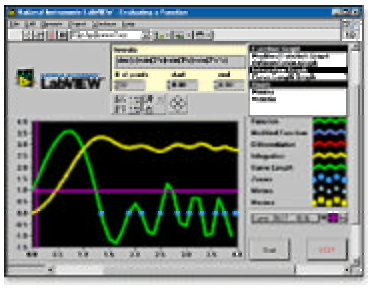
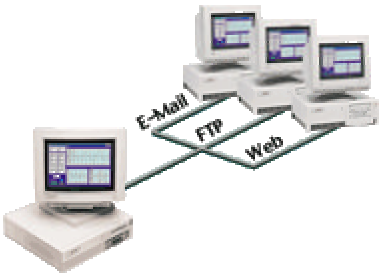
CONSIGLIO NAZIONALE DELLE RICERCHE

CENTRO STUDI DINAMICA DEI FLUIDI

FONDAMENTI DI ACQUISIZIONE DATI
E
INTRODUZIONE AL SISTEMA LabVIEWä

R. Malvano **P.G. Spazzini**





NATIONAL INSTRUMENTS

LEZIONI
DEL CORSO DI AERODINAMICA SPERIMENTALE

Introduzione

La prima versione del programma LabVIEW (LabVIEW significa: **Lab**oratory **V**irtual **I**nstrument **E**ngineering **W**orkbench) risale al 1987 e, fino ad oggi, ha comportato un impegno pari a circa due secoli uomo.

E' questo un dato statistico che più di ogni altro fornisce la dimensione dello sforzo che la National Instruments ha dedicato a questo prodotto che si è imposto come vero e proprio standard di riferimento per la gestione e la elaborazione di una sessione di acquisizione dati.

Penso che queste ore dedicate a venire a contatto con uno strumento così potente e versatile assumano una valenza che va al di là degli scopi che si prefigge un corso di aerodinamica sperimentale; infatti tra coloro che sono qui presenti non molti si occuperanno, a livello professionale, di aerodinamica sperimentale ma più di uno di voi verrà a contatto con il problema di *monitorare* un fenomeno facendo misure in laboratorio o sul campo.

La filosofia che ha ispirato il team che ha studiato e sviluppato LabVIEW si può schematizzare in due linee guida essenziali:

1. facilità nell'individuazione del flusso dei dati,
2. facilità nell'individuazione delle strutture che costituiscono il programma.

Per realizzare questi due obiettivi si è inventato il linguaggio G (G è l'iniziale di Graphical Language): il programmatore ha a propria disposizione un'interfaccia ad alto livello di facile utilizzo.

Una importante peculiarità di questo linguaggio è dovuta al fatto che LabVIEW traduce in C le istruzione che l'utente definisce durante la fase di stesura del codice.
Pertanto l'utente LabVIEW non opera con un linguaggio **interpretato** come il BASIC che, in fase di esecuzione, traduce ogni singola istruzione in linguaggio macchina ma opera con un linguaggio **compilato** come il FORTRAN che prevede una fase di compilazione durante la quale si creano i vari oggetti ed una successiva fase di LINK che collega tra loro i vari oggetti creando in questo modo il programma eseguibile.
Tutte queste operazioni avvengono in modo assolutamente trasparente; pertanto il programmatore ha la sensazione di operare con un sistema interpretato (immediata operatività in fase di correzione del codice e grande facilità nelle operazioni di debug del programma) ma ha a disposizione i grandi vantaggi in termini di velocità di esecuzione del codice che può fornire solo un linguaggio compilato.

Per poter operare LabVIEW richiede un sistema operativo che sia in grado di fornire un'interfaccia grafica molto potente e sofisticata.
E' questo il motivo per cui, nelle fasi iniziali, LabVIEW è stato sviluppato solamente per il mondo Macintosh; con l'avvento di Windows 3.1 e successivamente di Windows 95 è stato possibile fornire LabVIEW anche agli utenti del mondo Microsoft che, forse ingiustamente, aveva di fatto conquistato il mercato con un prodotto più scadente di quello offerto da Macintosh.
Questa è una prova evidente che non sempre il mercato seleziona i prodotti migliori ma chi riesce a controllare il mercato riesce anche ad imporre standard peggiori; e una volta che si è creato uno standard è il mercato che si deve adeguare a tale standard.

Dopo questa breve introduzione in cui si è cercato di tracciare a grandi linee la storia di LabVIEW, vediamo in sintesi gli scopi che si prefiggono queste poche ore di lezione.

Per motivi di tempo non riusciremo ad acquisire se non i fondamenti di questa tecnica di programmazione (occorrerebbero parecchie ore di laboratorio informatico che purtroppo non abbiamo a disposizione); cercheremo tuttavia di vedere insieme le potenzialità che questo linguaggio offre e, mettendo insieme le varie conoscenza acquisite, cercheremo di simulare una sessione di acquisizione dati.

Infine analizzeremo i parametri che possono influenzare in modo significativo una misura quali la frequenza di acquisizione, il corretto dimensionamento del buffer dati, l'esigenza di conciliare le alte velocità di acquisizione con quella di visualizzare in tempo reale l'andamento di un fenomeno senza che ciò comporti perdita di dati.

Attualmente LabVIEW opera sulle seguenti piattaforme:
1. Macintosh e cloni,
2. Sun SPARCstation,
3. HP 9000/700,
4. PC operanti sotto Microsoft Windows 3.1/95/NT

La possibilità che l'utente ha, mediante LabVIEW ed il computer, di crearsi il proprio strumento di misura personale ha portato ad una enorme diffusione di questo prodotto; il Lawrence Livermore Laboratory, il Jet Propulsion Laboratory, la NASA, il CERN di Ginevra sono esempi illustri di utilizzatori di questo moderno linguaggio di programmazione.

LabVIEW è adottato sullo space shuttle, sui sottomarini della marina militare americana, sulle piattaforme petrolifere che operano nei mari del nord.

Anche nelle piccole realtà industriali, molto diffuse in Italia, è molto frequente l'esigenza di tenere sotto controllo un fenomeno, ad esempio misurare la temperatura di un forno, di un congelatore, di una serra e, contemporaneamente, di controllarne il valore.

Data la grande diffusione del personal computer, LabVIEW è il catalizzatore ideale che lega il PC al fenomeno da misurare, ne consente la misura e ne controlla il processo.

1. Fondamenti della programmazione in LabVIEW

Un programma scritto in LabVIEW non richiede la stesura di una serie di istruzioni utilizzando un editor come si è soliti fare se si utilizza un linguaggio di programmazione tradizionale.

L'interfaccia che consente la stesura del programma è di tipo grafico e fa uso di due piani di lavoro distinti:

1. Il **front panel** (pannello frontale)

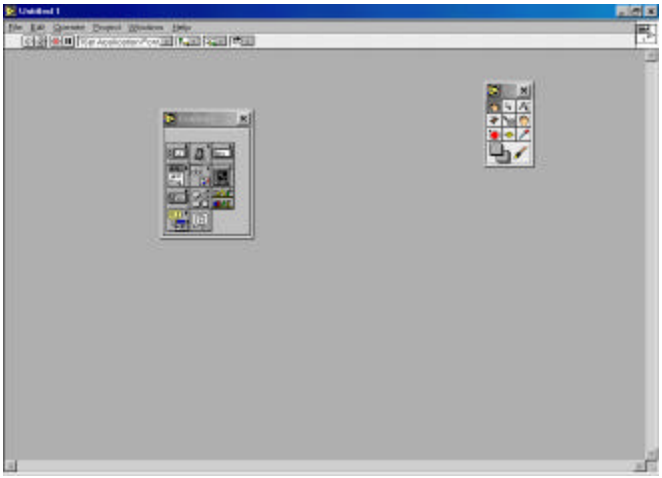


Fig. 1.1 Front Panel.

2. Il **block diagram** (diagramma a blocchi)

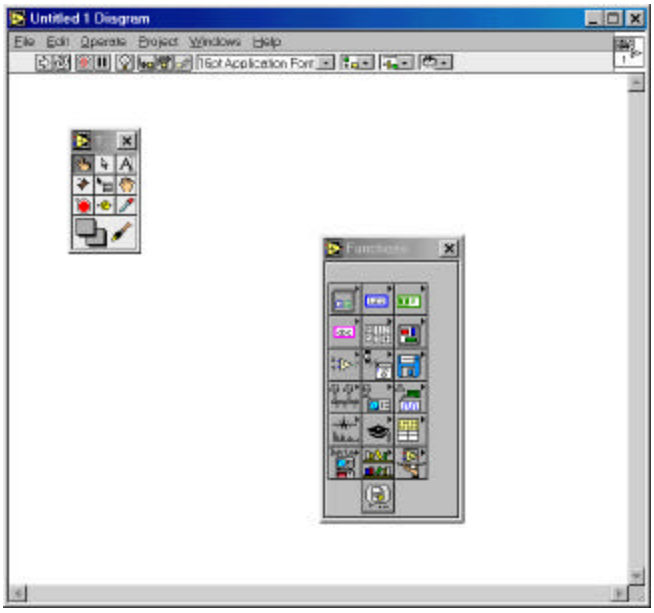


Fig. 2.1 Block Diagram.

Esaminiamo ora in dettaglio il significato di ognuno di questi due piani di lavoro.

Front Panel

Il front panel è l'interfaccia grafica che permette di definire ed introdurre tutte le grandezze in ingresso (input del problema) e le grandezze in uscita (valori delle misure, risultati dei calcoli e delle varie elaborazioni che il programma ha effettuato durante la fase di esecuzione).

L'aspetto che il pannello frontale assume una volta definite la grandezze sopra elencate è quello tipico di uno strumento di misura: per questo il programma prende il nome di strumento virtuale (Fig. 3.1).

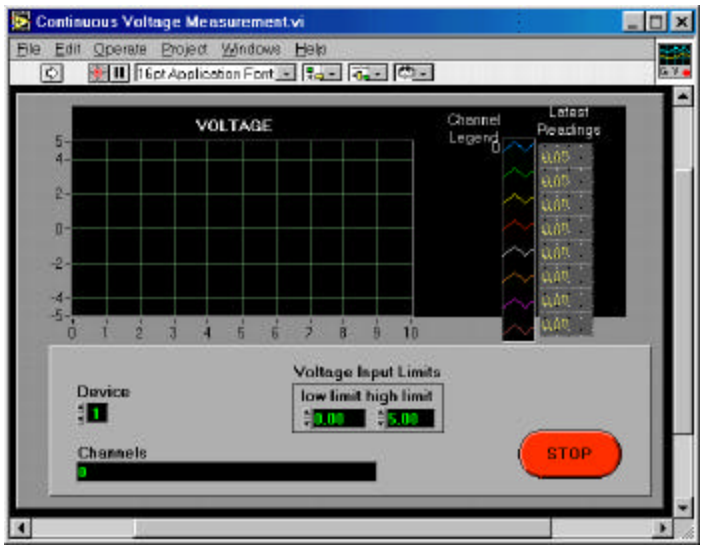


Fig. 3.1 Esempio di strumento virtuale.

Le grandezze Booleane appaiono sotto forma di pulsanti di varia foggia. Il valore dei parametri in ingresso viene visualizzato in forme differenti quali caselle numeriche, potenziometri a cursore, potenziometri ad ago rotante, ecc. Tutti questi oggetti prendono il nome di **controlli.**

Analogamente i risultati possono essere visualizzati mediante **indicatori** che, anche in questo caso, assumono aspetti differenti: lampadine accese o spente per indicare il valore assunto da una variabile booleana, indicatori ad ago od indicatori digitali per visualizzare il valore assunto dalle variabili intere o reali.

In conclusione sul pannello frontale si costruisce l'interfaccia utente che assume l'aspetto di un vero e proprio strumento (virtuale).

Block Diagram:

il block diagram è lo strumento grafico che consente di scrivere il codice di calcolo vero e proprio.

Si presenta sotto una forma che può ricordare a grandi linee lo schema di un circuito elettrico. I vari elementi di questo circuito appaiono sotto forma di icone ed effettuano le varie operazioni richieste dalla logica del programma.
Tali elementi sono collegati tra loro da fili che sono percorsi dal flusso dei dati.

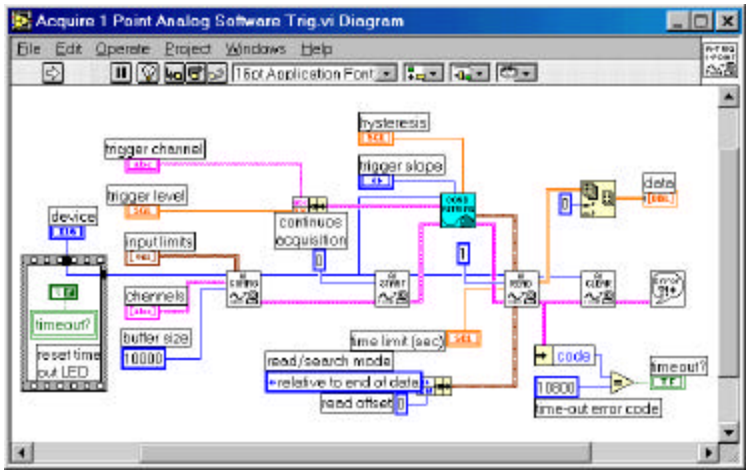


Fig. 4.1 Esempio di block diagram.

In base a quanto fin qui detto scriviamo un programma in grado di effettuare una sola banalissima operazione: convertire le Lire in Euro.

Prima fase:

individuiamo innanzitutto le variabili in ingresso; esse sono l'importo in Lire di cui vogliamo conoscere il corrispettivo in Euro ed il tasso di cambio Lira/Euro. In uscita avremo il valore risultante dal calcolo.

Per poter operare è necessario che siano presenti sul pannello frontale il **Tools Palette** ed il **Control Palette**, che si attivano andando sulla barra di controllo, scegliendo **Windows** e successivamente **Show Tools Palette** e **Show Controls Palette**.

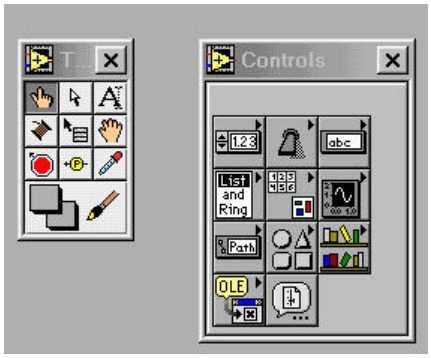


Fig. 5.1 Tools Palette e Controls Palette

Sul Tools Palette rendere attiva la **freccia**.

Posizionare la freccia sulla funzione **Numeric** del Control Palette, trascinare un digital control sul Front Panel e digitare il nome della prima variabile (Lire); ripetere la stessa operazione per la seconda variabile (Tasso di Cambio). Infine posizionare la freccia sulla funzione **Numeric** del Control Palette, trascinare un digital indicator sul Front Panel e digitare il nome della terza variabile (Quoziente).

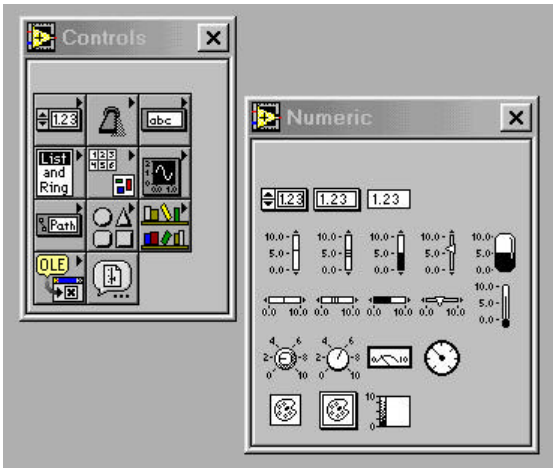


Fig. 6.1 Sottomenu Numeric del Controls Tools

Pertanto, in base a quanto detto, scegliendo due **digital contro**l avremo creato sul pannello frontale due controlli che ci consentiranno di assegnare il valore alle due grandezze in ingresso e, scegliendo un **digital indicator**, avremo creato un indicatore che consentirà di visualizzare il risultato ottenuto.

E' importante osservare che è possibile assegnare ad ogni grandezza (sia che si tratti di un controllo o di un indicatore) un nome. Come vedremo meglio in seguito questo è molto utile e conviene sempre farlo per agevolare il riconoscimento delle variabili.

Seconda fase:

Passiamo ora al block diagram. Su questo pannello, sul quale prima che fossero state definite le grandezze sul front panel non vi era nulla, ora appaiono tre simboli che rappresentano tra numeri reali il cui significato è chiaro grazie alle **label** (etichette) scritte in precedenza. Già da questo risulta evidente che è utile assegnare una label; ciò aiuta ad individuare la corrispondenza delle varie grandezze definite nel front panel.

E' utile osservare che sul block diagram gli indicatori appaiono con bordi sottili mentre i controlli hanno bordi spessi.

Ora siamo finalmente in grado di scrivere il codice di calcolo che, in questo caso, è banalissimo.

Per poter operare è necessario che siano presenti sul block diagram **i Tools Palette** ed il **Functions Palette**, che si attivano andando sulla barra di controllo, scegliendo **Windows** e successivamente **Show Tools Palette** e **Show Functions Palette**.

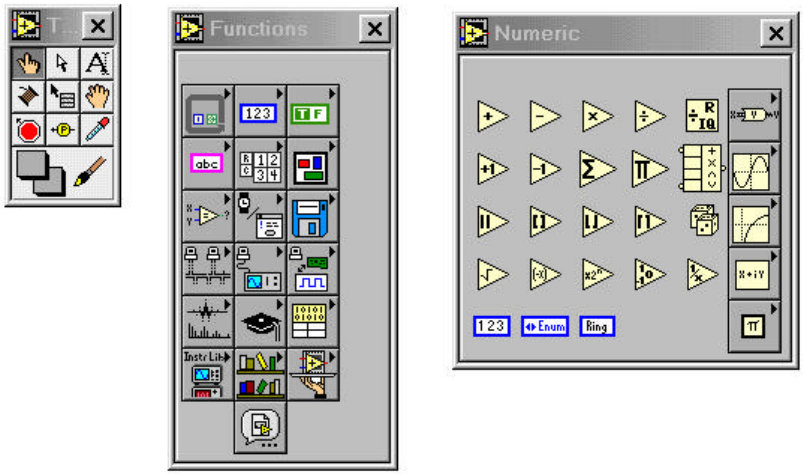


Fig. 7.1 Tools palette, Functions Palette e Numeric sub-palette

Sul Tools Palette rendere attiva la **freccia**.

Posizionare la freccia su **Numeric** del Functions Palette e trascinare l'operatore **divide** sul block diagram.

La **Functions Palette** mette a disposizione del programmatore moltissime funzioni di vario tipo: matematiche, logiche, statistiche, funzioni che consentono l'elaborazione dei risultati, la loro scrittura su file, l'acquisizione di dati, la gestione di un colloquio con periferiche di vario tipo, ecc.

Tutte queste funzioni si presentano sotto forma di icona.

Ad ogni icona sono associate due famiglie di connettori: alla prima famiglia si collegano le grandezze in ingresso, alla seconda quelle in uscita; nel caso che stiamo considerando e cioè l'operazione di divisione vi sono tre connettori dei quali uno è relativo al dividendo, l'altro al divisore ed il terzo al quoziente.

Sul Tools Palette rendere attivo il **rocchetto** di filo e collegare la variabile di controllo delle Lire al dividendo, la variabile di controllo del tasso di cambio al divisore ed il quoziente alla variabile indicatore del valore espresso in Euro.

Se avremo effettuato i collegamenti in modo corretto i fili presenteranno un andamento continuo; in caso contrario risulterà evidente che sono interrotti.

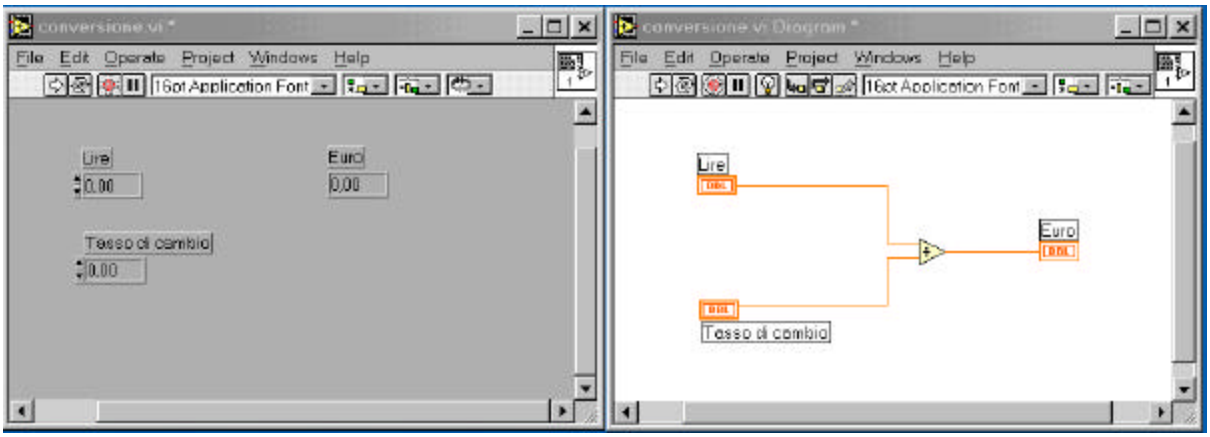


Fig. 8.1 Programma di Conversione Lire/Euro

A questo punto è possibile eseguire immediatamente il programma scritto sopra.

Sul pannello frontale occorre impostare il valore corretto del tasso di cambio e l'importo in Lire di cui si vuol conoscere il corrispondente importo in Euro.

A tal fine posizionarsi sul **Tools Palette** e scegliere il simbolo in alto a sinistra che simboleggia una mano con il dito indice alzato.

Posizionarsi all'interno della casella Tasso di cambio e digitare il valore corretto; ripetere l'operazione per il valore in Lire.

Spostare il cursore sul simbolo raffigurante una freccia diretta verso destra e premere la suddetta freccia; in questo modo si esegue il programma ed in corrispondenza della casella Euro appare il valore convertito.

Se si desidera che il programma conservi i valori che appaiono dopo questa esecuzione si opera come segue:

1. spostarsi sul menu **operate**
2. scegliere l'opzione M**ake Current Values Defaul**
3. salvare il programma con l'opzione **save** di **file**

Un'ulteriore strumento, che è utile conoscere in quanto consente di effettuare modifiche e correzioni con grande facilità, consiste nella possibilità di eseguire il programma evidenziando il flusso dei dati (**Highlight Execution**).

Per attivare questa funzione selezionare sul block diagram la lampadina.
Premendo il pulsante che inizia l'esecuzione si osserva che un cursore si muove seguendo il flusso dei dati ed evidenzia il valore che una determinata variabile assume durante l'esecuzione.

E' anche possibile eseguire il programma step by step; prima bisogna attivare la lampadina e successivamente premere il tasto Step-Into.

2. Principali funzioni di LabVIEW

Esempi d'uso

Nel corso di questi appunti verranno riportati alcuni esempi di codici scritti in LabVIEW; ognuno di essi utilizza vari tipi di istruzioni. Pertanto è indispensabile, prima di procedere oltre, fornire le informazioni essenziali che ne consentano un uso elementare.

E' evidente che LabVIEW, essendo un linguaggio potente e sofisticato, dispone di una serie di comandi la cui complessità è direttamente proporzionale alle potenzialità che mette a disposizione dell'utente; tuttavia dovremo limitarci ovviamente ad alcune considerazioni di base.

Matrici

Una matrice è una raccolta di elementi dello stesso tipo; essa può avere una o più dimensioni fino ad un massimo di 2^{31} elementi per dimensione (memoria RAM permettendo). L'elemento di una matrice può essere costituito vari tipi di grandezze (ad esempio un numero intero, un numero reale, un insieme di caratteri, una variabile booleana).
Per poter utilizzare una matrice è necessario, analogamente a quanto viene fatto nella maggior parte dei linguaggi (vedi ad esempio il Basic ed il Fortran) definire la matrice.
Gli elementi di una matrice sono individuati mediante un indice il cui valore varia tra 0 ed N-1, ove N è il numero totale degli elementi della matrice stessa.

Impariamo ora a creare una matrice di controlli, formata da cinque numeri, che scrive su di una matrice di indicatori.
La prima operazione consiste nel definire sul control panel due matrici; tale operazione può essere effettuata in due fasi successive:

1. Si seleziona e si trascina sul control panel la funzione **Array;** a tale funzione si accede mediante il **Controls Palette** selezionando il sottogruppo **Array & Cluster**.

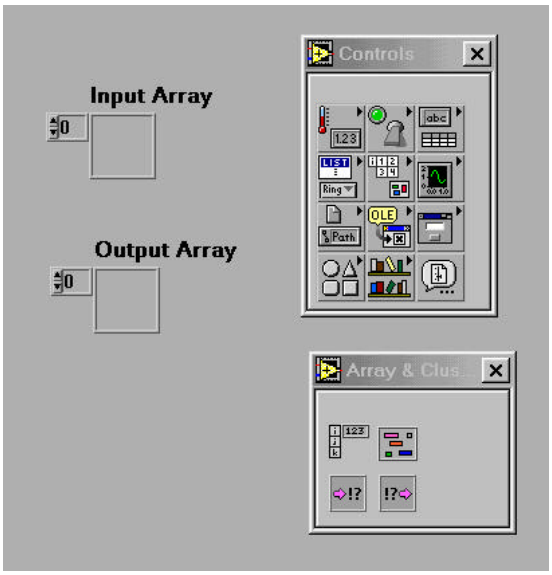


Fig. 1.2 Creazione di una matrice (fase 1)

2. Si definisce il tipo di array e la sua funzione; a tal fine, mediante il **Controls Palette**, si seleziona il tipo di variabile (Numerica, booleana, sequenza di caratteri) ed il relativo tipo (controllo od indicatore) e la si colloca fisicamente all'interno dell'oggetto creato in precedenza. La Fig.2.2 mostra questa seconda fase nel caso di una array costituito da variabili stringa.

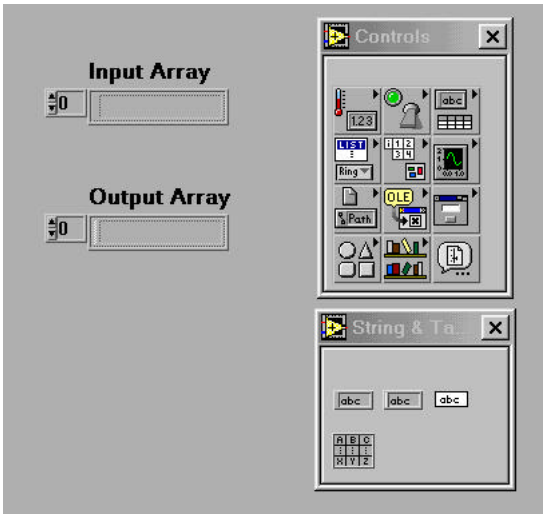


Fig. 2.2 Creazione di una matrice (fase 2)

3. Si accede al block diagram e si collega l'Input Array con l'Output Array (Fig. 3.2).

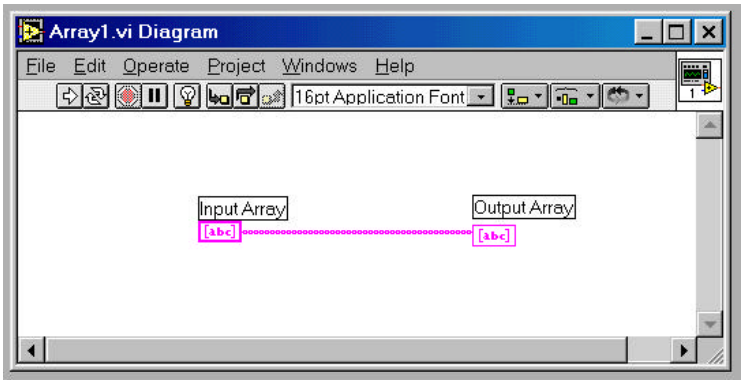


Fig. 3.2 Creazione di una matrice (fase 3)

4. Si definiscono i vari elementi della matrice e si esegue il programma così completato:il programma trasferisce i dati in ingresso nella matrice di output (Fig. 4.2). Per accedere da un elemento al successivo posizionare il dito della mano sul contatore della matrice e scegliere l'indice che corrisponde all'elemento che si desidera visualizzare. Per vedere più di un elemento per volta all'interno della matrice è sufficiente posizionare la freccia sullo spigolo in basso a destra della matrice stessa e trascinarlo verso il basso; in questo modo si possono visualizzare più elementi contemporaneamente (Fig. 5.2).

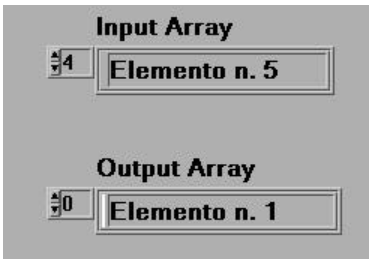


Fig. 4.2 Creazione di una matrice (fase 4)

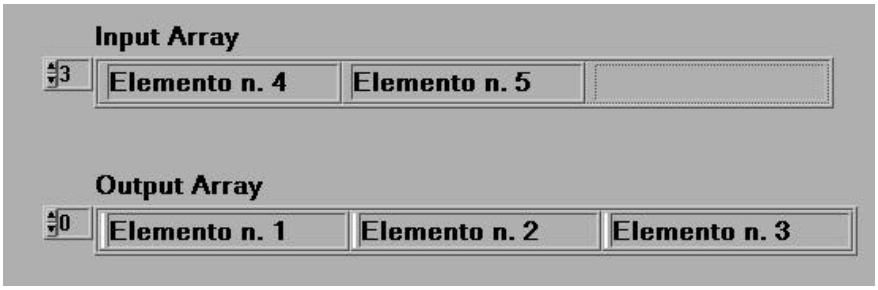


Fig. 5.2 Creazione di una matrice (fase 5)

Analogamente è possibile creare un **array a più dimensioni**. Per fare ciò è sufficiente definire il numero di dimensioni dell'array; tale operazione si effettua trascinando verso il basso l'angolo inferiore destro del contatore dell'array. Trascinandolo ulteriormente verso il basso si aggiunge una ulteriore dimensione. E' evidente che questa operazione deve essere effettuata sia sull'array che contiene la serie di controlli numerici che sull'array che contiene la serie di indicatori numerici. A questo punto non resta che introdurre i dati e eseguire il programma come fatto in precedenza nel caso di un array avente una sola dimensione. In Fig. 6.2 è riportato il control panel di un caso semplice di array a due dimensioni ed in Fig. 7.2 il relativo block diagram.

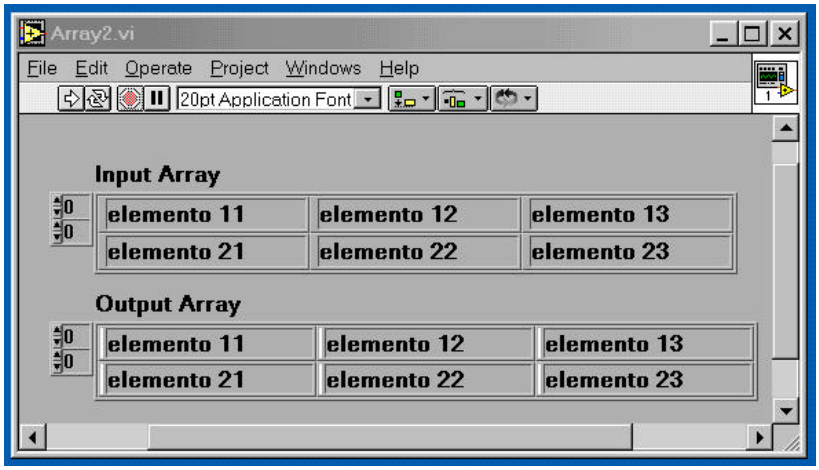


Fig. 6.2 Esempio di matrice a due dimensioni (Contrlo Panel)

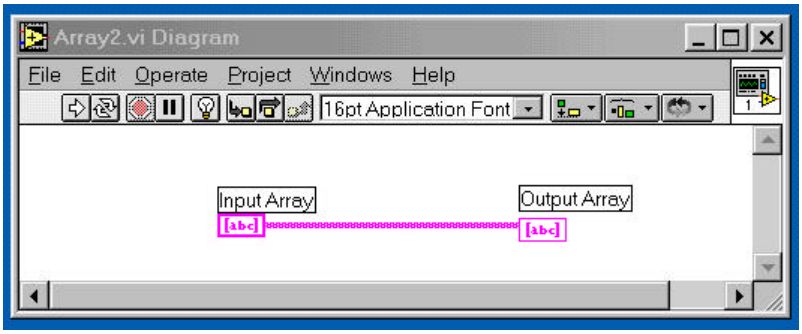


Fig. 7.2 Esempio di matrice a due dimensioni (Block Diagram)

Tramite il cammino **Functions** e **Array** si accede ad un insieme di funzioni che consentono di effettuare tutte le più importanti operazioni su di una matrice quali ricavare il numero di elementi di una matrice, estrarre un generico elemento di una matrice, trasporre la matrice ecc.
Alcuni esempi di queste funzioni sono illustrati in Fig. 8.2.

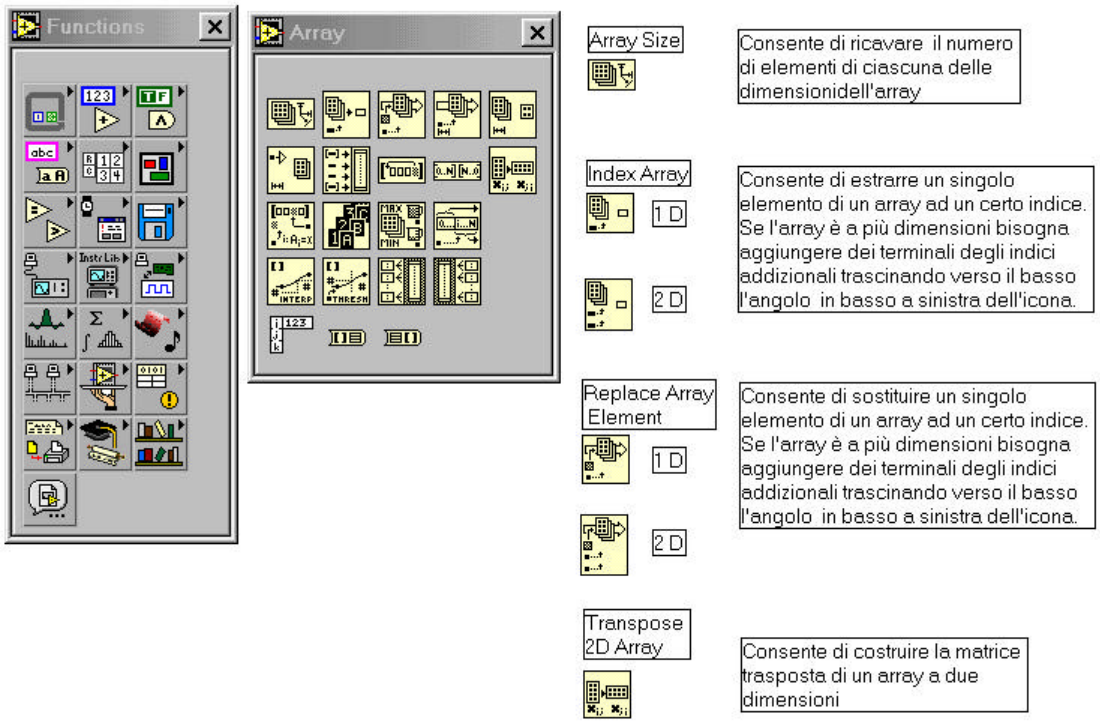


Fig. 8.2 Esempi di funzioni che operano sugli array

Cicli For e cicli While

Qualunque linguaggio di programmazione prevede una struttura che consenta la ripetizione di una sezione del codice. LabVIEW offre due strutture che consentono questo tipo di operazione: **For Loop** e **While Loop**. La prima esegue la parte di codice annidata al suo interno per un numero definito di volte, mentre la seconda la esegue fino a che una specifica condizione non è più vera. Si accede ad entrambi i loop tramite il cammino **Functions** e **Structures** (Fig. 9.2).

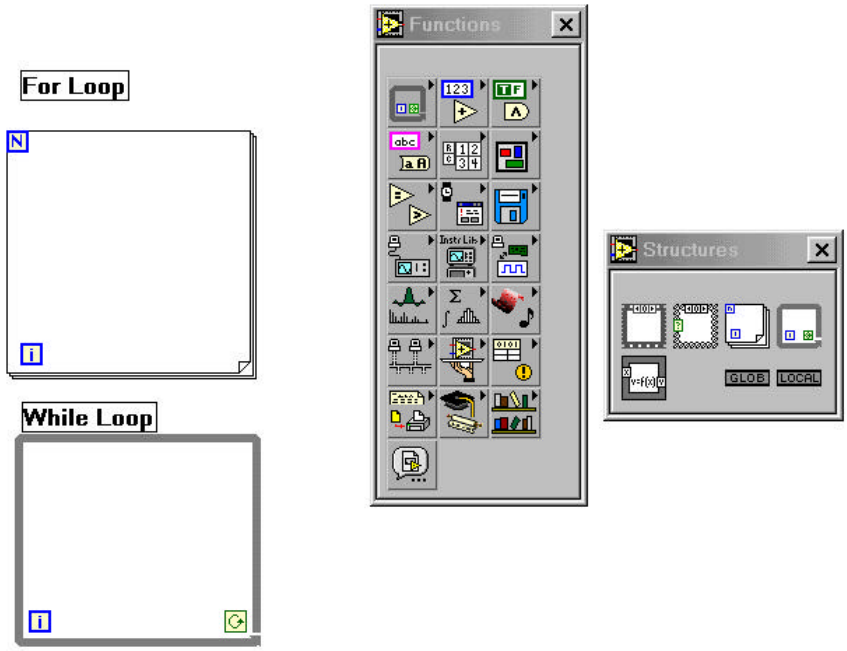


Fig. 9.2 Cicli For Loop e While Loop

La struttura For Loop esegue il codice contenuto all'interno del suo perimetro un numero di volte pari al valore assegnato alla costante N.

Si procede come segue:

1. si trascina la struttura sul block diagram e la si dimensiona in modo che possa contenere al suo interno la parte di codice che deve essere eseguita N volte trascinandone lo spigolo in basso a destra,
2. si definisce il valore di N collegandolo ad un controllo o ad una costante (dato che il contatore **i** parte da 0, per eseguire N volte il ciclo il valore della costante N deve essere uguele ad N-1),
3. si scrive la parte di codice che deve essere eseguita N volte all'interno del perimetro del For Loop.

Una struttura For Loop è equivalente al seguente metacodice:

```
For i = 0 to N-1
        Esecuzione del subcodice
End For
```

A titolo di esempio costruiamo un semplice codice che effettui il quadrato della variabile **i**, ne visualizzi il risultato ed attenda 2 secondi; questa sequenza di operazioni deve essere eseguita 10 volte. Questo esempio ci consente di utilizzare una nuova funzione, molto importante nel caso in cui si debba temporizzare una sequenza di operazioni, e cioè la funzione **Wait** a cui si accede tramite il cammino **Functions** e **Time & Dialog**; tale funzione richiede come parametro di ingresso il valore del tempo di attesa espresso in millisecondi (Fig. 10.2).

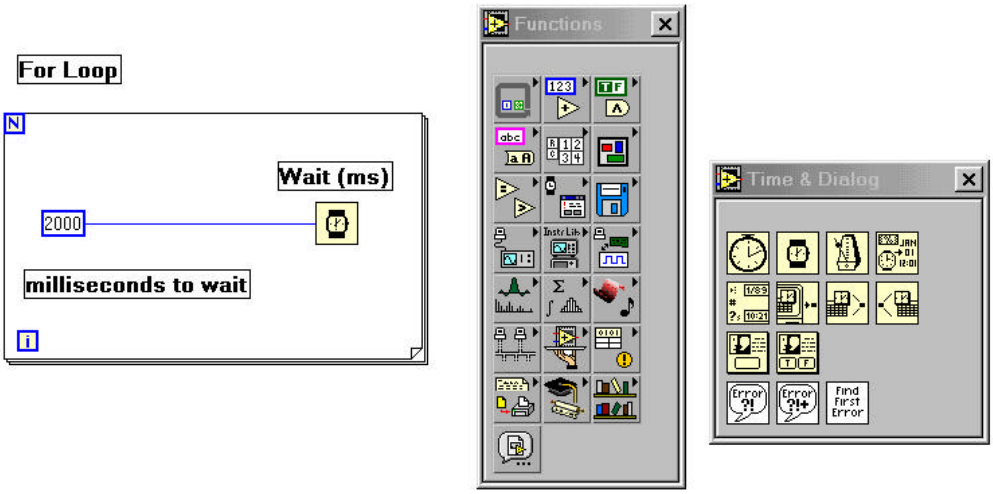


Fig. 10.2 Esempio di uso della funzione Wait

Il control panel del codice finale, che richiede solamente la presenza di un indicatore che visualizzi il valore della operazione compiuta all'interno del For Loop, è rappresentato in Fig. 11.2; il relativo control panel in Fig. 12.2.

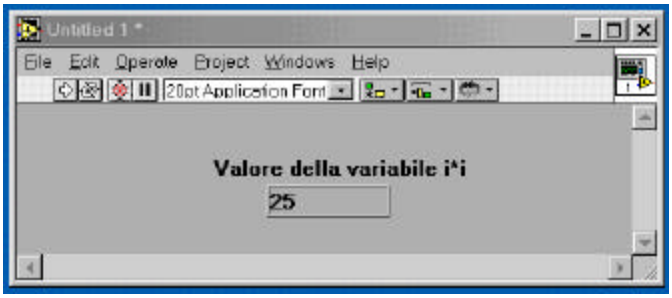


Fig. 11.2 Esempio di uso della funzione For Loop (Contrl Panel)

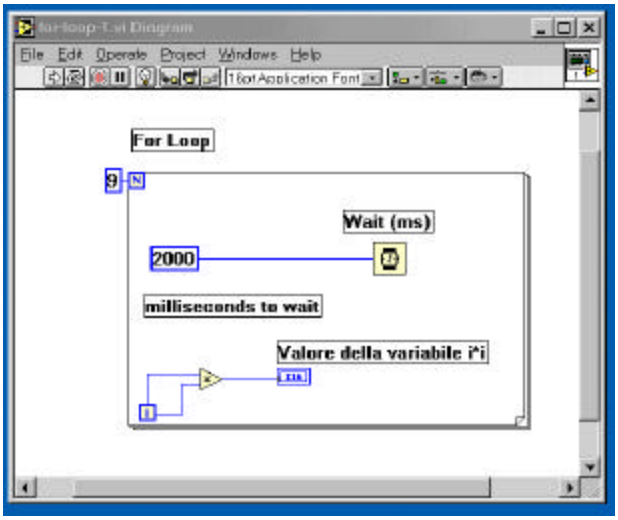


Fig. 12.2 Esempio di uso della funzione For Loop (Block Diagram)

Se si trascina fuori dal for loop l'indicatore del valore della variabile, si nota che il filo si interrompe. Se si cerca di collegare, mediante il rocchetto, il terminale dell'*operatore prodotto* con l'indicatore, si nota che fino al bordo del for loop il filo è pieno, mentre dopo il bordo il filo si interrompe. Questo è dovuto al fatto che il ciclo for loop consente di immagazzinare, durante ciascun ciclo, i dati in un deposito rappresentato dal rettangolo nero presente sul bordo del ciclo stesso (Fig. 13.2); questa funzione prende il nome di auto-indexing.

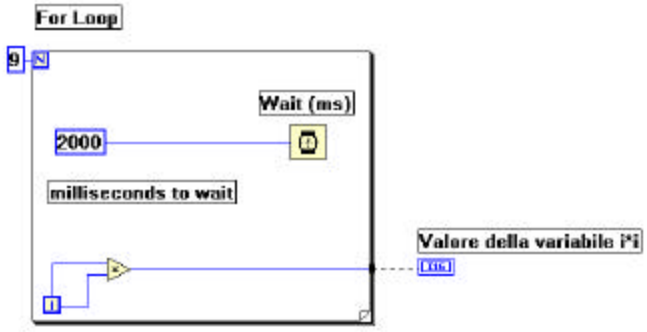


Fig. 13.2 Esempio di uso della funzione For Loop per la creazione di una matrice (Fase a)

E' sufficiente trasformare l'indicatore in una matrice (Fig. 14.2) e ripristinare il collegamento mediante il rocchetto (Fig. 15.2). In questo modo, mediante un ciclo for loop, si è inizializzata una matrice.

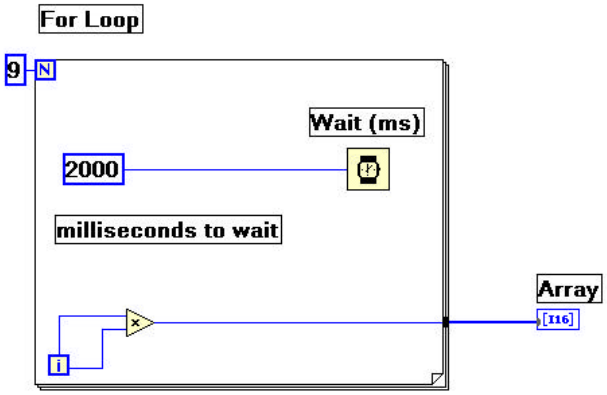


Fig. 14.2 Esempio di uso della funzione For Loop per la creazione di una matrice – Block Diagram (Fase b)

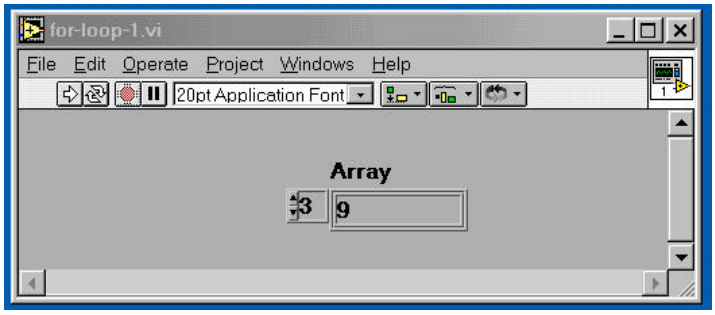


Fig. 15.2 Esempio di uso della funzione For Loop per la creazione di una matrice – Control Panel (Fase b)

Nel caso simmetrico al precedente (matrice entrante nel ciclo for loop) la funzione auto-indexing **consente di definire automaticamente il numero di cicli e di estrarre in successione i vari elementi della matrice (Fig. 16.2); in altre parole non è necessario definire il valore di N: il valore di N è uguale alla dimensione del vettore entrante.**

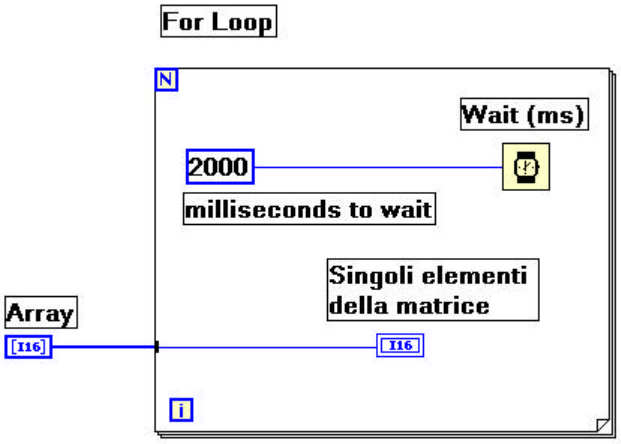


Fig. 16.2 Esempio di uso della funzione For Loop nel caso di matrice in ingresso – Block Diagram

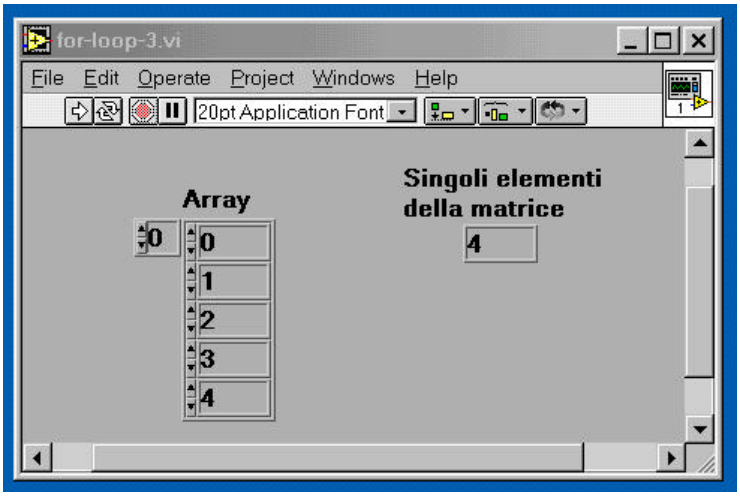


Fig. 17.2 Esempio di uso della funzione For Loop nel caso di matrice in ingresso – Control Panel

Una ulteriore funzione che è opportuno ricordare prende il nome di Shift Register**; essa serve a trasferire, in un loop, il valore di una variabile da una iterazione alla iterazione successiva.**

Per creare uno shift register si opera come segue (Fig. 18.2):

1. **posizionare il mouse sul contorno della struttura for loop;**
2. **premere il tasto destro del mouse;**
3. **scegliere l'opzione** Add Shift Register**.**

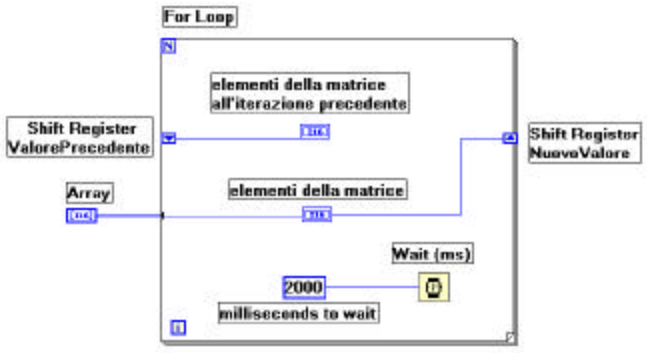


Fig. 18.2 Esempio di Shift Register in un ciclo For Loop – Block Diagram

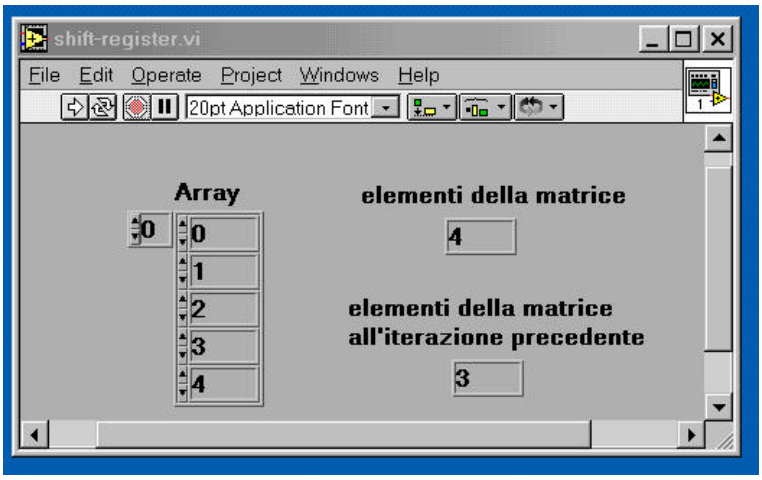
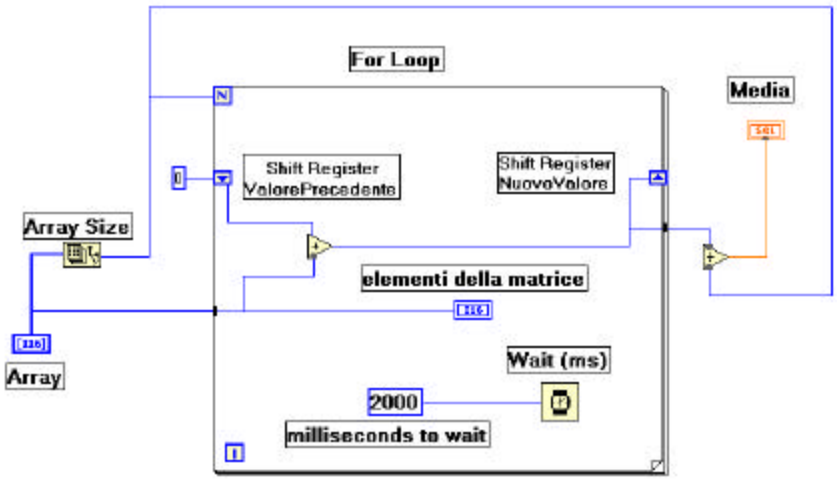


Fig. 19.2 Esempio di Shift Register in un ciclo For Loop– Control Panel

Un esempio d'uso della funzione shift register è la media dei dati contenuti nella matrice dell'esempio di Fig. 19.2; i relativi block diagram e control panel sono



rappresentati rispettivamente in Fig. 20.2 e 21.2.

Fig. 20.2 Shift Register: esempio di uso in un ciclo For Loop per il calcolo della media – Block Diagram

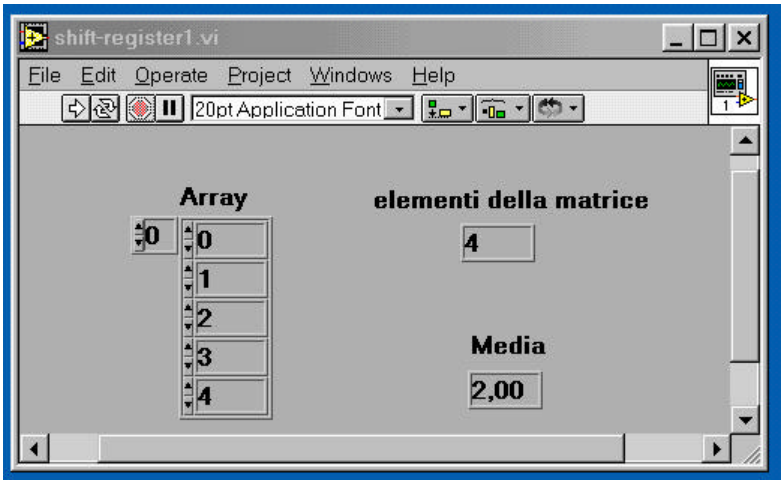


Fig. 21.2 Shift Register: esempio di uso in un ciclo For Loop per il calcolo della media – Control Panel

Relativamente all'esempio di Fig. 20.2 è opportuno fornire una descrizione della sequenza delle operazioni che vengono compiute alle varie iterazioni:

1. prima che venga eseguito il ciclo for loop viene azzerato lo shift register di sinistra (relativo al valore precedente);
2. all'iterazione corrispondente ad i = 0, viene effettuata la somma tra il valore contenuto nello shift register di sinistra (che in questo caso è, come detto al punto 1, è uguale a zero) ed il valore del primo elemento del vettore contenente i numeri da cui estrarre la media;
3. il valore di tale somma viene inviato allo shift register di destra che lo memorizza e lo trasferisce allo shift register di sinistra;
4. all'iterazione i=1 viene effettuata la somma tra il valore contenuto nello shift register di sinistra (contenente la somma precedentemente calcolata) ed il valore del secondo elemento del vettore;
5. il valore di tale somma viene inviato allo shift register di destra che lo memorizza; in altre parole lo shift register memorizza la somma parziale e mette tale valore a disposizione dell'iterazione successiva;
6. questo ciclo di operazioni continua fino a che dal vettore sono stati estratti in successione tutti gli elementi che lo compongono;
7. è importante osservare che tale somma parziale viene inviata anche sul perimetro del ciclo for loop: disabilitando la funzione auto indexing tali valori non vengono memorizzati in una matrice ma ad ogni iterazione il valore attuale sostituisce il valore precedente di modo che al termine dell'intero for loop si ha in uscita l'ultimo numero calcolato; nel presente esempio tale numero corrisponde alla somma complessiva;
8. una volta completata la somma di tutti gli elementi del vettore, non resta che dividere tale valore per il numero di elementi del vettore; tale numero si ricava tramite la funzione array size, già descritta in precedenza.

La struttura While Loop esegue il codice contenuto all'interno del suo perimetro fino a che una specifica condizione non è più vera.

Si procede come segue:

1. si trascina la struttura sul block diagram e la si dimensiona in modo che possa contenere al suo interno la parte di codice che deve essere eseguita trascinandone lo spigolo in basso a destra;
2. si definisce sul control panel un controllo booleano;
3. si scrive la parte di codice che deve essere eseguita all'interno del perimetro del While Loop.

Una struttura While Loop è equivalente al seguente metacodice:

Do

Esecuzione del subcodice

If (condizione = true)

continue

Else Stop

A titolo di esempio costruiamo un semplice codice, analogo al precedente, che effettua il quadrato della variabile **i**, ne visualizza il risultato ed attende 2 secondi; questa sequenza di operazioni deve essere eseguita fino a che l'interruttore booleano passa dalla condizione *true* alla condizione *false*.

Il control panel è rappresentato in Fig. 22.2 ed il relativo block diagram in Fig. 23.2.

E' importante sottolineare la presenza di un nuovo oggetto: il pulsante di stop. Questo pulsante opera come segue:

1. la condizione iniziale corrisponde al pulsante non premuto; a tale condizione è associato lo stato *false*;
2. quando il pulsante viene premuto commuta il suo stato da *false* a *true*;
3. il pulsante conserva tale stato fino a che il codice non legge lo stato del controllo;
4. una volta effettuata la lettura il controllo ritorna allo stato originario.

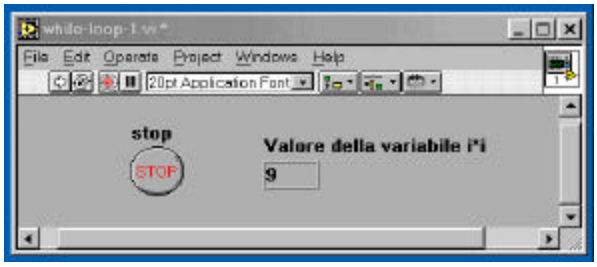


Fig. 22.2 Esempio di uso della funzione While Loop (Contrl Panel)

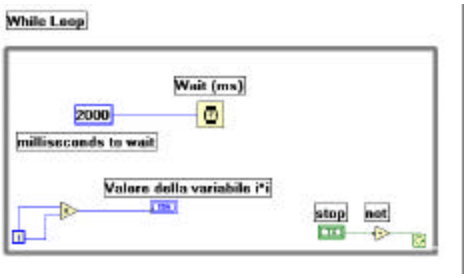


Fig. 23.2 Esempio di uso della funzione While Loop (Block Diagram)

E' importante sottolineare il fatto che nel block diagram compare l'operatore booleano **not;** tale operatore converte l'informazione di stato in uscita dal controllo booleno stop (*false*) nel valore *true*; ad ogni esecuzione del ciclo while loop, il sistema controlla lo stato del controllo e si arresta solo quando in uscita dall'operatore booleano **not** c'è lo stato *false*.

Occorre fare una ulteriore osservazione: se si trascina fuori dal ciclo while loop l'indicatore del valore della variabile, anche in questo caso il filo si interrompe in modo analogo a quanto avveniva nel caso del for loop visto in precedenza; ma se si collega mediante il rocchetto il terminale dell'operatore prodotto con l'indicatore, il filo non presenta interruzioni (Fig. 24.2). Questo è dovuto al fatto che il ciclo while loop non ha abilitata di default la funzione di auto-indexing; in altre parole ad ogni ciclo il valore che viene calcolato giunge fisicamente al piccolo rettangolo nero posto sul perimetro della funzione while loop ma non si accumula in una matrice: ogni nuovo valore so si sovrascrive al precedente. Pertanto al termine del ciclo viene trasferito all'indicatore solamente l'ultimo valore calcolato.

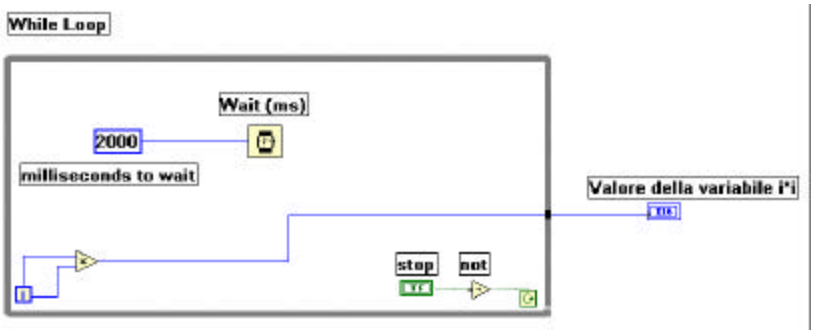


Fig. 24.2 Esempio di uso della funzione While Loop con la funzione auto-indexing disabilitata (Block Diagram)

Si posizioni la freccia sul suddetto piccolo rettangolo e si abiliti la funzione auto-indexing premendo il tasto destro del mouse e si proceda in modo analogo a quanto fatto nel caso del ciclo for loop definendo un array (Fig. 14.2) e ristabilendo il collegamento mediante il rocchetto. In questo modo ci si pone nelle identiche condizioni del ciclo for loop precedentemente illustrato.

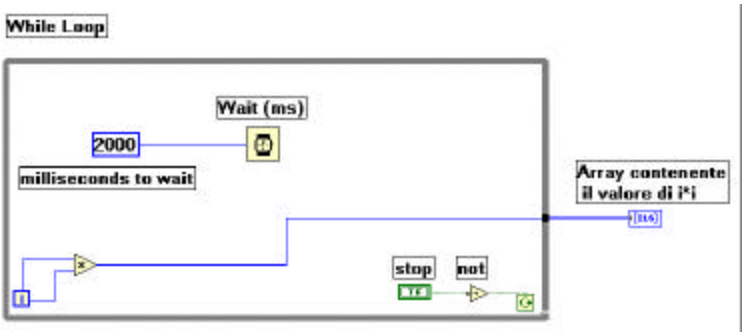


Fig. 25.2 Esempio di uso della funzione While Loop con la funzione auto-indexing abilitata (Block Diagram)

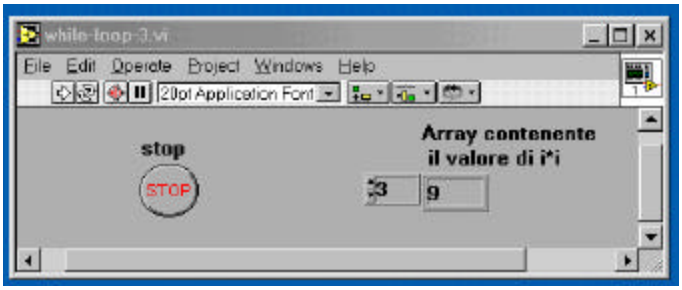


Fig. 26.2 Esempio di uso della funzione While Loop con la funzione auto-indexing abilitata (Control Panel)

In modo assolutamente analogo si può operare nel caso di un ciclo for disabiltando la funzione di auto-indexing.
Riepilogando: l'auto indexing è abilitato per default nel caso del for loop (i dati si accumulano in una matrice) e disabilitato nel caso del while loop (i dati si sovrascivono).

Per concludere facciamo ancora un esempio di for-loop concatenati per la generazione di una matrice 2 D. Si opera come segue:

1. tramite il cammino **Controls** e **Array & Cluster** creare sul control panel una matrice trascinandovi l'icona **Array** (Fig 27.2);

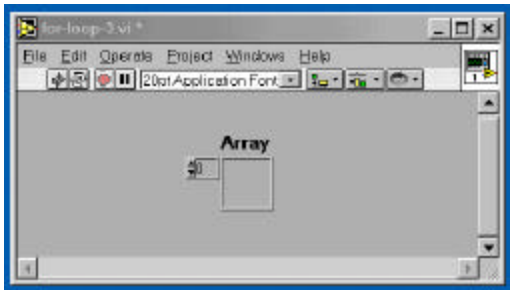


Fig. 27.2 Esempio di uso della funzione For Loop per la creazione di un array 2 D – Fase A (Control Panel)

2. trascinare verso il basso il vertice in basso a destra del rettangolo contenente il contatore dell'array; in questo modo si aggiunge una nuova dimensione all'array (Fig. 28.2);

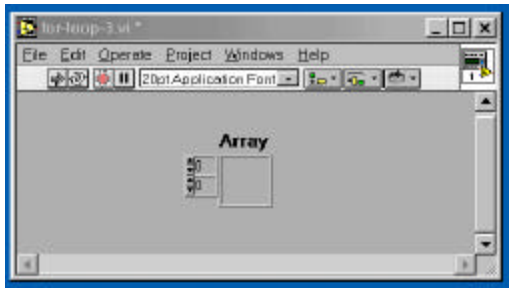


Fig. 28.2 Esempio di uso della funzione For Loop per la creazione di un array 2 D – Fase B (Control Panel)

3. definire il tipo dell'array (in questo caso si tratta di un array di numeri interi): per fare ciò si trascina all'interno dell'array un indicatore numerico digitale (digital numeric indicator) e, successivamente, mediante il tasto destro, si definisce il tipo di rappresentazione (Representation) di tale indicatore. Nel presente caso si è scelto il tipo seguente: numero intero I16 (cioè un numero intero espresso mediante una parola a 16 bit);

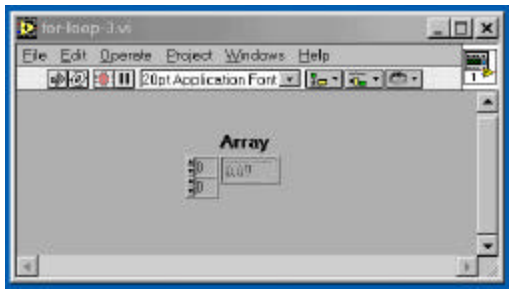


Fig. 29.2 Esempio di uso della funzione For Loop per la creazione di un array 2 D – Fase C (Control Panel)

4. passare al block diagram e creare un codice come quello rappresentato in Fig. 30.2. Esso opera come segue:
   - la funzione for loop esterna esegue un primo ciclo ponendo il proprio contatore al valore 0. Dato che al suo interno è contenuto a sua volta un ciclo for, la funzione for loop interna esegue 5 volte il proprio ciclo generando ad ogni ciclo un numero che è dato dal prodotto del valore numerico del contatore del ciclo interno per il valore numerico del contatore del ciclo esterno (0□0=0, 1□0=0, 2□0=0, 3□0=0, 4□0=0) e accumula questi numeri nella prima riga di un array temporaneo;
   - la funzione for loop esterna esegue una seconda volta il proprio ciclo ponendo il proprio contatore al valore 1; in seguito la funzione for loop interna esegue nuovamente 5 volte il proprio ciclo generando ad ogni ciclo un numero che è dato dal prodotto del valore numerico del contatore del ciclo interno per il valore numerico del contatore del ciclo esterno (0□1=1, 1□1=1, 2□1=2, 3□1=3, 4□1=4) e accumula questi numeri nella seconda riga di un array temporaneo;
   - la suddetta operazione viene ripetuta 5 volte; al termine l'intero array viene trasferito nell'indicatore.

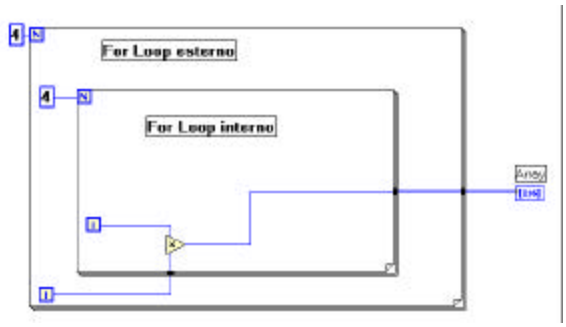


Fig. 30.2 Esempio di uso della funzione For Loop per la creazione di un array 2 D – Fase D (Block Diagram)

Al termine dell'esecuzione del programma si può espandere l'array e contemporaneamente vederne le varie righe (Fig. 31.2).

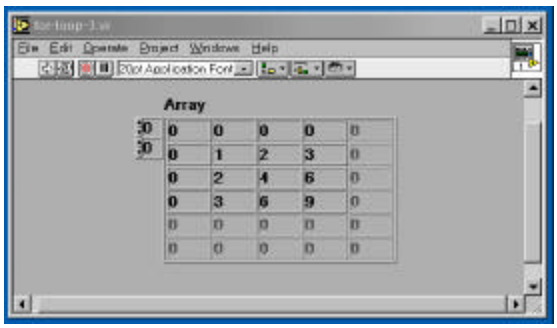


Fig. 31.2 Esempio di uso della funzione For Loop per la creazione di un array 2 D – Fase E (Control Panel)

La struttura sopra descritta è equivalente al seguente codice FORTRAN:

```
      DIMENSION A(0:4)
      INTEGER A
      DO 100 I=0,4
      DO 100 J=0,4
            A(I,J)=I* J
100   CONTINUE
```

Sequence

L'ordine di esecuzione delle istruzioni che costituiscono un codice di calcolo si impone ponendo in una sequenza prestabilita i vari elementi che lo compongono; tale strategia di programmazione prende il nome di *controllo del flusso.*
Mentre linguaggi come C, BASIC, FORTRAN hanno un controllo del flusso interno poiché le istruzioni vengono eseguite nell'ordine in cui sono scritte, nel programma LabVIEW si deve ricorrere alla struttura **Sequence** per assicurare che le varie operazioni vengano effettuate nella successione voluta. Per maggior chiarezza ricorriamo all'esempio illustrato in Fig. 32.2.

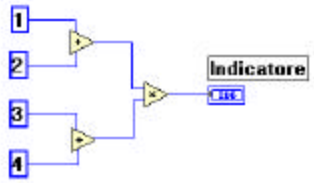


Fig. 32.2 Esempio di una successione di operazioni - Block Diagram

Se sul block diagram si attiva la funzione **Highlight Execution**, che consente di visualizzare passo passo le singole operazioni, si osserva che la prima operazione effettuata è quella di somma del numero 1 con il numero 2; successivamente viene effettuata la somma del numero 3 con il numero 4. Infine viene effettuata l'operazione prodotto tra il risultato della prima somma e quello della seconda. In generale questa sequenza di operazioni di somma potrebbe risultare invertita. E' evidente che nel caso in cui si operi con dei semplici calcoli la successione delle operazioni non è importante perché anche se le due operazioni di somma vengono eseguite invertendone la successione il risultato finale evidentemente non cambia.
Invece, se ad esempio si vuole valutare quanto tempo ci impiega una certa operazione ad essere eseguita è necessario prima far partire il cronometro, poi eseguire l'operazione che si vuole effettuare, infine arrestare il cronometro.

Per fare questo si procede come segue:

1. tramite il cammino **Functions** e **Structures** trascinare sul block diagram la funzione sequence;

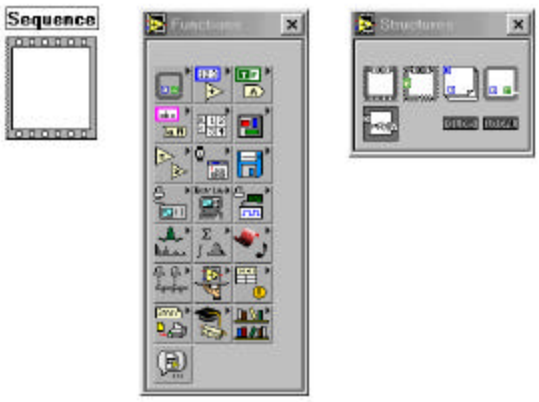


Fig. 33.2 Esempio di creazione di una Sequence – Fase A (Block Diagram)

2. posizionarsi sul bordo della struttura Sequence e premere il tasto destro del mouse: scegliere la funzione **Add frame after**; si crea in questo modo una prima sequence (Fig. 34.2). Già dall'aspetto grafico (ricorda un fotogramma di una pellicola fotografica) si comprende il suo

significato: in ogni fotogramma si esegue una certa funzione; la funzione che deve essere seguita dopo appartiene al fotogramma successivo;

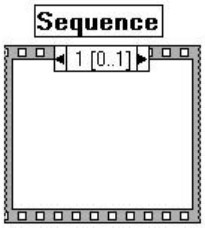


Fig. 34.2 Esempio di creazione di una Sequence – Fase B (Block Diagram)

3. posizionarsi nuovamente sul bordo della struttura Sequence e premere il tasto destro del mouse: scegliere la funzione **Add frame after** (Fig. 35.2); si crea in questo modo una seconda Sequence.

Ogni fotogramma della Sequence è identificato da un numero: il primo dal numero 0, il secondo dal numero 1 ed infine il terzo dal numero 2. Si passa dall'uno all'altro mediante la funzione operate value **schematizzata con una mano con il dito indice teso.**

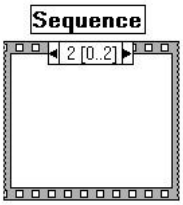


Fig. 35.2 Esempio di creazione di una Sequence – Fase C (Block Diagram)

Supponiamo ora di voler calcolare quanto tempo ci impiega un ciclo For Loop ad effettuare centomila volte l'operazione somma ed a scrivere ogni volta il valore calcolato.

Nel primo frame (identificato dal numero 0) viene inserita la funzione **Tick Counts (ms)** a cui si accede tramite il cammino **Functions** e **Time & Dialog**. Questa funzione misura il tempo a partire da un'origine arbitraria.

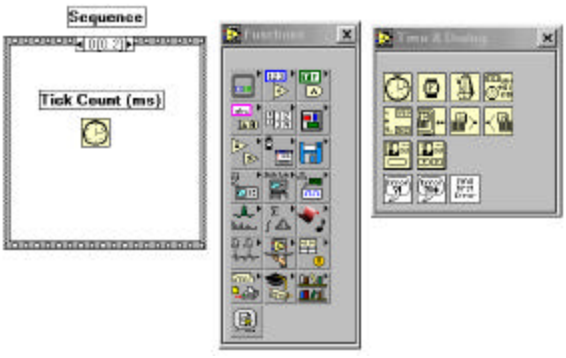


Fig. 36.2 Esempio di creazione di una Sequence – Fase D (Block Diagram)

E' necessario trasferire questo tempo ai frame successivi; per fare ciò occorre creare sul frame un **sequence local** operando come segue: posizionare la freccia sul bordo del frame, premere il tasto destro del mouse scegliendo l'opzione **Add sequence local**. Una volta collegato il misuratore di

tempo al sequence local appare una freccia diretta verso l'esterno del frame; tramite questa freccia si trasferisce l'informazione sul tempo ai frame successivi (Fig. 37.2).

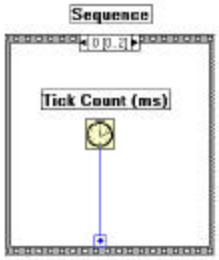
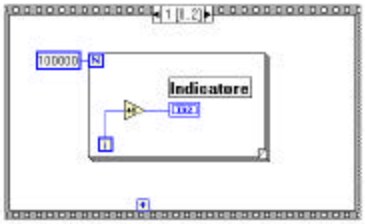


Fig. 37.2 Esempio di creazione di una Sequence – Fase E (Block Diagram)

Posizionarsi sul secondo frame (identificato dal numero 1) e creare un codice che, per esempio, aggiunga la cifra 1 all'indice dei For Loop ed ogni volta ne



mostri il risultato sul video (Fig.38.2).

Fig. 38.2 Esempio di creazione di una Sequence – Fase F (Block Diagram)

Nel terzo ed ultimo frame si dovrà prima effettuare la misura del tempo e poi la differenza tra il tempo misurato nel presente frame e quello misurato nel primo frame (Fig. 39.2).

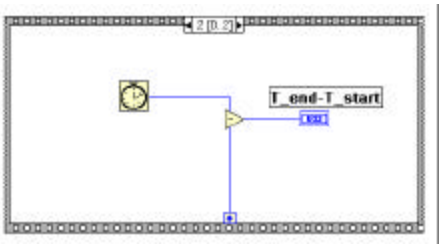


Fig. 39.2 Esempio di creazione di una Sequence – Fase G (Block Diagram)

Eseguendo il programma di Fig. 40.2 risulta un tempo pari a 144 ms.

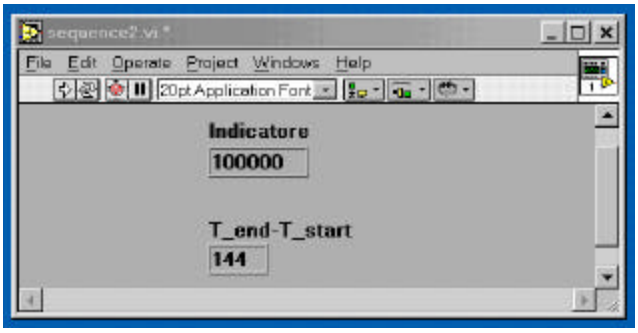


Fig. 40.2 Esempio di creazione di una Sequence – (Control Panel)

Se si elimina la fase di visualizzazione dei vari risultati parziali e tali risultati vengono memorizzati in una matrice trasferita all'ultimo frame mediante un ulteriore Sequence local (Figg. 41.2 e 42.2) il tempo si riduce drasticamente a 30 ms (Fig. 43.2). Questo è dovuto al fatto che l'operazione di scrittura su video è estremamente dispendiosa in termini di impiego di tempo. Si deve ricorrere alle visualizzazioni parziali solo se assolutamente indispensabile.

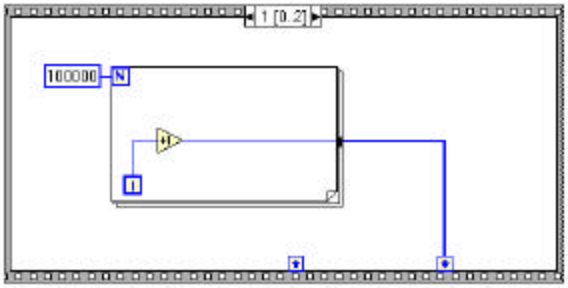


Fig. 41.2 Esempio di creazione di ottimizzazione della Sequence precedente – Fase A (Control Panel)

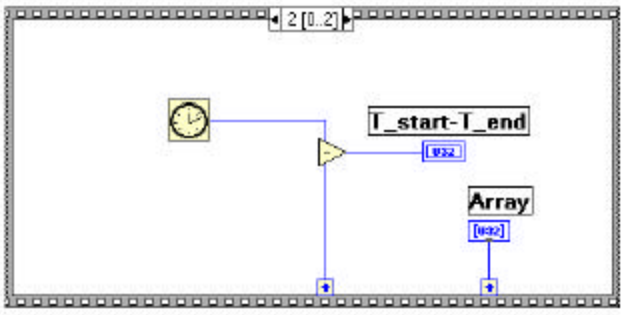


Fig. 42.2 Esempio di creazione di ottimizzazione della Sequence precedente – Fase B (Control Panel)

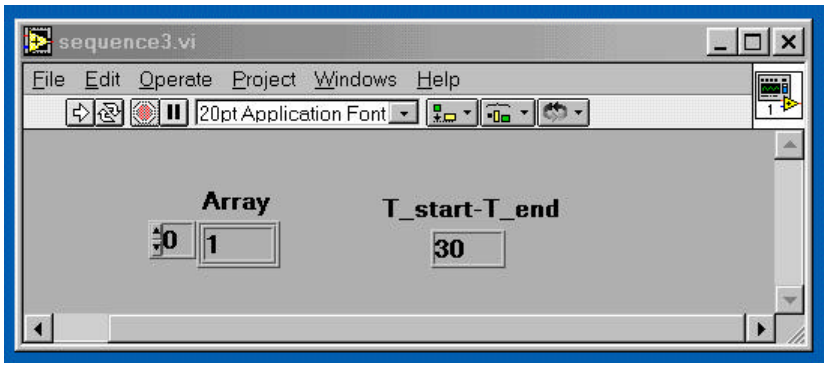


Fig. 43.2 Esempio di creazione di ottimizzazione della Sequence precedente (Block Diagram)

Case

La struttura **Case** è una struttura che si comporta in modo analogo all'istruzione **if-then-else** utilizzata nei linguaggi BASIC, FORTRAN e PASCAL; si accede ad essa tramite il cammino **Functions** e **Structures** (Fig. 44.2).

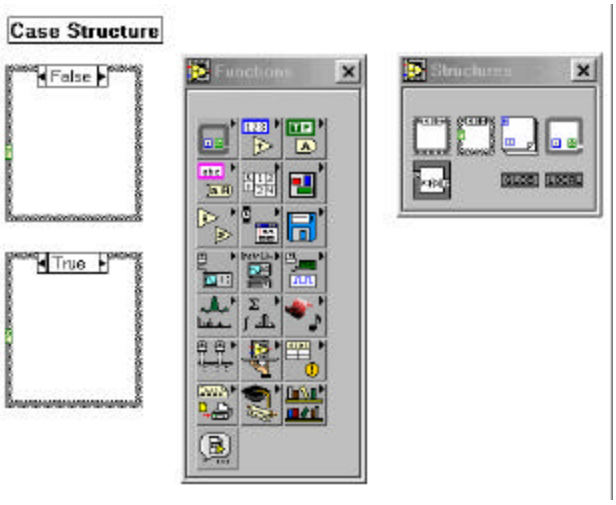


Fig. 44.2 Struttura Case

La struttura case illustrata in Fig. 44.2 è divisa in due sottodiagrammi: se la variabile booleana collegata con il punto interrogativo della funzione **case** assume il valore *true* si accede al sottodiagramma *true*; in caso contrario si accede al sottodiagramma *false*. Si passa da un sottodiagramma all'altro posizionando l'indice della mano virtuale (Operate Value della Tools Palette) sulla freccia e premendo tale freccia.
Per meglio illustrare il funzionamento di questa importante struttura ricorriamo ad un facile esempio: si voglia calcolare la radice quadrata di un numero. Nel caso in cui il numero introdotto sia negativo il programma deve segnalare tale fatto all'operatore. Il block diagram di tale programma è illustrato in Fig. 45.2 ed il relativo Control Panel in Fig. 46.2.

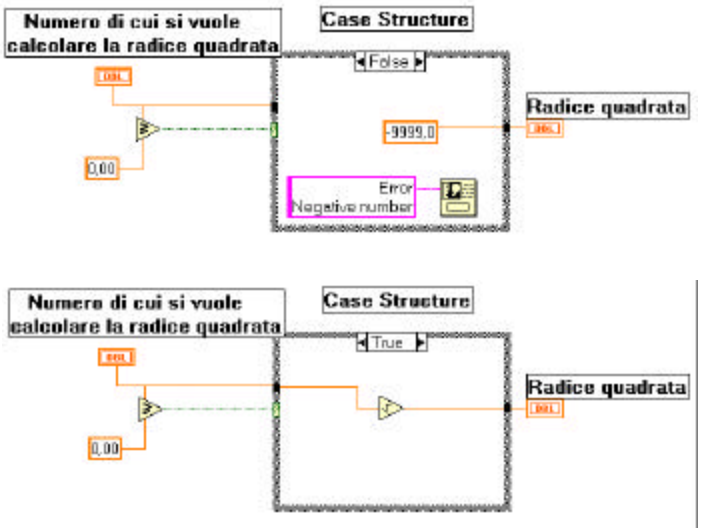


Fig. 45.2 Esempio di uso della Struttura Case – Block Diagram

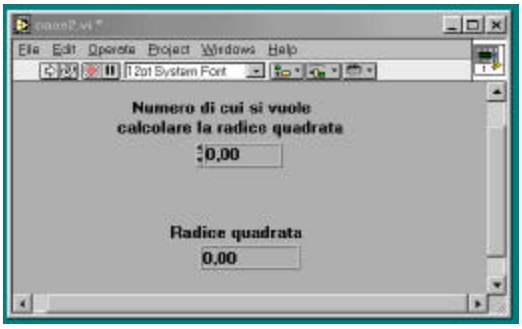


Fig. 46.2 Esempio di uso della Struttura Case – Block Diagram

Nel block diagram compare una funzione che viene qui utilizzata per la prima volta: il **dialog box**. Tale funzione, quando viene eseguita, evidenzia sullo schermo un messaggio (Fig. 47.2). Si accede ad essa tramite il cammino **Functions** e **Time & Dialog.** Il messaggio è costituito da una stringa di caratteri collegata all'icona della funzione **One Button Dialog**.

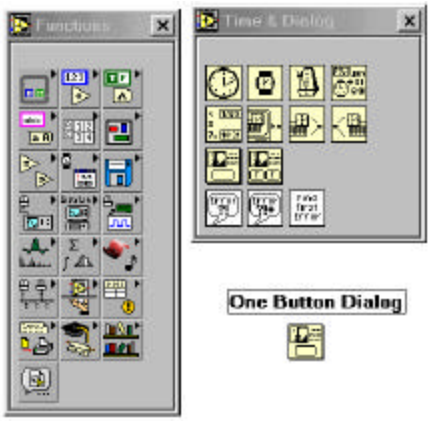


Fig. 47.2 Dialog Box

Esaminiamo infine l'esempio di un codice che consente di calcolare il valore massimo di una matrice di numeri interi. In realtà LabVIEW possiede una funzione in grado di effettuare un calcolo di questo tipo; tuttavia, mediante questo codice, è possibile, utilizzando gran parte delle funzioni illustrate fino ad ora, fare un riepilogo che consente di approfondire ulteriormente le problematiche fino ad ora descritte in questo paragrafo. In Fig. 48.2 è illustrato il control panel di tale esempio; il relativo bloock diagram è illustrato in Fig. 49.2.

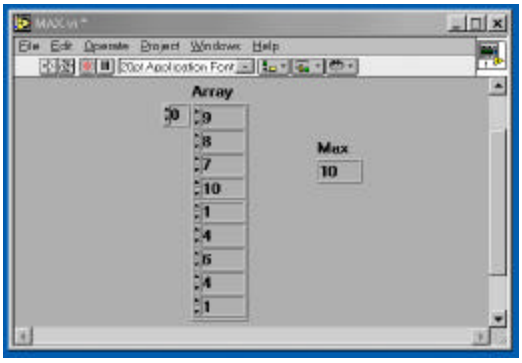


Fig. 48.2 Ricerca del massimo di un vettore – Control Panel

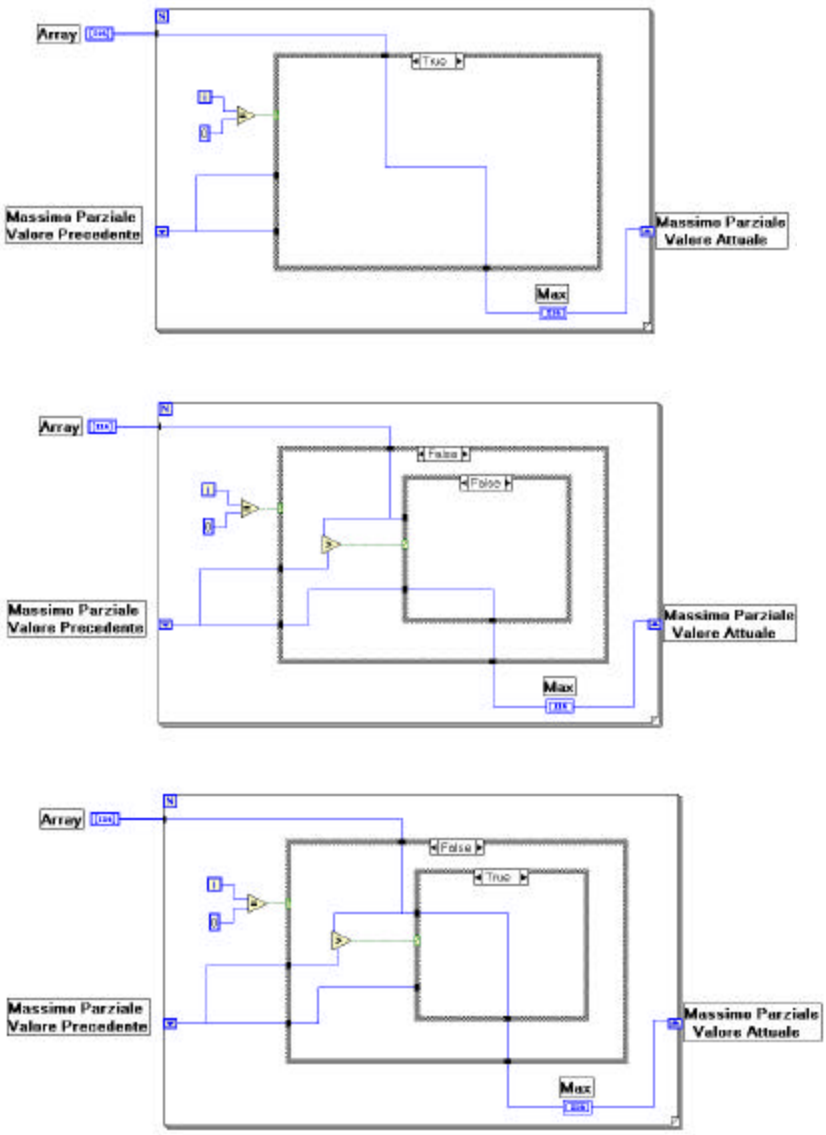


Fig. 49 Ricerca del massimo di un vettore - Block Diagram

Il programma di Fig. 49 esegue la seguente sequenza di operazioni:

1. la funzione **Case** controlla se si sta eseguendo la prima iterazione o una iterazione successiva;
2. nel caso si verifichi la condizione i=0 (prima iterazione), il primo valore dell'array di cui si vuole determinare il valore massimo viene inviato sia alla variabile Max sia allo shift register di destra che lo memorizza e lo trasferisce allo shift register di sinistra;
3. all'iterazione successiva viene confrontato il valore dell'elemento successivo dell'array con il valore contenuto nello shift register di sinistra; si possono verificare due casi:
   a. tale valore è inferiore al valore contenuto nello shift register; in questo caso nella variabile Max viene trasferito il Max precedente (proveniente dallo shift register di sinistra);
   b. tale valore è superiore al valore contenuto nello shift register; in questo caso nella variabile Max viene trasferito il valore dell'elemento dell'array.
4. il procedimento continua fino a che sono stati esaminati tutti gli elementi dell'array.

SubVI

In ambiente LabVIEW è possibile creare delle subroutine; tali subroutine, che prendono il nome di subVI, consentono di semplificare in modo notevole il block diagram di un codice, come vedremo in dettaglio nell'esempio.
E' utile sottolineare che vi è una stretta analogia nell'utilizzo di queste subVi con l'utilizzo delle subroutine nei linguaggi BASIC, FORTRAN, PASCAL; tuttavia, mentre nei linguaggi di tipo tradizionale una subroutine non può essere eseguita autonomamente ma deve essere inserita in un programma chiamante, nel caso di LabVIEW è possibile eseguire autonomamente una SubVI; questo ne agevola notevolmente la messa a punto.

La prima operazione che si deve eseguire al momento della creazione di una subVI è quella della stesura del testo del codice; questa operazione è identica a quella della stesura di un normale codice LabVIEW.
Qualunque chiamata ad una subroutine richiede l'individuazione dei dati in ingresso e dei dati in uscita, e questo è valido per qualunque linguaggio di programmazione; nel caso di LabVIEW i dati in ingresso sono costituiti da controlli mentre quelli in uscita da indicatori.

A titolo di esempio utilizzeremo due codici che sono stati sviluppati in precedenza: il codice che consente di calcolare la media di una serie di numeri immagazzinati in un vettore ed il codice che calcola il massimo tra i suddetti valori.

La Fig. 50.2 riporta il control panel del nuovo codice ottenuto collegando tra loro tali codici; essi sono sostanzialmente invariati rispetto alla versione originale; in comune hanno l'array dei numeri che devono essere esaminati per determinare la media ed il massimo.

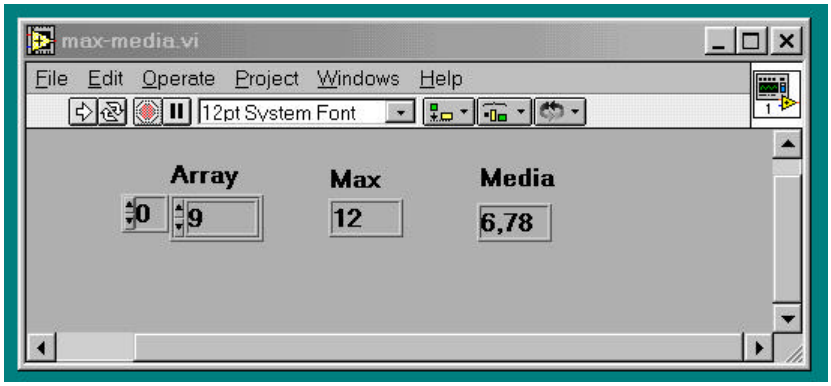


Fig. 50.2 Codice per il calcolo del massimo di un array di dati ed il loro valor medio – Control Panel

La Fig. 51.2 riporta il relativo block diagram; rispetto ai codici precedenti è stato eliminato solamente l'indicatore dei massimi relativi, che era utilizzato solo da un punto di vista didattico per mostrare la progressione nella variazione del valore del massimo al variare dei vari elementi dell'array. Inoltre è stata soppressa anche la funzione di temporizzazione perché in questo caso assolutamente superflua.

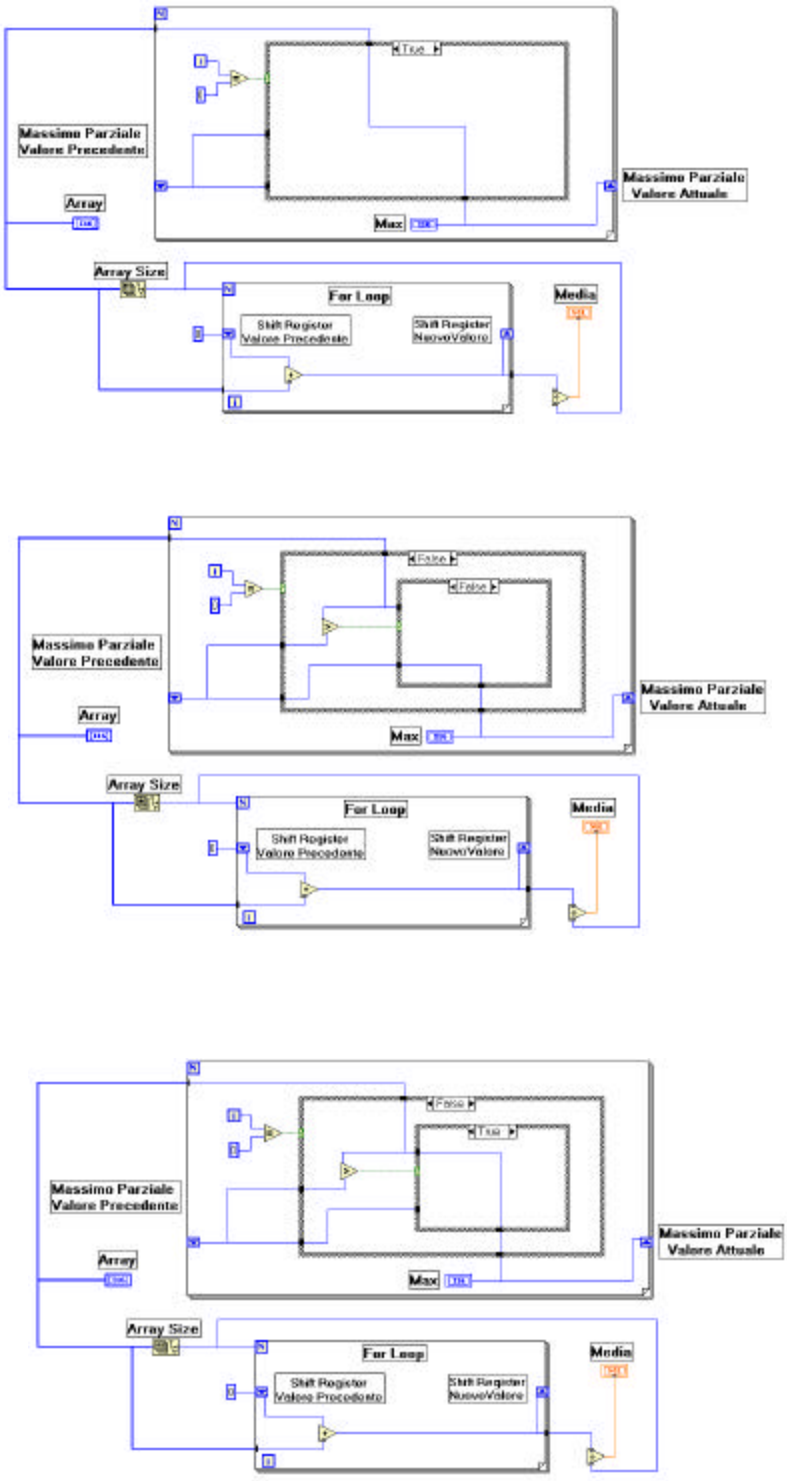


Fig. 51.2 Codice per il calcolo del massimo di un array di dati ed il loro valore medio – Block Diagram

La successione delle procedure per creare una subVI è la seguente:

1. posizionare il mouse sull'icona in alto a destra del control panel;
2. premere il tasto destro del mouse;
3. scegliere l'opzione **Edit Icon**; ci si trova in un ambiente grafico, molto simile a quello di Paint di Windows 95/98, che consente di personalizzare il disegno dell'icona (Fig. 52.2). Nell'esempio si è scelto un logo molto semplice (M-M) in grado comunque di richiamare visivamente il significato della subVI (ricerca del massimo e calcolo della media);

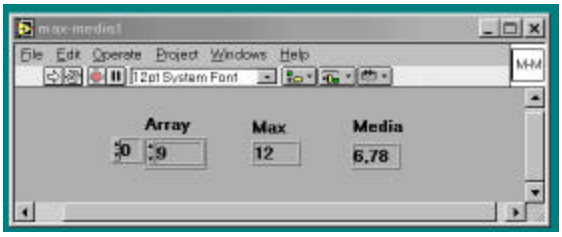


Fig. 52.2 Esempio di icona personalizzata sul contro panel

4. posizionare il mouse sull'icona in alto a destra del control panel;
5. premere il tasto destro del mouse;
6. scegliere l'opzione Show Connector; questa opzione divide l'immagine dell'icona in alcuni rettangoli. Il numero di tali rettangoli dipende dal numero di controlli ed indicatori presenti sul control panel; comunque è possibile sovradimensionare questa struttura in vista di futuri sviluppi del codice (Fig. 53.2).

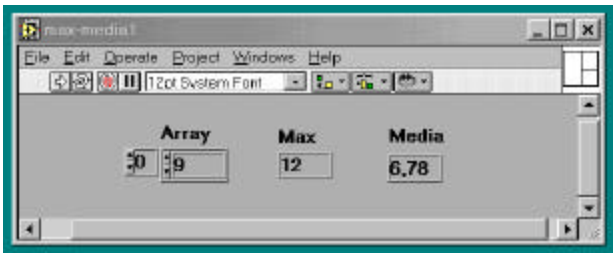


Fig. 53.2 Prima fase per la creazione dei connettori sul contro panel

7. a questo punto non resta che stabilire una connessione tra i connettori ed i vari elementi (controlli ed indicatori) che compaiono sul contro panel. Per fare ciò occorre effettuare due operazioni in sequenza:
   - posizionare il rocchetto del filo su uno dei rettangoli che serviranno ad individuare il connettore e successivamente premere il tasto sinistro del mouse;
   - posizionare il rocchetto del filo su un controllo od un indicatore e successivamente premere il tasto sinistro del mouse. In questo modo si è stabilito un collegamento tra il controllo e/o l'indicatore ed il connettore (Fig. 54.2); se si dispone di un monitor a colori ognuno di questi rettangoli assume un specifico colore; se il collegamento è con un numero intero assume il colore blu, se con un numero reale arancio, se con una variabile stringa cyan, ecc.
8. Salvare la subVI in una directory a piacere.

In questo modo si è conclusa la procedura di creazione di una subVI: vediamo ora come la si può utilizzare.

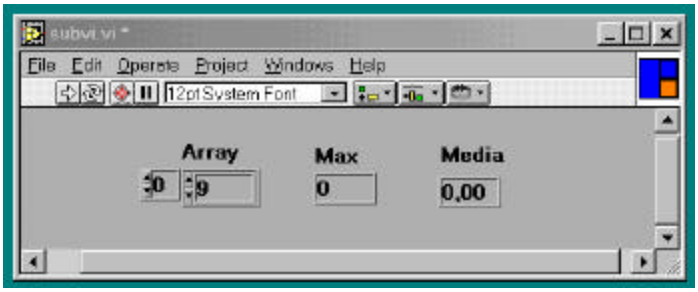


Fig. 54.2 Seconda fase per la creazione dei connettori sul contro panel

A questo fine creiamo un programma che genera N numeri mediante il generatore di numeri casuali, ne calcola la media ed il valore massimo (il generatore di numeri casuali genera un numero compreso tra o ed 1).

Il block diagram è illustrato in Fig. 55.2 ed il relativo control panel in Fig. 56.2.

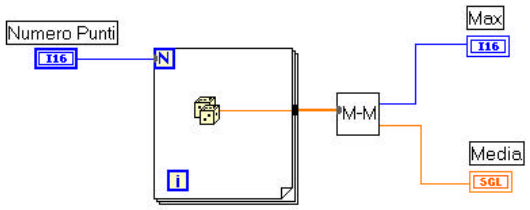


Fig. 55.2 Block Diagram di un programma che calcola la media ed il valore massimo di una sequenza di numeri casuali

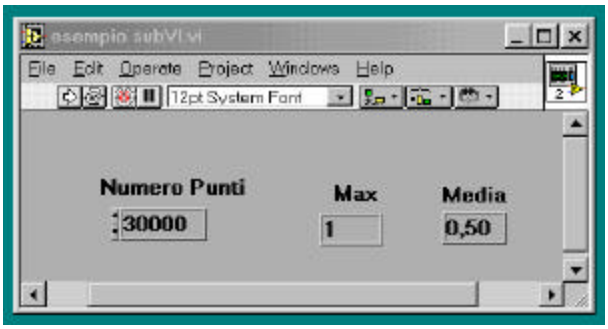


Fig. 56.2 Control Panel di un programma che calcola la media ed il valore massimo di una sequenza di numeri casuali

Per inserire l'icona della subVI è sufficiente seguire il seguente percorso:
Function e **Select a VI**; a questo punto viene richiesto di indicare il percorso del file di tale subVI.

Una breve considerazione sul programma appena realizzato che utilizza il generatore di numeri casuali: al crescere del numero di punti il valore medio tende a 0,5 come era lecito attendersi.

Funzioni grafiche in Labview

LabVIEW mette a disposizione del programmatore numerosi tipi di grafici. Prenderemo in esame solamente tre tipi di uso molto frequente: waveform chart, waveform graph, XY graph**.**

Waveform chart

Questo tipo di grafico, posto nella subpalette **Graph** della palette **Controls,** consente di disegnare una o più curve sullo stesso diagramma. Esso è utilizzato molto spesso nei loop in quanto disegna i vari punto man mano che essi vengono calcolati od acquisiti, aggiungendo nuovi dati non appena essi sono disponibili.
In un waveform chart i punti sono riportati nell'ordine di acquisizione ed in ordinata è riportato il valore della grandezza che rappresentano. In Fig. 51.2 è rappresentato il control panel di un programma che genera 100 numeri casuali e via via che questi vengono calcolati li disegna su di un grafico; in Fig. 52.2 è rappresentato il relativo block diagram.

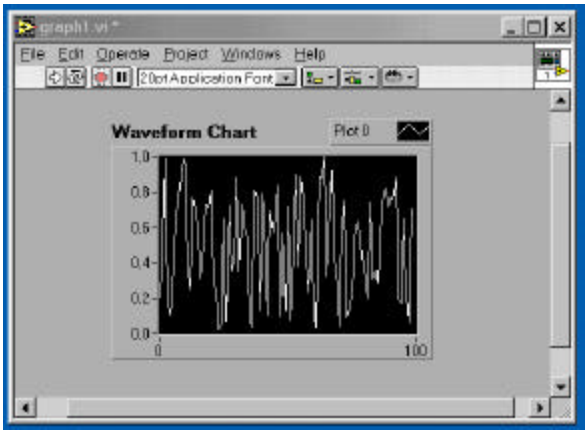


Fig. 51.2 Esempio di uso della funzione Waveform Chart– Control Panel

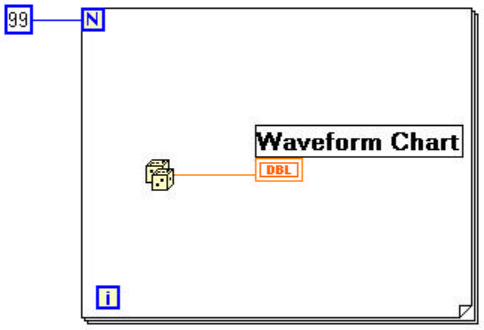


Fig. 52.2 Esempio di uso della funzione Waveform Chart – Block Diagram

Se devono essere visualizzate più curve sullo stesso diagramma si deve ricorrere alla funzione **bundle**, accessibile dal block diagram tramite il cammino **Function**, **Cluster**, **Bundle** (Fig. 53.2). Tale funzione ha lo scopo di convogliare insieme più dati; nel presente caso ogni singola sorgente di dati sarà collegata al proprio terminale in ingresso del bundle mentre il terminale in uscita sarà

collegato al waveform chart. Il bundle è una funzione espandibile; infatti in condizioni iniziali la funzione bundle convoglia due insiemi di dati; tuttavia posizionando il rocchetto su uno degli ingressi del bundle e premendo il tasto destro appare un menu: scegliere l'opzione **add input** tante volte quanti sono gli ingressi che si desidera generare.

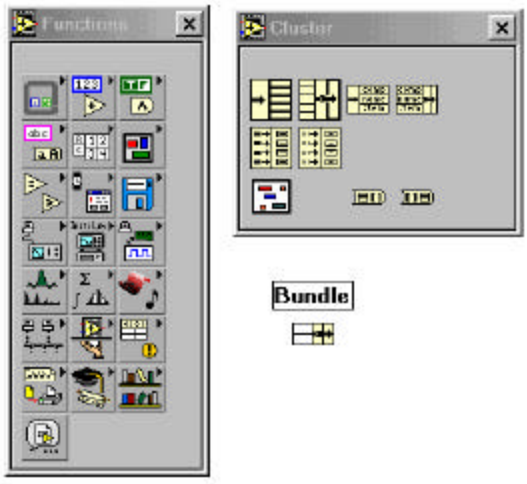


Fig. 53.2 Funzione bundle

Nell'esempio di Figg. 54.2 e 55.2 è riportato il caso in cui sono utilizzati tre simulatori di termometro; al primo valore in uscita è aggiunta la costante 10 mentre al terzo valore in uscita è sottratta la costante 10 per differenziare tra loro le curve. Il diagramma di figura 53.2 presenta sulla destra anche una legenda che identifica le singole curve. In condizioni iniziali vi è una sola legenda relativa alla prima curva diagrammata; posizionando sul vertice in basso a destra mediante il mouse la freccia e trascinando verso il basso tale vertice si aggiungono le leggende delle altre curve che possono essere facilmente personalizzate posizionando la freccia sulla curva della leggenda e premendo il tasto destro del mouse: appaiono tutte le opzioni utili alla personalizzazione delle curve.

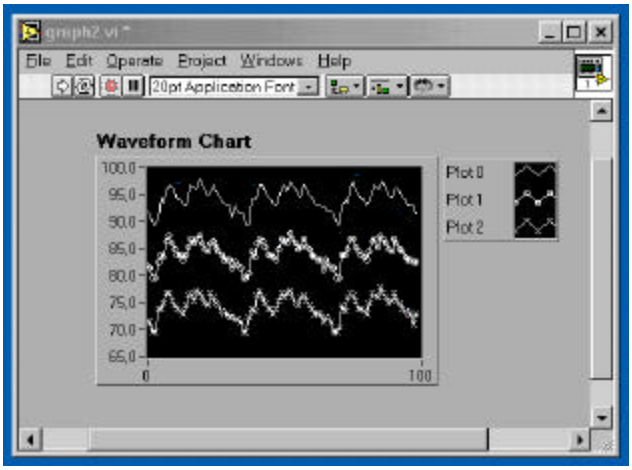


Fig. 54.2 Esempio di Waveform Chart con più curve sullo stesso diagramma – Control Panel

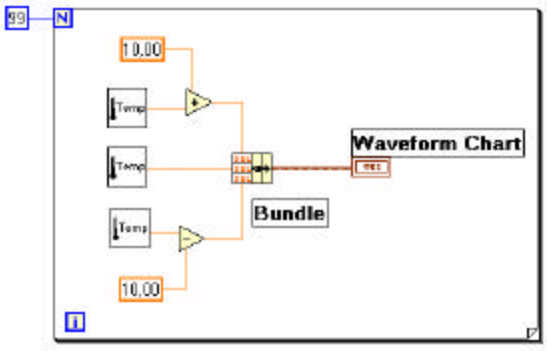


Fig. 55.2 Esempio di Waveform Chart con più curve sullo stesso diagramma – Block Diagram

Waveform graph e XY graph

A differenza della funzione chart, che disegna i dati man mano che questi vengono generati, la funzione **graph** li disegna tutti in una volta sola. Vi sono vari tipi di **graph**; faremo menzione di due soli: il **waveform graph** e l' **XY Graph**.

Waveform graph può essere utilizzato solo nel caso in cui i vari punti siano equispaziati; esso riporta sull'asse delle ordinate il valore dei vari punti e sull'asse delle ascisse il numero del punto od un valore ad esso proporzionale.
In Fig. 57.2 è rappresentato un caso analogo a quello illustrato in precedenza in Fig. 52.2; l'unica differenza è legata al fatto che i dati sono diagrammati tutti al temine del ciclo for.

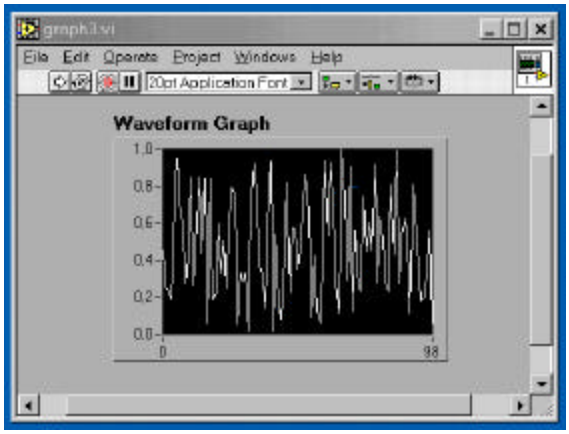


Fig. 56.2 Esempio di uso della funzione Waveform Graph – Control Panel

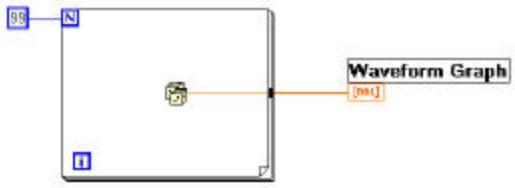


Fig. 57.2 Esempio di uso della funzione Waveform Graph – Block Diagram

Spesso può essere utile avere la possibilità di passare da un asse delle ascisse che fornisce solamente informazioni sulla successione degli eventi ad una vera e propria base tempi; ciò si può fare se si conosce l'istante in cui è stato acquisito il primo punto e la distanza temporale tra un punto e l'altro (in questo caso i vari punti devono essere equispaziati nel tempo). Anche in questo caso bisogna ricorrere alla funzione bundle come illustrato in Figg. 58.2 e 59.2. Nell'esempio si immagina che il primo punto sia stato acquisito all'istante 10 ed i vari punti distino tra loro di 2 s.

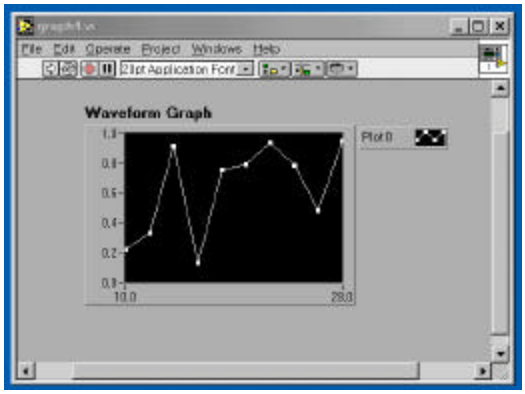


Fig. 58.2 Esempio di uso della funzione Waveform Graph con introduzione di una base tempi – Block Diagram

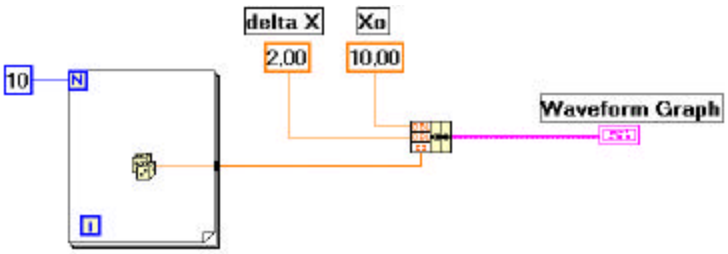


Fig. 59.2 Esempio di uso della funzione Waveform Graph con introduzione di una base tempi – Control Panel

Nel caso di più diagrammi sullo stesso grafico (caso analogo a quello illustrato in Fig. 55.2) il block diagram ha la struttura illustrata in Fig. 62.2.
Al posto della funzione bundle questa volta si deve utilizzare la funzione build array tramite il cammino **Function**, **Array**, **Buid Array** (Fig.60.2). Tale funzione consente, dati più array unidimensionali, di costruire un array bidimensionale come richiesto dalla funzione Waveform graph nel caso di più curve sullo stesso diagramma Fig. 61.2 e 62.2.

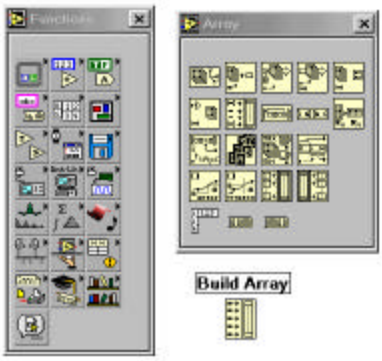


Fig. 60.2 Funzione Build Array

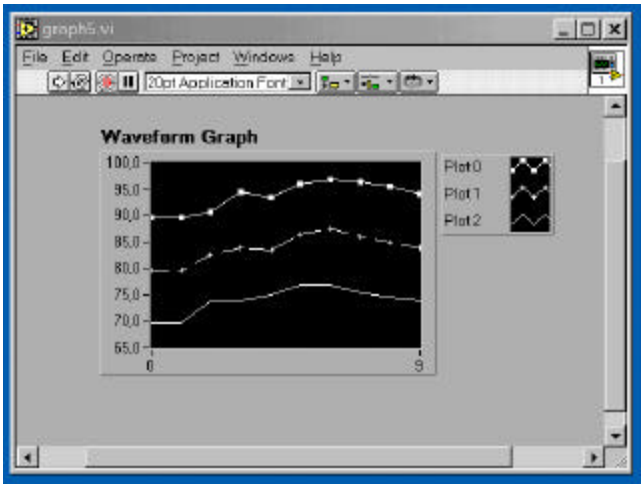


Fig. 61.2 Esempio di Waveform Graph con più curve sullo stesso diagramma – Control Panel

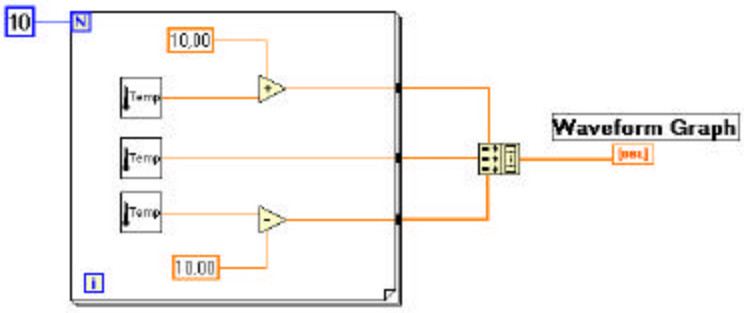


Fig. 62.2 Esempio di Waveform Graph con più curve sullo stesso diagramma – Block Diagram

Tralasciamo il caso in cui al diagramma di Fig. 61.2 si voglia sostituire sull'asse delle ascisse alla semplice numerazione degli eventi una base tempi; infatti questa operazione, peraltro possibile, presenta un grado di complessità che non è compatibile con gli scopi del corso dato che prevede l'uso combinato della funzione **build array** e **bundle**.

Passiamo ora ad esaminare l'ultimo caso: **XY Graph**.

Nel caso in cui l'acquisizione sia effettuata ad intervalli di tempo non costanti occorre conoscere la storia temporale dell'acquisizione; in altre parole occorre creare un vettore contenente i valori delle ascisse ed un vettore contenente quelli delle ordinate. Il block diagram di un programma che utilizza questo tipo di grafico è rappresentato in Fig. 63.2; il relativo Control Panel in Fig. 64.2.

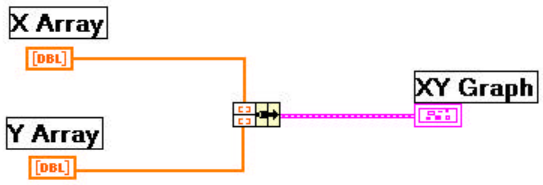


Fig. 63.2 Esempio di XY Graph – Block Diagram

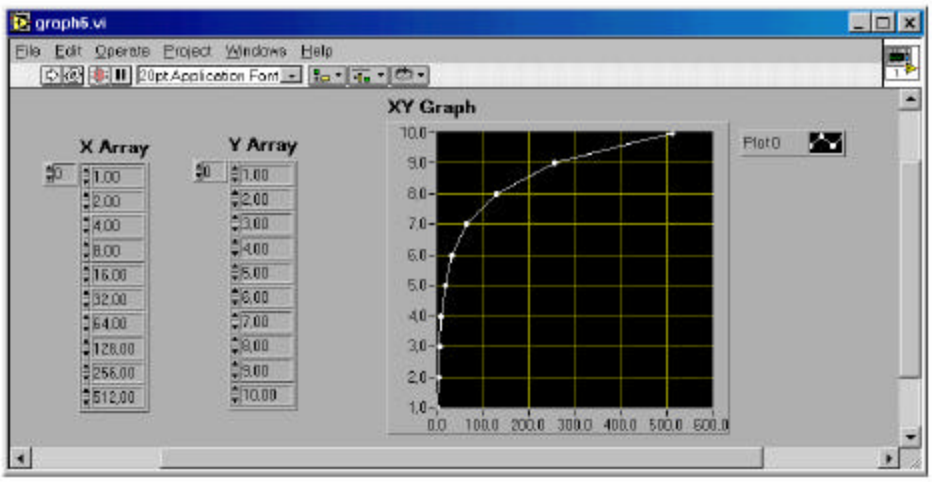


Fig. 64.2 Esempio di XY Graph – Control Panel

3. Introduzione all'acquisizione dati

Prima di passare alla descrizione di alcuni esempi di programmi per l'acquisizione dati e per il controllo di strumenti remoti, forniamo l'elenco ed il relativo significato di alcuni **acronimi** di uso comune in questo campo ed alcune nozioni basilari di questa tecnica.

Elenco dei principali acronimi

AC : **A**lternating **C**urrent
DC : **D**irect **C**urrent

Il significato originario di questi due acronimi era connesso al tipo di alimentazione dello strumento utilizzato. Nel caso di alimentazione tramite rete, si sta utilizzando uno strumento a corrente alternata (**AC**); nel caso di alimentazione tramite batteria, si sta utilizzando uno strumento a corrente continua (**DC**). Oggi questo acronimo ha assunto un significato più generale e fa riferimento al tipo di segnale che si deve acquisire; se il segnale varia rapidamente nel tempo, si ha a che fare con un segnale del tipo **AC**. Se invece la variazione nel tempo del segnale è trascurabile, si sta operando con un segnale del tipo **DC**.

ADC : **A**nalog to **D**igital **C**onversion o **A/D**

Questa conversione trasforma un valore analogico nel corrispondente valore digitale. La risoluzione della conversione dipende dal numero di bit a disposizione.
Se si opera con una **parola di 8 bit** il più grande numero esprimibile in notazione digitale è $\mathbf{2^8 = 256}$

Con una **parola di 12 bit** è $\mathbf{2^{12} = 4096}$

Con una **parola di 16 bit** è $\mathbf{2^{16} = 65536}$

Con una **parola di 24 bit** è $\mathbf{2^{24} = 16.777.216}$

Pertanto se il segnale analogico in ingresso viene acquisito mediante una scheda avente a bordo un convertitore Analogico-Digitale a 12 bit ed esso varia in un range compreso tra 0 e 10 Volt, la risoluzione effettiva è 10/4096 = 2,44 mV.

Se si vogliono apprezzare salti di tensione più piccoli, è necessario ricorrere a sistemi di acquisizione che operino con un numero maggiore di bit.

Per maggior chiarezza ricorriamo ad un esempio: mediante un trasduttore di pressione si intende calcolare la velocità di una corrente fluida; vale la ben nota relazione:

$$U = \sqrt{\frac{2\,p}{\rho}}$$

ove: U = velocità del fluido p = pressione dinamica ρ = massa volumica

Si intenda misurare la pressione mediante un trasduttore avente le seguenti caratteristiche:

$p_{max} = 500$ [Pa] $V_{max} = 2{,}5$ [V]

Pertanto il coefficiente k del trasduttore vale k = 200 [Pa/V].

Se la temperatura ambiente T_{amb} è pari a 15°C (288,16 °K) e la pressione atmosferica p_{atm} è pari ad una atmosfera (101300 Pa), la densità dell'aria è data da:

$$\rho = \frac{p_{atm}}{287 \cdot T_{amb}^{\circ}} = \frac{p_{atm}}{287 \cdot T_{amb}^{\circ}} = \frac{101300}{287 \cdot 288.16} = 1{,}22 \text{ kg/m}^3$$

Sostituendo si ha:

$$U = \sqrt{\frac{2\,k\,V}{r}}$$

Nella conversione analogico digitale avremo:

$$V = \frac{V_{max}}{N_{max}}\ N$$

ove $N_{max} = 2^{N_{bit}}$ è il massimo numero scrivibile se si adotta una parola di N_{bit}.
Pertanto:

$$U = \sqrt{\frac{2\,k}{r}\frac{V_{max}}{N_{max}}N} = \sqrt{\frac{2\,k\,V_{max}}{r}}\ \sqrt{\frac{N}{N_{max}}} = \sqrt{819{,}7}\ \sqrt{\frac{N}{N_{max}}}$$

Ne segue immediatamente che:

$$N = Int\left[\frac{r}{2\,K\,V_{max}}N_{max}\,U^2\right] = Int\left[\frac{1}{819{,}7}N_{max}\,U^2\right] \qquad e \qquad \left|\frac{\Ä U}{U}\right| = \frac{1}{2}\left|\frac{\Ä N}{N}\right|$$

Calcoliamo ora la precisione che si ottiene a 10 m/s se si utilizza una scheda da 8 bit od una da 12 bit:

- se utilizziamo una **scheda di acquisizione da 8 bit** $N_{max} = 256$; pertanto:

$$N = Int\left[\frac{N_{max}}{819{,}7}U^2\right] = Int\left[\frac{256}{819{,}7}10^2\right] = Int\left[31{,}231\right] = 31$$

dato che $\Delta N=1$, avremo:

$$\left|\frac{\Delta N}{N}\right| = 3{,}23\,\% \qquad \text{e quindi} \qquad \left|\frac{\Delta U}{U}\right| = 1{,}6\,\%$$

- se utilizziamo una **scheda di acquisizione da 12 bit** $N_{max} = 4096$; pertanto:

$$N = Int\left[\frac{N_{max}}{819{,}7}U^2\right] = Int\left[\frac{4096}{819{,}7}10^2\right] = Int\left[499{,}7\right] = 499$$

dato che $\Delta N=1$, avremo:

$$\left|\frac{\Delta N}{N}\right| = 0{,}2\,\% \qquad \text{e quindi} \qquad \left|\frac{\Delta U}{U}\right| = 0{,}1\,\%$$

Dimezzando la velocità (5 m/s) si ha:**12bit**

8 bit

$$N = \mathrm{Int}\left[\frac{256}{819{,}7}5^2\right] = \mathrm{Int}[7{,}81] = 7 \qquad \left|\frac{\Delta N}{N}\right| = 14{,}3\,\% \quad \text{e} \quad \left|\frac{\Delta U}{U}\right| = 7{,}14\,\%$$

12 bit

$$N = \mathrm{Int}\left[\frac{4096}{819{,}7}5^2\right] = \mathrm{Int}[124{,}92] = 124 \qquad \left|\frac{\Delta N}{N}\right| = 0{,}8\,\% \quad \text{e} \quad \left|\frac{\Delta U}{U}\right| = 0{,}4\,\%$$

E' evidente che al diminuire della velocità occorre aumentare la capacità di risoluzione del sistema, cioè il numero di bit del convertitore analogico digitale, per mantenere la precisione della misura.

DAQ : Data Acquisition

Acronimo per indicare l'operazione della raccolta dei dati che, nella maggior parte dei casi, avviene effettuando una conversione **ADC**.

DAC : Digital to Analog Conversion

L'operazione duale della **ADC** .

DMA : Direct Memory Access

La maggior parte delle moderne schede di acquisizione ha a bordo il **DMA** che consente di trasferire direttamente i dati nella **RAM** del computer, aumentando in questo modo la velocità di trasmissione dei dati.

GPIB: General Purpose Interface Bus (noto anche come **HP-IB = Hewlett Packard Interface Bus**) e IEE 488.2 (**Institute of Electrical and Electronic Engineers Standard** 488.2)

E' uno standard mondiale di comunicazione tra strumenti e computers; sviluppato negli anni '60 dalla Hewlett Packard per consentire la programmazione tramite PC remoto, è diventato un protocollo di comunicazione accettato universalmente.

RS-232: Recomended Standard # 232

E' uno standard proposto dalla Instrument Society of America per le comunicazioni seriali

Nel caso di una acquisizione dati il primo problema è quello di individuare il "range" di variazione delle grandezze da acquisire e dotarsi del trasduttore avente le caratteristiche di sensibilità e precisione adeguate ad effettuare le misura. Il compito del trasduttore è quello di convertire la grandezza da acquisire in un segnale di tensione. Ad esempio nel campo delle misure di temperatura esistono trasduttori che operano in base a principi fisici differenti per convertire la temperatura in un segnale elettrico (un elenco sommario dei vari tipi di trasduttori disponibili sul mercato è riportato in tabella 1).

Fenomeno	Trasduttore
Temperatura	Termocoppia Rilevatore di temperatura a resistenza Termistore Sensore a circuito integrato
Luce	Tubo fotomoltiplicatore Cella fotoconduttiva
Suono Forza e pressione	Microfono Strain gauge Trasduttore piezoelettrico Cella di carico
Posizione (spostamento)	Potenziometro Trasformatore differenziale lineare Encoder ottico
Flusso di fluido	Misuratore di flusso a pressione differenziale Misuratore di flusso rotativo Misuratore di flusso ad ultrasuoni

Tabella 1

In generale tutti i segnali sono variabili nel tempo; essi possono essere inquadrati in due grandi categorie:

1. segnali digitali
2. segnali analogici

I **segnali digitali** (o segnali binari) ammettono due soli livelli discreti: un livello detto "high level" (o On) ed un livello detto "low level" (o Off).

I **segnali analogici** contengono informazioni sulla variazione continua del segnale rispetto al tempo.

I segnali digitali si suddividono in due ulteriori tipi:

1. Segnali On-Off

Fig. 1.3 Esempio di segnali On-Off

Un segnale On-Off trasferisce solamente l'informazione sullo stato di un dispositivo; ad esempio luce accesa / luce spenta ma non trasferisce alcuna informazione sull'intensità della luce o sulla sua variazione nel tempo. In altre parole trasferisce l'informazione sullo stato di uno switch (ad esempio l'output di un **TTL** cioè di un **T**ransistor **T**ransistor **L**ogic).
Lo strumento in grado di misurare questo tipo di segnale è semplicemente un "digital state detector".

2. Segnali a treno di impulsi

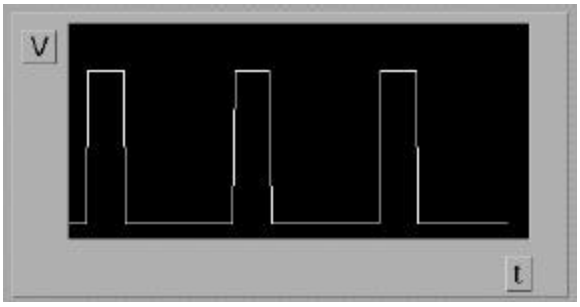


Fig. 2.3 Esempio di segnale a treno di impulsi

Questo segnale consiste in una serie di stati di transizione; l'informazione è associata alla distanza temporale tra uno stato ed il successivo.
Un tipico esempio di strumento che genera questo tipo di segnale è l'**encoder tachimetrica** che serve per misurare la velocità di rotazione di un albero.

I **segnali analogici** si suddividono in tre tipi:

1. Segnali analogici DC

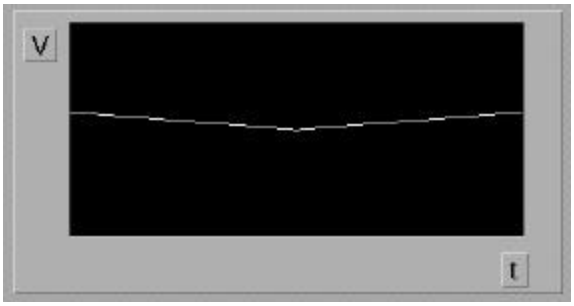


Fig. 3.3 Esempio di segnale analogico DC

Il segnale analogico DC è un segnale che non varia nel tempo o varia molto lentamente e l'informazione è solamente associata al livello del segnale. In questo caso è molto meno importante la frequenza di acquisizione della precisione della misura del livello del segnale.
La misura della temperatura in un ambiente o della tensione di una batteria sono esempi di questo tipo di segnale.

2. Segnali analogici nel dominio del tempo

Fig. 4.3 Esempio di segnale analogico nel dominio del tempo

Segnali di questo tipo sono caratterizzati dal fatto che l'informazione non è solo legata al livello del segnale ma a come questo segnale vari nel tempo (questo tipo di segnale viene anche detto forma d'onda). Queste misure devono essere effettuate ad una frequenza di acquisizione che riproduca in modo adeguato la forma d'onda. Pertanto la scheda di acquisizione che misura un segnale di questo tipo consiste in un ADC, un orologio (clock) che misura il trascorrere del tempo ed un "trigger" che assicura di far partire la misura ad un ben preciso istante.
Un elettrocardiogramma è un tipico esempio di questo tipo di segnale.

3. Segnali analogici nel dominio delle frequenze

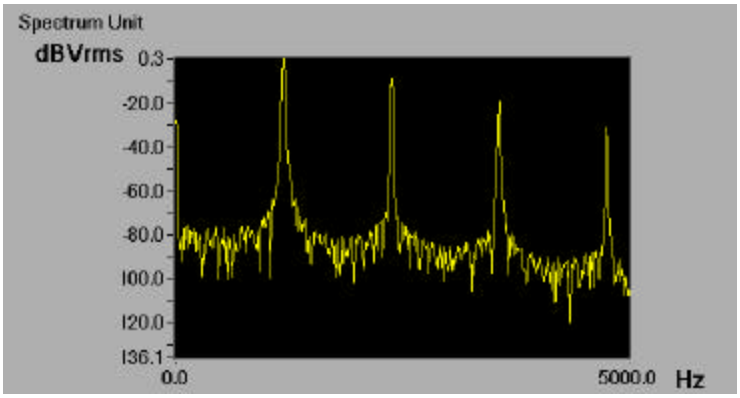


Fig. 5.3 Esempio di segnali analogici nel dominio delle frequenze

Questo tipo di segnale fornisce informazioni sullo spettro in frequenza del segnale stesso. Lo strumento per effettuare questo tipo di misure deve contenere un ADC, un clock ed un trigger; in aggiunta lo strumento deve includere la necessaria capacità di analisi per estrarre dal segnale le informazioni sulla sua distribuzione in frequenza. Esempio tipico è l'analisi di una risposta acustica.

Componenti di un sistema di acquisizione dati

La catena che costituisce un sistema di acquisizione dati è costituita dai seguenti elementi:

1. generazione fisica del segnale,
2. sensore o trasduttore che converta il segnale fisico in un segnale elettrico come una tensione o una corrente,
3. eventuale amplificazione del segnale in uscita dal trasduttore,
4. scheda di acquisizione dati che converte il segnale analogico in ingresso in un segnale digitale,
5. computer dotato di software dedicato che controlla il sistema di acquisizione dati, analizza i dati acquisiti e presenta i risultati elaborati.

Vi sono casi in cui si preferisce ricorrere ad un modulo di acquisizione dati esterno; il colloquio con il computer remoto avviene tramite porta parallela o porta seriale.

I parametri fondamentali che caratterizzano un scheda di acquisizione dati sono la risoluzione della scheda, il range di misura, il guadagno, la frequenza di campionamento:

1. **risoluzione** della scheda: questo parametro è stato esaminato in dettaglio in precedenza parlando di conversione analogico digitale;
2. **intervallo di misura** o **range**: riguarda i valori di tensione minimi e massimi consentiti dalla scheda (in genere da 0 a 10 V, o da –5 V a 5 V); questo consente di adattare il range dell'acquisitore al range del segnale, in modo da misurare il segnale con la massima risoluzione possibile;
3. **guadagno**: sta ad indicare una qualunque operazione di amplificazione o di attenuazione del segnale prima che esso venga digitalizzato.

   Se ad esempio il segnale in ingresso è compreso tra 0 e 5 V e la scheda di acquisizione ha un range che varia tra 0 e 10 V, occorre amplificare il segnale con un guadagno di 2; in questo modo la scheda, nella conversione A/D, utilizza interamente la sua capacità di risoluzione. Infatti si immagini di operare con una scheda di acquisizione con un convertitore a 3 bit (raramente utilizzato, tuttavia esplicativo da un punto di vista didattico): il convertitore divide l'intervallo di misura in $2^3 = 8$ sotto-intervalli, ciascuno di ampiezza pari a 1,25 V; pertanto un valore di tensione compreso tra 0 e 1,25 V è associato al numero binario 000 (livello n. 1), un valore di tensione compreso tra 1,25 e 2,5 V è associato al numero binario 001 (livello n.2), ecc. (Fig.11.3).

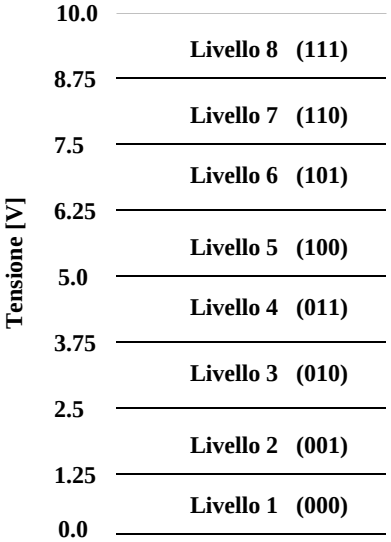


Fig. 11.3 Schema di conversione livello tensione numero binario

nell'intervallo di misura 0 ÷ 5 V il convertitore A/D utilizza solo quattro delle otto sotto-divisioni disponibili per effettuare la conversione con una perdita evidente di precisione.
Se invece si amplifica il segnale con un guadagno 2 **prima** di effettuare la digitalizzazione, il convertitore A/D utilizza tutte le 8 sotto-divisioni possibili e la rappresentazione del segnale è conseguentemente più accurata.

Il range di misura, la risoluzione e il guadagno di una scheda DAQ determinano la **variazione di tensione minima** misurabile, che rappresenta il bit meno significativo del valore digitale, ed è spesso chiamato **larghezza del codice (code width)**. La minima variazione del segnale che può essere rilevata è calcolata come segue:

$$\text{minima var iazione del segnale} = \frac{\text{range di tensione}}{\text{guadagno} \times 2^{\text{risoluzione in bit}}}$$

Per esempio, una scheda DAQ a 12 bit con range in ingresso da 0 a 10 V e un guadagno pari a 1 è in grado di misurare variazioni di 2,4 mV, mentre la stessa scheda con un range in ingresso da – 10 V a 10 V misurerebbe solo una variazione di 4,8 mV:

$$\frac{10}{1*2^{12}} = 2{,}4 \text{ mV} \qquad \frac{20}{1*2^{12}} = 4{,}8 \text{ mV}$$

4. **frequenza di campionamento**: è la frequenza con cui ha luogo la conversione A/D. Una frequenza di campionamento elevata consente di ottenere una migliore rappresentazione del segnale originale rispetto ad un frequenza di campionamento inferiore. Tutti i segnali in ingresso devono essere campionati ad una frequenza sufficientemente elevata da rappresentare adeguatamente il segnale analogico.
Una frequenza di campionamento troppo bassa può determinare una rappresentazione scadente del segnale analogico. Questa cattiva rappresentazione del segnale, detta **aliasing**, fa sì che il segnale sembri avere una frequenza completamente diversa dalla frequenza reale. Vedremo in pratica un esempio di questo tipo.
La figura 12.3 mostra un segnale campionato in modo adeguato ed uno sotto-campionato con evidente presenza di aliasing.

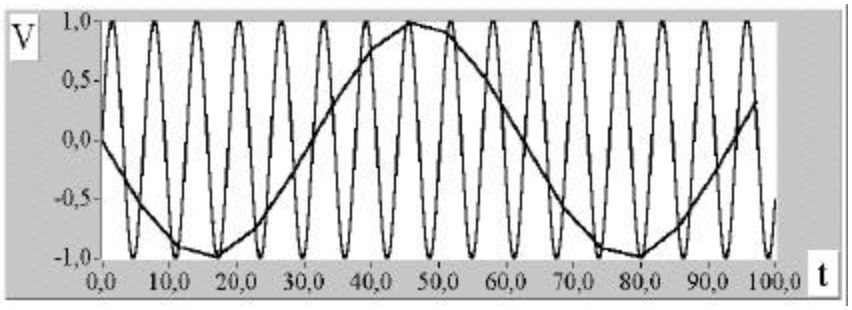


Fig. 12.3 Esempio di aliasing

Secondo il **Teorema del Campionamento di Nyquist**, per digitalizzare un segnale in modo adeguato è necessario che i campionamenti vengano effettuati come minimo con una frequenza di acquisizione pari a due volte la componente massima in frequenza che si vuole analizzare. Per esempio, segnali audio convertiti in segnali elettrici tramite microfoni hanno componenti di frequenza che possono raggiungere i 20 kHz; pertanto è necessario effettuare l'acquisizione utilizzando una scheda con frequenza di campionamento superiore a 40 kHz per acquisire il

segnale in modo appropriato. Invece i sensori di temperatura non richiedono elevate frequenze di acquisizione perché la temperatura in genere non subisce variazioni rapide; pertanto per effettuare misure di temperatura si può utilizzare una scheda con frequenza di campionamento anche non molto elevata.

5. **filtri per attenuare il rumore**: prima di essere convertito in un segnale digitale, il segnale analogico in generale è soggetto a distorsioni a causa della presenza del rumore, la cui sorgente può avere l'origine più svariata.

   Si possono in generale presentare i tre casi seguenti:

   - rumore a frequenza molto inferiore alla frequenza del fenomeno che si deve acquisire (tipico il caso dei 50 Hz di rete): in questo caso si può ricorre ad un condizionatore di segnale, a monte del sistema di acquisizione, dotato di un filtro passa alto che lasci passare solo i segnali a frequenza più alta
   - rumore a frequenza molto superiore alla frequenza del fenomeno che si deve acquisire: in questo caso si può ricorre ad un condizionatore di segnale, a monte del sistema di acquisizione, dotato di un filtro passa basso che lasci passare solo i segnali a frequenza più bassa
   - rumore nel campo di frequenza del fenomeno che si deve analizzare; è questo il caso più delicato.
     Si può operare come segue: acquisire ad una frequenza molto più elevata della frequenza caratteristica del fenomeno eseguendo un numero di campionamenti superiore al necessario. In questo modo, grazio al sovra-campionamento ogni singolo punto di acquisizione risulta essere la media dei punti acquisiti nel suo intorno: si riduce così il livello del rumore di un fattore

$$\frac{1}{\sqrt{\text{numero dei punto mediati}}}$$

     Per esempio, se si esegue la media su 100 punti, l'effetto del rumore sul segnale è ridotto di un fattore 0,1.

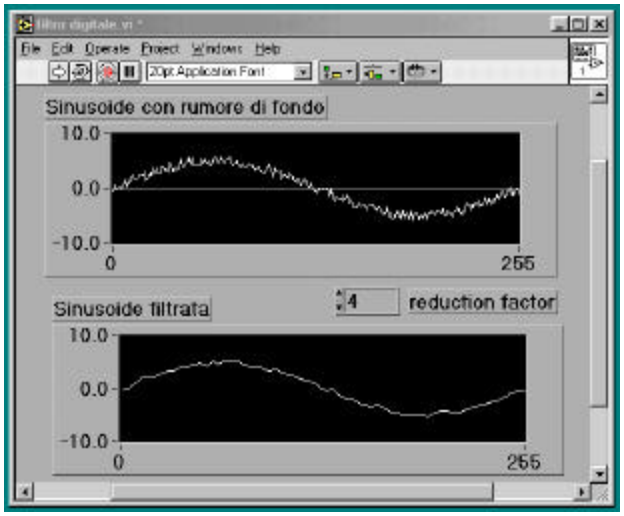


Fig. 13.3 Control Panel di un codice che filtra un segnale in ingresso

In base a quanto detto sopra, sviluppiamo un esempio di un semplice codice LabVIEW in grado di filtrare un segnale mediante la media di un segnale sovracampionato.

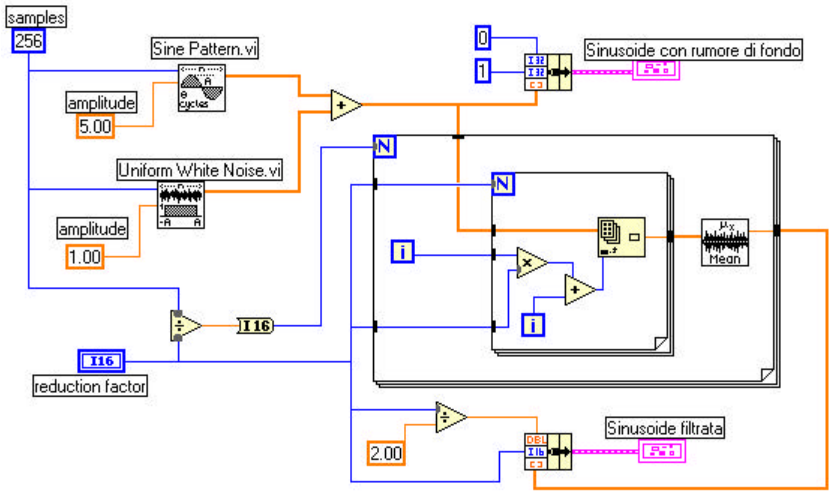
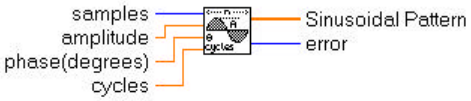


Fig. 14.3 Block Diagram di un codice che filtra un segnale in ingresso

In Fig.12.3 è riportato il Control Panel del suddetto codice ed in Fig.13.3 il relativo Block Diagram Descriviamo ora in breve la logica di tale programma.

Sul block diagram compaiono le due icone seguenti:

- **Sine Pattern**:



Sine Pattern.vi

Generates an array containing a sinusoidal pattern. If Sinusoidal Pattern is represented by the sequence Y, the VI generates the pattern according to the following formula:

y[i] = a * sin(x[i]), for i = 0, 1, 2, ..., n-1,

where
x[i] = 2 PI * i * k / n + ph * PI / 180,
a is the amplitude,
k is the number of cycles,
ph is the initial phase in degrees,
n is the number of samples.

- **Uniform White Noise**:

Uniform White Noise.vi

Generates a uniformly distributed pseudorandom pattern whose values are in the range [-a:a], where a is the absolute value of the amplitude.

Ciascuna di queste due icone genera un vettore: il primo contiene i valori della sinusoide, il secondo i valori del rumore. Dato che l'operatore somma esegue la somma degli elementi corrispondenti dei due vettori, il vettore risultante contiene quindi la sinusoide a cui è sovrapposto il rumore come risulta evidente osservando la figura 12.3: il diagramma superiore che compare nella suddetta figura visualizza tale andamento.

Il vettore contenente il segnale in presenza del rumore viene elaborato da due cicli **For Loop** concatenati.

Il loop esterno fa variare il suo contatore $\mathbf{I_e}$ tra il valore zero ed il valore risultante dal rapporto tra il numero di campioni **N** ed il fattore di riduzione **R**: nel caso in esame i campioni sono 256 ed il fattore di riduzione adottato è 4, cioè un punto della curva filtrata è la media algebrica di 4 punti consecutivi.

Il loop interno fa variare il suo contatore $\mathbf{I_i}$ tra il valore zero ed il valore **R-1** ed estrae dal vettore contenente la sinusoide con rumore un sottovettore contenente **R** elementi

Ad esempio:

- quando il contatore del loop esterno $\mathbf{I_e}$ vale 0, il contatore del loop interno $\mathbf{I_i}$ (che varia tra 0 e R-1 e nel presente esempio tra 0 e 3) genera ad ogni esecuzione del loop interno un numero **P** dato dalla seguente relazione: $\mathbf{P = I_e \times R + I_i}$ cioè un numero che varia tra 0 e 3.
- Se invece il contatore del loop esterno $\mathbf{I_e}$ vale 1, il contatore del loop interno $\mathbf{I_i}$ genera ad ogni loop un numero **P** che varia tra 4 e 7.

In altre parole ad ogni loop si genera un contatore che consente di estrarre dal vettore dei valori con rumore un sottovettore contenente 4 valori contenuti in 4 posizioni consecutive del suddetto vettore.

Al termine di ogni loop interno il VI **Mean** effettua la media aritmetica dei 4 valori estratti; questo valore è diagrammato ad ascissa pari alla media delle ascisse dei 4 punti sopra citati.

E' interessante osservare come durante l'esecuzione del programma l'efficacia del filtro diminuisca, come è facilmente intuibile, al decrescere del "Reduction Factor".

Un'ultima considerazione: se si impone che il "Reduction Factor" valga 0 il programma si arresta in quanto il rapporto tra 256 e 0 genera un NaN (Not A Number); se invece si impone che il "Reduction Factor" valga 1 il programma effettua la media su di un solo punto e qiondi l'azione filtrante risulta nulla.

Un codice "robusto" dovrebbe considerare questo caso imponendo una condizione IF – THEN che, ad esempio, sostituisca i valori 0 ed 1 con il valore 2 (in questo caso la media è effettuata ogni volta su due valori consecutivi estratti dal vettore principale.

Importanza del riferimento di terra in una misura di tensione

Come già detto in precedenza, tra il fenomeno fisico da studiare mediante acquisizione dati ed il sistema di acquisizione è sempre interposto un trasduttore che converte la grandezza da acquisire in un segnale di tensione.
Alcune grandezze fisiche sono di tipo **assoluto**, almeno da un punto di vista ingegneristico; la massa, ad esempio (i fisici non sarebbero assolutamente d'accordo con questa affermazione; tutto si complica e si comporta in modo bizzarro man mano che ci si avvicina alla velocità della luce. Pertanto nulla è assoluto all'occhio di un fisico).
La tensione è decisamente una grandezza **non assoluta**; per avere un significato essa richiede di avere sempre un riferimento. La tensione è sempre la misura di una differenza di potenziale tra due punti. Uno di questi due punti è, in genere, assunto come riferimento e gli è assegnato un valore nullo di tensione: quando diciamo che stiamo misurando un segnale di 3,55 V, dobbiamo anche precisare rispetto a che punto stiamo effettuando questa misura. Quando non si specifica tale punto, è sottinteso che si adotta come punto di riferimento la "famosa" **terra**.

In generale una moderna scheda di acquisizione dati può essere configurata in tre differenti modi:

1. **R**eferenced **S**ingle-**E**nded **(RSE)**
   In questo caso la misura è fatta rispetto alla terra del sistema; lo schema elettrico è illustrato in Fig. 6.3. La maggior parte delle apparecchiature che generano segnali di questo tipo è costituito da generatori di segnali connessi alla rete di alimentazione.

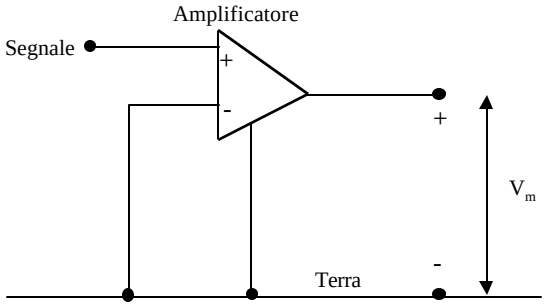


Fig. 6.3 Segnale tipo Referenced Single-Ended

2. **N**on **R**eferenced **S**ingle-**E**nded **(NRSE)**
   In questo caso il polo comune del segnale non coincide con la terra del sistema di acquisizione; lo schema elettrico è illustrato in Fig. 7.3. La maggior parte delle apparecchiature che generano segnali di questo tipo è costituito da trasformatori, batterie, ecc.

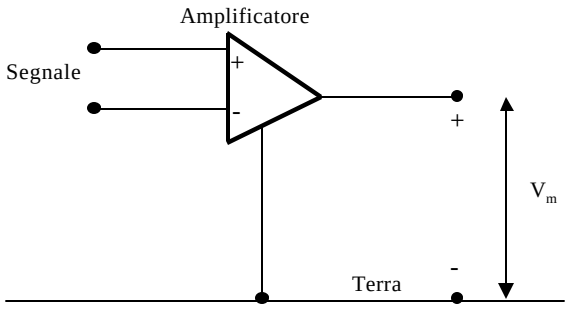


Fig. 7.3 Segnale tipo Non Reference Single-Ended

Si utilizzano gli ingressi single-ended quando si devono acquisire segnali elevati (superiori a 1 V) ed i cavi che collegano la sorgente del segnale all'hardware non superano i tre metri di lunghezza.
E' evidente che un disturbo che si sovrappone al segnale entrando sulla linea che collega il sensore all'amplificatore, si farà risentire sull'intera catena di acquisizione.

3. Segnali di tipo **Differenziale**

Un sistema di misura differenziale (detto anche bilanciato) presenta tre terminali:

a. un terminale che coincide con la terra comune,
b. un terminale al cui capo è presente la differenza di tensione ΔV^+ misurata rispetto a terra,
c. un terminale al cui capo è presente la differenza di tensione ΔV^+ invertita, che chiameremo ΔV^- .

Un semplice sommatore effettua l'operazione $(\Delta V^+ + \Delta V^-) = V^+ - V_G + V_G + V^- = V^+ - V^-$.

E' importante sottolineare che un sistema di misura differenziale è praticamente insensibile a quei disturbi che agiscono sulla terra innalzandone o abbassandone il livello di riferimento; infatti l'operazione di somma annulla gli effetti di eventuali variazioni.
Lo schema elettrico è illustrato in Fig. 8.3. La maggior parte degli amplificatori di rango elevato generano delle tensioni differenziali.

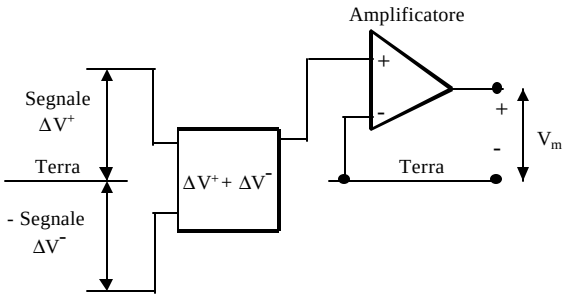


Fig. 8.3 Segnale tipo di tipo differenziale

Per meglio comprendere il comportamento di un segnale di tipo differenziale, ricorriamo all'esempio illustrato nelle figure 9.3 e 10.3.
Impostati il numero delle sinusoidi ed il numero dei punti per ogni sinusoide, un ciclo **for loop** calcola i vari punti della sinusoide; è evidente che il numero di punti totali da calcolare è dato dal prodotto del numero d'onde per il numero di punti per ogni onda. Ogni punto di questa sinusoide viene moltiplicato per un fattore 10; essa varia quindi tra i valori –10 e 10.
Moltiplicando per –1 tali valori, si ottengono i corrispondenti valori invertiti che sono in opposizione di fase rispetto alla sinusoide di origine. I diagrammi **Segnale A** e **Segnale B** riportano l'andamento di tali valori.
Contemporaneamente la funzione che genera numeri casuali (rappresentata mediante l'icona raffigurante due dadi affiancati) genera ad ogni for loop un numero compreso tra 0 ed 1. Questo numero viene a sua volta moltiplicato per un numero (**factor**) che può variare tra 0 e 10.
Questo numero casuale viene sommato al segnale origine ed al corrispondente segnale invertito. In tal modo alle due sinusoidi di partenza viene sovrapposto un rumore, che simula il rumore che possono catturare le due linee del segnale bilanciato. I relativi andamenti sono rappresentati nei grafici **Segnale + Rumore (A+R)** e **Segnale invertito + Rumore (B+R)** .
Infine vengono effettuate le seguenti differenze:

- Segnale (A) – Segnale invertito (B)
- Segnale (A) + Rumore (R) – [Segnale invertito (B) + Rumore (R)]

I loro relativi andamenti sono rappresentati nel grafico in basso a sinistra ed in quello in basso a destra della figura 9.3.

E' evidente che al crescere del valore del fattore **factor** risulta sempre più evidente l'effetto del rumore sul segnale. Tuttavia i diagrammi in basso a sinistra ed in basso a destra della suddetta figura risultano identici.

Questo semplice esempio mette in evidenza come un segnale bilanciato sia insensibile al "rumore" catturato dai cavi che collegano il sensore alla scheda di acquisizione.

E' altresì evidente che il rumore si annulla completamente se sulle due linee si sovrappone un rumore identico. Questo è vero solo a livello puramente teorico in quanto l'intensità del disturbo può differire lievemente tra i due collegamenti. E' evidente comunque che anche in questo caso meno favorevole di quello teorico, è sempre conveniente ricorrere a collegamenti bilanciati specie se la distanza tra il sensore e l'acquisitore supera i tre metri ed il segnale in uscita dal sensore è molto basso.

Sottolineiamo infine che invece nel caso di collegamenti single-ended tutto il rumore catturato lungo la linea di collegamento entra nella catena di acquisizione.

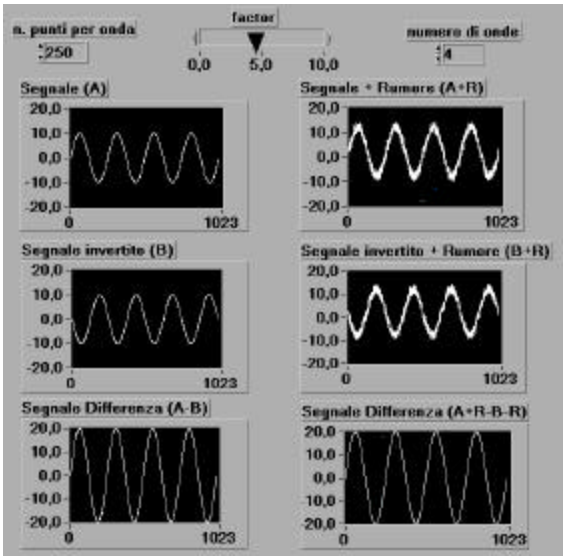


Fig. 9.3 Esempio di segnale tipo di tipo differenziale in presenza di rumore (control pannel)

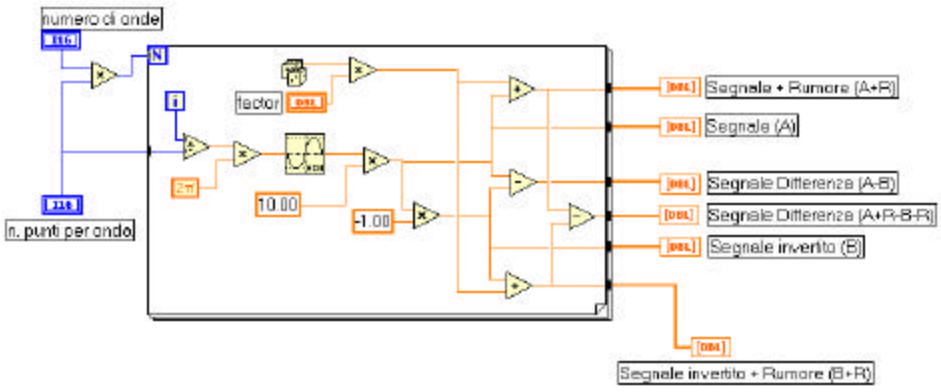


Fig. 10.3 Esempio di segnale tipo di tipo differenziale in presenza di rumore (block diagram)