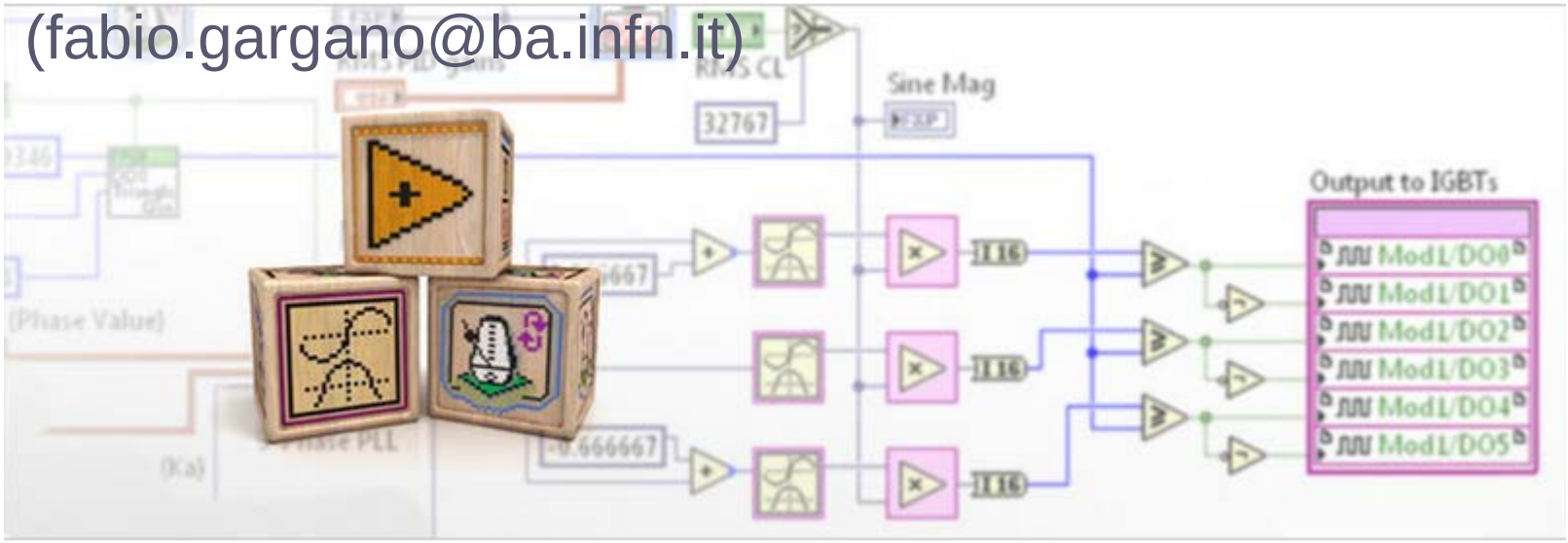
CORSO DI LABVIEW
SCUOLA DI DOTTORATO
XXXI CICLO

Dott. Fabio Gargano

(fabio.gargano@ba.infn.it)



Introduzione

- L'ambiente di sviluppo integrato **LabVIEW** (abbreviazione di **Laboratory Virtual Instrumentation Engineering Workbench**) nasce ufficialmente nel 1983, quando National Instruments, azienda statunitense specializzata nella produzione di hardware e software per l'acquisizione dati, decide di dotarsi di uno strumento software per il testing rapido dei propri prodotti hardware.
- Lo sviluppo di tale prodotto, inizialmente pensato per uso interno all'azienda, ebbe un tale successo che già nel **1986** vide la luce la prima versione commerciale, denominata LabVIEW 1.0, inizialmente pensata solo per sistemi Macintosh.
- Il contenuto innovativo di questo versatile ambiente di sviluppo software risiedeva nel tipo di linguaggio di programmazione utilizzato: il **linguaggio-G**, abbreviazione di **Graphich Language**. A differenza di altri linguaggi di programmazione, come ANSI C e BASIC, che si basano su editor testuali e parole chiave per la generazione del codice, il linguaggio-G usa un approccio di tipo grafico alla programmazione.

Introduzione

- Un programma o sottoprogramma G, denominato **VI** (**Virtual Instrument**, data la sua naturale predisposizione per i sistemi di acquisizione dati) non esiste sotto forma di testo, ma può essere salvato solo come fi le binario, visualizzabile e compatibile solo con LabVIEW.
- Gli algoritmi e le strutture dati in LabVIEW sono generati attraverso un editor grafico, che fa uso di icone, blocchi ed altri oggetti grafici, uniti attraverso uno o più elementi di connessione.
- Tale tipologia di approccio rende il codice estremamente simile ad uno schema a blocchi. I blocchi e gli altri oggetti grafi ci implementano in genere funzioni di alto livello, in modo che lo sviluppo di codice anche piuttosto complesso diventi estremamente rapido.
- Tale contenuto, fortemente innovativo, ha permesso ad NI di ottenere, nei primi mesi del **1990**, un brevetto a protezione dell'idea, rilasciato dalla US Patent Office. Due anni dopo, nel **1992**, viene divulgata la prima versione multipiattaforma, compatibile con Microsoft Windows, Mac OS e SunOS. Pochi mesi dopo viene realizzata anche una versione per sistemi Linux.

Introduzione

- Da allora lo sviluppo e la diffusione di LabVIEW è in continuo aumento, il supporto per l'hardware NI è stato esteso anche ad altri grandi produttori di hardware per l'acquisizione dati e recentemente, tramite l'uso di LabVIEW e del linguaggio-G, è possibile sviluppare software non solo per Personal Computer (che è sempre stato il target tipico di questo ambiente) ma anche per sistemi embedded, come sistemi a microcontrollore, FPGA, Palmari e PLC.
- Programmi sviluppati in LabVIEW possono essere trovati, ai nostri giorni, in svariati contesti applicativi: si va dall'automazione industriale alle applicazioni di monitoraggio ambientale, dalla programmazione di dispositivi embedded al campo biomedico. Recentemente è addirittura possibile utilizzare LabVIEW con i prodotti Mindstorms di Lego.
- Dal 2009 la numerazione passa da progressiva a numerazione su base annua.

Elenco delle versioni di Labview

- 1986 – LabVIEW 1.0, versione per Macintosh.
- 1990 – LabVIEW 2.0, versione con linguaggio compilato.
- 1992 – Rilasciate versioni per Microsoft Windows e per SunOS.
- 1993 – LabVIEW 3.0, prima versione multipiattaforma.
- 1997 – LabVIEW 4.0.
- 1998 – LabVIEW 5.0, prima versione per applicazioni real-time.
- 2000 – LabVIEW 6.0, Internet ready.
- 2003 – LabVIEW 7 Express, aggiunto supporto per PDA e FPGA.
- 2005 – Prima versione con supporto per DSP e dispositivi embedded.
- 2006 – LabVIEW 8.20, versione del ventennale.
- 2007 - LabVIEW 8.5 ottimizzazione dei sistemi multicore
- 2008 - LabVIEW 8.6, dedicato alle architetture e tecnologie parallele, FPGA e wireless
- 2009 – LabVIEW 2009
- 2010 – LabVIEW 2010
- 2011 – LabVIEW 2011
- **2012 – LabVIEW 2012**
- **2013 – LabVIEW 2013**
- **2014 – LabVIEW 2014**
- **2015 – LabVIEW 2015**

Download

- Versioni di prova di LabVIEW possono essere scaricate gratuitamente dal sito della National Instruments (www.ni.com), dove è possibile scaricare anche i vari driver per la strumentazione hardware e i tool aggiuntivi.
- Tali versioni contengono tutte le funzionalità delle versioni commerciali, ma dopo **6 giorni** devono essere acquistate o divengono inutilizzabili trattandosi di versioni trial.
- E' possibile chiedere una licenza Student valida 6 mesi dal sito:
  - https://decibel.ni.com/content/docs/DOC-30610

Programmazione grafica

- La caratteristica più importante di LabVIEW è l'uso del **linguaggio-G** per lo sviluppo del codice, in luogo dei più tradizionali linguaggi di programmazione text-based. Ad un esperto programmatore, che utilizza frequentemente C, Visual Basic o Java, questo tipo di approccio potrebbe apparire quantomeno **inconsueto** e ci si potrebbe chiedere che livello di controllo delle applicazioni si riesca ad ottenere e soprattutto, quali vantaggi si possono ricavare dalla programmazione grafica.
- Il livello di controllo dell'applicazione realizzata si spinge fino al singolo byte (e volendo fino al bit) di informazione, tramite l'uso di blocchi o funzioni specifici. Ciò è reso possibile dalla grande varietà di blocchi presenti, che vanno dalla manipolazione di stringe e stream di dati, fino alle operazioni orientate al byte e al bit.
- Naturalmente, come per ogni linguaggio di programmazione, il livello di controllo dipende dall'**esperienza** del programmatore nell'uso delle varie funzioni messe a disposizione dall'ambiente. Comunque, per venire incontro anche agli utenti più esigenti, all'interno di un programma LabVIEW possono essere integrati (sotto forma di VI) anche DLL o eseguibili scritti in C o in altri linguaggi di programmazione, in modo da rendere l'ambiente di sviluppo ancora più flessibile. I vantaggi ottenuti da questa filosofia di programmazione sono innumerevoli.

Programmazione grafica

- Utilizzando il linguaggio-G è possibile realizzare applicazioni molto complesse unendo tra loro pochi blocchi e strutture grafiche di controllo ed interconnessione.
- Poiché i dati possono scorrere contemporaneamente attraverso blocchi e strutture di connessione non consecutive, viene implicitamente realizzato il **multithreading**, senza bisogno di esplicita gestione da parte del programmatore.
- È importante sottolineare che, nonostante il linguaggio supporti nativamente il multithreading, è possibile forzare un'esecuzione di tipo sequenziale, tramite l'uso di opportune strutture di controllo.
- Nonostante l'approccio di alto livello (si consideri che in una scala di valori il linguaggio-G è sicuramente più astratto del Visual Basic), il tempo di esecuzione del codice è paragonabile a quello di un'eseguibile scritto in ANSI C.

L’ambiente di sviluppo

- L’ambiente di sviluppo comprende tutta una serie di **strumenti** che permettono la realizzazione di un **progetto** completo, che porta, al termine del processo, alla realizzazione di un eseguibile o, eventualmente, di un pacchetto di installazione.
- Gli strumenti forniti dall’ambiente di sviluppo sono:
  - una finestra di programmazione, attraverso la quale può essere generato un VI (composta da più elementi, che vedremo nel dettaglio in seguito)
  - una finestra di *Project Manager*, per la gestione del progetto e la generazione di eseguibili ed installer;
  - una ricca dotazione di librerie ed esempi di codice.
- Tutti questi elementi sono simultaneamente accessibili una volta lanciato LabVIEW (da qui la definizione di *ambiente di sviluppo integrato*) e possono essere utilizzati per sviluppare l’applicazione finale.

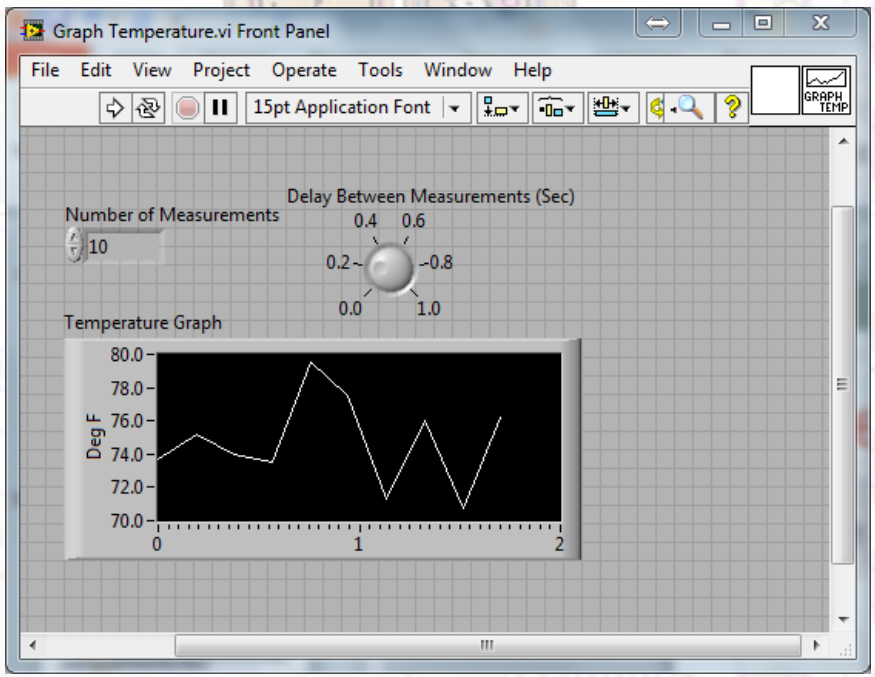
Struttura di un VI

- Un programma sviluppato in LabVIEW prende il nome di VI (*Virtual Instrument*).
- La struttura di un VI comprende tre elementi principali: un *front panel*, un *block diagram* ed una icona/pannello connettori.
- Il *front panel* è l'interfaccia grafica di un programma LabVIEW (la denominazione deriva dalla somiglianza con i pannelli frontali degli strumenti reali) e rappresenta la porzione dell'applicazione che l'utente finale *vede* realmente quando il programma è in esecuzione.
- Il *block diagram* è invece il foglio di lavoro, ossia l'editor che il programmatore usa per la stesura del codice ed è naturalmente invisibile quando l'applicazione è in esecuzione.
- Se il programma è lanciato all'interno dell'ambiente di sviluppo, il *block diagram* può essere visualizzato per scopi di debug ed è possibile inserire indicatori (probe) e *breakpoint*.

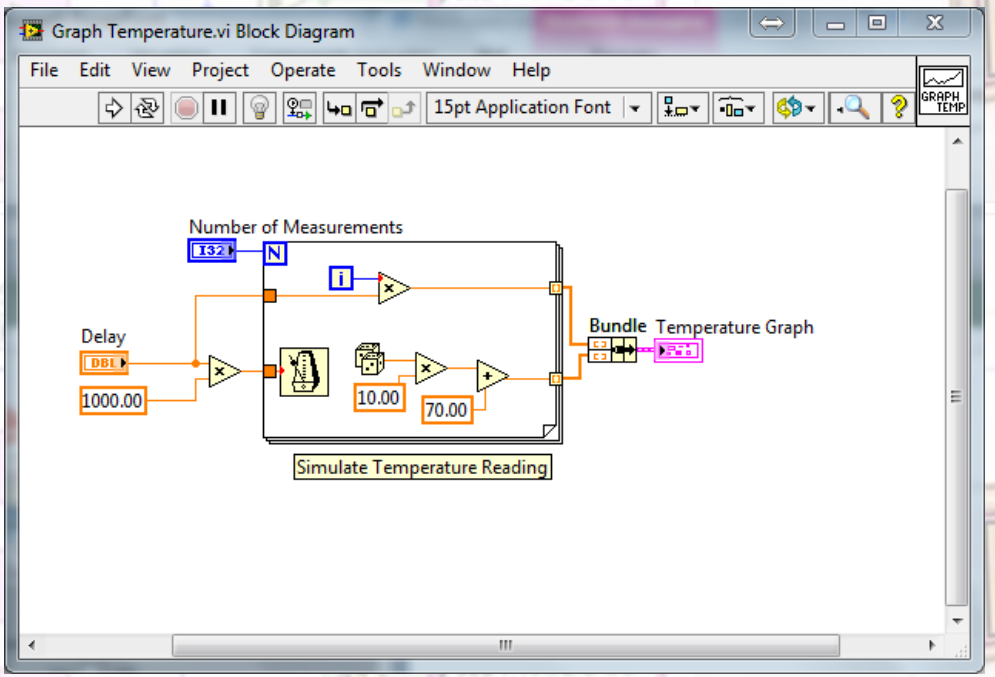
Struttura di un VI

- un VI può essere rappresentato anche sotto forma di icona grafica, quando il VI stesso viene riutilizzato all'interno di un altro VI, come si fa in altri linguaggi di programmazione con le subroutine o le funzioni di libreria.
- Quando un VI viene utilizzato all'interno di un altro, si chiama s*ubVI*; allo scopo bisogna definire quali variabili accoglie in ingresso e quali presenta in uscita al termine dell'elaborazione, cosa che avviene tramite l'uso del pannello connettori.
- Un VI può avere un numero indefinito di ingressi ed uscite. Notate che di fatto anche le funzioni di libreria di LabVIEW sono dei *subVI*, con la differenza che non tutti presentano un *block diagram* e un *front panel*.

Struttura di un VI



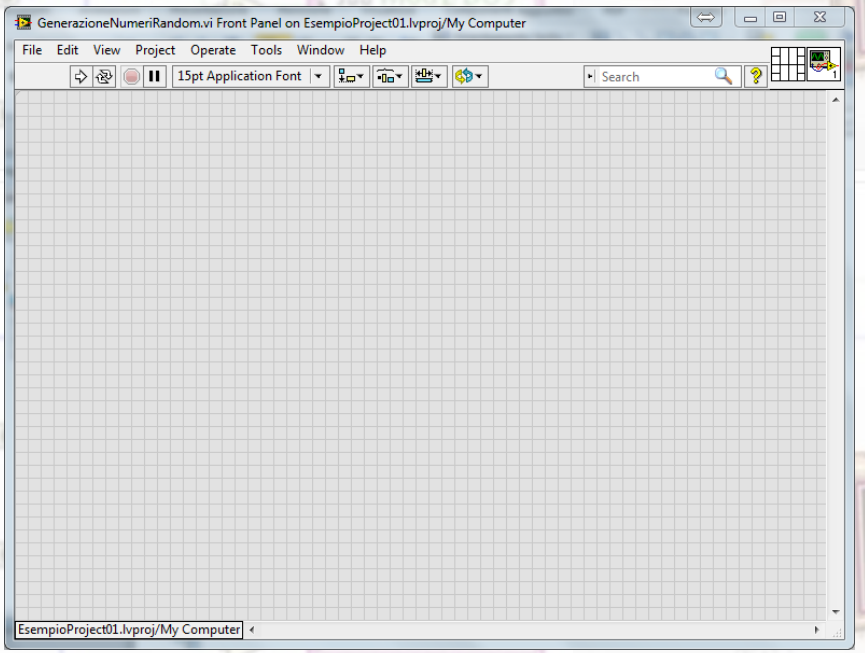
Front Panel



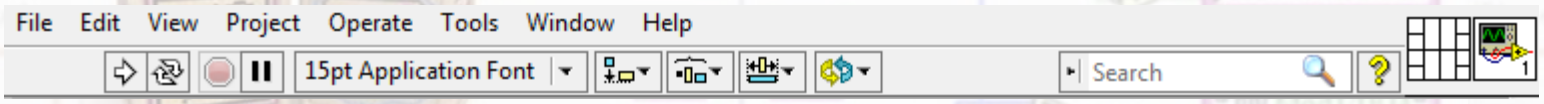
Block Diagram

Front Panel

- Il *front panel* è l'interfaccia utente di un programma LabVIEW.
- All'interno di un *front panel* possono essere inseriti controlli, indicatori e grafici che poi è possibile gestire, dal *block diagram*, come sorgenti di input e di output.
- Un *front panel* contiene
  - un foglio di lavoro,
  - una *toolbar*
  - una *control palette*.
- Il foglio di lavoro è la parte in grigio, che può essere utilizzata per l'inserimento dei vari oggetti grafi ci che compongono l'interfaccia. L'area di lavoro è virtualmente illimitata, dato che possono essere utilizzate *scrollbar* orizzontali e verticali per spostarsi nelle varie direzioni.



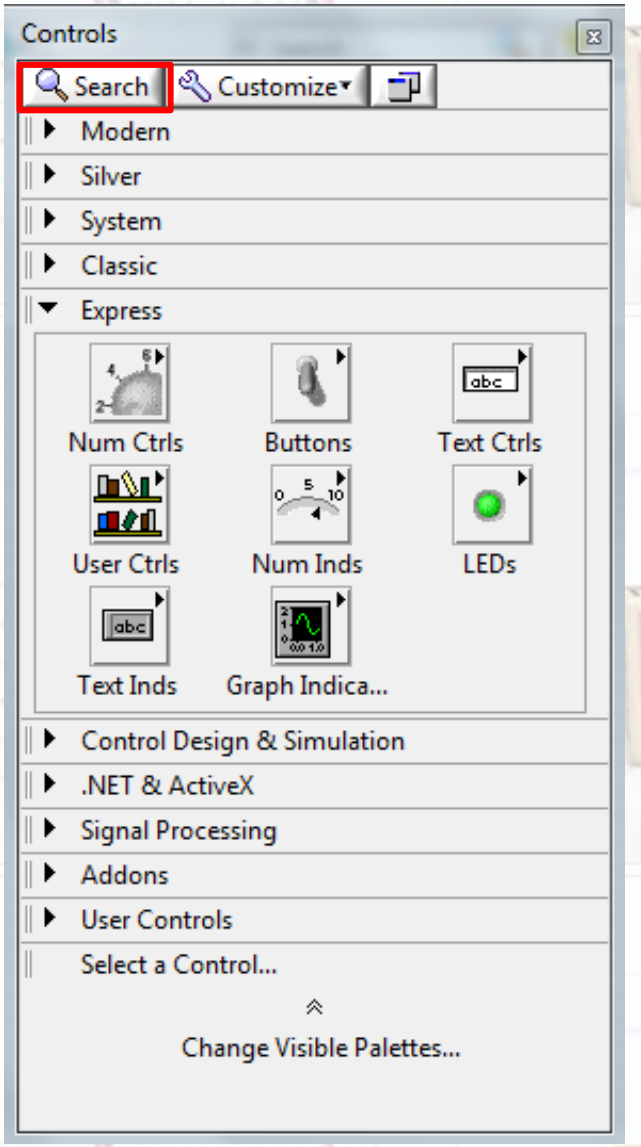
Front Panel



- La *toolbar* è collocata in alto, immediatamente sotto le voci del menu; contiene i comandi principali per la gestione dell'applicazione e per il debug.
  - tasto con una freccia bianca come icona questo pulsante è denominato ***Run***
    - serve a mandare in esecuzione il VI. Se il VI viene lanciato all'interno dell'ambiente di sviluppo, si attiva automaticamente la modalità di debug ed il *block diagram* può essere aperto per visualizzare lo scorrimento dei dati, sistemare *breakpoint*, lanciare esecuzioni step-by-step ed altro ancora.
  - pulsante con due frecce indicanti un ciclo ripetuto, denominato ***Run continuosly***
    - serve a generare un'esecuzione continua del codice, molto utile in fase di debug.
  - pulsante ***Abort Execution***
    - serve ad arrestare immediatamente l'esecuzione del programma.
  - il pulsante ***Pause***
    - serve per mettere in pausa l'esecuzione del codice.
  - I pulsanti successivi
    - servono a settare dimensioni, posizionamento e stile dei font, nonché all'allineamento degli oggetti grafici sul pannello.

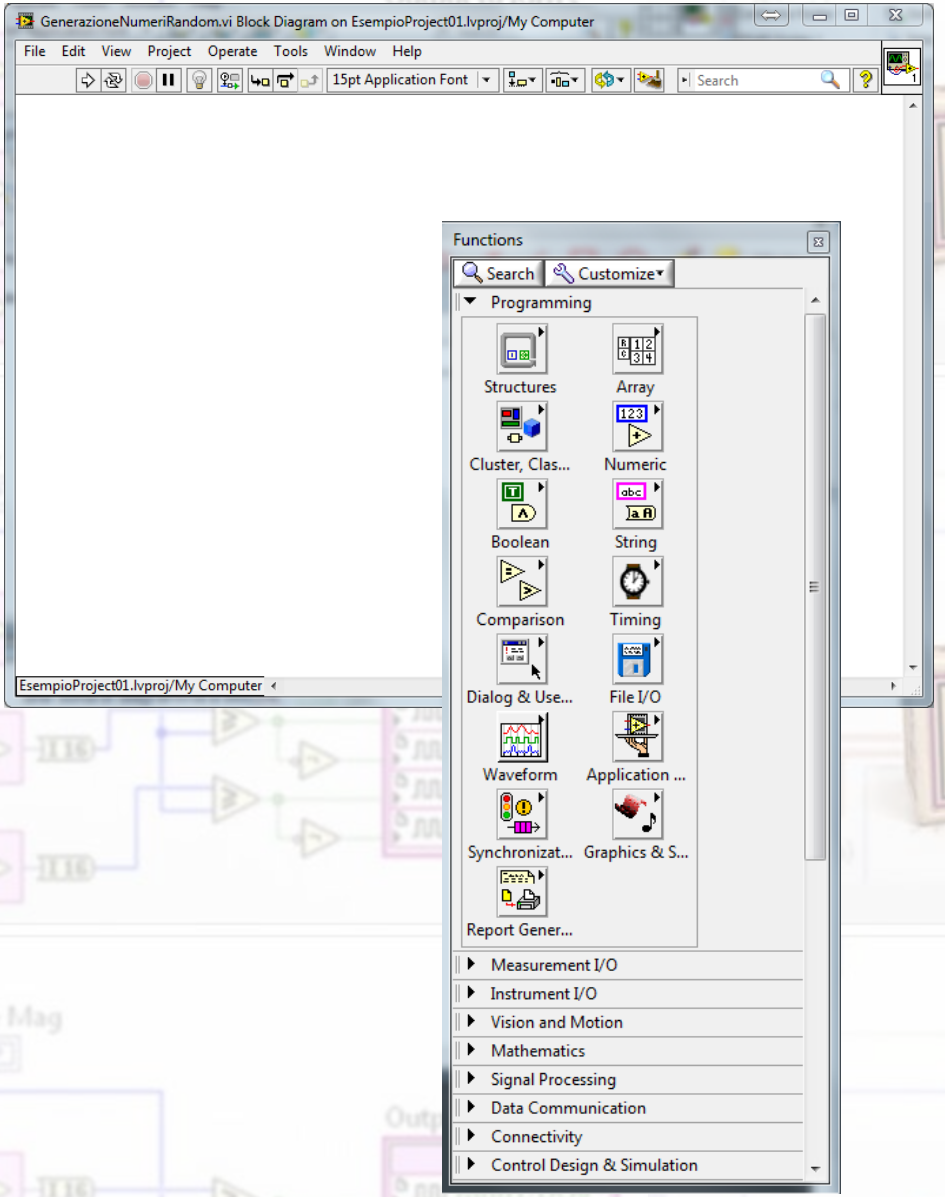
Front panel

- La **control palette** è uno degli strumenti più importanti presenti in quest'ambiente di sviluppo;
- Contiene tutti gli oggetti grafici che possono essere inseriti all'interno del pannello:
  - indicatori numerici, testuali
  - manopole
  - LED
  - interruttori, pulsanti
  - grafici
  - …
- Tutti gli elementi sono raggruppati per **categorie**, come ad esempio controlli ed indicatori numerici, controlli ed indicatori booleani, grafici, controlli ed indicatori testuali, e via di seguito.
- Le categorie di oggetti sono innumerevoli, quindi le prime volte si potrà avere l'impressione di perdersi all'interno dei vari sottomenu, ma con un po' di esperienza si riesce facilmente a trovare ciò che si sta cercando.
- Per trovare un oggetto all'interno dei vari menu è possibile utilizzare la funzione di ricerca.



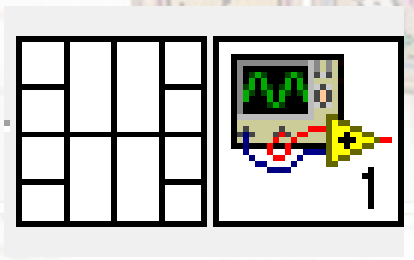
Block Diagram

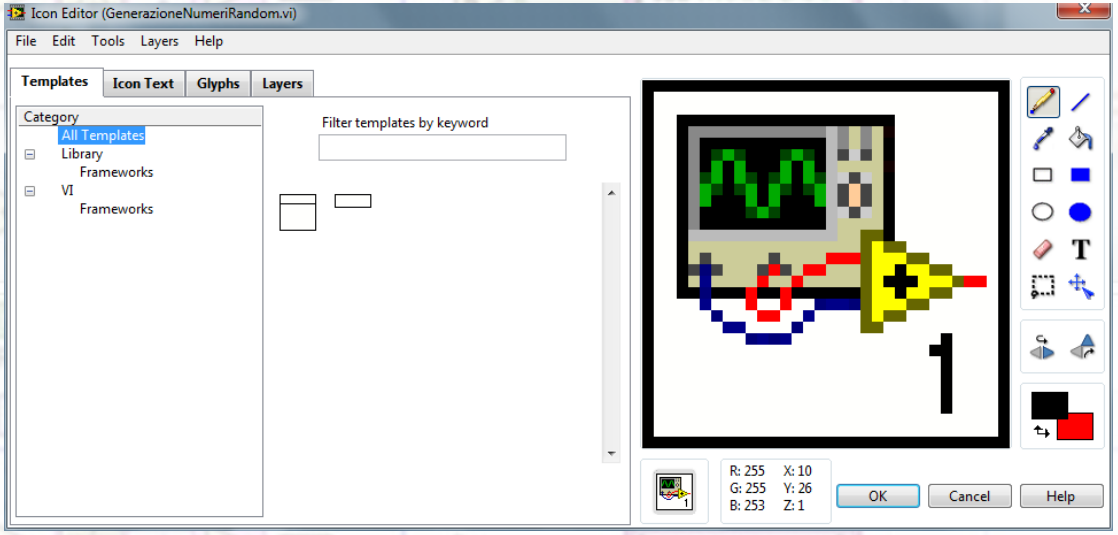
- Ogni front panel è sempre accompagnato da un **block diagram**, che costituisce l'interfaccia di programmazione.
- Diversamente dagli editor testuali viene utilizzato il linguaggio-G come linguaggio di programmazione.
- Come vedremo il codice assomiglierà ad una sorta di diagramma a blocchi.
- Il block diagram contiene
  - un'area di lavoro
  - una tool bar
  - una functions palette.
- La toolbar contiene in parte gli stessi comandi accessibili dal front panel, con la medesima funzione, più alcuni strumenti di debug.
- La functions palette, che invece degli oggetti grafi ci contiene subVI, strutture di controllo, costanti, variabili ed altri elementi utili alla generazione del codice.



Icona

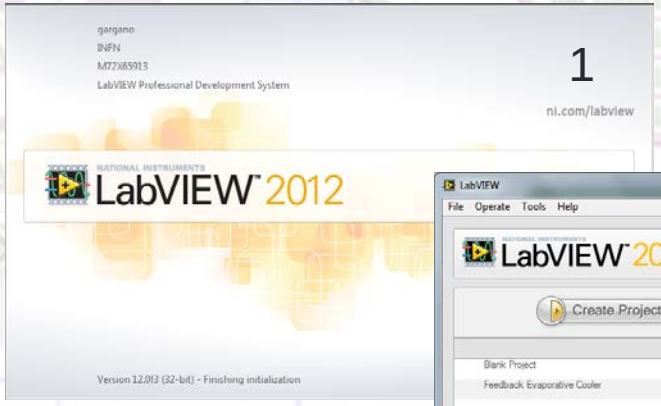
- Ogni VI è caratterizzato da un'icona, che diventerà quella dell'applicazione nel caso tale VI diventi un eseguibile, oppure la sua rappresentazione grafica su un *block diagram* nel caso venga utilizzato come subVI.
- L'icona del VI è visibile nell'angolo in alto a sinistra del *front panel*. Quest'icona è interattiva: cliccando su di essa è possibile editarla o creare dei connettori (i connettori sono necessari solo nel caso il VI debba essere utilizzato come subVI).

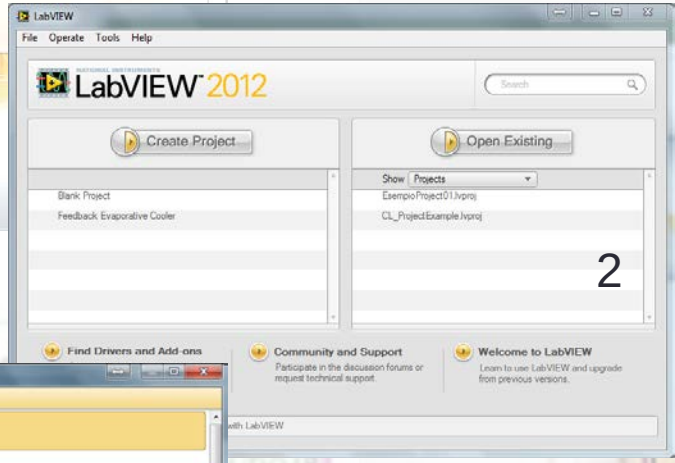


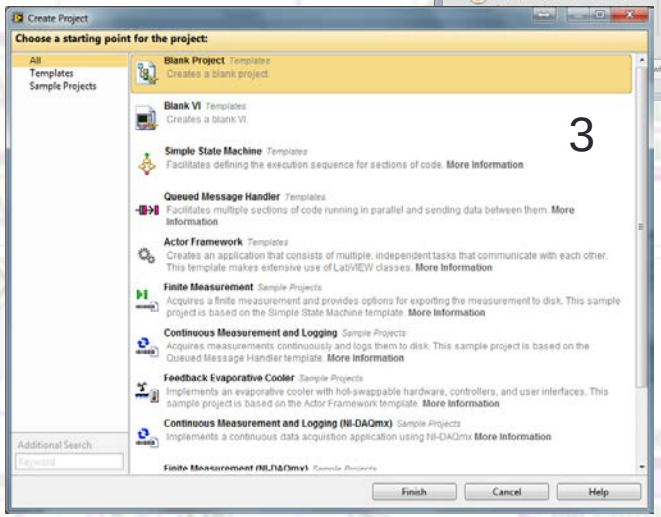


Attivare il programma

1. Fare click sull'icona del programma ed attendere che esso venga caricato
2. Selezionare **Create Project** nella prima schermata
3. Selezionare **Blank VI** nella schermata di selezione e quindi premere il tasto *finish*
4. Ora si è pronti a lavorare sul front panel e sul block diagram del nuovo VI







Strutture dati in LabVIEW

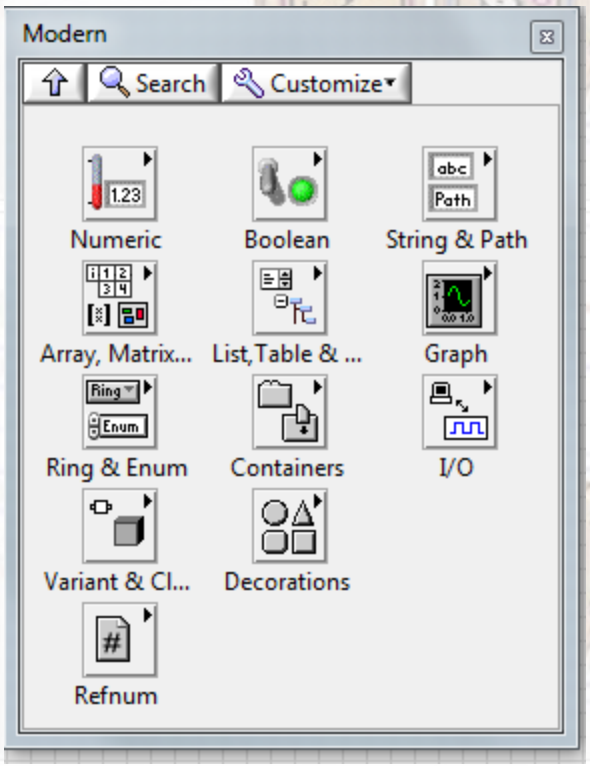
- Come ogni altro linguaggio di programmazione, LabVIEW gestisce diversi tipi di dati e dispone dei più comuni tipi di strutture dati.
- I più importanti tipi di dati gestiti in LabVIEW sono:
  - ***Interi***; con segno o senza, a 8, 16 e 32 bit;
  - ***Floating Point***; complessi o meno, a precisione singola, doppia ed estesa;
  - *Stringhe*; questa tipologia racchiude sia i caratteri singoli, che gli insiemi di caratteri, ossia appunto, le stringhe;
  - ***Boolean***; dati di tipo booleano (true e false).
- Oltre a questi tipi di dati, sono disponibili delle strutture dati. Le più comuni strutture dati in LabVIEW sono le seguenti:
  - ***Array***: insieme di dati dello stesso tipo, con indice. Gli array possono essere sia monodimensionali (vettori) che bidimensionali (matrici)
  - ***Cluster***: il tipo *cluster* è di fatto un macrocontenitore, all'intero del quale vengono inseriti insiemi di dati indicizzati (come per gli *array*), ma di diversi tipi. Un tipico esempio di *cluster* è quello all'interno del quale vengono veicolati i messaggi di errore, che per loro natura contengono dati di carattere diverso (ad esempio un codice errore di tipo *string* ed un segnalatore di errore di tipo *boolean*).
- Per aiutare il programmatore nell'identificazione di un particolare tipo di dato veicolato da un cavo tra due o più blocchi, LabVIEW usa le icone ed i colori. Le associazioni tra i tipi di dati elencati precedentemente ed i colori sono riportate nella **Tabella**

Tipo di dato	Colore
Interi	Blu
Floating-point	Arancio
Stringhe	Rosa
Boolean	Verde

Controlli e indicatori

- In LabVIEW è possibile inserire una vasta gamma di controlli e indicatori, che sono rappresentati sotto forma di oggetti grafi ci nel front panel e sotto forma di variabili nel block diagram. Gli oggetti grafi ci che è possibile inserire richiamano (nella forma e nel nome) i controlli e gli indicatori che tipicamente possono essere trovati su un pannello frontale di uno strumento da laboratorio.
- Gli oggetti grafi ci più comunemente utilizzati in LabVIEW sono i seguenti:
  - pulsanti, LED e Switch: associati a varibilidi tipo *boolean*;
  - controlli ed indicatori numerici, barre e *knob* (manopole): associati a varibili numeriche, sia di tipo intero che *floating-point;*
  - controlli ed indicatori di tipo stringa: associati a variabili di tipo stringa;
  - tabelle: associate ad *array* sia monodimensionali che bidimensionali;
  - grafici: possono essere associati a variabili numeriche o ad *array*.

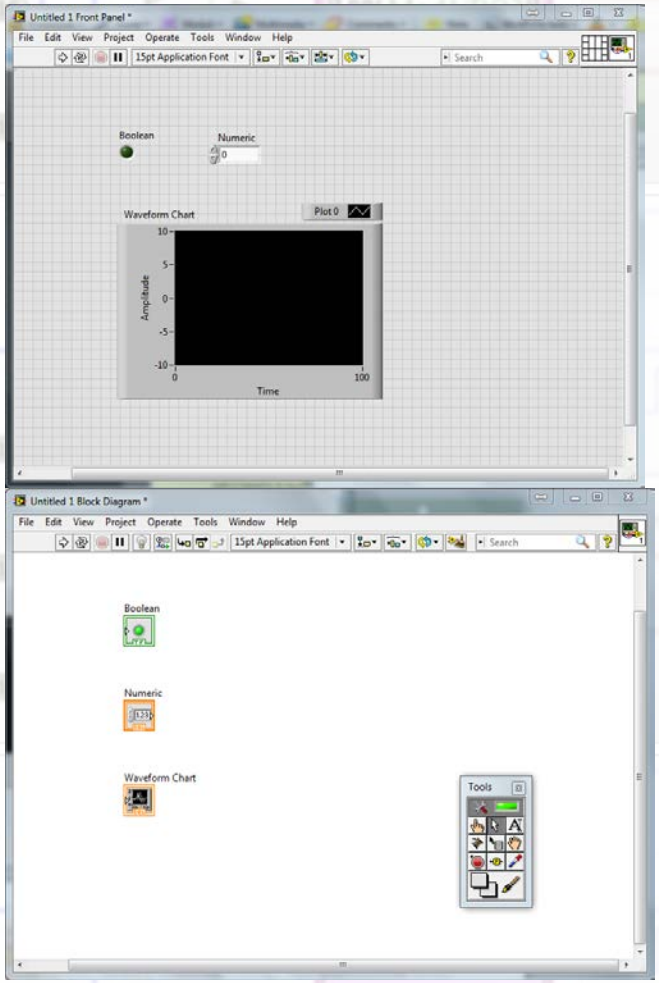
Controlli e indicatori



- Sul front panel del VI appena aperto accediamo alla control palette, premendo il pulsante destro del mouse in una zona qualsiasi dell'area di lavoro.
- La control palette contiene una serie di oggetti grafici, divisi per sezione.
- Per inserire un oggetto bisogna navigare con il puntatore del mouse all'interno dei menu (i quali si espandono automaticamente al passaggio del puntatore) fino ad individuare l'oggetto desiderato, selezionarlo e trascinarlo con il mouse nella posizione voluta, quindi premere il tasto sinistro per posizionarlo.

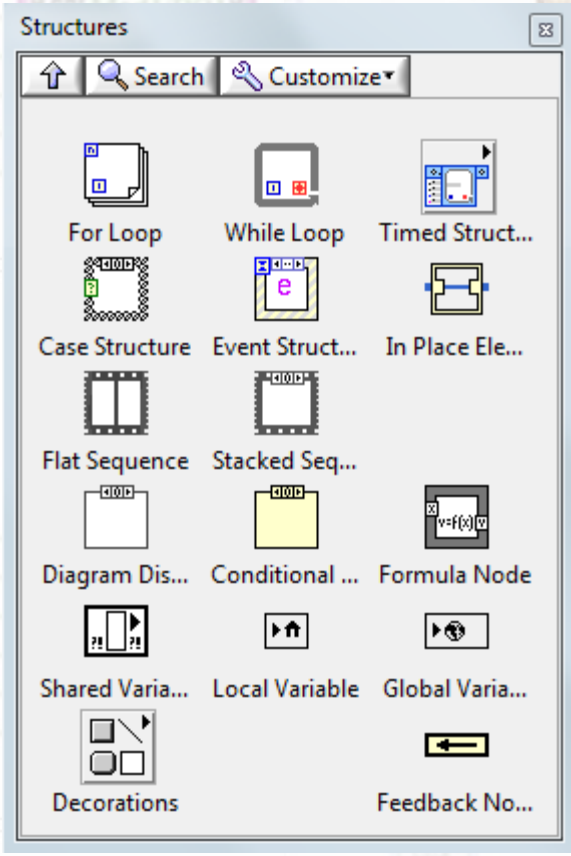
Controlli e indicatori

- Inseriamo i seguenti oggetti:
  - un LED rotondo, percorso *Boolean/Round LED*;
  - un controllo numerico, percorso *Numeric/Numeric Control*;
  - un grafico di tipo chart, percorso *Graph/Waveform Chart.*
- Notate che l'inserimento dei tre oggetti grafi ci sul front panel ha prodotto la generazione automatica, da parte dell'ambiente, di tre variabili associate:
  - la variabile *boolean* (associata al LED),
  - la *numeric* (associata al controllo numerico) e
  - la *waveform chart* (associata al grafico).
- I nomi vengono assegnati automaticamente (ed in maniera univoca) dall'ambiente, ma possono essere modificati a piacimento dal programmatore.



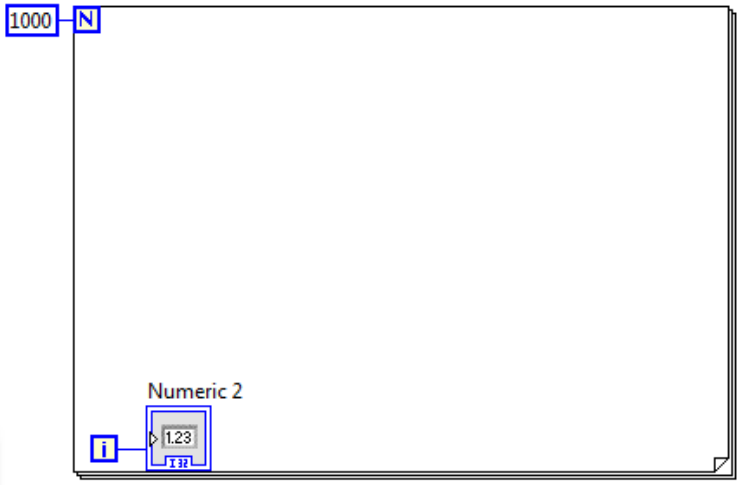
Strutture di controllo

- Come ogni altro linguaggio di programmazione, LabVIEW dispone di una serie di strutture di controllo, costituenti gli strumenti che il linguaggio mette a disposizione del programmatore per la realizzazione degli algoritmi.
- Le strutture utilizzate sono quelle classiche dei linguaggi di programmazione; si tratta quindi di **cicli** e **strutture di selezione multipla**.
- Oltre a queste strutture di base, l'ambiente ne mette a disposizione altre, rese necessarie dalla particolare tipologia di programmazione, come si vedrà più chiaramente nelle lezioni successive.
- Le strutture di controllo, essendo elementi relativi alla programmazione, possono essere utilizzate solo all'interno di un block diagram e non sono accessibili sui front panel.
- Sono tutte raggruppate sotto la voce *Structures* della control palette.



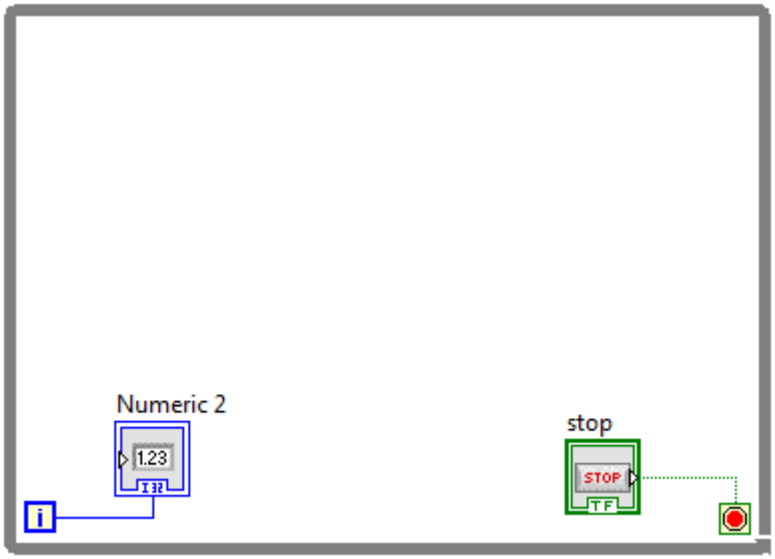
Ciclo **for**

- È il classico ciclo a conteggio, che ritroviamo normalmente anche nei tradizionali linguaggi di programmazione basati su stringhe di testo.
- Tutto il codice grafico contenuto all'interno dei margini del ciclo for viene ripetuto finché l'indice di conteggio non raggiunge il valore indicato dal programmatore.
- L'icona che lo rappresenta è facilmente riconoscibile, ed è costituita da tre fogli sovrapposti, ad indicare la ripetizione del ciclo per un numero stabilito di volte.
- La struttura dispone di due parametri di controllo: in alto a sinistra troviamo il parametro ***N*** (intero) ed in basso a sinistra il parametro ***i*** (sempre intero).
  - ***N*** è un ingresso cui può essere collegato sia un valore costante, sia una variabile, ed indica il numero di volte per cui verrà ripetuto il ciclo, mentre
  - la variabile ***i*** rappresenta il contatore del ciclo.



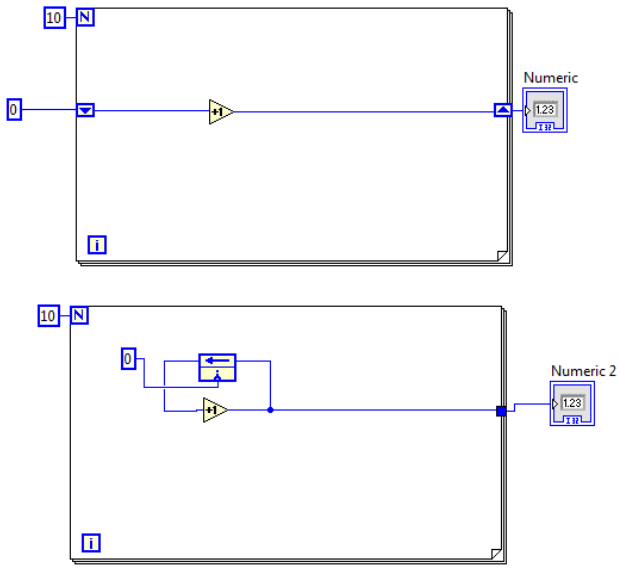
Ciclo **while**

- Il ciclo *while* , come il ciclo *for*, è presente in molti altri linguaggi di programmazione ed è caratterizzato da un'esecuzione ciclica del codice contenuto al suo interno finché non si verifichi una certa condizione di uscita, che viene controllata ad ogni esecuzione del ciclo stesso.
- Anche l'icona del ciclo *while* è piuttosto intuitiva, in quanto costituita da una freccia disposta in modo da richiudersi su se stessa.
- Il ciclo dispone di un ingresso (condizione di uscita dal ciclo) e di una uscita (indice); continua ad eseguire il codice contenuto al suo interno finché la condizione di uscita non viene verificata.
- In figura , la condizione di uscita è stata collegata ad un controllo booleano (un tasto di stop) sul front panel. In questo modo il ciclo *while* viene ripetuto fin quando non si interviene sul front panel premendo il pulsante di stop.



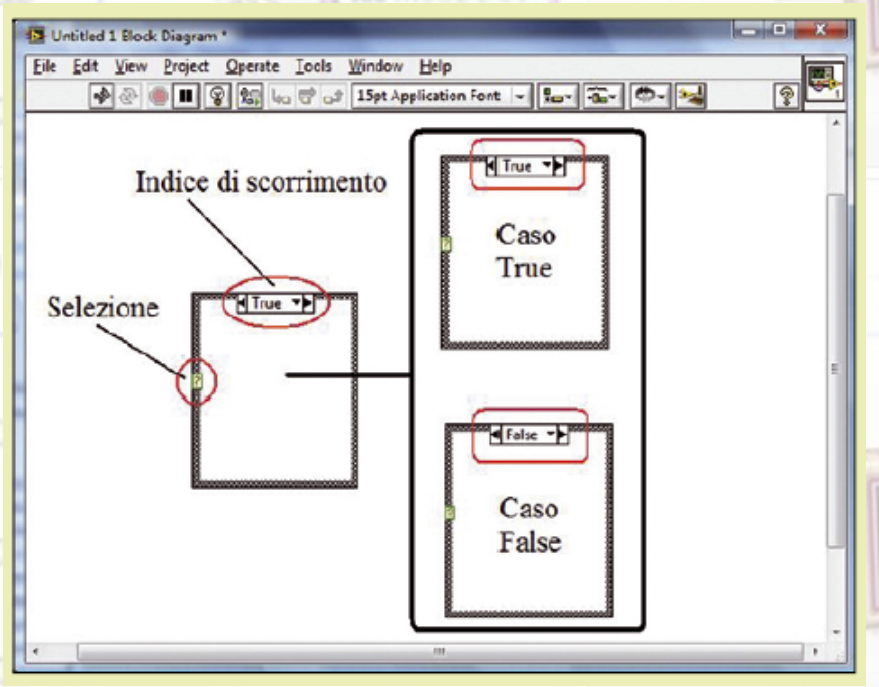
Shift Register

- Sui bordi delle strutture di iterazione while e for è possibile inserire degli shift register. Uno shift register è una variabili locale che trasferisce il valore prodotto da un ciclo di iterazione al successivo. Uno shift register può contenere qualunque tipo di dato numerico semplice o array, Booleano, stringa, ecc.
- Gli *shift* devono essere inizializzati prima del loro utilizzo altrimenti il dato in essi memorizzato è il valore memorizzato per *default* all'inizio del ciclo. Per questo è necessario collegare al terminale una costante del tipo di dato da memorizzare (numerico scalare, array, stringa, ecc.) ed inizializzarla al valore richiesto.
- Nelle versioni attuali di LabVIEW possono essere realizzati in due modi differenti come indicato in figura



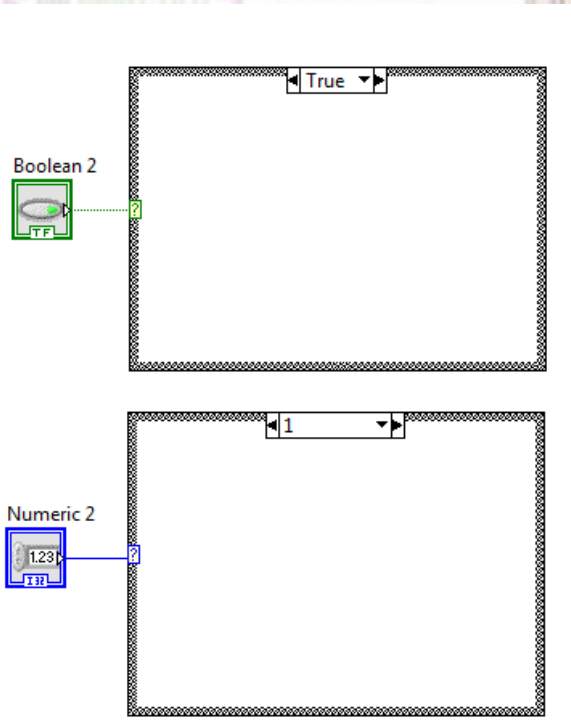
Struttura **Case**

- Come altri linguaggi di programmazione, LabVIEW dispone delle cosiddette strutture di selezione multipla, ossia di strutture che permettono di eseguire due porzioni differenti di codice in funzione di una variabile di selezione.
- In figura è rappresentata un struttura di tipo *Case*; come vedete, essa è dotata di un indice di scorrimento che permette di scorrere tra i vari casi.
- Per impostazione predefinita, la struttura, prevedendo un ingresso di selezione di tipo booleano, presenta due casi: *True* e *False*.
- Oltre all'indice di scorrimento, la struttura è dotata di un **ingresso**, rappresentato con il punto interrogativo e colorato di verde (ad indicare, in questo caso, un ingresso di tipo *boolean)*.
- Su quest'ingresso va cablata la variabile di selezione, che può essere un pulsante, un interruttore o anche un controllo di tipo numerico; in base allo stato di tale variabile di selezione verrà eseguito il caso associato.



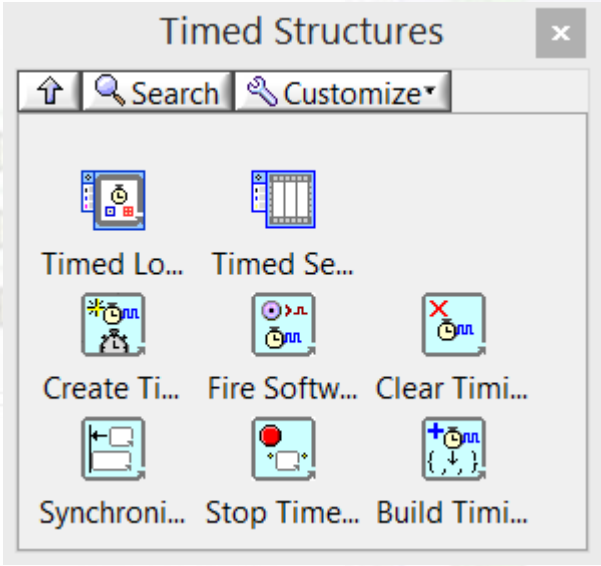
Struttura **Case**

- Nel caso sull'ingresso venga collegata una variabile di tipo differente dal *boolean* (ad esempio un controllo numerico), la struttura case si adatta al differente tipo di input, con un comportamento simile alla *switch…case* del C.
- Quanto descritto è un tipico esempio di **polimorfismo**, in cui una funzione cambia in accordo al tipo di controllo o variabile ad essa collegata.
- Nell'ipotesi che la variabile presenti un numero di casi superiore a due, per aggiungerne altri bisogna cliccare con il tasto destro del mouse sul bordo e selezionare una delle due opzioni *Add Case After* o *Add Case Before.*



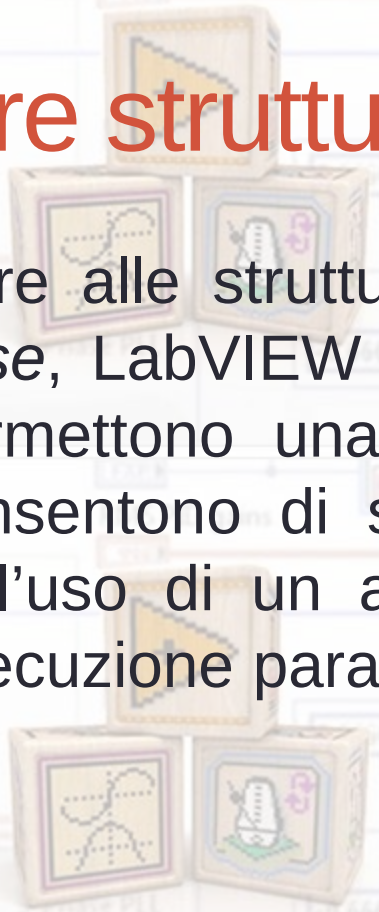
Timed Structures

- Executes one or more subdiagrams, or frames, sequentially each iteration of the loop at the period you specify. Use the Timed Loop when you want to develop VIs with multirate timing capabilities, precise timing, feedback on loop execution, timing characteristics that change dynamically, or several levels of execution priority. Right-click the structure border to add, delete, insert, and merge frames.

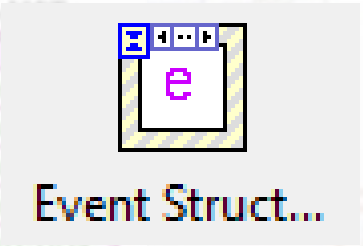


Altre strutture di controllo

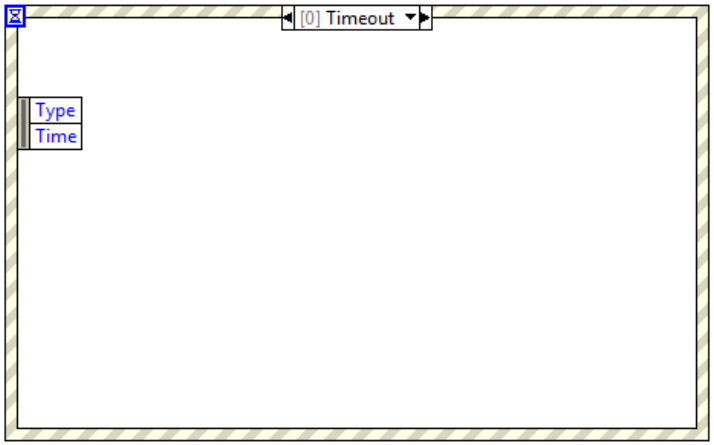
- Oltre alle strutture classiche, come la *for*, la *while* e la *case*, LabVIEW introduce altre strutture di controllo, che permettono una migliore gestione del flusso di dati e consentono di superare alcuni limiti intrinseci derivanti dall'uso di un ambiente di programmazione grafico ad esecuzione parallela delle istruzioni.



Struttura Event Case

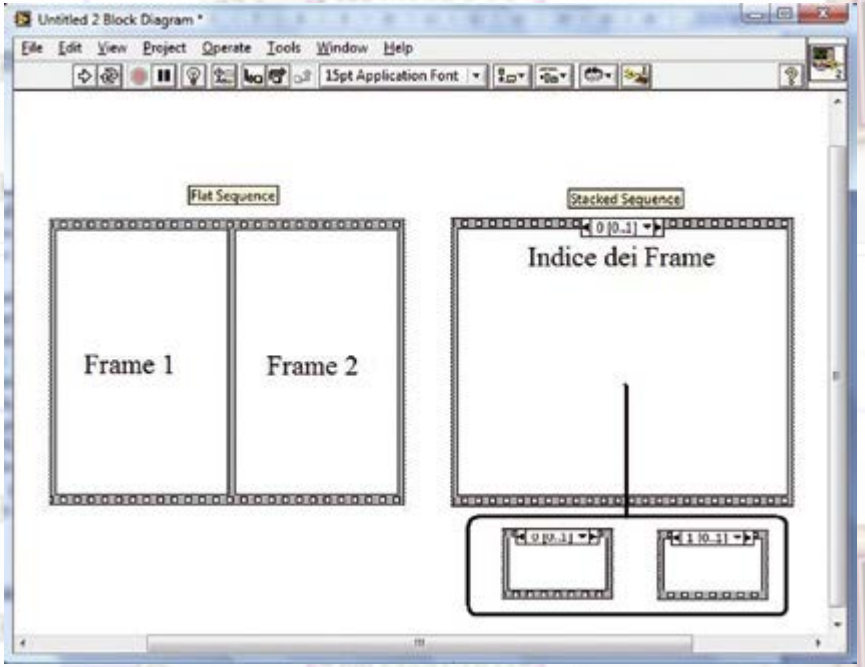


- La event case è una struttura *case* gestita ad eventi.
- Attende fino al verificarsi di un determinato evento e non appena questo si verifica lo gestisce con il caso appropriato. In sostanza si tratta di un *case* gestito ad **interrupt**.
- L'uso di questa struttura permette di modificare l'esecuzione dei programmi LabVIEW, introducendo la possibilità di generare algoritmi event driven.
- Nella struttura Event Case è sempre presente, per impostazione predefinita, il caso **Timeout**, che viene gestito appena trascorre un tempo indicato dal programmatore senza alcun tipo di attività sul pannello frontale.
- Il valore di attesa viene cablato, in millisecondi, sul simbolo della clessidra posto in alto a sinistra della struttura stessa.



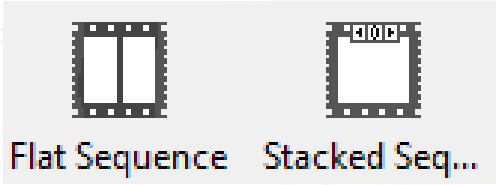
Struttura Sequence

- Un altro esempio di struttura che permette di modificare il comportamento nativo di LabVIEW è la struttura *Sequence*.
- LabVIEW è un ambiente a **flusso di dati** e non ad esecuzione sequenziale, come è invece nella maggior parte dei linguaggi di programmazione text-based.
- Ciò vuol dire che l'esecuzione del codice di un VI avviene in maniera parallela, all'interno di uno stesso foglio di lavoro.
- Un tale modo di operare può essere un vantaggio (si pensi infatti che in questo modo può venire realizzato spontaneamente il **multithreading**, cosa che invece è molto più difficoltosa da ottenere nei linguaggi classici), ma può essere anche un limite, nel momento in cui si desidera che una certa porzione di codice venga eseguita in maniera sequenziale.
- Per ovviare a questo problema si usa la struttura *Sequence*.
- Di questa tipologia di struttura,in LabVIEW, esistono due versioni: la*stacked sequence* e la *flat sequence*.
- Ognuna di queste strutture è frazionata in uno o più frame

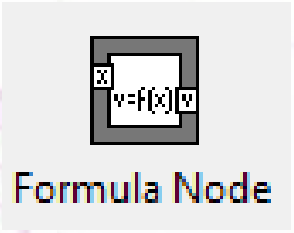


Struttura Sequence

- Una sequence (sia essa di tipo flat o stacked) permette di forzare un'esecuzione sequenziale del codice, ossia il codice contenuto all'interno di un frame viene eseguito per intero prima di passare a quello contenuto nel frame successivo.
- Per aggiungere una sequence all'interno di un block diagram bisogna aprire la control palette e selezionare la sequence desiderata all'interno del menù *Structures*.
- Quando una sequence viene aggiunta è costituitada un singolo frame, per aggiungere altriframe bisogna posizionare il puntatore del mouse sul bordo a forma di pellicola cinematografica e cliccare con il tasto destro delmouse.
- Dal menu a tendina che appare a questo punto,bisogna selezionare una delle due voci *Add Frame After* o *Add Frame Before*.
- Si noti come i frame della flat sequence sonosempre tutti visibili, mentre per visualizzare iframe della stacked sequence bisogna utilizzarel'indice di scorrimento posizionato sulbordo in alto al centro, come si faceva per le strutture di tipo case.

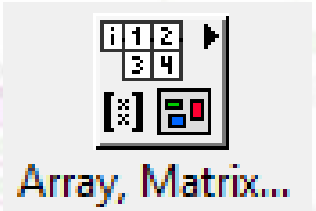


Formula Node



- LabVIEW dispone di un tipo particolare di struttura chiamato Formula Node.
- Il *Formula Node* è un oggetto grafico che funge da contenitore per l'introduzione all'interno di un block diagram di equazioni e codice testuale.
- Questo tipo di struttura evidenzia in maniera particolare la flessibilità di LabVIEW, infatti permette di introdurre all'interno di un programma sviluppato tramite linguaggio-G, dei frammenti di codice testuale.

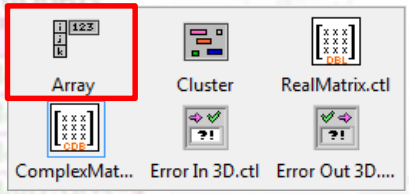
Introduzione sugli array

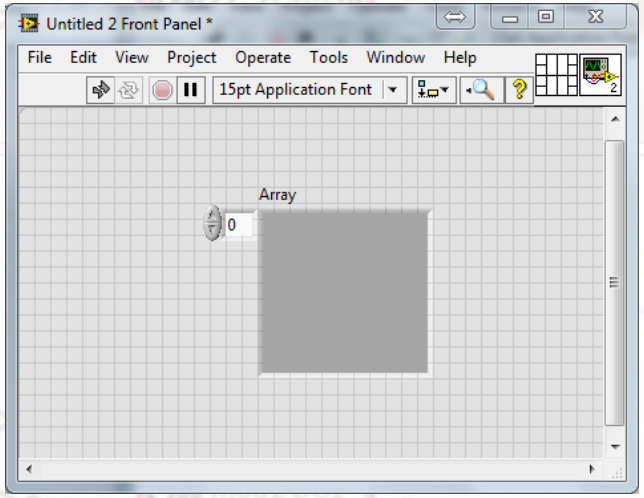


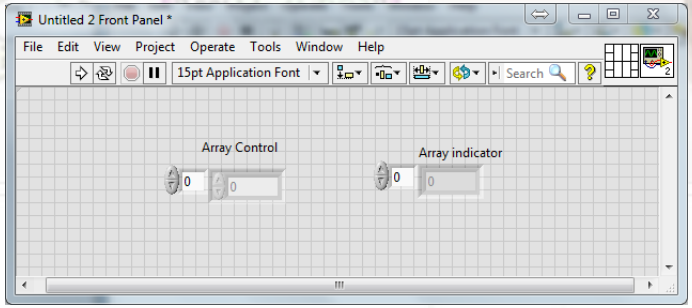
- I dati composti che il linguaggio **G** può manipolare sono gli array usati per rappresentare collezioni di dati omogenei aventi tutti lo stesso nome, ma individuati singolarmente da un indice.
- Il linguaggio **G**, come il **C** ed altri linguaggi di programmazione può manipolare dati a struttura composta come gli array ad una o più dimensione.
- Con il linguaggio **G** i dati contenuti negli array possono essere sia immessi da tastiera sia letti da strumenti di misura.
- Per l'immissione da tastiera il linguaggio **G** supporta sia **Control** di tipo array sia **Indicator** di tipo array per la visualizzazione in forma numerica di array.
- Nei linguaggi di programmazione tradizionali tipo il C, l'utilizzo dei dati tipo array richiede sia la dichiarazione del tipo di elementi che deve contenere sia quella del numero massimo degli elementi che esso può contenere.
- Nel linguaggio **G** la costruzione di un **Control** o di un **Indicator** tipo array sul **front panel** equivale alla dichiarazione dell'array.
- Il traduttore del linguaggio automaticamente riconosce il numero massimo di elementi che esso può contenere dal numero di elementi che sono state inseriti nel **Control**.
- Inoltre il **G**, a differenza del linguaggio C esegue un controllo affinché non sia superato il limite massimo predefinito.

Inizializzazione di un Array

- Per dichiarare nel **Front panel** un dato tipo array si seguono i seguenti passi:
- Dalla **Control Palette** aprire il menù **Array & Cluster** e trascinare nel **Front Panel**, il pattern denominato **Array**. Esso è composto da un quadrato centrale e da un rettangolino contenente uno 0 e da ovale con due frecce. Queste sono frecce di incremento e di decremento degli indici degli elementi dell'array.
- Per dichiarere l'array come variabile di ingresso dobbiamo definirlo come **Control**. Per questo seguiamo il percorso **Controls** → **Numeric** e trasciniamo nel quadrato grande il **Digital Control**. In questo modo abbiamo dichiarato l'**Array** come **Control**.
- Se invece vogliamo creare una variabile **Indicator** di tipo array trasciniamo nel quadrato grande un **Indicator** numerico di tipo digitale;

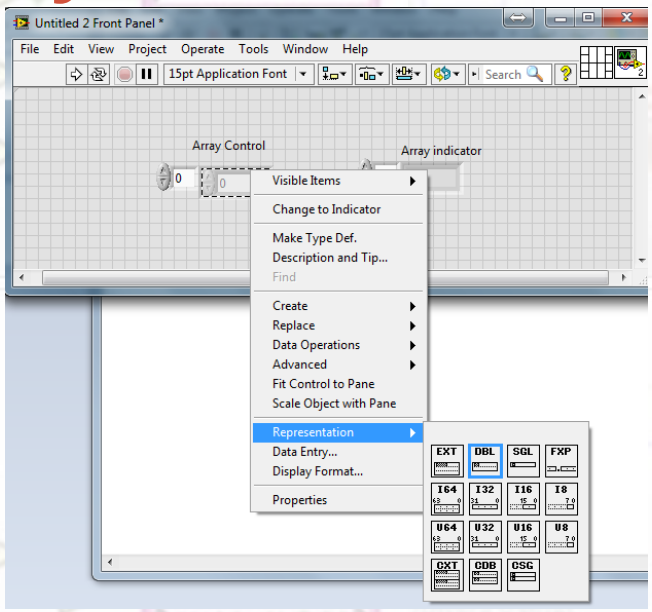






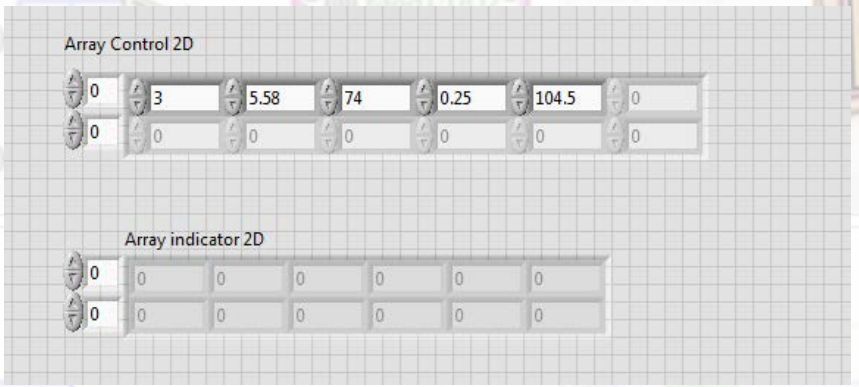
Inizializzazione di un Array

- Per dichiarare il tipo di dato degli elementi dell'array pop-up sul **Control** e sull'**indicator** creati, scegliamo la voce **Representation** e successivamente scegliamo, per il tipo di dato quello **DBL** corrispondente alla rappresentazione *floating point* in doppia precisione;
- Per definire il numero di elementi che l'array deve contenere, selezioniamo il Control creato e col puntatore di Position/size puntiamo sul lato destro e trasciniamolo verso destra e creiamo tante casella quanti sono gli elementi che vogliamo memorizzare. Eseguiamo la stessa operazione anche sull'Indicator. Nel control, da tastiera, introduciamo 5 numeri arbitrari.
- Si osservi che il **control** denominato **Array Control** ha il sesto elemento scuro e contiene uno 0. Esso indica la fine dell'array introdotto. Tutti gli elementi dell'**Indicator** sono scuri e contengono lo 0 perché in essi non è stato scritto alcun elemento perché non è stata eseguita alcuna VI che eseguisse la scrittura.



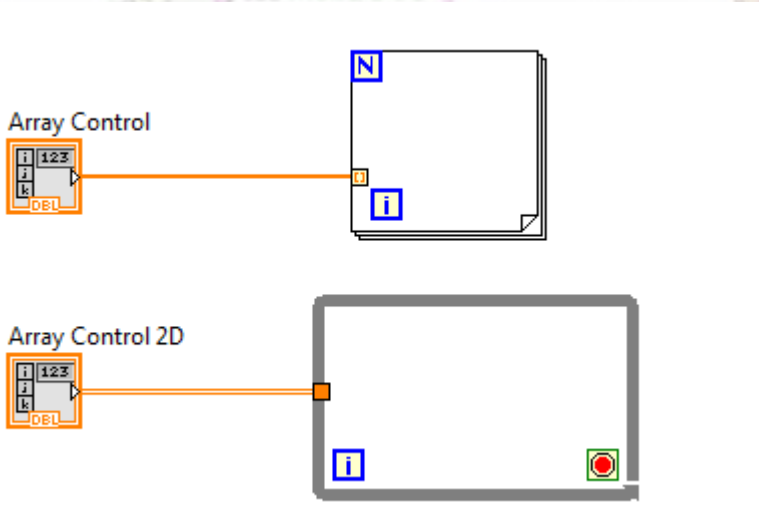
Inizializzazione di un Array

- Per creare un array bidimensionale col *tool* di Position/size puntare sul bordo inferiore del quadratino del Control e trasciniamolo verso il basso fino a quando ne compare un altro.
- Selezioniamo l'array ed eseguiamo la stessa operazione anche su array.
- Poi eseguiamo le stesse operazioni anche sull'Indicator.
- Per aumentare il numero delle righe della matrice ripetere l'operazione di risaizing prima eseguita.



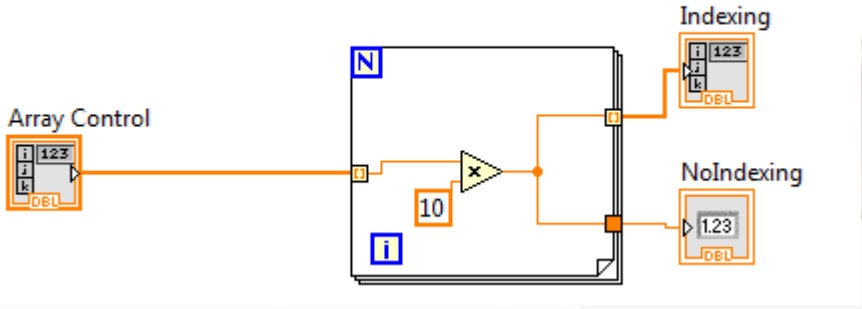
Autoindexing

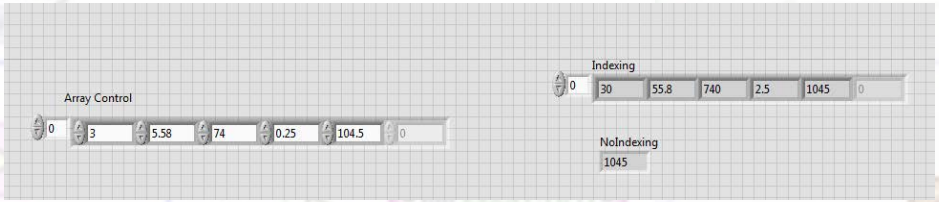
- Nel linguaggio **G** quando, nel **block diagram**, ad una struttura di tipo **for** o di tipo **while**, si collega un **Control terminal** o un **Indicator terminal** di tipo array, sul bordo si crea un tunnel individuato da un piccolo quadrato.
- Per le struttura **for** il tunnel, per *deflault* è caratterizzato dalla proprietà di **autoindexing**. Comunque la proprietà può essere disattivata.
- Nella struttura **while** la funzione di **auto-indexing** non è attivata per default. Per attivarla quando non attiva o per disattivarla quando attiva si deve eseguire un pop-up sul tunnel e selezionare la voce **Enable Indexing** o **Disabile Indexing**.
- Un tunnel che gode della proprietà di **autoindexing** è riconoscibile dal fatto che il quadrato non è pieno ma contiene nell'interno un rettangolino vuoto. Inoltre il filo di collegamento è spesso.



Autoindexing

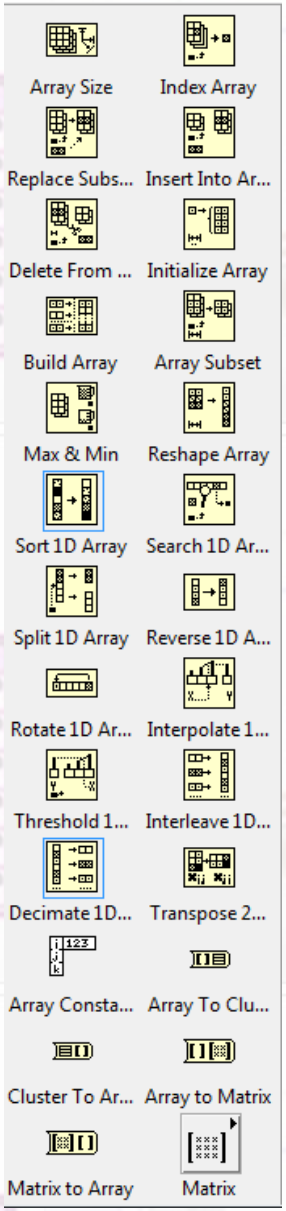
- Quando si introduce un array in una struttura for, se l'auto-indexing del tunnel è abilitato, il terminale di conteggio è automaticamente inizializzato al numero di elementi dell'array.
- Se il tunnel creato quando si collega l'icona corrispondente al Control terminal alla struttura for è auto-indexing, gli elementi entrano nella struttura uno per volta all'inizio di ogni ciclo.
- Il loop indicizza gli elementi scalari da un array ad una dimensione, gli array ad una dimensione da array a due dimensioni, ecc.
- L'azione opposta avviene nel tunnel che si crea quando colleghiamo l'uscita di un nodo interno al Block diagram al bordo esterno di una struttura di iterazione che genere un dato ad ogni ciclo. Se il tunnel è auto-indexing, in esso, gli elementi sono accumulati uno per volta alla fine di ogni ciclo. L'array è trasferito all'uscita in blocco quando il processo di iterazione ha termine.
- Se il tunnel di uscita non è autoindicizzato, quando il processo di iterazione è ultimato, è messo in uscita solo l'ultimo dato generato.
- La caratteristica di autoindexing può essere utilizzata in alternativa alla funzione Index Array per indicizzare singolarmente gli elementi di un vettore.



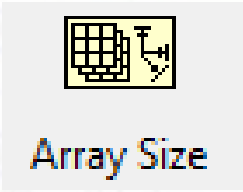


Operazioni su Array

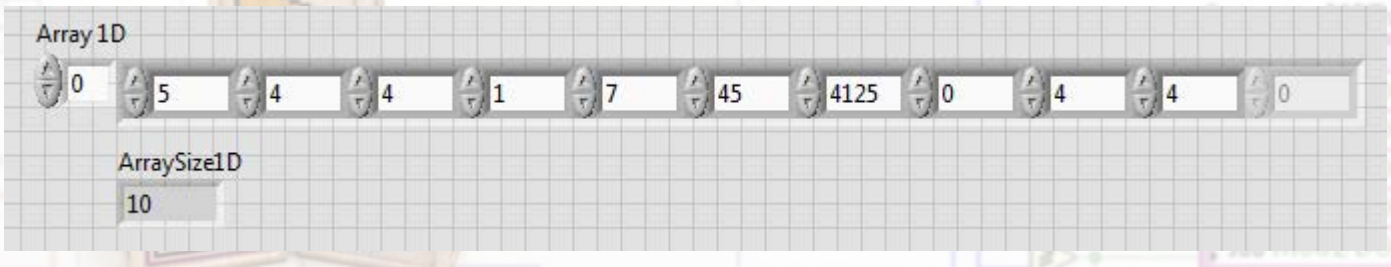
- Il linguaggio **G** supporta molteplici funzioni per la manipolazione degli array (vedi figura)
- Per accedere alle funzioni per il trattamento degli array che il LabVIEW supporta seguire il percorso Functions → Array. Per utilizzarle è opportuno consultare le note esplicative contenute per ciascuna di esse nell'Help.
- Vediamo ora qualche esempio esplicativo (Esempio 3)

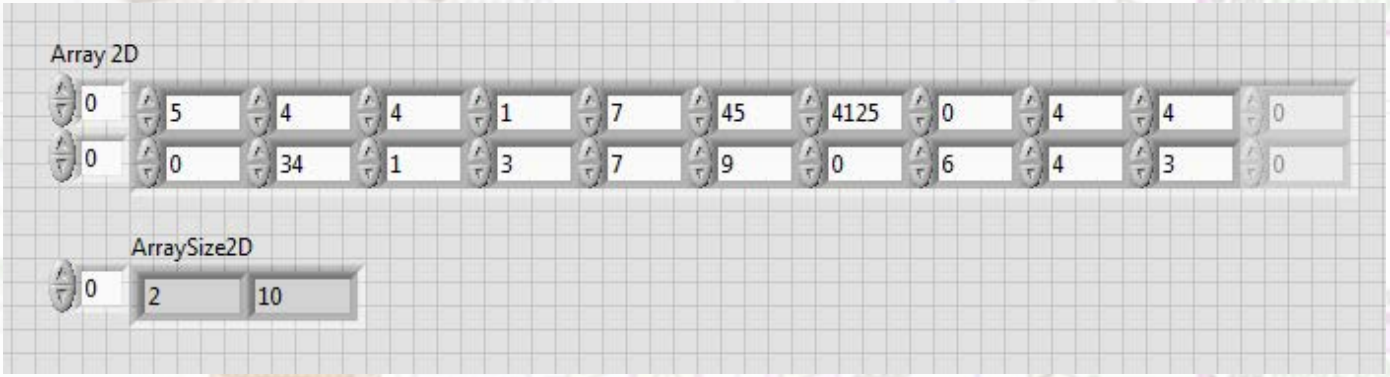


Array size

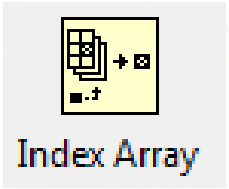


- Tale funzione, in restituisce: a) il numero degli elementi contenuto nell'array collegato al suo ingresso se l'array è unidimensionale; b) un vettore unidimensionale il cui primo elemento indica il numero di righe e il secondo indica il numero di colonne contenute nell'array collegato al suo ingresso.

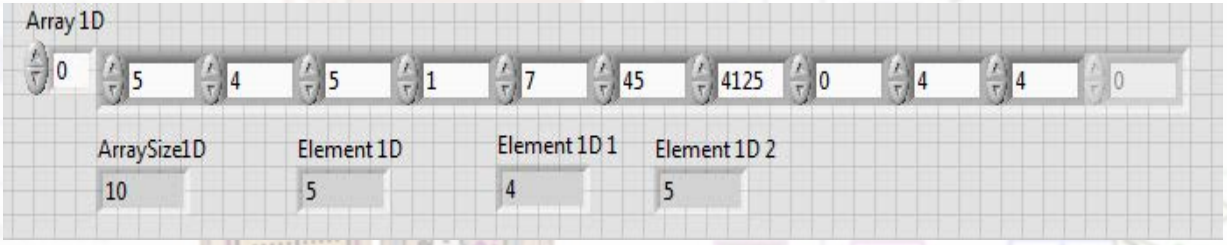


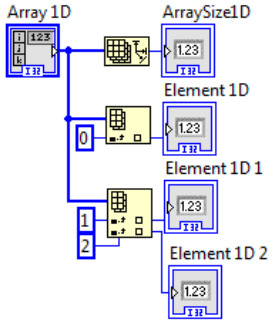


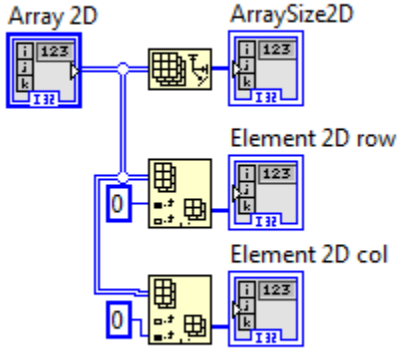
Index Array



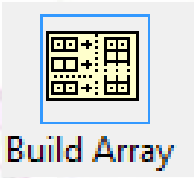
- La funzione Index Array restituisce l'elemento o il sub-array all'indice specificato.
- Quando si collega un array a questa funzione essa si ridimensiona automaticamente per visualizzare gli indici di ingresso per ciascuna dimensione nell'array (**polimorfismo**).
- La funzione ha una uscita per la visualizzazione dell'elemento o del sub-array indicizzato.



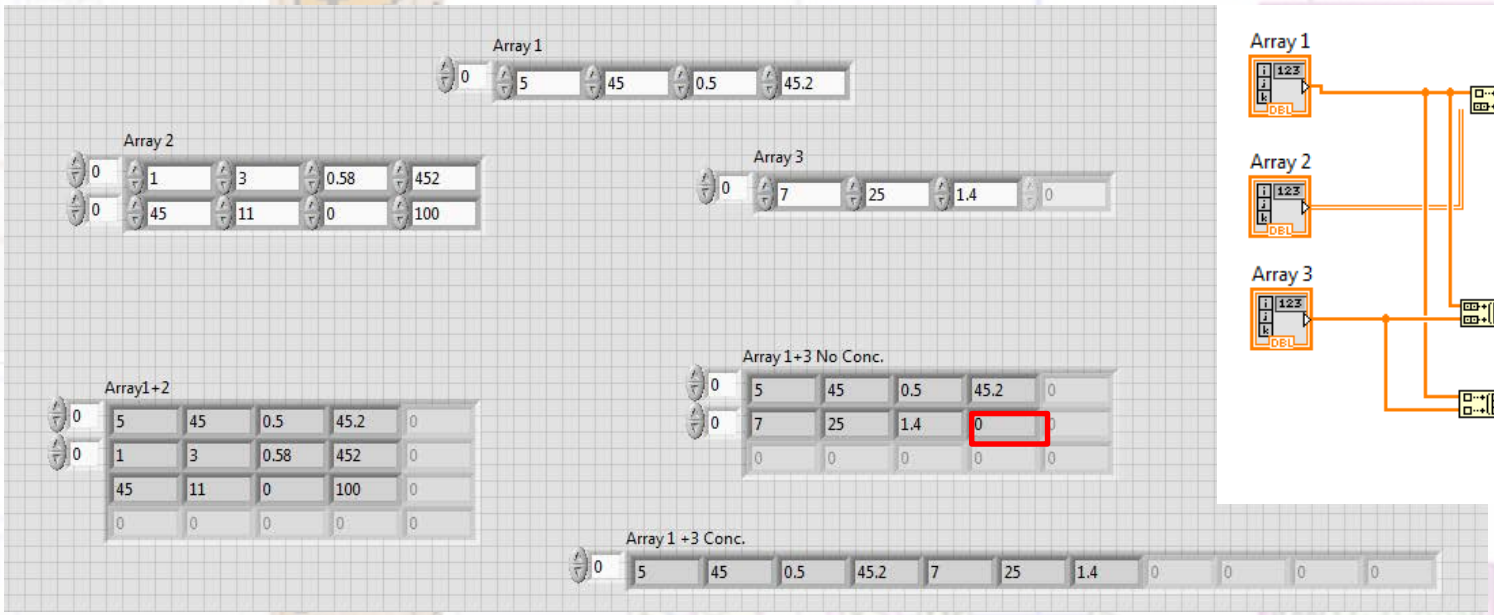




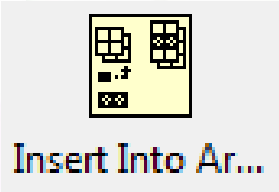
Build Array



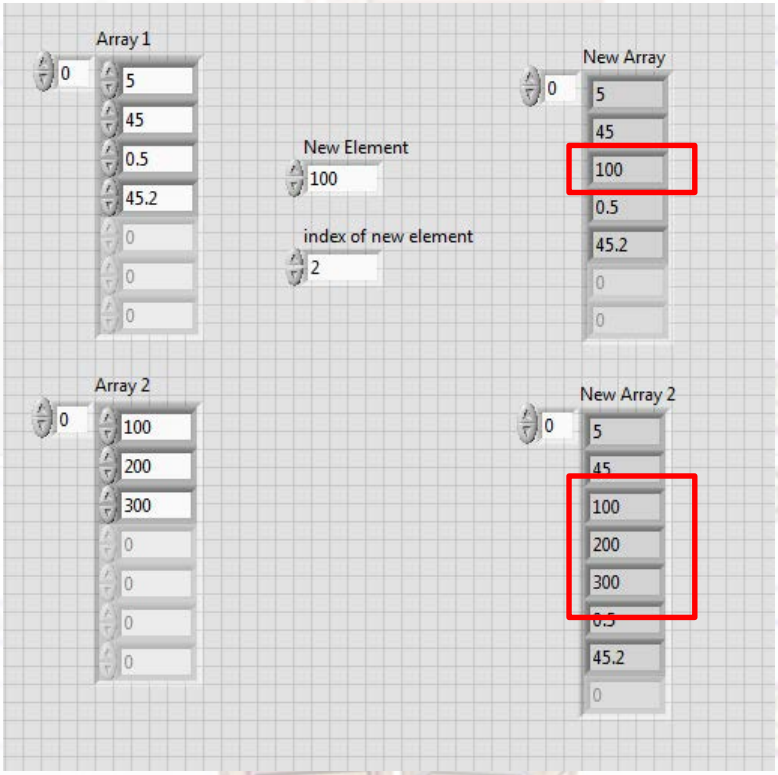
- La funzione Build array consente di concatenare o di appendere elementi ad un array di qualunque dimensione.
- La funzione **Build array** opera in due modi distinti in funzione al fatto che, con un'operazione di pop-up si sceglie o meno l'opzione **Concatenate Inputs**.
- L'opzione di concatenamento è selezionabile solo se gli array sono della stessa dimensione anche se non contengono lo stesso numero di elementi.
- Se si seleziona l'opzione **Concatenate** la funzione appende in coda all'array gli ingressi.
- Se l'opzione non è selezionata l'array costruito ha una dimensione in più rispetto a quello di partenza. Se i due array hanno dimensione differente la dimensione finale sarà pari a quella dell'array a dimensione maggiore (verranno introdotti dei valori a 0 per compensare)

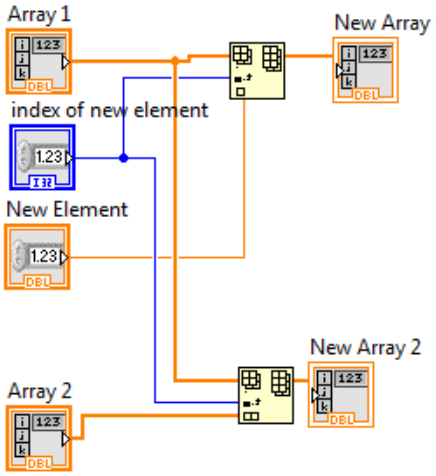


Insert into array



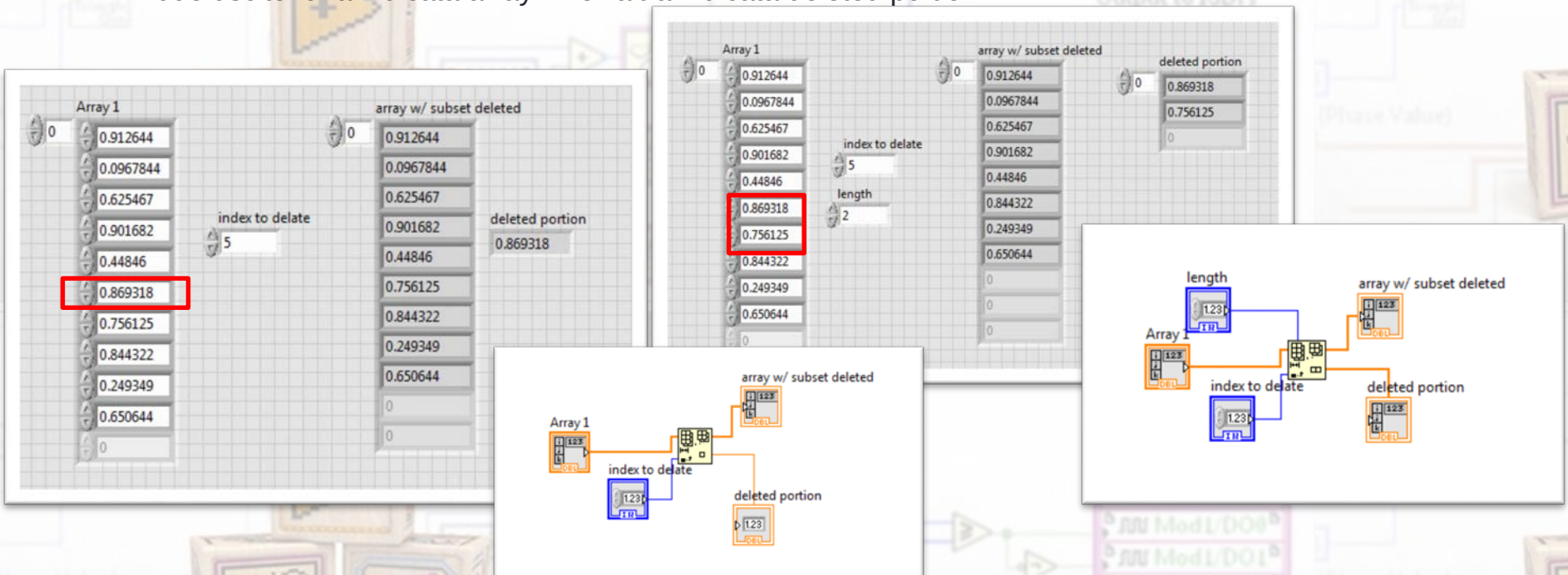
- La funzione Insert into array permette di inserire elementi scalari o array in un array preesistente. La funzione dispone di un ingresso N-dim array, ingressi Index e di un ingresso n or n-1 dim array per inserire nuovi elementi nell'array di ingresso.
- Se l'indice indicato è minore di quello contenuto nell'array originario gli elementi che lo seguono sono spostati dopo l'ultimo elemento dell'array inserito. Se invece l'indice indicato è maggiore di quello massimo dell'array originario, il nuovo array non è inserito.





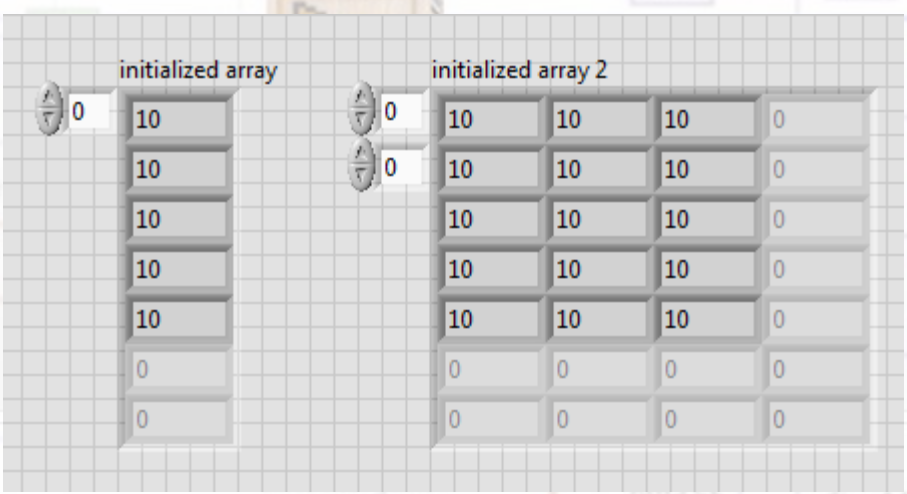
Delete From array

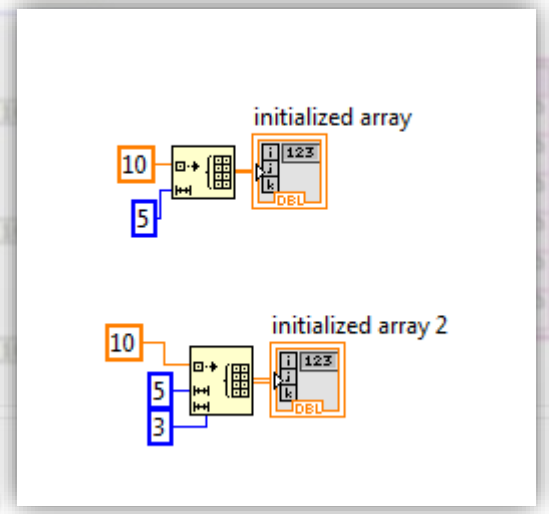
- La funzione *Delete from array* consente di cancellare un elemento da un array.
- La funzione ha:
  - un ingresso n-dimension array al quale deve essere collegato l'array dal quale si vuole cancellare un elemento;
  - un ingresso indicato length che deve essere collegato ad una costante di tipo intero.
    - Per cancellare un elemento da un array unidimensionale, una riga o una colonna da un array bidimensionle, tale ingresso non deve essere collegato ad alcuna costante. In questo caso nell'uscita deleted portion è restituito l'elemento specificato dall'indice;
  - un ingresso indicato index al quale deve essere collegata una costante di tipo intero che individua l'elemento da cancellare. Un solo indice per volta può essere collegato;
  - due uscite: una indicata array w e l'altra indicata deleted portion.



Initialize Array

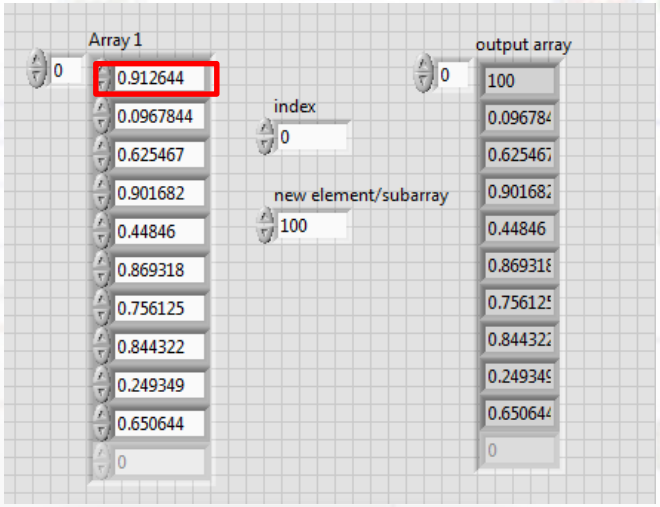
- Crea un array inizializzato ad un valore predefinito.
- La funzione permette di inizializzare anche array con dimensione maggiore di espandendo l'icona della funzione ed utilizzando più connessioni di ingresso

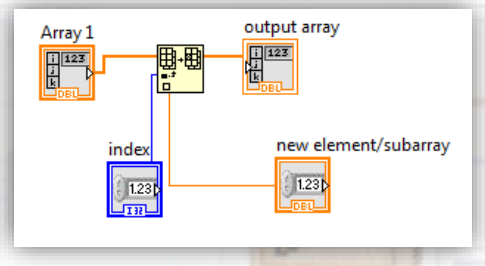


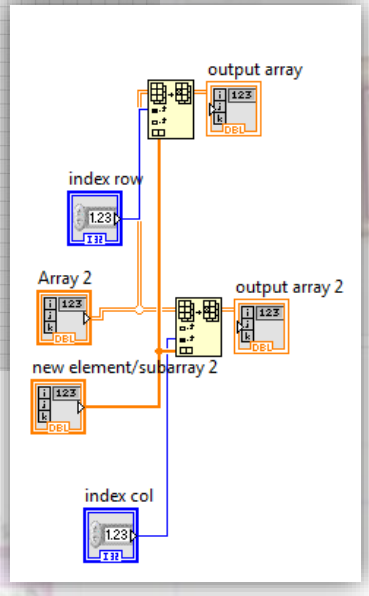


Replace Array Subset

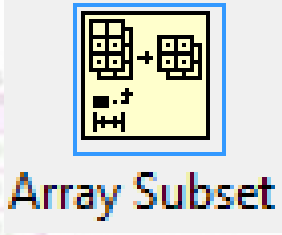
- Sostituisce un elemento di un array nella posizione indicato dal parametro di ingresso index col valore fornito in ingresso
- Può essere utilizzata per sostituire più elementi contemporaneamente
- E' polimorfica e pertanto può essere usata per sostituire anche dei sub-array



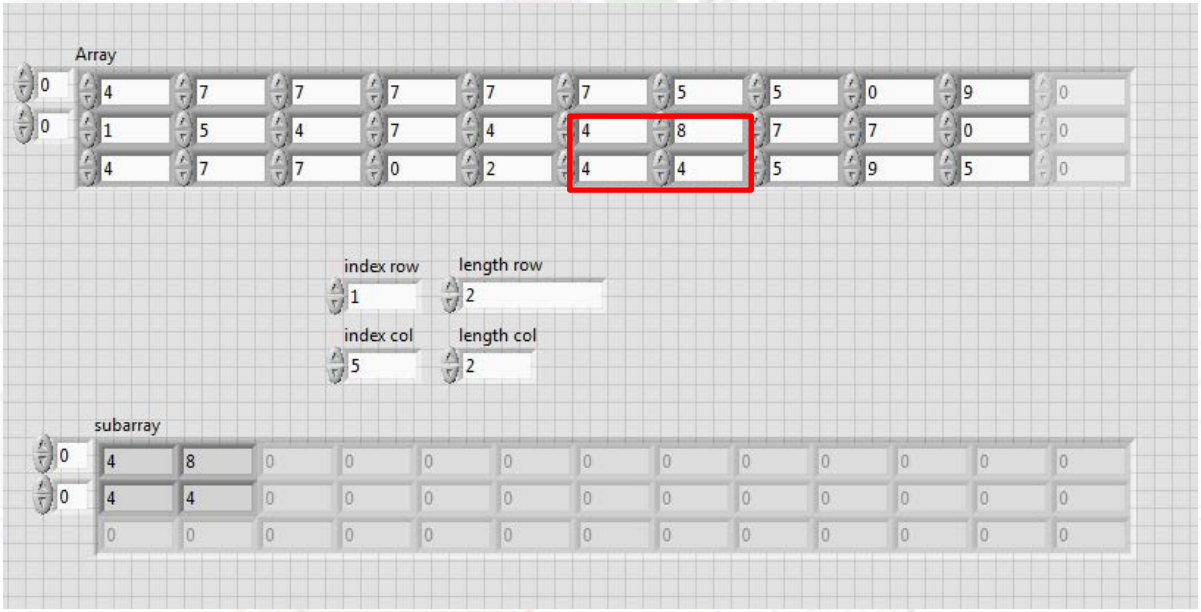


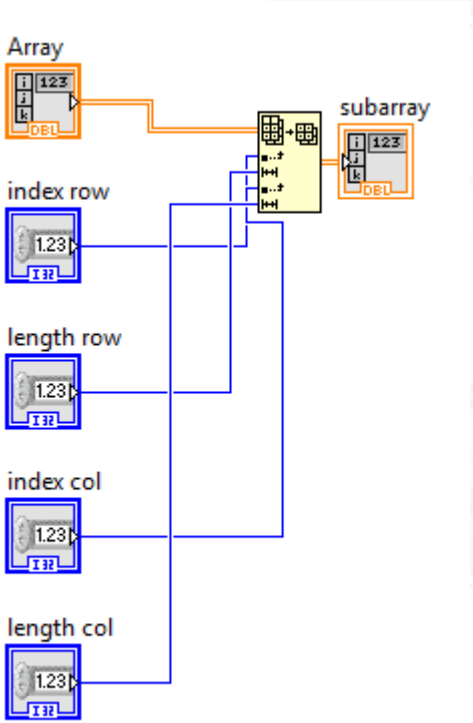


Array subset

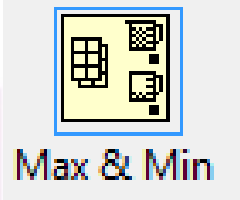


- Estrae da un array una porzione selezionata mediante gli indici in ingresso e le lunghezze dei campi .

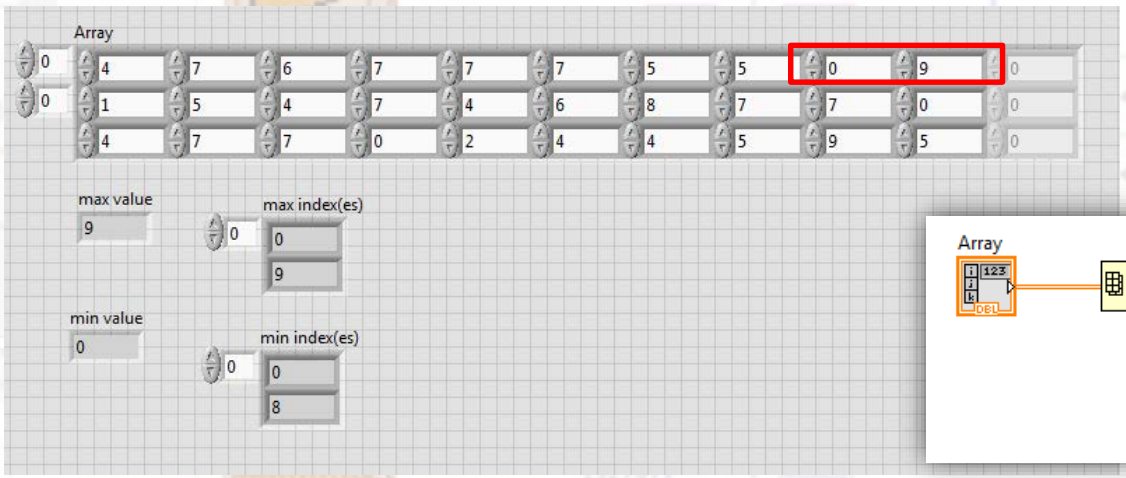


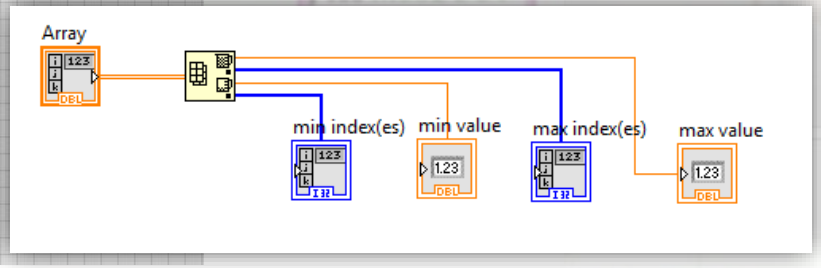


Max & Min

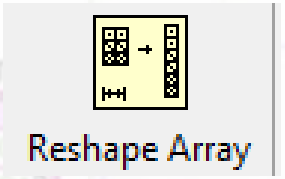


- Restituisce il valore e la posizione del numero massimo e minimo in un array.
- Se vi sono più valori corrispondenti al massimo/minimo la posizione restituita si riferisce a quello caratterizzato dagli indici più piccoli
- La funzione è polimorfica, ovvero restituisce come posizione uno scalare se l'array in ingresso è 1D od un array se l'array in ingresso ha dimensione >1

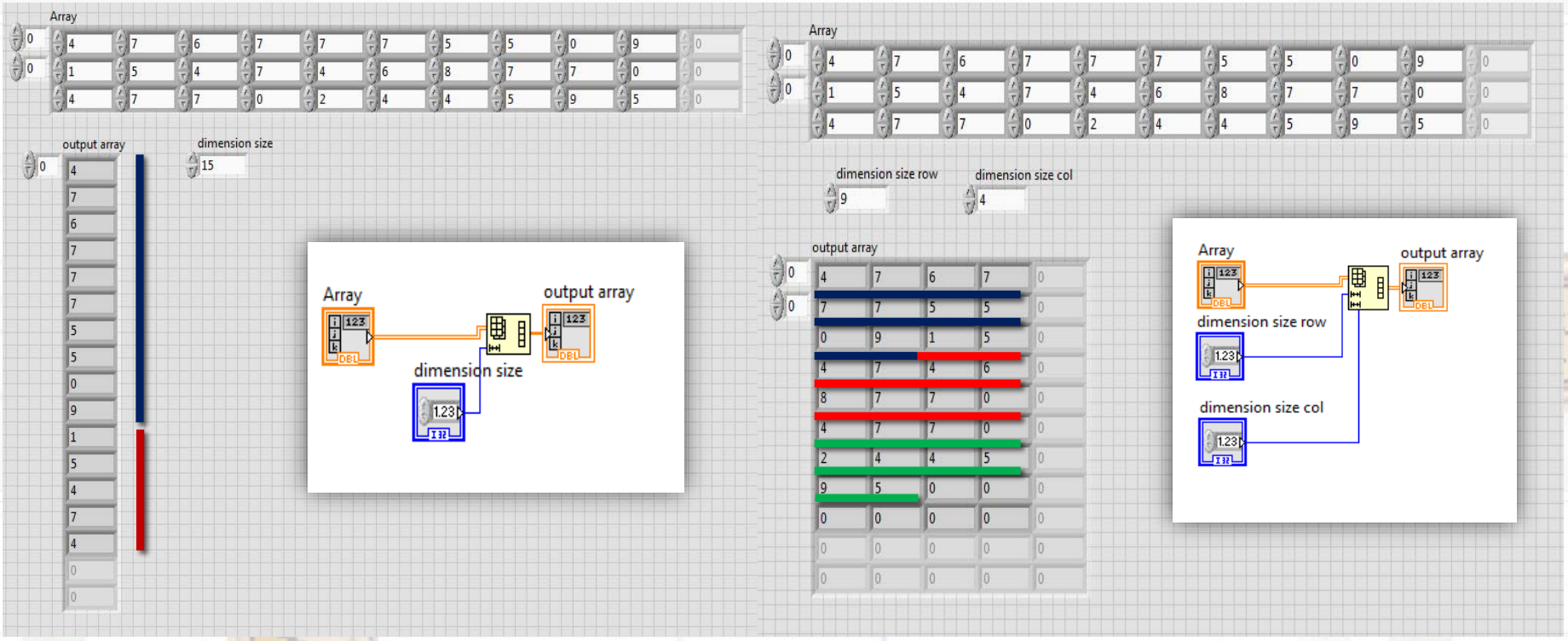




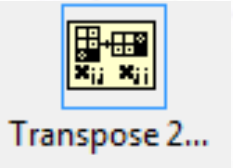
Reshape array



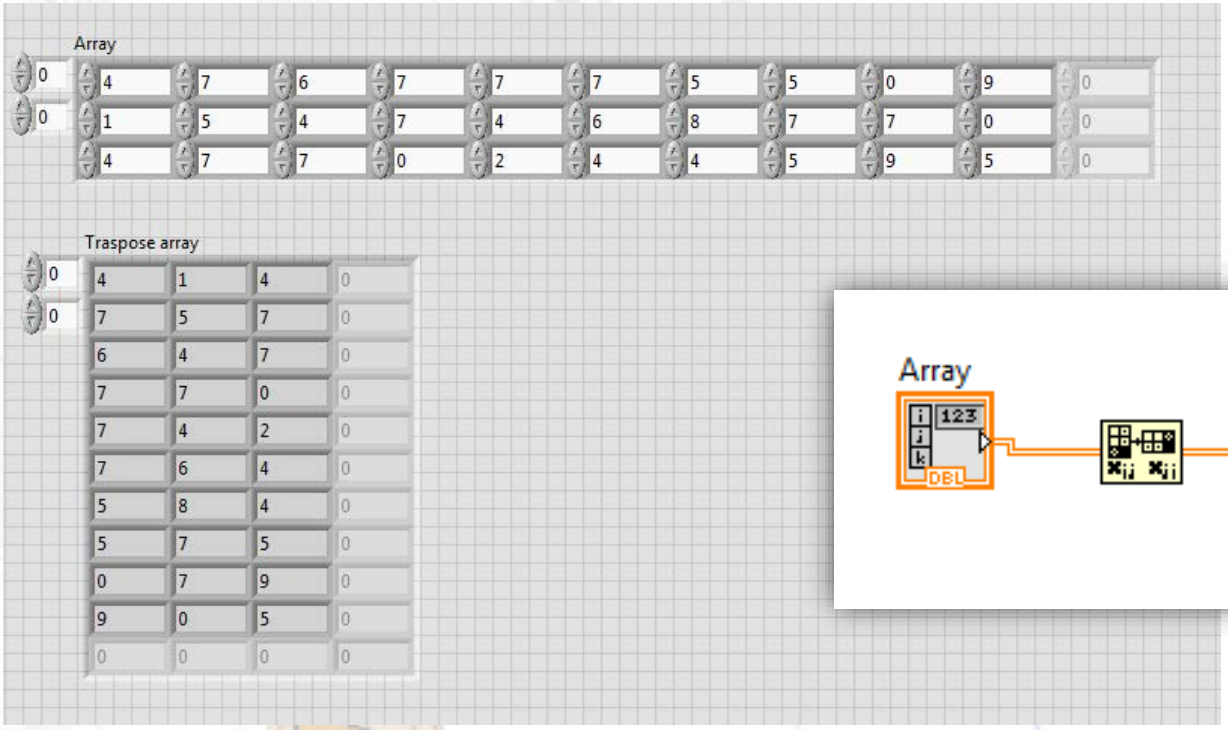
- Cambia la dimensione di un array in funzione della parametro *dimension size* in ingresso

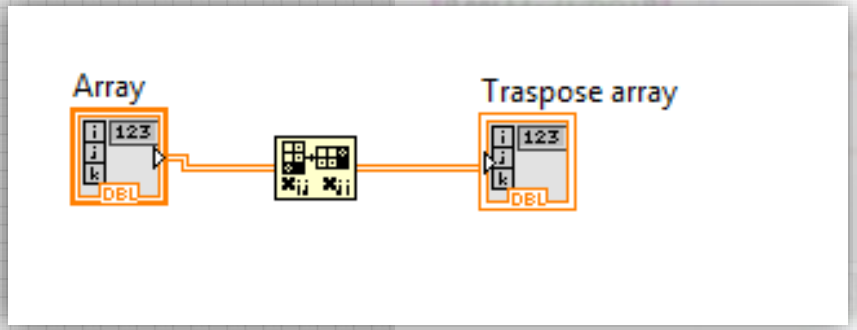


Traspose 2D Array



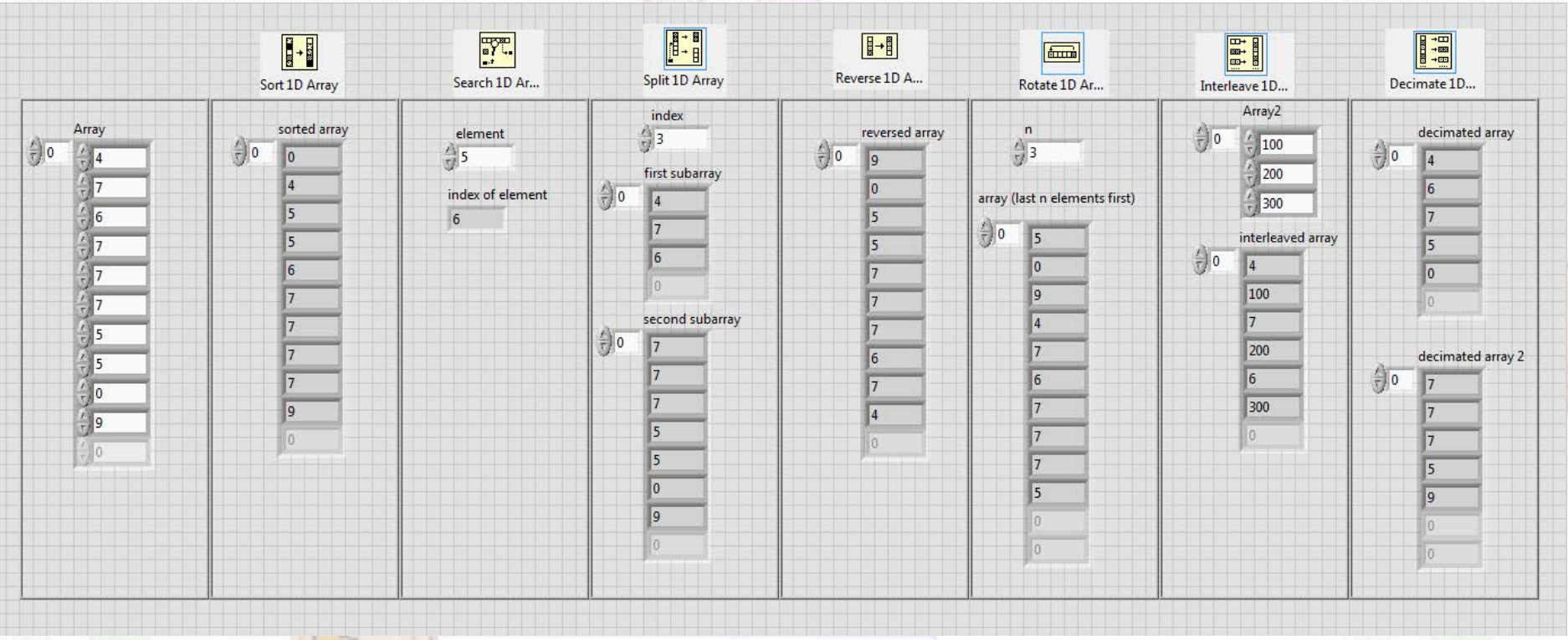
- Esegue la trasposta di un array bidimensionale (inverte righe e colonne)





Operazioni su vettori 1D

- Esempi di funzioni che operano solo su vettori 1D
- Esempio 5

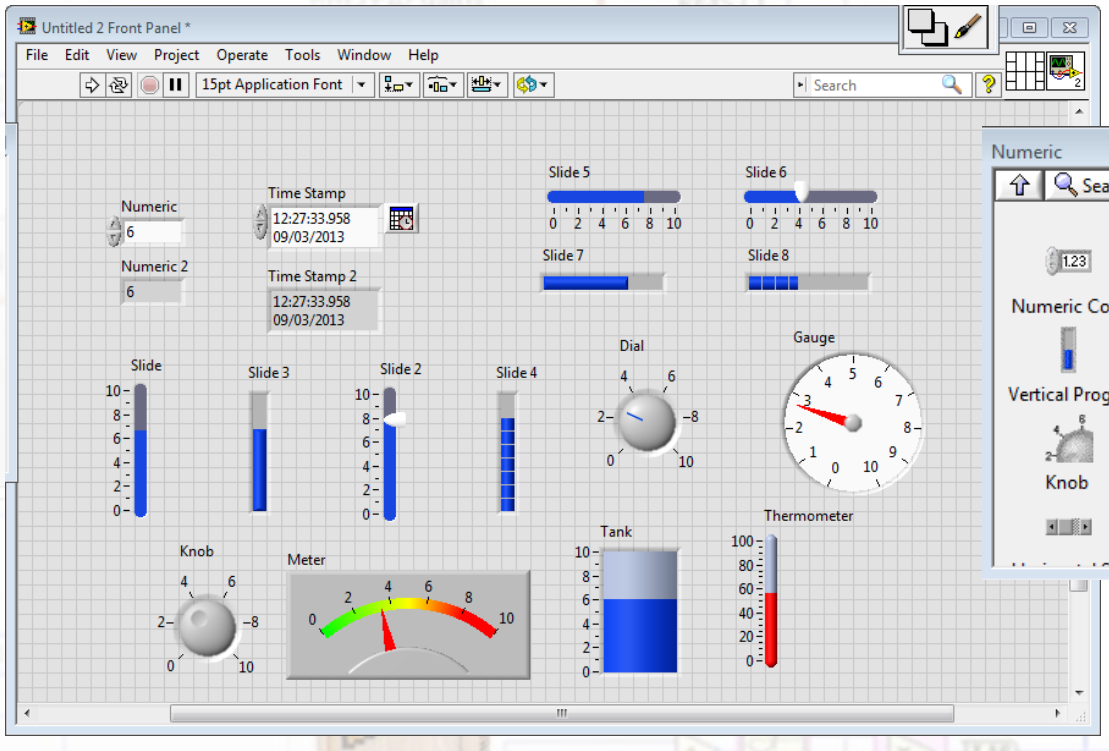


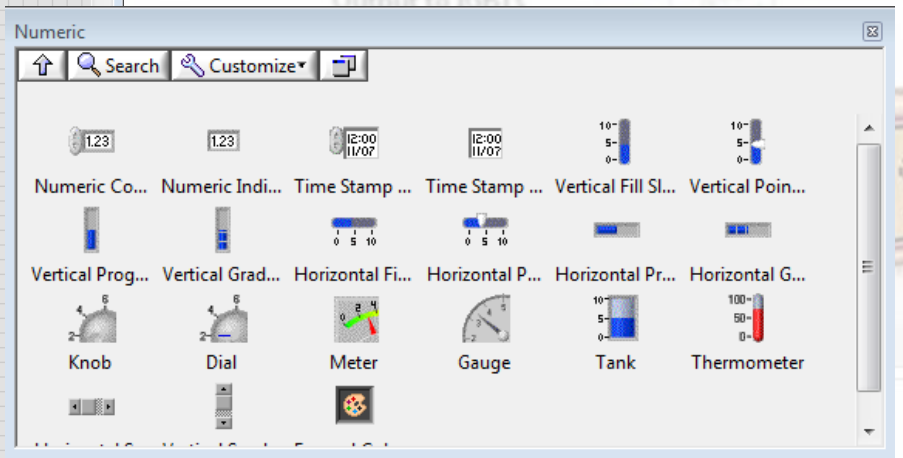
Oggetti grafici

- Nelle precedenti lezioni sono stati già introdotti alcuni oggetti grafici (che abbiamo già utilizzato nei primi esempi di programmazione) come LED, pulsanti e grafi ci. LabVIEW dispone di una notevole varietà di questi oggetti, che permettono di personalizzare a piacimento i pannelli frontali.
- Vediamo quali sono i più importanti, quali le caratteristiche principali e come utilizzarli nelle nostre applicazioni.
- Gli oggetti grafici di LabVIEW si dividono fondamentalmente in due categorie: controlli ed indicatori.
- Un *controllo* è un oggetto che prende un input dall'operatore che agisce sul pannello e lo trasferisce all'interno del VI
- Un *indicatore* prende in ingresso un valore dal VI (una variabile o una costante) e lo rende visibile sul front panel.
- A loro volta, controlli ed indicatori esistono in varie tipologie e sono raggruppati in base al tipo di variabile che possono rappresentare.

Oggetti grafici - Numeric

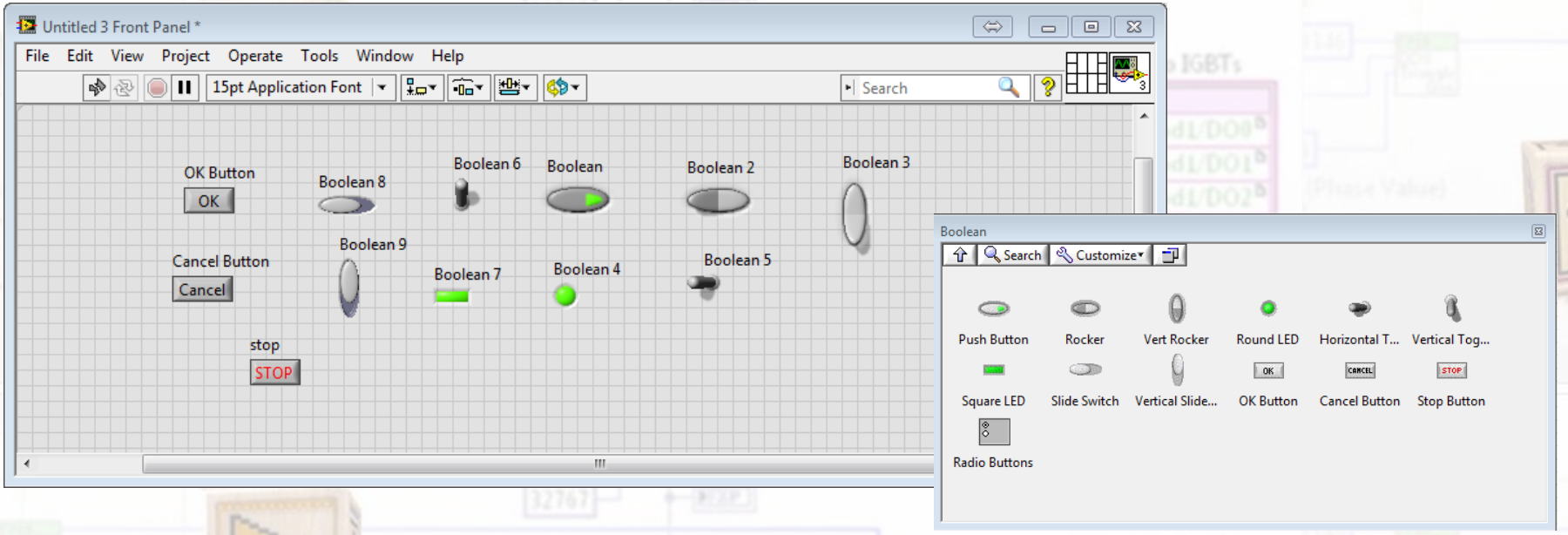
- Questo gruppo contiene controlli ed indicatori numerici; ne fanno parte i controlli e gli indicatori a display, gli slider (barre), knob e dial (manopole), meter e gauge (indicatori a lancetta) ed altri ancora.
- Sono sempre associati a variabili numeriche, di qualsiasi tipo, quindi numeri sia puri che in virgola mobile.





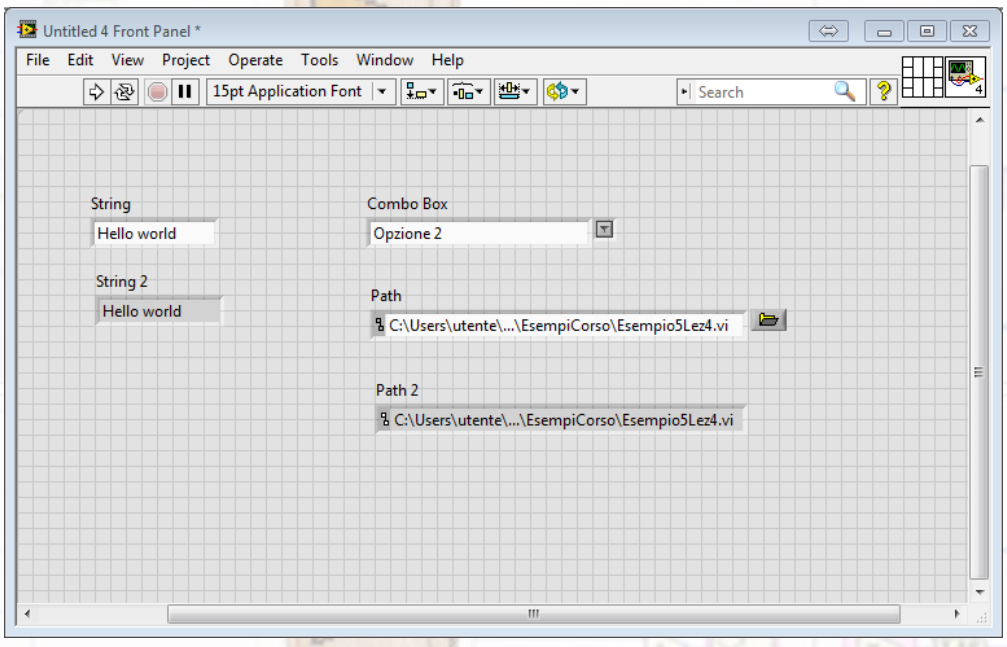
Oggetti grafici - Boolean

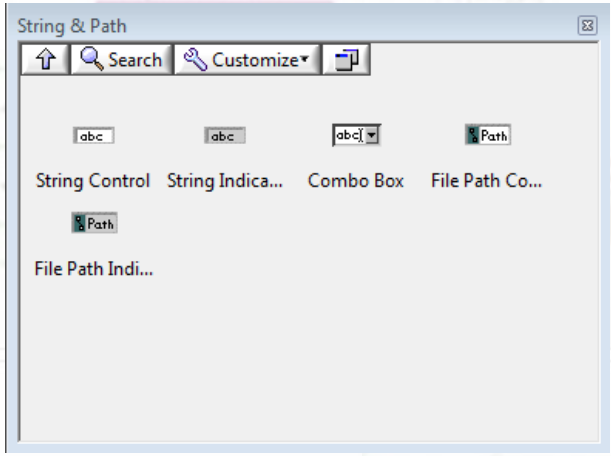
- Fanno parte di questo gruppo i controlli e gli indicatori booleani (quindi con solo due valori: vero/falso).
- Essenzialmente si tratta di vari tipi di LED e di pulsanti, alcuni esempi dei quali sono i LED tondi e quadrati, i pushbutton, gli switch, ecc.



Oggetti grafici – String e Path

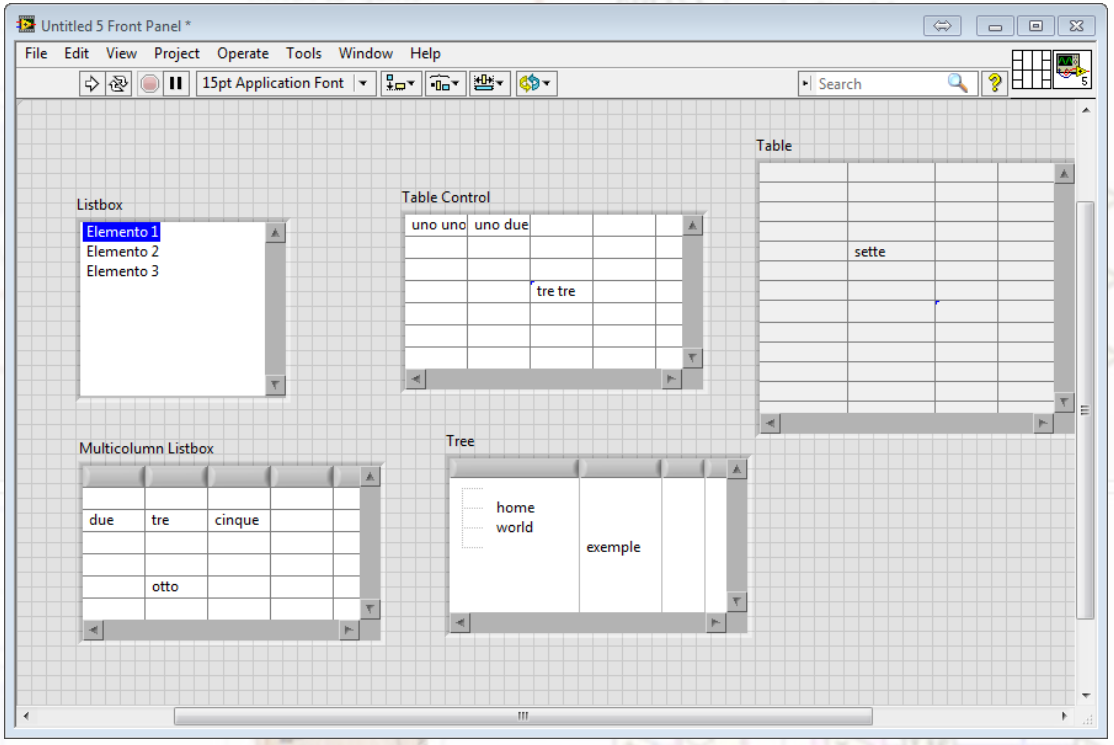
- Questo gruppo contiene controlli ed indicatori di tipo stringa, e controlli ed indicatori di tipo percorso file (path).
- I controlli di tipo stringa sono molto utili per l'introduzione dei dati di tipo testo nelle nostre applicazioni, mentre i relativi indicatori sono utili per la visualizzazione dei messaggi.
- I controlli e gli indicatori di tipo path sono utili per la gestione della lettura e scrittura dei file, anche se in molte circostanze possono essere rimpiazzati dalle finestre di dialogo per la scelta dei path di windows

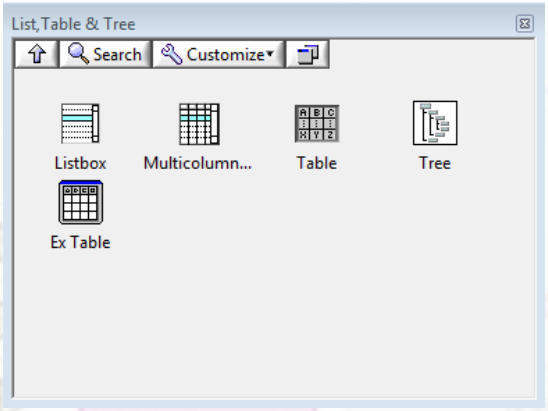




Oggetti grafici – List & Table

- In questa categoria troviamo controlli ed indicatori di tipo tabellare, di varie tipologie: matrici, tabelle, alberi, listbox ed altro ancora.
- Sono usatissimi per presentare i dati in formato tabellare e anche per la visualizzazione di strutture tipo file system (alberi).





Visualizzatori grafici

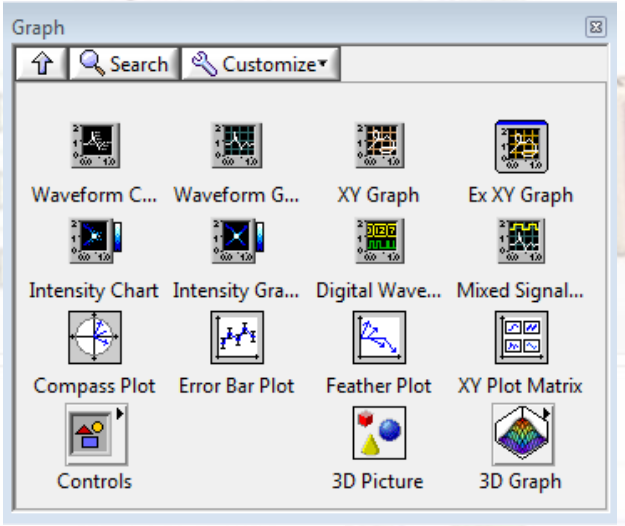
- LabVIEW dispone di quattro tipi di indicatori per la visualizzazione grafica.
  - Chart;
  - Graph.
  - 2D Graph
  - 3D Graph
- Vi sono 2 tipi di Chart:
  - Waveform Chart;
  - Intensity Chart.
- Vi sono 6 tipi di Graph:
  - Waveform Graph;
  - XY Graph;
  - Ex XY Graph
  - Intensity Graph ;
  - Digital Waveform Graph.
  - Mixed Signal Graph

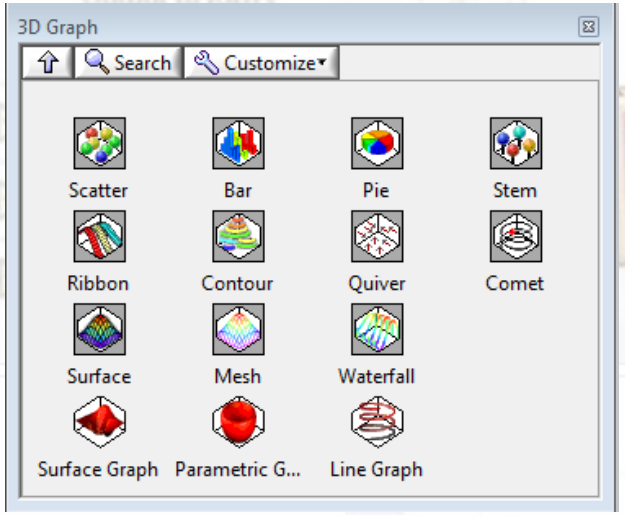
Vi sono 4 tipi di 2D Graph

Compass
Error Bar
Feather Plot
XY Plot Matrix

Vi sono 14 tipi di 3D Graph

Scatter
Stem
Comet
Surface
Contour
Mesh
Waterfall
Quiver
Ribbon
Bar
Pie
3D Surface Graph
3D Parametric Graph
3D Line Graph

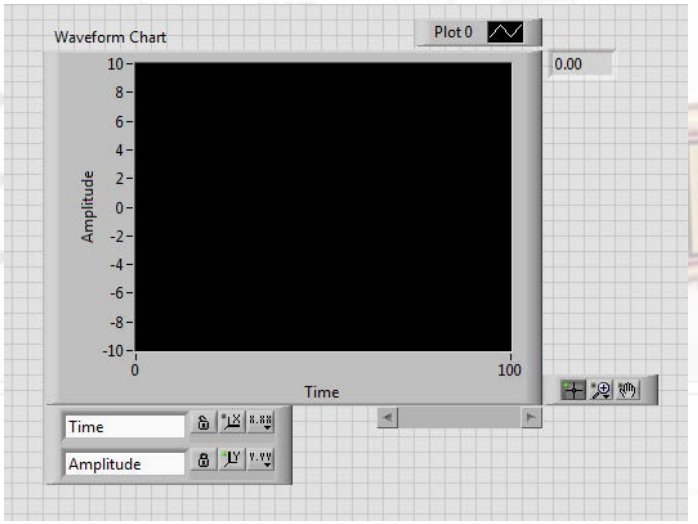




Visualizzatori grafici

- **Waveform Graphs and Charts**
  - Sono utilizzati per mostrare dati acquisiti a rate costante
- **XY Graphs**
  - Sono utilizzati per mostrare dati acquisiti a rate non costante o dati proveniente da funzioni a più valori
- **Intensity Graphs and Charts**
  - Mostrano plot 2D e 3D usando I colori per mostrare I valori della seconda o terza dimensione
- **Digital Waveform Graphs**
  - Mostrano I dati come impulsi o gruppi di linee digitali
- **Mixed Signal Graphs—**
  - Mostrano I dati che di solito sono mostrati da graphs, XY graphs, e digital waveform graphs in un unico display. Mostrano anche cluster che contengono qualunque combinazione di queste tipologie
- **2D Graphs**
  - Mostrano dati 2D in un plot 2D
- **3D Graphs**
- Mostrano dati 3D in un plot 3D

Waveform chart



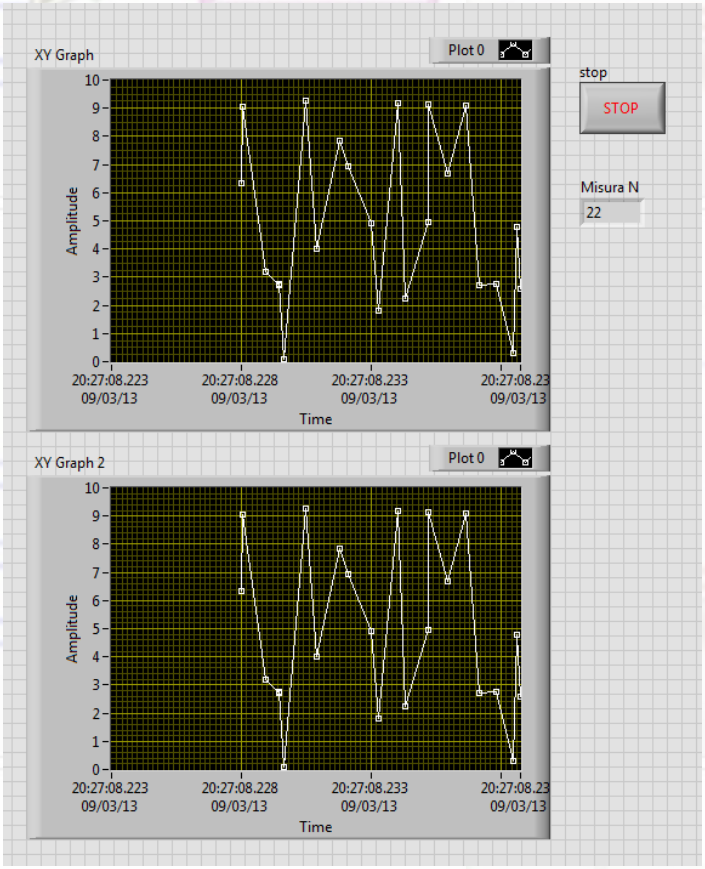
- Questo tipo di grafico, posto nella subpalette **Graph** della palette **Controls,** consente di disegnare una o più curve sullo stesso diagramma.
- Esso è utilizzato molto spesso nei loop in quanto disegna i vari punto man mano che essi vengono calcolati od acquisiti, aggiungendo nuovi dati non appena 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.
- Mediante un pop-up menu è possibile personalizzare le Chart per adattarle alle proprie esigenze relative alla visualizzazione dei dati o per visualizzare più forme d'onda.
- Gli elementi di configurazione che vogliamo visualizzare sono contenuti nella voce: Visibile Items;
- In figura è mostrata una Waveform chart con tutte le opzioni evidenziate: la scrollbar, la legend, la palette, il display digitale, la rappresentazione delle scale ecc.
- Altre opzioni utili contenute nel pop-up menu sono il Data Operation che, fra le altre voci contiene il comando Clear chart per pulire il display, le X scale e la Y scale che contengono sotto menù per modificare i parametri relativi all'asse delle X e quello delle Y, e l'opzione Chart History Lenght che permette di definire la lunghezza del buffer per la memorizzazione dei dati. Il valore stabilito come default è 1024.

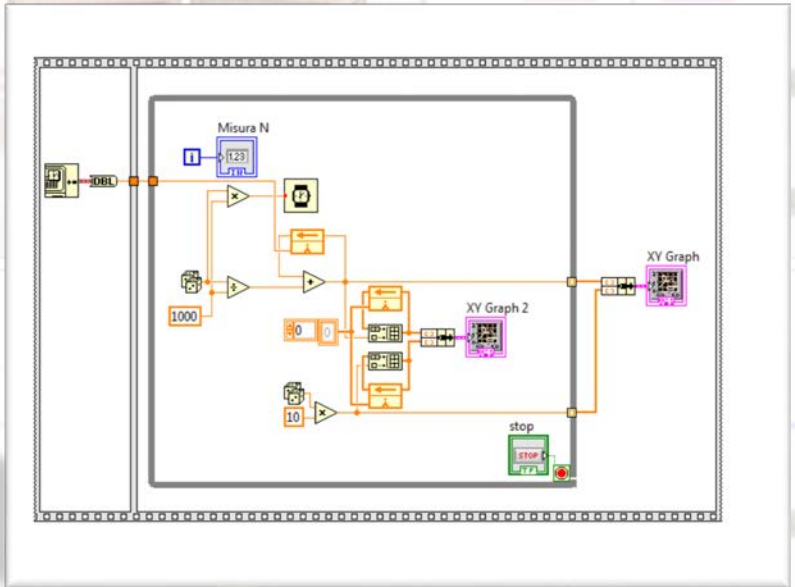
Waveform Graph

- 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.
- 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
- 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).
- Nel caso di più diagrammi sullo stesso grafico al posto della funzione bundle si deve utilizzare la funzione build array tramite il cammino Function, Array, Buid Array.
- 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
- L'esempio 2 chiarisce meglio il concetto

XY Graph e Ex XY Graph

- XY Graph è un modo generale di rappresentare coppie di punti in un piano.
- Può essere utilizzato per riportare in grafico due misure per studiarne la correlazione oppure per riportare in grafico misure non equispaziate in tempo
- Esempio 3: Misure acquisite ad intervalli di tempo non costanti e mostrate con un grafico XY usato in due modi differenti:
  - Grafico mostrato solo alla fine della misura
  - Grafico mostrato durante la misura





Intensity Chart e Graph

- Intensiity graph e chart sono utilizzati per mostrarti dati 2D e 3D posizionando blocci colorati in un piano Cartesiano
- Ogni numero nell'array rappresenta un colore specifico.
- Gli indici degli elementi di un array 2D indicano la posizione del blocco colorato.
- Esempio 5

Input Array

Column = y

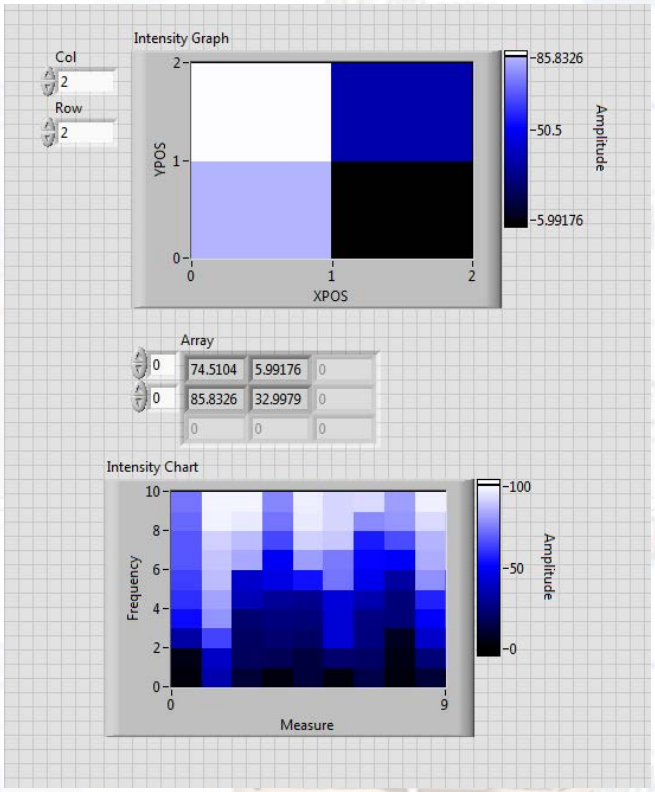
Row = x	0	1	2
0	50	50	13
1	45	61	10
2	6	13	5

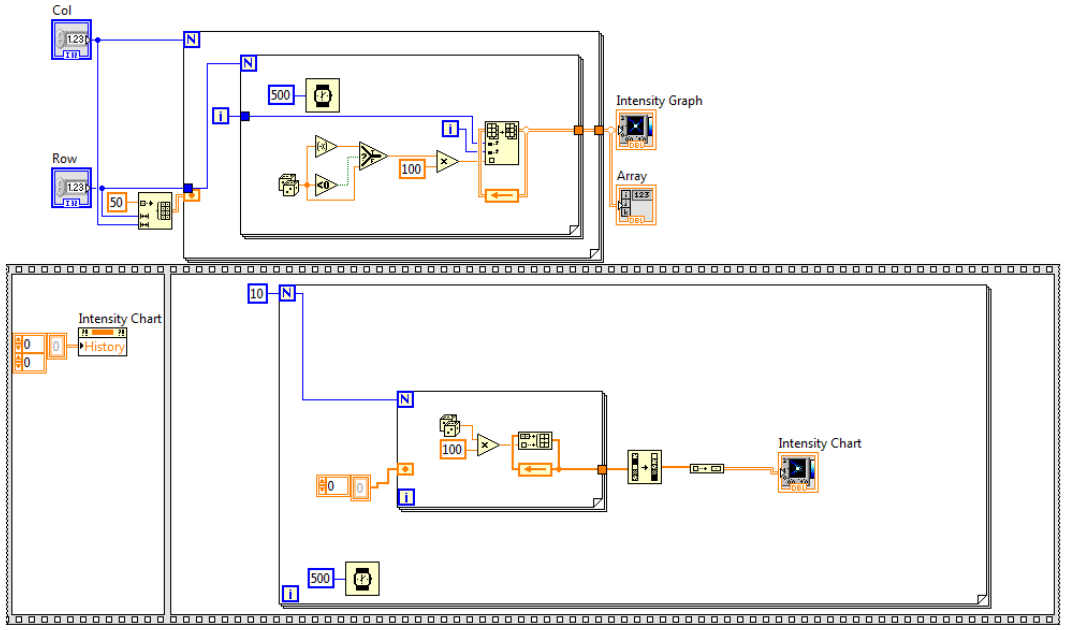
Color Map Definition

Array Element = z	Color
5	blue
6	purple
10	lt red
13	dk red
45	orange
50	yellow
61	green

Resulting Plot

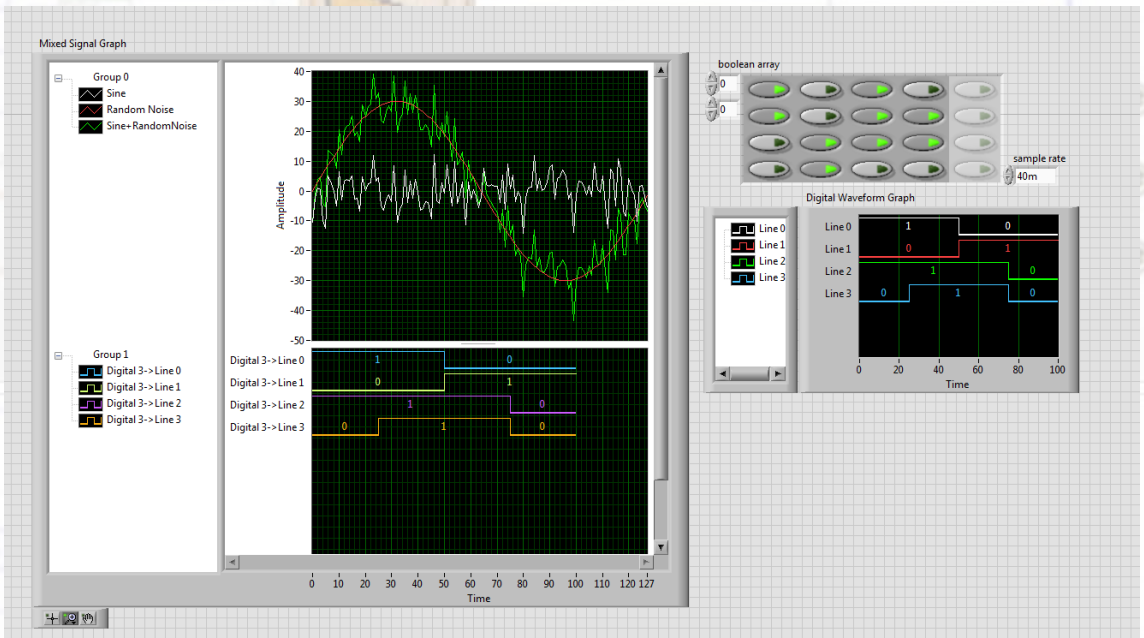
2	dk red	lt red	blue
1	yellow	green	dk red
0	yellow	orange	purple
	0	1	2

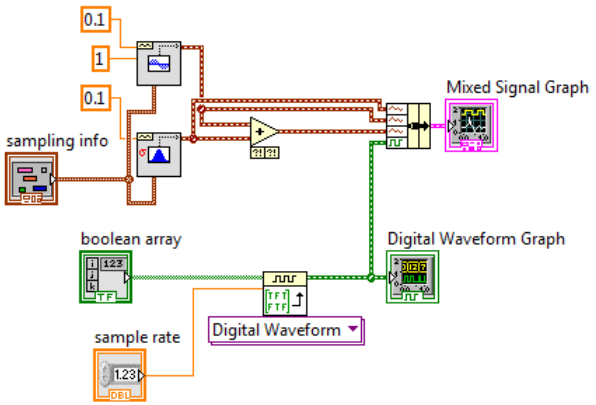




Digital Waveform e Mixed Signal Graph

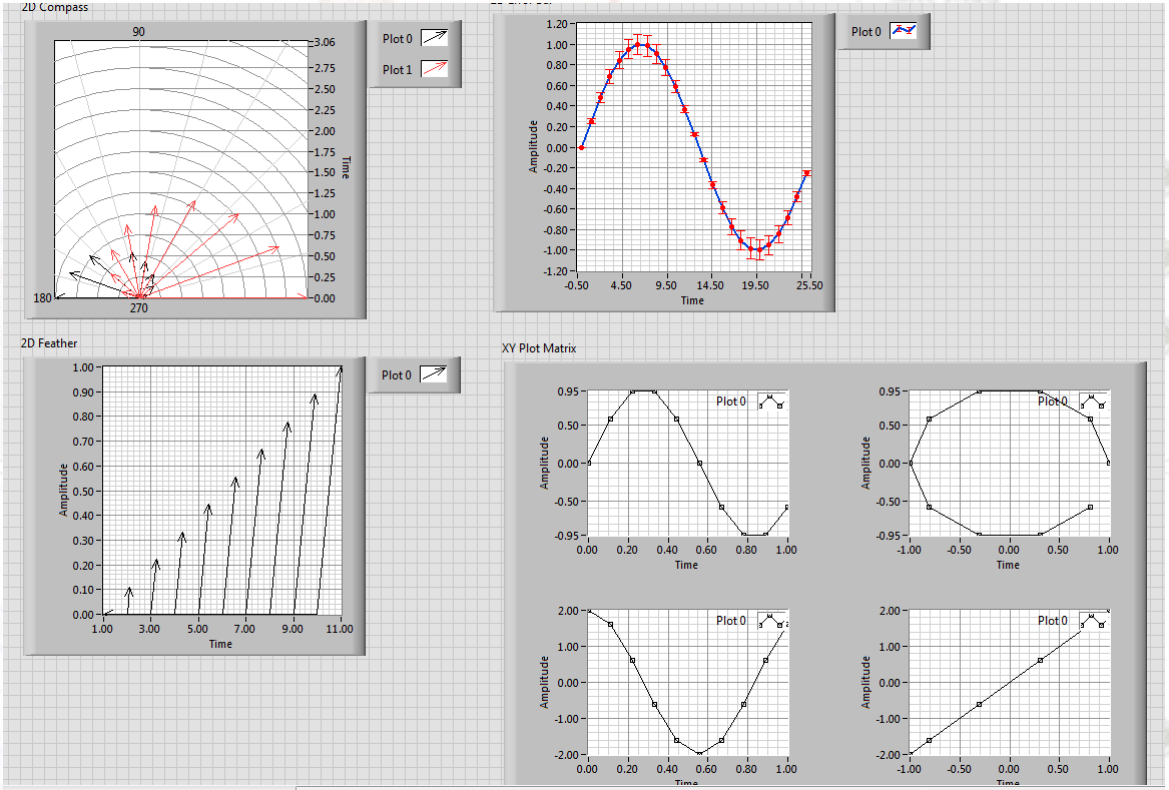
- Il digital waveform graph è molto utile quando si devono mostrare dati digitali e quando si lavora con analizzatori logici
- Il DWG accetta in ingresso waveform digitali, dati di tipo digitali ed array di queste tipologie di dati.
- Il DWG mostra le differenti waveform come linee digitali.
- Il Mixed Signal Graphs invece mostra sia dati analogici che digitali.
- MSG accetta tutti i tipi di dati accettati da waveform graphs, XY graphs, e digital waveform graphs
- Esempio 6

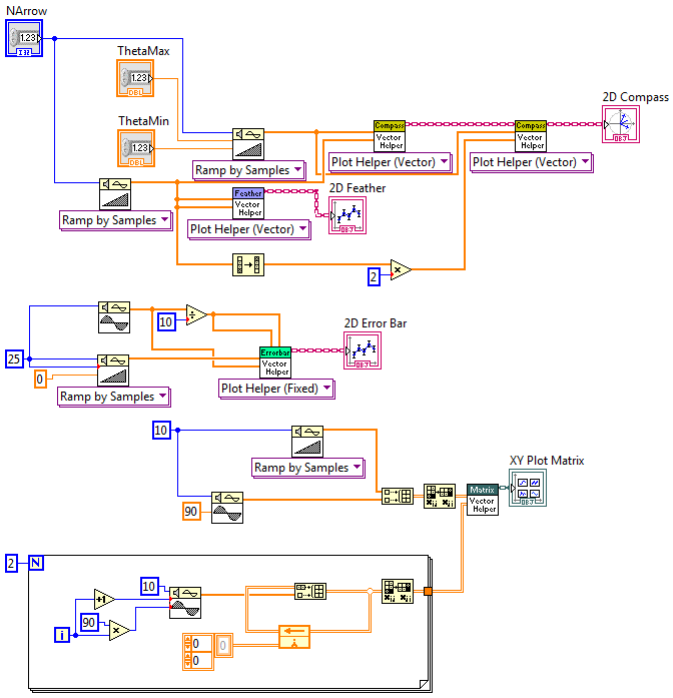




Grafici 2D

- **Compass Plot**—Graphs vectors that emanate from the center of a compass graph.
- **Error Bar Plot**—Graphs the error bar at each point above and below the line graph.
- **Feather Plot**—Graphs vectors that emanate from equally spaced points along a horizontal axis.
- **XY Plot Matrix**—Graphs rows and columns of scatter graphs.



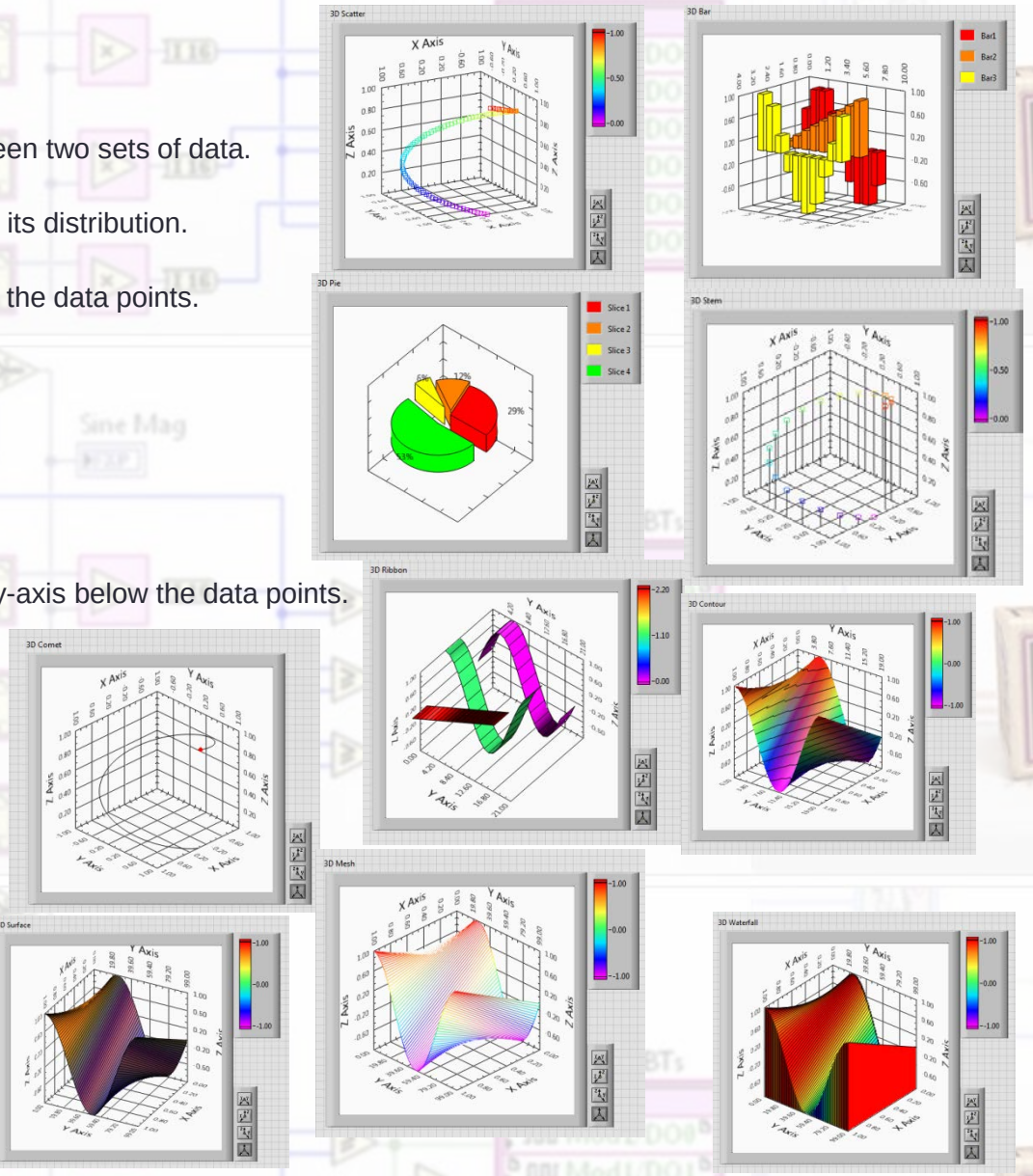


Esempio 7

Esempio 8

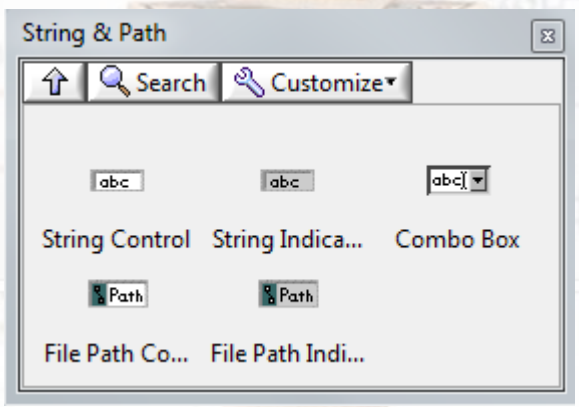
Grafici 3D

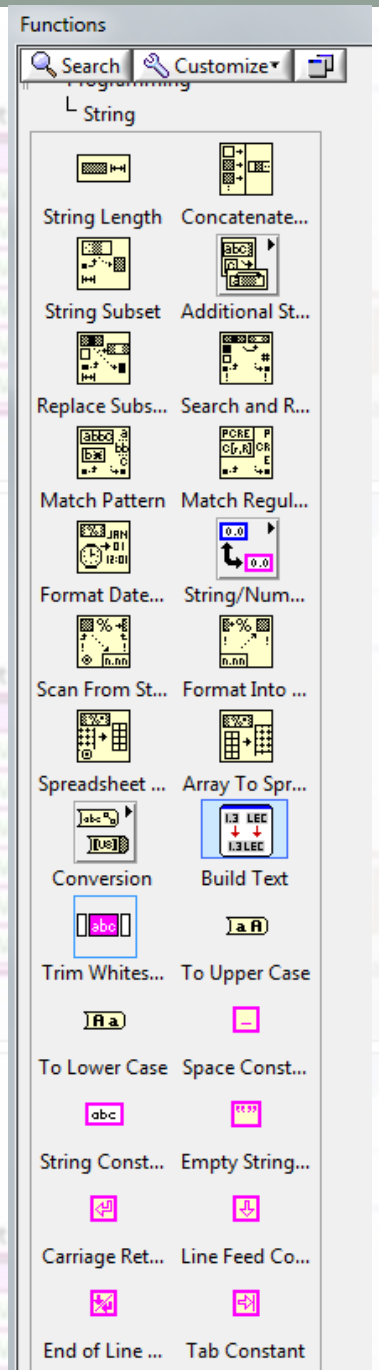
- **Scatter**
  - Shows trends in statistics and the relationship between two sets of data.
- **Stem**
  - Displays an impulse response and organize data by its distribution.
- **Comet**
  - Creates an animated graph with a circle that follows the data points.
- **Surface**
  - Graphs data with a connecting surface.
- **Contour**
  - Graphs a plot with contour lines.
- **Mesh**
  - Graphs a mesh surface with open spaces.
- **Waterfall**
  - Graphs the surface of the data and the area on the y-axis below the data points.
- **Quiver**
  - Generates a plot of normal vectors.
- **Ribbon**
  - Generates a plot of parallel lines.
- **Bar**
  - Generates a plot of vertical bars.
- **Pie**
  - Generates a pie chart.
- **3D Surface Graph**
  - Draws a surface in 3D space.
- **3D Parametric Graph**
  - Draws a parametric surface in 3D space.
- **3D Line Graph**
  - Draws a line in 3D space



String e file path

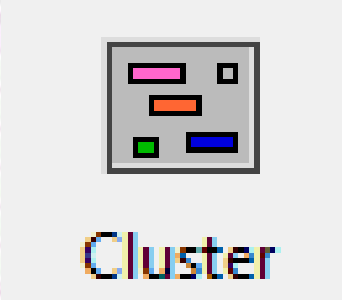
- Una stringa è una collezione di caratteri ASCII.
- In LabVIEW esistono molte funzioni per operare sulle stringhe e per convertire i numeri in stringhe e viceversa
- Il file path è un particolare tipo di stringa utilizzato per indicare il percorso di un file



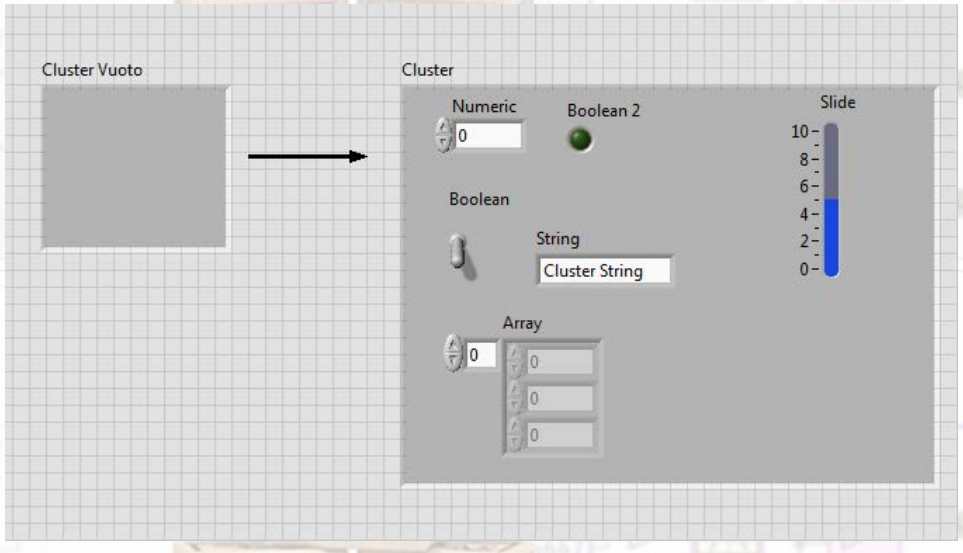


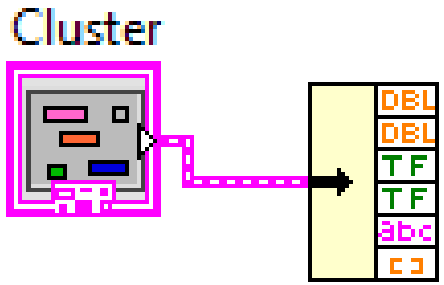
Esempio 1 per applicazione di alcune funzioni

Cluster



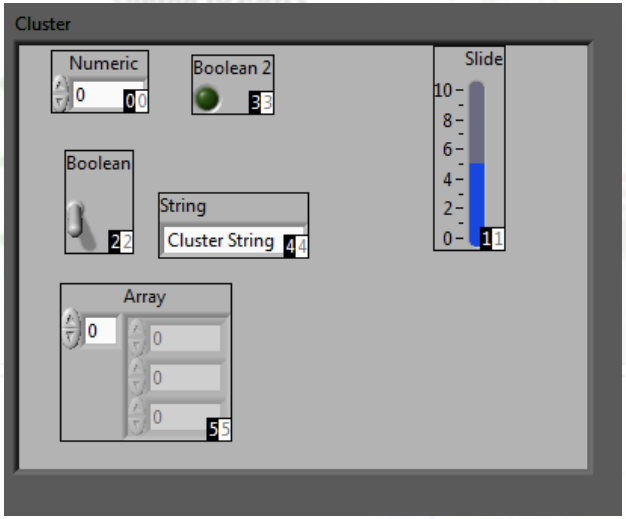
- Un **cluster**, analogo ad una *struct* in C, combina uno o più tipi di data in un unico tipo. Il cluster può contenere differenti tipi di dati quali booleani, interi, stringhe etc.
- L'utilizzo dei cluster permette di semplificare il layout di un block diagram.
- E' ovviamente possibile accedere ai singoli dati contenuti in un cluster mediante specifiche funzioni.
- Un cluster è concettualmente simile ad un array, ma può unire insieme oggetti di tipo differente

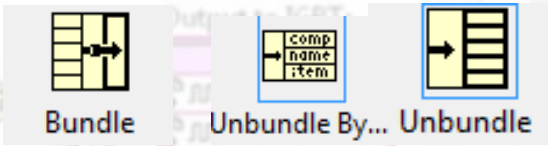


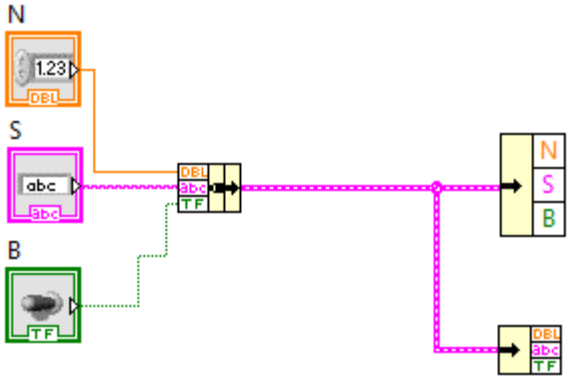


Cluster

- E’ possibile cambiare l’ordine degli elementi di un cluster per mezzo del comando **Reorder Controls in Cluster**
- Un cluster può essere assemblato anche direttamente nel bock diagram mediante la funzione *bundle*
- Per accedere ai singoli elementi di un cluster invece si utilizza la funzione opposta *unbundle* e *unbandle by name*

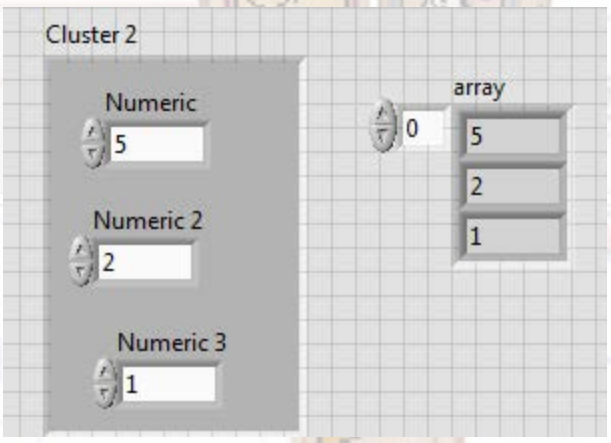


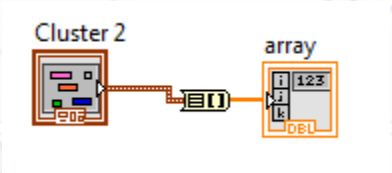




Cluster

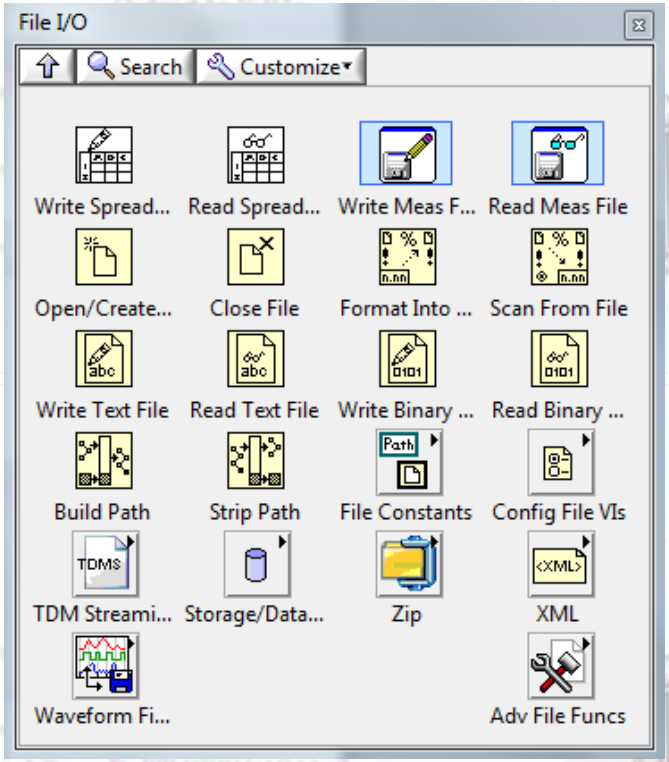
- Se un cluster contiene elementi tutti dello stesso tipo può essere convertito in un array o viceversa
- Più cluster possono essere uniti insieme per formare un array di cluster
- Esempio 2
  - Riempire delle schede per ogni studente che supera un esame e fra queste selezionare solo il nominativo degli studenti che hanno avuto la lode





File I/O

- Vediamo adesso come è possibile, in LabVIEW, effettuare la scrittura e la lettura di file di testo;
- Queste sono tra le operazioni più importanti lavorando con un sistema di sviluppo software, in quanto ci permettono di leggere e salvare informazioni in maniera permanente, sfruttando uno dei supporti di memorizzazione fissa del computer sul quale lavoriamo (dischi, memorie USB, memorie SD).
- Per la manipolazione dei file, LabVIEW dispone di una serie di blocchi specifici, che si trovano un'apposita sezione della control palette, denominata appunto File I/O e rappresentata.
- Alcune delle operazioni accessibili tramite i blocchi presenti in questa sezione sono:
  - apertura e chiusura di un file;
  - scrittura e lettura di un file di testo;
  - scrittura e lettura di un file binario.
- Per iniziare vediamo un semplice, ma significativo esempio, che permette di salvare una stringa, inserita dall'utente attraverso un controllo di tipo stringa, all'interno di un comune file di testo, il quale può essere aperto, per esempio, con Blocco Note di Windows.



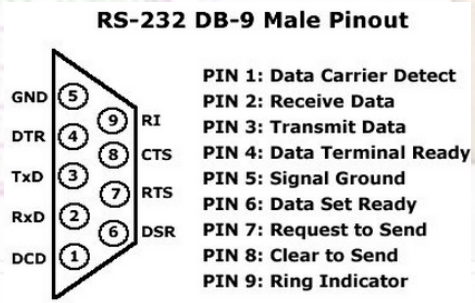
Protocolli di comunicazione in LabVIEW

- Iniziamo con l'analizzare quali sono le tipologie di interfacce disponibili per il collegamento di un device esterno (una scheda di acquisizione dati) al PC, nel caso si usi LabVIEW come ambiente di sviluppo del software.
- Naturalmente ogni bus ha le sue caratteristiche e la scelta di quello da utilizzare va considerata attentamente in funzione dell'applicazione finale.
- LabVIEW consente di gestire diversi protocolli di comunicazione, sia cablati che wireless, sia point-to-point che relativi a configurazioni più complesse (bus, reti, alberi).
- Oltre ai protocolli di comunicazione tipici delle applicazioni industriali, vengono forniti driver per la gestione di protocolli più evoluti come ad esempio l'SMTP, che è quello utilizzato per l'invio di e-mail.

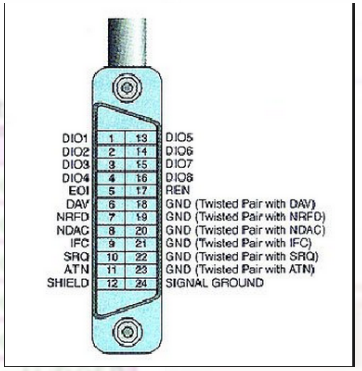
Protocolli di comunicazione in LabVIEW

- **Wired**

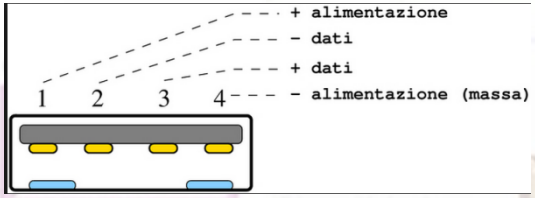
- **Seriale**: si tratta del protocollo usato dalle porte seriali del PC (la più comune è la RS-232, usata per le vecchie stampanti e i modem, prima dell'introduzione dell'USB). Questo tipo di interfaccia non è più molto utilizzato e tende sempre più a scomparire dai PC. Ciononostante i driver utilizzati per la sua gestione sono ancora attuali, in quanto utilizzati (per via della loro semplicità) pe la gestione delle periferiche USB usate in emulazione seriale (classe USB CDC).



- **GPIB** (IEEE-488): il General Purpose Interfac Bus (noto anche come IEEE-488) è un bus per l'interconnessione ed il controllo di strumentazione elettronica. Discende da uno standard proprietario della Hewlett-Packard, che lo aveva implementato per consentire l'interfacciamento verso i PC dei propri strumenti di misura. La sua caratteristica è quella di utilizzare un collegamento daisy-chain (ossia in serie, con il segnale passato da un dispositivo all'altro) per collegare fino ad un massimo di 15 dispositivi ad un'unica porta su PC. Avendo avuto una buona diffusione come standard intorno agli anni '80 del secolo scorso, è ancora piuttosto utilizzato.

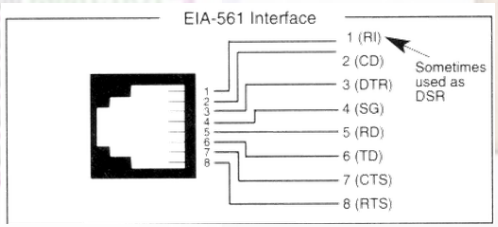


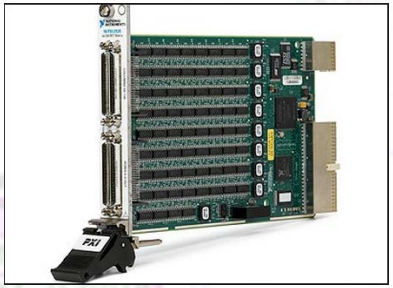
- **USB**: l'Universal Serial Bus è uno protocollo seriale usato per l'interfacciamento di periferiche di vario tipo al PC. Pur non essendo nato per applicazioni di misura, le sue caratteristiche di economicità, semplicità e larghezza di banda, hanno creato una certa diffusione anche in quest'ambito, principalmente per dispositivi low-cost e di prestazioni non elevate.



Protocolli di comunicazione in LabVIEW

- **Ethernet**: lo standard nato nel 1973 per l'implementazione di reti LAN ha trovato un notevole campo di applicazione anche nel settore delle misure elettroniche e del controllo industriale. NI produce una gamma di DAQ ethernet (NI ENET) che permettono l'acquisizione di segnali di vario tipo. Le schede sono costituite da un carrier (che ospita solo l'interfaccia) sul quale si possono connettere diversi tipi di cartucce. La modularità del sistema permette di riutilizzare i carrier e, nel caso sorgano nuove esigenze di misura, cambiare solo la cartuccia.
- **PCI e PCI Express**: le due ben note interfacce per l'interconnessione delle più svariate periferiche sulle motherboard dei PC sono utilizzate da National Instruments per l'implementazione di schede sia low-cost che ad elevate prestazioni
- **PC Card**: generalmente indicato come PCMCIA (Personal Computer Memory Card Interface Association) è uno standard di interconnessione per PC portatili, il cui scopo all'atto della sua creazione era quello di estendere le funzionalità dei PC portatili tramite l'interconnessione di periferiche esterne. Al giorno d'oggi lo standard soffre molto la concorrenza dell'USB
- **PXI**: è un bus proprietario NI nato nel 1997. Deriva dal più noto standard PCI (il suo acronimo vuol dire infatti Pci eXtensions for Instrumentation) e si distingue per le elevate prestazioni e la grande robustezza ed affidabilità, che lo hanno reso estremamente adatto all'impiego nel settore dell'automazione industriale. NI commercializza una vasta gamma di schede PXI, ed ha anche creato un'estensione basata su PCI Express, denominata PXI Express.





Protocolli di comunicazione in LabVIEW

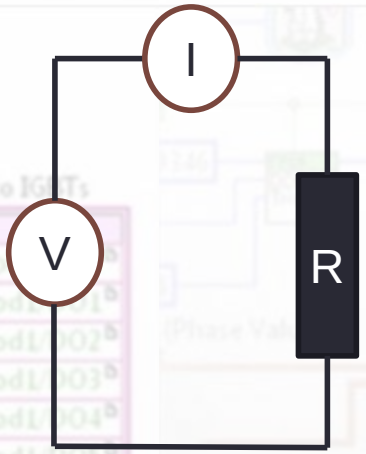
- **Wireless**
- **Bluetooth**: tramite LabVIEW è possibile gestire il protocollo Bluetooth sia come server che come client, sfruttando le interfacce commerciali che comunemente si trovano nei moderni Personal Computer.
- **Wi-Fi**: il protocollo, noto come IEEE 802.11, è gestito all'interno dell'ambiente LabVIEW e per esso è disponibile una serie di driver. Inoltre NI commercializza una famiglia di schede di acquisizione dati (per vari tipi di sensori) note come NI WLS.

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.
- **AC** : **A**lternating **C**urrent e **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 2^8 = 256
    - Con una parola di 12 bit è 2^12 = 4096
    - Con una parola di 16 bit è 2^16 = 65536
    - Con una parola di 24 bit è 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 e10 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.

Introduzione all'acquisizione dati

- **ADC Esempio**
  - Per maggior chiarezza ricorriamo ad un esempio: Vogliamo misura il valore di una resistenza R=100Ω mediante un amperometro (con fondo scala I_{max}=1°) quando ai capi della resistenza è applicata una tensione costante V=5V. Il valore di corrente è letto mediante due ADC di 8bit e 12bit rispettivamente.
  - Il valore digitale della corrente misurato è dato $N_I = I*N_{max}/I_{max}$ dove N_{max} vale 256 per l'ADC a 8 bit e 4096 per quello a 12bit.
  - Nel nostro caso avremo N_I (8bit)=3 e N_I(12bit)=41 con I=50mA.
  - Si dimostra che $|\Delta R/R| = |\Delta N/N|$ e pertanto, assumendo $\Delta N=1$, si ha una precisione maggiore nella misura di R se si usa un ADC a 12 bit
  - $|\Delta R/R|$ (8bit) = **33.3%** e $|\Delta R/R|$ (12bit) = **2.4%**



Introduzione all'acquisizione dati

- **DAQ : Data Acquisition**
  - Acronimo per indicare l'operazione della raccolta dei dati che, nella maggior parte dei casi, avviene effettuando una conversione ADC.
- **DAC** : **D**igital to **A**nalog **C**onversion
  - L'operazione duale della ADC .
- **DMA** : **D**irect **M**emory **A**ccess
  - 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.
- 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

Grandezza Fisica	Trasduttore
Temperatura	Termocoppia Rilevatore di temperatura a resistenza Termistore Sensore a circuito integrato
Luce	Tubo fotomoltiplicatore Fotodiodo Cella fotoconduttiva
Suono	Microfono
Forza e pressione	Strain gauge Trasduttore piezoelettrico
Posizione (spostamento)	Potenziometro Trasformatore differenziale lineare Encoder ottico

Introduzione all'acquisizione dati

- In generale tutti i segnali sono variabili nel tempo; essi possono essere inquadrati in due grandi categorie:
  - segnali digitali
  - segnali analogici
- I segnali digitali (o segnali binari) ammettono due soli livelli discreti: un livello detto "high level" (**On**) ed un livello detto "low level" (**Off**).
- I segnali digitali si suddividono in due ulteriori tipi:
  - 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. Lo strumento in grado di misurare questo tipo di segnale è semplicemente un "digital state detector".
  - Segnali 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.

Introduzione all'acquisizione dati

- I segnali analogici contengono informazioni sulla variazione continua del segnale rispetto al tempo.
- I **segnali analogici** si suddividono in tre tipi:
  - Segnali analogici 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.
  - Segnali analogici 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.
  - 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.

- Esempio 1: Acquisizione di un segnale e calcolo della Power Spectral Density
  - La densità spettrale di un onda, se moltiplicata per un fattore opportuno, restituisce la potenza trasportata dall'onda per unità di frequenza, nota come PSD del segnale

Componenti di un sistema di acquisizione dati

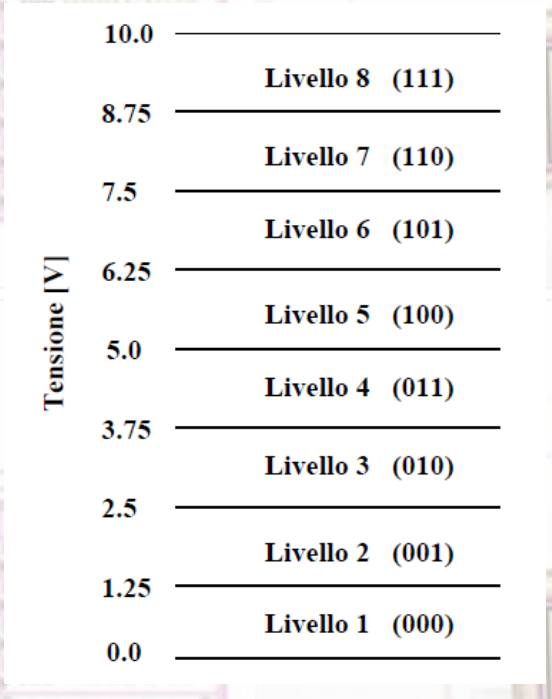
- La catena che costituisce un sistema di acquisizione dati è costituita dai seguenti elementi:
  1. generazione fisica del segnale,
  2. sensore o trasduttore che converta il segnale fisico in un segnale elettrico come una tensione o una corrente,
  3. eventuale amplificazione del segnale in uscita 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.

I parametri fondamentali di un scheda di acquisizione

- 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:
  - **risoluzione** della scheda: questo parametro è stato esaminato in dettaglio in precedenza parlando di conversione analogico digitale;
  - **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;
  - **guadagno**: sta ad indicare una qualunque operazione di amplificazione o di attenuazione del segnale prima che esso venga digitalizzato.

I parametri fondamentali di un scheda di acquisizione

- 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: 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), e cosi via.
- Nell'intervallo di misura 0 ÷ 5 V il convertitore A/D utilizza solo quattro delle otto sottodivisioni 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.



I parametri fondamentali di un scheda di acquisizione

- 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:

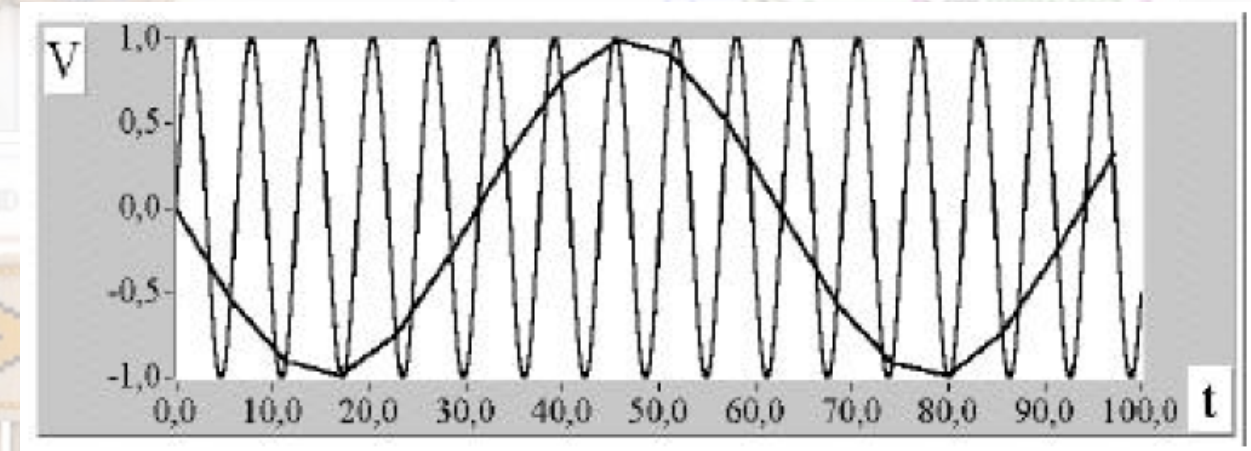
$$code\ width = \frac{MaxV - MinV}{Gain \times 2^{Nbit}}$$

- 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:

$$\frac{10}{1 \times 2^{12}} = 2.4mV \qquad \frac{20}{1 \times 2^{12}} = 4.8mV$$

I parametri fondamentali di un scheda di acquisizione

- **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.
- Secondo il **Teorema del Campionamento di Nyquist (http://it.wikipedia.org/wiki/Teorema_del_campionamento_di_Nyquist-Shannon)**, 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



I parametri fondamentali di un scheda di acquisizione

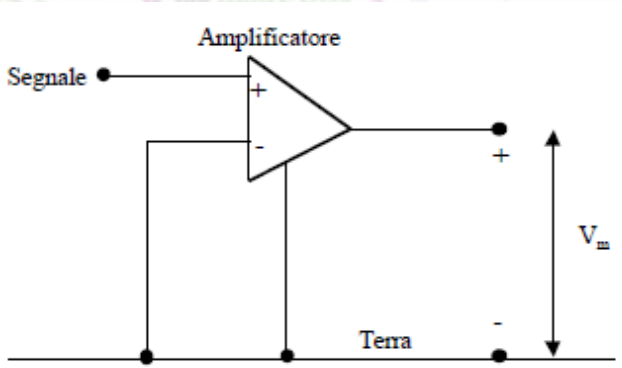
- **filtri per attenuare il rumore** : prima di essere convertito in un segnale digitale, il segnale analogico in generale è soggetto a distorsioni a causa della presenza del rumore, la cui sorgente può avere l'origine più svariata.
- Si possono in generale presentare i tre casi seguenti:
  - rumore 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 $1/\sqrt{Numero\ di\ Punti\ Mediati}$
  - Per esempio, se si esegue la media su 100 punti, l'effetto del rumore sul segnale è ridotto di un fattore 0,1.
  - Vedere Esempio 2 in cui una forma d'onda con rumore bianco è ricampionata
  - Vedere esempio 4 per alcune applicazioni dei filtri.

Importanza del riferimento di terra in una misura di tensione

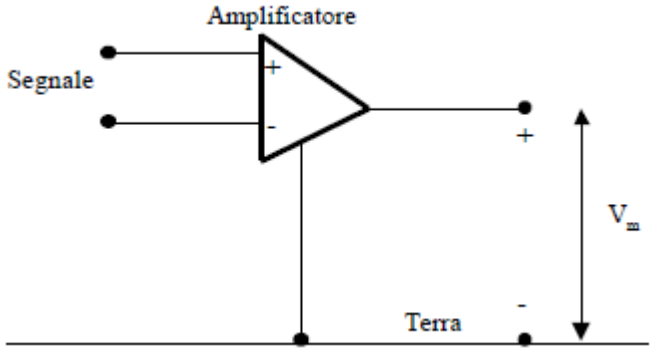
- Come già detto in precedenza, tra il fenomeno fisico da studiare mediante acquisizione dati ed il sistema di acquisizione è sempre interposto un trasduttore che converte la grandezza da acquisire in un segnale di tensione.
- La tensione è sempre la misura di una differenza di potenziale tra due punti. Uno di questi due punti è, in genere, assunto come riferimento e gli è assegnato un valore nullo di tensione (**ground** o **terra**).
- In generale una moderna scheda di acquisizione dati può essere configurata in tre differenti modi:

Importanza del riferimento di terra in una misura di tensione

- **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 figure
  - La maggior parte delle apparecchiature che generano segnali di questo tipo è costituito da generatori di segnali connessi alla rete di alimentazione.



- **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 figura La maggior parte delle apparecchiature che generano segnali di questo tipo è costituito da trasformatori, batterie, ecc.



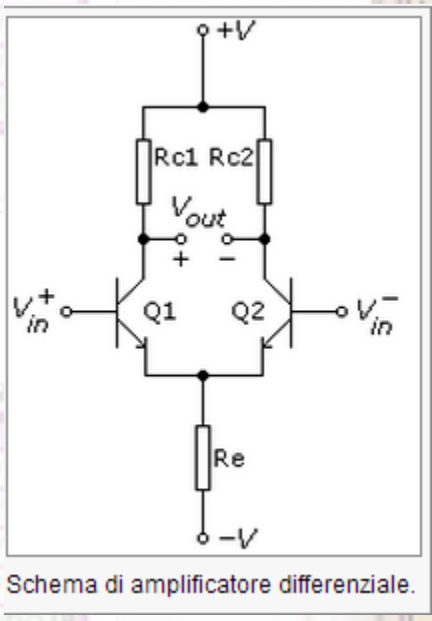
- Si utilizzano gli ingressi single-ended quando si devono acquisire segnali elevati (superiori a 1 V) ed i cavi che collegano la sorgente del segnale all'hardware non superano i tre metri di lunghezza.
- E' evidente che un disturbo che si sovrappone al segnale entrando sulla linea che collega il sensore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

Importanza del riferimento di terra in una misura di tensione

- Segnali di tipo **Differenziale**
- Un amplificatore differenziale presenta tre terminali:
  - un terminale che coincide con la terra comune,
  - un terminale al cui capo è presente la differenza di tensione V_{in}^{+} misurata rispetto a terra,
  - un terminale al cui capo è presente la differenza di tensione V_{in}^{+} invertita, che chiameremo V_{in}^{-}.

$$V_{\text{out}} = A_{\text{d}}(V_{\text{in}}^{+} - V_{\text{in}}^{-}) + A_{\text{c}}\left(\frac{V_{\text{in}}^{+} + V_{\text{in}}^{-}}{2}\right)$$

- in cui A_d è il guadagno di modo differenziale e A_c il guadagno di modo comune.
- 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.
- Vedi esempio 3 per un esempio di amplificatore differenziale

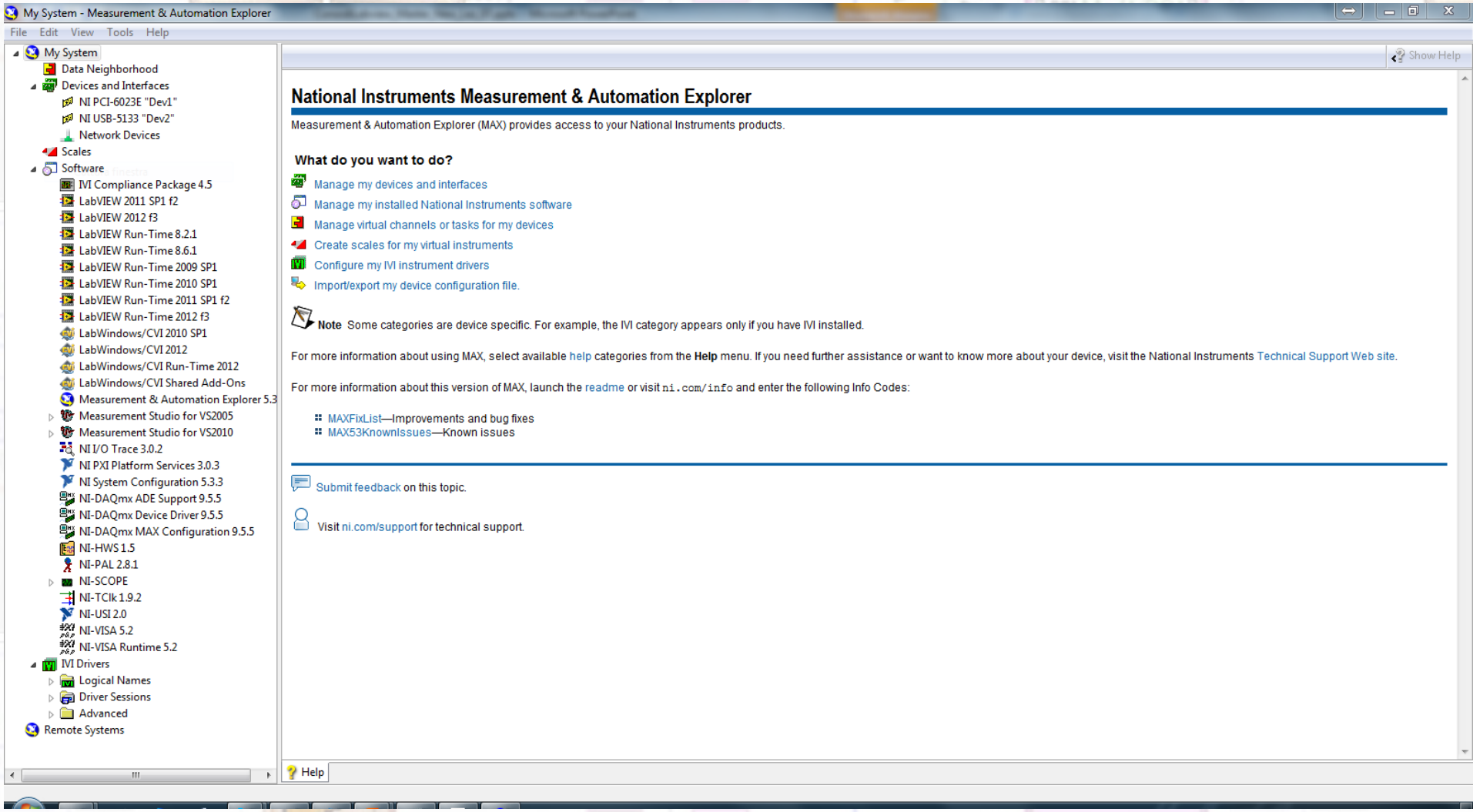


Schema di amplificatore differenziale.

NI-DAQmx

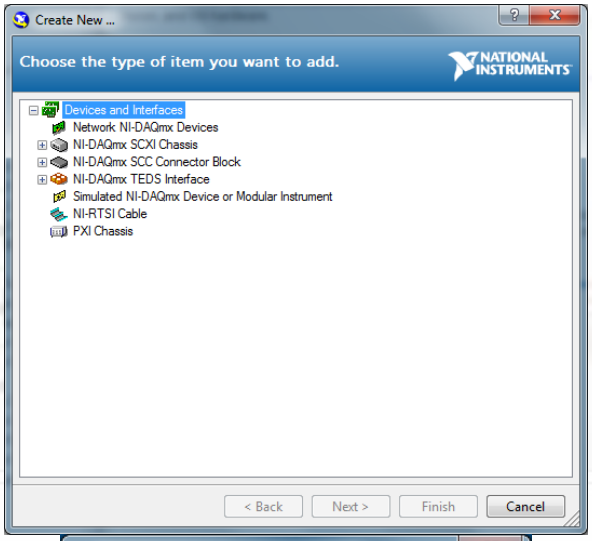
- Passiamo ora a descrivere il software NI-DAQmx che permette di configurare gli strumenti che poi useremo in ambiente LabVIEW
- Come detto nella prima lezione questo software è scaricabile gratuitamente dal seguente link:
  - http://joule.ni.com/nidu/cds/view/p/id/3811/lang/it
- NI-DAQmx permette di configurare gli strumenti collegati al PC, sia direttamente che via rete, per poterli utilizzare in diversi ambienti di sviluppo, anche differenti da LabVIEW

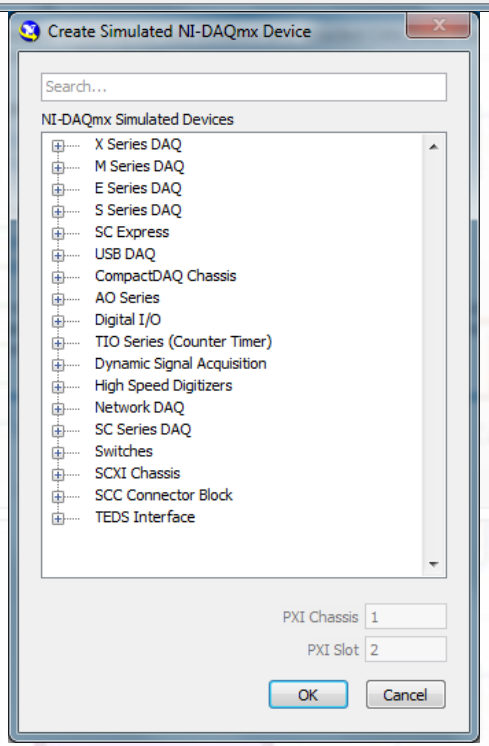
NI-DAQmx


My System - Measurement & Automation Explorer
File Edit View Tools Help
My System
Data Neighborhood
Devices and Interfaces
NI PCI-6023E "Dev1"
NI USB-5133 "Dev2"
Network Devices
Scales
Software
IVI Compliance Package 4.5
LabVIEW 2011 SP1 f2
LabVIEW 2012 f3
LabVIEW Run-Time 8.2.1
LabVIEW Run-Time 8.6.1
LabVIEW Run-Time 2009 SP1
LabVIEW Run-Time 2010 SP1
LabVIEW Run-Time 2011 SP1 f2
LabVIEW Run-Time 2012 f3
LabWindows/CVI 2010 SP1
LabWindows/CVI 2012
LabWindows/CVI Run-Time 2012
LabWindows/CVI Shared Add-Ons
Measurement & Automation Explorer 5.3
Measurement Studio for VS2005
Measurement Studio for VS2010
NI I/O Trace 3.0.2
NI PXI Platform Services 3.0.3
NI System Configuration 5.3.3
NI-DAQmx ADE Support 9.5.5
NI-DAQmx Device Driver 9.5.5
NI-DAQmx MAX Configuration 9.5.5
NI-HWS 1.5
NI-PAL 2.8.1
NI-SCOPE
NI-TClk 1.9.2
NI-USI 2.0
NI-VISA 5.2
NI-VISA Runtime 5.2
IVI Drivers
Logical Names
Driver Sessions
Advanced
Remote Systems
Show Help
National Instruments Measurement & Automation Explorer
Measurement & Automation Explorer (MAX) provides access to your National Instruments products.
What do you want to do?
Manage my devices and interfaces
Manage my installed National Instruments software
Manage virtual channels or tasks for my devices
Create scales for my virtual instruments
Configure my IVI instrument drivers
Import/export my device configuration file.
Note Some categories are device specific. For example, the IVI category appears only if you have IVI installed.
For more information about using MAX, select available help categories from the Help menu. If you need further assistance or want to know more about your device, visit the National Instruments Technical Support Web site.
For more information about this version of MAX, launch the readme or visit ni.com/info and enter the following Info Codes:
MAXFixList—Improvements and bug fixes
MAX53KnownIssues—Known issues
Submit feedback on this topic.
Visit ni.com/support for technical support.
Help

NI-DAQmx

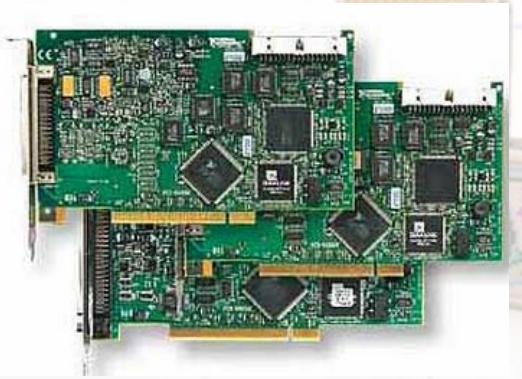
- Una della caratteristiche più interessanti di NI-DAQmx è quella di poter creare dei device simulati.
- Un device simulato è una replica di uno reale con il quale è possibile scrivere e testare un programma senza collegare al pc alcun dispositivo.
- Per creare un nuovo device cliccate col tasto destro su **My System>>Devices and Interfaces** e selezioante **Create New...** quindi selezionate **Simulated NI-DAQmx Device or Modular Instrument**.
- Selezione a questo punto il dispositivo che volete simulare dalla lista dei dispositivi supportati.
- Ovviamente NI-DAQmx supporta solo i dispositivi della National Instruments
- Creiamo come esempio il dispositivo NI-PCI-6023E





NI-PCI-6023E

- DAQ multifunzione con 16 input analogici a 12 bit, 200 kS/s
- Informazioni
  - http://sine.ni.com/nips/cds/view/p/lang/it/nid/10967
- NI CB-68LP Blocco connettore I/O a 68-pin non schermato



Models		NI 6023E, NI 6024E, NI 6025E
Measurement Sensitivity[1] (mV)		0.008
Nominal Range (V)		Absolute Accuracy (mV)
Positive FS	Negative FS	
10	-10	16.504
5	-5	5.263
2.5	-2.5	–
2	-2	–
1	-1	–
0.5	-0.5	0.846
0.25	-0.25	–
0.2	-0.2	–
0.1	-0.1	–
0.05	-0.05	0.106
10	0	–
5	0	–
2	0	–
1	0	–
0.5	0	–
0.2	0	–
0.1	0	–

[1]Smallest detectable voltage change in the input signal at the smallest input range.

Family	Bus	Analog Inputs	Input Resolution	Max Sampling Rate	Input Range	Analog Outputs	Output Resolution	Output Rate	Output Range	Digital I/O	Counter/Timers	Triggers
NI 6023E	PCI	16 SE/8 DI	12 bits	200 kS/s	±0.05 to ±10 V	0	-	-	-	8	2, 24-bit	Digital

NI-MyDAQ

Analog Input

Number of channels ..... 2 differential or 1 stereo audio input

ADC resolution ..... 16 bits

Maximum sampling rate ..... 200 kS/s

Timing accuracy ..... 100 ppm of sample rate

Timing resolution ..... 10 ns

Range
- Analog input ..... ±10 V, ±2 V, DC-coupled
- Audio input ..... ±2 V, AC-coupled

Passband (-3 dB)
- Analog input ..... DC to 400 kHz
- Audio input ..... 1.5 Hz to 400 kHz

Connector type
- Analog input ..... Screw terminals
- Audio input ..... 3.5 mm stereo jack

Input type (audio input) ..... Line-in or microphone

Microphone excitation (audio input) ..... 5.25 V through 10 kΩ

Absolute accuracy

Nominal Range			
Positive Full Scale	Negative Full Scale	Typical at 23 °C (mV)	Maximum (18 to 28 °C) (mV)
10	-10	22.8	38.9
2	-2	4.9	8.6

Input FIFO size ..... 4,095 samples, shared among channels used

Maximum working voltage for analog inputs (signal + common mode) ..... ±10.5 V to AGND

Common-mode rejection ratio (CMRR) (DC to 60 Hz) ..... 70 dB

Input impedance
- Device on
  - AI+ or AI- to AGND ..... >10 GΩ || 100 pF
  - AI+ to AI- ..... >10 GΩ || 100 pF
- Device off
  - AI+ or AI- to AGND ..... 5 kΩ
  - AI+ to AI- ..... 10 kΩ

Anti-aliasing filter ..... None

Overvoltage protection AI+ or AI - to AGND ..... ±16 V

Overvoltage protection (audio input left and right) ..... None

Analog Output

Number of channels ..... 2 ground-referenced or 1 stereo audio output

DAC resolution ..... 16 bits

Maximum update rate ..... 200 kS/s

Range
- Analog output ..... ±10 V, ±2 V, DC-coupled
- Audio output ..... ±2 V, AC-coupled

Maximum output current (analog output)[1] ..... 2 mA

Output impedance
- Analog output ..... 1 Ω
- Audio output ..... 120 Ω

Minimum load impedance (audio output) ..... 8 Ω

NI-MyDAQ

Connector type
- Analog output ..... Screw terminals
- Audio output ..... 3.5 mm stereo jack

AC-coupling high-pass frequency (audio output with 32 Ω load) ..... 48 Hz

Absolute accuracy

Nominal Range			
Positive Full Scale	Negative Full Scale	Typical at 23 °C (mV)	Maximum (18 to 28 °C) (mV)
10	-10	19.6	42.8
2	-2	5.4	8.8

Slew rate ..... 4 V/µs

Timing accuracy ..... 100 ppm of sample rate

Timing resolution ..... 10 ns

Overdrive protection ..... ±16 V to AGND

Maximum power-on voltage[1] ..... ±110 mV

Output FIFO size ..... 8,191 samples, shared among channels used

Digital I/O

Number of lines ..... 8; DIO <0..7>

Direction control ..... Each line individually programmable as input or output

Update mode ..... Software-timed

Pull-down resistor ..... 75 kΩ

Logic level ..... 5 V compatible LVTTL input; 3.3 V LVTTL output

V_{IH} min ..... 2.0 V

V_{IL} max ..... 0.8 V

Maximum output current per line[2] ..... 4 mA

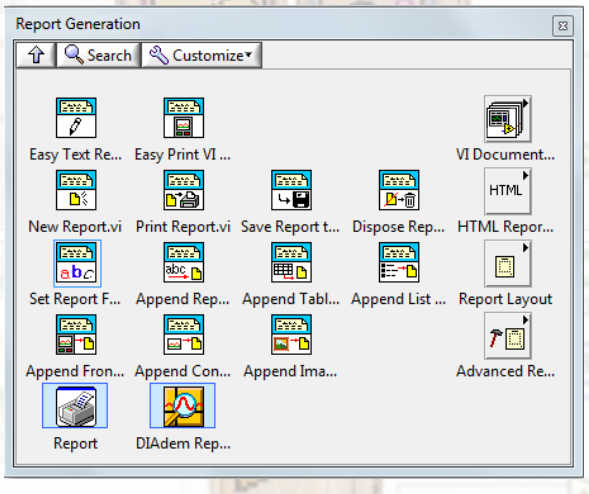
http://www.ni.com/pdf/manuals/373061f.pdf

DAQ Assistant

- National Instruments ha sviluppato una serie di strumenti software che semplificano notevolmente il lavoro del programmatore. Uno di questi è, appunto, il **DAQ Assistant**: si tratta di un modulo che permette di creare, editare ed infine eseguire, un task, sfruttando la collezione di driver contenuti all'interno del pacchetto NIDAQmx.
- A questo punto è importante chiarire il concetto di task in LabVIEW: un task è una collezione di uno o più canali virtuali caratterizzati da una serie di parametri specifici (frequenza di campionamento, triggering ed altro). In sostanza, il task è l'interfaccia tra la scheda di acquisizione ed il software di alto livello che gira sul PC.
- La caratteristica dei task è di poter operare simultaneamente, consentendo quindi di acquisire dati da più periferiche in contemporanea, senza essere vincolati ad utilizzarle una alla volta.
  - La configurazione dei task può avvenire anche utilizzando strumenti diversi dal DAQ Assistant, che però rimane il metodo più veloce per chi ha iniziato da poco ad usare LabVIEW.

Report Generation

- LabVIEW dispone nativamente di una serie di blocchi dedicati al *report generation*, che si trovano nella control palette (relativa ai block diagram) all'interno del percorso *Report Generation*.
- Alcuni di questi blocchi sono accessibili solamente se si dispone del *report generation toolkit*, mentre altri sono attivi direttamente con la semplice suite di base.
- In pratica, con i blocchi di base è possibile generare report stampati (viene quindi gestito uno spooler di stampa) o HTML (pubblicabili su una pagina web).



Report Generation

Report Type	Descrizione
0	**Standard Report:** crea un report solo stampabile.
1	**HTML:** crea un report HTML, che può essere salvato, pubblicato sul web o stampato.
2	**Word:** crea un report (nuovo o da un template) su un documento di Microsoft Word, che può essere salvato, modificato o stampato. Necessita del *report generation toolkit*.
3	**Excel:** crea un report (nuovo o da un template) su un documento di Microsoft Excel, che può essere salvato, modificato o stampato. Necessita del *report generation toolkit*.

- Facciamo una carrellata sui blocchi più importanti che possono essere utilizzati.
- *New Report*: questo è il blocco di partenza, che crea il nuovo report e lo predispone per l'invio allo spooler di stampa o ad altri applicativi (per esempio una pagina di Internet Explorer o un documento di Microsoft Word).
  - La destinazione del report viene decisa in questo blocco, tramite l'ingresso *report type*. Possono essere generati quattro tipi di output differenti.
- *Set Report Font*: permette di settare font, colori, allineamenti e molte altre proprietà del testo del report.
- *Append Report Text*: inserisce delle stringhe di testo all'interno del report. Il report selezionato è quello collegato all'ingresso *report in*.
- *Append Table to Report*: inserisce un array bidimensionale (matrice) all'interno del report, sotto forma di tabella. La larghezza delle celle può essere definita dall'utente.

Report Generation

- *Append Control Image to Report*: inserisce l'immagine di un oggetto del front panel (passato con un reference) all'interno del report. L'oggetto da aggiungere viene selezionato creando una reference e passandola all'ingresso *ctrl reference* del blocco.
- *Append Image to Report*: inserisce un'immagine generica all'interno del report. Il path dell'immagine da inserire viene passato tramite l'ingresso *path or URL of image*.
- *Dispose Report*: chiude il report e rilascia tutta la memoria da esso impiegata.

Panoramica sui moduli aggiuntivi

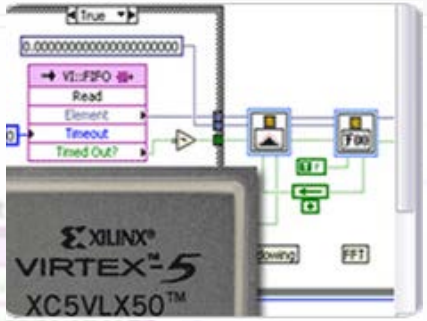
- LabVIEW dispone di una serie di moduli e di toolkit aggiuntivi, acquistabili separatamente (ma sono comunque disponibili delle versioni di valutazione a scadenza) che permettono di espandere le funzionalità della suite di sviluppo. Vediamo qui di seguito quali sono i più importanti.
- **LabVIEW Mobile Module**: questo modulo è utilizzato per sviluppare applicazioni per i più recenti handheld device e smartphone. Con questo toolkit aggiuntivo è possibile creare sistemi di misura portatili che acquisicono, analizzano e visualizzano dati, usando i dispositivi della linea DAQ NI. Il modulo consente inoltre di sfruttare le connessioni wireless (comunemente presenti su PDA e smartphone) per interfacciarsi con strumentazione esistente.
  - http://www.ni.com/labview/mobile.htm
  - http://www.ni.com/white-paper/7705/en

Panoramica sui moduli aggiuntivi

- **LabVIEW Real-Time Module**: il modulo realtimedi LabVIEW, unito alla serie RT dei prodotti hardware NI, permette lo sviluppo di applicazioni deterministiche, con performance di tipo real-time, per applicazioni di misura e controllo. Sfruttando le caratteristiche della programmazione grafica tramite il Linguaggio-G, è possibile sviluppare la propria applicazione di controllo su una macchina desktop e trasferirla su un target indipendente. L'hardware real-time proposto da NI comprende un processore embedded, dotato di un suo RTOS (RealTimeOperetingSysytem), memoria onboard, local storage ed accesso a periferiche come porte seriali, ethernet, GPIB, USB, ecc.
  - http://sine.ni.com/nips/cds/view/p/lang/it/nid/2381
  - http://www.ni.com/labview/realtime/i/

Panoramica sui moduli aggiuntivi

- **LabVIEW FPGA Module**: questo modulo aggiuntivo permette di estendere le funzionalità di LabVIEW e della programmazione grafica alle FPGA (Field Programmable Gate Arrays) dei moduli NI RIO (Reconfigurable IO). La programmazione tramite LabVIEW è particolarmente indicata per le logiche programmabili, in quanto realizza in maniera nativa i parallelismi ed il flusso dati.
- Tramite questo modulo è possibile realizzare hardaware customizzato per applicazioni di misura e controllo, senza dover utilizzare linguaggi di basso livello per la gestione dell'hardware. L'uso di FPGA permette di realizzare applicazioni ad altissima velocità, interfacciarsi con protocolli di comunicazione standard, gestire comunicazioni a radiofrequenza e molto altro ancora.
  - http://sine.ni.com/nips/cds/view/p/lang/it/nid/11834
  - http://www.ni.com/labview/fpga/i/



Panoramica sui moduli aggiuntivi

- **LabVIEW WSN Module**: le Wireless Sensor Network (WSN) costituiscono un interessantissimo campo di applicazione delle tecnologie elettroniche che in questi ultimi anni sono diventate di grande attualità. Il modulo NI WSN Pioneer estende le funzionalità del linguaggio alla programmazione dei nodi NI, permettendo lo sviluppo e la distribuzione di applicazioni basate su WSN. Usando il WSN Module è possibile estendere la vita delle batterie ottimizzando il funzionamento dei dispositivi, includere routine di analisi ed acquisizione dati, fornire intelligenza e facoltà decisionali ai singoli nodi. Il modulo fornisce, inoltre, la possibilità di aggiornare il software residente sui dispositivi in remoto, sfruttando il canale wireless.
  - http://sine.ni.com/nips/cds/view/p/lang/it/nid/207289
  - http://www.ni.com/labview/wsn/i/

Panoramica sui moduli aggiuntivi

- ***LabVIEW Touch-Panel Module***: il modulo *Touch Panel* di LabVIEW permette lo sviluppo di applicazioni Human Machine Interface (HMI) in grado di comunicare con i moduli hardware real time di National Instruments, come ad esempio CompactFieldPoint, Compact RIO e CompactVisionSystem. I sistemi HMI costituiscono un'efficace sistema per la visualizzazione delle informazioni ed il controllo delle applicazioni.
  - http://sine.ni.com/nips/cds/view/p/lang/it/nid/202908
  - http://www.ni.com/labview/touch-panel/i/

Panoramica sui moduli aggiuntivi

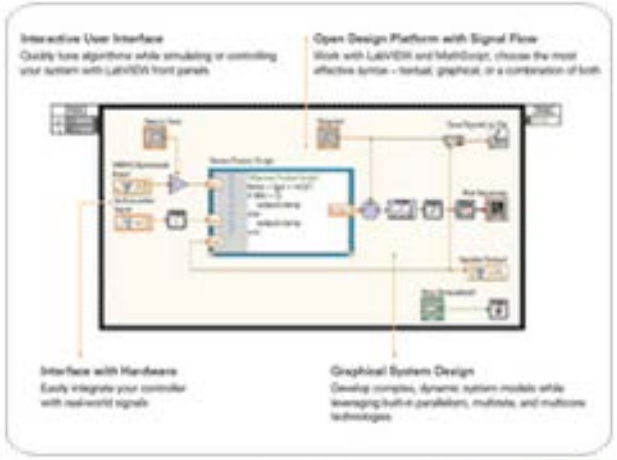
- **LabVIEW Vision Development Module**: il Vision Development Module è una ricca libreria con centinaia di algoritmi di image-processing e computer vision. I blocchi presenti nella libreria coprono problematiche di computer vision come l'image enhancements, il controllo di presenza, l'indentificazione di oggetti e la loro localizzazione nello spazio.
  - http://www.ni.com/labview/vision/i/

- **LabVIEW Report Generation Toolkit**: il Report generation Toolkit for MS Offi ce di LabVIEW è una libreria di VI che permettono la realizzazione di report su MS Word e MS Excel. Oltre ai blocchi base, il toolkit è stato arricchito da una serie di blocchi express altamente specializzati che permettono lo sviluppo di report customizzati in un tempo ancora inferiore.
  - http://sine.ni.com/nips/cds/view/p/lang/it/nid/209050
  - http://www.ni.com/white-paper/5900/en

- ***LabVIEW Robotics Module***: il Robotics Module è un toolkit aggiuntivo che permette la progettazione di software per il controllo robotico. Il modulo comprende un libreria per la gestione di sensori ed attuatori di uso comune in robotica, algoritmi di controllo e funzioni di movimento.
- Alcune delle tipologie di robot di movimentorealizzabili sono le seguenti:
  - veicoli autonomi e semiautonomi, inclusi veicoli agricoli e militari;
  - piattaforme di soccorso;
  - veicoli sottomarini ed UAV (Ummanned Aerial Veicles);
  - robot di servizio;
  - dispositivi robotici per il settore medico.
  - http://www.ni.com/labview/robotics/i/

Panoramica sui moduli aggiuntivi

- **LabVIEW Control Design and Simulation Module**: tramite questo modulo è possibile analizzare il comportamento in anello aperto di un determinato sistema, progettarne il controllore in anello chiuso, simulare il sistema e realizzarne l'implementazione fisica.
  - http://www.ni.com/labview/cd-sim/i/



Panoramica sui moduli aggiuntivi

- LabVIEW Datalogging and Supervisory Control Module: il modulo DSC (Datalogging and Supervisory Control) aggiuntivo di LabVIEW è lo strumento ideale per lo sviluppo di applicazioni HMI/SCADA o di datalogging con alto numero di canali in ingresso. Il modulo include tools per il datalogging, controllo real-time, controllo di allarmi ed eventi ed altro ancora.
  - http://www.ni.com/labview/labviewdsc/i/