio testuale:

```
Delta=b^2-4ac
if Delta>=0
        x1=(-b+sqrt(Delta))/(2a)
        x2=(-b-sqrt(Delta))/(2a)
        led spento
else
        x1=0
        x2=0
        led acceso
end if
```

I blocchi che realizzano le operazioni matematiche richieste (prodotto, divisione, sottrazione, somma, prodotto per -1, radice quadrata) possono essere individuati selezionando **Numeric** nella tavolozza delle funzioni. Un blocco di confronto dovrà essere impiegato per stabilire se $\Delta > 0$ (selezionare **Comparison≫Greater Or Equal To?** sempre nella tavolozza delle funzioni). Sarà inoltre necessario utilizzare alcune costanti di tipo numerico (**Numeric≫Numeric Constant**) e di tipo booleano (**Boolean≫Boolean Constant**) per il calcolo delle radici dell'equazione e per l'azionamento dell'indicatore luminoso.

Si osservi che i vari blocchi possiedono uno o più ingressi ed una uscita. Ingressi ed uscite non sono necessariamente dello stesso tipo; per esempio il blocco di confronto **Greater Than 0** ha un ingresso numerico ed una uscita logica.

Una volta posizionati sulla finestra del diagramma di flusso, i blocchi dovranno essere opportunamente collegati mediante il **wiring tool** (dall'aspetto di un rocchetto di filo) che si trova sulla tavolozza degli strumenti (**Tools**).

Il flusso del programma dovrà essere in qualche modo controllato dalla condizione posta sul valore del discriminante ($\Delta \geq 0$?). A questo scopo può essere utilizzata la struttura **case** (**Structure**≫**Case**) che possiede due o più sottodiagrammi, dei quali uno solo viene eseguito quando la struttura stessa viene eseguita; la selezione viene effettuata sulla base del valore assunto da una variabile di controllo logica o scalare. Per default la struttura case offre la possibilità di scegliere tra due alternative (**true/false**) ed è quindi governata da una variabile di tipo booleano. Nel caso in esame il valore della variabile di controllo coincide col valore della variabile di uscita del blocco di confronto (vedi figura 7).

Se tale confronto fornisce come risultato **true**, i risultati delle operazioni matematiche eseguite sui tre coefficienti a, b e c dovranno essere passati agli indicatori `x1` e `x2`, mentre il led rimarrà spento. In questo caso all'interno

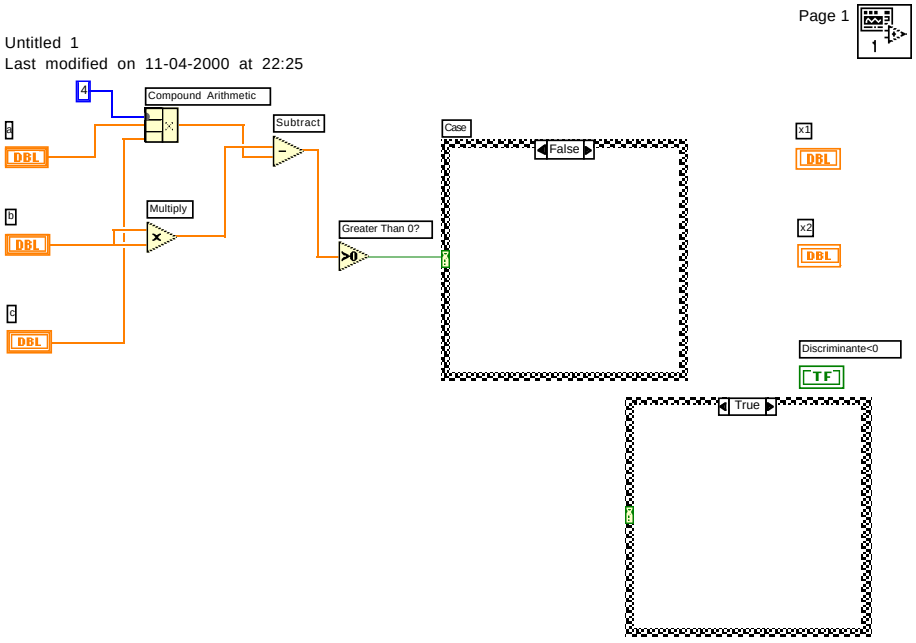

Figura 7: controllo della struttura case mediante il blocco condizionale $\Delta \geq 0$?.

della struttura case con intestazione true passeranno le connessioni verso gli indicatori numerici ed una costante booleana *false* posta all'interno della struttura verrà connessa all'indicatore luminoso.

Se il risultato del confronto è **false** agli indicatori verrà passato il valore 0 mentre il led si accenderà. In questo caso una costante 0 posta all'interno della struttura case con intestazione false verrà connessa ad entrambi gli indicatori numerici mentre una costante booleana *true* verrà collegata all'indicatore luminoso.