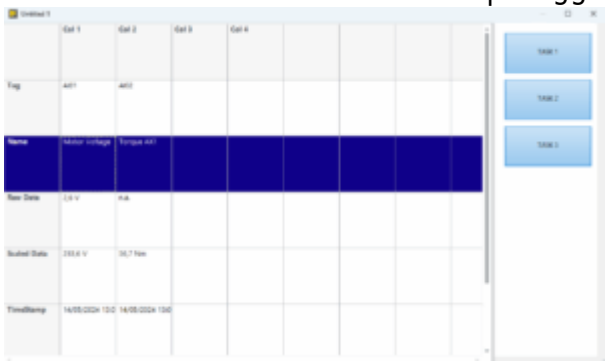
GUI Responsive con Multi-Listbox in LabVIEW

Category: LabVIEW Talk
scritto da Davide Oratelli | Maggio 24, 2025



Sviluppando software con LabVIEW, la creazione di interfacce utente intuitive e responsive è fondamentale per migliorare l'ergonomia e l'esperienza d'uso dell'utente finale.

Le multi-listbox rappresentano un controllo versatile che consente di visualizzare dati strutturati su più colonne, combinando l'aspetto organizzato di una tabella con le funzionalità di selezione tipiche di un listbox.

Questo articolo fornisce una panoramica su come integrare e personalizzare questo controllo, cercando di trasmettere la nostra esperienza, segnalandone vantaggi, limitazioni e trick per ottenere il massimo in termini di design e interattività.



Esempio: Schermata di un'applicazione LabVIEW che mostra una multi-listbox ben strutturata, con colonne chiaramente definite.

Panoramica delle Multi-Listbox in LabVIEW

Le [redacted] sono controlli grafici che permettono di visualizzare dati organizzati in più colonne.

A differenza dei tradizionali listbox a colonna singola, questi controlli offrono una presentazione strutturata simile a quella di una tabella, mantenendo però la facilità d’uso di un controllo di selezione.

È importante notare la differenza con il Table Control: mentre la multi-listbox si focalizza sulla selezione e sulla visualizzazione dei dati, il Table Control gestisce i dati come un array bidimensionale di stringhe e offre funzionalità di editing che ne fanno uno strumento diverso.

Vantaggi delle Multi-Listbox

L’utilizzo delle multi-listbox in LabVIEW offre numerosi vantaggi, tra cui:

- **Visualizzazione Organizzata dei Dati:**
  La capacità di suddividere i dati in più colonne permette di presentare informazioni complesse in modo ordinato e leggibile. Questo facilita l’analisi e la comparazione dei dati.
- **Personalizzazione Grafica:**
  Grazie alle proprietà dei controlli, è possibile personalizzare aspetto, colori, font e perfino integrare icone migliorate. Ciò consente di creare interfacce visivamente accattivanti e coerenti con il resto dell’applicazione.
- **Gestione Avanzata della Selezione:**
  Le multi-listbox permettono di selezionare intere righe, semplificando la logica di validazione e la sincronizzazione con altri controlli o dati presenti nella GUI.
- **Ridimensionamento Responsive:**
  Con una programmazione adeguata, è possibile far sì che le multi-listbox si adattino automaticamente al ridimensionamento del pannello. Questo permette di mantenere la distribuzione proporzionale delle colonne anche su schermi di diverse dimensioni.

La multicolumn listbox è quindi ampiamente utilizzata nella maggior parte dei software di test, dove è necessario andare a gestire analizzare e visualizzare i dati in maniera semplice e chiara.

Svantaggi e Limitazioni

Nonostante i vantaggi, l’uso delle multi-listbox presenta alcune limitazioni:

- **Ridimensionamento Orizzontale:**
  In situazioni in cui la finestra viene ridimensionata, le colonne potrebbero non adattarsi in maniera ottimale. Ad esempio, potrebbe capitare che vengano generate colonne in eccesso o che lo spazio venga distribuito in modo non uniforme, rendendo l’interfaccia meno gradevole.
- **Editing delle Celle Limitato:**
  Le multi-listbox non offrono funzionalità native per l’editing in linea dei dati. Se è necessario modificare il contenuto delle celle, occorrerà

implementare soluzioni personalizzate mediante property node e logica aggiuntiva.

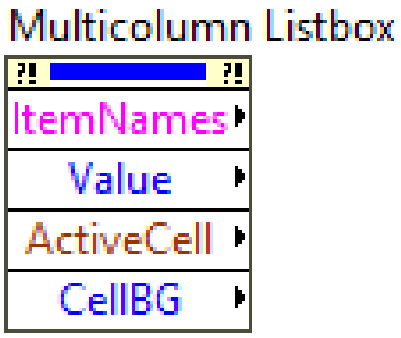
- **Gestione degli Eventi Complessa:**
  La configurazione degli eventi, come il double-click o la selezione di una cella, richiede una gestione accurata. Occorre distinguere tra eventi sui dati e quelli su altri elementi (ad esempio, le barre di scorrimento), il che può complicare il codice.
- **Aggiornamenti Dinamici:**
  La modifica in tempo reale dei dati (ad esempio, l'aggiunta o la rimozione di righe/colonne) può risultare complessa e richiede una programmazione dettagliata per mantenere la coerenza dello stato del controllo.

Si può dire quindi, che da una parte abbiamo tutti i vantaggi di organizzazione dei dati, dall'altra parte però abbiamo una gestione lato sviluppo un po' più complessa di quanto potrebbe apparire

Programmaticità e Gestione degli Eventi

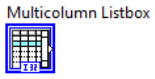
Per sfruttare appieno le potenzialità delle multi-listbox, è fondamentale una gestione programmatica accurata:

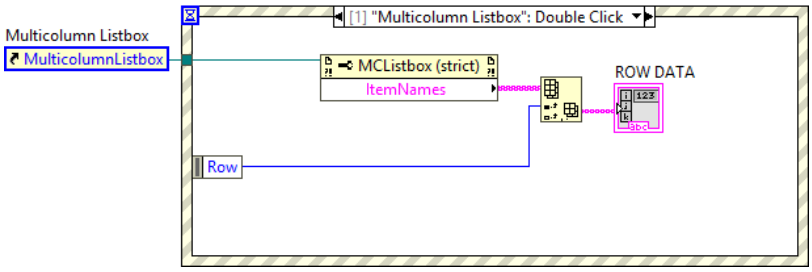
- **Uso dei Property Nodes:**
  Le proprietà delle multi-listbox, come "Item Names", larghezza delle colonne, colori di sfondo e di evidenziazione, possono essere modificate dinamicamente tramite property nodes. Questo permette di aggiornare il controllo in base alle esigenze dell'applicazione, ad esempio al ridimensionamento della finestra.



Property node implicitamente collegato alla multilistbox del programma

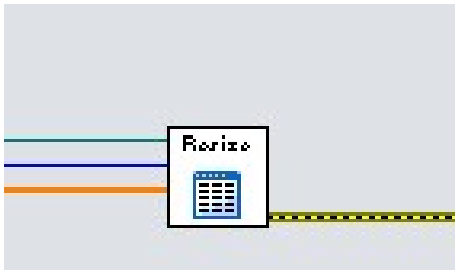
- **Utilizzo della Event Structure:**
  Integrare le multi-listbox in una struttura eventi consente di intercettare azioni dell'utente (come il double-click o la selezione di una riga) e di eseguire logiche specifiche. Un esempio pratico è la selezione automatica della prima riga al caricamento del VI.





Struttura a eventi, utile per triggerare eventi sulla Multilistbox, per esempio svuotare la tabella con un pulsante. (magari utilizzando la proprietà Item Names

- **Creazione di SubVIs Modulari:**
  Per gestire funzionalità complesse come il ridimensionamento responsive o l'aggiornamento dinamico dei dati, è consigliabile sviluppare subVIs dedicati. Questi moduli rendono il codice più chiaro e facilitano la manutenzione e il riutilizzo.



Farò il resize diverse volte, conviene creare un SubVI

- **Sincronizzazione e Coerenza dei Dati:**
  Quando si aggiornano dinamicamente i dati visualizzati, è importante garantire che l'array di stringhe associato alla proprietà "Item Names" venga modificato in modo coerente, mantenendo lo stato di selezione e la sincronizzazione con altri controlli della GUI.

Strategie per una GUI Responsive

Per creare un'interfaccia utente che si adatti a diverse risoluzioni e garantisca una buona user experience, è possibile adottare le seguenti strategie:

- **Layout Dinamici:**
  Integrare le multi-listbox in layout che si adattano automaticamente (ad esempio, utilizzando splitter, pannelli flessibili e griglie dinamiche) permette di mantenere un aspetto coerente su schermi di dimensioni diverse.

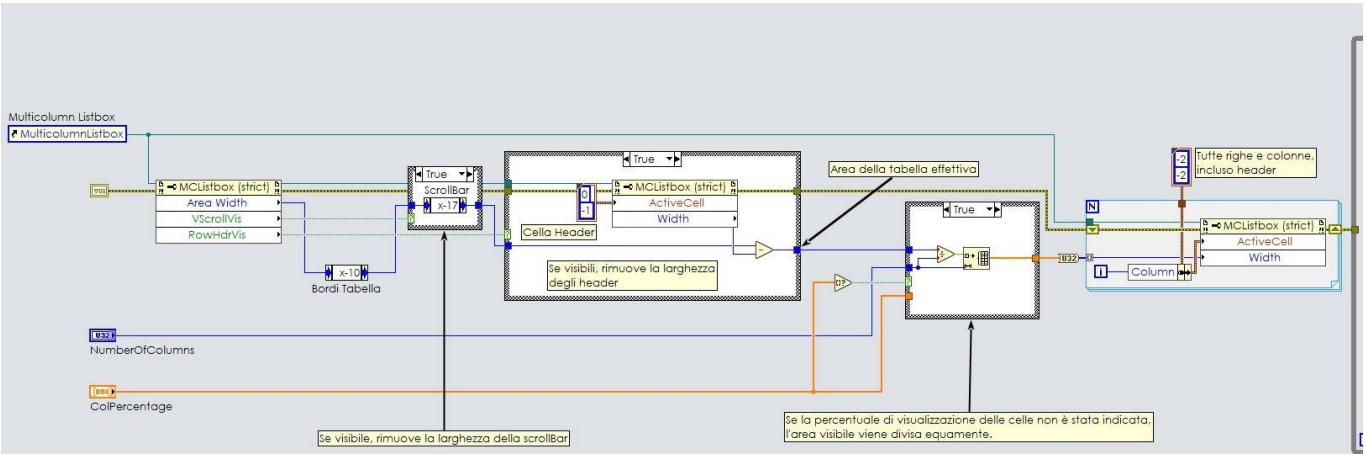
- **Gestione degli Eventi di Ridimensionamento:**
  Catturare l'evento "Panel Resize" consente di ricalcolare la distribuzione delle colonne in tempo reale, in modo da evitare che il controllo appaia troppo compattato o troppo disperso.
- **Feedback Visivo Immediato:**
  È fondamentale che le azioni dell'utente (come la selezione di una riga) abbiano un riscontro immediato, per esempio cambiando il colore o evidenziando la riga selezionata, in modo da migliorare l'interazione.
- **Test su Diverse Risoluzioni:**
  Verifica il comportamento della GUI su schermi con risoluzioni differenti per garantire che l'interfaccia rimanga leggibile e funzionale in ogni contesto.

Il ridimensionamento della Multilistbox

Una cosa molto carina che abbiamo accennato prima, che è possibile applicare sulle Multilistbox, è un algoritmo di ridimensionamento dinamico.

Sarebbe molto utile utilizzare un singolo modulo, magari chiamato banalmente "Resize", che attraverso un evento di pane size (ridimensionamento di un Pane), vada a mantenermi un numero di colonne fisso grandi un valore % riferito alla massima dimensione della tabella visibile (escludendo quindi bordi e scrollbar).

Iniziamo quindi a sviluppare il codice;

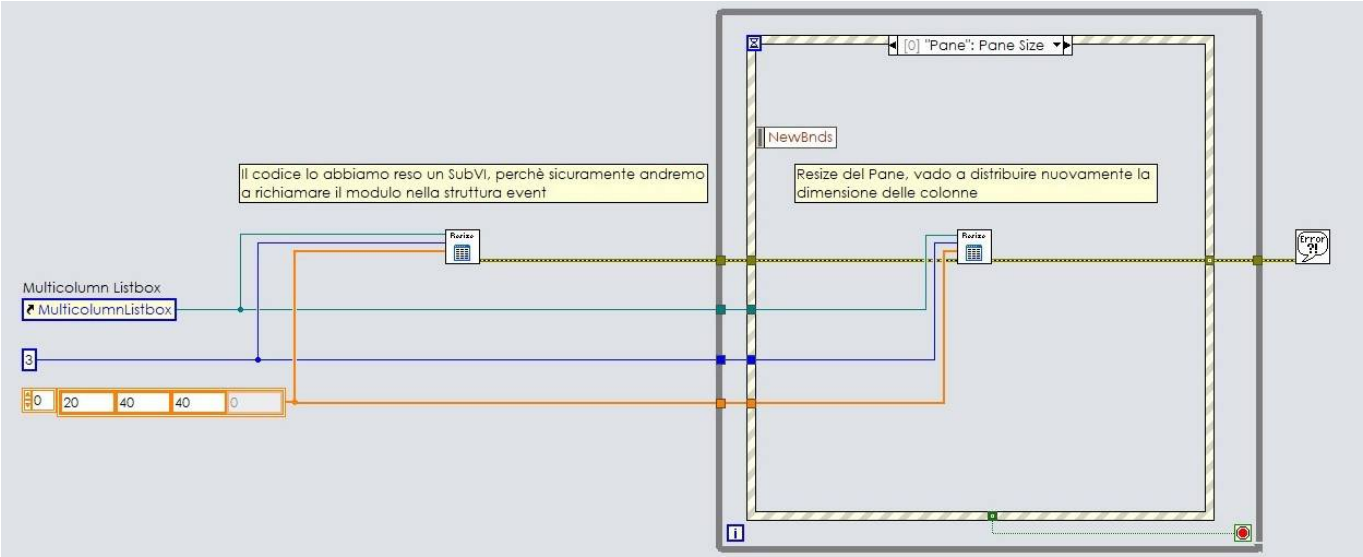


Il codice sopra, mostra come si dovrebbe gestire il ridimensionamento, nello specifico fa questo:

- Property node ci restituisce l'area della tabella, estraendo anche informazioni come la presenza o meno del header o della scrollbar verticale, se visibili, andrò a tenere in considerazione la loro dimensione per fare i calcoli successivamente.
- Calcolo l'area effettiva della tabella, e la divido per il numero di colonne che desidero.
- Se mi viene specificato una dimensione in percentuale per la divisione delle colonne, allora vado a calcolarmi quanto equivale in dimensione

1%, e andrò a moltiplicare il valore per ogni ColumnPercentage definito. Questo mi permette di andare a settare per ogni colonna definita la sua dimensione.

Come anticipato prima, andremo a rendere il codice un SubVI, cosi da poterlo integrare facilmente in più parti del programma, oltre che nel rendere il software più pulito e modularizzato (come dovrebbe essere ogni software).

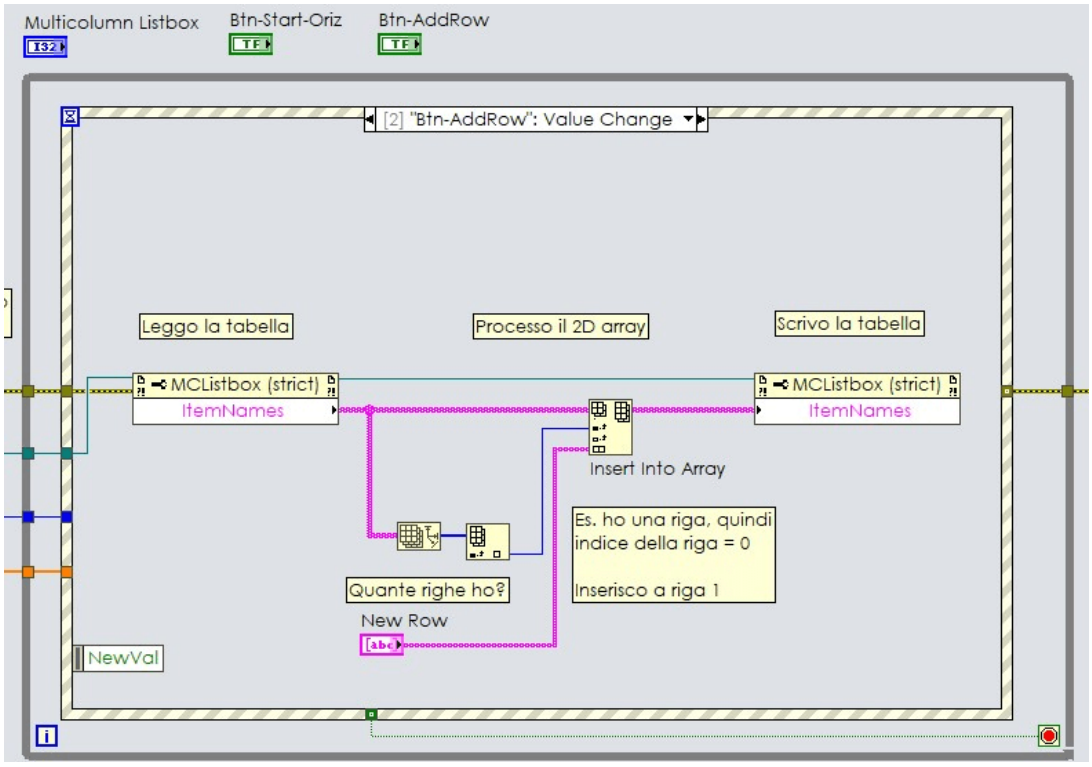


Abbiamo reso il Resize un SubVI del nostro programma, possiamo utilizzarlo senza preoccuparci di creare il codice nuovamente

Gestione del valore e dei dati utilizzando ItemNames

Come abbiamo accennato prima, la gestione della Multilistbox (per quanto riguarda la gestione dei suoi elementi), avviene tramite l'utilizzo dei property node.

La proprietà principale che troviamo e ItemNames, questa property ci permette di leggere o scrivere il valore della tabella come un Array 2D di stringhe.



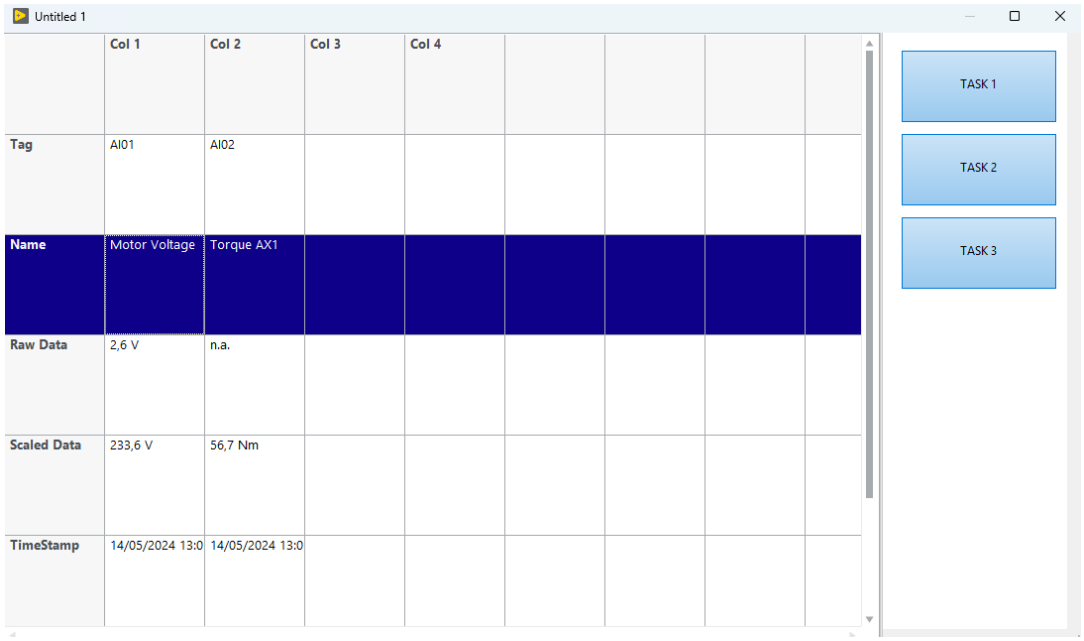
La tabella è da considerare come un array 2D

Nell’esempio, abbiamo creato un piccolo algoritmo che inserisce una riga sempre in fondo alla tabella.

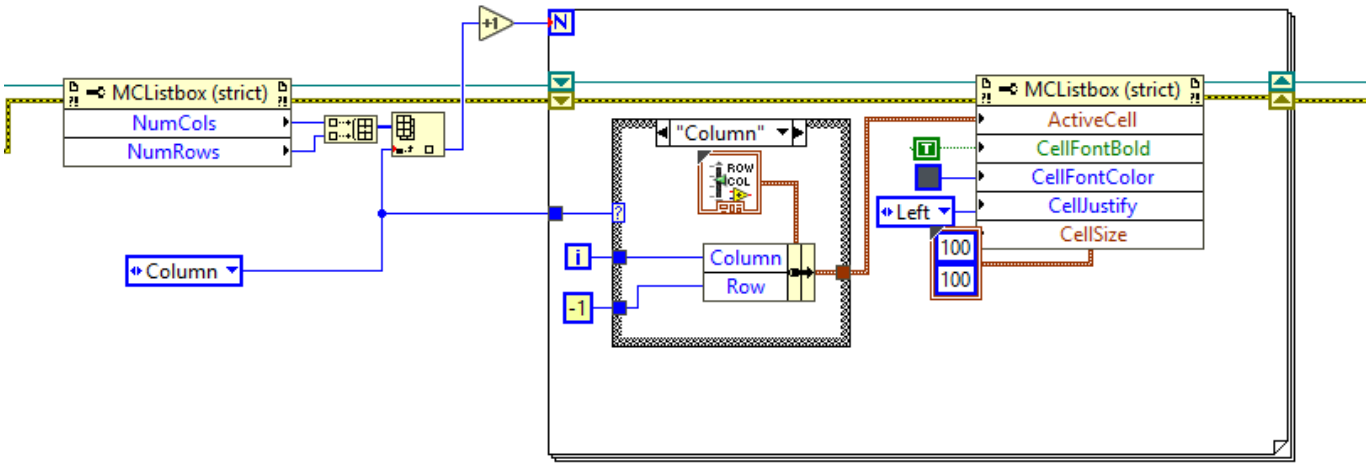
Come potete immaginare, diventa in realtà semplice gestire la tabella in questo modo attraverso magari degli editor customizzati.

Una tipica applicazione nella quale è presente una Multilistbox, ha solitamente anche diversi editor per aggiungere/rimpiazzare/eliminare righe in maniera dinamica, facendo riferimento al valore della tabella che di fatto è un numero intero con segno (rappresentante la riga selezionata)

Intestazioni di Riga e di Colonna come formattarle programmaticamente?



Partiamo da come formattare le celle di intestazione per righe e colonne, si devono selezionare come active cell sapendo che la loro posizione di riga(per le column) o colonna(per le row) è -1.



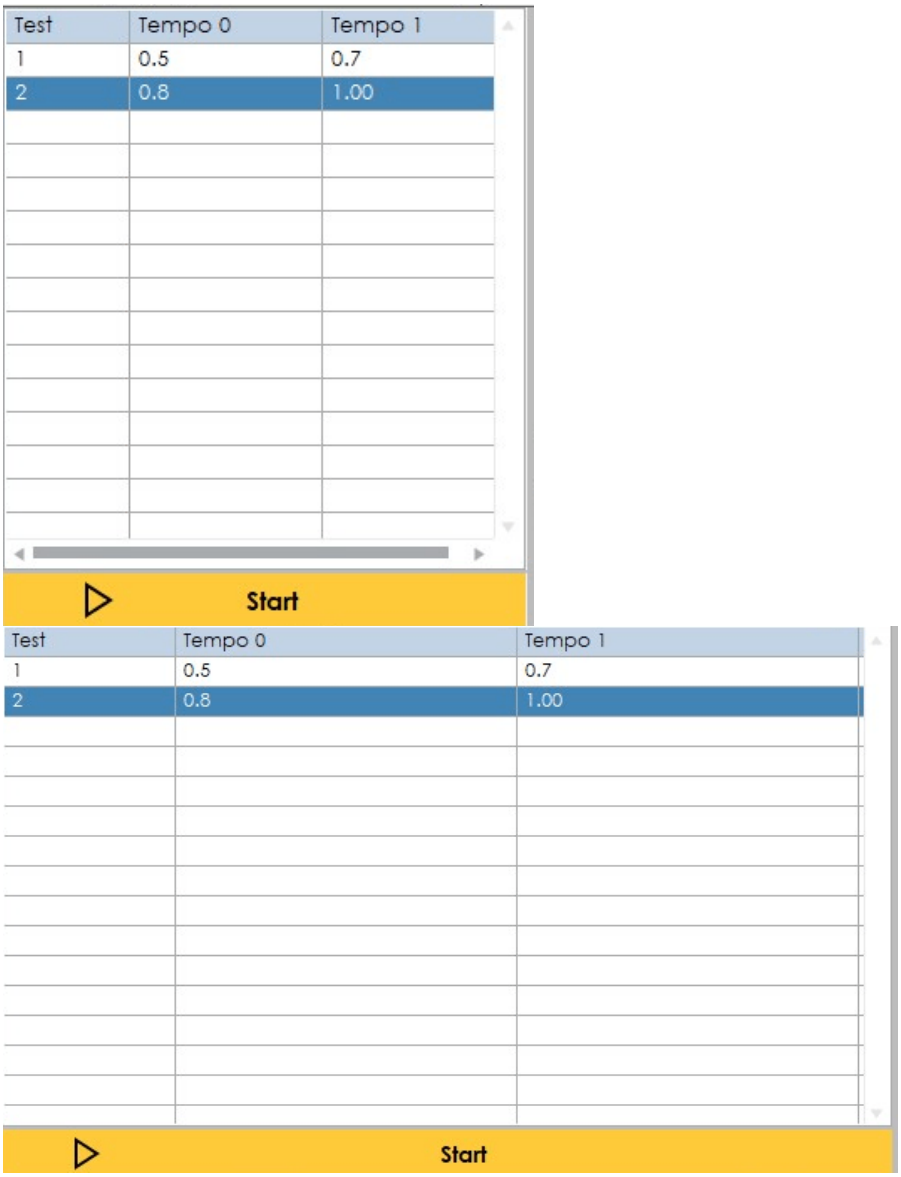
Abbiamo realizzato questo blocco, pensando di poter selezionare se modificare Righe o Colonne.

Ovviamente si possono raccogliere i parametri da assegnare in un cluster da passare come inp

Conclusioni

L’utilizzo delle multi-listbox in LabVIEW è una soluzione potente per la realizzazione di interfacce utente moderne e responsive. Esse offrono vantaggi significativi in termini di organizzazione dei dati e personalizzazione grafica, pur presentando alcune limitazioni che richiedono una gestione programmatica accurata. L’adozione di best practices – come

l’impiego dei property nodes, l’utilizzo di Event Structure e la creazione di subVIs modulari – permette di superare queste limitazioni, garantendo una user experience di alto livello.



Esempio: GUI che mostra come le colonne della multi-listbox si ridimensionano proporzionalmente al variare delle dimensioni della finestra.

Integrare correttamente questi controlli nella propria applicazione LabVIEW significa anche documentare accuratamente il codice e testarlo su differenti scenari, affinché l’interfaccia risulti sempre intuitiva, reattiva e in linea con le aspettative dell’utente finale.

In conclusione, la gestione della Multi-Listbox può essere complessa a causa delle sue molteplici proprietà da gestire per i diversi scenari possibili di utilizzo, però le sue caratteristiche la rendono un oggetto molto utilizzato nell’ambito test ed estremamente richiesto per la gestione di ricette e steps.