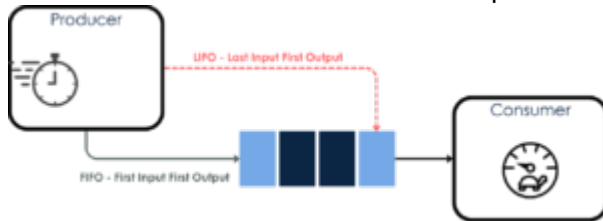


Design Pattern #6- Producer/Consumer

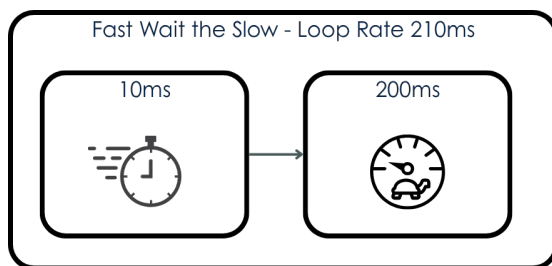
Category: LabVIEW Talk

scritto da Davide Oratelli | Marzo 24, 2020

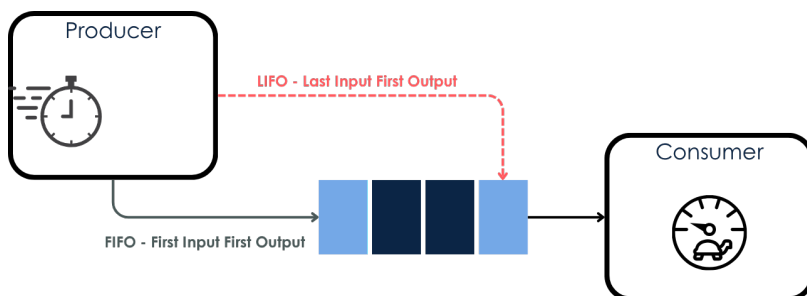


Nello scorso [articolo](#) abbiamo visto la QSM, oggi vedremo la Producer/Consumer Design Pattern

Il Producer/Consumer Design Pattern nasce dall'esigenza di **disaccoppiare** la **produzione** dalla **consumazione di dati**, garantendo massima efficienza e scalabilità anche in applicazioni real-time complesse come l'acquisizione dati industriale, il controllo di processo o la gestione di interfacce utente responsive.



Questo disaccoppiamento utilizzando come mezzo di comunicazione una FIFO, permette di risolvere il problema di dover gestire lo scambio dei dati tra un processo produttore veloce, a un processo consumatore lento.



- Il **producer** genera dati e li inserisce nel buffer
- Il **consumer** preleva i dati dal buffer e li elabora
- Il **buffer** è la memoria di accumulo per lo scambio

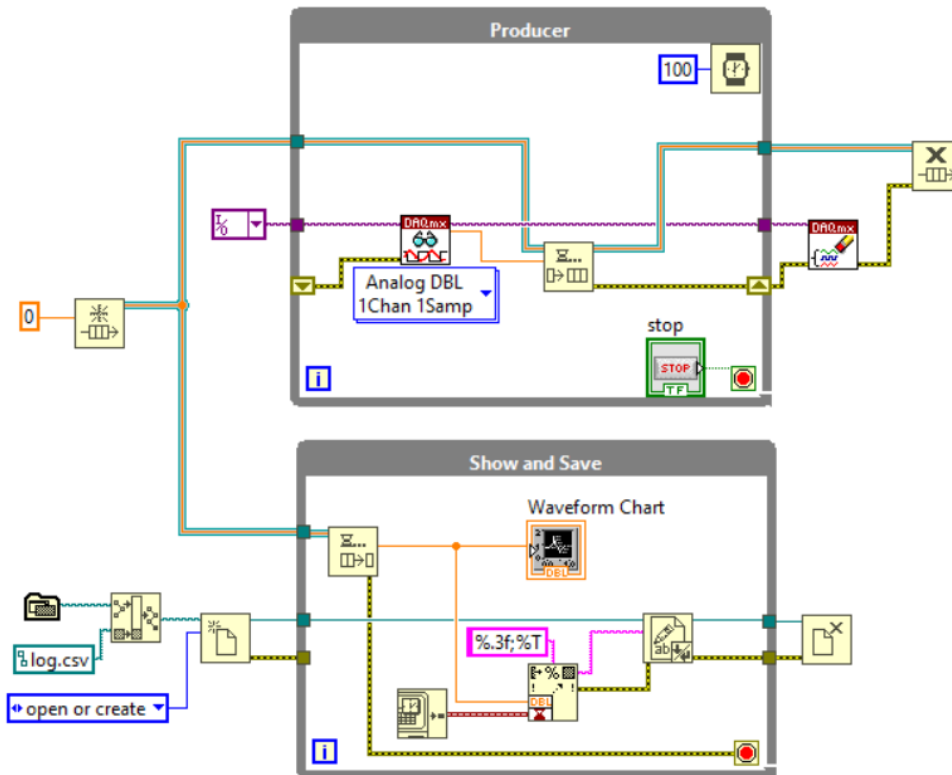
Quindi il design pattern separa due processi rendendo autonomi e favoranedo lo scambio tra processi a differenti velocità senza perdita di dati.

Per compensare la differenza di velocità, i dati vengono memorizzati in un Buffer [FIFO \(Firts In Firts Out\)](#), che regola il flusso di dati.

Quando esistono più processi in esecuzione a velocità diverse, la comunicazione bufferizzata tra i processi è estremamente efficace.

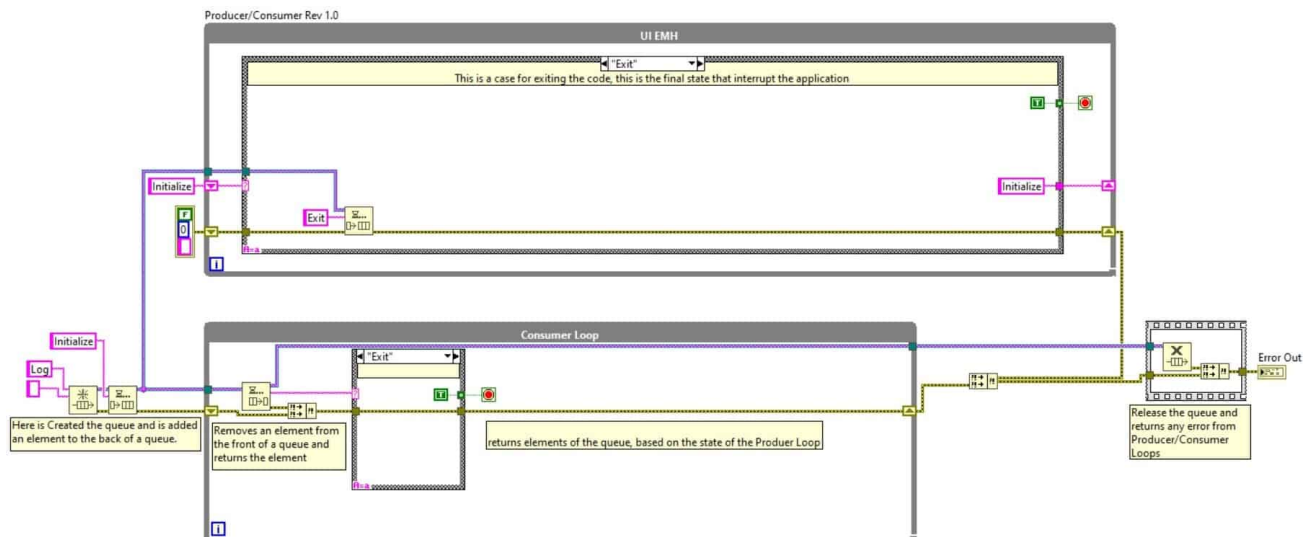
Esempio Acquisizione e Salvataggio

Un' applicazione è composta da due processi, il primo **acquisisce dati** per il secondo a una velocità doppia, verrà utilizzata l'architettura Producer/Consumer.



Nota: Il ciclo consumatore si ferma per **Errore** (perde la reference della coda), questo è possibile perchè il produttore si ferma con pulsante di stop, poi chiude e **rilascia la reference**. Il **dequeue** va in errore perchè perde il riferimento al file.

Se pensiamo alla state machine viste negli articoli precedenti, possiamo gestire gli stati con messaggi inviati sul consumatore, che ha una struttura case per selezionare il caso richiesto dal messaggio.



Il del modello produttore / consumatore a questa applicazione consiste nel produrre i messaggi per la macchina a stati consumatore disaccoppiando l'esecuzione della macchina a stati in due processor di elaborazione.

Per il caso Exit, verrà accodato il messaggio di uscita nel loop producer, e il consumer eseguirà poi l'evento associato ad esso.

La tecnica del producer/consumer è utilizzata anche per disaccoppiare la gestione degli eventi UI, verso una macchina a stati consumatore che elabora i messaggi, rendendo responsiva la GUI.

Il pattern producer/consumer è alla base del framework QMH che utilizza macchine a stati con un enumeratore stringa per il messaggio e un variant o stringa per i parametri.