POLITECNICO DI TORINO

Corso di Laurea Magistrale
in Ingegneria Meccanica

Tesi di Laurea

Integrazione di un sistema di acquisizione ad alta frequenza di un banco prova per servocomandi di volo elettromeccanici

Relatori
prof. Massimo Sorli
prof. Andrea De Martin
ing. Matteo Gaidano

Candidato
Francesco Cicchinè

Anno Accademico 2023-2024

† *Alla mia famiglia*

Sommario

L'elaborato propone l'integrazione di un sistema di acquisizione, attraverso l'utilizzo del software LabVIEW, di un banco prova dedicato al test di servocomandi di volo elettromeccanici. L'obiettivo principale consiste nella cattura di segnali ad alta frequenza di tensione e corrente provenienti da un motore elettromeccanico. Infatti, grazie a quest'ultimi, sarà possibile approfondire lo studio di prognostica che si sta facendo da tempo su questi dispositivi, così da prevederne guasti e manutenzioni in maniera più efficiente e,in tal modo, nel prossimo futuro, ampliare il loro utilizzo nel settore aeronautico. Il lavoro include anche una revisione completa del codice, con particolare attenzione all'ottimizzazione dei metodi di salvataggio dati, un miglioramento della comunicazione tra le diverse VI e il perfezionamento della visualizzazione dei segnali. La tesi contribuisce significativamente al miglioramento della precisione e dell'affidabilità del sistema, fornendo un ambiente più efficiente per lo studio del comportamento dei servocomandi di volo elettromeccanici in qualsiasi condizione di lavoro.

Indice

Elenco delle tabelle

Elenco delle figure

Guida alla comprensione dell'elaborato

Per facilitare la trattazione degli argomenti, sono stati utilizzati termini che potrebbero creare malintesi, nel caso in cui non venissero chiariti. Tra cui:

- *Segnali Fisici* e *Segnali Elettrici*. Le diciture Segnali Fisici e Segnali Elettrici. Con i primi si intendono tutti quei segnali che provengono da sensori in grado di misurare delle grandezze meccaniche, come LVT per la velocità, l'AE per i gradi, la cella di carico per la misura di forza e i sensori di pressione. Con i secondi invece si intendono esclusivamente i segnali di corrente e tensione.

  Questa distinzione è stata fatta in quanto queste due tipologie di segnali sono state trattate in modo diverso a livello di progettazione del software.

- *Main FPGA e Main RT*. Molte volte per abbreviare la dicitura "il codice Main FPGA" viene utilizzata semplicemente "FPGA" e allo stesso modo "il codice Main RT" viene semplificato con "RT o Real-Time".

- *While Loop*. Allo stesso modo la sigla "While Loop" verrà abbreviata semplicemente con "loop" o "ciclo".

- *Buffer*. Con questo termine ci si riferisce a una zona di memoria temporanea utilizzata per immagazzinare dati in attesa di elaborazione o trasferimento.

- *Velocità di campionamento*. Molte volte questo termine viene semplicemente abbreviato con "velocità" (quando non specificato).

Inoltre, si presenta la lista degli acronimi che verranno utilizzati all'interno dell'elaborato:

- ACC - *Acquisition and Control Cabinet (Armadio di Acquisizione e controllo)*
- AE - *Angular Encoder (Encoder angolare)*
- AEA - *All Electric Aircraft*
- AI - *Analog Input*
- AO - *Analog Output*
- AUT - *Actuator Under Test (Attuatore sotto test)*
- CNC - *Controllo Numerico Computerizzato*
- DEMA - *Driver dell'EMA*
- DI - *Digital Input*
- DO - *Digital Output*

- ECU - *Electronic Control Unit*
- EHA - *Electro Hydraulic Actuator (Attuatore elettroidraulico)*
- EMA - *Electro Mechanical Actuator (Attuatore elettromeccanico)*
- FEMCA *Failure Mode, Effects, and Criticality Analysis*
- FPGA - *Field-Programmable Gate Array*
- HLA - *Attuatore idraulico lineare di banco (Hydraulic Load Actuator)*
- HSA - *Hydraulic Servo Actuator (Attuatore servo idraulico)*
- HW - *Hardware*
- I/O - *Input/Output*
- LC - *Load Cell (Cella di carico)*
- LVT - *Linear Velocity Transducer*
- NI - *National Instruments*
- PBW - *Powered By Wire*
- PWM - *Pulse Width Modulation*
- RT - *Real Time*
- SOV - *Valvola di Shut-Off*
- SV - *ServoValve (Servovalvola)*
- SW - *Software*
- TB - *Test-Bench (Banco prova servocomandi)*
- UI - *User Interface*
- US - *User Station*
- VI - *Virtual Instruments*

Capitolo 1

Panoramica nel mondo degli attuatori di volo

1.1 Stato dell'arte

Nel vasto panorama dell'ingegneria aeronautica, gli attuatori di volo rivestono un ruolo cruciale nell'assicurare il controllo e la manovrabilità degli aeromobili. Questi dispositivi, essenziali per il funzionamento dei velivoli, svolgono una serie di compiti fondamentali: dalla regolazione delle superfici di controllo, alla stabilizzazione durante il volo, fino alla movimentazione dei carrelli di atterraggio.

Possono essere suddivisi in tre macro categorie: gli idraulici (e elettroidraulici), gli elettromeccanici e i pneumatici. I primi due sono i più utilizzati nell'aviazione (commerciale e militare), in quanto considerati affidabili, compatti e potenti, mentre gli ultimi vengono soprattutto sfruttati per piccoli aeromobili e nel settore industriale (quest'ultimi non verranno analizzati all'interno di questo elaborato).

Gli attuatori elettromeccanici EMA (Electro Mechanical Actuator) (figura 1.1.b) sono dei dispositivi atti a convertire l'energia elettrica in energia meccanica, generando così movimenti rotativi o lineari. Generalmente, si ha che la potenza elettrica in ingresso alimenta una ECU (Electronic Control Unit), che a sua volta, si occupa di pilotare un servo motore a cui sarà poi collegato un opportuno riduttore, in funzione dello scopo a cui sarà destinato. Il loro campo di utilizzo spazia attraverso vari settori, tra cui: robotico, trasporti, packaging, manifatturiero e via dicendo. Per esempio, possiamo trovarne nei sistemi di attuazione per l'apertura e chiusura di dighe, nelle macchine CNC, nell'orientamento dei pannelli solari, nell'inclinazione dei cannoni sparaneve e così via [1].

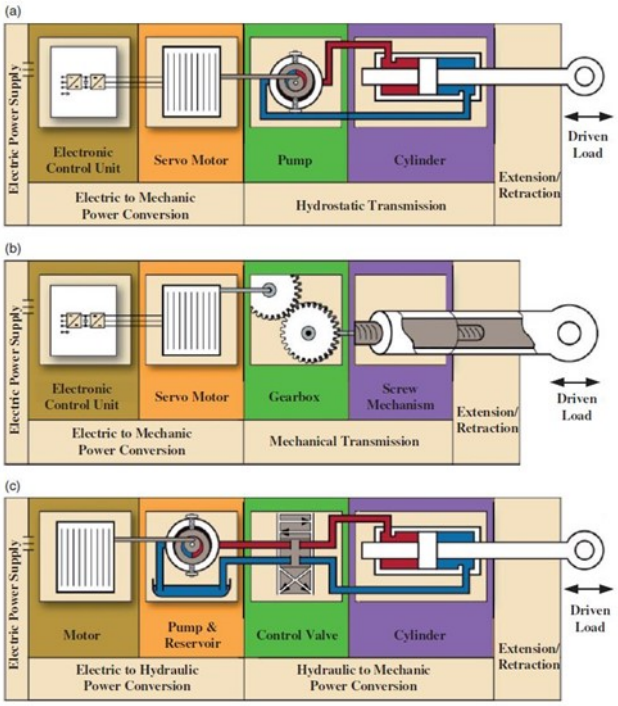


Figura 1.1: Attuatori PBW e HSA: EHA (a), EMA (b) e HSA (c). Fonte: [1].

I primi aeromobili ad adottarli furono i Boeing 777 nel 1990, dove li utilizzarono per azionare i flaps e gli slats, poi seguirono gli Airbus A380 (figura 1.2), usandoli negli slats e negli equilibratori nella coda. Infine negli aerei più recenti, come il Boeing 787, sono stati utilizzati anche per movimentare 4 spoilers su un totale di 14 [2].

Figura 1.2: Airbus A380. Fonte: [3].

Nella figura 1.3, si riporta un esempio di EMA:

Figura 1.3: Electro-Mechanical Actuator (EMA). Fonte: [4].

Per quanto riguarda gli attuatori idraulici, invece, ne esistono svariati tipi. Principalmente se ne possono riconoscere di due configurazioni: gli HSA (Hydraulic Servo Actuator, figura 1.1.c) e gli elettroidraulici EHA (Electro-hydraulic actuators, figura 1.1.a). Entrambi sono dispositivi che consentono la conversione dell'energia elettrica in energia meccanica, servendosi dell'energia idraulica. I primi sono una versione più vecchia, che sfruttano un sistema idraulico per generare e controllare il movimento delle superfici di controllo dell'aeromobile. Sono alimentati da una fonte di pressione idraulica e ancora vengono utilizzati in quanto possono fornire forze notevoli per controllare grandi superfici di volo. I secondi invece, come gli EMA, sono costituiti da una ECU, la quale è collegata ad una pompa reversibile, in grado a sua volta, di pilotare un pistone idraulico. In questi, la potenza idraulica è proporzionale alla velocità del motore, grazie alla quale, modulando la portata di fluido all'interno delle camere, si riesce ad effettuare un controllo in posizione sul pistone [1].

Gli EHA e gli EMA appartengono alla categoria dei dispositivi PBW (Powered By Wire), ovvero dove la potenza viene trasferita attraverso cavi elettrici e non meccanicamente o tramite sistemi idraulici, servendosi di sistemi di controllo come la ECU. Questo approccio nasce intorno gli anni 50-60 in America dagli ingegneri NASA con l'obiettivo di aumentare l'affidabilità, l'efficienza e ridurre la manutenzione degli aeromobili. I dispositivi PBW, rispetto ai suoi concorrenti tradizionali (tipo HSA), come viene riportato nell'articolo "A review of electromechanical actuators for More/All Electric aircraft systems", (Qiao et all, 2018), permettono di di ridurre notevolmente il peso, eliminare eventuali perdite di olio o altri liquidi corrosivi e infiammabili, trasportare meno carburante bruciato (quindi meno inquinamento), meno costi di manutenzione e una notevole riduzione (intorno al 30 %-50 %) dei GSM (Ground Support Equipment, ovvero tutto quell'equipaggiamento che viene usato nelle operazioni di rifornimento, carico/scarico merci e movimentazione dell'aereo a terra, nel lasso di tempo che intercorre tra un volo e l'altro). Anche per il settore militare rappresenta un netto vantaggio, poiché meno massa viene trasportata,

maggiore è l'agilità in combattimento, minore è il tempo di decollo e si riduce la probabilità (di circa il 14 %) che un proiettile possa colpire serbatoi e altre parti sensibili, inibendo così i comandi di volo [5].

Attualmente gli HSA e EHA, vengono impiegati per i comandi di volo primari (quindi alettoni, elevatori e timone di direzione), occupandosi di gestire la traiettoria del mezzo agendo sui tre gradi di libertà (imbardata rollio e beccheggio), e per azionare i carrelli di atterraggio. Gli elettromeccanici, invece, sono utilizzati principalmente per pilotare i comandi secondari (come flap, aerofreni e spoilers) (figura 1.4).

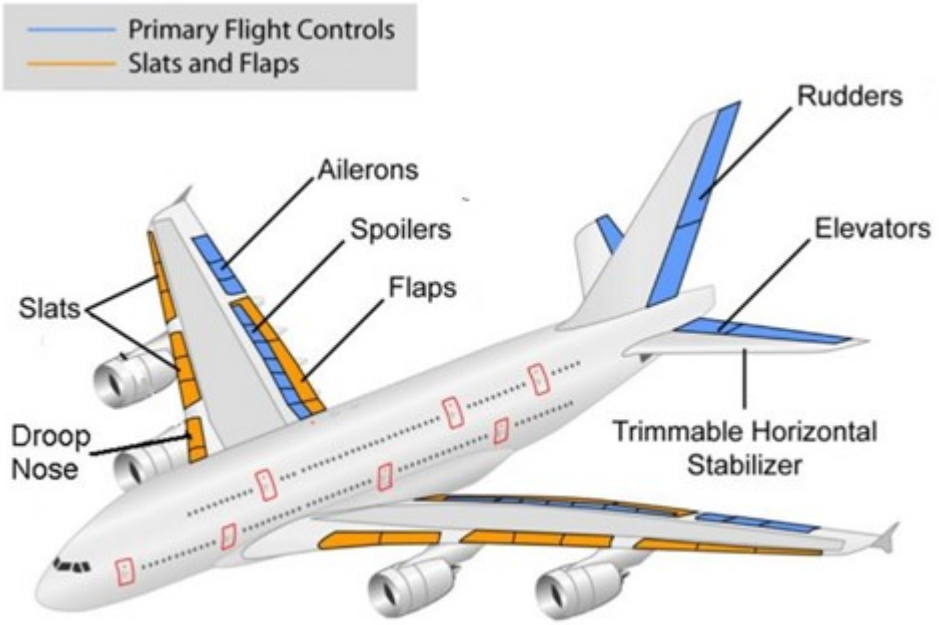


Figura 1.4: Illustrazione comandi primari (blu) e secondari (arancione) di volo. Fonte: [5]

Nonostante gli EHA rappresentino un'evoluzione degli HSA, hanno comunque, seppur in misura minore, un'importante componente idraulica, che comporta pesi, costi e necessità di manutenzione maggiori rispetto agli EMA. Questi ultimi, invece, sono soggetti a problematiche riguardanti l'affidabilità e la sicurezza, poiché possono incorrere in guasti significativi, come il fenomeno del Jamming (che verrà poi approfondito nel paragrafo 1.3), causando una forte diminuzione del loro fattore di sicurezza. Per questo motivo, gli attuatori idraulici, che sono considerati molto più sicuri, sono attualmente impiegati nei comandi primari di volo mentre gli elettromeccanici solamente per i secondari.

A causa di ciò, negli ultimi anni si sta studiando il modo di procedere verso la direzione del Full Electric o anche detto AEA (All Electric Aircraft), ovvero, cercare di portare il mondo dell'aviazione verso l'utilizzo esclusivo degli EMA, andando così ad ottenere vantaggi sia in termini economici, sia ambientali.

Tutti questi interessi, rivolti a questo tipo di attuatore, ha portato la comunità scientifica a adottare una policy PHM, ovvero Prognostic Health Management [6]. L'obiettivo di questa è quello di andare a migliorare l'affidabilità del dispositivo, cercando di prevedere il più possibile il manifestarsi di guasti e manutenzioni straordinarie, attraverso l'utilizzo di sensori ed algoritmi molto complessi. Tale metodo verrà poi spiegato nel paragrafo 1.4.

1.2 Struttura EMA

Generalmente, esistono due tipi di attuatori elettromeccanici: lineari e rotativi . La differenza tra i due, come si può intuire, consiste nel modo in cui viene trasformato il moto all'uscita del dispositivo, ovvero lineare o rotativo. Il funzionamento è approssimativamente lo stesso. In questa tesi verranno considerati solamente attuatori del primo tipo.

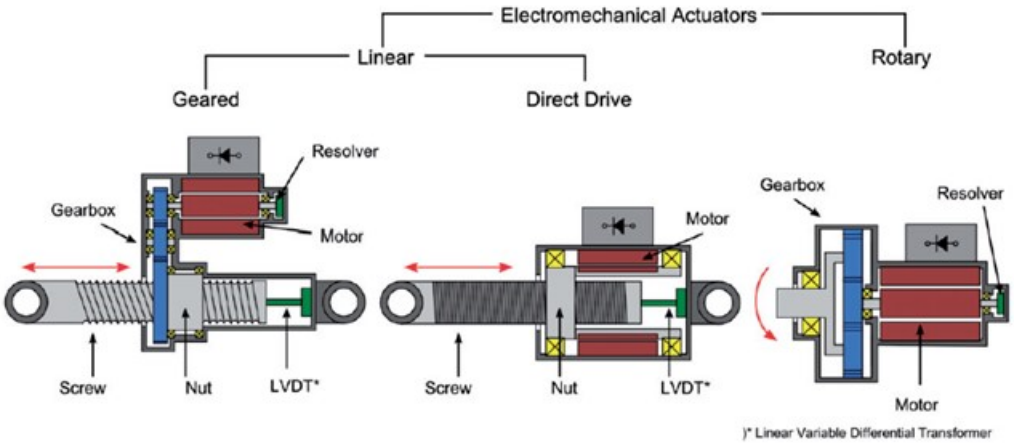


Figura 1.5: Differenti tipologie di EMA. Fonte: [1].

Gli elementi che lo compongono sono:

- Motore: lavorando nel settore aeronautico deve sicuramente essere un dispositivo affidabile, che dissipa bene il calore e con una elevata potenza. Vengono utilizzati diversi tipi, tra cui: Motori sincroni a magneti permanenti (PMSM), motori Brushless (BLDC) e motori elettrici a riluttanza magnetica (SRM).

- Riduttore: volendo ottenere una grande coppia in uscita, solitamente si utilizzano dei riduttori cicloidali, planetari o armonici, i quali offrono compattezza, gioco ridotto e un'efficienza discretamente elevata [1].

- Meccanismo a vite: usato per convertire il moto da rotatorio a lineare.

- ECU: è il sistema di controllo dell'attuatore. I sensori inviano dei feedback a questa unità la quale poi si occupa, in funzione di questi ultimi, di inviare al motore gli opportuni segnali in tensione e migliorarne così il controllo. Di questi, ne esistono svariati tipi, tra cui:
  - Sensori di corrente: utili per misurare la corrente di ogni fase del motore, così da chiudere l'anello di controllo in corrente;
  - Sensori di posizione nell'albero motore: per conoscere il momento esatto in cui commutare le varie fasi del motore (es. sensori ad effetto Hall o encoder);
  - Sensori di velocità: per conoscere la posizione dell'attuatore (es. LVT - Linear Varible Velocity - che sono sensori di velocità);
  - Sensori di temperatura: essendo un parametro critico, viene tenuto sotto controllo anche esso.

1.3 Guasti

Molti sono i tipi di guasto che possono verificarsi all'interno di un EMA e compromettere il suo funzionamento. I principali sono:

- Il Jamming: Si tratta del fenomeno di grippaggio dell'attuatore. È uno dei più frequenti e consiste nel bloccaggio delle superfici in contatto, causando così l'impossibilità di controllare lo strumento. Dato che il motore è composto da diverse parti in movimento, il contatto metallo-metallo lo porta ad usurarsi maggiormente, innescando così il blocco. La probabilità che si verifichi è di 1 ogni 40 milioni di ore di volo [7]. Tuttavia, a causa degli standard molto severi, questa frequenza non gli permette di posizionarsi allo stesso livello del concorrente idraulico, il quale presenta una probabilità di inceppamento di 1 ogni miliardo di ore di volo;
- Il surriscaldamento dell'impianto elettrico e quello dovuto al contatto metallo-metallo tra le superfici.

Mentre, i malfunzionamenti meno frequenti ma non per questo meno importanti sono:

- ECU (electronic power unit) failure: l'ECU, ma in generale il sistema di controllo, può essere stressato dalle numerose vibrazioni, temperature elevate, polveri metalliche, cariche elettrostatiche e quindi andare verso la rottura o il malfunzionamento;
- Malfunzionamenti al motore elettrico:
    - Winding turn-to-turn short e Winding phase-to-phase short: ovvero si verifica quando due avvolgimenti della stessa fase, a causa della degradazione del materiale isolante, entrano in contatto e creano corto-circuito;
    - Winding phase-to-ground short: accade quando gli avvolgimenti entrano in contatto con il case esterno e creano un corto circuito con il ground;
    - Perdita delle proprietà magnetiche dei magneti nei motori;
    - Rottura dei cuscinetti: soprattutto nel motore elettrico, a causa dall'usura e dalla lubrificazione insufficiente;
- Pitting e corrosione in generale: dovuto all'ossidazione di componenti;
- Cricche, usura, scheggiatura, mancata lubrificazione, deformazioni di superfici: sono tra i principali motivi di rottura degli ingranaggi e della vite senza fine;
- Scariche elettriche tra i vari componenti: è un fenomeno che provoca la formazione di archi elettrici tra i vari elementi e causa un'usura sempre maggiore;
- Cristallizzazione, emulsione e decomposizione del lubrificante e perdita delle proprietà meccaniche dei grassi [6].

1.4 PHM (Prognostics and Health Management) e HPM (Hierarchical Process Model)

Sia il PHM che il HPM sono dei metodi e modelli che servono a studiare, prevenire ed agire, attraverso elementi di diagnostica, contro tutte quelle problematiche che possono portare l'EMA alla rottura. Infatti, attraverso l'utilizzo di sensori e specifici algoritmi, è possibile prevedere (in funzione del tempo di utilizzo, carico e altre condizioni esterne) quando si verificheranno dei guasti. In particolar modo il protocollo PHM si occupa di:

- Migliorare costantemente la robustezza e affidabilità del dispositivo;
- Ridurre il numero di interventi di manutenzione straordinaria;
- Facilitare le operazioni di manutenzione ordinarie;
- Migliorare la supply chain.

Fondamentale è comprendere il numero e il tipo di sensori che si desidera utilizzare, poiché un basso numero non consente una raccolta sufficiente di informazioni, mentre un numero elevato porta a creare un sistema troppo complesso da gestire e manipolare.

Quando si sviluppa un PHM, è molto importante stabilire quali siano i criteri di valutazione e le priorità che hanno i vari tipi di guasto. Per questo molte volte si fa riferimento ad un approccio di tipo FMECA (Failure Mode, Effects, and Criticality Analysis). Quest'ultimo è un metodo di analisi che permette di determinare l'impatto che un potenziale guasto avrebbe su un prodotto o su un processo e valutarne i rischi. Il metodo FMECA è considerato un approccio dal basso verso l'alto. L'analisi inizia con il mettere insieme i dati disponibili. Ogni componente viene esaminato attentamente per trovare tutte le possibili cause di guasto e, per ognuna, vengono evidenziati i rispettivi effetti che potrebbero avere sul prodotto. Infine, a queste vengono associate una probabilità e un tasso di guasto.

Vengono assegnati dei punteggi al malfunzionamento in funzione di:

- Defect rate: ovvero la probabilità che questo difetto ricapiti.
- Severity: quanto il guasto è importante e a quali situazioni potrebbe portare.
- Replaceability: cioè, quanto risulta facile e veloce sostituire il componente rotto.
- Testability: capacità di studiare e tenere sotto controllo l'oggetto, utilizzando sensoristica e quant'altro.

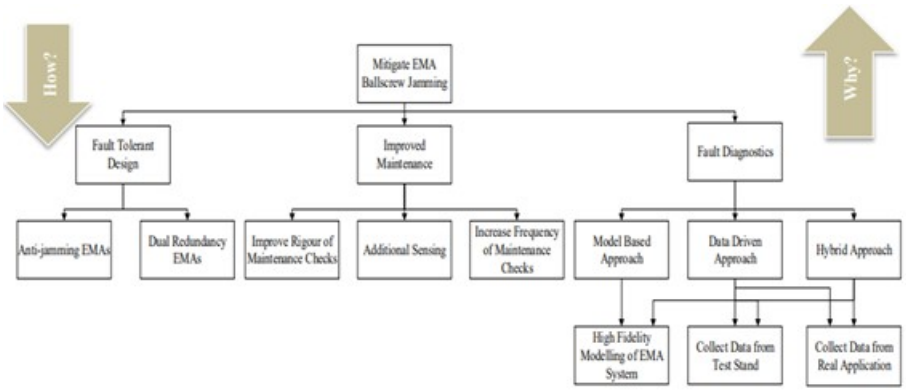


Figura 1.6: HPM. Fonte: [2] p.3.

HPM invece, è un framework sviluppato per limitare il fenomeno del jamming. Sostanzialmente è un modello che permette di affrontare problemi complessi, sviscerandoli sempre di più e riducendo così il grado di difficoltà. È studiato per permettere, a chi lo usa, di agire correttamente in funzione di determinati eventi e ridurre così il rischio di incertezza. Questo metodo agisce a livello di:

- Progettazione: come si può osservare nella figura 1.6, al di sotto di "Fault Tolerant Design", si propongono delle soluzioni per prevenire a monte il fenomeno del Jamming, come per esempio, avere ridondanza di EMA a bordo così da utilizzarne un altro in caso di guasto o utilizzare degli Anti-jamming EMAs che sono degli attuatori migliorati proprio per evitare questo tipo di problema;
- Manutenzione;
- Diagnostica.

Ulteriori delucidazioni riguardo l'approccio PHM possono essere ritrovate all'interno del documento: "SAE: AIR8012 - Prognostics and Health Management Guidelines for Electro-Mechanical Actuators" (De Martin e Jacazio), mentre, riguardo il metodo HPM e lo studio del fenomeno del jamming, si possono trovare approfondimenti nell'articolo:" A Review of Techniques to Mitigate Jamming in Electromechanical Actuators for Safety Critical Applications" di Yameen M. Hussain, Stephen Burrow, Leigh Henson e Patrick Keogh nel "International Journal of Prognostics and Health Management, ISSN 2153-2648, 2018 012".

Quindi, in conclusione, seguendo questi approcci, è possibile sviluppare algoritmi e metodi che possano in qualche modo prevenire, studiare e migliorare gli attuatori elettromeccanici. Come strumento di supporto per fare test e raccogliere dati, è stato costruito il **Banco in controllo forza per servocomandi di volo presente all'interno del Dipartimento di Ingegneria Meccanica e Aerospaziale del Politecnico di Torino**, con lo scopo di collaudare gli attuatori e ricavare elementi utili da analizzare. Nei prossimi capitoli, verrà approfondita la sua struttura, il suo funzionamento, il nuovo software e i risultati ottenuti.

Capitolo 2

Banco prove per servocomandi di volo

2.1 Introduzione al banco prove e al lavoro di tesi

Il banco in controllo forza per servocomandi di volo si presenta come un macchinario in grado di simulare dei carichi aerodinamici su un attuatore in prova, con l'obiettivo di studiarne il comportamento e analizzarne i dati ricavati. Quest'ultimi saranno sfruttati per creare degli algoritmi di prognostica sempre più efficienti, in grado di prevedere in anticipo eventuali guasti o manutenzioni straordinarie. Grazie al banco è possibile testare attuatori di qualsiasi tipo (EMA, EHA ecc...) degradati e non, andando a ricreare condizioni nominali di volo o persino i peggiori scenari di funzionamento. La macchina nasce grazie al programma Clean2sky-ASTIB (Development of Advanced Systems Technologies and hardware/software for the flight simulator and Iron Bird demonstrators for regional aircraft) frutto di un partenariato pubblico-privato tra la Commissione europea e l'industria aeronautica europea. Quest'ultimo è un progetto di ricerca e sviluppo, che mira a supportare il miglioramento tecnologico per una serie di importanti equipaggiamenti considerati di fondamentale importanza per il futuro dell'aviazione civile [8].

L'obiettivo principale di questa tesi nasce dalla necessità di andare ad ottimizzare il software della macchina, per cercare di ottenere informazioni che fin'ora non si erano riuscite a salvare, ovvero i segnali di tensione e corrente provenienti dal motore elettrico. Infatti, nell'ultimo aggiornamento Hardware della macchina, è stato aggiunto un rack contenente dei sensori in grado di leggere quei dati. La prima persona che dopo l'aggiornamento, si è dedicata alla scrittura del codice è stata Romanini Riccardo, nel suo lavoro di tesi: "Implementazione Hardware e Software di nuove funzionalità per il Banco Prova Servocomandi Volo". Successivamente si è notato che l'ottenimento di queste informazioni poteva essere migliorato e da ciò è derivato il presente elaborato:"Integrazione di un sistema di acquisizione ad alta frequenza di un banco prova per servocomandi di volo elettromeccanici". Qui si è cercato di ottimizzare il lavoro precedente, acquisendo i dati ad una frequenza maggiore, costruendo dei sistemi di controllo in grado di garantire

il corretto funzionamento della Compact-Rio ed infine creando un metodo più efficiente per salvarli.

2.2 Descrizione del banco

Il banco prove (figura 2.1), oltre alle funzioni precedentemente descritte, nasce con la peculiarità di essere compatto. Infatti, la prima cosa che si può notare è la presenza di un giunto che collega l'HLA (Hydraulic Load Actuator) posto verticalmente, con l'AUT (Actuator Under Test), posto orizzontalmente. Grazie all'elemento a gomito, infatti, è possibile andare a scambiare le forze fra i due dispositivi, salvaguardando una notevole quantità di spazio e grazie ad un encoder, si conosce la posizione relativa fra i due. L'attuatore sotto test, attualmente in uso, è di tipo elettromeccanico e viene controllato in posizione, mentre l'HLA viene controllato in forza. Per garantire il pilotaggio di quest'ultimo, sul banco è presente una servo valvola che si occupa del controllo, e una valvola di shut-off usata per mettere in sicurezza il tutto, in caso di emergenza (figura 2.2). All'esterno della struttura è presente un Manifold (figura 2.3) in grado di regolare la pressione di alimentazione proveniente dalla centrale principale (figura 2.4), gestire le valvole di esclusione, regolare l'avviamento progressivo ecc... A integrazione della macchina è presente l'ACC (Acquisition and Control Cabinet), che contiene il sistema di acquisizione e controllo, il Driver del motore elettrico, un rack contenente i sensori di tensione e corrente, con i rispettivi alimentatori e la postazione utente.

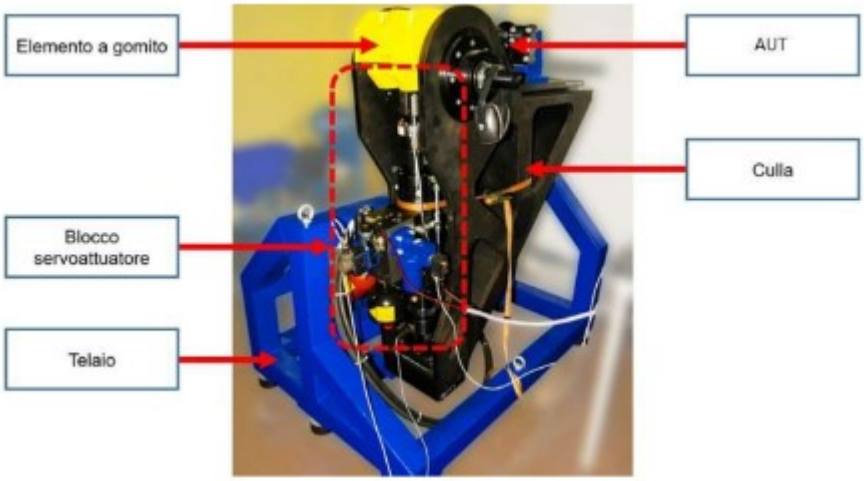


Figura 2.1: Banco Prove.

Figura 2.2: Compartimento idraulico del HLA.

Figura 2.3: Manifold.

Figura 2.4: Centrale idraulica.

All'interno dell'ACC è presente una Compact-Rio 9039, ovvero il cervello della macchina, responsabile del controllo e acquisizione di tutti i segnali.

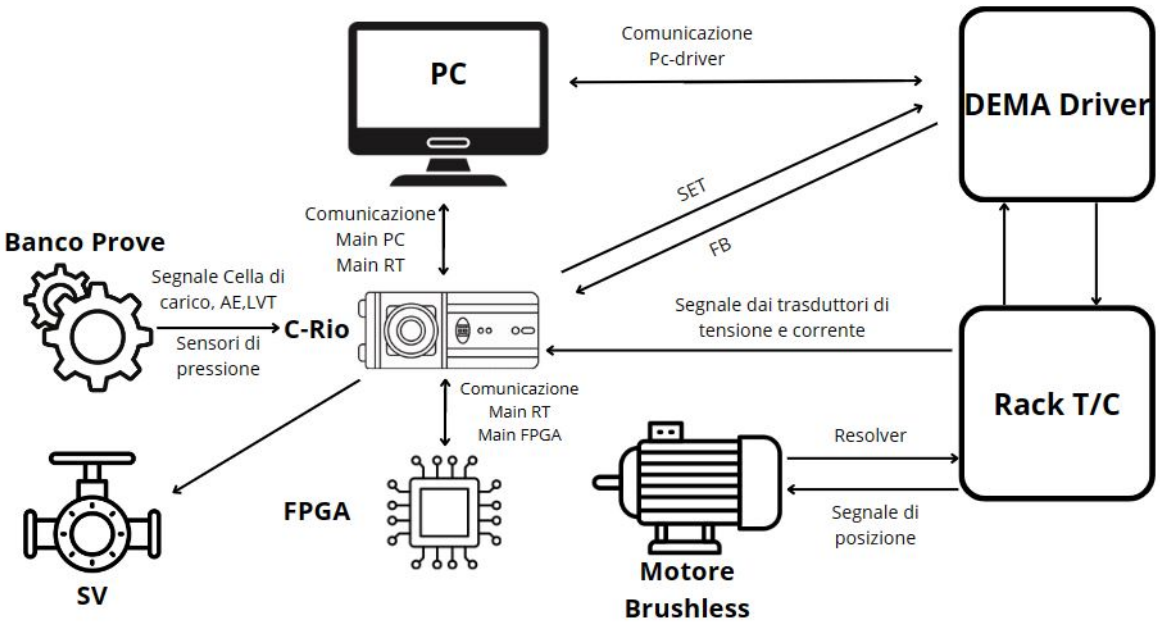


Figura 2.5: Collegamenti fra i vari device

Osservando la figura 2.5, si vede come quest'ultima gestisca il flusso di informazioni diretto e proveniente da tutti i dispositivi presenti in questo ambiente. Partendo dal banco prove, come già accennato, l'attuatore elettroidraulico viene pilotato da una servovalvola (SV). A quest'ultima la Compact-Rio invia un segnale di Set di forza.

L'HIL si muove e la cella di carico invia il Feedback all'ACC, insieme a tutte le altre informazioni provenienti dagli altri sensori presenti sul banco, ovvero: Encoder Angolare (AE) presente sul giunto, LVT (Linear Varible Velocity) presente sullo stelo dell'attuatore idraulico e sensori di pressione posizionati in prossimità delle camere del cilindro. Invece, all'AUT viene inviato un Set di posizione, il cui feedback viene elaborato dal resolver che possiede al suo interno. Il Set, una volta uscito dalla C-Rio, passa per il Driver DEMA, il quale trasforma l'input di controllo in un output di potenza. L'EMA è un motore trifase, quindi grazie al Rack contenente i sensori (esterno al banco), posto tra quest'ultimo e il driver, vengono letti tre segnali di tensione e tre di corrente. Infine, la C-Rio è collegata tramite la postazione utente, dalla quale l'operatore gestisce il tutto.

Nella figura 2.6 è presente uno schema più dettagliato dei collegamenti presenti nel banco, indicando anche i vari moduli che fanno da tramite fra i sensori e la Compact-Rio. Come si può notare, tratteggiato è presente l'ACC che racchiude C-Rio e moduli.

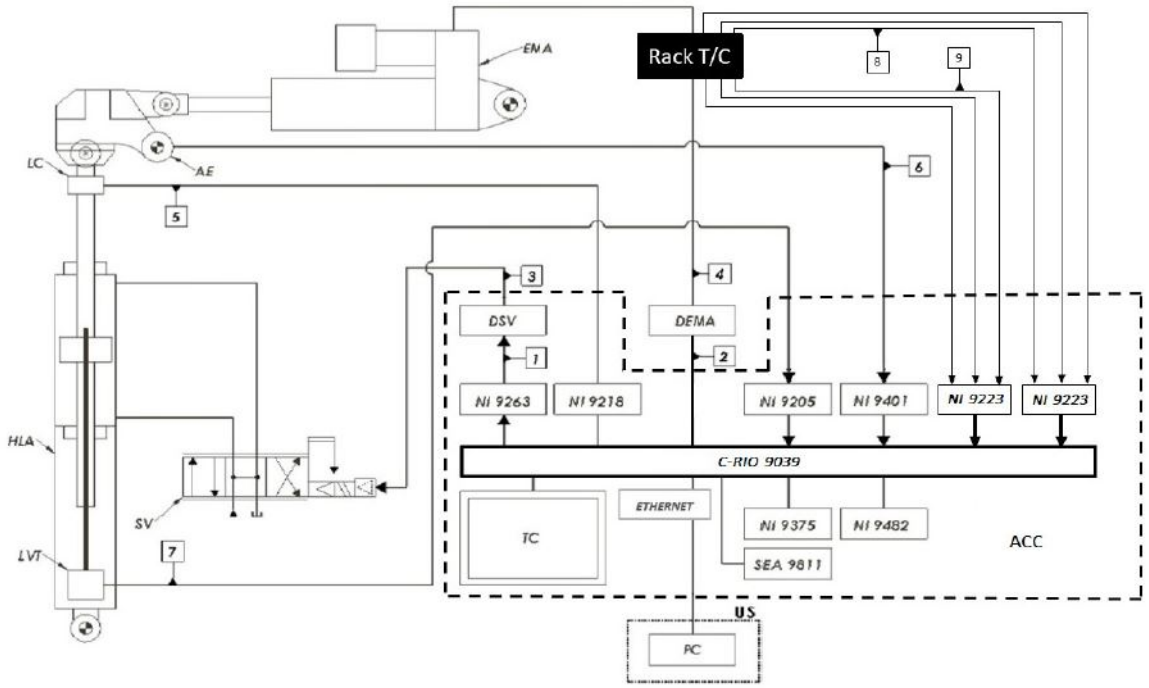


Figura 2.6: Schema collegamenti del banco. Fonte: [7].

Numero	Descrizione
1	Segnale di comando della SV in uscita dal controllo
2	Segnale di riferimento per l'EMA in ingresso al driver
3	Segnale di comando della SV in uscita dal driver
4	Segnale di comando dell'EMA in uscita dal driver
5	Segnale in uscita dalla LC
6	Segnale in uscita dall'encoder
7	Segnale in uscita dall'LVT
8	Segnale di tensione in uscita dal Rack tensione-corrente
9	Segnale di corrente in uscita dal Rack tensione-corrente

Tabella 2.1: Legenda dei segnali

2.3 Dispositivi hardware utilizzati nel banco

2.3.1 HLA – caratteristiche generali

L'HLA è un cilindro idraulico a basso attrito con idro-sostentamento dello stelo prodotto dalla Bosch Rexroth. All'interno dello stelo è presente un sensore di tipo LVT, mentre, in cima, si trova la cella di carico. La corsa nominale è di circa 100 mm, con un'extra corsa di 20 mm, arrivando ad un totale di 120 mm. L'alesaggio del cilindro è di 65 mm mentre il diametro dello stelo è di 50 mm, da cui si può ricavare un'area utile di spinta di $1.355*10^{-3}m^2$. Dal manuale del banco è possibile ricavare il grafico della forza di stallo in funzione della differenza di pressione tra quella di alimentazione Pp e quella di ritorno PT. Per farlo sono state adottate le seguenti ipotesi:

- Pressione di ritorno PT = 0 bar;
- Forze di attrito interne trascurabili;
- Leakage attraverso le camere ed attraverso il bypass nullo;

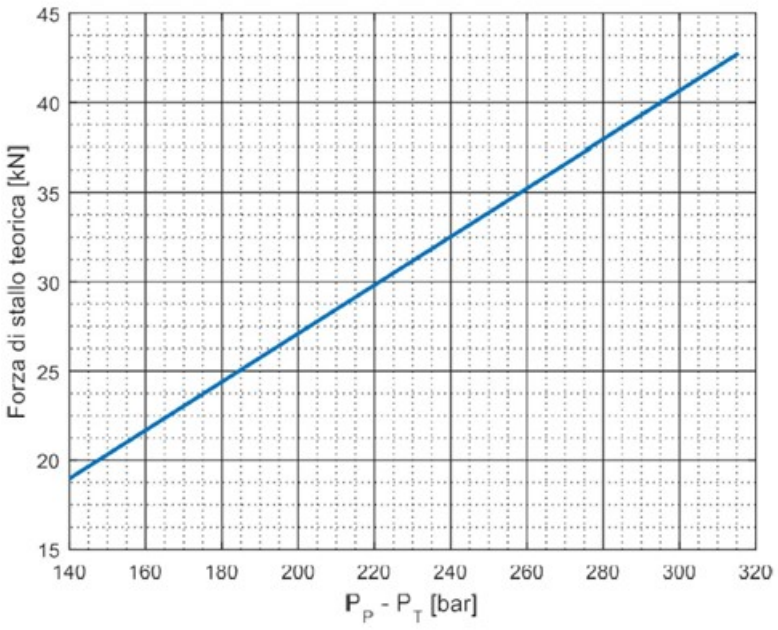


Figura 2.7: Grafico forza di stallo teorica.

2.3.2 Servovalvola

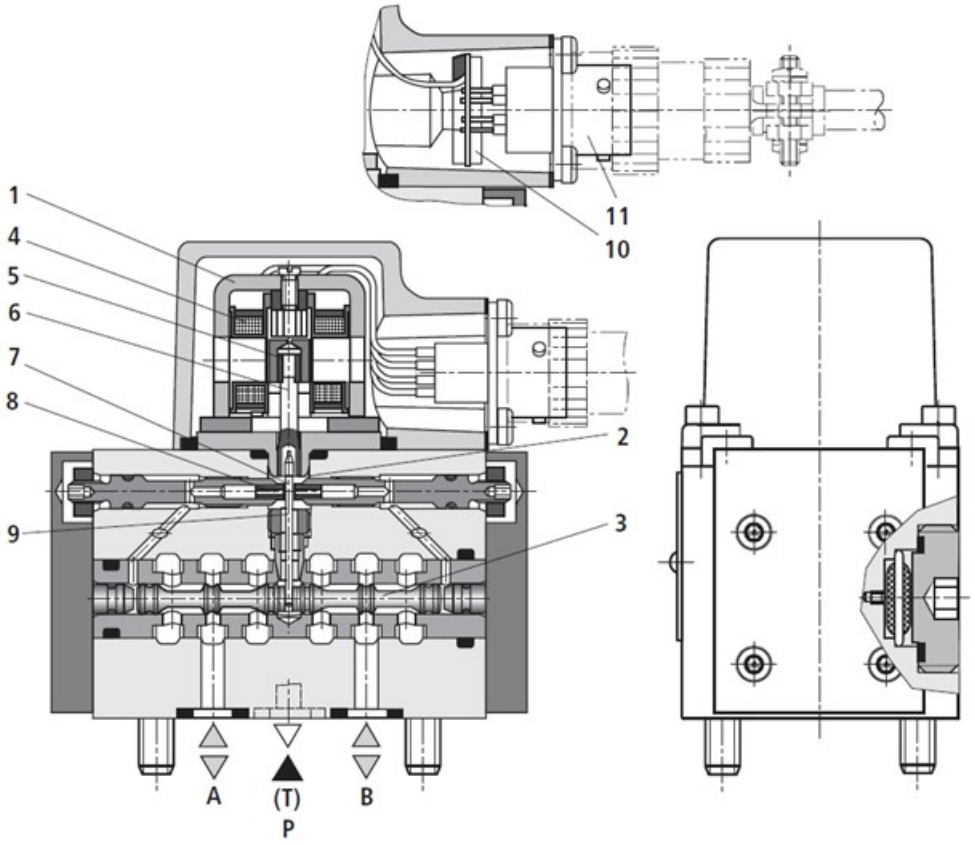


Figura 2.8: Servovalvola Rexroth Bosch Group s.d.

La valvola in questione è un prodotto Bosch Rexroth ed è il modello 4WSE2EM-6-2X-20-8-ET-315-K17-E-V. Il modulo che si occupa del controllo di questo dispositivo è il NI9263.

Il principio di funzionamento di questo strumento (figura 2.8), si basa principalmente sulla presenza di un motore coppia (*torque motor*) ed un amplificatore idraulico di tipo *flapper nozzle*, collegati entrambi un feedback meccanico, chiamato balestra. Nel momento in cui arriva un segnale elettrico alla bobina, si genera una coppia magnetica, che fa ruotare di pochi gradi il perno (6), il quale è solidale alla piastra valvola a cerniera (7). Nel momento in cui questo ruota, sposta quest'ultima dalla sua posizione centrale che va a chiudere uno dei due sensori a contropressione (flapper nozzle 8) e si genera così una pressione differenziale ai capi del pistone (3) che lo sposta dalla sua posizione originale e permette al fluido di passare. Grazie alla presenza del feedback meccanico (9), nel momento in cui il segnale elettrico viene meno, il pistone viene riportato nella sua configurazione

di riposo.

Importante è il fatto che la portata della servo-valvola viene regolata in proporzione al segnale elettrico. Infatti, più il segnale è elevato, più la coppia magnetica è maggiore e quindi più viene chiuso l'ugello del sensore. Di conseguenza maggiore sarà la differenza di pressione e maggiore sarà lo spostamento. Grazie alla presenza della balestra (9), la coppia elastica applicata sarà sempre proporzionale alla coppia magnetica e quindi al movimento del pistone.

Per quanto riguarda l'amplificazione del segnale in ingresso, è presente un controllo elettronico nella valvola (10).

Si riporta il simbolo pneumatico della valvola (fig. 2.9):

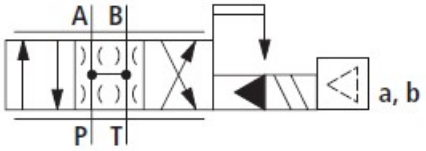


Figura 2.9: Simbolo pneumatico di una valvola proporzionale a 4 vie, 3 posizioni.

Informazioni generali sulla servovalvola:

Portata nominale

La portata nominale calcolata con una Δp (pressione differenziale di valvola) di 70 bar, è di 20 l/min, definita come:

$$\Delta p = P_p - P_T - \Delta p_L$$

Dove:

- P_p è la pressione di alimentazione;
- P_T è la pressione di ritorno;
- Δp_L è la differenza di pressione sul carico

Con riferimento alla Figura 2.10, supponendo che la pressione di alimentazione PP sia di 70 bar e la pressione di ritorno PT di 0 bar, in assenza di carico la pressione

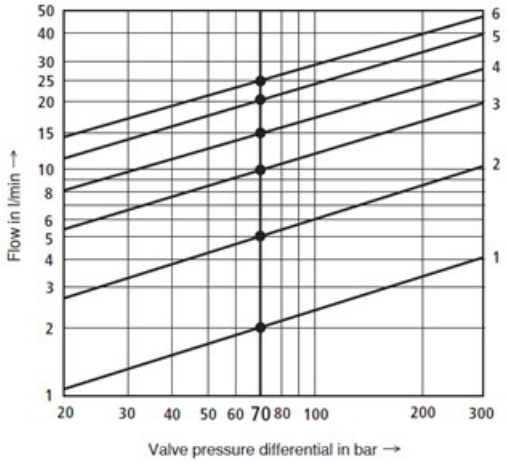


Figura 2.10: Curva di portata in funzione della pressione differenziale. La valvola utilizzata fa riferimento alla curva 5.

differenziale di valvola Δp vale 70 bar, valore per cui viene fornita dal costruttore la portata nominale. Inoltre, supponendo la portate S→A e B→T siano uguali, le cadute di pressione su ciascuna linea valgono entrambe 35 bar.

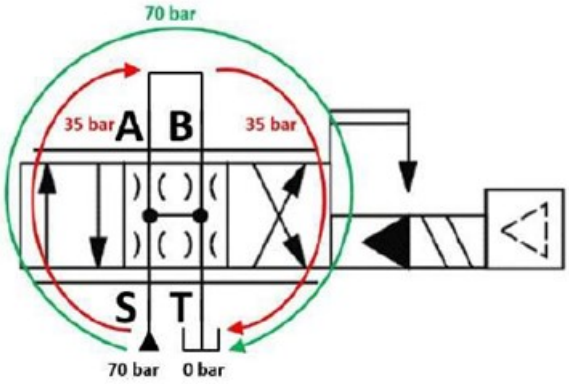


Figura 2.11: Cadute di pressione nella servovalvola.

Ricoprimento

In funzione delle informazioni ricavate dal datasheet del costruttore, la servo-valvola presenta un ricoprimento negativo che va dallo 0% al 5%. Le valvole impiegate nei servocomandi di volo utilizzano spesso ricoprimenti negativi (intorno i 4-10 micron), per rendere più reattivi i comandi. Infatti, un minimo spostamento è sufficiente per avere una reazione. Di seguito si riporta il grafico di risposta della servo-valvola in funzione dello spostamento del cassetto.

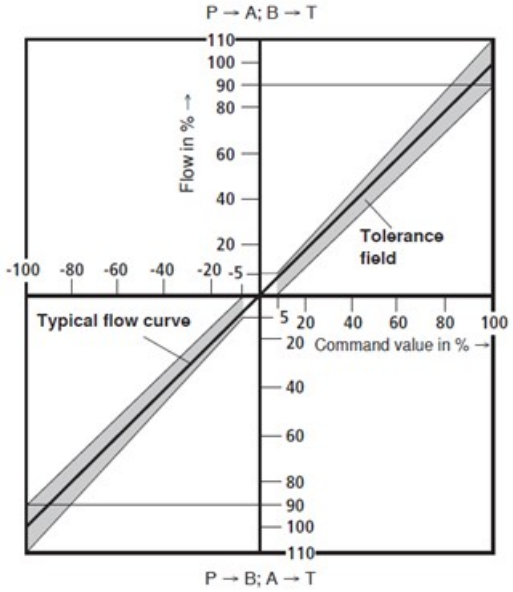


Figura 2.12: Portata in funzione dello spostamento.

Come si può osservare, in grigio viene rappresentato il discostamento dalla linearità, ovvero il campo di tolleranza. Per completezza si riporta anche il grafico del tempo di risposta della valvola, al variare dei diversi Δp (la valvola di riferimento ha Δp=70bar). Interessante notare come a 70 bar non sia presente overshoot, al contrario invece delle

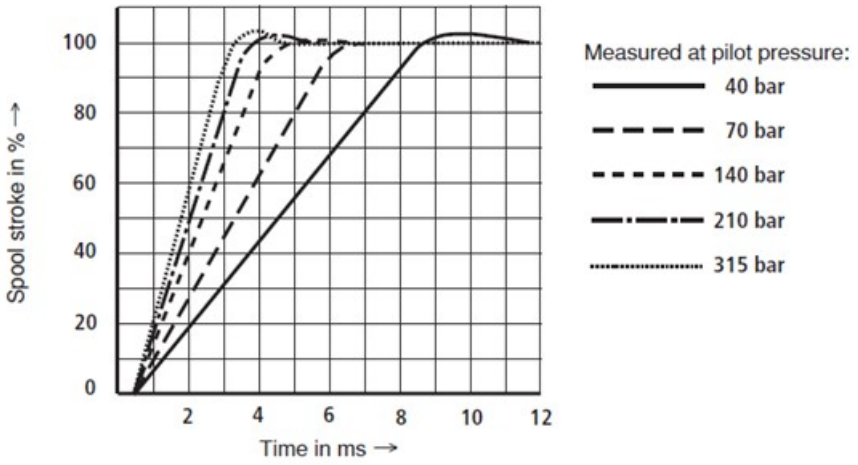


Figura 2.13: Tempi di risposta della servo-valvola.

altre configurazioni, compresa quella a 40 bar.

2.3.3 Valvola di shut-off

Anche questo componente è stato acquistato presso la Bosch Rexroth, il cui codice identificativo è 6-E63-3X-E-G24-N9-K4. Il suo compito è quello di interrompere il flusso di olio all'attautore pneumatico nel caso in cui si verifichi un'emergenza. Essendo una valvola normalmente chiusa, nel momento in cui l'alimentazione viene meno, interrompe il flusso e mette in sicurezza l'impianto. Il modulo che si occupa del suo controllo è il NI 9482. Il rispettivo simbolo è osservabile nella figura 2.14

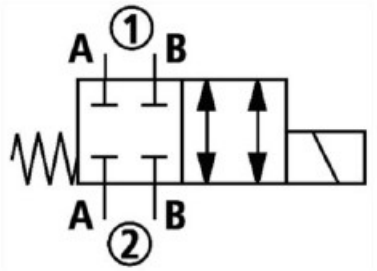


Figura 2.14: Shut-off.

2.3.4 Attuatore in prova

L'attuatore montato all'interno del banco prove non è un dispositivo utilizzato in ambito aeronautico. Tale modello è stato selezionato in quanto era già stato provato e si conoscevano le sue caratteristiche. Quindi, essendo il banco sperimentale, la scelta di tale motore è stata fatta in modo tale da testare anche le performances della macchina.

Si tratta di un LEMC-S-2105, con servo motore Lenze, dove la sigla LEMC indica "Linear Electro-Mechanical Actuator". L'insieme (fig. 2.17) è formato da tre blocchi interni al dispositivo, più il driver esterno, ovvero:

- Servomotore sincrono a corrente alternata brushless della Lenze;
- Trasmissione a cinghia;
- Sistema vite-madrevite a rulli satelliti;
- Driver (Servo-Drive Lenze 9400) per il controllo del motore (esterno all'EMA).

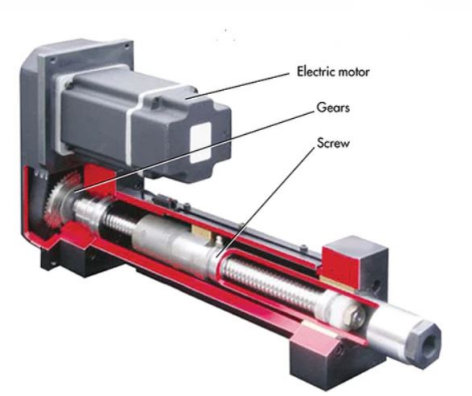


Figura 2.15: Esempio di un attuatore elettromeccanico con lo scopo di mostrarne i componenti interni

Come si vede nella figura 2.15, il servo motore è collegato ad un sistema a cinghia, il cui rapporto di trasmissione (in questo caso specifico) è di 1:1. A questo, più in basso, è connesso un attuatore lineare a vite con rulli a satelliti SKF. Dalla figura 2.17 si nota come da sopra escano due cavi, uno di alimentazione del motore e uno del resolver. Quest'ultimo è utilizzato come feedback dal driver per generare il comando di retroazione.

Lo schema a blocchi si presenta così:

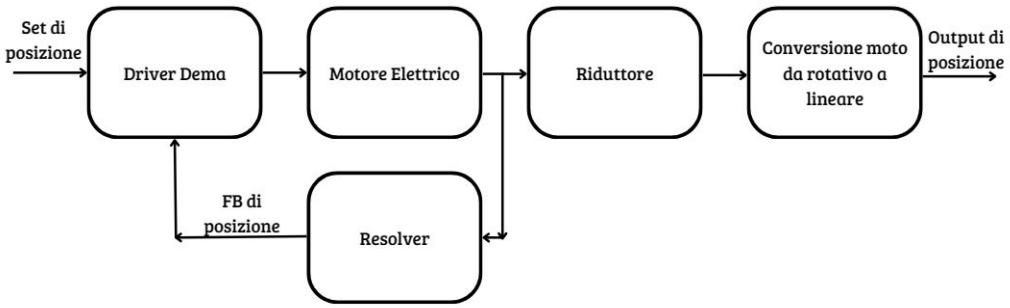


Figura 2.16: Schema a blocchi dell'EMA

Come si nota nella figura 2.16, al Driver arriva un Set di posizione dalla Compact-Rio, il quale viene confrontato con il Feedback proveniente dal resolver del motore. L'errore in uscita viene condizionato ed inviato al motore elettrico che si traduce così in un output lineare.

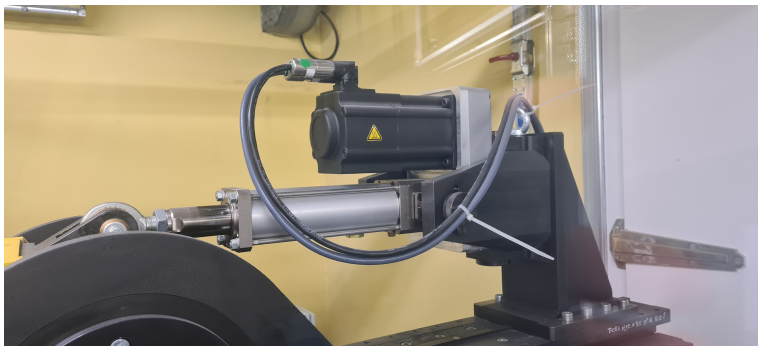


Figura 2.17: Attuatore elettromeccanico attualmente montato sul banco prove

Nella tabella i parametri fondamentali del LEMC:

Descrizione	Valore	Unità di misura
Rapporto di trasmissione	1:1	-
Diametro della vite	21	mm
Passo	5	mm
Massima forza assiale	40	kN
Massima velocità di traslazione	500	mm/s
Massima velocità rotativa	6000	rpm
Corsa nominale	100	mm
Momento d'inerzia parti rotanti	4.00×10^{-4}	Kgm^2
Coppia nominale	5.5	Nm
Coppia massima	18	Nm
Potenza nominale	1.1	kW
Corrente nominale	2.6	A
Corrente massima	10	A

Tabella 2.2: Dati tecnici

Servo motore MCS12D20 Lenze

E' un motore sincrono brushless a corrente alternata prodotto dalla Lenze. E' un tipo attuatore che non ha spazzole meccaniche per il trasferimento di potenza elettrica. Utilizza invece, magneti permanenti e un controllo elettronico per generare un campo magnetico rotante che interagisce con il campo magnetico prodotto dall'avvolgimento statorico per creare movimento. La sua struttura è composta da un motore sincrono, il quale ospita un rotore che contiene magneti permanenti e uno statore con avvolgimenti di bobine di rame. Il rotore è alloggiato sull'albero del motore e può ruotare liberamente all'interno dello statore. Il Driver regola l'alimentazione delle fasi del motore per creare un campo magnetico rotante che spinge il rotore. Il feedback fornisce informazioni sulla posizione del rotore, consentendo al sistema di controllo di regolare la corrente inviata al motore per raggiungere la posizione desiderata.

Il motore riceve in ingresso un'alimentazione di 400V, nel grafico 2.18 si può osservare la rispettiva caratteristica.

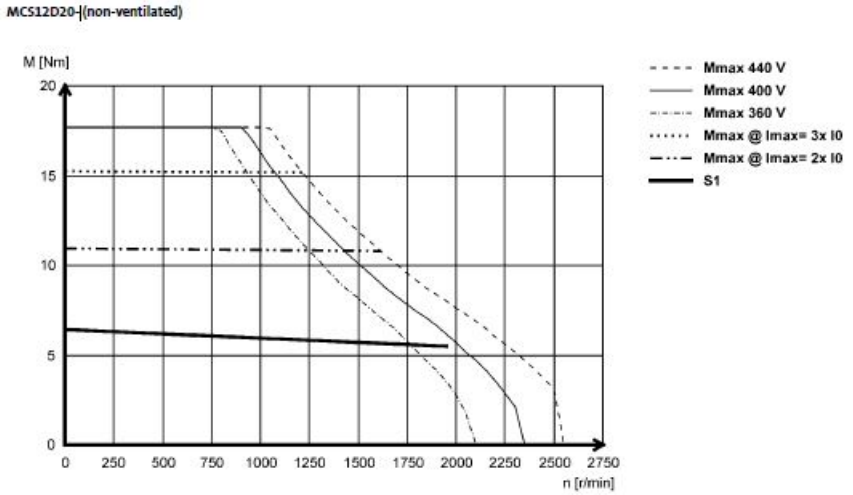


Figura 2.18: Caratteristica MCS12D20. Fonte: [9].

Mentre nella tabella 2.3 sono presenti i dati principale del motore elettrico.

Dato	Valore	U.M.
Velocità nominale n_N	1950	rpm
Coppia nominale M_N	5.5	Nm
Coppia massima M_{max}	18	Nm
Potenza nominale P_N	1.1	kW
Corrente nominale I_N	2.6	A
Corrente massima I_{max}	10	A
Voltaggio nominale U_N AC	345	V
Frequenza nominale f_N	130	Hz
Efficienza motore η	79	%
Inerzia motore I_M	4	$Kgcm^2$
Costante di tensione k_e	1.31	V/(rad/s)
Resistenza statore R_s	5.8725	Ω
Induttanza nominale L_N	52.2	mH
Costante di coppia k_c	2.34	Nm/A
Velocità massima n_{max}	6000	rpm

Tabella 2.3: Descrizione dei dati tecnici

Trasmissione a cinghia

La trasmissione a cinghia serve per trasferire il moto dal Servo motore Lenze al sistema vite-madrevite. Il rapporto di trasmissione è unitario quindi non ci sono variazioni di velocità fra i due elementi.

Sistema a vite con rulli satelliti

Figura 2.19: Sistema a vite con rulli satelliti. Fonte: [10].

In riferimento alla figura 2.19: la madrevite (elemento più esterno), con filettatura interna e filetto complementare a quello della vite, costituisce il primo elemento di trasmissione della catena ed è connessa al sistema di trasmissione a cinghia. I rulli satelliti (i più piccolini, intorno la madrevite principale) presentano una filettatura arrotondata che facilita il trasferimento del contatto, simile a quello delle sfere. La vite, l'ultimo componente della trasmissione, agisce come elemento di attuazione che si muove traslando grazie al meccanismo a rulli satelliti. Infine, la gabbia mantiene la posizione relativa tra i rulli satelliti.

2.3.5 Servo-Drive Lenze 9400

Anche se non appartiene direttamente al banco prove, è un dispositivo fondamentale per garantire il funzionamento del motore elettrico. Il Servo-Drive Lenze 9400 (fig. 2.20) è il driver dell' EMA. E' un prodotto della Lenze, il cui modello è E94ASHE004.

Si occupa di ricevere i segnali di controllo dalla C-RIO e trasformarli in segnali di potenza per il motore.

Figura 2.20: E94ASHE004. Fonte: [11].

Il driver (schema generale, fig. 2.21) riceve l'alimentazione elettrica (corrente trifase alternata a 50 HZ) dalla rete. Utilizzando un raddrizzatore (1° blocco blu), converte questa corrente AC in corrente continua DC e la invia al DC bus link (2° blocco blu) per il filtraggio tramite resistenze e capacità, eliminando così le oscillazioni indesiderate. La corrente continua viene quindi inviata all'inverter (3° blocco blu), dove tramite un ponte ad H a 6 transistor controllati dalla logica di controllo, viene convertita nuovamente in corrente alternata AC con la frequenza e l'ampiezza desiderate per alimentare il motore. Questo processo utilizza la tecnica di modulazione PWM per controllare la frequenza di alimentazione del motore in corrente alternata, consentendo di regolarla in modo indipendente dalla frequenza della rete.

Infine, il driver invia la corrente alla bobina delle tre fasi attraverso il cavo motore e riceve il segnale di trasduzione dal resolver.

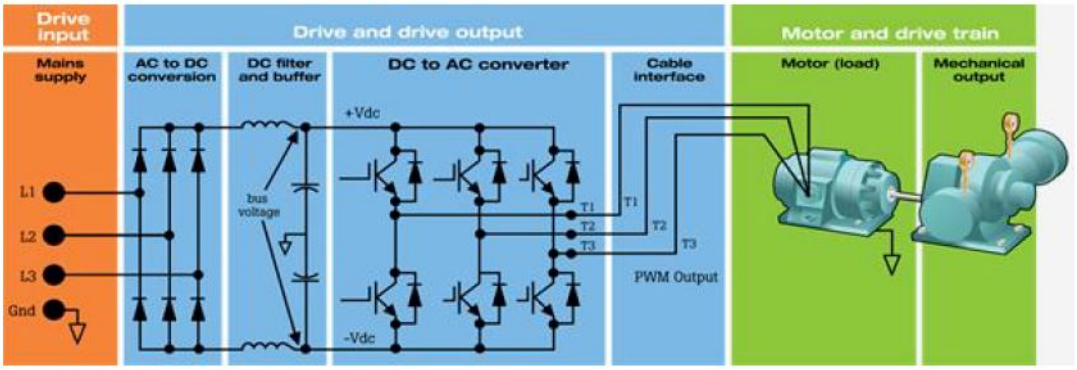


Figura 2.21: Schema collegamento rete-motore. Fonte: [7].

Dalla tabella 2.4 si possono osservare le caratteristiche principali del dispositivo:

Specifiche	Valore	Unità di misura
Dati in ingresso		
Rete	3/PE AC	-
Tensione	400	V
Frequenza	50	Hz
Corrente nominale	5.5	A
Numero di fasi	3	-
Dati in uscita		
Rete	3/PE AC	-
Tensione	0-400	V
Frequenza	0-599	Hz
Corrente nominale	4	A
Numero di fasi	3	-

Tabella 2.4: Specifiche del dispositivo

Il driver lavora con un PWM e fornisce una tensione tra fasi e neutro di +/-400V sotto-forma di onda quadra con una frequenza del chopper di 8kHz. Quest'ultimo è responsabile del controllo della tensione applicata al motore, che viene modulata in modo da ottenere l'onda PWM desiderata. Agisce tagliando la tensione di alimentazione in intervalli di tempo regolari, regolando così la durata e la frequenza degli impulsi. La media risultante corrisponde ad una sinusoide di frequenza compresa tra 0 e 599Hz. L'informazione sul chopper è molto importante in quanto, in origine durante la fase di progettazione del nuovo sistema di acquisizione di tensioni e correnti, è quella su cui si è basata la scelta dei rispettivi moduli (NI 9223).

2.4 ACC

L'ACC (Acquisition And Control Cabinet), schematizzata in figura 2.22, è l'armadio che si occupa di contenere:

- C-RIO e relativi moduli;
- Driver della servovalvola;
- Un monitor touch screen da 17";
- Diverse Breakout Box;
- Un alimentatore da 24V;
- Tutti i relè necessari per le emergenze.

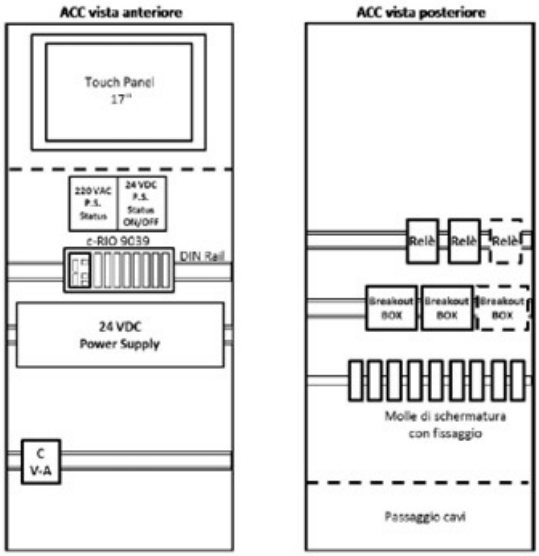


Figura 2.22: ACC. Fonte:

Nelle prossime pagine verrà analizzata la C-Rio con i rispettivi moduli.

2.4.1 Compact Rio 9039

Il dispositivo responsabile di tutte le operazioni acquisizione e controllo è la C-RIO 9039 (fig. 2.23).

Figura 2.23: c-Rio 9039. Fonte: [12]

Gli elementi pilotati sono la servovalvola SV e l'attuatore elettromeccanico in prova, dove i segnali vengono generati direttamente dalla C-RIO, la quale comunica con l'operatore attraverso un PC. Anche la valvola di shut-off, che viene alimentata a 24V, è pilotata tramite un impulso digitale proveniente sempre dal suddetto dispositivo.

Per quanto riguarda la lettura dei segnali, si acquisiscono: la cella di carico, il sensore LVT, i sensori di pressione, l'encoder angolare e, più di recente, sono stati introdotti nuovi moduli che permettono la lettura dei segnali di tensione e corrente dell'attuatore in prova.

Il compact Rio possiede otto moduli I/O C- Series (immagine 2.24), grazie ai quali gestisce l'ingresso e l'uscita dei segnali dalla macchina. È un dispositivo compatto e robusto, che garantisce prestazioni molto elevate. Il controllo ha doppi ingressi di alimentazione che vanno da 9 VDC a 30 VDC e possiede due porte Gigabit Ethernet, due USB Hi-speed Host, una USB device e due porte seriali.

Caratteristiche tecniche nella tabella 2.5:

Tabella 2.5: Specifiche del Sistema. Fonte: [12]

Processore	Quad-core Intel Atom E3845 da 1.92 GHz con 2 Mb di cache
Memoria	16 GB (non volatile) espandibile fino a 32 GB con SD
Sistema operativo	NI Linux Real-Time
FPGA	Xilinx Kintex – 7 7K328T
Alimentazione	Continua da 9-30 V
Consumo massimo	46 W

La macchina permette l'utilizzo di LabView per la creazione, debug e distribuzione di logica su FPGA su scheda sul processore NI Linux Real-Time OS.

Figura 2.24: Moduli installati sul banco

Di seguito un elenco dei moduli presenti sulla C-Rio.

2.4.2 NI 9205

È un modulo di ingresso analogico (fig. 2.25) da 16 canali differenziali o 32 a terminazione singola.

Misura il segnale in uscita dal LVT, lavorando nel range di ± 1 V.

Figura 2.25: Ni 9205. Fonte: [13]

Caratteristiche tecniche:

Tabella 2.6: Specifiche del modulo

Risoluzione	16 bit
Frequenza di campionamento	250 kS/s
Canali	16
Input range	±10 V, ±5 V, ±1 V, ±0.2 V
Larghezza di banda (-3dB)	370 kHz

2.4.3 NI 9218

Questo modulo (fig. 2.26) è capace di acquisire segnali da diversi tipi di sensori. È dotato solamente di 2 canali ed è l'ideale per applicazioni di test di sistemi data-logging e automotive. In questa applicazione viene utilizzato per condizionare ed acquisire il segnale proveniente dalla cella di carico.

Figura 2.26: Ni 9218. Fonte: [14]

Caratteristiche tecniche:

Tabella 2.7: Specifiche del modulo

Risoluzione	24 bit
Frequenza di campionamento	51.2 kS/s
Canali	2

2.4.4 NI 9263

È un modulo analogico di output (fig. 2.27) a 4 canali in grado di generare segnali in tensioni di ± 10 V. In questo caso particolare, viene utilizzato per pilotare la servo-valvola (SV).

Figura 2.27: Ni 9263. Fonte: [15]

Caratteristiche tecniche:

Tabella 2.8: Specifiche del sistema

Risoluzione	16 bit
Range di funzionamento	±10 V
Frequenza di acquisizione (per canale)	100 kS/s

2.4.5 NI 9482

È un modulo in grado di generare segnali digitali a 4 canali (fig. 2.28). Ogni canale è dotato di un led di stato per indicarne il funzionamento ed in questa applicazione viene utilizzato per generare i segnali di emergenza da inviare alla servovalvola (SOV) e ad altri dispositivi di sicurezza come gli interlock delle porte.

Figura 2.28: Ni 9482. Fonte: [16]

Caratteristiche tecniche:

Tabella 2.9: Specifiche dei Canali Digitali

Canali digitali	4
Tempo di switching	1 s/S
Segnali elettrici	30 VDC con una corrente massima di 1.5 A

2.4.6 NI 9401

È un modulo digitale di input o output, fino ad 8 canali, che lavora su un range di tensioni che va da 0 a 5 V (fig. 2.29). In questa applicazione viene usato per acquisire i dati provenienti dall'encoder angolare.

Figura 2.29: Ni 9401. Fonte:[17]

Caratteristiche tecniche:

Tabella 2.10: Specifiche dei Canali

Canali	8 (input o output o 4 canali I/O)
Massima frequenza di switching (per ogni canale)	9 MHz
Tensione di lavoro	0 – 5 V

2.4.7 NI 9375

È un modulo in grado di generare (e ricevere) segnali digitali da 24 V, con 16 canali di input e 16 di output (fig. 2.30). È utilizzato per gestire l'interfaccia di sicurezza, ovvero: avvio ciclo, stop ciclo, reset allarmi, emergenza ripristinata, presenza allarmi, apertura ripari ecc...

Figura 2.30: Ni 9375. Fonte: [18]

Caratteristiche tecniche:

Tabella 2.11: Frequenza di Aggiornamento

Frequenza di aggiornamento (ingresso)	7 μs
Frequenza di aggiornamento (uscita)	500 μs

2.4.8 NI 9223

Gli ultimi moduli introdotti dopo l'aggiornamento hardware sono i NI 9223 (fig. 2.31), i quali si occupano della gestione dell'acquisizione dei segnali di tensione (generati con la tecnica PWM) e corrente provenienti dal motore. Prima della loro scelta, è stato analizzato il funzionamento del chopper del driver, ovvero il dispositivo responsabile della generazione di questo tipo di onda. Una volta verificato che la frequenza era di 8 kHz, in accordo con il teorema di Nyquist, per ricostruire correttamente il segnale bisogna campionare ad una frequenza che sia almeno il doppio di quella che si sta acquisendo. Tuttavia, volendo non solo riconoscere la forma d'onda, ma anche i singoli picchi del PWM, gli overshoot e tutti quei fenomeni che richiedono una risoluzione molto elevata, si è deciso di optare per un modulo in grado di arrivare fino a 1 MS/s, così da ottenere il maggior numero possibile di informazioni.

Figura 2.31: Ni 9223. Fonte: [19]

Caratteristiche tecniche:

Tabella 2.12: Specifiche di Risoluzione e Frequenza di Campionamento

Risoluzione	16 bit
Numero di canali input	4
FPGA User-Controlled I/O Sampling	1 MS/s
FPGA I/O Nodes	350 kS/s

Le ultime due righe della tabella indicano le modalità con cui il modulo può acquisire e le rispettive velocità di campionamento.

2.5 Sensoristica

2.5.1 Rack di acquisizione tensioni e correnti

Il rack è un dispositivo custom, introdotto dopo l'ultimo aggiornamento hardware, realizzato con lo scopo di acquisire i segnali di tensione e corrente provenienti dal motore brushless. Viene posto tra l'EMA e il Driver ed al suo interno sono presenti:

- Tre sensori di corrente DaniSense DS50UB-10V (fig. 2.32a);
- Tre alimentatori dei sensori di corrente MBRD-0-A-XX;
- Un sensore di tensione VeriVolt IsoBlock-V4c (fig. 2.32b);
- Alimentazione del sensore di tensione;

(a) DaniSense Ds50UB-10V. Fonte: [20]

(b) VeriVolt IsoBlock -V4c. Fonte: [21]

Dopo l'introduzione del rack, quando si accende il driver del motore, si sono iniziati a notare disturbi elettromagnetici durante il funzionamento della macchina. Quindi lo schema seguente (2.33) non è per tanto una versione definitiva dello stesso.

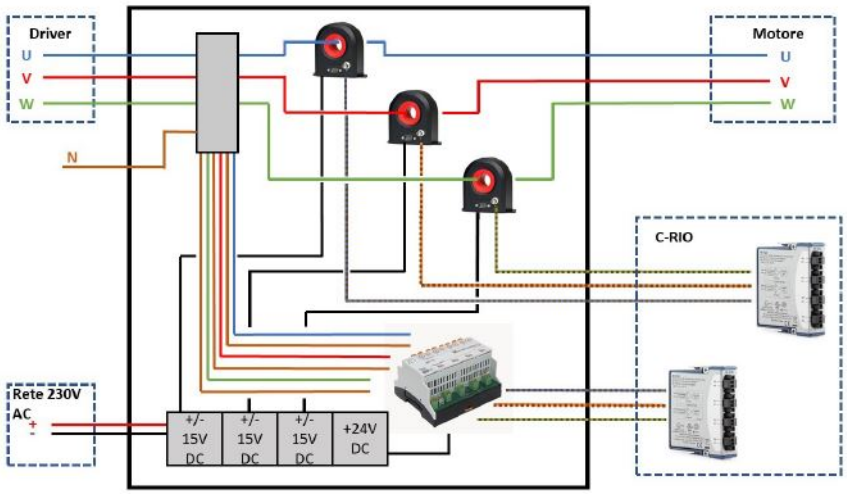


Figura 2.33: Schema rack. Fonte: [7]

I dispositivi DaniSense sono collegati sia ai rispettivi alimentatori sia ai moduli NI 9223. Al loro interno passano i cavi delle tre fasi del motore (U,V,W). Allo stesso modo il sensore VeriVolt è collegato al rispettivo alimentatore e al driver del motore elettrico. In uscita si trovano le tre fasi che vanno al secondo modulo NI 9223.

DaniSense DS50UB-10V

Le caratteristiche riportate nella tabella 2.13:

Parameter	Unit	Min	Max
Nominal primary AC current	Arms		36
Nominal primary DC current	A	-50	50
Measuring range	A	-55	55
Ratio error	ppm	-50	50
Bandwidth (-3dB)	kHz	500	

Tabella 2.13: Description of parameters

Il diagramma di Bode (2.34), mostra come all'aumentare della frequenza, l'accuratezza del segnale viene meno.

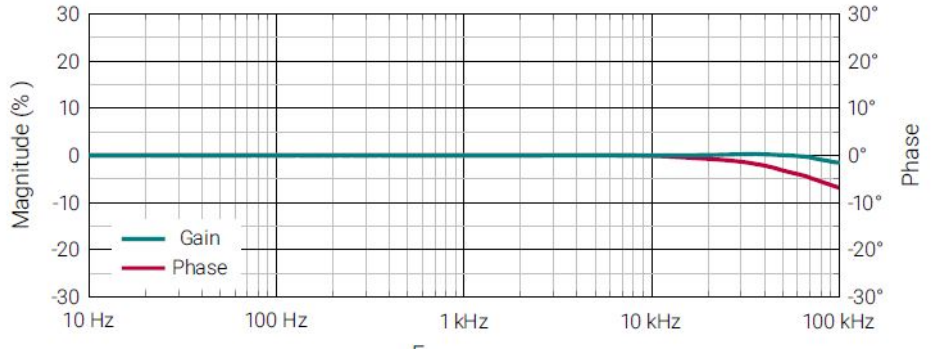


Figura 2.34: Bode Magnitudo-fase su frequenza. Fonte: [11].

Si riportano anche i valori di accuratezza in percentuale e sfasamento in gradi nella figura 2.35

Total accuracy without offset		ϵ_{tot}	*% of reading + % of full scale*
	<10 Hz		0.0055 + 0.0001
	<100 Hz		0.0055 + 0.0002
	<1 kHz		0.01 + 0.0003
	<10 kHz		0.02 + 0.001
	<100 kHz		2 + 0.01
	<500 kHz		30 + 0.02
Phase shift	<10 Hz		0.01°
	<100 Hz		0.01°
	<1 kHz		0.02°
	<10 kHz		0.2°
	<100 kHz		8°

Figura 2.35: Accuratezza e sfasamento all'aumentare della frequenza. Fonte: [11].

VeriVolt IsoBlock -V4c

Allo stesso modo, nella tabella 2.14, sono presenti i parametri fondamentali dello strumento. Nella figura 2.36, il rack contenente gli elementi precedentemente elencati.

Parameter	Value
Input ranges	±50V, ±100V, 150VAC, ±300V, ±500V, 500VAC, ±750V, ±1000V, ±1500V, Custom
Bandwidth (-3dB point)	100kHz (1MHz option)
Input-Output non-linearity	$< 0.04\%$
Output voltage	±10V, 7VAC, ±5V

Tabella 2.14: Description of parameters

Figura 2.36: Rack contenente i sensori di tensione e corrente

2.5.2 Cella di carico

La cella di carico in questione è un prodotto della Metior, modello CVC 50 kN (fig.2.37). Viene utilizzata per misurare la forza esercitata dall'attuatore in esame. Sfrutta un ponte di Wheatstone per misurare una grandezza in ingresso (in questo caso una forza). Il ponte è un circuito elettrico che fornisce una misura precisa di variazioni di resistenza e può essere utilizzato per convertire lo sforzo meccanico applicato sulla cella di carico in un segnale elettrico misurabile. In questo caso particolare, il costruttore non fornisce alcun dato sul tipo di ponte utilizzato.

Caratteristiche tecniche (tab. 2.15):

Caratteristica	Valore
Forza massima misurabile	50 kN
Sensibilità nominale	2 mV/V
Accuratezza	0.1% FS
Larghezza di banda	-3 dB
Guadagno	0.4 mV/kN

Tabella 2.15: Caratteristiche del Sensore di Forza.

Figura 2.37: Cella di carico

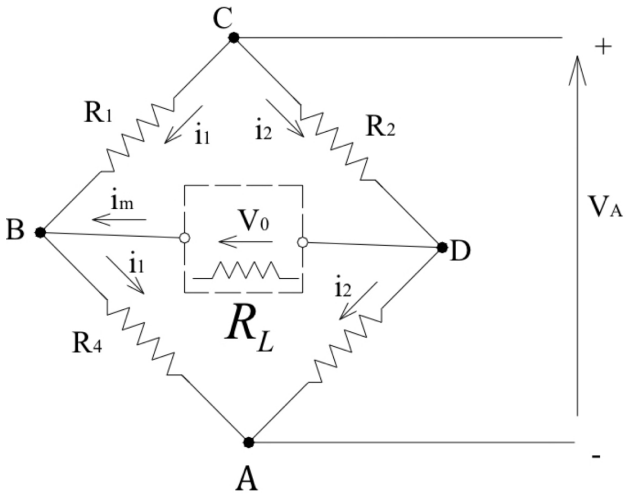


Figura 2.38: Ponte di Wheatstone

In generale, un ponte di Wheatstone (fig. 2.38)è una configurazione di resistenze disposte "a diamante". Due terminali sono alimentati da una tensione (VA) e se si verifica una variazione di una o più resistenze variabili, si ottiene variazione di tensione in uscita (V0). Inoltre, a seconda del numero di resistenze variabili disponibili, la tensione in uscita può assumere valori diversi. Generalmente, con quest'ultime si intendono degli estensimetri, ovvero dei dispositivi che se deformati, forniscono in uscita una variazione di resistenza.

Il modulo di condizionamento della LC (Load Cell) che riceve in ingresso il segnale è il NI 9218.

2.5.3 Sensore di velocità

Figura 2.39: Sensore di velocità. Fonte: [22]

Il sensore in questione è un prodotto della Trans-Tek (fig. 2.39), il quale si trova all'interno dell'attuatore idraulico e si occupa di misurare la velocità dello stelo. Il modello è il 0114-0001.

Generalmente un LVT è composto da un sensore di posizione e da un circuito elettronico di elaborazione del segnale. Il sensore di posizione può essere costituito da un encoder ottico o magnetico, un sensore ad effetto Hall, un trasduttore di posizione lineare, o un altro dispositivo in grado di rilevare il movimento lineare. Il suo funzionamento si basa sul fatto che il sensore di posizione rileva la posizione dell'oggetto in movimento rispetto al tempo. Grazie a ciò ricava la velocità instantanea.

Il condizionamento è offerto dal modulo NI 9205.

Caratteristiche tecniche:

Caratteristica	Valore
Range di lavoro	100 mm
Massima velocità misurabile	100 mm/s
Guadagno	10 mV s/mm
Larghezza di banda (-3dB)	400 Hz

Tabella 2.16: Caratteristiche del sensore di velocità

2.5.4 Encoder incrementale

Questo strumento permette di misurare la posizione angolare dell'elemento a gomito, in funzione della quale è possibile determinare la posizione dell'attuatore pneumatico, sia di quello elettromeccanico. Il prodotto è dell'azienda Renishaw.

L'anello (ReSM, fig. 2.40) è realizzato internamente in acciaio inossidabile, dove nel suo perimetro sono state ricavate due tracce da 20 micrometri, corrispondenti ai due segnali in uscita dal lettore ottico, ovvero A e B. Sulla circonferenza sono presenti 18000 tacche, con una risoluzione di 0.02°. Inoltre, è presente un'ulteriore tacca di zero ottico, allineata con uno dei fori non filettati.

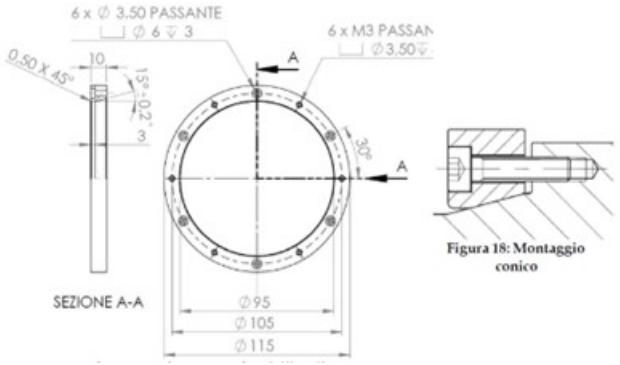


Figura 2.40: Anello ReSM. Fonte: .

Il lettore ottico VIONiC, invece, si occupa di andare a scansionare le tacche, dalle quali poi ricavare un'onda quadra che verrà successivamente elaborata. Il modulo che si occupa di acquisire i dati è il NI9401, della National Instruments.

Caratteristiche tecniche (tab. 2.17):

Tabella 2.17: Specifiche dell'Encoder Ottico

Caratteristica	Valore
Codice	PN RESM20USA115
Diametro USA	115 mm
Numero di linee	18000
Optical reader	VIONiC
Risoluzione (X4 Interpolazione)	18" di grado

Un parametro molto importante, è quello della frequenza di clock, in quanto da essa dipende la massima velocità di rotazione che può avere l'anello, rispetto al lettore, senza che si verifichino delle perdite di informazioni. Infatti, questa frequenza, deve essere necessariamente maggiore della velocità di rotazione dell'anello, ma allo stesso tempo minore della frequenza di campionamento del modulo NI9401, il quale ha uno "switching

rate" di 9 MHz e lo si fa lavorare ad una frequenza di 8MHz. Con queste condizioni la massima velocità angolare ammessa è di 1147.6 rpm.

2.5.5 Sensori di Pressione

Ci sono tre sensori di pressioni nel blocco del HLA (fig. 2.41), i quali si occupano di misurare la pressione di alimentazione in ingresso valvola, a monte dell'attuatore e la pressione nelle due camere del cilindro idraulico. I trasduttori in questione sono GEFRAIN TPSA.

Caratteristiche tecniche (tab. 2.18):

Caratteristica	Valore
Pressione massima misurabile	35 MPa
Sensibilità nominale	2.8 V/MPa
Larghezza di banda (-3dB)	160 Hz

Tabella 2.18: Specifiche del Sensore di Pressione

Figura 2.41: Sensori di pressione. Fonte:

2.6 Postazione utente

La postazione utente (2.42) è il centro di controllo dell'operatore. Sono presenti due monitor da 25", un pc collegato con la C-Rio, un pulsante di emergenza generale.

Figura 2.42: Postazione utente

2.7 Introduzione al software LabView

National Instruments LabVIEW (abbreviazione di Laboratory Virtual Instrumentation Engineering Workbench) è un ambiente di sviluppo software grafico, progettato per l'automazione, il controllo e l'acquisizione di dati. In parole semplici, è uno strumento che consente agli utenti di creare programmi mediante l'assemblaggio di blocchi grafici anziché scrivere codice tradizionale.

Figura 2.43: Logo LabView. Fonte: [23].

La sua potenzialità risiede non solo nella maggiore versatilità nello scrivere un codice, ma anche nella sua capacità di integrare hardware diversi, consentendo agli utenti di controllare e monitorare una vasta gamma di dispositivi e strumenti da un'unica piattaforma. Questa flessibilità lo rende uno strumento potente per progettare e implementare soluzioni personalizzate per una varietà di compiti e applicazioni.

Il funzionamento standard di qualsiasi applicativo LabView si basa sulla creazione di un progetto (project), il quale racchiude, in funzione del lavoro che si sta effettuando, un certo numero di VI (Virtual Instruments). Quest'ultime sono gli elementi con cui l'utente interagisce per costruire il codice e sono composti da due parti: Block Diagram (fig. 2.44 a sinistra) e Front Panel (fig. 2.44 a destra). Il primo permette di costruire la struttura, le funzionalità e il comportamento del codice. Il secondo invece è l'interfaccia utente, ovvero quella che permette allo user finale di interagire con il software.

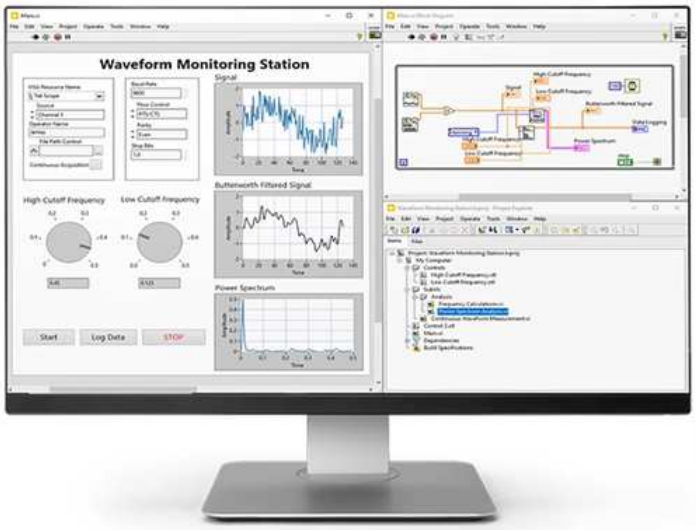


Figura 2.44: Interfaccia LabView (sinistra), schema a blocchi (destra in alto) e progetto (in basso). Fonte: [23]

2.7.1 Real-Time

LabVIEW Real-Time è un modulo aggiuntivo di LabVIEW progettato per lo sviluppo e l'esecuzione di applicazioni in tempo reale. Questo è particolarmente utile in contesti dove è necessario un controllo deterministico e affidabile dei sistemi, come nell'automazione industriale, nei sistemi di monitoraggio e controllo, nei test di dispositivi elettronici e nei sistemi embedded. Nasce, in particolar modo, per soddisfare due tipi di problemi:

- Risposta agli eventi: rispondere ad uno stimolo entro un ben determinato lasso di tempo;
- Sistemi di controllo in anello chiuso: elaborano continuamente un segnale di retroazione (feedback) con lo scopo di regolare il valore da assegnare alle variabili di output controllate;

Con determinismo si intende un sistema in grado di rispondere all'evento entro limiti di tempo ben definiti. Infatti questo concetto nasce con la necessità di avere dispositivi sempre più performanti e che raggiungano velocità di esecuzione molto elevate e in grado di compiere diversi task contemporaneamente. Quest'ultima caratteristica viene chiamata True Parallelism e consente alle VIs o alle singole funzioni (cicli while, cicli for ecc...) di essere eseguite simultaneamente su core o processori separati, invece che in modo sequenziale su un singolo core. Ciò permette di sfruttare appieno le risorse di elaborazione disponibili e di distribuire in modo efficiente il carico di lavoro.

I dispositivi su cui vengono eseguite queste architetture sono per esempio la CompactRio, le quali hanno un proprio sistema operativo. Quella utilizzata nel banco prove, per esempio, possiede NI Linux Real-Time, nel quale esegue LabVIEW Real-Time. Questi strumenti sono molto potenti e performanti e permetto di effettuare operazioni computazionali molto complesse. I PC con i quali comunicano, invece, sono utilizzati per l'interazione con l'utente e la programmazione del codice.

2.7.2 Fpga module

Una FPGA, acronimo di Field-Programmable Gate Array (Matrice di Porte Logiche Programmabili in Campo), è un tipo di dispositivo di calcolo elettronico che può essere programmato e configurato. Si differenzia dai circuiti integrati tradizionali perché anziché avere una funzionalità definita in modo fisso, le FPGA possono essere personalizzate dall'utente per svolgere una vasta gamma di funzioni specifiche. Sono composte da una griglia di blocchi logici configurabili e di interconnessioni programmabili, che consentono a chi la usa di creare circuiti digitali personalizzati. Uno dei software in grado interfacciarsi con questi dispositivi è proprio LabView, ma ne esistono anche altri, come: VHDL, Verilog, C/C++ con HLS (High-Level Synthesis) e altri ancora.

I vantaggi che portano questo modulo di Labview sono molteplici, tra cui:

- Possibilità di creare, via software, un hardware personalizzabile in funzione delle esigenze dell'utente;

- Affidabilità come un qualsiasi altro chip hardware;
- Velocità elevate, nell'ordine di un singolo clock;
- Determinismo, con risoluzioni nell'ordine dei nano secondi;
- Parallelismo, ovvero è in grado di eseguire blocchi contemporaneamente e non in serie, il che permette di ottenere velocità molto elevate;
- DMA FIFO, il quale permette di trasferire dati tra il codice target e l'host, in maniera rapida ed efficiente, attraverso l'utilizzo di un FIFO. Questa logica di comunicazione verrà approfondita in seguito.

2.8 Architettura del codice per il banco ASTIB

Nel caso del Banco Prova per Servocomandi di Volo ASTIB, il codice viene strutturato su più livelli. In particolar modo, si hanno tre VI principali che comunicano tra di loro ed ognuna con un compito ben preciso.

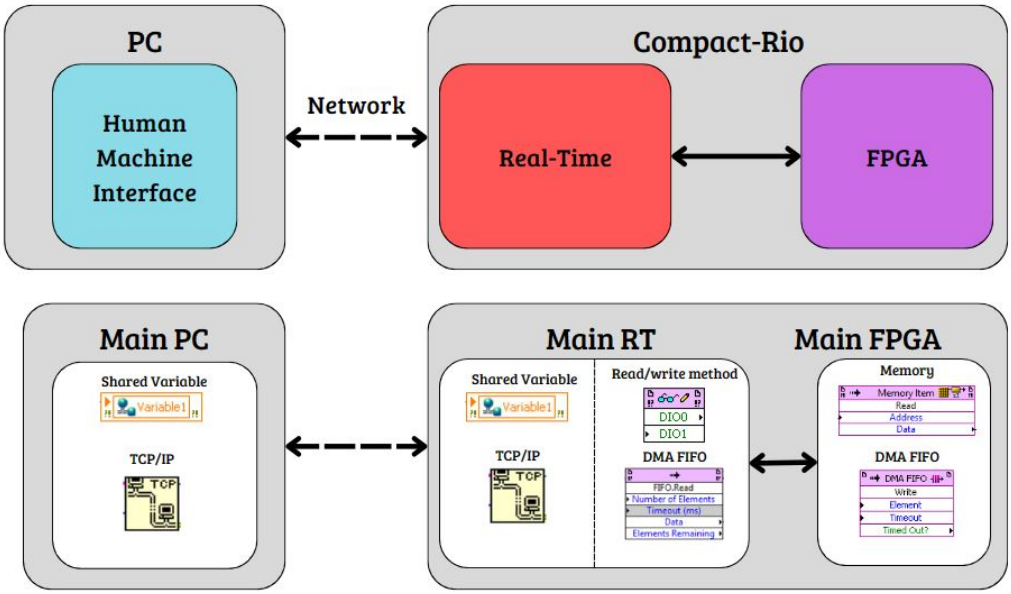


Figura 2.45: Comunicazione tra PC, Real-Time e Fpga.

Partendo dal basso della figura 2.45:

- Main FPGA: questo codice ha il compito di acquisire i dati attraverso i vari sensori presenti sulla macchina, elaborarli e chiudere gli anelli di controllo. Per quanto riguarda la comunicazione con il RT, si utilizza il protocollo DMA FIFO, il quale permette una comunicazione veloce e sicura. In questo caso, come negli altri, la programmazione del codice avviene tramite PC, ma l'esecuzione nel FPGA.
- Main RT: Si pone come intermediario tra il PC e la FPGA. Presenta un'interfaccia meno elaborata rispetto a quella del PC e viene utilizzata principalmente per le operazioni di debugging.
- Main PC: questo codice si pone al livello più alto di tutti ed ha il compito di visualizzare i dati, salvarli e naturalmente impostare i parametri di controllo. Si pone solamente come un codice di interfaccia e le informazioni al suo interno vengono trasferite attraverso i canali TCP/IP e Shared Variables.

Queste tre strutture verranno approfondite nei prossimi capitoli.

2.9 Interfaccia grafica del Software

Due sono le interfacce grafiche presenti in questo progetto, una del Main RT, una del Main PC. La prima risulta più essenziale in quanto viene utilizzata principalmente per le operazioni di debugging nel Real Time e nel FPGA. La seconda invece è più evoluta e permette all'utente di svolgere tutte le operazioni possibili. Essendo alcune interfacce ridondanti, verranno spiegate quelle del Main PC, quindi dalla figura 2.52.

Partendo dal **Main RT** si ha:

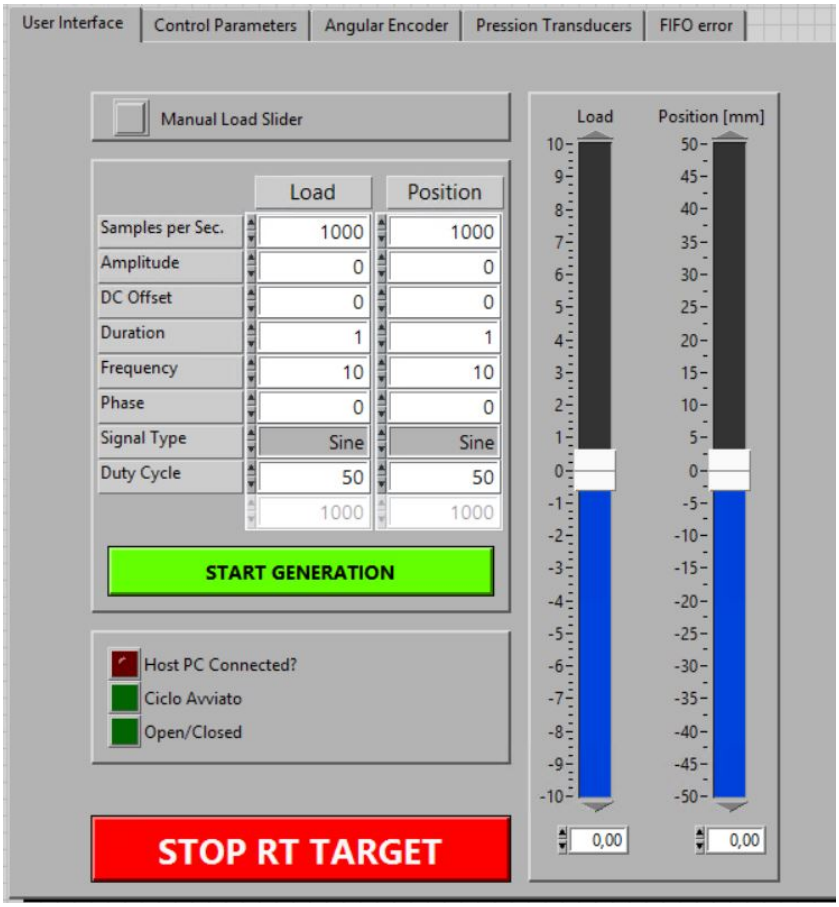


Figura 2.46: Interfaccia utente principale

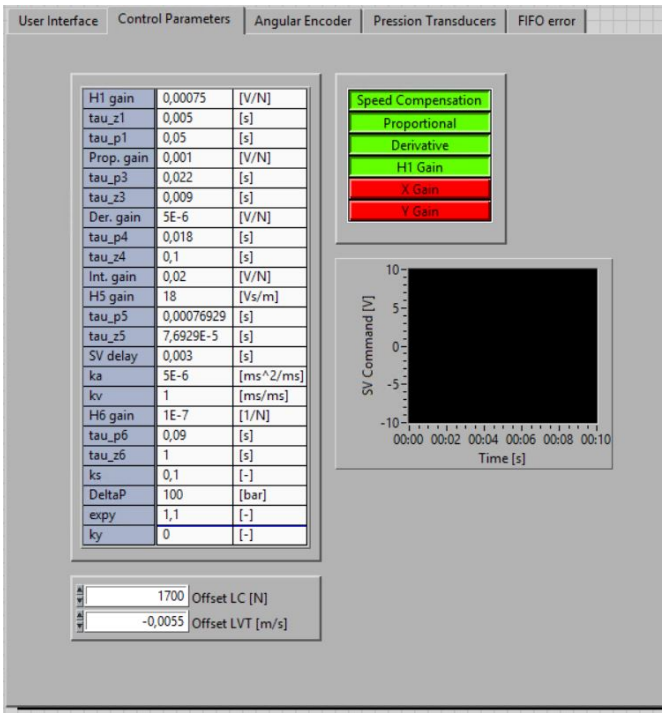


Figura 2.47: Parametri del controllo

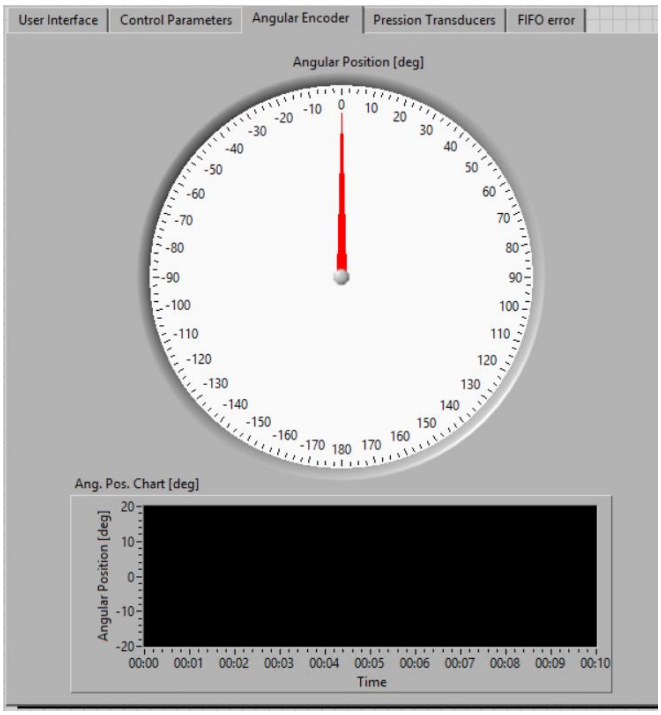


Figura 2.48: Indicatori dell'encoder

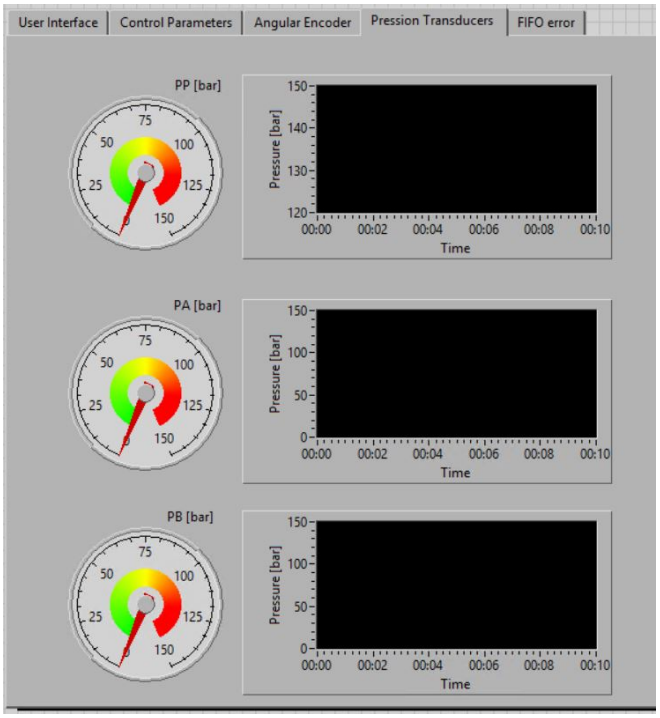


Figura 2.49: Indicatori di pressione

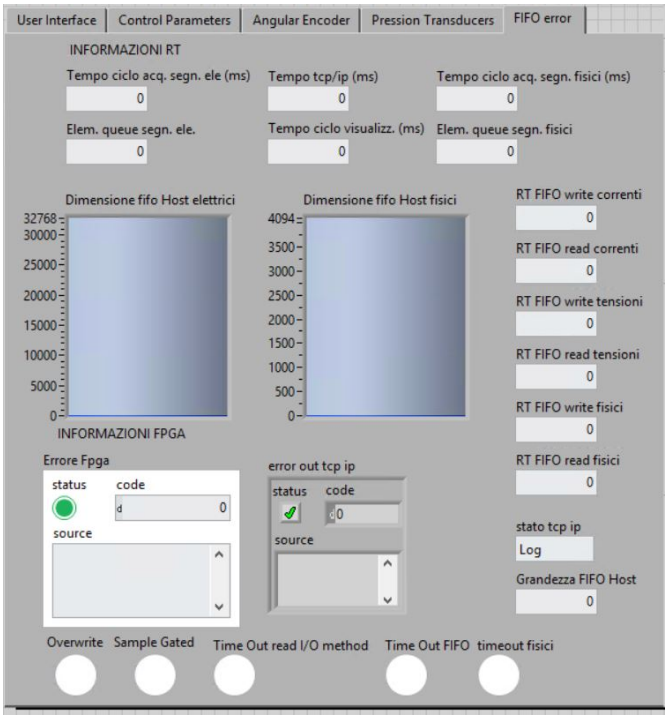


Figura 2.50: Pagina contenente i principali indicatori di performance e debugging

In particolare, dalla figura 2.50, si possono ricavare svariate informazioni riguardo le performances della Compact-Rio.

Partendo dall'alto:

- *Tempo ciclo acq. segn. ele. (ms):* tempo impiegato in una iterazione nel Timed Loop di acquisizione dei segnali elettrici;
- *Tempo tcp/ip:* tempo impiegato in una iterazione nel Timed Loop del TCP/IP;
- *Tempo ciclo acq. segn. fisici (ms):* tempo impiegato in una iterazione nel Timed Loop di acquisizione dei segnali fisici.

Dalla seconda riga:

- *Elem. queue segn. ele.:* il numero di elementi presenti nella Queue che porta i segnali elettrici dal loop di acquisizione al loop di sincronizzazione;
- *Tempo ciclo visualizzazione (ms):* il tempo ciclo del loop di sincronizzazione;
- *Elem. queue segn. ele.:* il numero di elementi presenti nella Queue che porta i segnali fisici dal loop di acquisizione al loop di sincronizzazione.

Al centro:

- *Dimensione fifo Host elettrici:* il numero di dati presenti nel DMA fifo Host dei segnali elettrici;
- *Dimensione fifo Host fisici:* il numero di dati presenti nel DMA fifo Host dei segnali fisici;

Alla loro destra:

- *RT FIFO write (read) correnti(tensioni/fisici):* l'indicatore del numero di elementi presenti nelle funzioni di scrittura e lettura del RT FIFO delle correnti, delle tensioni e dei segnali fisici;
- *stato tcp/ip:* indica lo stato attuale del case selector presente nel While loop del TCP/IP (*log, connected, close*);
- *Grandezza FIFO Host:* indica la grandezza massima del fifo host dei segnali elettrici.

In basso:

- *Error Fpga:* indica un errore proveniente dall'FPGA;
- *Error out tcp ip:* indica un errore proveniente dal tcp/ip;

I led:

- *Overwrite*;

- *Sample gated*;
- *Timeout del I/O method*;
- *Timeout FIFO*;
- *Timeout fisici*;

Questi ultimi verrano spiegati poi nel capitolo 4.1.

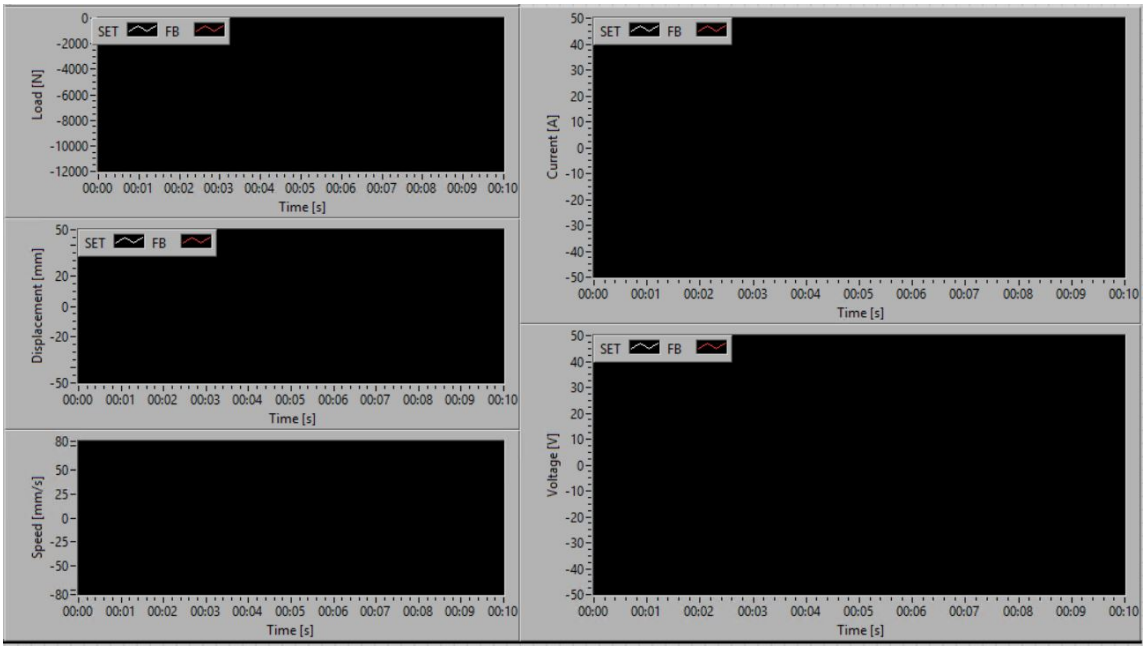


Figura 2.51: Grafici nel Main RT

Invece, nel **Main PC**, come già accennato, si ha un'interfaccia più completa, dove:

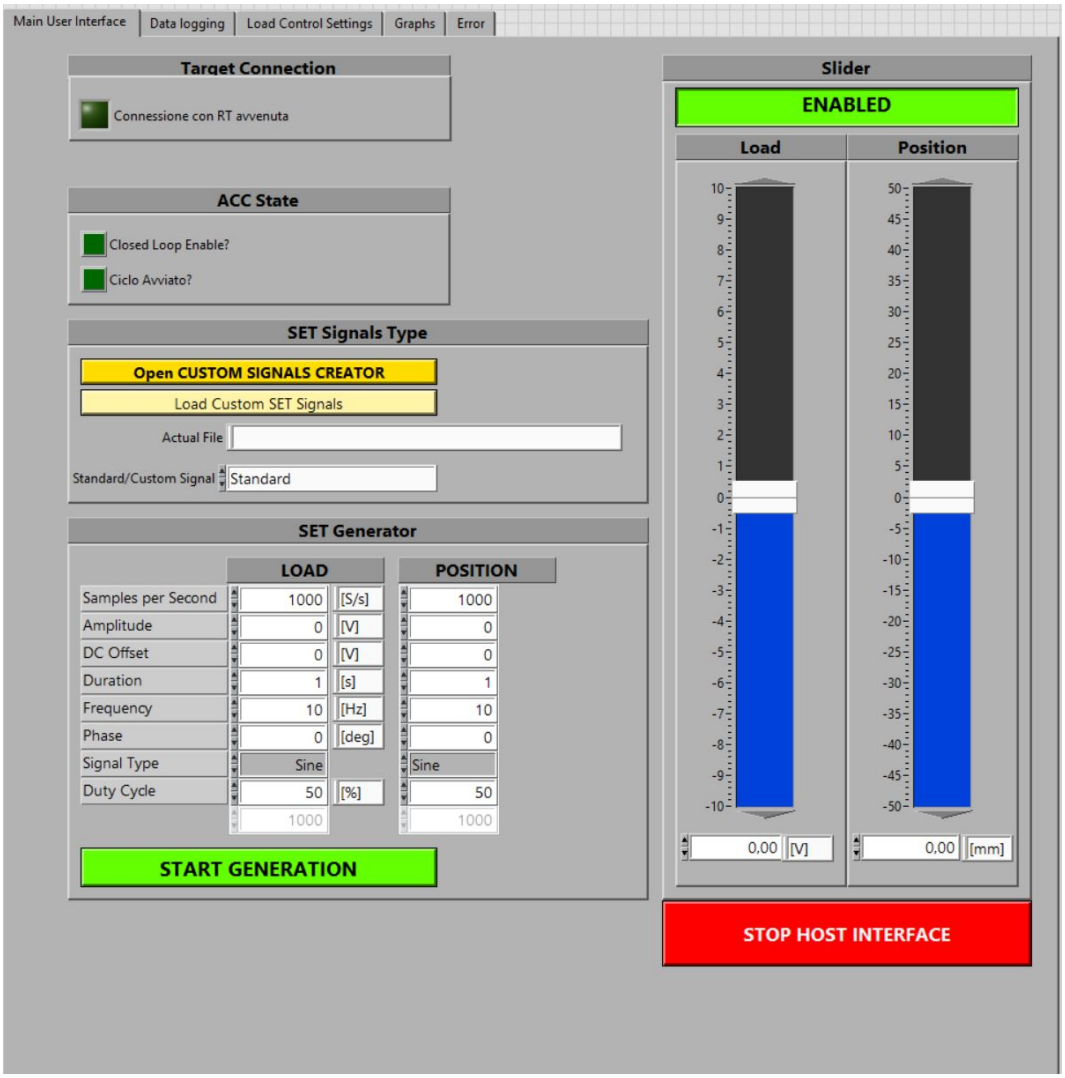


Figura 2.52: Schermata principale

- *Target Connection:* mostra se la connessione con il Main RT ha avuto successo;
- *Acc State:* ha due spunte, la prima si accende quando viene chiuso l'anello di corrente. Questa operazione viene effettuata girando una chiave nel ACC. La seconda spunta di Ciclo Avviato si accende quando viene premuto il rispettivo pulsante nel ACC, così facendo si iniziano a vedere i segnali reali provenienti da tutti i sensori;
- *Set Signals Type:* permette di creare dei segnali personalizzati di Set (sinusoidale, gradino, rampa e via dicendo) di Carico (N) e Posizione (mm) o di caricare

direttamente dei file esterni contenenti il fenomeno da riprodurre, come ad esempio delle perturbazioni o dei movimenti particolari.

- *Set Generator:* permette di modificare i parametri del set;
- *Slider:* permette di pilotare manualmente i due attuatori andando ad esercitare una determinata forza da applicare (per l'HIL) o una determinata posizione da mantenere (per l'AUT);;

In questo modo (fig.2.52) il compartimento idraulico può simulare le forze aerodinamiche che agiscono come disturbo sul motore lineare, così da riprodurre diversi scenari di volo.

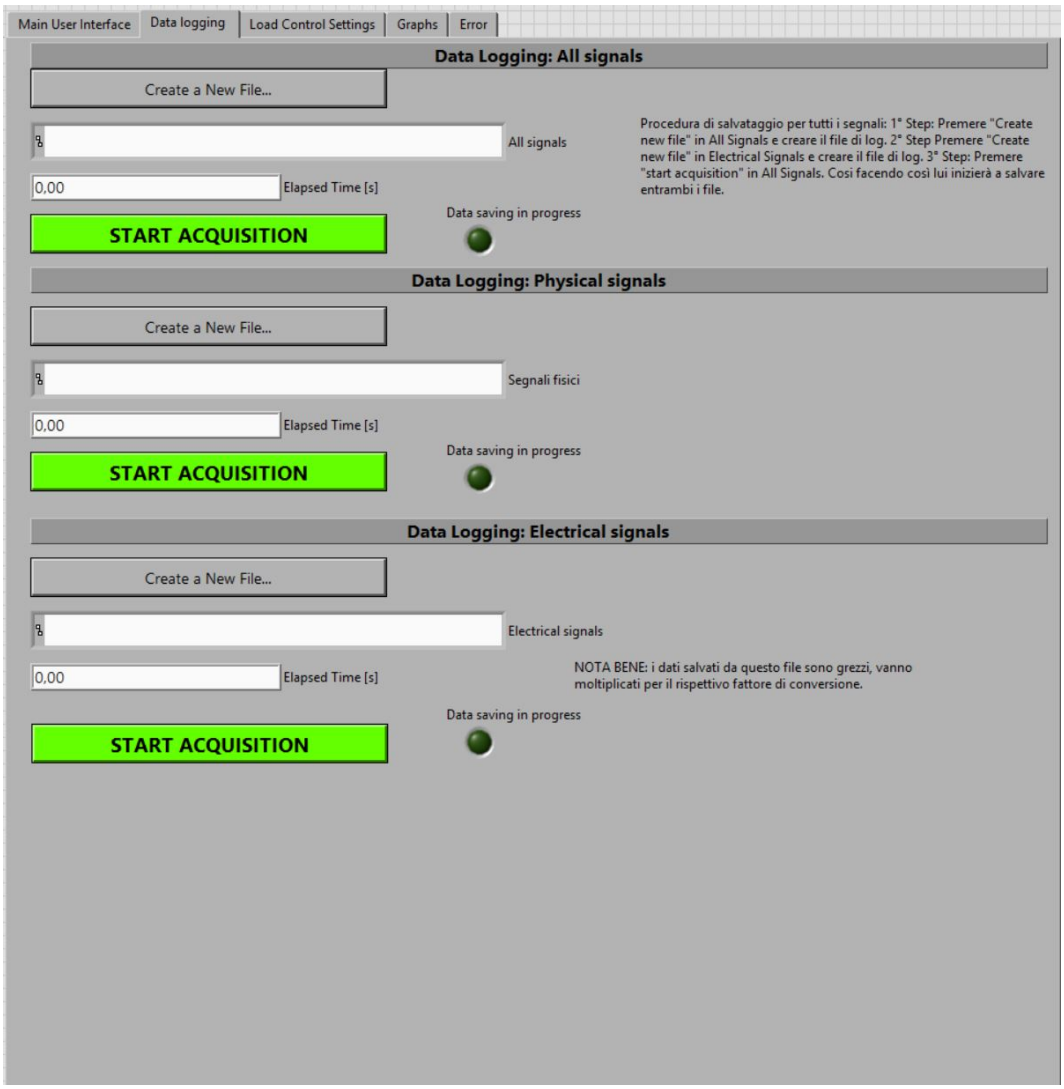


Figura 2.53: Schermata di log

Grazie alla figura 2.53 è possibile creare nuovi file e iniziare il salvataggio dati. La procedura è molto semplice: tramite il pulsante "Create a New File..." è possibile generare un nuovo file in formato ".tdms". Una volta creato, premendo "Start Acquisition" si inizia la procedura di salvataggio. Ripremendo il pulsante è possibile fermarlo.

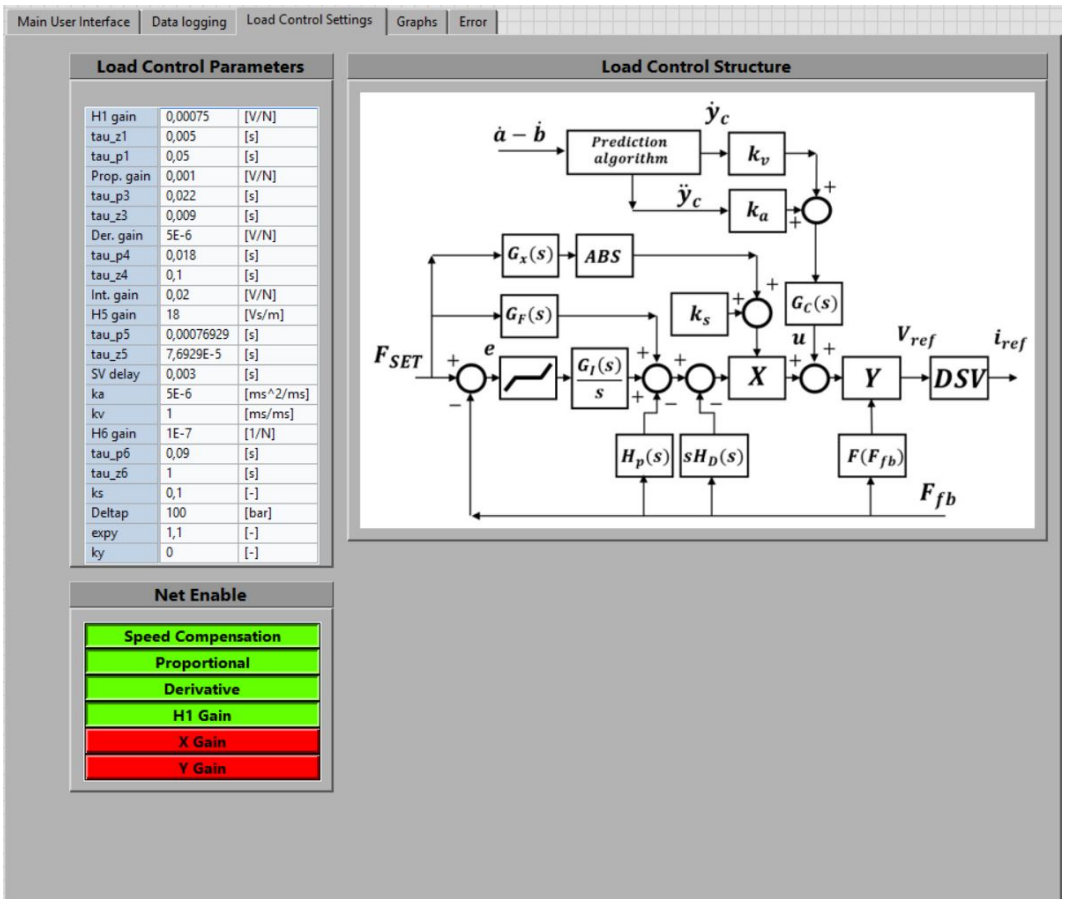


Figura 2.54: Schermata contenente le impostazioni del controllo

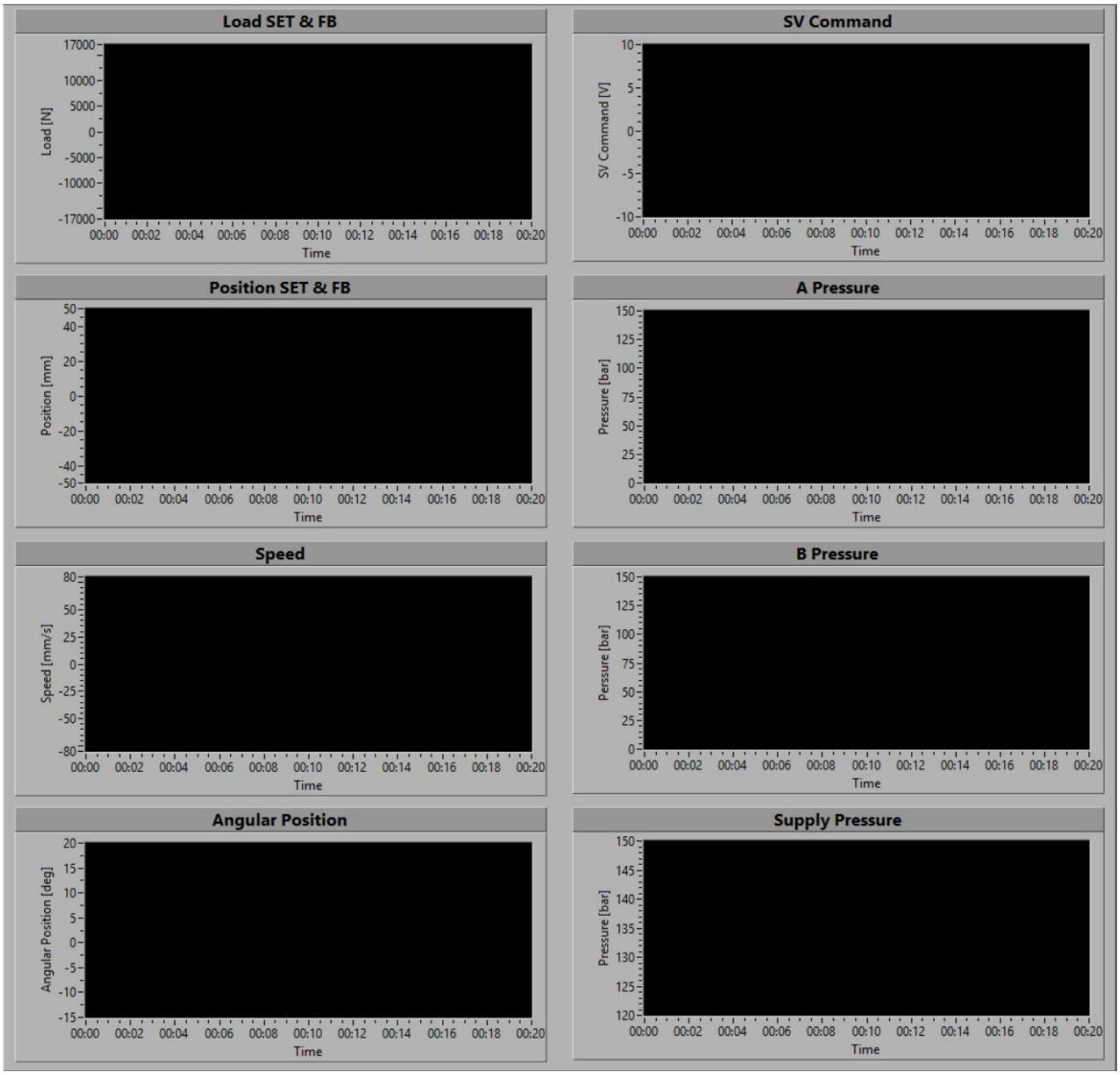


Figura 2.55: Grafici

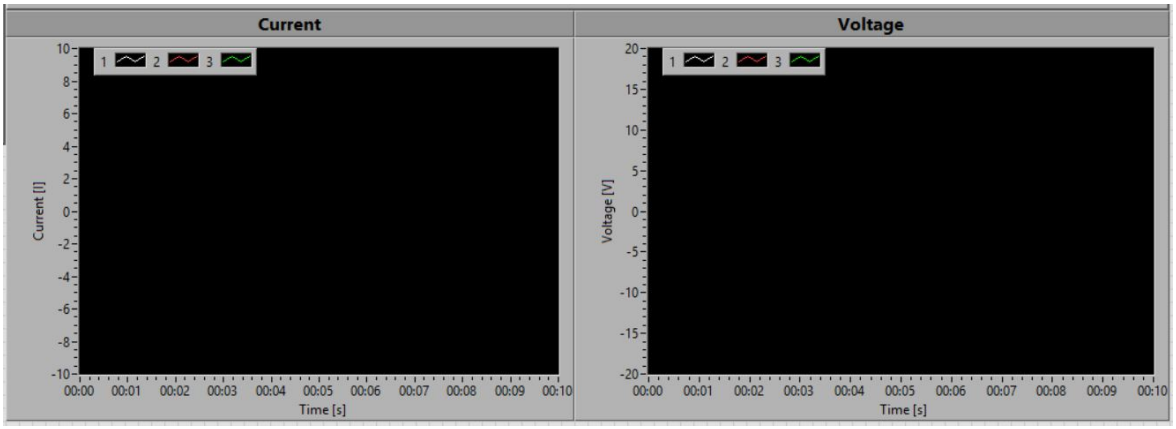


Figura 2.56: Grafici

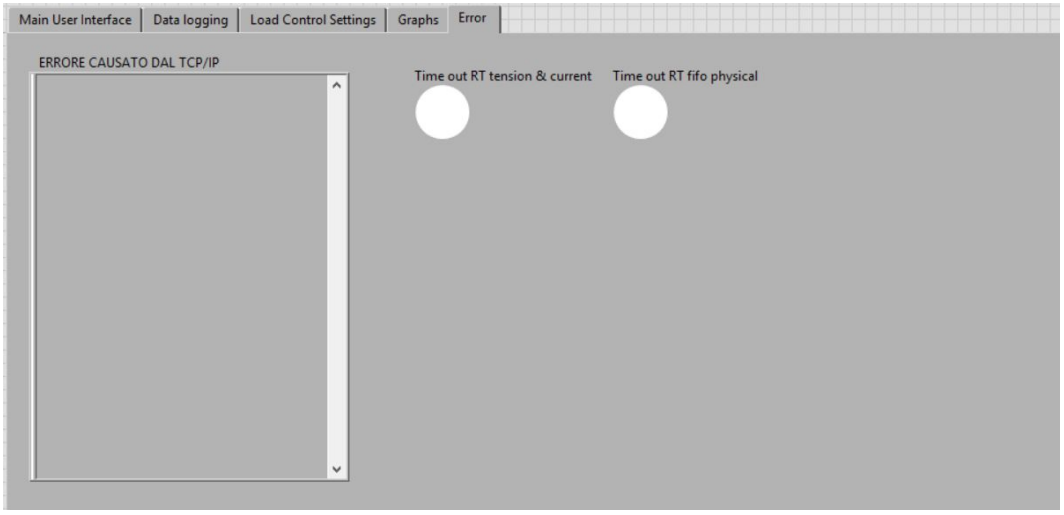


Figura 2.57: Schermata contenente le impostazioni di debugging

Rispetto al Main RT sono poche le informazioni sull'errore. Questo perchè ci sono meno punti "sensibili". Si può osservare una schermata di feedback dell'errore proveniente dal TCP/IP e due Led che indicano il Time out per i due RT FIFO.

Capitolo 3

Studio e progettazione del software

3.1 Punti di partenza

Prima di questo lavoro, l'ultimo aggiornamento software del banco prova Astib risale al lavoro di tesi di Romanini Riccardo, intitolato: "Implementazione hardware e software di nuove funzionalità del banco prova servocomandi di volo", 2021. Lo scopo principale di quest'ultimo è stato quello di implementare un codice LabView in grado di riuscire a visualizzare i segnali di tensione e corrente provenienti dall'attuatore elettromeccanico.

Nonostante la precedente attività abbia permesso di osservare per la prima volta tali segnali, sono stati rilevati alcuni punti da rinforzare, dai quali è poi nato questo lavoro di tesi, tra cui:

- La frequenza di campionamento desiderata del fenomeno fisico elettrico è di 1 Mhz, ma quella che si è riuscita ad ottenere è di 350 kHz;
- Dopo un determinato lasso di tempo di funzionamento del banco, il software si blocca e non permette più l'acquisizione dei dati;
- Numero di bit dei Fixed-Point, impostati all'interno del DMA FIFO, era sovradimensionato rispetto la risoluzione del modulo NI 9223;
- Implementazione di una procedura di interruzione sincronizzata dei cicli While più efficiente.
- Ricerca di nuovi protocolli di comunicazione e salvataggio che possano reggere un'elevata mole di dati.

In funzione dei problemi riscontrati nella precedente versione di codice, si è proceduto nel realizzarne una nuova che cercasse di ottimizzare i punti precedenti e portare nuove

funzionalità al banco.

Di seguito verrà prima mostrato nella sua interezza, per poi successivamente approfondire le parti che sono state le protagoniste di questo studio.

3.2 Analisi generale delle Main VI

In questo capitolo verranno mostrate le tre VI principali ed i While loop che le compongono, senza scendere nei dettagli, cosa che invece verrà fatta nel capitolo successivo per le parti di codice che sono stati principale oggetto di questo elaborato.

3.2.1 Main FPGA

Questo codice ha il compito principale di:

- Raccogliere i dati provenienti dai sensori;
- Pilotare la servovalvola in funzione dei parametri di controllo impostati;
- Gestire i vari segnali in ingresso/uscita;
- Eseguire il set di forza in funzione dei parametri di controllo.

I loop presenti all'interno di questa VI sono cinque :

Initialization, acquisition and control loop

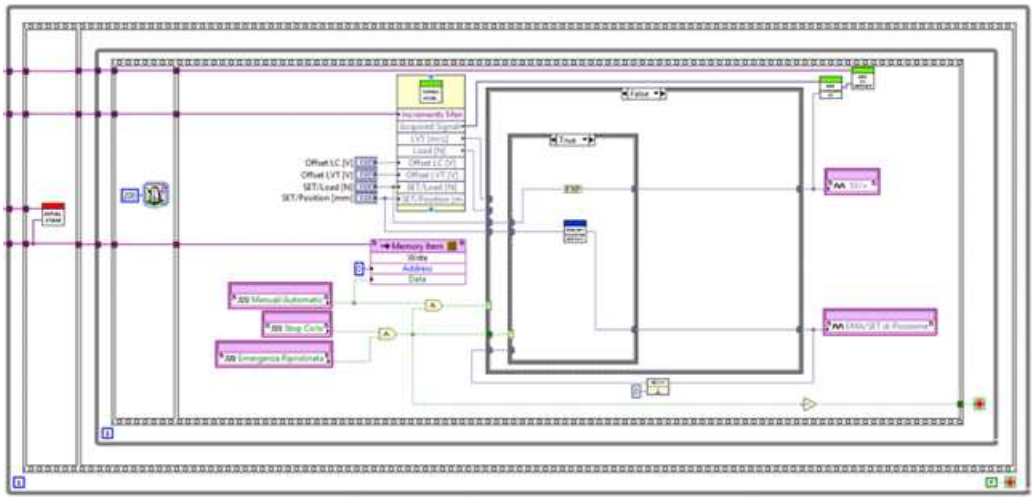


Figura 3.1: Initialization, acquisition and control loop

Questa parte di codice (fig. 3.1) si occupa di:

- Inizializzare le memorie;
- Acquisire i dati fisici provenienti dai vari sensori;
- Verificare lo stato delle emergenze;
- Inviare i segnali di Forza e Posizione all'EMA e alla servovalvola.

In particolar modo, la prima sequenza della figura 3.1, si occupa di gestire l'inizializzazione di tutte le memorie. Aprendo la Sub-VI al suo interno (immagine 3.2), si può notare la presenza di un *While Loop* al centro. Al suo interno si trovano due trigger, dove uno viene attivato quando vengono ripristinate le memorie e l'altro quando viene premuto il pulsante presente nel ACC, di Avvio Ciclo. Entrambi, permettono di uscire loop, ma fino a quando questo non si verifica, la memoria presente al suo interno, dalla quale passano tutti i segnali fisici, ha il compito di creare degli zeri ad ogni iterazione. Quindi i dati che vengono inviati al Main RT (e quindi anche al Main PC), fino a quando non vengono azionati quei due pulsanti, sono valori fittizi (zeri) che non provengono dai rispettivi sensori ma dalla memoria. Questa procedura viene fatta proprio per avere un'omogeneità nella rappresentazione grafica dei segnali e per evitare di vedere il rumore degli stessi quando ancora il ciclo non è stato avviato.

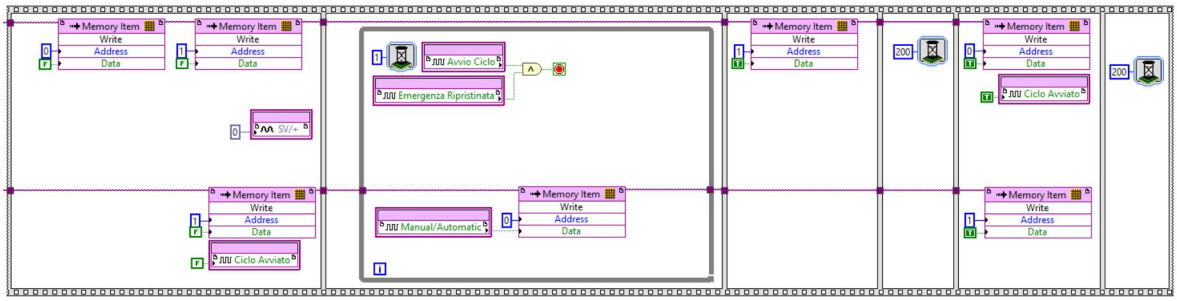


Figura 3.2: Initial stage

Naturalmente le proprietà della memoria (fig. 3.3) non permettono soltanto di generare delle costanti, come in questo caso, ma svariati tipi funzioni, come per esempio: delle rampe cresenti, decresenti, dei gradini, sinusoidi e via dicendo.

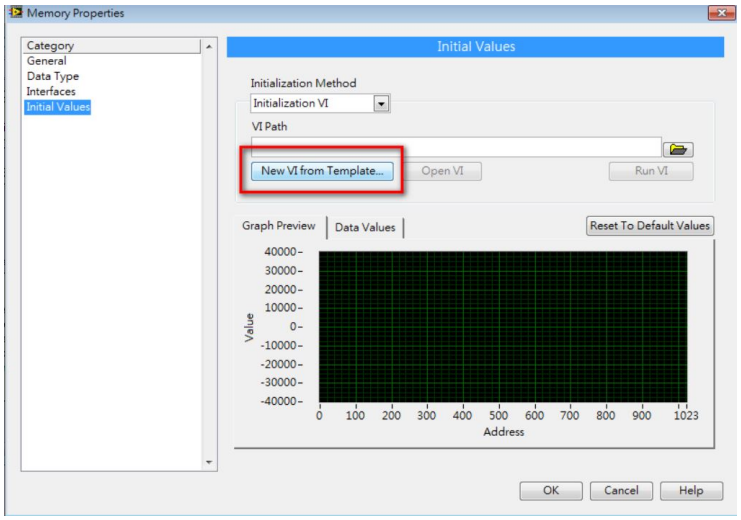


Figura 3.3: Proprietà memoria

Angular Encoder Reading

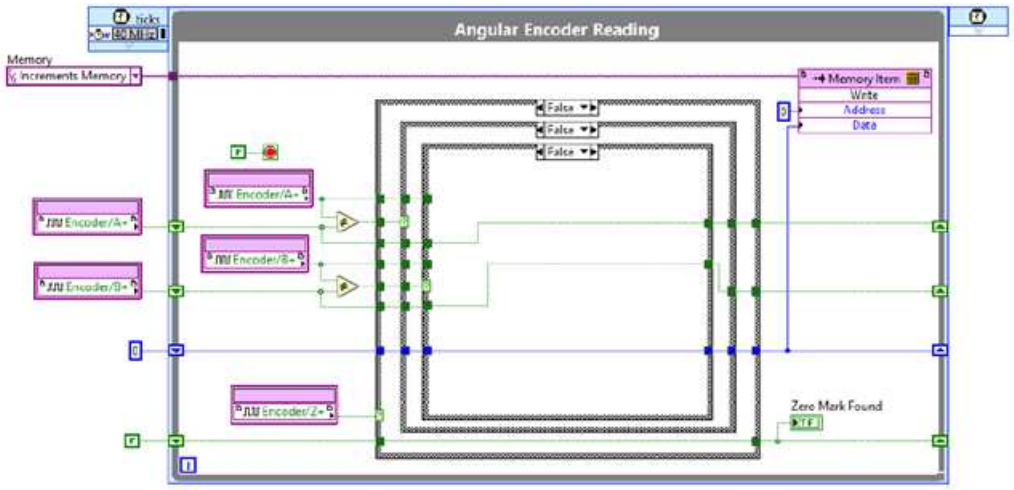


Figura 3.4: Angular Encoder Reading Loop. Fonte: Main FPGA VI.

Nella figura 3.4 è presente un Single Cycle Timed Loop (SCTL) [24], ovvero un particolare ciclo While. Infatti, è possibile impostare al suo interno una determinata frequenza di clock (in questo caso a 40 MHz, ovvero la frequenza di clock della c-RIO), obbligando la macchina a fare un'iterazione di questo loop ogni 25 ns, poiché vale la relazione:

$$t_{\text{ciclo}} = \frac{1}{40 \times 10^{-6}} = 25\,\text{ns}$$

Dall'encoder angolare provengono tre segnali digitali: A, B e Z. I primi due permettono di capire, in funzione della combinazione di 0 e 1, il verso di rotazione del giunto a gomito. L'ultimo segnale invece è il segnale di zero.

Da come si può osservare all'interno del ciclo, sono presenti: i generatori di segnale (a sinistra), i case selector al centro ed una memoria collegata (passando per uno shift register) con una variabile numerica intera (in blu). Quindi si ha che in funzione delle combinazioni di 0 e 1 (true e false) all'interno del ciclo la variabile numerica possa incrementare o diminuire. Nel primo caso significa che il giunto ruota in senso orario, nel secondo caso in senso antiorario.

Security and Booleans Management Loop

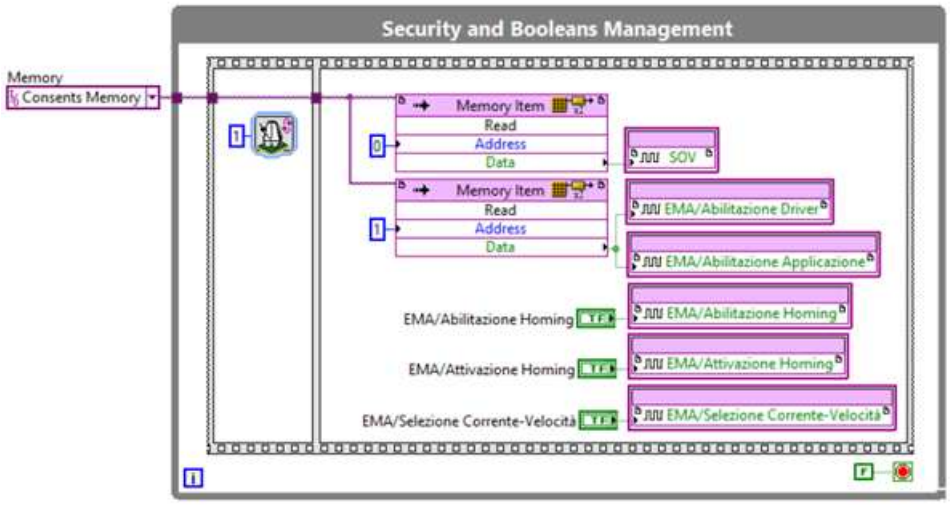


Figura 3.5: Security and Booleans Management Loop

In questo loop invece sono presenti le memorie con indirizzo “Consents Memory”, ovvero quelle che gestiscono i segnali di emergenza da inviare alla servo-valvola (SOV) o i segnali di abilitazione dei Driver dell’EMA e l’homing. Sono aggiornati ogni µs in quanto le operazioni di sicurezza necessitano una risposta molto rapida.

To RT Target Loop

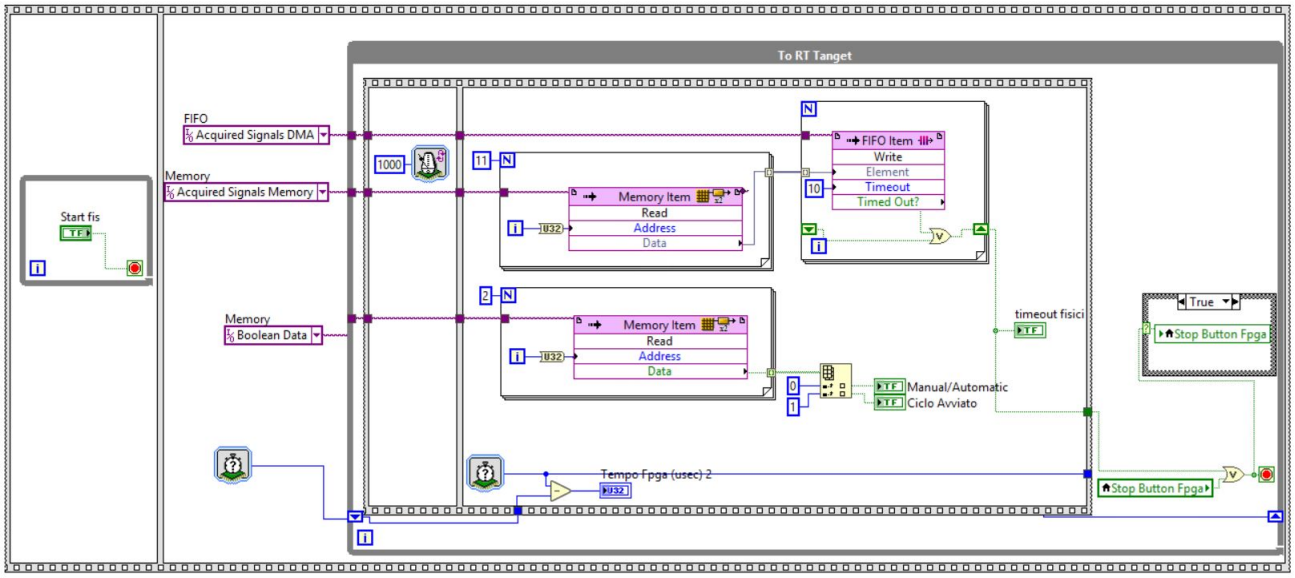


Figura 3.6: Security and Booleans Management Loop

Questo ciclo (fig. 3.6) , invece, è quello che si occupa di costruire il collegamento DMA FIFO con il Main RT VI ed inviare ogni 1 ms, 11 dati provenienti dai vari sensori che si occupano di raccogliere i segnali “fisici” e due segnali Booleani.

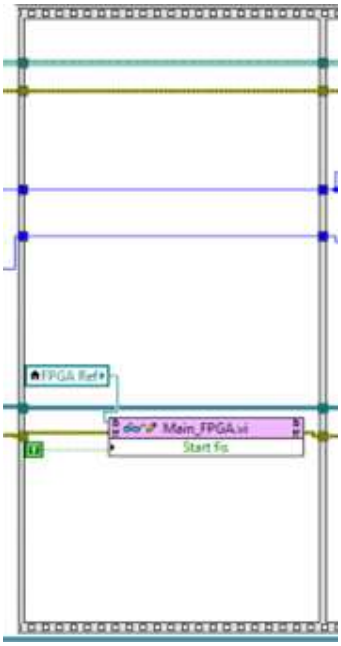


Figura 3.7: Sequenza che precede Acquired signals DMA reading, Main RT

Il loop, rispetto alla versione di codice precedente presenta:

- Ciclo While di inizializzazione: nella prima sequenza (figura 3.6) è presente un While loop. Questo ha il compito di aspettare la ricezione di un segnale booleano TRUE dalla Main RT VI (figura 3.7 a sinistra). Quando il dato arriva, viene superata la prima sequenza (3.6) e si inizia con il secondo While Loop. Questa tecnica viene usata per far si che il Real Time sia pronto a ricevere dati nel momento in cui FPGA inizia ad inviarli.
- Procedura di stop del ciclo: il codice della figura 3.6 può essere interrotto solamente se viene premuto il pulsante di Stop nel Real-Time oppure se si verifica un Timeout. La procedura utilizzata nel sincronizzare l'interruzione dei vari cicli verrà analizzata nel paragrafo 4.1.4.

To RT Target Voltage e Current

L'ultimo ciclo del FPGA (fig. 3.8) si occupa di inviare i segnali di tensione e corrente, attraverso un DMA FIFO, al Main RT. Essendo questo loop oggetto di studio di questa tesi, verrà approfondito nel prossimo capitolo.

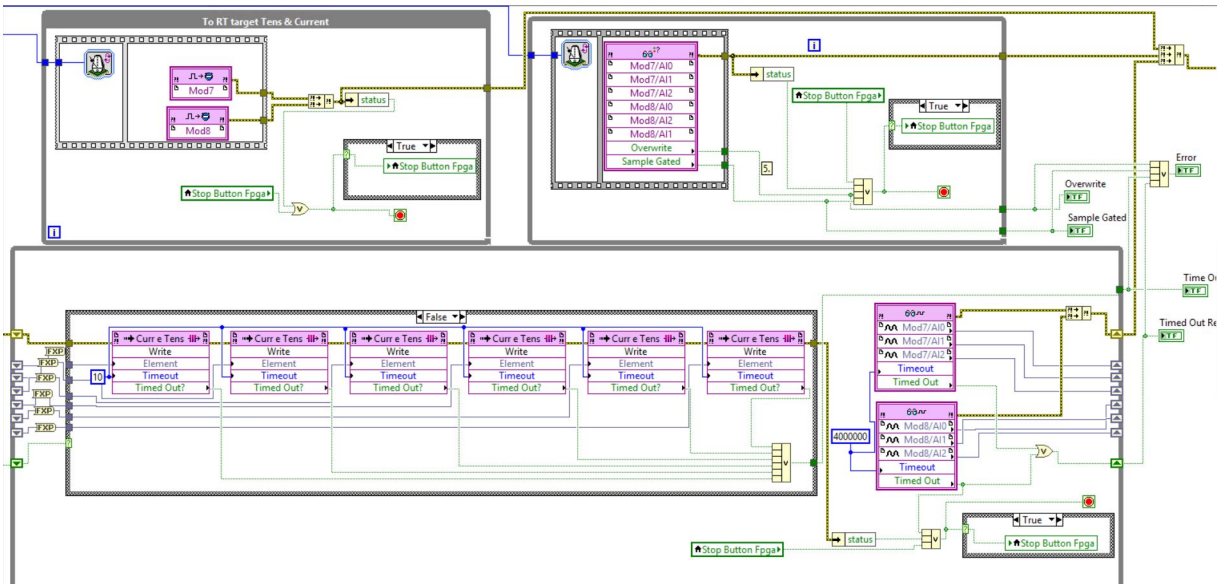


Figura 3.8: Ciclo di acquisizione dei segnali elettrici

3.2.2 Parentesi sul protocollo di comunicazione: DMA FIFO

Come accennato nei precedenti paragrafi, il DMA FIFO è un metodo di comunicazione in logica Fifo (first-in first-out) costituito da due buffer che si scambiano dati unidirezionalmente dal FPGA al RT. Osservando la figura 3.9, si può vedere come i dati in ingresso

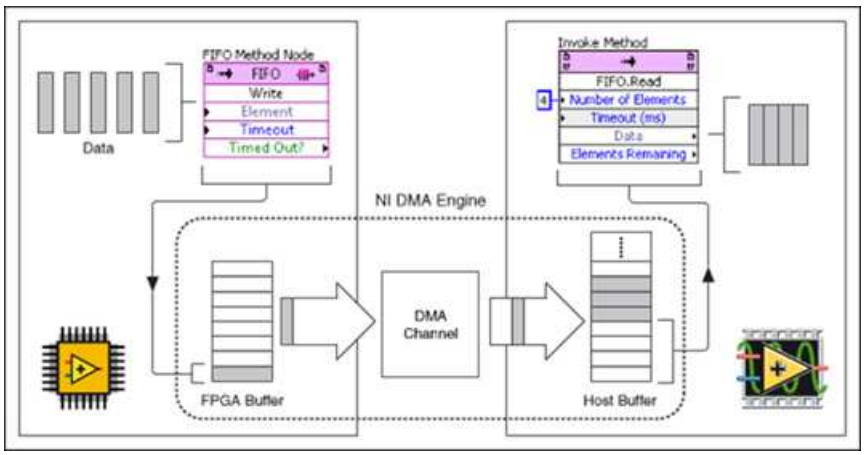


Figura 3.9: DMA FIFO. Fonte: [25]

(singoli elementi) vadano ad impilarsi uno alla volta all'interno del buffer FPGA ed escano con la stessa logica in un altro buffer, ovvero l'HOST (che in questo caso è nel Main RT VI).

Due sono i DMA FIFO utilizzati: uno per i segnali "fisici" ed uno per i segnali "elettrici" provenienti dall'EMA.

Per quanto riguarda la loro dimensione, attraverso le impostazioni di LabView è possibile configurare il numero di elementi che può contenere il buffer FPGA, mentre la dimensione dell'HOST verrà impostata automaticamente come il doppio di quest'ultima oppure a 10000 elementi (configurazione che si può modificare attraverso l'utilizzo di metodi nel Main RT VI). Riferendosi al buffer FPGA, per i segnali fisici è stato impostato il valore di 2048, mentre per gli elettrici si hanno 32768 (fig. 3.10) elementi (multiplo di 1024, dimensione minima del fifo).

Questi valori sono stati scelti in maniera tale da gestire al meglio i picchi di dati all'interno dei buffer, così da evitare il timeout.

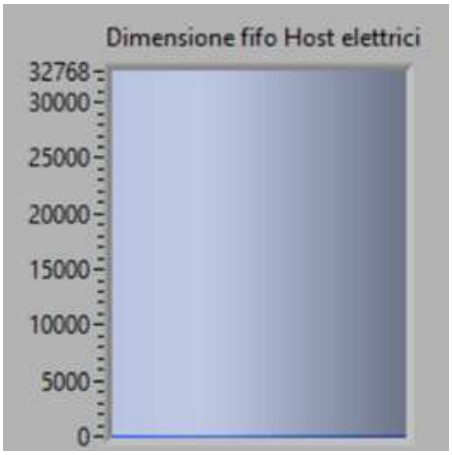


Figura 3.10: Buffer Host. Parametro che si può osservare nel front panel della VI.

Per gli elettrici la comunicazione FPGA-HOST avviene in questo modo: i sei canali provenienti dal modulo NI 9223, dove tre sono per le correnti e tre per le tensioni (essendo un motore trifase), leggono 6 samples ogni microsecondo, ovvero 6000 S/ms. Allo stesso modo, il codice Main RT VI, acquisisce 6000 S/ms, così facendo, il buffer non si dovrebbe mai riempire e mantenersi, in media, sempre a zero elementi (in quanto 6000 dati entrano e 6000 escono). Per cercare di rispettare la precedente condizione, si è inserito un Timed-Loop (nel Main RT), con un clock di 1 Mhz e un periodo di 1000 microsecondi.

La stessa cosa è stata fatta per i fisici, con la differenza che vengono letti 11 dati ogni millisecondo.

Inoltre, all'origine di questo lavoro di tesi, il DMA FIFO era stato impostato a 16384 elementi. Nonostante fosse una dimensione elevata, ogni tanto era solito manifestarsi un timeout che provocava il blocco della VI. La causa di tale fenomeno era dovuta alla presenza di picchi di dati all'interno del buffer che lo saturavano e ne provocavano l'arresto. Per ovviare quindi, si è raddoppiata la sua dimensione (32768) risolvendo così il problema.

Per quanto riguarda la configurazione del FIFO, in entrambi i casi si è utilizzato come formato i FXP (fixed-point). Infatti la FPGA deve necessariamente garantire velocità ed affidabilità nell'effettuare le operazioni e per farlo bisogna cercare di non esaurire le sue potenzialità. Per questo sono stati utilizzati, in quanto essendo un formato a virgola fissa, permette di mantenere sempre lo stesso numero di bit e ottimizzare le risorse.

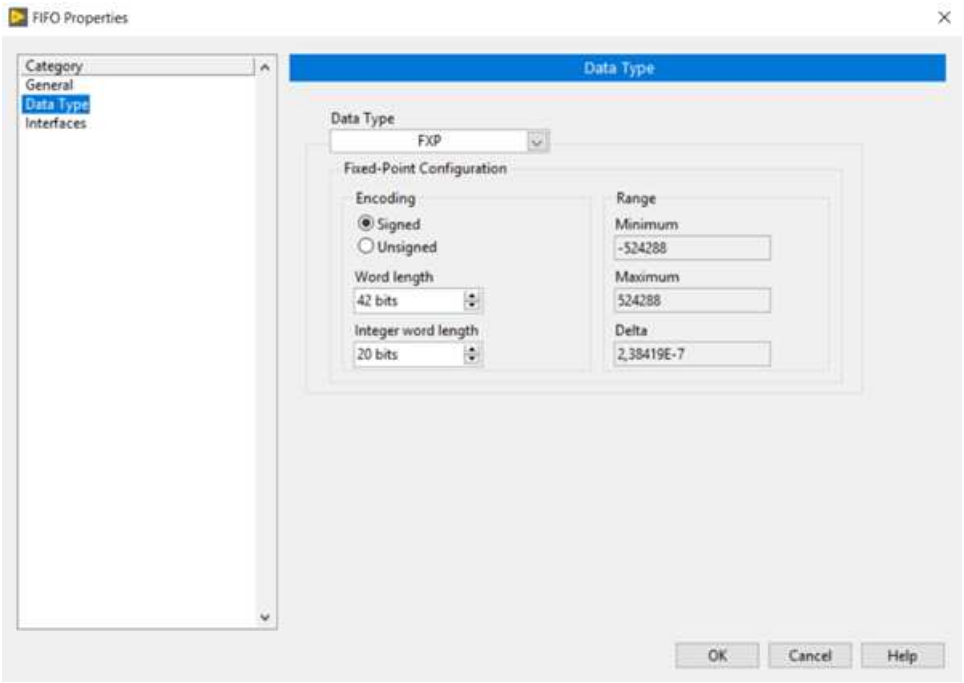


Figura 3.11: Impostazioni Fifo nella vecchia versione del codice

La vecchia versione (di Romanini Riccardo), presentava alcuni punti da migliorare sotto questo aspetti. Prima di tutto il FIFO era sovradimensionato (a 60000 elementi) rispetto alle reali necessità. In questa versione è stato abbassato a 32768. Inoltre, il FIFO era stato configurato per trattare FXP a 42 bit, invece che a 16, allocando così più spazio del dovuto. Invece, in questa versione, i segnali fisici sono stati lasciati invariati a 42 bit in quanto in esso convergono diversi tipi di moduli, che lavorano con diversi formati.

3.3 Main RT

Questa VI è formata principalmente da 9 While Loop, dove sono responsabili di:

- Inizializzare le variabili e stabilire la connessione con FPGA;
- Inviare e ricevere informazioni provenienti dal FPGA;
- Generare segnali di SET di posizione e forza;
- Inviare informazioni al MAIN PC;
- Offrire un'interfaccia utente per impostare i controlli e visualizzare i grafici;
- Sincronizzare i dati fisici con quelli elettrici.

È presente anche un loop di inizializzazione delle shared variables e della referenza con FPGA, che però non verrà mostrato.

Enable/Disable parameters Loop

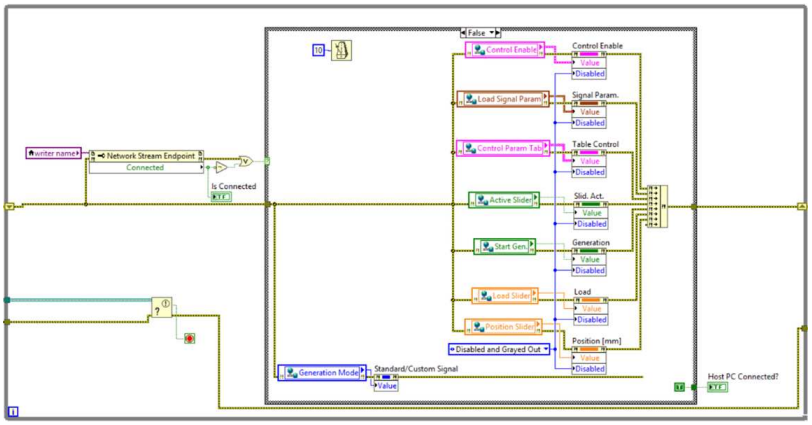


Figura 3.12: Enable/Disable parameters Loop, False case

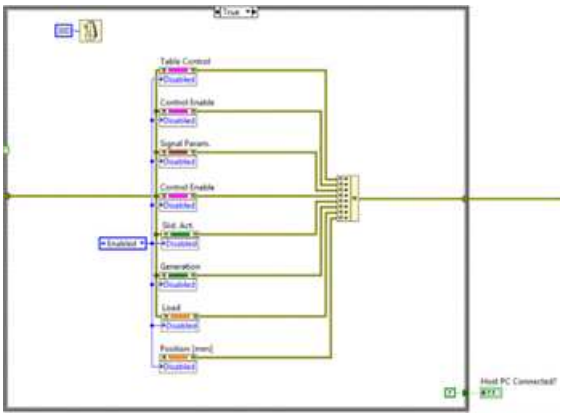


Figura 3.13: True case

Questo While Loop (fig. 3.12 e 3.13) viene utilizzato per abilitare o disabilitare tutti i parametri di controllo presenti nel front panel, nel momento in cui si effettua la connessione con il Main PC. La visualizzazione dei grafici nel Real-Time viene inibita dal loop di sincronizzazione.

Control Parameters Computation e Sensor Offset Correction Loop

Come da titolo, questo ciclo While (fig. 3.14) si occupa di inviare al FPGA i parametri del controllo e gli offset della cella di carico e del LVT.

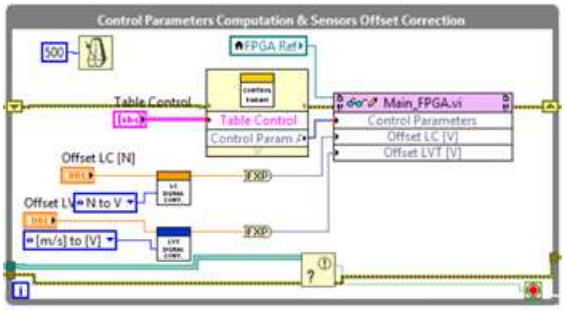


Figura 3.14: Control Parameters Computations e Sensor Offset Correction Loop

FPGA Booleans Management Loop

È un Loop (fig. 3.15) che si muove in sincronia con il precedente. Infatti, mentre il Loop antecedente si occupa di inviare i valori numerici al FPGA, questo trasmette le abilitazioni del controllo, quindi gestisce quali operazioni consentire o meno.

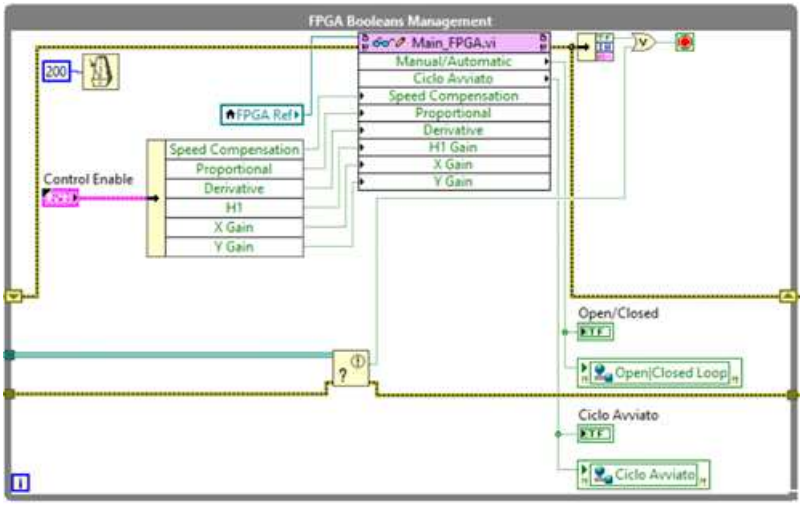


Figura 3.15: FPGA Booleans Management Loop

Generation Loop

È un Loop che si occupa di generare i segnali di SET di forza e posizione da inviare al FPGA. I segnali possono essere generati in due modi:

- Attraverso degli slider presenti nel front panel (fig. 3.16):

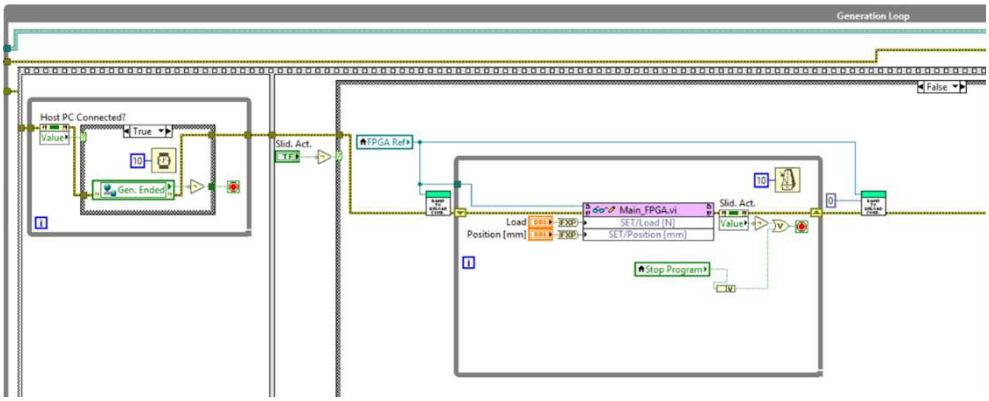
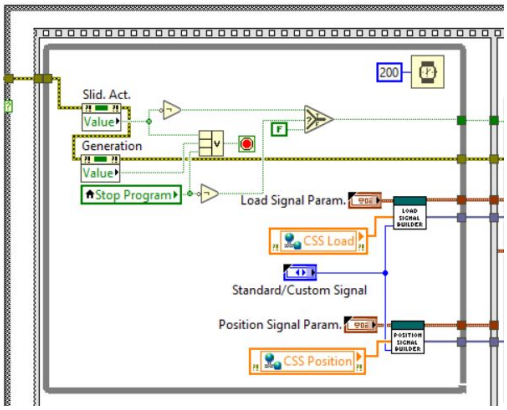
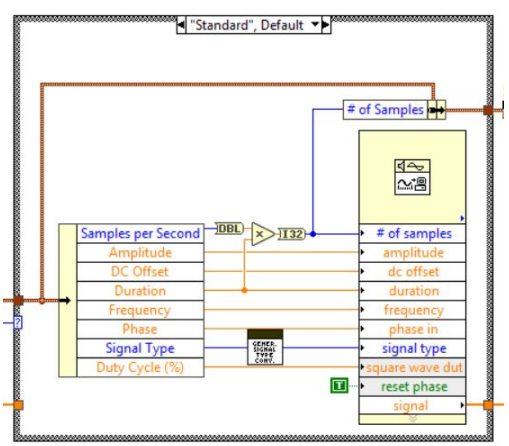


Figura 3.16: Generazione segnali attraverso slider

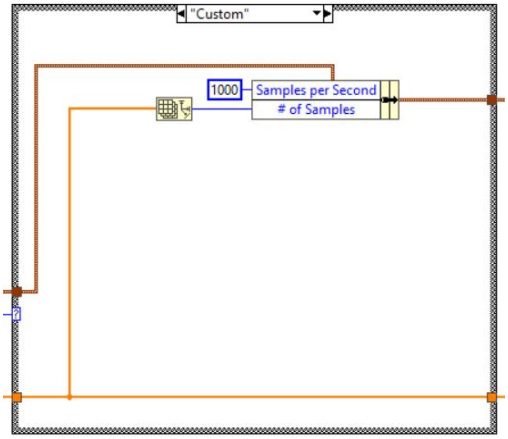
- Attraverso un segnale customizzato o standard (fig. 3.17):



(a)



(b)



(c)

Figura 3.17: Generazione di segnale customizzato o standard

Infine, è presente un Timed Loop (fig. 3.18) in modo tale da garantire a questo ciclo la precedenza rispetto agli altri e inviare con una determinata frequenza, i segnali al FPGA.

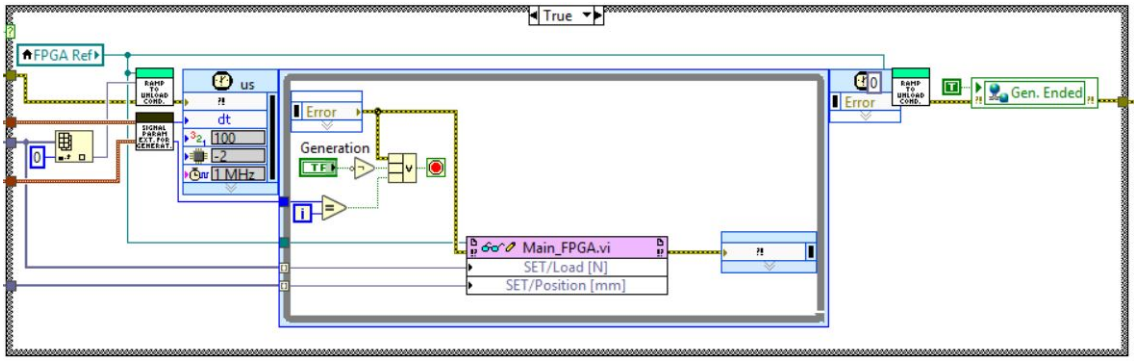


Figura 3.18: Invio segnali al FPGA

Data acquisition: voltages e currents

Acquisizione segnali tensione e corrente (fig. 3.19):

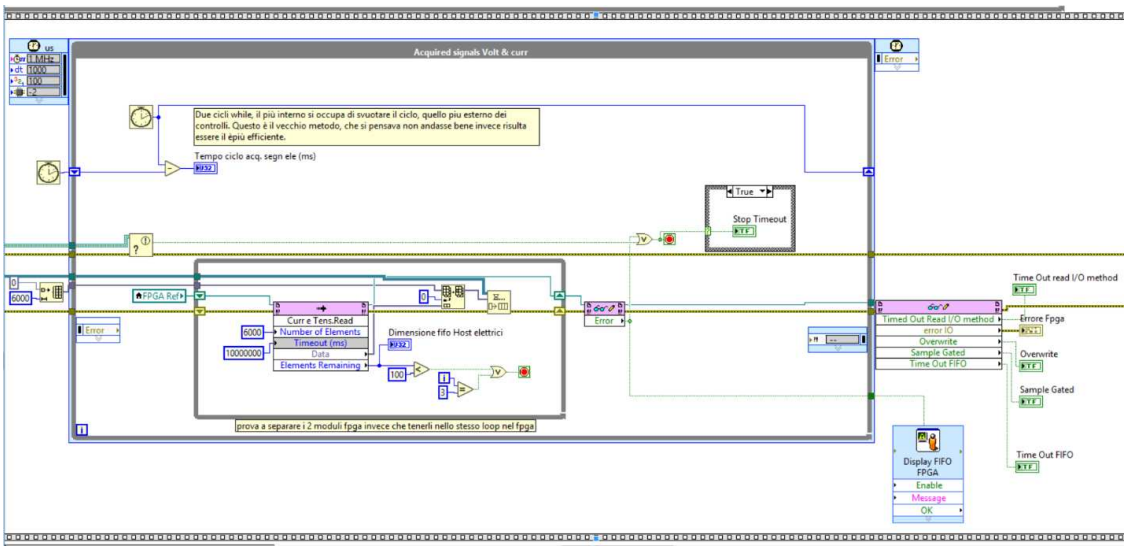


Figura 3.19: Ciclo di acquisizione di tensioni e correnti

Data acquisition: physicals

Acquisizione segnali fisici (fig. 3.20):

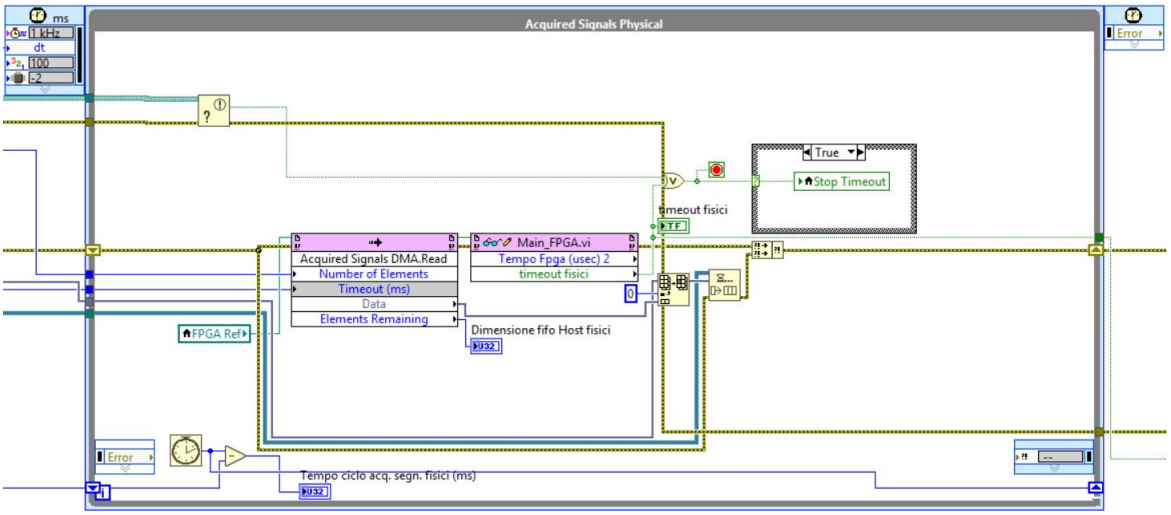


Figura 3.20: Ciclo di acquisizione dei segnali Fisici

Synchronization Loop

Loop di sincronizzazione (fig. 3.21):

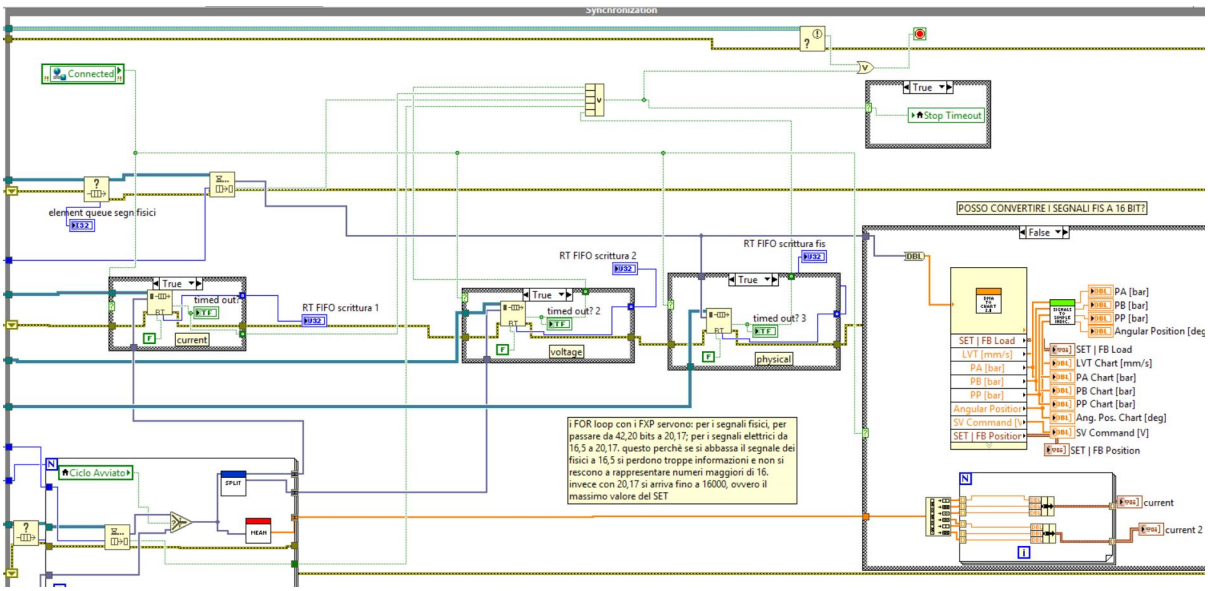


Figura 3.21: Synchronization Loop

TCP/IP: RT to PC

Loop di comunicazione RT-PC (fig. 3.22): Questi ultimi quattro loop verranno approfon-

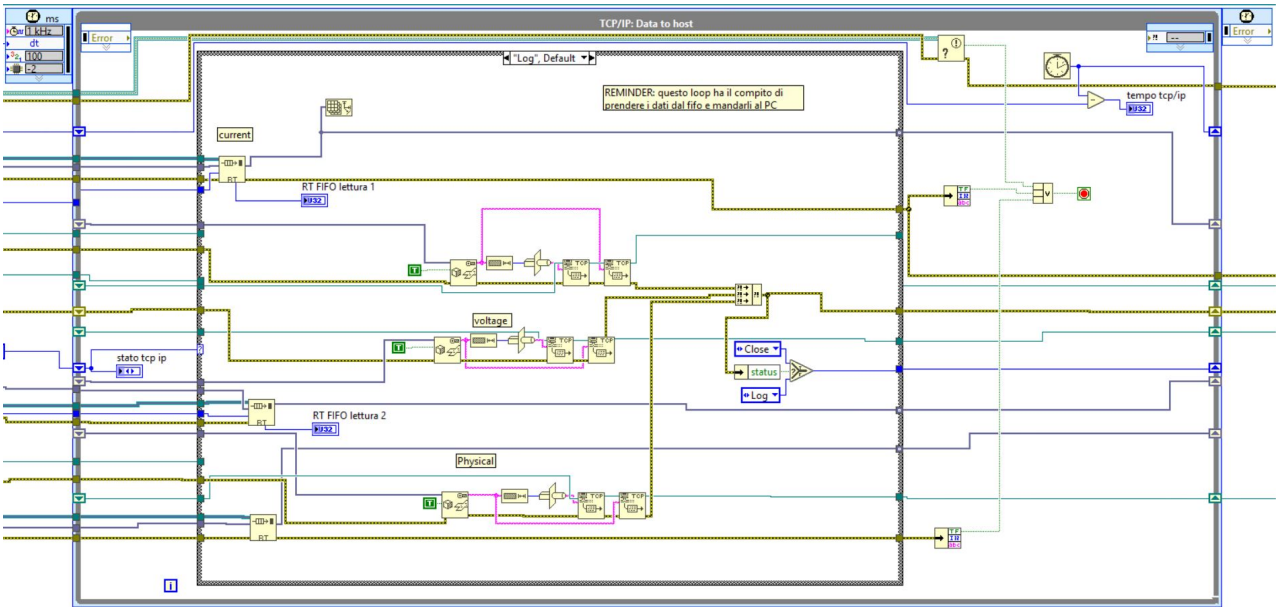


Figura 3.22: TCP/IP: RT to Pc

diti nel capitolo 4, in quanto oggetto principale di questa tesi.

3.4 Main PC

Il Main PC è la VI che permette di gestire controlli, acquisire e salvare dati direttamente all'interno del PC dell'utente. È composta principalmente da una parte di inizializzazione delle *Shared Varibles*, una di inizializzazione del collegamento tramite *TCP/IP* e sei While Loop. In particolar modo permette di:

- Delocalizzare il controllo dalla Compact Rio al PC;
- Visualizzare i grafici di tutti i segnali;
- Salvare porzioni di grafico che si ritengono rilevanti;
- Generare segnali di Set customizzati;

Di seguito, una breve descrizione dei vari cicli:

Main Event Case Structure e Update Graphs

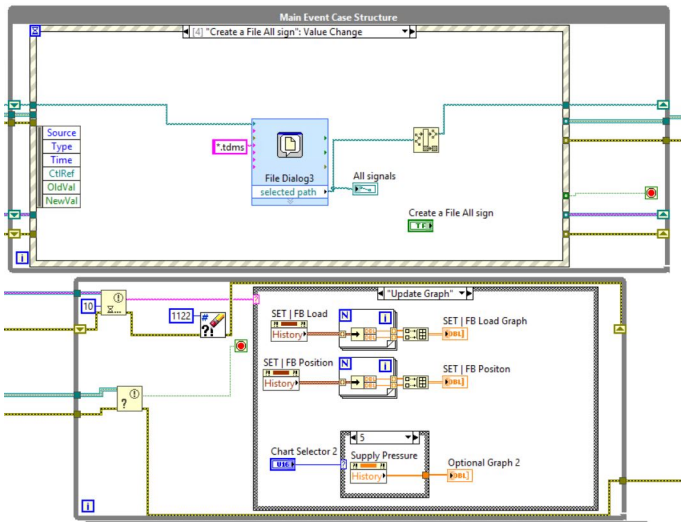


Figura 3.23: Main Event Case Structure e Update Graphs

Il primo ciclo in alto (fig. 3.23) è composto da un event structure, il quale si occupa di determinate operazioni, tra cui:

- Stop Program: Ovvero lo stop generale di tutti i loop della VI;
- Update Graphs: Si occupa di acquisire una porzione di grafico e permette anche il suo salvataggio;
- Creare un File per il salvataggio dei segnali fisici;
- Creare un File per il salvataggio dei segnali elettrici;

- Creare un File per il salvataggio dei segnali fisici ed elettrici;
- Creare dei segnali "customizzati"
- Importare da File dei segnali "customizzati"

Il secondo ciclo ha il compito di eseguire il secondo punto "Update Graphs".

Shared Variables to RT Main

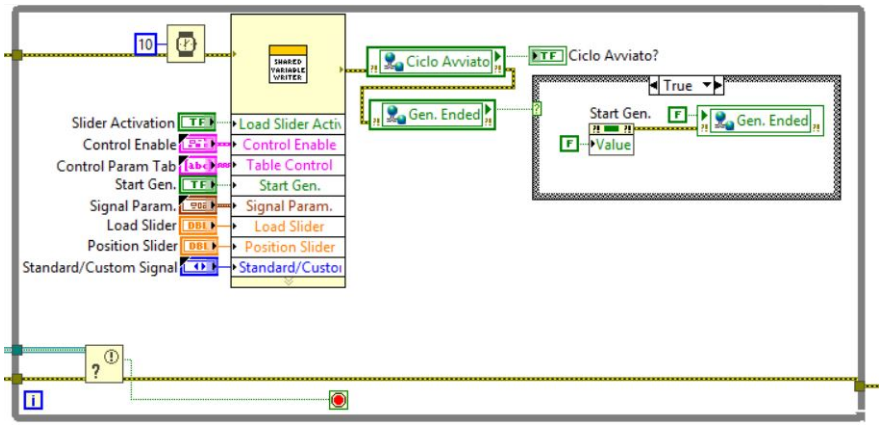


Figura 3.24: Shared Variables to RT Main

Questo ciclo (fig. 3.24) si occupa di inviare tramite shared variable i parametri di controllo e vari segnali al Main RT, con un timing di 10 ms per ciclo.

Update Units Loop

Si occupa di aggiornare le unità di misura del Set Generator e degli slider (fig. 3.25).

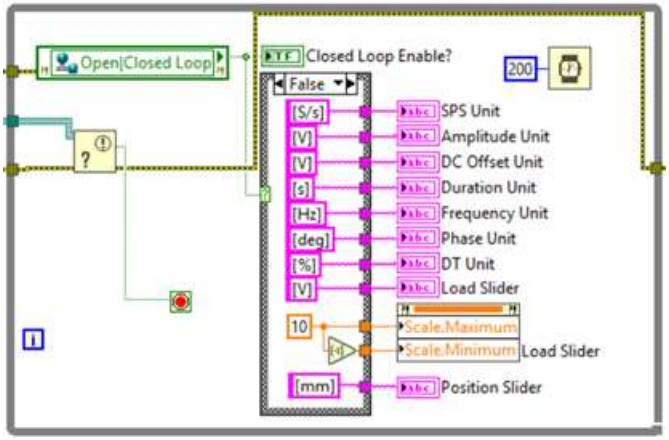


Figura 3.25: Update Units Loop

TCP/IP

I dati vengono prelevati dal TCP/IP e immessi in un RT FIFO per poi mandarli ai grafici e al salvataggio (fig. 3.26).

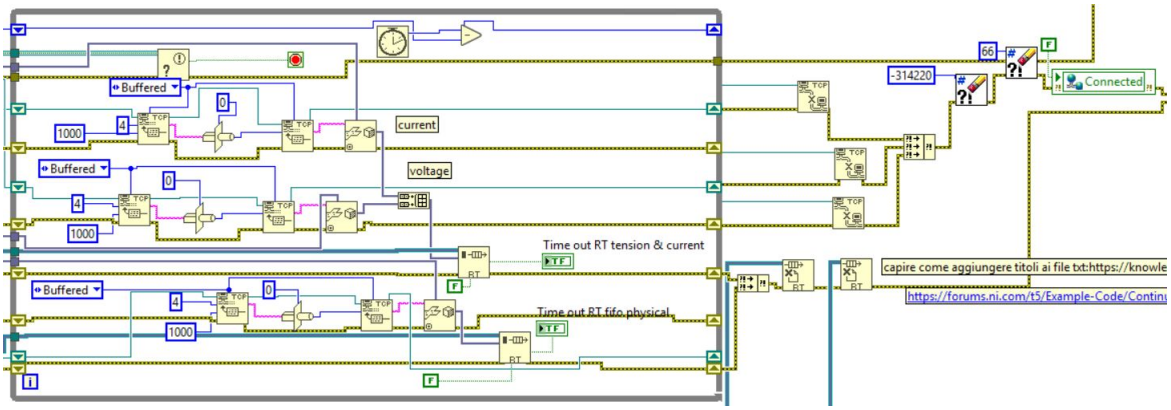


Figura 3.26: TCP/IP: Reader

Data logging and Graphs

Dal RT Fifo i dati vengono presi e mandati al plot e al salvataggio (fig. 3.27).

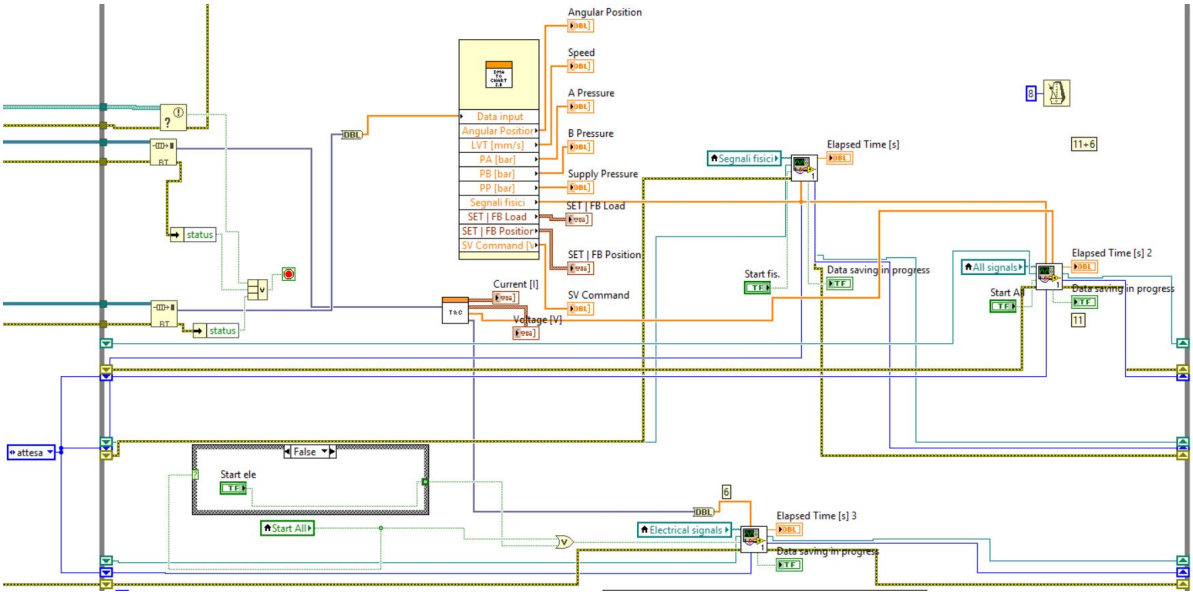


Figura 3.27: Data logging and graphs

Capitolo 4

Approfondimento sulle VI principali

In questo capitolo verranno analizzate le VI protagoniste di questa tesi e tutte le considerazioni effettuate durante la progettazione del software.

4.1 Acquisizione dei segnali di tensione e corrente a livello FPGA

Come precedentemente accennato, il Loop "To RT Target Voltages e Currents", analizzato nel precedente capitolo, si occupa di prendere i dati provenienti dai moduli NI 9223 e inviarli al Main RT VI attraverso un DMA FIFO.

4.1.1 Versione del codice risalente al 2021

In questo paragrafo viene presentata una versione precedente del codice, sia per FPGA sia per Real Time, al fine di evidenziarne i punti da migliorare e successivamente illustrare le modifiche adottate.

Main FPGA

Dalla figura 4.1, si vede come nel While loop siano presenti sei canali di acquisizione, tre delle tensioni (modulo 8) e tre delle correnti (modulo 7). Da questi vengono prelevati i dati ogni microsecondo (teoricamente), vengono inglobati in un array 1D (6x1) ed inviati al DMA FIFO, tramite un For Loop. Quest'ultimo viene usato poichè la funzione di trasferimento dei dati (DMA FIFO) accetta soltanto elementi singoli in ingresso (impostazione modificabile nelle proprietà del FIFO), quindi il ciclo in questione attraverso "l'indexing" (figura 4.2) preleva un dato alla volta dal vettore e lo invia all'interno del buffer.

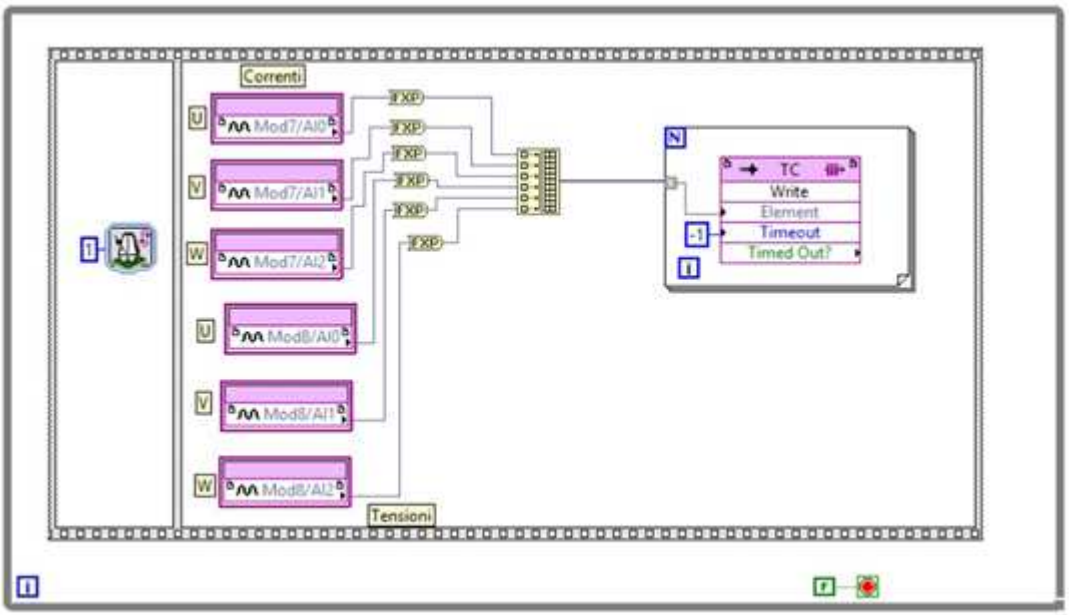


Figura 4.1: Acquisizione segnali elettrici FPGA, Versione di Romanini Riccardo

Come si può osservare dalla figura 4.1, non erano ancora stati implementati controlli sull'*overflow*. LabView, mette a disposizione la possibilità di sfruttare la funzione di *Timeout*, la quale si occupa di inviare un segnale booleano "True" nell'eventualità in cui il tempo (espresso in clock) impostato dall'utente, sia inferiore a quello impiegato dal buffer per liberare nuovo spazio al suo interno.

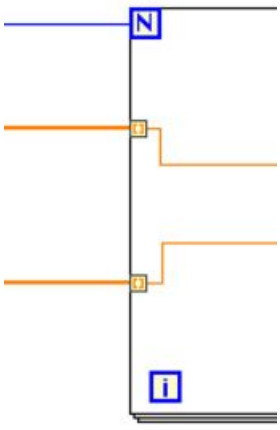


Figura 4.2: Auto-Indexing. Fonte: [26]

Main RT

I dati provenienti dal FPGA vengono letti attraverso le funzioni “Invoke Method.Read” (prima sequenza fig. 4.3), nelle quali è anche possibile impostare il numero di elementi esatti da estrapolare ed il rispettivo Timeout.

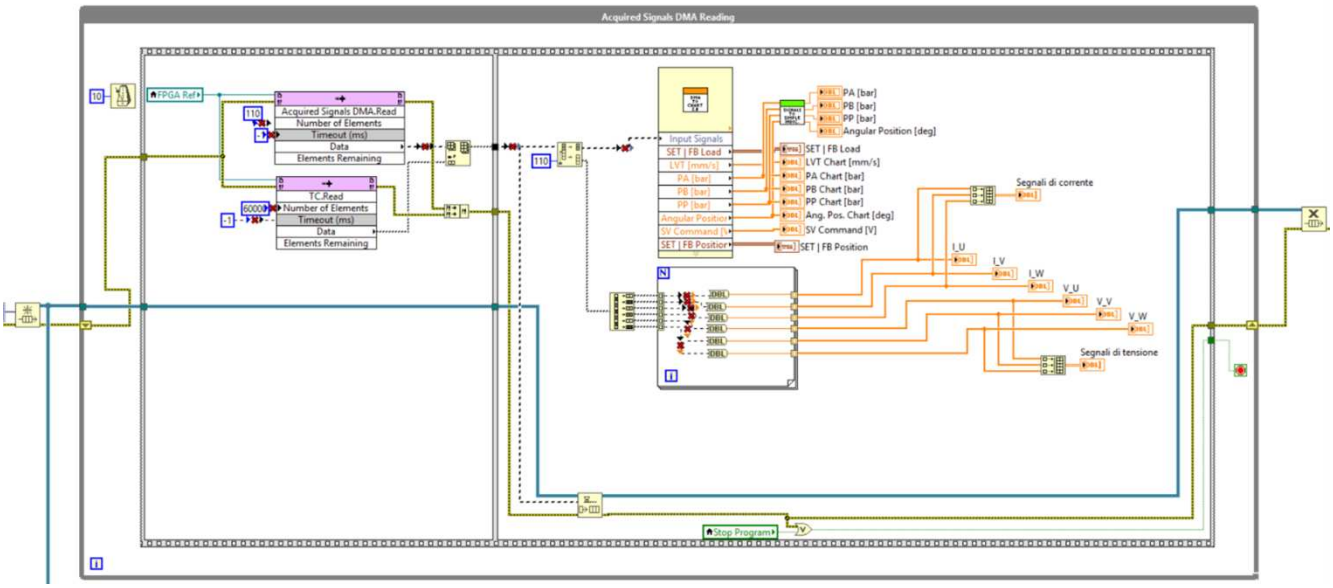


Figura 4.3: Main RT: Lettura dei segnali provenienti dal FPGA e inserimento dei dati nella Queue. Fonte: [7]

Il loop è sincronizzato dalla funzione “Wait Until Next ms Multiple” ogni 10 ms (in alto a sinistra fig. 4.3), quindi dai metodi vengono prelevati rispettivamente:

- Per i segnali fisici 110 dati, in quanto FPGA dovrebbe leggere 11 dati ogni ms;
- Per i segnali elettrici 60000 dati, in quanto FPGA dovrebbe leggere 6000 dati ogni ms.

Nella seconda parte di codice, invece, i dati vengono predisposti all’interno di un unico array ed inviati al Main PC e contemporaneamente visualizzati all’interno di appositi grafici.

Il problema principale di queste due VI (FPGA e RT), risiede sia nelle funzioni sia nella struttura, in quanto non sono ottimizzate per reggere tali prestazioni. Infatti, come poi verrà spiegato nel successivo paragrafo, la struttura adottata nella figura 4.1 non è predisposta per raggiungere un campionamento di 1 MS/s. Di conseguenza anche il RT subisce rallentamenti.

4.1.2 Versione attuale del codice

Anche in questa versione la priorità del codice è quella di acquisire sei segnali diversi ad una frequenza di 1 MHz, utilizzando però funzioni diverse.

Main FPGA: Typical I/O nodes operation VS Typical User-Controlled I/O Sampling operation

La precedente VI, nonostante fosse esplicito il fatto che dovesse acquisire i segnali ogni microsecondo, non riesce nell'intento. Questo perché la struttura del codice stesso, definita come *Typical I/O nodes operation* (4.4) non permette di raggiungere tali prestazioni.

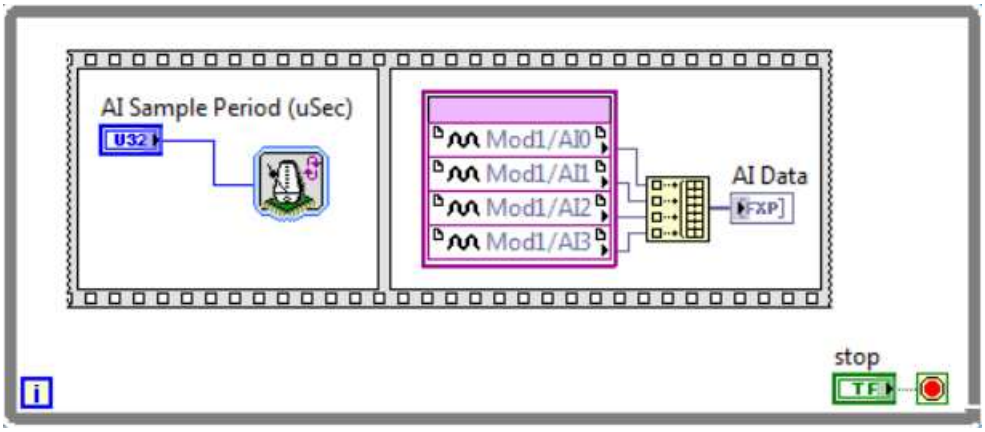


Figura 4.4: esempio struttura tipica I/O nodes. Fonte: [27]

Quando la funzione I/O Node (in viola nella figura 4.4) viene chiamata, avvisa il rispettivo modulo (NI 9223) di leggere il dato, prenderlo e trasferirlo al FPGA e non può essere richiamata nuovamente fino a quando tutti questi passaggi non sono stati completati. Infatti, il massimo della velocità ottenibile da questa procedura è di 350 kS/s [27].

Questo spiega il motivo per il quale la precedente versione di codice non riesce a scendere sotto i 3-4 microsecondi a iterazione nel ciclo di acquisizione.

Al contrario invece, la struttura *Typical User-Controlled I/O Sampling operation* (figura 4.5), permette di fare più operazioni contemporaneamente.

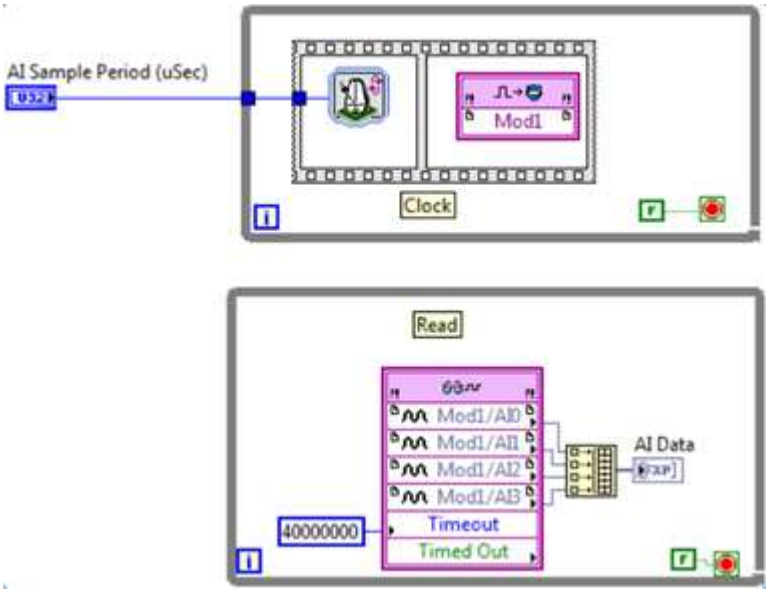


Figura 4.5: Typical User-Controlled I/O Sampling operation. Fonte: [27]

Il ciclo "Clock" (in alto, fig. 4.5), attraverso la funzione *Generate I/O Sample Pulse*, nel momento in cui viene attivata, si occupa di inviare un trigger al modulo per indicargli di acquisire il dato. Per ognuno avviene una conversione Analogico-Digitale che impiega circa 1 microsecondo. Una volta finita l'ADC, il dato viene inviato alla lettura nel codice FPGA, la quale ha un suo buffer interno da cui poi estrapola il dato.

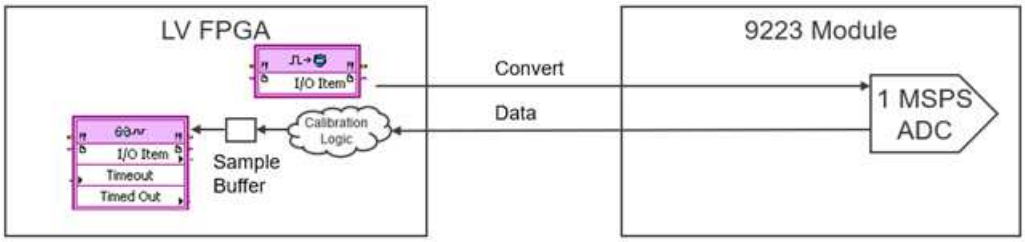


Figura 4.6: User Controlled I/O Sampling. Fonte: www.Ni.com

Questo meccanismo permette così di separare le due operazioni ed effettuare separatamente ciclo di "Clock" da quello di "Read" arrivando ad 1 MS/s.

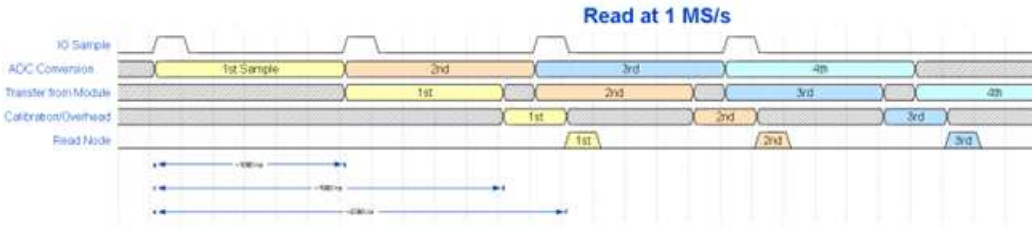


Figura 4.7: Diagramma per spiegare il funzionamento dello "User Controlled I/O Sampling". Fonte: [28]

Come si può osservare nella figura precedente, solamente la prima operazione impiegherà in totale 2.3 microsecondi, mentre dalla seconda, si otterrà una lettura ogni microsecondo, questo grazie al fatto che le operazioni di acquisizione e lettura vengono effettuate separatamente.

Per quanto riguarda il codice di acquisizione per il banco ASTIB, le cose sono un po' più complesse.

Partendo dal ciclo "clock" (in alto a sinistra, fig. 4.8), si trovano le funzioni che hanno il compito di dettare al modulo la frequenza di acquisizione dei dati e sono presenti sia per l'ingresso 7 (correnti) che 8 (tensioni).

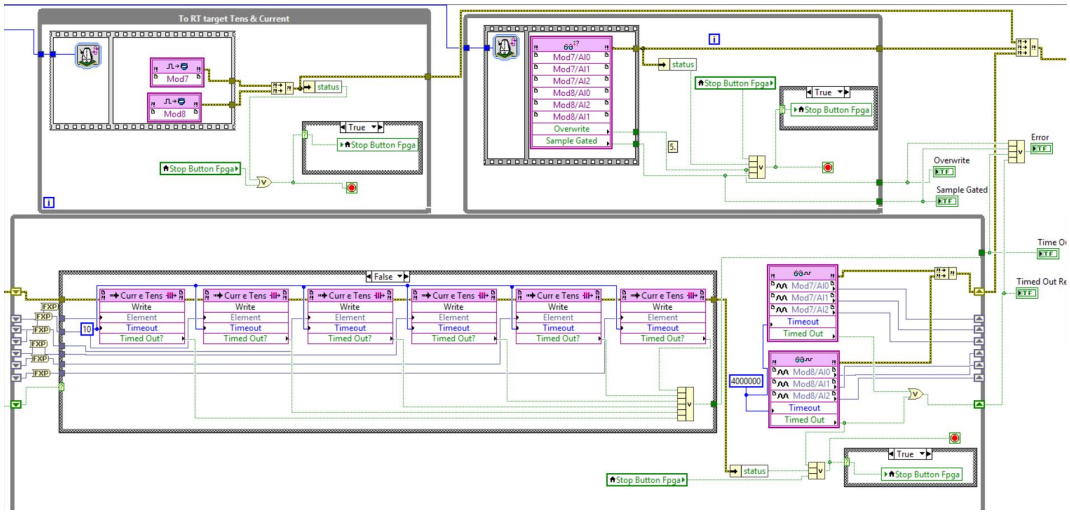


Figura 4.8: Main FPGA

In basso a destra invece si trova un ciclo While contenente le funzioni di lettura (Read

I/O method), che però in questo caso, invece di costruire un vettore con i sei dati e inviarli dentro un *ciclo For*, attraverso l'utilizzo di *Shift Register*, vengono inviati direttamente al FIFO senza subire i rallentamenti causati dall'indexing (come invece accadeva nella fig. 4.1). Infatti, mettendo sei DMA FIFO tutti uguali in serie, i dati vengono inseriti all'interno di quest'ultimi (uno alla volta) in ordine da sinistra verso destra e poi estratti nel RT, sotto forma di array monodimensionale. Così facendo si velocizza l'acquisizione e si riescono a mantenere le velocità di campionamento di 1MS/s anche con sei canali.

Per quanto riguarda la gestione dell'errore ci sono diversi controlli: due nel loop in basso, dove si monitora la presenza di un Timeout del FIFO o della funzione di lettura, altri due invece nel loop in alto, dove si controlla la presenza di *overwrite* e il *sample gated*.

Nell'eventualità si verificasse uno dei sovracitati eventi, i cicli verrebbero arrestati tutti insieme, grazie all'utilizzo delle Local Variables.

4.1.3 Overwrite e Sample Gated

- Overwrite (fig. 4.9): si verifica quando la funzione "Read I/O method" viene chiamata più volte della funzione di lettura. Questo porta ad una sovrascrittura dei dati e quindi ad una perdita degli stessi;

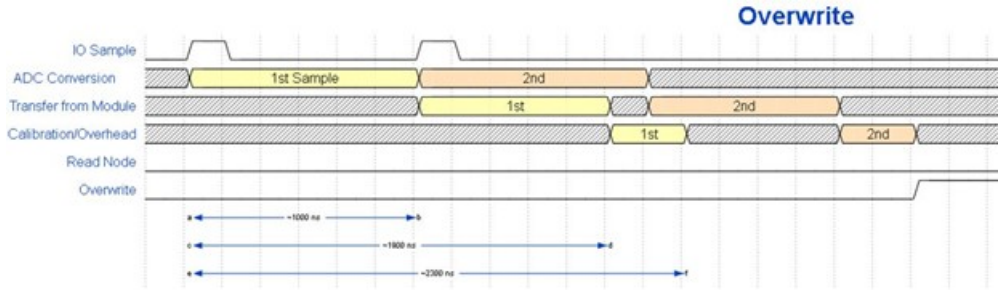


Figura 4.9: Overwriting. Fonte: [28]

- Sample Gated (fig. 4.10): si ha quando la funzione "Read I/O method" viene chiamata troppo velocemente rispetto alla velocità impostata (può verificarsi anche che superi i 1 MS/s). Questo fa si che il trigger successivo venga ignorato, in quanto ancora non è stata completata l'operazione precedente.

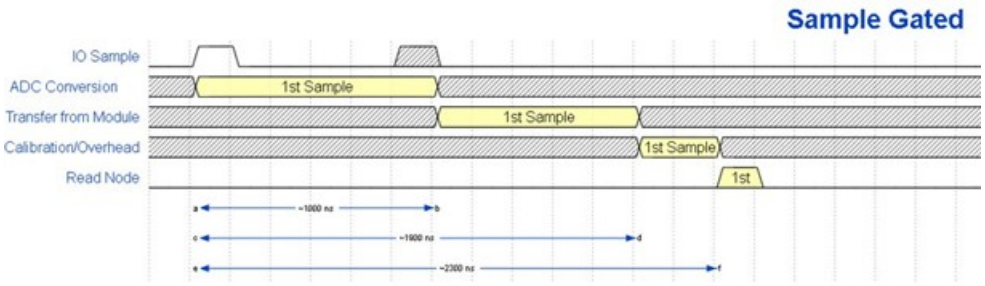


Figura 4.10: Sample gated. Fonte:[28]

In questo caso, nel diagramma si può osservare come il trigger sia stato attivato ancor prima di 1 microsecondo, quindi non viene considerato.

Tutto il codice principale viene posto all'interno di un case selector, dove il suo stato viene gestito dalla presenza o meno di un errore proveniente dalla fase di inizializzazione del codice.

4.1.4 Parentesi sulla gestione e sincronizzazzione del segnale di stop tra i vari loop

Figura 4.11: Gestione dei segnali di stop

Per sincronizzare i segnali di stop nel FPGA è stata adottata una configurazione che prevede l'utilizzo di *Local Variables*. Questa struttura prevede una prima fase di inizializzazione della variabile di stop, dopodiché, all'interno dei While Loop vengono poste le Local Variables.

La struttura della figura 4.11 (a destra) si basa sulla seguente logica:

- La Local Variable "Stop Fpga Button" legge il segnale proveniente dal segnale di stop, inviato dal Real-Time;
- Due cavi leggono la presenza di eventuali errori o timeout provenienti dai vari blocchi (possono essere anche di più o di meno, in base al numero di funzioni che si vuole controllare);
- Un blocco "OR" racchiude i cavi e li manda all'interno di un case selector, dove: TRUE, indica la presenza di un errore o di uno stop che viene sovrascritto all'interno della local variable "Stop FPGA Button"; il False invece prevede una sezione del case selector vuoto, in quanto non si vuole salvare l'informazione.

Grazie a ciò è possibile utilizzare una sola Local Variable per fermare tutti i cicli ("Stop FPGA Button") e fa sì che l'errore presente in un Loop, possa non solo fermare il ciclo in cui si trova, ma anche tutta la VI.

Questa logica, però, è succube delle problematiche che si incorrono nell'utilizzo di queste variabili, ovvero le "race conditions". Si verificano quando due o più parti del codice concorrono per accedere o modificare la stessa variabile nello stesso momento. Avendo integrato questo metodo di stop solamente nei due cicli di acquisizione, ovvero per i segnali elettrici e fisici, non si incorre in questo fenomeno. Al contrario questo accade nel Real Time a causa dell'elevato numero di loop. Per questo, nel Main RT, si è implementato un metodo differente per fermare i cicli, utilizzando i *Notifier*, il quale verrà approfondito in seguito.

4.2 Main RT: Lettura, sincronizzazione ed invio al Main PC dei dati acquisiti

Prima di iniziare con la spiegazione dei singoli cicli, si riporta uno schema (fig. 4.39) che riassume la logica di trasferimento dei dati attraverso quattro While Loop.

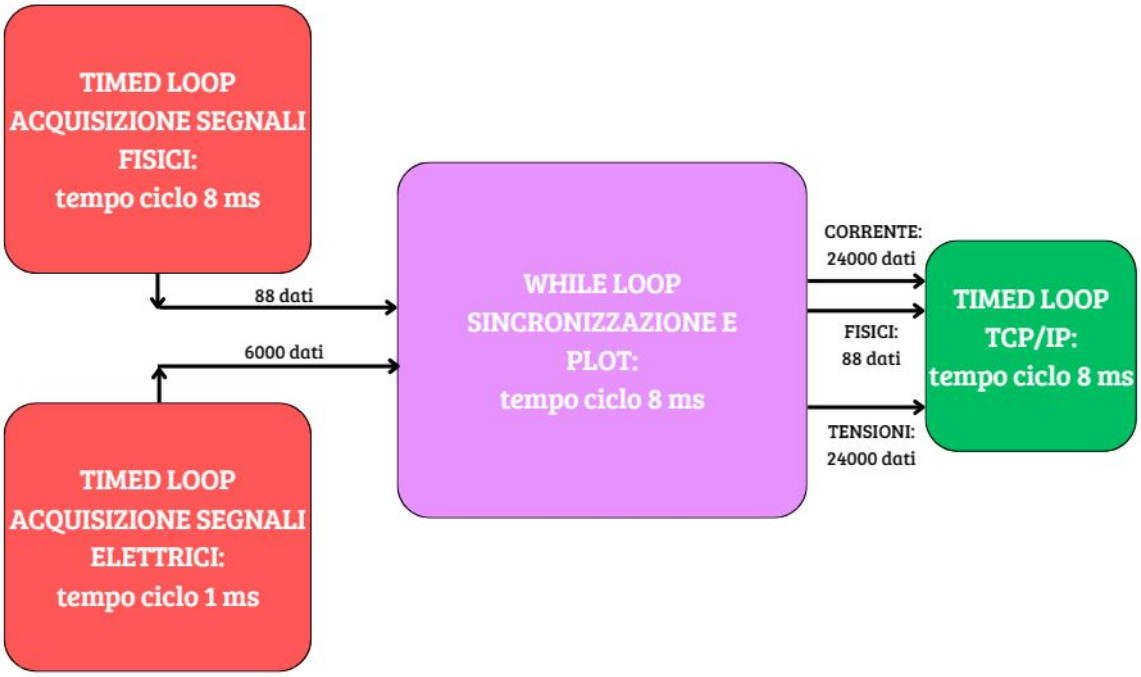


Figura 4.12: Schema del meccanismo di lettura, sincronizzazione ed invio dei dati al Main PC

I quadrati in rosso rappresentano i While loop di acquisizione dei segnali fisici (in alto) ed elettrici (in basso), dove vengono estratti dal primo 88 dati ogni 8 ms e dal secondo 6000 dati ogni 1 ms. Questi entrano all'interno del ciclo di sincronizzazione e visualizzazione. Una volta avviata la connessione con il Main PC, vengono trasferiti in un altro ciclo dove si trovano le funzioni TCP/IP che si occupano del loro invio.

4.2.1 Lettura dei segnali elettrici

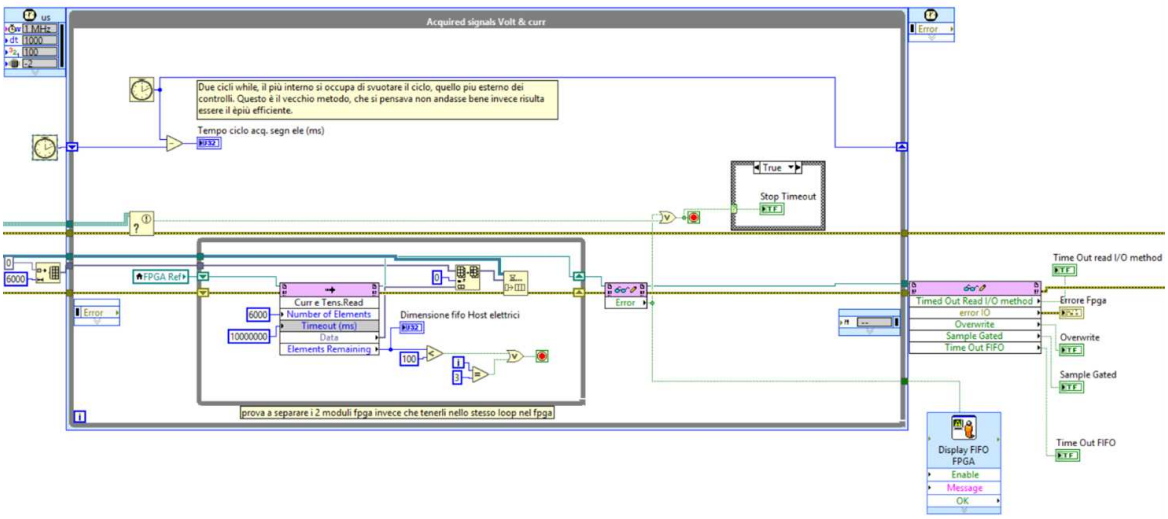


Figura 4.13: Main RT

Il ciclo While presente nella figura 4.13 si occupa della lettura dei segnali elettrici provenienti dal DMA FIFO. Nella versione precedente del codice, il loop ha una struttura diversa (immagine 4.3), la quale però non è adatta a ricevere 1 MS/s, quindi si è proceduto a costruirne una seconda.

Come prima cosa si è separata l'acquisizione dei segnali fisici da quelli elettrici. Essendo entrambi scritti all'interno del FIFO con velocità di campionamento diverse, effettuando alcuni test si è visto che:

- Il timing migliore per leggere i segnali fisici va dai 4 ms o più;
- Il timing migliore per leggere le tensioni e le correnti è invece solo ed esclusivamente di 1 ms.

Questa differenza di tempo è probabilmente causata dal fatto che la configurazione per acquisire i segnali fisici (Typical I/O nodes operation) risulta essere meno performante rispetto all'altra (Typical User-Controlled I/O Sampling operation). Inoltre c'è da considerare anche che solamente i moduli NI 9223 possono adottare quest'ultima configurazione (per approfondimenti, vedere paragrafo 4.1).

Per quanto riguarda la sua struttura, per il ciclo esterno si è utilizzato un *Timed Loop*, con lo scopo di acquisire dati esattamente ogni ms, sfruttando un clock di 1 MHz ed un periodo di 1000 microsecondi, evitando così ritardi e accumuli di dati all'interno del FIFO. Inoltre, si occupa anche di controllare se dal metodo Read/Write (blocco in viola appena fuori dal loop a destra della fig. 4.13) proviene un errore *Error* (fig. 4.13 a destra, fuori dal loop) (nel FPGA, il cavo di errore che arriva qui, si vede nella figura 4.8 in alto), il quale può comprendere un segnale di:

- Overwrite;
- Sample Gated;
- Timeout;

Nel While loop più interno è presente un *Invoke Method.Read*, per leggere i segnali, e un *Enqueue Elements* per inserirli in una coda. Questa tecnica, con doppio ciclo, permette al software di dedicarsi esclusivamente al prelevamento dei dati presenti all'interno del DMA FIFO e una volta uscito dall'anello più interno, di controllare eventuali stop o errori. Così facendo la potenza viene ripartita meglio e non si verificano problemi di alcun tipo. Come condizione di uscita dal ciclo interno si ha:

- Elementi presenti nel FIFO < 1000;
- Numero di cicli > 3.

Queste vengono inserite in modo tale da evitare che il software rimanga intrappolato all'interno di questa struttura.

4.2.2 Lettura dei segnali fisici

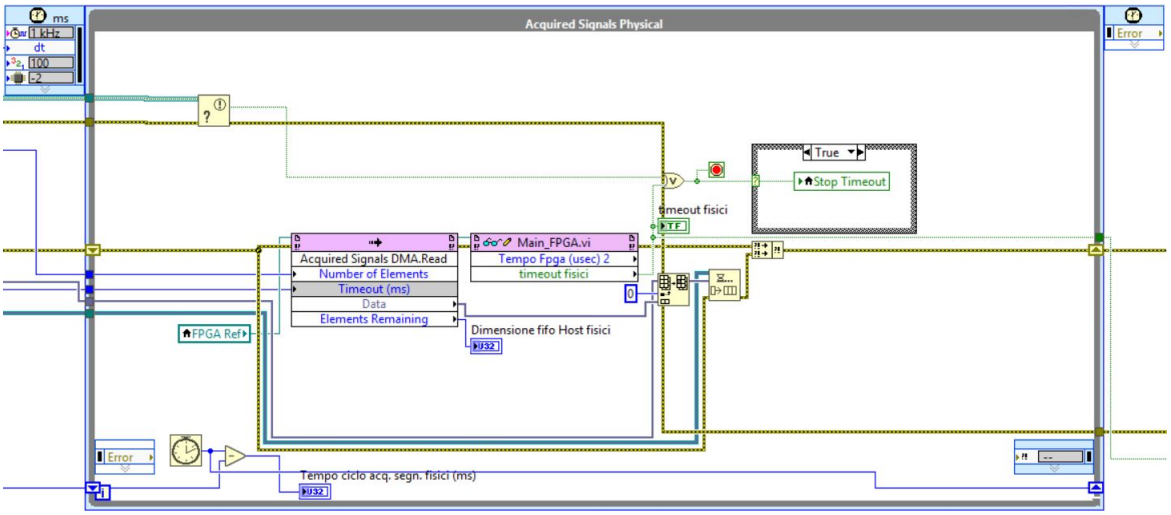


Figura 4.14: Acquisizione dei segnali fisici nel Main RT

In questo caso viene utilizzato un singolo Timed Loop con un clock di un kHz ed un periodo di 8 millisecondi. Infatti la mole di dati elaborata in questo caso è molto minore e quindi non è necessario avere una configurazione con due Loop.

Al suo interno, come nel paragrafo precedente, è presente un metodo per la lettura dei dati ed uno per ricevere i segnali di errore provenienti dal FPGA. Qui vengono prelevati dal FIFO 88 dati ogni 8 ms, per poi essere immessi all'interno della funzione "Enqueue Elements".

Inoltre, all'uscita di entrambi i cicli di acquisizione, ci sono delle funzioni che si occupano di mostrare delle finestre di dialogo, nel caso in cui manifesti un errore, con l'obiettivo di avvisare l'utente.

4.2.3 Ciclo di sincronizzazione e visualizzazione

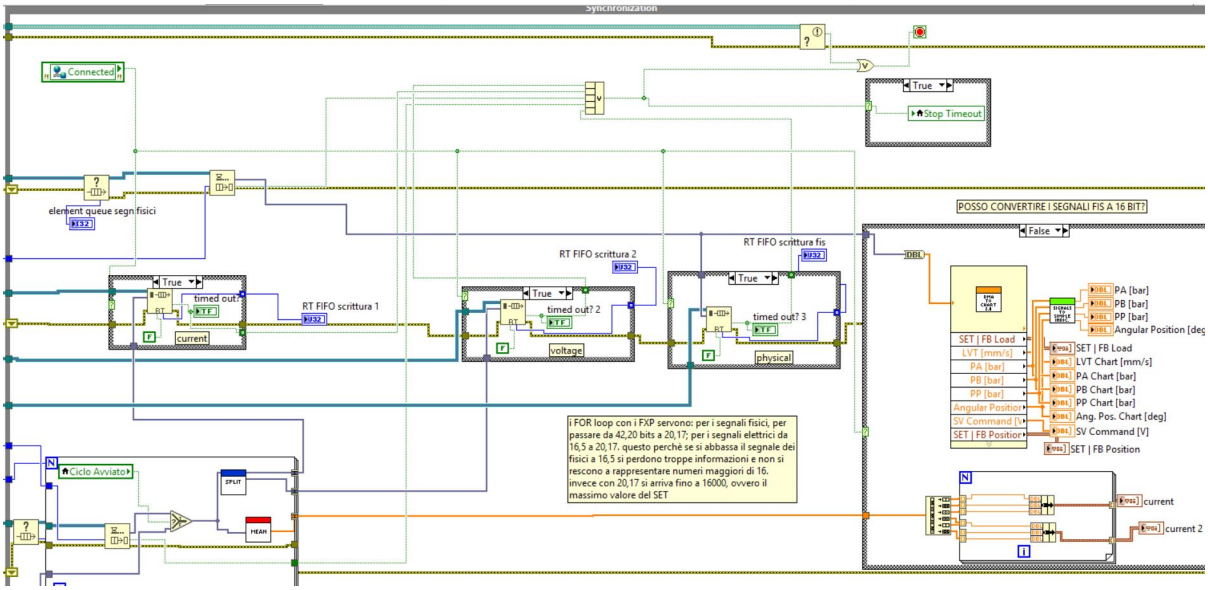


Figura 4.15: Synchronization Loop

Da come si può intuire dal titolo del paragrafo, il ciclo di sincronizzazione nasce dall'esigenza di riportare allo stesso timing i due cicli di lettura.

Osservando in basso la figura 4.15 (o ingrandita la fig. 4.16) si trova un For Loop contenente la funzione Dequeue Elements proveniente dal loop "Data acquisition: voltages e currents" , una funzione Select (il cui scopo verrà approfondito nei paragrafi successivi) al centro e due SubVI a destra, denominate *Mean* e *Split*.

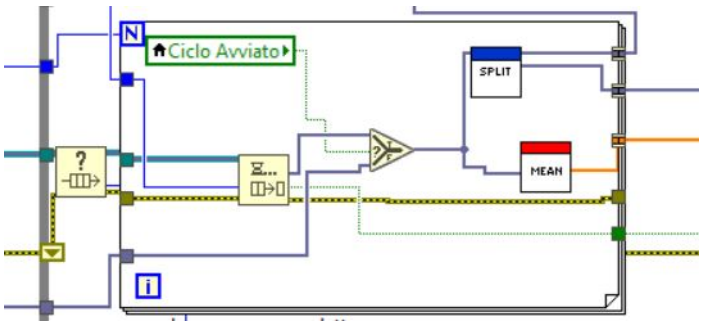


Figura 4.16: For Loop del ciclo di sincronizzazione

La sincronizzazione avviene proprio grazie a questo. All'interno di un singolo ciclo del While Loop, il ciclo For effettua 8 iterazioni. Questo fa si che dalla Queue vengano estratti 6000 elementi **per ogni iterazione del For Loop**, che vengono poi elaborati

ed accumulati all'uscita di quest'ultimo grazie alla funzione *Concatenate*. Prima di ciò, questi dati vengono processati passando sia per la SubVI Split sia per la Mean. Nello specifico, le due si occupano di:

- Split: Dividere il vettore in ingresso di 6000 elementi in due vettori da 3000 ciascuno, uno contenente le correnti ed uno le tensioni. Questo viene fatto per alleggerire il volume di dati che viene inviato ad ogni porta del TCP/IP, cosa che verrà approfondita nel dettaglio in seguito. Facendo così all'uscita del for loop si avranno, dopo le 8 iterazioni, 24000 elementi per ognuno di questi due vettori;

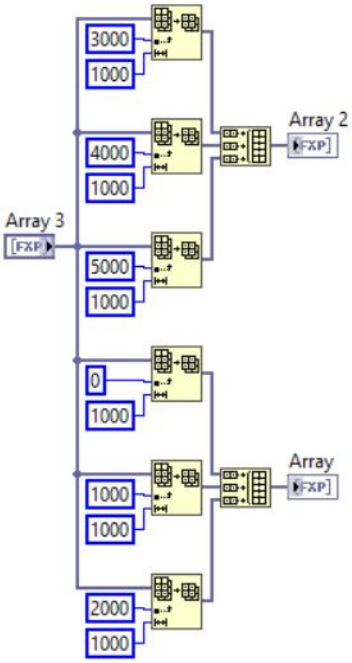


Figura 4.17: SubVI Split

- Mean: Prendere il vettore di 6000 elementi, dividerlo in 6 (in quanto dai due moduli NI 9223 vengono usati 3 canali ciascuno, quindi 1000 dati per ogni canale) e fare la media su quei 1000. Così facendo si ottiene ad ogni iterazione 6 elementi in uscita, che vengono poi accumulati nel ciclo For, ottendo 48 dati per ogni iterazione del ciclo While. Tutto questo viene fatto per evitare di inviare al plot un mole di dati enorme e per sincronizzare la **visualizzazione** con i dati fisici (che così facendo sono dello stesso ordine di grandezza).

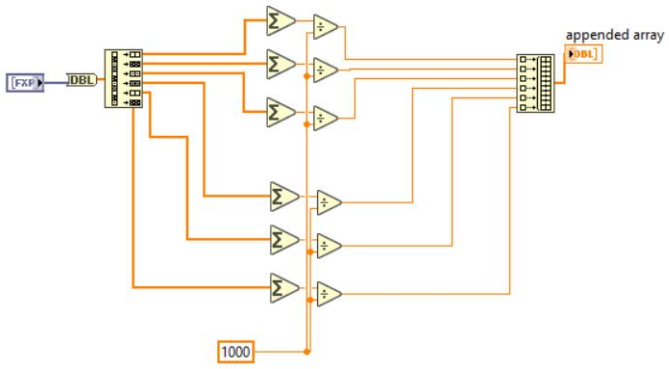


Figura 4.18: SubVI Mean

Dallo schema 4.19 si può osservare come i dati che sono passati per il blocco "Mean", vadano diretti al plot, insieme ai dati fisici, mentre gli altri, vadano direttamente al RT FIFO per essere poi inviati al Main PC.

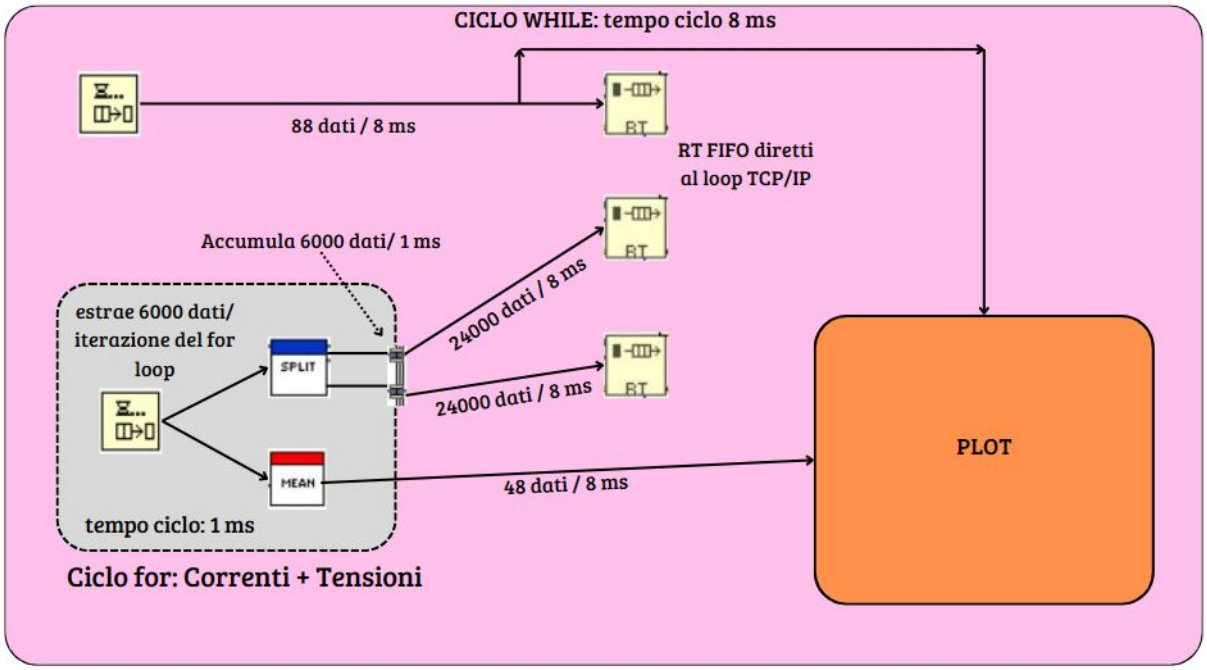


Figura 4.19: Schema logico del While loop di sincronizzazione (fig.4.15): In Rosa il While Loop di sincronizzazione, in Grigio il For Loop e in arancione il Case Selector di visualizzazione dei grafici.

Dalla figura 4.15, si vede come siano presenti anche quattro case selector che circondano le funzioni precedentemente descritte. Quello che contiene le opzioni di visualizzazione (a destra nella figura), ha lo scopo di disattivare l'aggiornamento dei grafici nel momento in cui viene avviata la connessione con il Main PC (in arancione nella figura 4.19). Gli altri tre (rappresentati semplicemente dal simbolo RT FIFO nella 4.19), invece, si occupano di aprire il collegamento con il ciclo "TCP/IP" ed iniziare ad inviare dati. Queste operazioni booleane vengono gestite dalla variabile globale "Connected" (in alto a sinistra), che si attiva nel momento in cui si stabilisce la connessione con l'altra VI.

Parentesi sulla gestione degli zeri matematici

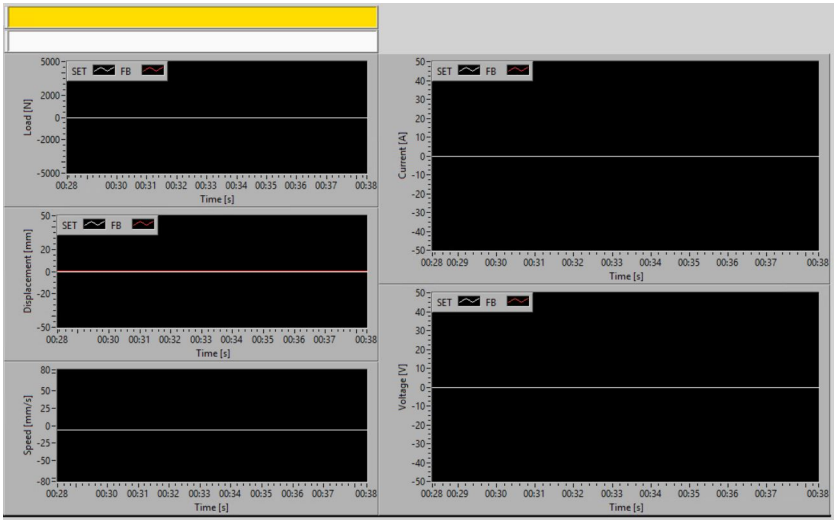


Figura 4.20: Grafici prima di aver premuto "Avvio Ciclo"

Nel momento in cui viene lanciato il Main PC o Main RT, si può osservare nei grafici come non siano mostrati i segnali veri e propri, ma al contrario ci sia una retta costante sullo zero. Infatti, quando vengono ripristinate le sicurezze e viene premuto il tasto "Avvia ciclo" presente nel ACC, gli zeri si "trasformano" nei segnali reali acquisiti dai sensori.

Questa procedura, per i segnali fisici, è stata già mostrata nel paragrafo 3.2.1 in "Initialization, acquisition and control loop" attraverso l'utilizzo delle memorie, mentre, per i segnali elettrici, non avendo utilizzato quest'ultime si è implementato il tutto nel Main RT per evitare di andare a complicare ancora di più l'acquisizione.

Nella figura 4.16, la funzione Select (il triangolo al centro) si occupa di leggere l'informazione della local variable "Ciclo Avviato" e nel caso di:

- FALSE: vengono inviati gli zeri, inizializzati all'esterno da un vettore di 48000 elementi;
- TRUE: vengono estratti i segnali veri e propri dalla Queue.

4.2.4 Comunicazione con il Main PC

Stabilire una comunicazione con il Main PC è stata una delle difficoltà maggiori riscontrate in questa tesi. Sono state fatte diverse ipotesi e si sono valutati svariati meccanismi per far comunicare le due VI, fino a constatare che il protocollo TCP/IP sia l'unico modo efficace per effettuare il tutto. Nei prossimi paragrafi verranno approfonditi:

- Il motivo per il quale è stato utilizzato il protocollo RT FIFO al posto della Queue per far comunicare il ciclo di sincronizzazione con quello TCP/IP;
- La causa che ha portato all'abbandono delle funzioni di Network streams;
- La scelta del protocollo TCP/IP.

Queue Vs RT FIFO

In LabVIEW, le Queue e le RT FIFO (Real-Time First In, First Out) sono strutture di dati utilizzate per la comunicazione tra thread o processi. Nonostante possano essere utilizzate per gli stessi scopi, presentano alcune nette differenze:

- Le RT FIFO nascono appositamente per essere utilizzate nei sistemi Real Time mentre le Queue possono essere usate sia in applicazioni in tempo reale o meno;
- Le Queue sono strutture nate per essere sfruttate in contesti di sincronizzazione dei dati e per come sono fatte, possono creare latenza rispetto ad un RT FIFO;
- Le RT FIFO permettono di gestire meglio le priorità all'interno del ciclo, offrendo anche la possibilità di scegliere la configurazione che si vuole impostare (polling o blocking), al contrario delle Queue che non permettono alcun tipo di modifica;
- In generale le Queue sono più semplici da implementare, in quanto sono più snelle rispetto alla sua concorrente.

Nella vecchia versione del codice (fig. 4.21), sono presenti soltanto Queue, ma nonostante ciò, apparentemente sembra funzionare correttamente. In realtà, a causa della mancanza dei controlli di errore e timeout, i dati vengono scartati in continuazione e quindi tutta l'acquisizione risulta errata.

Con l'inserimento di questi ultimi e con la successiva complicazione del codice, le Queue, a causa dell'eccessiva latenza e della minore ottimizzazione in contesti real-time, hanno portato definitivamente il software a non funzionare più, lasciando così posto al suo concorrente.

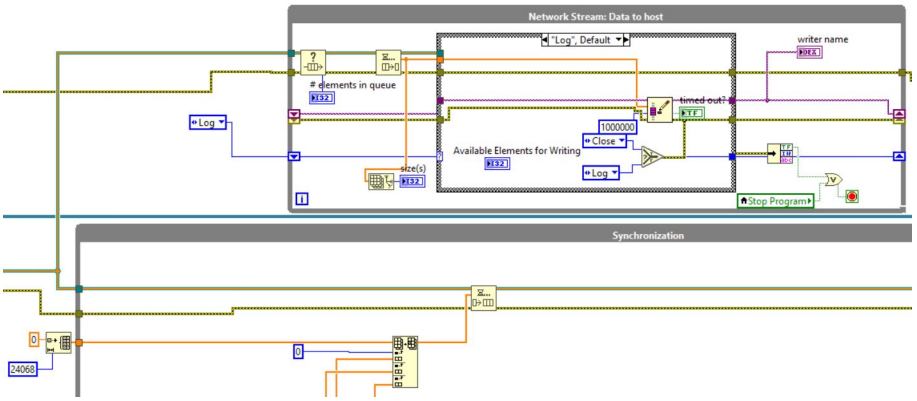


Figura 4.21: Primo approccio (errato) di comunicazione utilizzando le Queue.

RT FIFO quindi ha permesso di ottimizzare la trasmissione di dati dal ciclo di sincronizzazione a quello dei Network Streams. Quest'ultimo ha due modalità di utilizzo:

- Polling: In questa modalità, se si tenta di inserire (o estrarre) un elemento in un FIFO pieno (o vuoto), il processo continua a provare ad aggiungere l'elemento ad intervalli regolari (polling) fino a quando il FIFO è pronto ad accettare nuovi dati. Questo significa che il processo controlla periodicamente la disponibilità del buffer. A causa di ciò, le risorse impiegate sono molto elevate e possono portare a sovraccarichi delle CPU.
- Blocking: Qui invece, se si tenta di inserire un elemento (o estrarre) in un FIFO pieno (o vuoto), il processo si blocca finché il buffer non è pronto ad accettare nuovi dati. Questo evita la necessità di controllare periodicamente la sua disponibilità. Così facendo si risparmia potenza rinunciando un minimo alle prestazioni.

Le impostazioni di default del RT FIFO, utilizzano la modalità polling. Questa però, nelle prime versioni create, portava inevitabilmente alla saturazione della CPU (figura 4.22). Si è deciso così di accantonarla e sfruttare la configurazione blocking, la quale ha permesso un leggero miglioramento delle prestazioni. In seguito però si è scoperto come il problema non fosse causato soltanto dalle eccessive performances del RT FIFO, ma dai Network Streams, principale collo di bottiglia della VI, che verrà analizzato nel prossimo capitolo.

Figura 4.22: Saturazione della CPU 3

Network Streams

Come già accennato, i Network Streams non sono riusciti a garantire le performances desiderate. Ecco la versione di codice precedente all'introduzione degli RT FIFO:

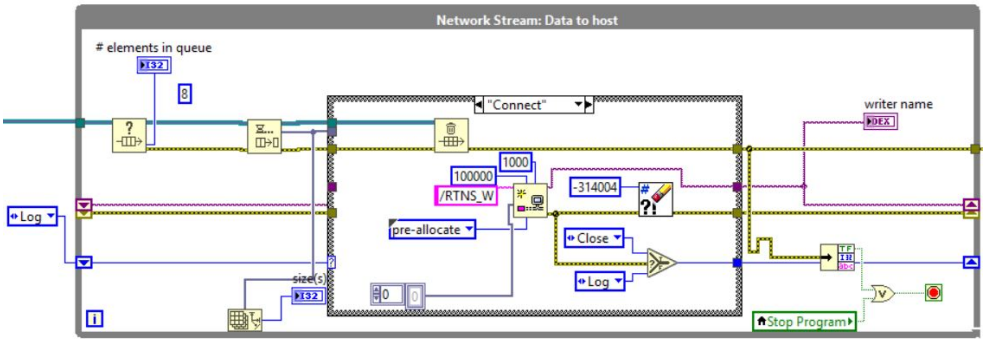


Figura 4.23: Network Streams, connessione all'endpoint Reader.

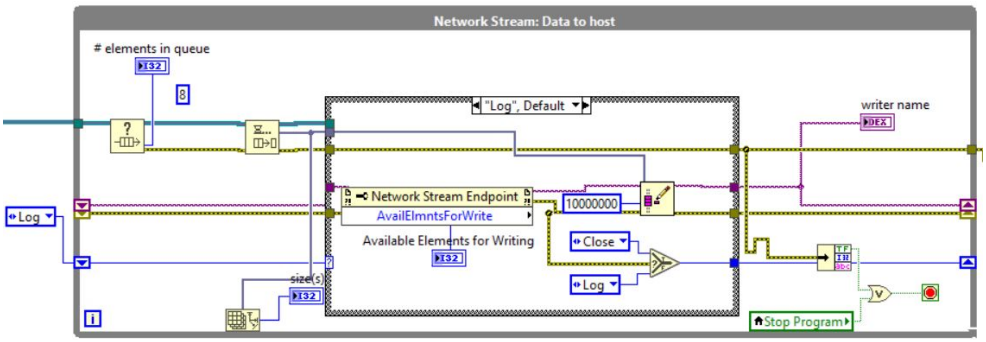


Figura 4.24: Network Streams, scrittura dati.

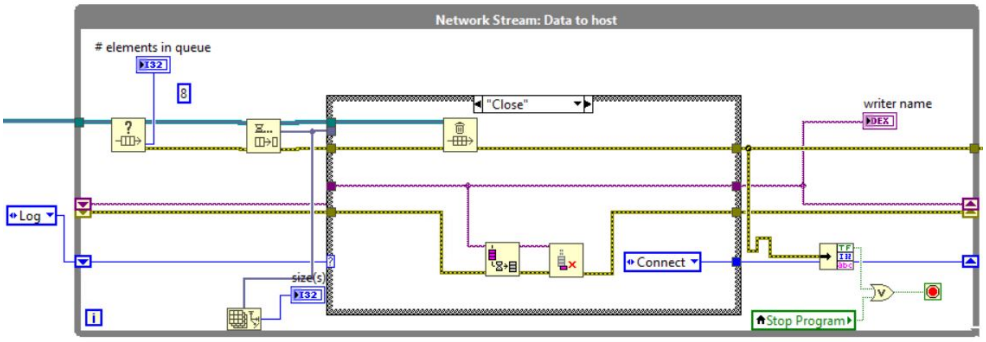


Figura 4.25: Network Streams, chiusura della connessione.

Una struttura molto semplice, formata da tre parti (creazione della comunicazione, log e chiusura della connessione) che prevedeva l'utilizzo delle Queue.

1° versione

Come si può osservare dalla figura 4.21, è stato commesso un errore importante , ovvero l'utilizzo del formato double, molto più pesante (64bit) rispetto ai fixed-point configurati a 16 bit. Infatti, nell'immagine 4.26 si vede come nel primo tentativo di inviare dati tramite Network Streams usando il formato double, la banda disponibile raggiunga la metà del suo massimo, al contrario dei fixed point invece che occupano neanche un 5%. Va sottolineato che il software impiegato per monitorare l'occupazione della rete è la 'Gestione Attività' di Windows, che considera anche lo spazio utilizzato da altre applicazioni. Quindi, sebbene il 50% indicato non sia esclusivamente attribuibile a LabView, è ragionevole presumere che una parte significativa lo sia..

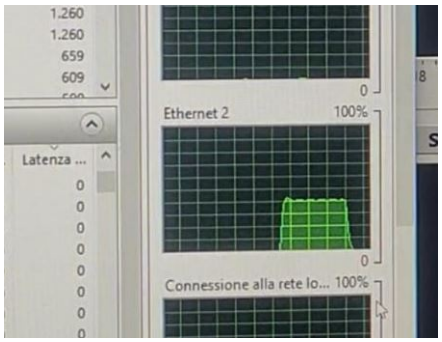


Figura 4.26: Spazio occupato dalla comunicazione di dati in formato double nella rete

2° versione

In seguito, una volta adottati i FXP (figura 4.23, 4.24, 4.25) si è proceduto a verificare a livello teorico se la mole di dati che si stava acquisendo potesse rispettare i requisiti di sistema. Si è effettuato così un semplice calcolo:

- La rappresentazione di un dato in formato fixed point utilizza 16 bit.
- Si ricevono 6000 dati al secondo (trascurando al momento i dati fisici in quanto sono di due ordini di grandezza inferiori, in termini bit occupati, rispetto agli altri), equivalenti ad una frequenza di campionamento di 6 MS/s.
- La velocità di trasmissione complessiva è di $6\,\mathrm{MS/s} \times 16\,\mathrm{bit} = 96\,\mathrm{Mbit/s}$.
- Se si moltiplica per un fattore 4 di sicurezza per sovrastimare il flusso di dati calcolato, la velocità totale raggiunge $96\,\mathrm{Mbit/s} \times 4 = 384\,\mathrm{Mbit/s}$.

Verficando il datasheet della C-Rio 9039 [29] si può vedere come massimo throughput è di 1 Gb/s, quindi superiore alla sovrastima calcolata.

3° versione

Il passo successivo è stato quello di sostituire le Queue delle figure 4.23, 4.24, 4.25 con gli RT FIFO ed infine controllare se i buffer dei Network Streams si riempissero o meno. Dopo alcuni test si è visto in realtà che risultavano perennemente vuoti, sia quello di Host, sia quello di Target. Questo stava a significare che il collo di bottiglia non fosse il trasferimento dei dati, ma la scrittura degli stessi nel "Write Single Element To Stream".

Per confermare questa tesi, si è andato a "sezionare" il While loop contenente i Network Streams (fig. 4.27), inserendo alcune sequenze ed andando a conteggiare quanto tempo impiegasse ognuna per completarsi, inserendo delle funzioni di timing.

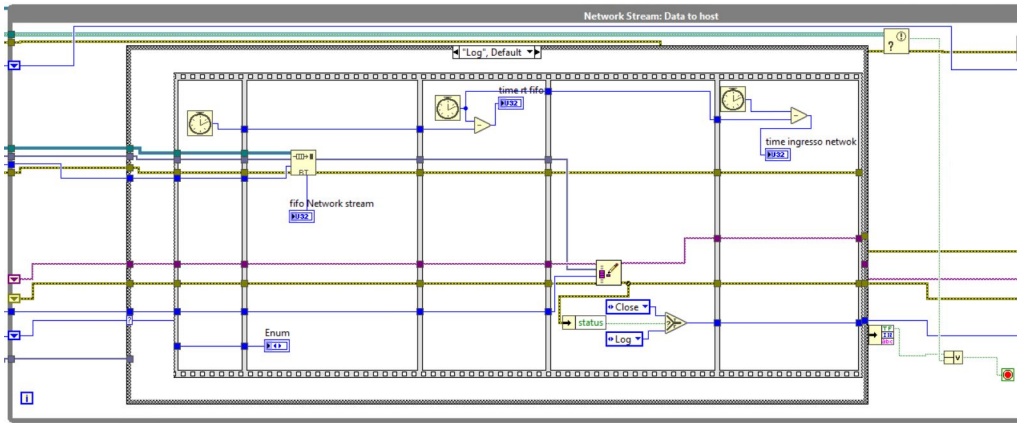


Figura 4.27: Verifica del ritardo di scrittura causato dai Network Streams

Se avesse funzionato correttamente, l'intero ciclo While avrebbe dovuto completarsi in 8 millisecondi, ovvero il tempo ciclo del Loop di sincronizzazione, da cui provengono i dati. Così non è stato in quanto complessivamente il timing si è aggirato intorno ai 14-20 millisecondi (immagine 4.28) e l'unica sezione responsabile questo ritardo è stata proprio quella del "Write Single Element To Stream", ovvero la quarta partendo da sinistra (figura 4.27).

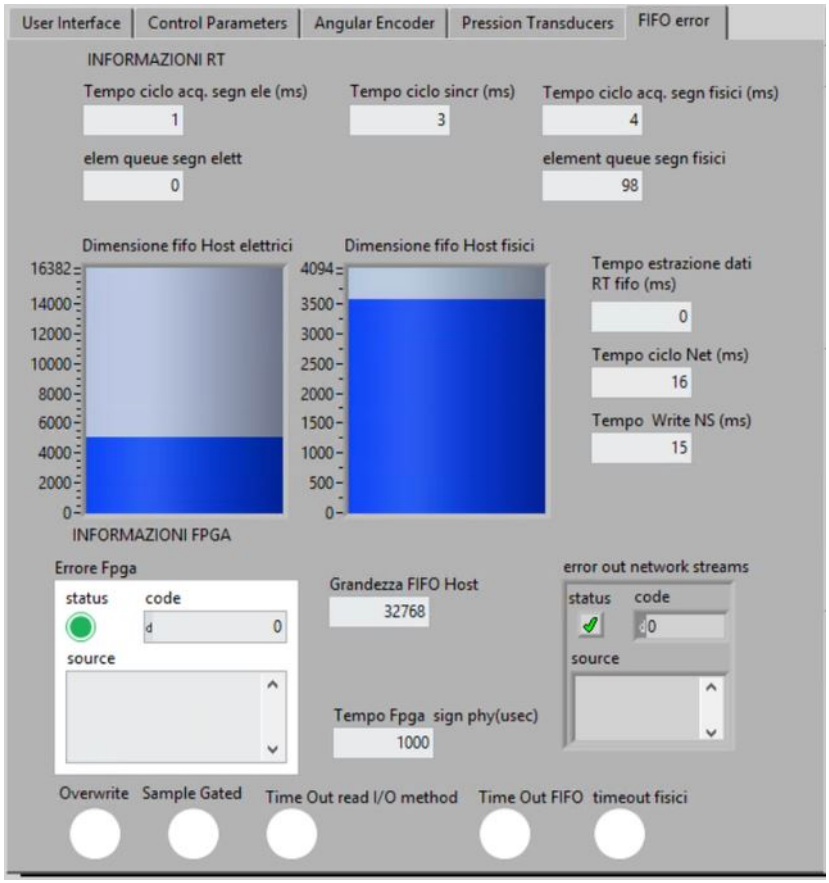


Figura 4.28: Verifica del ritardo di scrittura causato dai Network Streams (indicatore "Tempo ciclo Net (ms)")

Con queste prestazioni, il Network Stream non sarebbe mai stato in grado di scrivere quella mole di dati. Come risultato, si è ottenuta una saturazione della CPU (4.29), quindi una diminuzione netta delle performances, che ha portato a sua volta ad un blocco del programma e ad una sovrascrittura di dati.

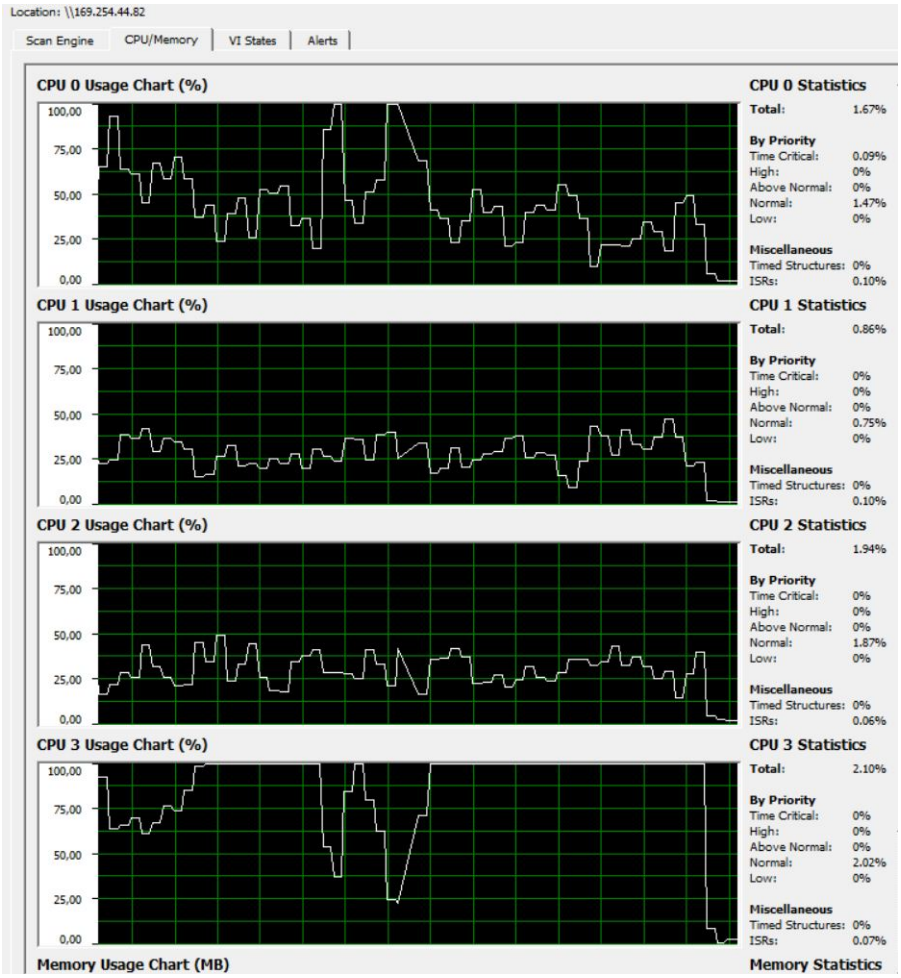


Figura 4.29: Saturazione della CPU 3 con conseguente blocco dell'applicazione

Interessante notare nella figura precedente come la Compact Rio abbia cercato di partizionare la potenza anche nella CPU 0, la quale presenta al centro due picchi complementari ai due "buchi" nel grafico della CPU 3. Probabilmente questo è stato fatto per preservare l'integrita fisica della macchina, cercando di dividere equamente le risorse, dovendo però poco dopo, cedere e bloccare l'applicazione.

Tentativo di salvare direttamente i file in locale utilizzando il TDMS

Durante questo percorso, una volta accantonata l'idea di utilizzare i Network Streams, si è anche pensato di salvare i dati in locale sulla C-Rio. Il salvataggio attraverso il TDMS (Technical Data Management Streaming) in LabVIEW è un processo utilizzato per memorizzare e gestire dati di misura e acquisizione in un formato binario efficiente e strutturato. E' un formato di file nativo di National Instruments utilizzato per la memorizzazione e l'analisi di dati acquisiti o generati da sistemi di misura e controllo.

Come si può osservare nella figura seguente (fig. 4.30), questo formato permette di ottenere le prestazioni migliori di tutti gli altri, garantendo soprattutto velocità elevate di scrittura.

	ASCII	Binary	XML	TDMS
Exchangeable	✓		✓	✓
Small Disk Footprint		✓		✓
Inherent Attributes			✓	✓
High-Speed Streaming		✓		✓

Figura 4.30: Confronto fra i vari formati per il salvataggio dati. Fonte: [30]

La struttura adottata (fig. 4.30, 4.30 e 4.32) è stata la seguente:

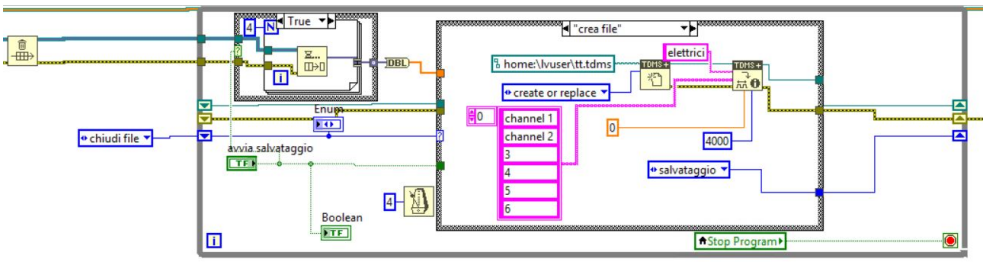


Figura 4.31: TDMS, creazione del file.

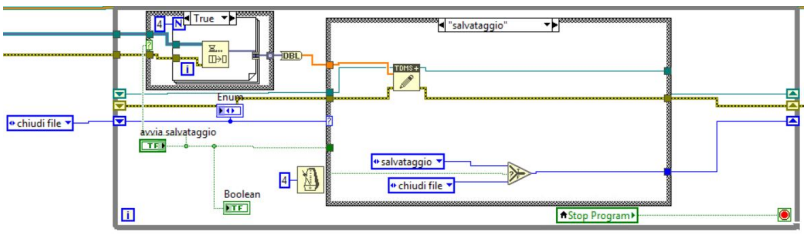


Figura 4.32: TDMS, scrittura dati.

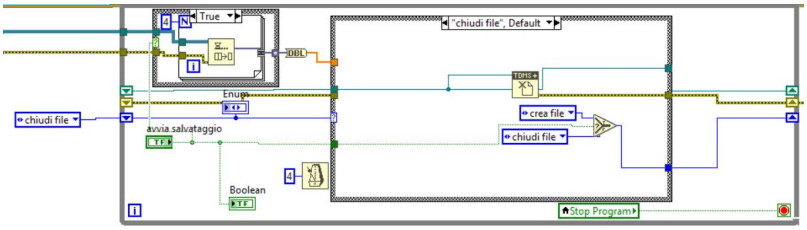


Figura 4.33: TDMS, chiusura del file.

Il suo utilizzo non è stato molto soddisfacente in quanto si è riuscito ad ottenere soltanto un leggero miglioramento delle prestazioni, arrivando a salvare anche fino a 30 secondi, anche se il problema del timeout è rimasto, indipendentemente dall'utilizzo di Queues o RT FIFO. Inoltre un altro problema che si è manifestato è stato quello di estrarre i dati dalla Compact Rio, che risulta un meccanismo molto articolato e scomodo da effettuare.

Alla fine di questi test, si è deciso di abbandonare questa strada e dedicarsi alla ricerca di un modo per inviare i dati al PC in tempo reale.

Scelta del protocollo TCP/IP

All'interno del documento LabView Real-Time 1 Course Manual viene spiegato come il protocollo TCP/IP sia il mezzo migliore per trasferire dati ad alta velocità. Anche il protocollo UDP sembra garantire le stesse prestazioni, anche se in realtà risulta meno affidabile in termini di integrità e ordine dei dati inviati.

Questo collegamento (TCP/IP: Transmission Control Protocol/Internet Protocol) è progettato per offrire un trasferimento affidabile attraverso una rete, riducendo al minimo la perdita di informazioni durante il processo di comunicazione. Presenta le seguenti caratteristiche:

- Stabilisce una connessione affidabile tra il mittente e il destinatario prima di trasferire i dati. Questo comporta una serie di fasi di handshaking per garantire che entrambe le estremità siano pronte per la comunicazione;
- Assicura che il mittente non invii dati più velocemente di quanto il destinatario sia in grado di gestire;
- Utilizza meccanismi di rilevamento degli errori, come il *checksum*, per garantire l'integrità dei dati trasmessi. Se si verificano errori durante la trasmissione, TCP è progettato per rilevarli e richiedere la ritrasmissione dei dati danneggiati;
- Gestisce il riordino dei pacchetti, garantendo che i dati vengano ricevuti nell'ordine corretto anche se vengono trasmessi fuori sequenza sulla rete.
- Nel caso in cui un pacchetto di dati non raggiunga correttamente il destinatario, TCP richiede automaticamente la ritrasmissione del pacchetto mancante.

Questo protocollo ha permesso finalmente la trasmissione completa dei dati, senza provocare timeout o sovrascritture. La prima versione di codice che lo ha utilizzato, è stata strutturata nel seguente modo:

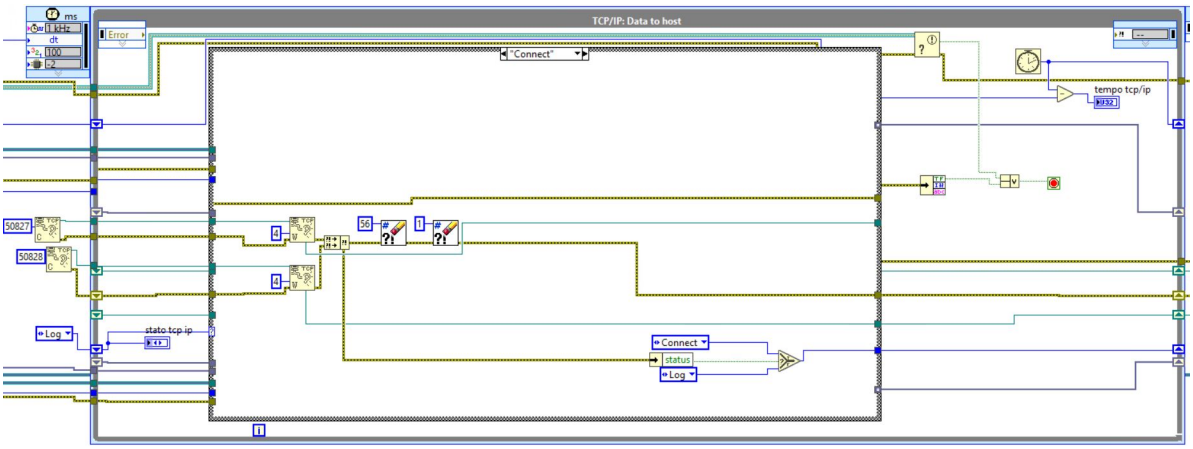


Figura 4.34: 1° versione del codice che ha adottato il protocollo TCP/IP: Connessione con il Main PC

Come si può notare dalla figura 4.34, la prima versione di codice contenente questo protocollo, conteneva due canali di comunicazione, non tre. Questo perchè i dati venivano inviati dal loop di sincronizzazione in modo differente. Infatti, si aveva solo due RT FIFO, uno contenete soltanto le correnti e l'altro le tensioni più i segnali fisici. Le porte quindi erano due: la "50827" e la "50828".

I dati venivano splittati secondo il seguente modo:

- la prima si occupava di inviare le correnti, quindi 24000 dati ogni 8 millisecondi;
- la seconda inviava invece le tensioni ed i segnali fisici, quindi 24088 dati ogni 8 millisecondi.

Durante il test, è saltato all'occhio un problema non di poco conto.

Come si osserva nella figura 4.35, nei grafici di posizione e carico, mancano i segnali di Set mentre sono presenti quelli di Feedback. La cosa particolare è che la macchina riceveva i segnali, si muoveva correttamente nella posizione desiderata, forniva i feedback ma non riusciva a rappresentare i segnali Set. In un primo momento si è pensato ad un semplice errore di collegamento dei fili nella parte di rappresentazione, ma così non era.

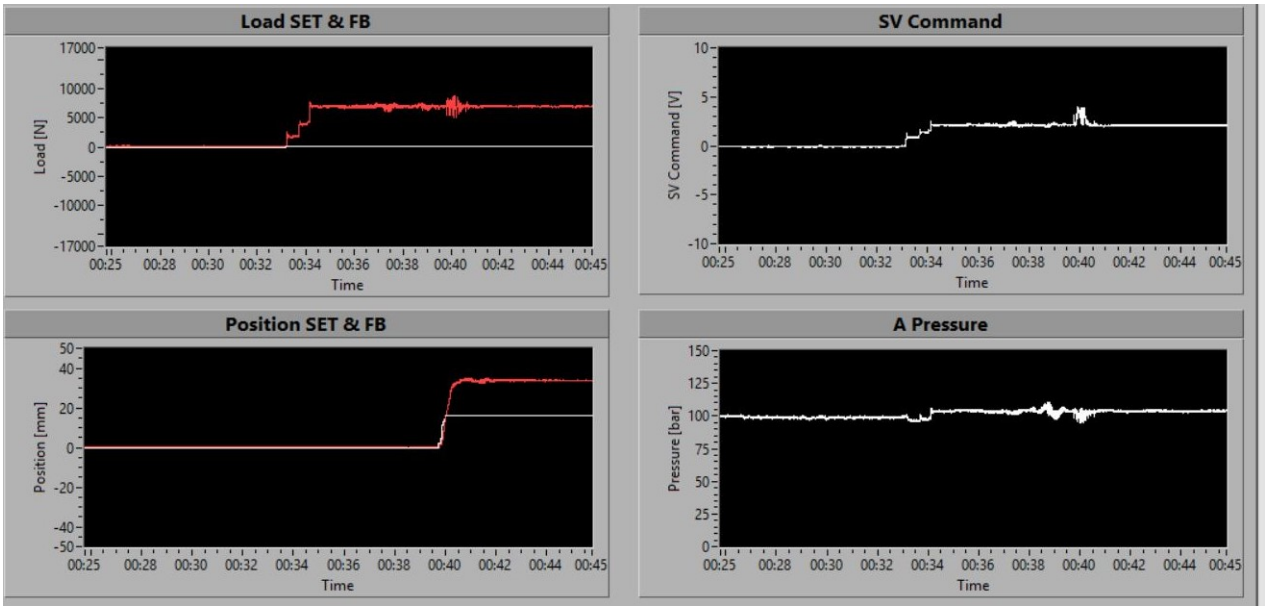


Figura 4.35: Saturazione del segnale di Set di Load e Position.

Osservando attentamente la figura, si nota come il Set del carico sia apparentemente assente, mentre il Set della posizione sia fermo a 16. Questo numero non è casuale, infatti calcolando i massimi e i minimi valori rappresentabili per un fixed-point signed di 16 bit, con parte intera di 5 bit viene fuori che:

$$\text{Integer Word Length} = 5, \quad \text{Word Length} = 16$$

$$\text{Numero massimo rappresentabile} = 2^{\text{Integer Word Length}-1} \times \left(1 - 2^{-\text{Numero di bit frazionari}}\right)$$

$$\text{Numero massimo rappresentabile} = 2^4 \times \left(1 - 2^{-11}\right) = 16$$

$$\text{Numero minimo rappresentabile} = -2^{\text{Integer Word Length - 1}} = -2^{5-1} = -16$$

Come si legge dalle formule precedentemente scritte, i numeri massimi e minimi rappresentabili con quel formato sono rispettivamente 16 e -16. Ciò dimostra perché entrambi i grafici sono saturati a quei valori (anche nel Load ma non si vede in quanto la scala è elevata).

Questo accade in quanto nel ciclo di sincronizzazione di figura 4.36, i segnali fisici, che sono a 42 bit (con una parte intera di 20) vengono immessi nello stesso vettore delle correnti e quindi tutto l'array viene portato a 16 bit (visibile al di sotto del case selector centrale, fig. 4.36), con 5 di parte intera, perdendo così una grande porzione di dati.

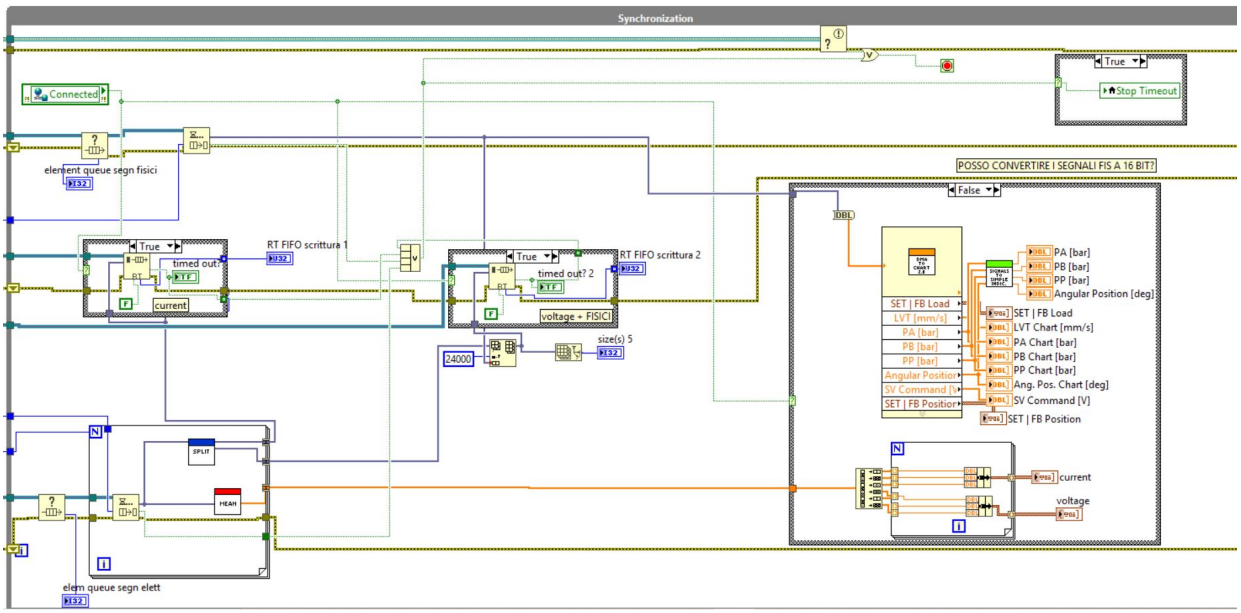


Figura 4.36: 1° versione del codice che ha adottato il protocollo TCP/IP:Ciclo di sincronizzazione

Invece, con un dato di 42 bit e 20 di parte intera si riescono a rappresentare valori con un massimo di 524288 ed un minimo di -524288.

Per questo si è realizzata una seconda versione (la definitiva) con tre canali TCP/IP, ovvero: uno per le tensioni, uno per le correnti e uno per i segnali fisici.

All'inizio si trova l'inizializzazione delle tre porte (4.37):

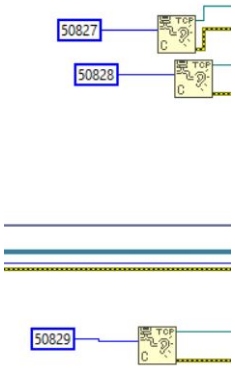


Figura 4.37: Inizializzazione TCP/IP

Di seguito la connessione all'interno del ciclo While (4.38):

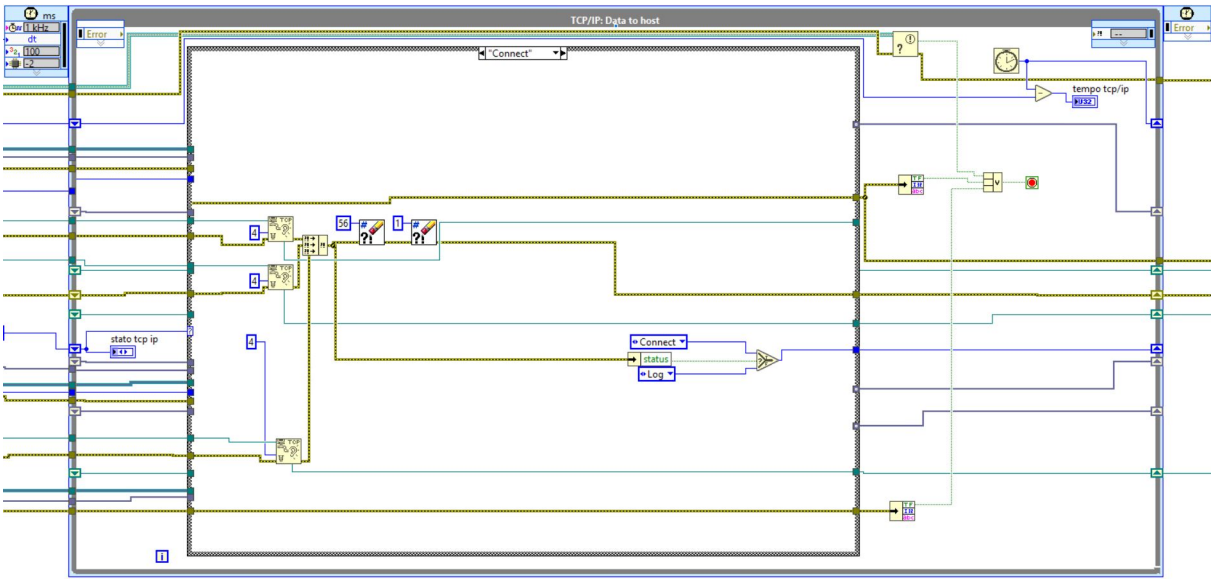


Figura 4.38: Ciclo While TCP/IP: Connessione

Le prime funzioni che si incontrano al di fuori del ciclo sono tre *TCP Create Listener*, che si occupano di creare le porte di "ascolto", mentre all'interno si trovano le *TCP Wait on Listener*, con il compito di aprire la connessione con l'altra VI (il Main PC). Quest'ultime sono infatti molto utili perchè grazie a loro non è necessario ad ogni ciclo creare e distruggere le suddette porte, ma basta inizializzarle una volta soltanto ed attendere che si stabilisca a comunicazione.

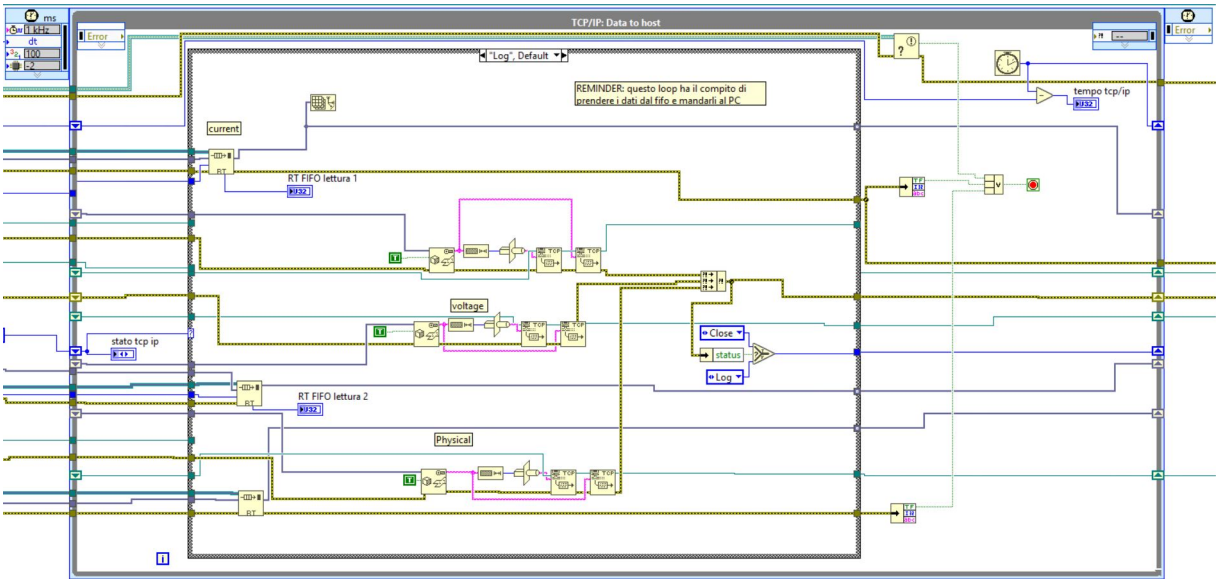


Figura 4.39: TCP/IP: Trasferimento dati

Una volta creata, si passa al log (figura 4.39). Qui i dati vengono convertiti in stringhe ed inviati al Main PC.

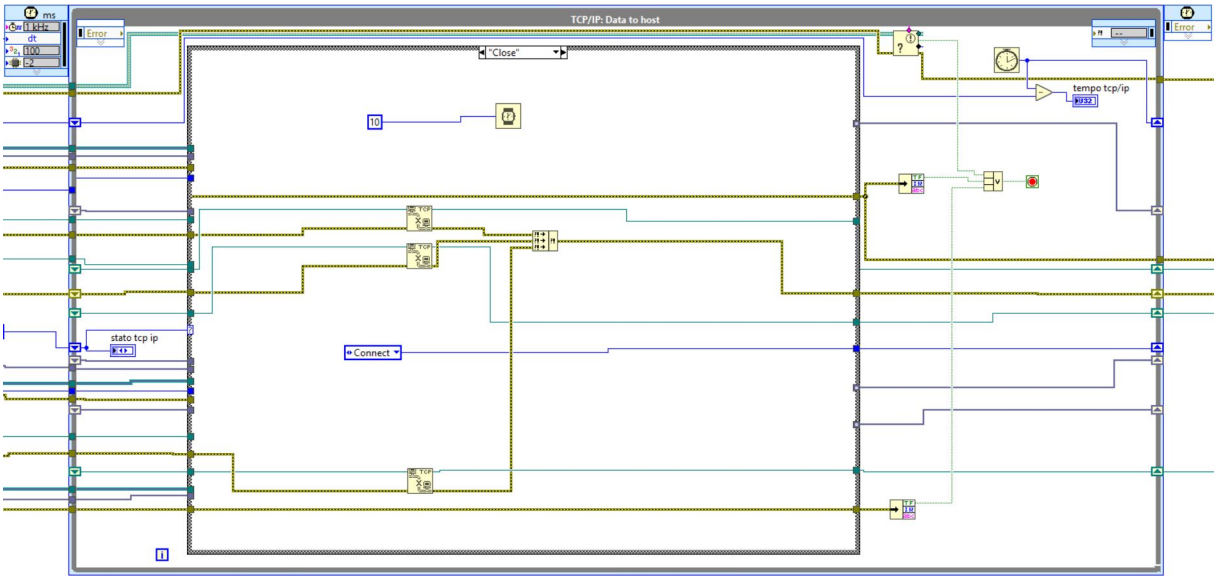


Figura 4.40: TCP/IP: Chiusura connessione

Infine, quando il Main PC viene fermato (fig. 4.40), la connessione si interrompe in sicurezza e il loop si riposiziona nella configurazione dell'immagine 4.38, attendendo così un nuovo collegamento.

4.3 Main PC: Ricezione dei dati, visualizzazione e salvataggio

4.3.1 TCP/IP Read

Con la stessa logica con cui si è costruito il protocollo TCP/IP nel Main RT, anche nel Main PC si è proceduto nel preparare le tre porte per la ricezione dei dati.

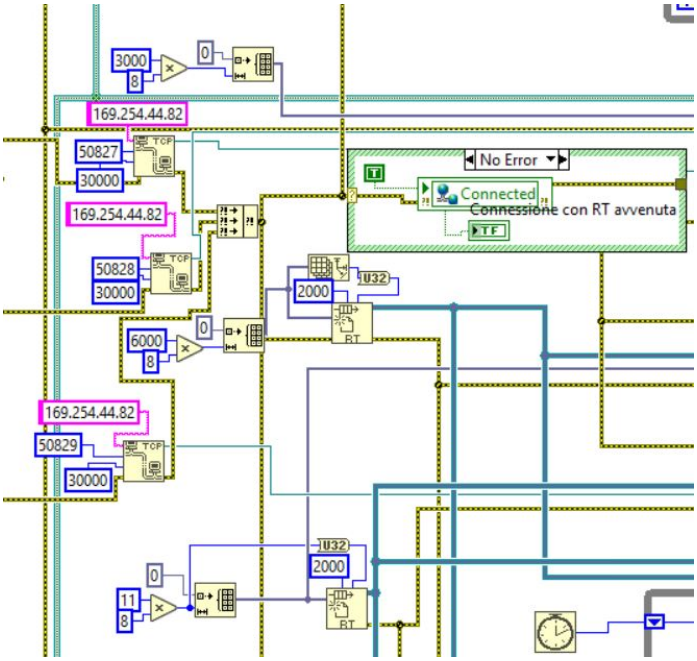


Figura 4.41: TCP/IP: Inizializzazione reader

Osservando la figura 4.41, le prime tre funzioni che si incontrano, da sinistra, servono per inizializzare le porte e configurare l'indirizzo IP della macchina. Poco più avanti si trovano le due funzioni di inizializzazione del RT FIFO, uno per i segnali di corrente e tensione e l'altro per i segnali fisici. Anche in questo caso, per entrambi, si utilizza a modalità Polling, come nel Main RT (in quanto dopo l'introduzione del TCP/IP non si hanno più problemi di saturazione della CPU), in modo da ottimizzare il trasferimento dell'elevato volume di dati.

Il case selector (in verde) è pilotato dal cavo di errore che fuoriesce dalle funzioni di inizializzazione del TCP/IP. Nel caso ci sia un'errore, viene mostrato un avviso per l'utente informandolo del tentativo di connessione fallito con il Main RT. In caso contrario, viene inviato un segnale di TRUE attraverso la variabile globale "Connected" la quale, come descritto nel paragrafo 4.2.3, permette di inviare i dati nel RT FIFO Write del ciclo di sincronizzazione e bloccare la visualizzazione dei grafici, nel Main RT.

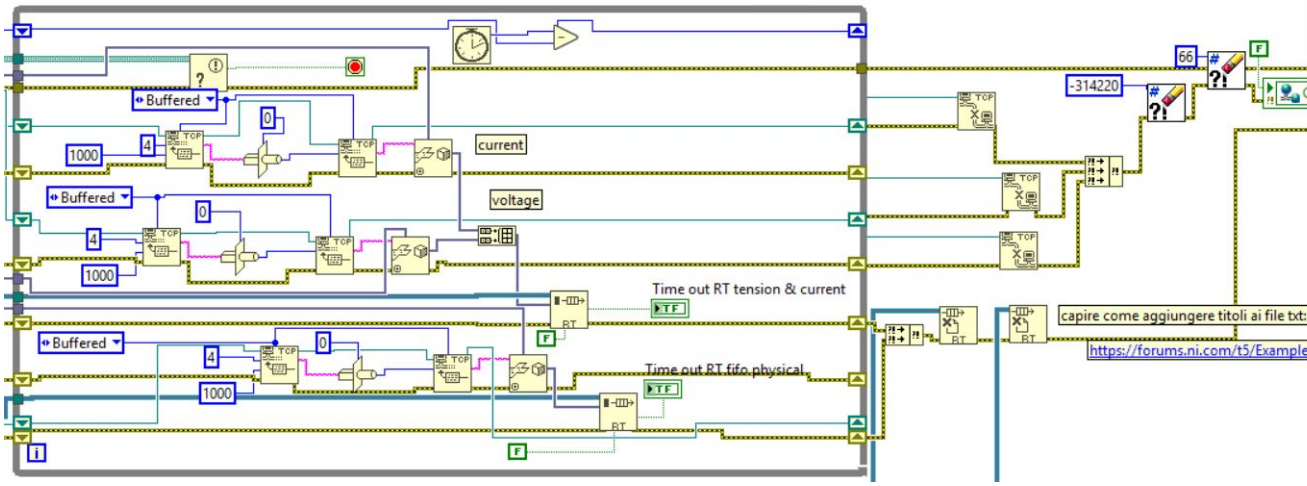


Figura 4.42: TCP/IP: While Loop reader

Per quanto riguarda il ciclo di ricezione (fig. 4.42), analogamente al precedente paragrafo, i dati arrivano suddivisi in tre canali, uno per le correnti, uno per le tensioni e uno per i fisici. Dal formato stringhe vengono riconvertiti in FXP e poi, i primi due vengono concatenati in unico array 1D, mentre l'altro viene inviato direttamente al rispettivo RT FIFO Write.

4.3.2 Visualization and Log Loop

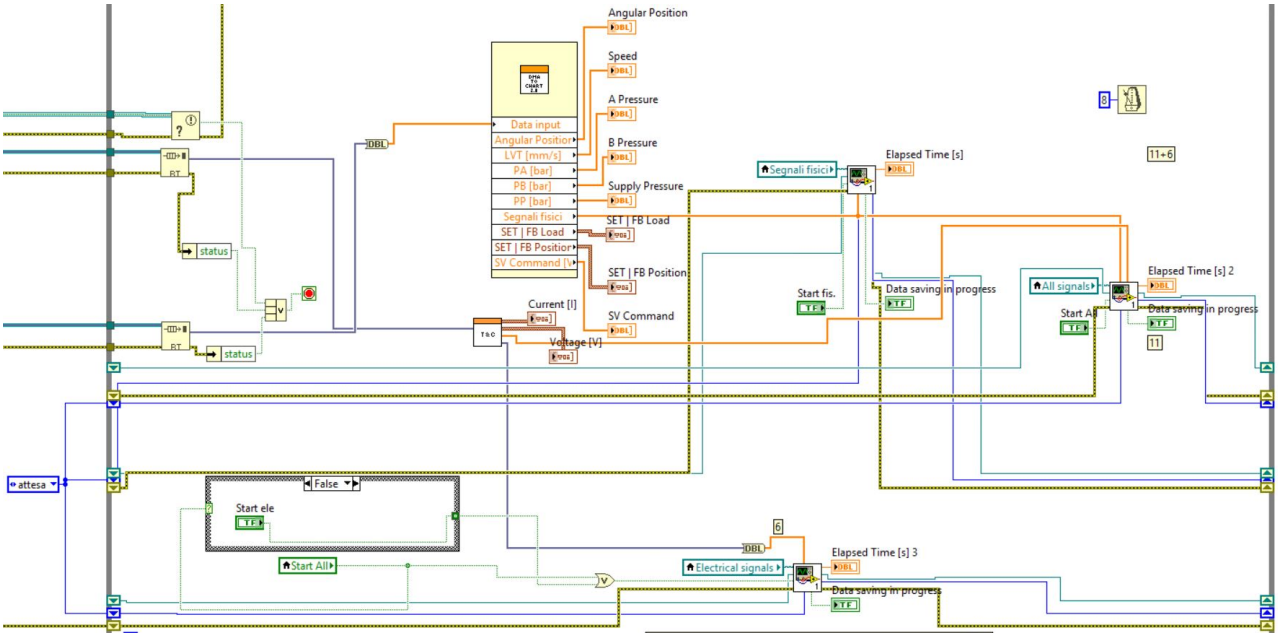


Figura 4.43: Ciclo di visualizzazione e salvataggio dei dati

Il ciclo di visualizzazione e salvataggio dati è uno dei While Loop principali di questa VI. Partendo da sinistra della figura 4.43 si trova il Notifier per bloccare il Loop e i due RT Fifo Read, dai quali escono, rispettivamente partendo dall'alto 48000 e 88 dati ogni 8 millisecondi.

Visualizzazione

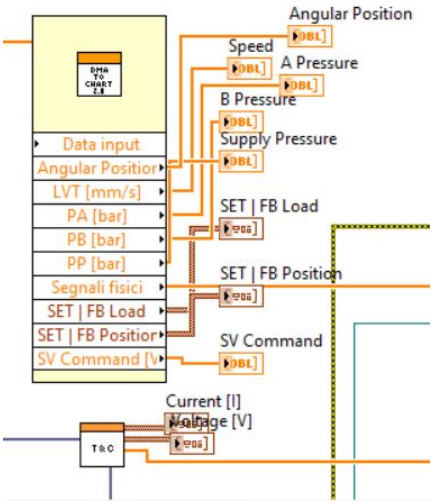


Figura 4.44: Le due SubVI per la visualizzazione dei dati: in alto per i fisici, in basso per gli elettrici

Nella prima SubVI in alto entra (fig. 4.44)il vettore dei segnali fisici, il quale viene diviso per ogni sensore e mandato ognuno nel rispettivo grafico (figura 4.45)
.

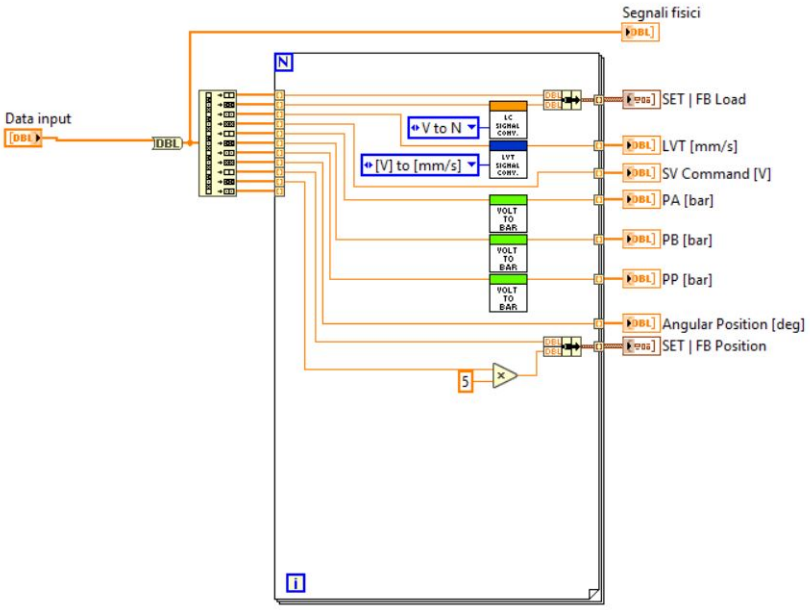


Figura 4.45: SubVI per la visualizzazione dei dati fisici

In quella più in basso (sempre di fig. 4.44)invece la situazione si complica leggermente. Infatti, osservando l'immagine 4.46 si vede come non ci sia una semplice divisione in 6 vettori, ma come spiegato nel ciclo di sincronizzazione (paragrafo 4.2.3), i dati devono essere decimati per essere visualizzati in un grafico altrimenti la rappresentazione risulterebbe troppo "rapida" (in quanto si dovrebbe riprodurre 48000 dati ogni 8 millisecondi) e quindi difficile da osservare.

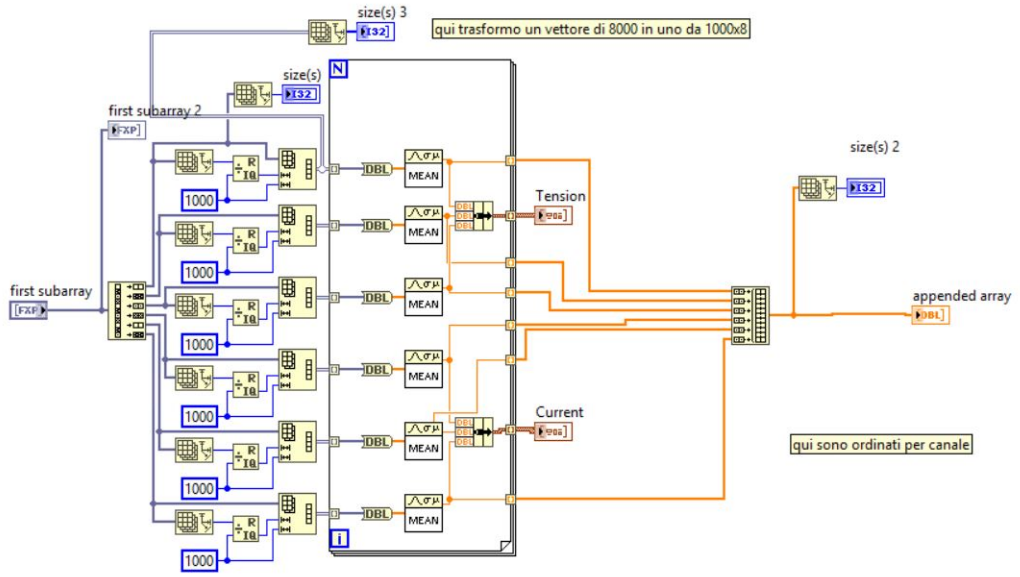


Figura 4.46: SubVI per la visualizzazione ed elaborazione dei dati elettrici

Per questo motivo si procede con la media dei singoli vettori, soltanto che a differenza della SubVI "Mean" (figura 4.20) dove venivano elaborati 6000 dati ogni millisecondo, qui ne arrivano 48000 ogni 8 millisecondi. Per questo, la prima funzione che si incontra (Decimate Array), serve per dividere il vettore di 48000 dati in 6 array da 8000 ciascuno. Successivamente, ognuno di questi viene separato in 8 parti da 1000 e viene fatta la media su tale numero. Così facendo, alla fine del processo vengono inviati al grafico 48 dati ogni 8 millisecondi come nel Main RT.

Log

Il salvataggio dei dati viene affidato a queste tre SubVI (fig. 4.47), le quali possono lavorare sia da sole, sia contemporaneamente.

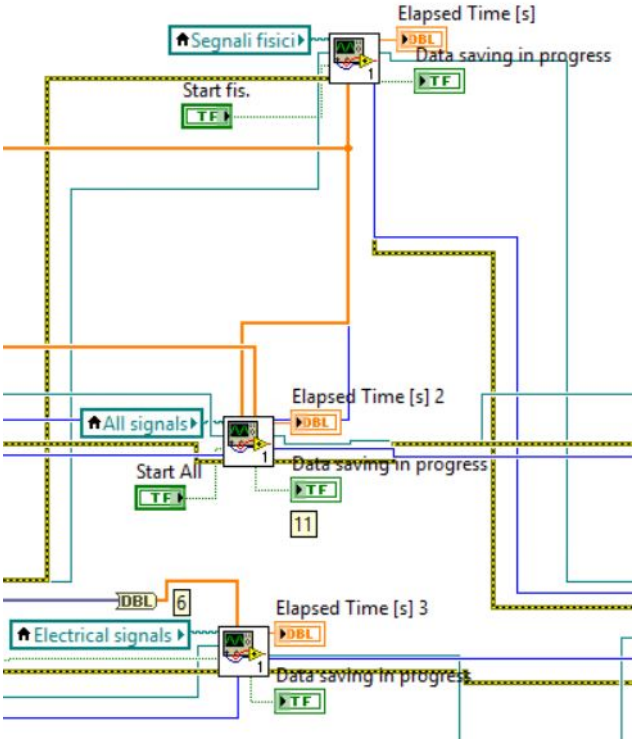


Figura 4.47: SubVI per il log dei dati

Ognuna ha un compito ben preciso, ovvero, quella in alto si occupa del log dei segnali fisici, quella in basso dei segnali elettrici "puri" e quella al centro invece si occupa di salvare tutti i dati, ovvero fisici, elettrici decimati e non.

Aprendo quest'ultima, si può notare come sia composta da una prima parte in cui si ha un ordinamento dei vettori (figura 4.48) ed una seconda in cui è presente un case selector (figura 4.49).

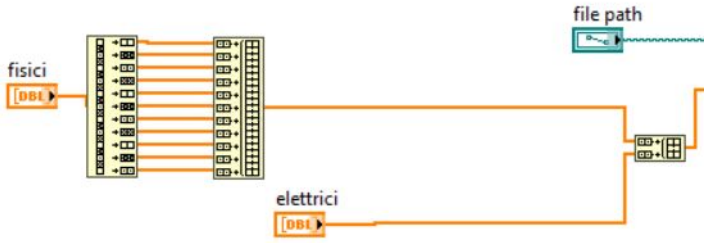


Figura 4.48: SubVI per il log dei dati: Ordinamento dei vettori

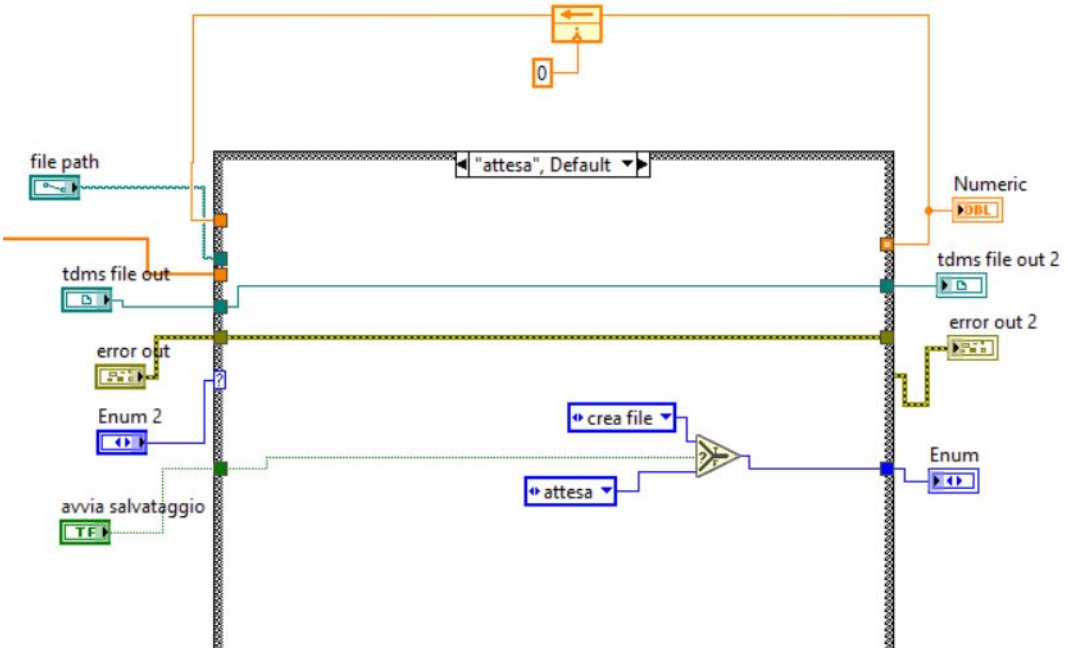
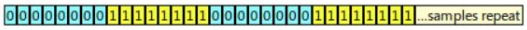
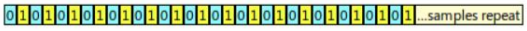


Figura 4.49: SubVI per il log dei dati: Case selector

Nella prima si trova sia il vettore dei segnali fisici (che passa per "Decimate 1D Array" e poi in "Build Array") e il vettore dei segnali elettrici decimati. Il motivo per il quale il primo array viene elaborato in tale modo è per una questione di ordinamento dei dati. Per spiegare meglio questo concetto, si riporta un esempio (figura 4.50).



(a) Ordinamento per canale dei dati



(b) Ordinamento temporale dei dati

Figura 4.50: Fonte: LabView Help, TDMS Set Channel Information Function

Si ipotizzi di acquisire due tipologie di dati diversi, il segnale 0 (azzurro) e il segnale 1 (giallo) (fig. 4.50). Ogni microsecondo nel FPGA il sensore legge questi due dati e li predispone all'interno di un unico array e li invia all'interno del DMA FIFO. Ipotizzando che non ci siano latenze di alcun tipo, nel Main RT, giungono quindi 2000 dati al millisecondo. Se si prova ad osservare quest'ultimo vettore, si può notare come i numeri vengano disposti come nella figura 4.50b, in quanto ad ogni acquisizione (quindi ogni microsecondo), si salva un dato del segnale azzurro ed uno del segnale giallo. Questo ordinamento è quello che il vettore dei dati fisici possiede prima di entrare all'interno di Decimate Array e Concatenate Array (figura 4.50) e viene definito "temporale".

I dati elettrici, al contrario, sono ordinati secondo la configurazione "per canale" (4.50a), poichè questa operazione è stata effettuata precedentemente (vedi immagine 4.46).

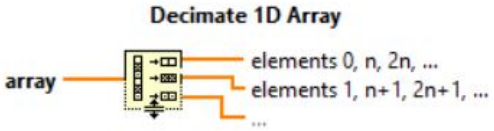


Figura 4.51: Decimate 1D Array

In particolar modo, Decimate Array è la responsabile di questa suddivisione dei dati, come si vede in 4.51, mentre Build Array li concatena soltanto in un unico vettore monodimensionale.

Quindi ritornando a 4.48, viene fatta questa operazione appunto per avere entrambe le tipologie di dati orientate nello stesso stesso modo.

Successivamente si trova un case selector con quattro finestre ovvero:

- Attesa: in questa zona si attende soltanto che il pulsante "Start Acquisition" presente nel Front Panel sia premuto (immagine 4.49 per il Block Diagram e la figura 2.53 per il Front Panel).

- Crea file: viene creato il file, dove si specificano le varie classi che lo compongono (ovvero i diversi segnali presenti), l'ordine con cui i dati arrivano (interleaved e non-interleaved, ovvero si intende: ordinati temporalmente o per canale), il formato e il numero di elementi per ogni classe indicato in basso (quindi 88 dati fisici sommati a 48 dati elettrici decimati, divisi per un totale di 17 canali, sono 8 elementi a canale).

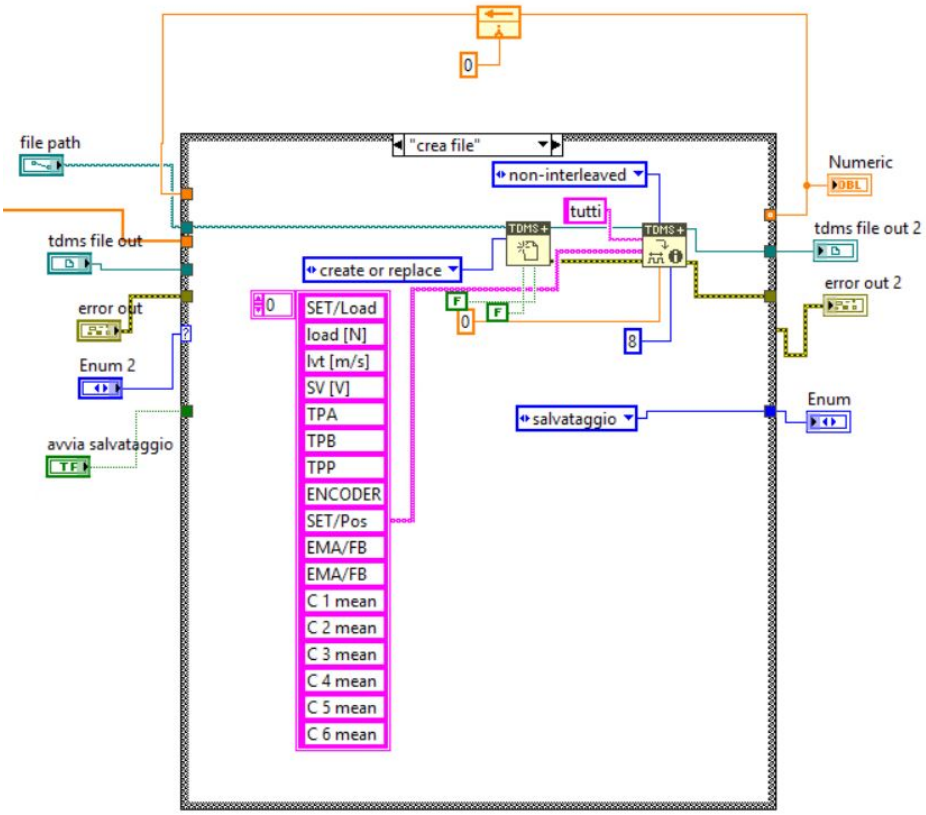


Figura 4.52: Crea file

- Salvataggio: si procede nell'operazione fino a quando il pulsante "Start Acquisition" non viene premuto nuovamente.

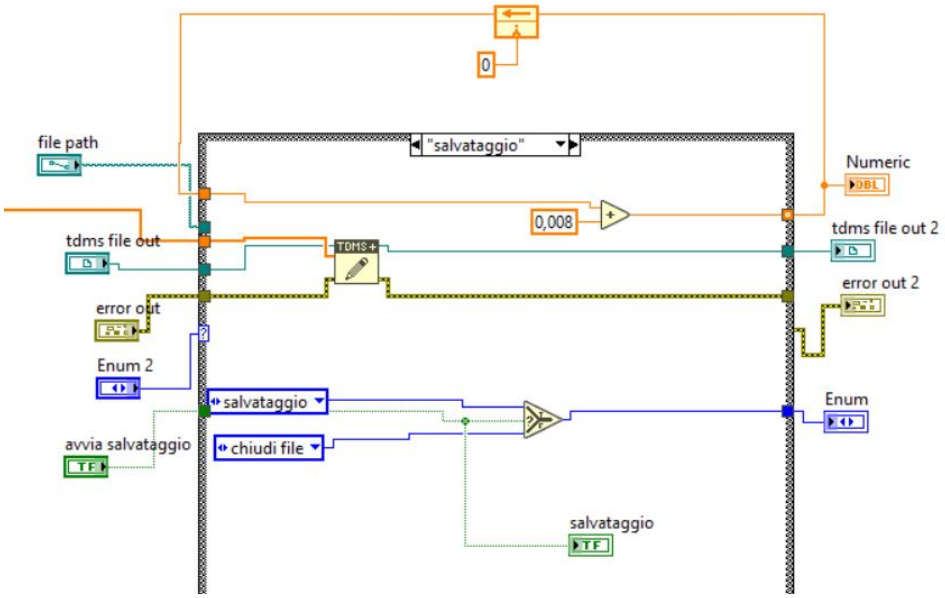


Figura 4.53: Salvataggio

- Chiusura file: una volta terminato, il file viene chiuso e si ritorna nella configurazione di attesa.

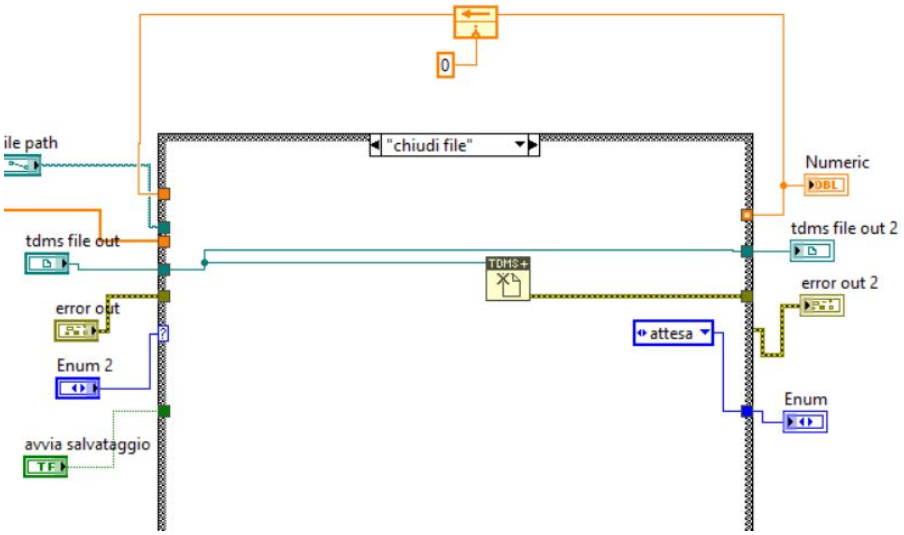


Figura 4.54: Chiusura

Dato che questa SubVI appena descritta, si occupa di salvare i vettori fisici e gli elettrici decimati, mancherebbero tutti i dati elettrici non elaborati.

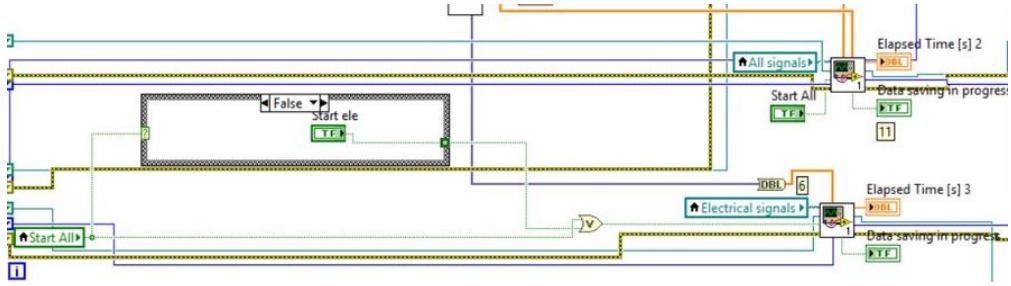


Figura 4.55: Spiegazione case selector

Per questo si è inserito un case selector prima di arrivare alle SubVI:

- Nel caso in cui venga premuto il tasto "Start Acquisition" nella sezione All Signals (figura 2.53), viene inviato un segnale True ad entrambe le SubVI della figura 4.55, quindi viene salvato tutto.
- Invece se venisse premuto soltanto il pulsante dei segnali elettrici, allora verrebbero salvati solamente loro.

Parentesi: Notifier Vs Local Variables

Come già visto, nel Main RT e nel Main PC, vengono utilizzate le funzioni Notifier per lanciare gli stop dei vari loop mentre nel Main FPGA sono rimaste le local variables.

Di seguito le differenze fra le due funzioni:

Notifier

Un Notifier (fig.4.56) in LabVIEW è un meccanismo di comunicazione asincrono tra diversi loop o parti di un programma. Consente la trasmissione di messaggi tra processi paralleli senza la necessità di utilizzare variabili globali o altre forme di condivisione di dati. Un Notifier può avere uno o più lettori (loop o parti del programma) che ricevono e processano i messaggi.

Local Variable

Una Local Variable (fig.4.57) in LabVIEW è un meccanismo di condivisione di dati tra parti di un programma, spesso utilizzato per sincronizzare o condividere informazioni tra loop paralleli. Consente a un loop di scrivere o leggere dati in un'area di memoria condivisa con altri loop.

La prima, rispetto alla seconda, nonostante sia un po' più complessa da implementare, in quanto ha un layout simile a quelle delle Queue, ma permette una sincronizzazione

migliore. Ogni messaggio viene elaborato da un destinatario alla volta, a differenza invece delle Local Variables che accedono insieme ad una memoria comune, potendo causare così delle *race conditions*. Quest'ultime sono situazioni indesiderate che si verificano quando due o più variabili concorrenti tentano di accedere e manipolare dati condivisi senza una sincronizzazione adeguata. Infatti, avendo molti input e output, a causa della presenza di timeout, errori e stop, sono un terreno fertile per il verificarsi di questo fenomeno.

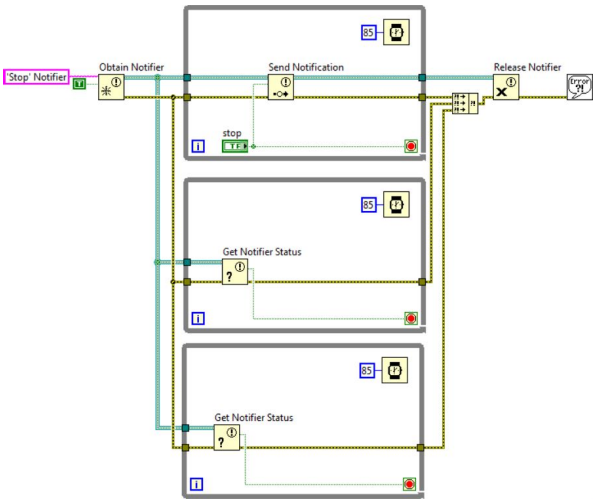


Figura 4.56: Struttura dei Notifier per fermare dei While loop che viaggiano in parallelo

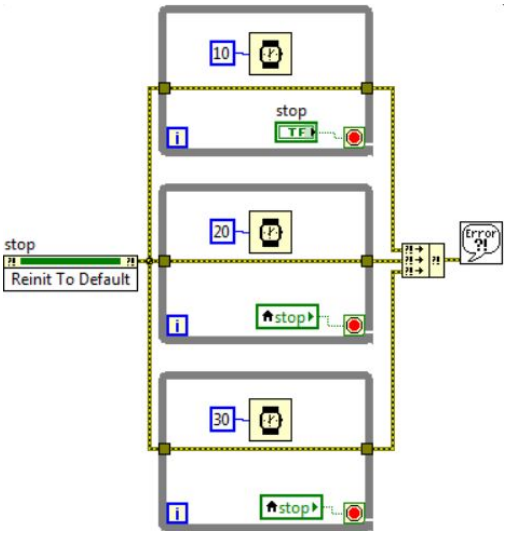


Figura 4.57: Struttura dei Local Variables per fermare dei While loop che viaggiano in parallelo

Capitolo 5

Verifica dei segnali e primo test del banco con il nuovo codice

5.1 Verifica dei segnali

Una volta ultimato il codice, lo step seguente è stato quello di verificarne la coerenza e controllare l'attendibilità dei segnali. Per fare ciò si è proceduto nel simulare l'acquisizione attraverso l'utilizzo di un generatore di segnale e di un oscilloscopio.

Figura 5.1: Postazione adottata per verificare i segnali: in basso l'oscilloscopio e, sopra al rack di acquisizione, il generatore di segnale.

Il test è stato strutturato nel seguente modo: il generatore di segnale è stato impostato in modo tale da creare un'onda di sinusoidale con una frequenza di 1 kHz (se ne sarebbero potute creare di svariate forme, anche se ai fini del test non sarebbe cambiato molto).

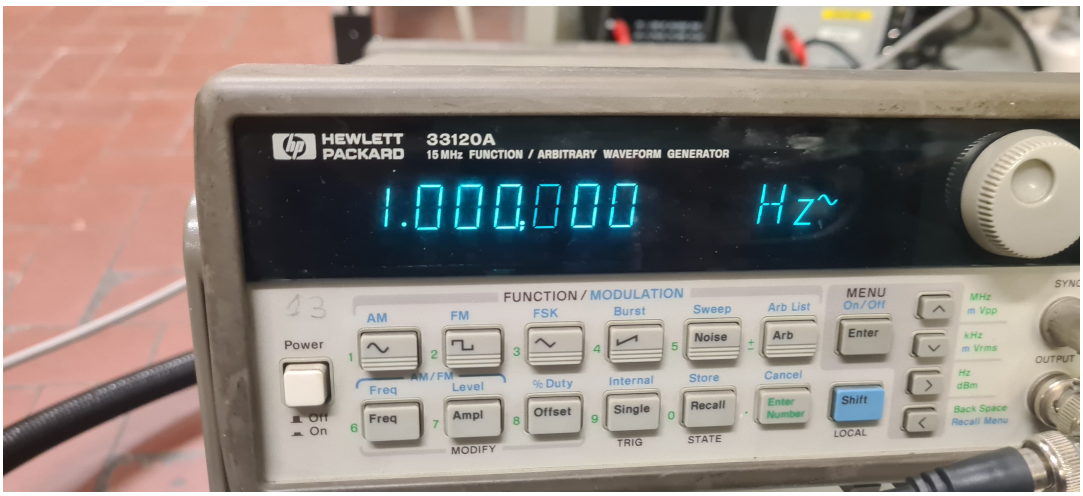
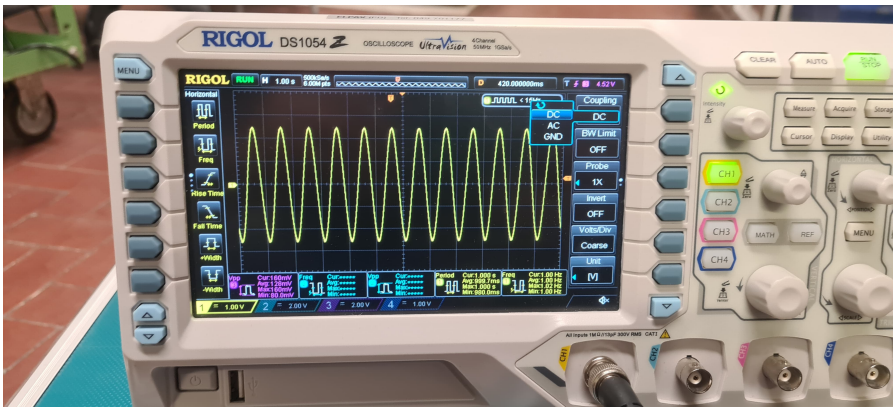
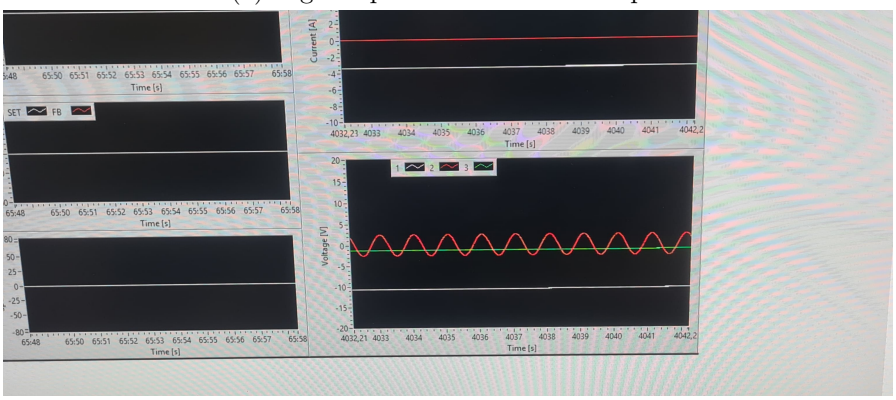


Figura 5.2: Generatore di segnale: onda sinusoidale ad 1 Hz.

Quest'ultimo è stato poi collegato sia all'oscilloscopio sia ai moduli NI 9223 della CompactRio. Così facendo si è andato a controllare se i segnali generati dal generatore fossero identici nei due dispositivi, così da confermare l'esattezza del codice. Questa operazione è stata effettuata per tutti i canali di tensione e corrente.



(a) Segnale presente nell'oscilloscopio



(b) Segnale presente nel Front Panel del codice LabVIEW

Figura 5.3: Dispositivi a confronto: entrambi riportano lo stesso segnale, sinonimo di un corretto funzionamento del test.

5.2 Primi test al banco con il nuovo codice

La prima verifica fatta è stata quella di andare a verificare se effettivamente la frequenza di acquisizione fosse 1 Mhz. Per farlo si è preso un file di salvataggio ".tdms" di un test effettuato, si è caricato all'interno di Matlab e si è andato a vedere se fossero stati salvati 1.000.000 di dati al secondo. Prendendo in considerazione il primo file, il numero di punti salvati in 17.016 secondi corrisponde a 17016000. Come facilmente si può intuire:

$$17.016.000/17{,}016 = 1.000.0000\,\text{dati/s}$$

Successivamente si è proceduto con il test vero e proprio dove si è andato a verificare quale fosse la massima velocità dell'attuatore in prova in grado di offrire, in acquisizione, una buona risoluzione dei segnali di corrente. Per farlo si è cercato di variare la velocità dell'EMA mantenendo su di esso un carico costante di 2000 N.

Quindi, la legge utilizzata per descrivere il segnale di forza è rimasta uguale per tutte le prove, in modo tale da averla come riferimento, mentre la legge del moto è variata per ognuna.

Ogni prova è composta da cinque fasi, ma solo la terza è rilevante ai fini dell'analisi. In questa fase, si è impostata una velocità diversa per ogni test, monitorando contemporaneamente i segnali di tensione e corrente.

Il segnale di forza generato ha un andamento trapezoidale (fig. 5.4),

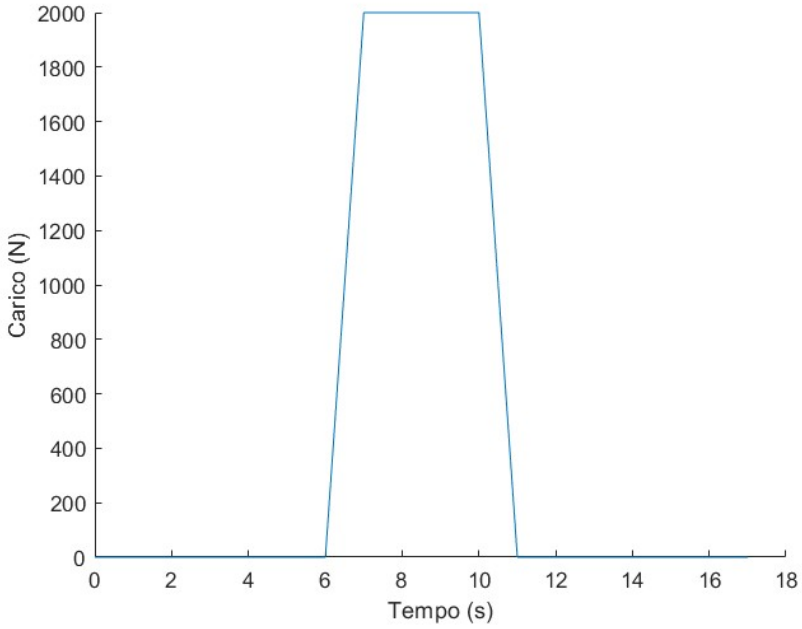


Figura 5.4: Set di forza (N), uguale per tutte le prove.

Avente le seguenti caratteristiche (tab. 5.1):

	Fase 1 (t=1s)	Fase 2 (t=1s)	Fase 3 (t=2s)	Fase 4 (t=1s)	Fase 5 (t=1s)
Legge del moto	Costante (0 N)	Rampa crescente (fino 2000 N)	Costante (2000 N)	Rampa decrescente (fino 0 N)	Costante (0 N)
Carico [N]	0	0->2000	2000	2000->0	0

Tabella 5.1: Descrizione delle fasi con relativi carichi e legge del moto

Al contrario del controllo forza, la legge del moto cambia ad ogni test (tredici in totale). Si è partiti da una velocità di 20 mm/s fino ad arrivare a 80 mm/s, con un incremento di 5 mm/s ad ogni prova.

Si riportano tre esempi, uno con una velocità nella terza fase di 20 (5.5) mm/s, uno di 40 mm/s (5.6) e l'ultimo di 80 mm/s (5.7).

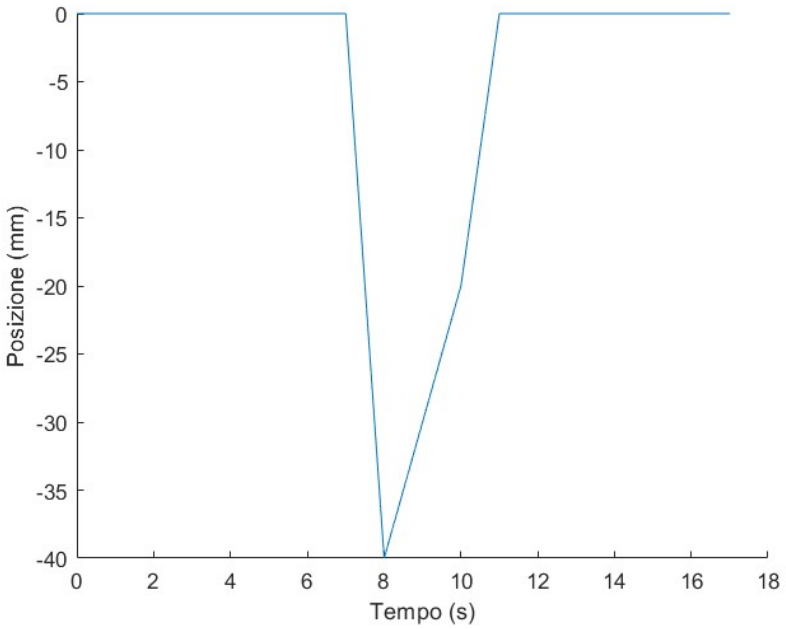


Figura 5.5: Set di posizione (fase tre a 20 mm/s)

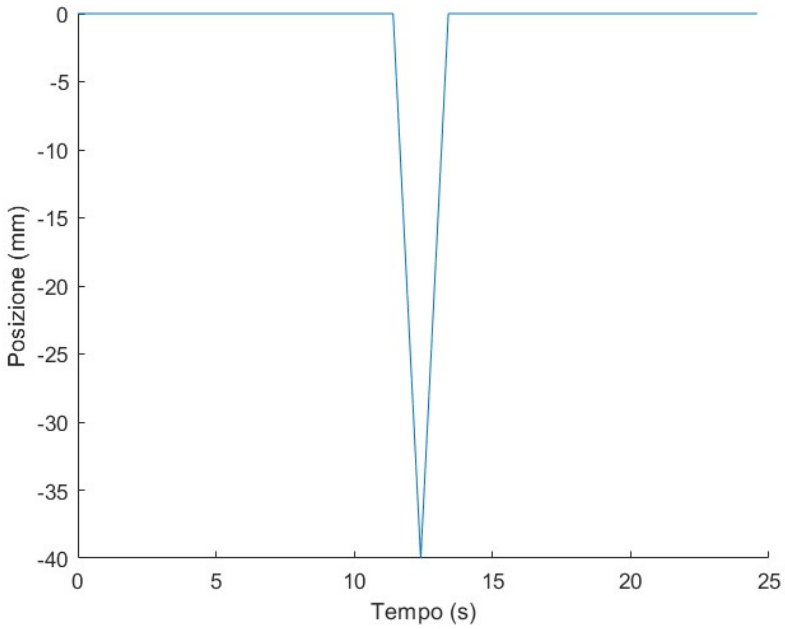


Figura 5.6: Set di posizione (fase tre a 40 mm/s)

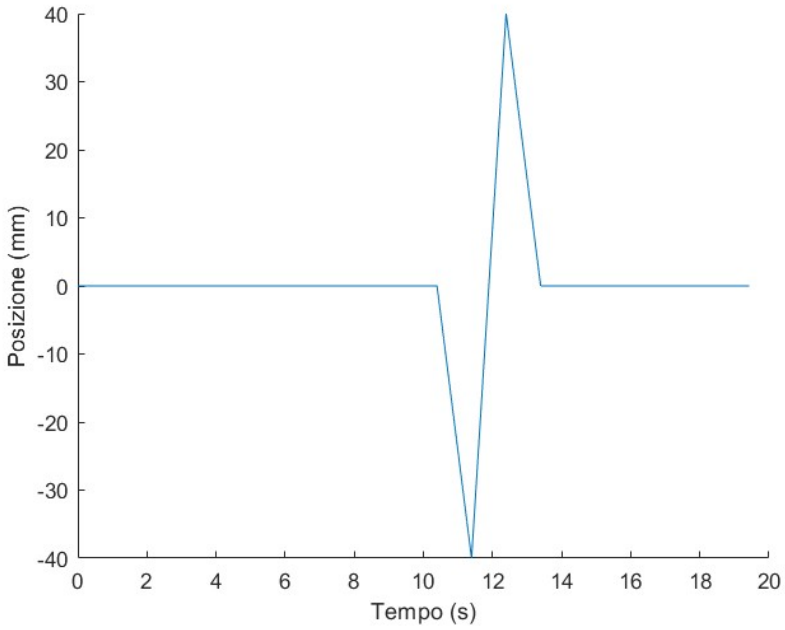


Figura 5.7: Set di posizione (fase tre a 80 mm/s)

Le tabelle riferite a tutte le prove (tab. 5.2, 5.3 e 5.4):

Velocità fase 3 (mm/s)	20 (mm/s)	25 (mm/s)	30 (mm/s)	35 (mm/s)	40 (mm/s)
	Posizione (mm)				
Fase 1 (t=1s)	0	0	0	0	0
Fase 2 (t=1s)	0->-40	0->-40	0->-40	0->-40	0->-40
Fase 3 (t=2s)	-40->-20	-40->-15	-40->-10	-40->-5	-40->0
Fase 4 (t=1s)	-20->0	-15->0	-10->0	-5->0	0
Fase 5 (t=1s)	0	0	0	0	0

Tabella 5.2: Tabella contenente i vari parametri delle leggi del moto adottate

Velocità fase 3 (mm/s)	45 (mm/s)	50 (mm/s)	55 (mm/s)	60 (mm/s)	65 (mm/s)
	Posizione (mm)				
Fase 1 (t=1s)	0	0	0	0	0
Fase 2 (t=1s)	0->-40	0->-40	0->-40	0->-40	0->-40
Fase 3 (t=2s)	-40->5	-40->10	-40->15	-40->20	-40->25
Fase 4 (t=1s)	5->0	10->0	15->0	20->0	25->0
Fase 5 (t=1s)	0	0	0	0	0

Tabella 5.3: Tabella contenente i vari parametri delle leggi del moto adottate

Velocità fase 3 (mm/s)	70 (mm/s)	75 (mm/s)	80 (mm/s)
	Posizione (mm)		
Fase 1 (t=1s)	0	0	0
Fase 2 (t=1s)	0->-40	0->-40	0->-40
Fase 3 (t=2s)	-40->30	-40->35	-40->40
Fase 4 (t=1s)	5->0	10->0	15->0
Fase 5 (t=1s)	0	0	0

Tabella 5.4: Tabella contenente i vari parametri delle leggi del moto adottate

Come già accennato, per ogni prova sono stati analizzati i segnali di tensione e corrente attraverso l'utilizzo del software Matlab.

Segnali di corrente

Sono stati visionati i segnali acquisiti ad 1 MHz, 1 kHz e, successivamente, a 8 kHz (ottenuti filtrando quelli ad 1 MHz). Infatti, quest'ultima risulta essere la frequenza del chopper presente all'interno del driver del motore elettrico, responsabile della generazione dei segnali in PWM. Per questo motivo si è applicata una media mobile con lo scopo di ottenere tale frequenza e quindi una visualizzazione più chiara con meno rumori.

Per una velocità di 20 mm/s (ma poteva essere qualunque altra)si sono sovrapposti i dati grezzi acquisiti ad 1 MHz, con quelli filtrati ad 8 kHz. Il motivo di questo test è stato quello di valutare l'efficacia del filtro e evidenziare l'andamento del segnale di controllo in PWM inviato al motore elettrico.

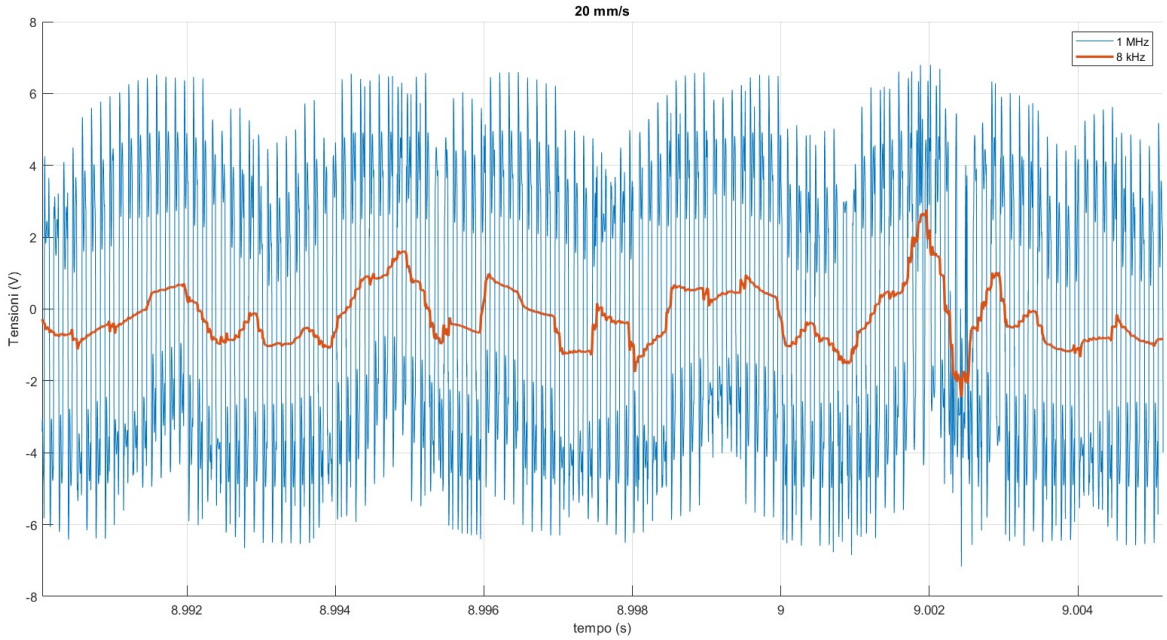


Figura 5.8: Confronto fra il segnale ad 1 MHz e 8 kHz.

Come si può osservare nella figura 5.8, il segnale rosso (8 kHz) segue perfettamente quello blu (1 MHz), a dimostrazione del fatto che il filtro funziona correttamente.

Successivamente, per i segnali di corrente, si è indagata la massima velocità che permettesse di visualizzare correttamente le tre fasi del motore elettrico, ovvero tre segnali sinusoidali sfasati di 120° ciascuno (utilizzando sempre il filtro a 8 kHz)). Come risultato, la velocità massima al di sopra della quale non è più possibile ottenere una risoluzione coerente, è di 40 mm/s (fig. 5.10). Di seguito si riportano alcuni esempi:

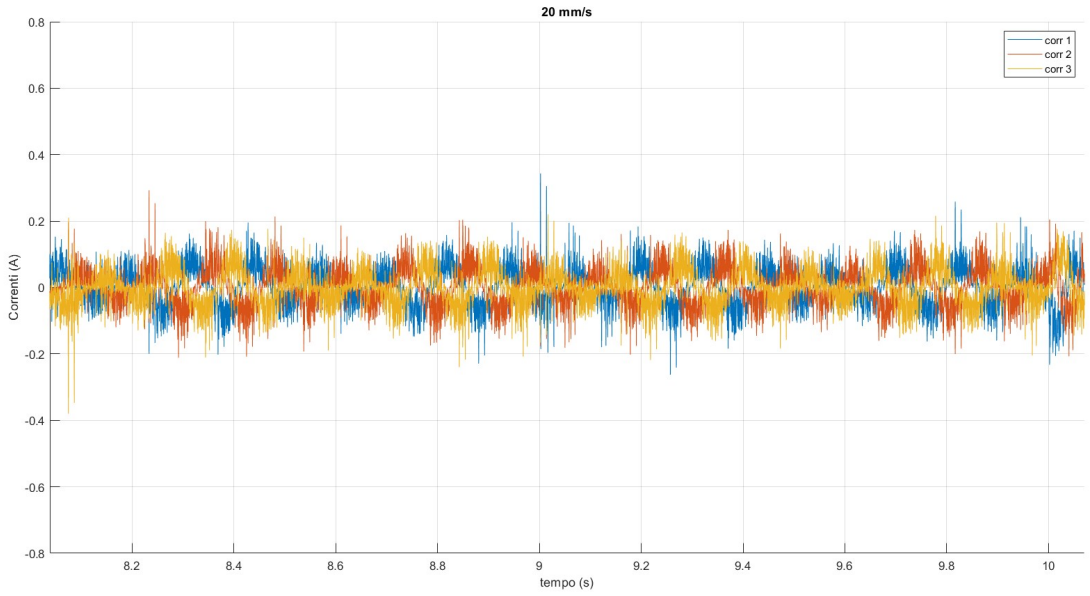


Figura 5.9: Segnale di corrente. Le tre fasi del motore rilevate per una velocità di 20 mm/s.

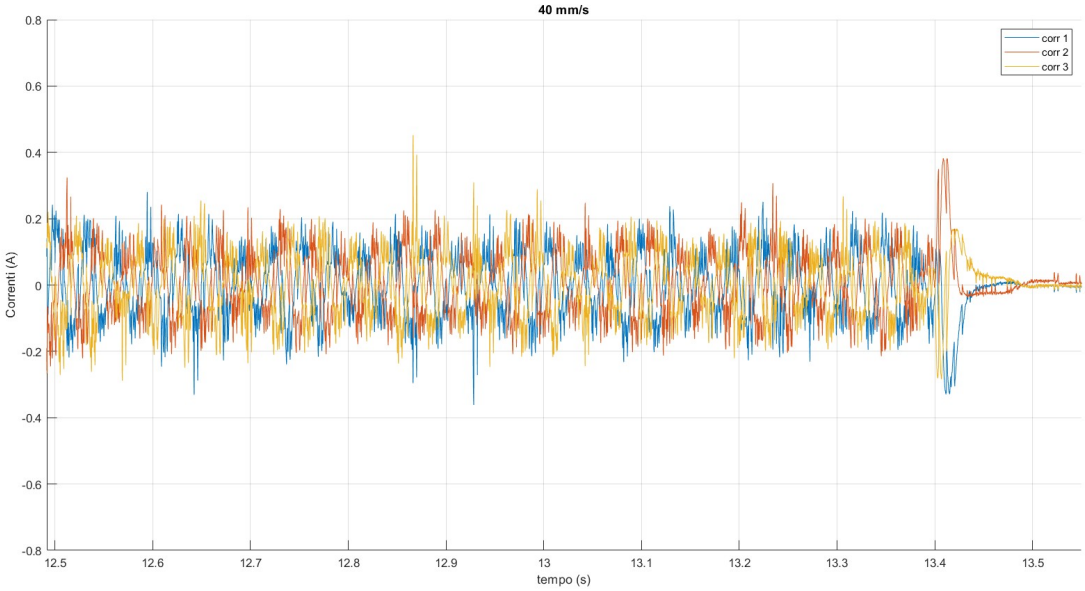


Figura 5.10: Segnale di corrente. Le tre fasi del motore rilevate per una velocità di 40 mm/s.

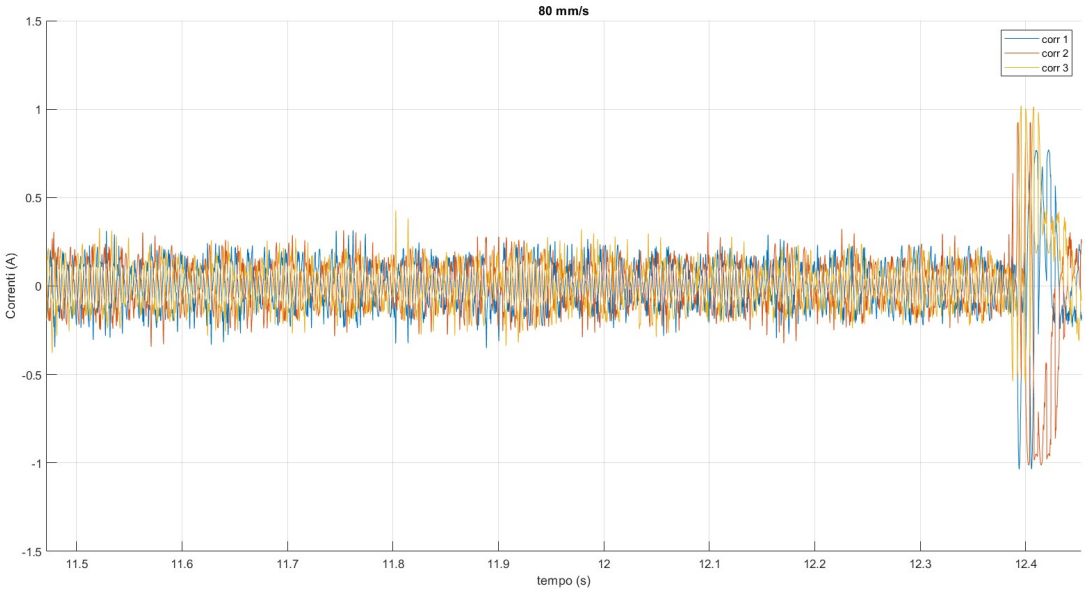


Figura 5.11: Segnale di corrente. Le tre fasi del motore rilevate per una velocità di 80 mm/s.

Osservando i grafici soprastanti, si può notare chiaramente come la miglior risoluzione si ottenga per una velocità di 20 mm/s (fig. 5.9). Per i 40 mm/s si riescono ancora a distinguere le tre fasi mentre al di sopra di tale valore risultano irriconoscibili (es. 80 mm/s, fig. 5.11).

Segnali di tensione

Contrariamente, per quanto riguarda i segnali di tensione, si è notato un comportamento che non si è riuscito ad indagare a causa della mancanza di tempo. Osservando l'immagine 5.12 si vede come la fase 1 sia completamente sovrapposta alla 3, quando in realtà tutte e tre dovrebbero essere sfasate di 120°. Questo è presente per tutte le prove effettuate.

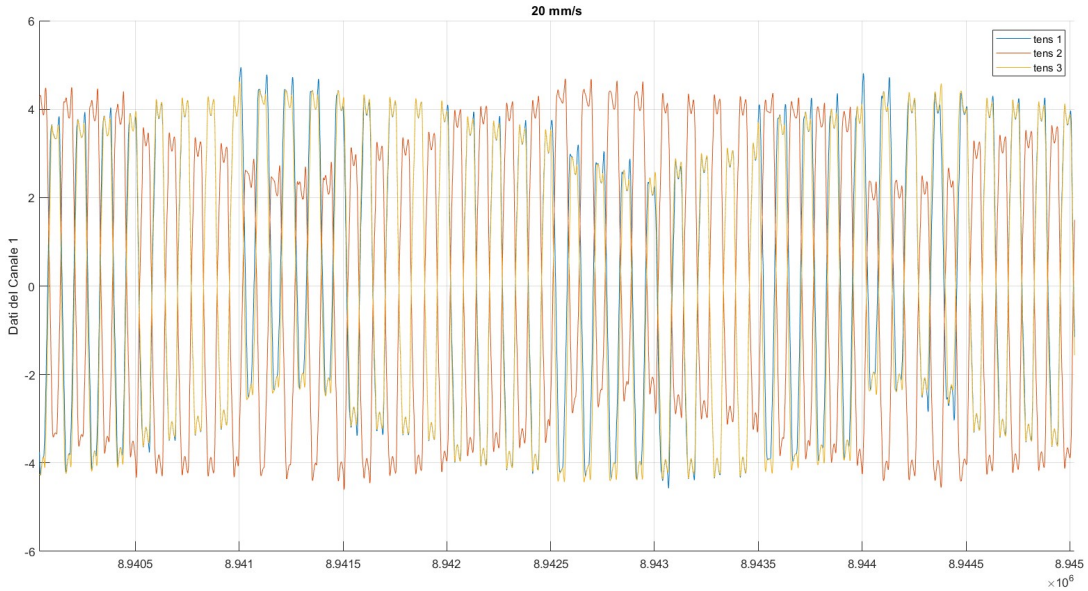


Figura 5.12: Segnale di tensione.

Tale fenomeno sarà argomento di sviluppi futuri.

Capitolo 6

Sviluppi futuri

Oltre all'ultimo fenomeno descritto nel paragrafo precedente, uno dei problemi irrisolti, è quello della presenza di disturbi elettromagnetici durante il funzionamento del banco. Infatti, come anche riportato nella tesi di Romanini Riccardo, "Implementazione Hardware e Software di nuove funzionalità per il Banco Prova Servocomandi Volo" (2021), questo disturbo ha iniziato a verificarsi dopo l'aggiunta del rack per acquisire i segnali di tensione e corrente (figura 6.1).

Figura 6.1: Rack contenente i sensori per misurare i segnali di tensione e corrente

In particolar modo i picchi si verificano quando il driver DEMA viene abilitato e vengono alimentati gli avvolgimenti del motore elettrico.

La frequenza di comparsa è casuale e si manifestano in tutti i grafici presenti nell'interfaccia utente. Infatti è come se l'interferenza rimbalzasse fra i vari sensori, in quanto la sua forma d'onda risulta identica in tutte le rappresentazioni (es. fig. 6.2 e 6.3).

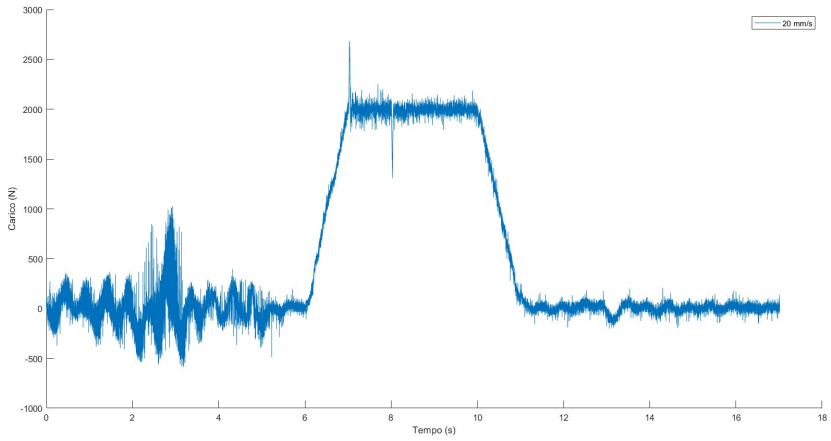


Figura 6.2: Disturbo presente sul feedback del carico (N), in basso a sinistra dell'immagine.

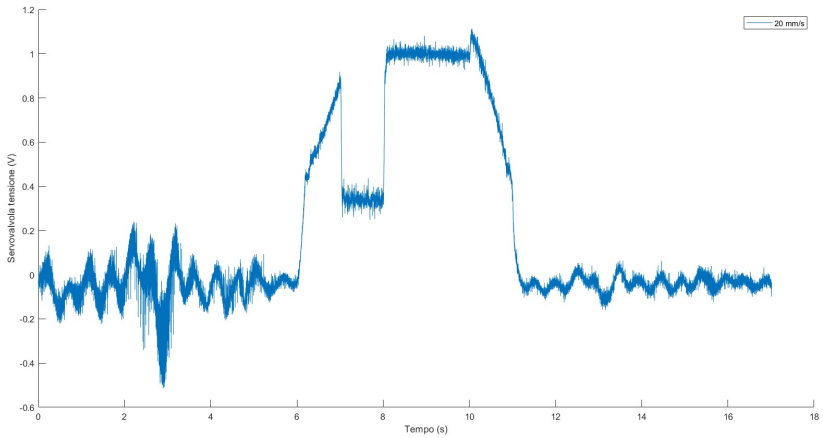


Figura 6.3: Disturbo presente sul feedback della servovalvola (V), in basso a sinistra dell'immagine.

Infine, si consiglia di effettuare ulteriori test in modo valutare effettivamente se l'aggiunta del rack possa essere un elemento utile per lo studio dei motori elettrici.

Capitolo 7

Conclusioni

In conclusione, si riassumono i concetti fondamentali che hanno caratterizzato questo progetto.

Attualmente, i comandi primari di volo sono affidati agli attuatori idraulici i quali, rispetto agli elettro-meccanici, presentano un fattore di sicurezza maggiore. Questi ultimi, oggigiorno, vengono impiegati soltanto per i secondari, come flap, aerofreni e spoiler.

I motivi per cui l'industria aeronautica sta promuovendo in modo sempre più marcato l'utilizzo di questa tipologia di motori, a discapito del concorrente idraulico, sono:

- Eliminare eventuali perdite di olio o altri liquidi corrosivi e infiammabili;
- Ridurre il peso, in quanto viene meno tutta l'infrastruttura idraulica che un motore di quel tipo richiede;
- Meno peso quindi meno carburante bruciato, meno costi di manutenzione, una notevole riduzione dei "ground support equipment" (GSM) e soprattutto meno inquinamento.

Il banco prove per servocomandi di volo, sviluppato nel contesto del programma europeo Clean2sky-ASTIB, ha l'intento di studiare il comportamento degli EMA sotto specifiche condizioni di stress, perseguendo l'obiettivo di raccogliere dati utili per le analisi prognostiche. Queste si basano sull'utilizzo di modelli e metodi predittivi in grado di preventivare il manifestarsi di un guasto, valutare la robustezza del dispositivo, ridurre la necessità di interventi straordinari e semplificare le attività di manutenzione ordinaria, individuando in modo sempre più preciso le cause dei malfunzionamenti.

Nell'ultimo aggiornamento hardware, per ricavare quante più informazioni possibili dagli attuatori in prova, è stato aggiunto al sistema un rack contenente dei sensori di corrente e tensione. Questi hanno il compito di raccogliere i segnali uscenti dal driver DEMA, diretti al motore elettrico. Questa tesi costituisce un ulteriore sviluppo rispetto a lavori precedenti; l'implementazione di un nuovo codice in grado di visualizzare e salvare con una frequenza maggiore i segnali di corrente e tensione, sviluppando un sistema in

grado di escludere latenze, riempimenti di buffer, timeout e sovrascritture. Infatti, grazie al protocollo Typical User-Controlled I/O Sampling operation, si è arrivati ad una velocità di campionamento di 1 MS/s, contrariamente al codice precedente, dove la massima raggiunta era di 350 kS/s.

Inoltre, l'aver impostato il DMA FIFO dei segnali elettrici a 16 bit ha permesso una maggiore velocità di comunicazione tra FPGA e Real-Time ma anche un minor ingombro della rete tra Real-Time e PC.

Per quanto riguarda il codice nel Main RT è stato ottimizzato cercando di renderlo il più preciso possibile, grazie all'utilizzo dei Timed Loop e del doppio ciclo While per la lettura dei segnali elettrici, oltre ad aver implementato tutta la struttura di sincronizzazione dei dati.

Il protocollo RT FIFO e TCP/IP ha permesso una comunicazione efficiente e rapida per trasferire i 6 milioni dati (e più) al secondo, diversamente dai Network Streams, troppo lenti per ottenere tali prestazioni.

L'utilizzo dei TDMS ha permesso di salvare tutti i dati nel PC senza creare alcun tipo di latenza.

Infine, i primi test effettuati hanno dimostrato come la massima velocità, per la quale si riesce ad ottenere una buona risoluzione dei segnali di corrente, è di 40 mm/s. Mentre per i segnali di tensione, si sono verificati fenomeni inattesi, i quali potranno essere analizzati in un lavoro successivo.

La tesi ha impiegato otto mesi. I primi due dedicati allo studio del software e i successivi alla stesura del codice. Lo studio ha reso possibile l'acquisizione di competenze, offerte dalla National Instruments, grazie ai corsi: LabView Core 1, LabView Core 2, LabView RealTime 1, LabView RealTime 2 e LabView FPGA. In aggiunta, è stato possibile conoscere più da vicino la Compact-Rio 9039, i vari moduli, le diverse tipologie di sensori esistenti, il funzionamento di un protocollo di telecomunicazione TCP/IP e soprattutto la gestione di grandi quantità di dati.

Un particolare ringraziamento va all'Ing. Gaidano Matteo che mi ha insegnato e motivato nei momenti più critici del progetto e al Professor De Martin Andrea il quale, grazie al lavoro svolto durante le sue lezioni (sviluppo in ambiente Labview del codice per un servosistema in controllo posizione elettrico), mi ha permesso di prendere ulteriore dimestichezza con il software, di utilizzare una MyRio e di apprendere le nozioni base del mondo della Meccatronica.

Bibliografia

[1] Z. S. Y. W. S. M. Guan Qiao, Geng Liu and T. C. Lim, "A review of electromechanical actuators for more/all electric aircraft systems," *J Mechanical Engineering Science 2018, Vol. 232(22) 4128–4151*, pp. 4128–4133, 2018. [Online]. Available: https://journals.sagepub.com/

[2] S. H. L. K. P. Hussain, Yameen M.; Burrow, "A review of techniques to mitigate jamming in electromechanical actuators for safety critical applications," *International Journal of Prognostics and Health Management*, p. 2, 2018.

[3] B. Airways, "Airbus a380-800," n.d. [Online]. Available: https://www.britishairways.com/content/it/it/information/about-ba/fleet-facts/airbus-a380-800

[4] C. Aerospace, "Primary flight controls trimmable horizontal stabilizer actuators," n.d. [Online]. Available: https://www.collinsaerospace.com/what-we-do/industries/military-and-defense/power-controls-actuation/actuation/primary-flight-controls

[5] F. D. Santis, "L'applicazione delle fibre ottiche nei sistemi di controllo degli aerei," 2020. [Online]. Available: https://it.emcelettronica.com/lapplicazione-delle-fibre-ottiche-nei-sistemi-di-controllo-degli-aerei

[6] . J. G. s. De Martin, A., "Prognostics and health management guidelines for electromechanical actuators," pp. 5–22, 2020.

[7] R. Romanini, *Implementazione hardware e software di nuove funzionalià per il banco prova servocomandi volo*, 2021.

[8] "Programma "clean aviation"," 2021. [Online]. Available: european-union.europa.eu

[9] Valinonline, "Mcs synchronous servo motors," n.d. [Online]. Available: https://valinonline.com/manufacturers/lenze-americas

[10] Ewellix, "Viti a rulli satelliti," n.d. [Online]. Available: https://www.ewellix.com/it/prodotti/viti-a-sfere-e-a-rulli/viti-a-rulli/viti-a-rulli-satelliti

[11] Lenze, "Servo-drive lenze 9400 (dema)," 2007. [Online]. Available: https://gifatrasmissioni.it/img_ins/files/Servo%20drive%20Lenze%209400%20HighLine%20manual.pdf

[12] N. Instruments, "crio-9039 specifications," 2022. [Online]. Available: https://www.ni.com/docs/en-US/bundle/crio-9039-specs/page/specs.html

[13] ——, "Ni 9205 datasheet," 2023. [Online]. Available: https://www.ni.com/docs/en-US/bundle/ni-9205-specs/page/specs.html

[14] ——, "Ni 9218 datasheet," 2023. [Online]. Available: https://www.ni.com/docs/en-US/bundle/ni-9218-specs/page/specs.html

[15] ——, "Ni 9263 datasheet," 2023. [Online]. Available: https://www.ni.com/docs/en-US/bundle/ni-9263-specs/page/specs.html

[16] ——, "Ni 9482 datasheet," 2023. [Online]. Available: https://www.ni.com/docs/en-US/bundle/ni-9482-specs/page/specs.html

[17] ——, "Ni 9401 datasheet," 2023. [Online]. Available: https://www.ni.com/docs/en-US/bundle/ni-9401-specs/page/specs.html

[18] ——, "Ni 9375 datasheet," 2023. [Online]. Available: https://www.ni.com/docs/en-US/bundle/ni-9375-specs/page/specs.html

[19] ——, "Ni 9223 datasheet," 2023. [Online]. Available: https://www.ni.com/docs/en-US/bundle/ni-9223-specs/page/specs.html

[20] DaniSense, "Danisense ds50ub-10v," 2023. [Online]. Available: https://danisense.com/wp-content/uploads/DS50UB-10V-2.pdf

[21] Verivolt, "Verivolt isoblock -v4c," 2023. [Online]. Available: https://www.verivolt.com/shop/isoblock-v-1c-310#attr=385,261,9,227

[22] Trans-Tek, "Trans-tek 0114-0001," n.d. [Online]. Available: https://transtekinc.com/product-categories/linear-velocity/

[23] N. Instruments, "Labview," n.d. [Online]. Available: https://www.ni.com/en/shop/labview.html

[24] ——, *LabVIEW Real Time 1- Partecipant Guide. s.l.: National Instruments.* National Instruments, 2014.

[25] ——, *LabVIEW FPGA - Partecipant Guide. s.l.: National Instruments.* National Instruments, 2014.

[26] ——, "Auto-indexing," 2023. [Online]. Available: https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z000000PAghSAG&l=it-IT

[27] ——, "Why does user controlled i/o sampling acquire faster than an i/o node?" 2023. [Online]. Available: https://knowledge.ni.com/KnowledgeArticleDetails?id=kA00Z0000019Km2SAE&l=it-IT

[28] ——, "Labview fpga module programming reference manual," 2024. [Online]. Available: https://www.ni.com/docs/en-US/bundle/lvfpga-api-ref/page/targets/ni/fpga/menus/fpgacategories/programming/fpgahsio-mnu.html

[29] ——, "crio-9039 specifications," 2022. [Online]. Available: https://www.ni.com/docs/en-US/bundle/crio-9039-specs/page/specs.html

[30] ——, "Comparing common file i/o and data storage approaches," 2023. [Online]. Available: https://www.ni.com/en/shop/data-acquisition-and-control/application-software-for-data-acquisition-and-control-category/what-is-diadem/comparing-common-file-i-o-and-data-storage-approaches.html