Politecnico di Torino

DIPARTIMENTO DI INGEGNERIA MECCANICA E AEROSPAZIALE

Corso di Laurea Magistrale in Ingegneria Meccanica

TESI DI LAUREA MAGISTRALE

IMPLEMENTAZIONE DI UN SERVOATTUATORE ELETTROMECCANICO LINEARE A RULLI PLANETARI: PROGETTAZIONE DEI SOFTWARE, MODELLAZIONE E VERIFICHE SPERIMENTALI

Relatore:
Prof. Massimo Sorli

Candidato:
Nicola Ambrosino

Correlatori:
Ing. Piergiorgio Chiavaroli
Ing. Giuseppe Evangelista

Ringraziamenti

Porgo i miei più sentiti ringraziamenti alla mia famiglia, che mi ha sempre sostenuto nei percorsi di studio e non solo.

Ringrazio il Politecnico di Torino per avermi stimolato a tirare fuori sempre il meglio, il Professore Sorli per avermi dato la possibilità di intraprendere questo percorso di tesi, l'Ingegnere Chiavaroli per i suoi preziosi consigli ed il personale del DIMEAS.

Infine, un grazie agli amici che hanno alleggerito i miei anni universitari.

Indice generale

Indice delle figure

Indice delle tabelle

Simbologia

Acronimo	Descrizione
BPSV	Banco prova servocomandi di volo
EMA	*Electro-Mechanical Actuator*
LEMC	*Linear Electro-Mechanical cilinder*
LE	*Lenze Engineer*
RT	*Real-Time*
FPGA	*Field Programmable Gate Array*
OL	*Open Loop*
CL	*Closed Loop*
NI	*National Instruments*
DI	*Digital Input*
DO	*Digital Output*
AI	*Analog Input*
AO	*Analog Output*
I/O	*Input/Output*
GND	*Ground*
GL	*Ground Loops*
AUT	Attuatori di comandi di volo
UI	*User interface*
VI	*Virtual Instrument*
MSMA	*Motor Signal Management & Acquisition*
PWM	*Pulse-Width modulation*
DC	*Direct Current*
AC	*Alternate Current*
FB	*Function Block*

1. Introduzione

La seguente tesi riassume il lavoro svolto presso il laboratorio di "Meccatronica e Servosistemi" all'interno del Dipartimento di Ingegneria Meccanica e Aerospaziale (DIMEAS), inerente alla messa in opera di un attuatore in tecnologia EMA (*Electro-Mechanical Actuator*) da testare sul banco prova servocomandi di volo.
Il BPSV si pone l'obiettivo di condurre prove per attuatori di comandi di volo, sia con l'obiettivo di tipo prestazionale atto a ottimizzare le performance del comando di volo, sia con la finalità di poter valutare lo stato di salute di comandi di volo, effettuando test su attuatori opportunamente degradati artificialmente o sbarcati dopo un percorso di vita operativa. Il banco prova verrà impiegato con lo scopo di applicare e verificare algoritmi di prognostica per AUT.
Il banco dispone di un attuatore lineare idraulico controllato in forza che può essere connesso con un attuatore lineare in prova, ad esempio in tecnologia EMA, controllato in posizione.
L'attuatore idraulico simula le forze aerodinamiche che agiscono come disturbo sul servocomando di volo disposto sulla culla del BPSV controllato in posizione durante una determinata condizione di volo, e genera quindi delle leggi di forza in accordo a un comando di set.
In una precedente tesi è stato testato l'attuatore idraulico con un controllo forza privo di disturbo di velocità poiché non presente un attuatore lineare in prova controllato in posizione e di conseguenza lo stelo era tenuto in posizione da un puntone montato sulla culla del banco.
Per cui, in attesa del servocomando di volo da testare, si effettuano test sull'EMA trattato nella seguente tesi, il quale sarà destinato in futuro ad essere impiegato sul BPSV come dispositivo atto ad introdurre un disturbo di velocità sull'attuatore idraulico in controllo forza.

Il banco prova attuale è mostrato in Figura 1. Sul banco è presente l'attuatore idraulico e un attuatore elettro-meccanico. È trascurato il disturbo forza sull'EMA poiché non è collegato all'attuatore idraulico.

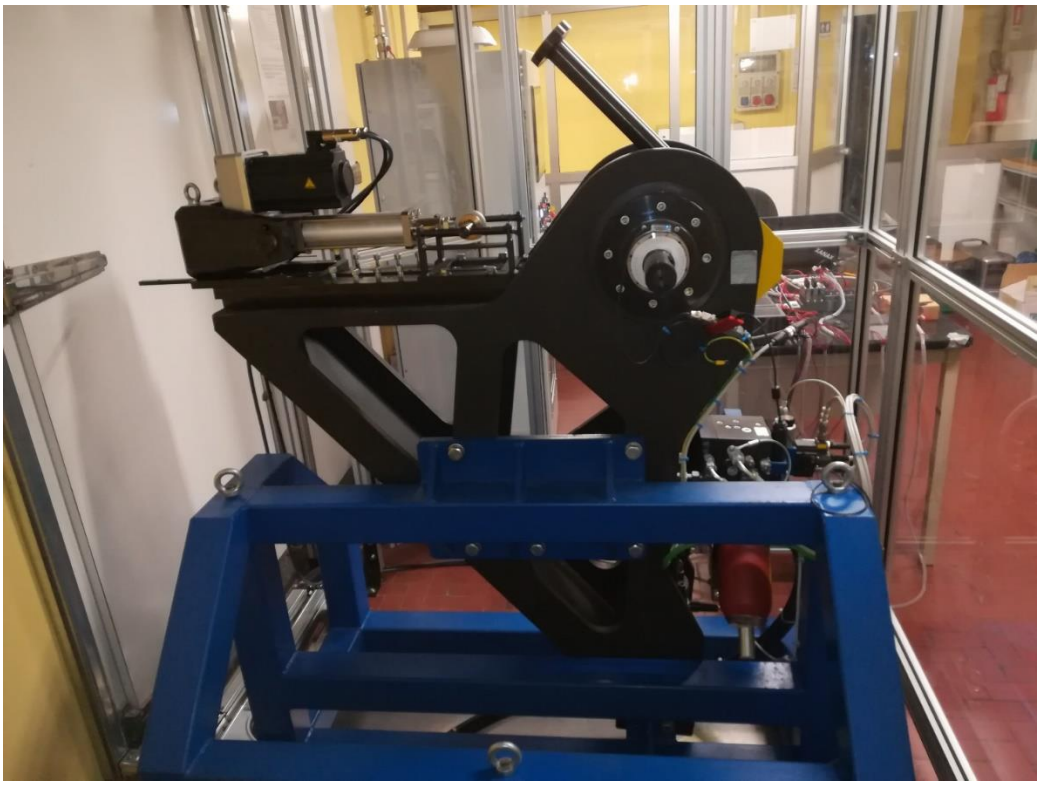


Figura 1: Banco prova servocomandi di volo.

Come si può notare dalla Figura 1 il banco prova risiede all'interno di una struttura protettiva accessibile mediante delle porte di servizio munite di *inter-lock* che, comandati da un armadio elettrico, impediscono ad un operatore l'accesso in caso di pressione idraulica armata. All'esterno è posizionato un Driver che svolge la funzione di azionamento elettrico per il motore, gli alimentatori a 24V che alimentano le apparecchiature elettroniche, un c-RIO che gestisce i segnali, alcuni pulsanti di emergenza e abilitazione e il PC come interfaccia utente. Il sistema consiste in un controllo digitale a più anelli annidati, gestito dal Driver e programmato dall'utente mediante il software Lenze installato sul PC, e dalla gestione dei segnali effettuata sfruttando il software LabVIEW e l'hardware c-RIO della National Instruments (NI).

1.1 Obiettivi

L'obiettivo principale è la messa in funzione dell'EMA e l'esecuzione di prove funzionali volte alla determinazione delle caratteristiche statiche e dinamiche del sistema, per poter giungere ad una taratura del controllo adeguata e lo sviluppo di un modello che approssimi il controllo del motore elettrico svolto dal Driver.
Per arrivare a ciò si è reso necessario svolgere i seguenti passi:

1. Studio del funzionamento del sistema in ogni suo componente (in particolare del Driver) in modo da assemblarlo correttamente e in sicurezza;
2. Cablaggi tra i componenti elettronici e di potenza necessari per il funzionamento dell'EMA in base a quanto studiato nel punto precedente;
3. Studio mediante il software Lenze installato sul PC, in base a quanto riportato nei manuali, della logica di controllo implementata nel Driver;
4. Progettazione di un'applicazione di controllo posizione, mediante software *Lenze Engineer,* in grado di gestire i segnali di *set* in arrivo dal c-RIO e di *feedback* dal motore

mediante il cavo resolver; prove "*offline*" tramite alimentatori regolabili in modo da eseguire test preliminari di funzionamento sulla applicazione creata;

5. Progettazione, mediante LabVIEW, del software di gestione segnali multipiattaforma (PC, Real-Time, FPGA)
6. Esecuzione di prove sperimentali per la valutazione della risposta statica e in frequenza in anello chiuso; taratura della legge di controllo implementata nel Driver;
7. In parallelo a tutto ciò si è sviluppato un modello che potesse approssimare il controllo del Driver a tre anelli annidati e quindi il funzionamento dell'EMA;
8. Confronto del modello EMA creato con le acquisizioni svolte attraverso prove sperimentali sull'EMA.

1.2 Layout generale del sistema

Lo schema riportato in Figura 2 riassume sinteticamente il layout e il funzionamento del sistema adottato.

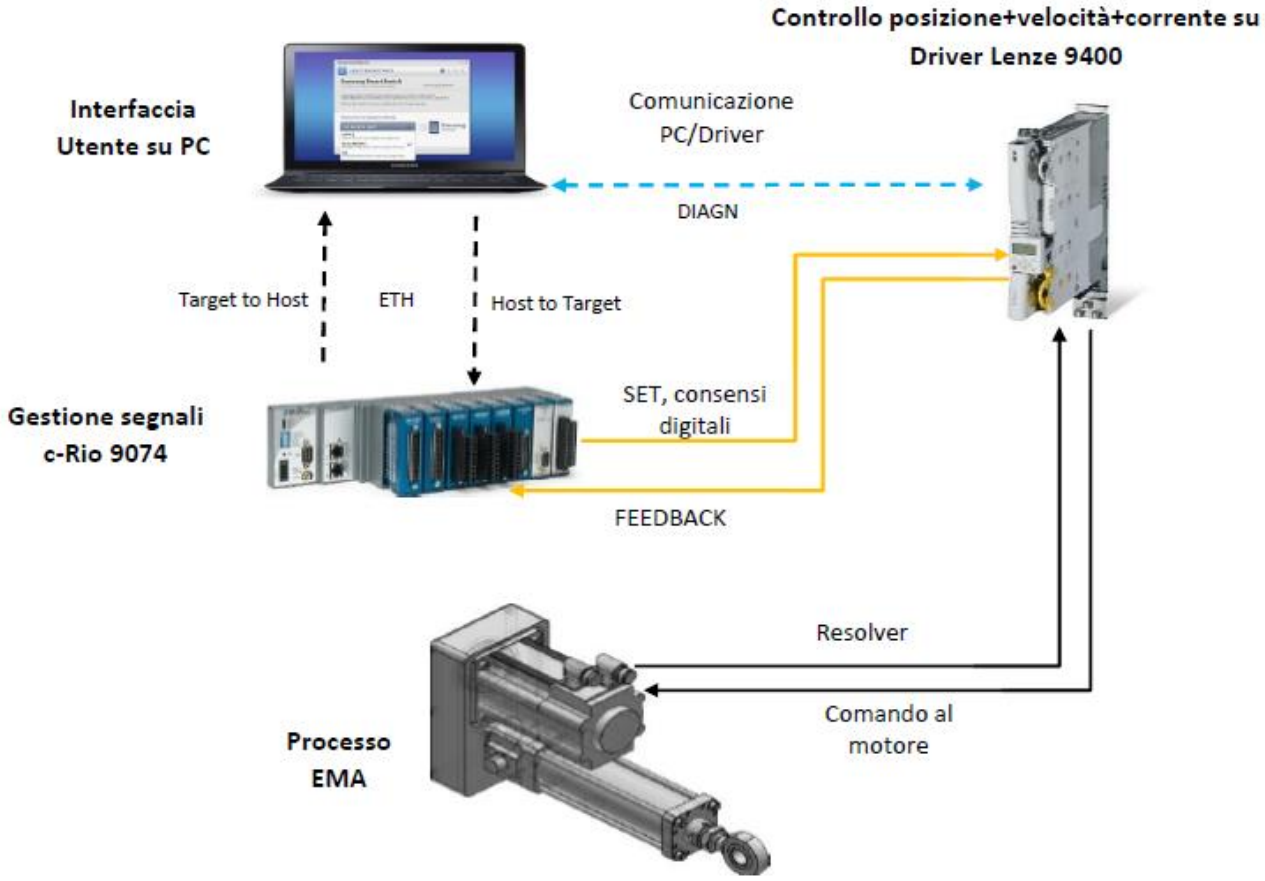


Figura 2: Layout generale del sistema.

L'utente agisce sul PC che svolge il ruolo di interfaccia utente mediante il software LabVIEW e l'applicazione creata mediante esso; tale applicazione quando è 'online' comunica mediante un cavo ethernet (ETH) con il c-Rio, in entrambi i versi (*Target to Host* e *Host to Target*, dove l'Host è il PC ed il Target è il c-Rio). Il c-Rio gestisce i segnali da trasmettere tra PC e Driver attraverso l'applicazione implementata al suo interno e programmata tramite il software LabVIEW su UI. C-Rio e Driver comunicano mediante segnali analogici e digitali in tensione. I primi, eccezion fatta per il segnale di SET di posizione che è diretto verso il Driver, consentono l'acquisizione di alcune

grandezze d'interesse quali la velocità angolare del motore e la corrente assorbita. Tra i segnali digitali invece si citano il comando di abilitazione del Driver, il comando di abilitazione del controllo posizione. Le comunicazioni sono rese disponibili da moduli di comunicazione analogici e digitali ospitati nel *chassis* del c-Rio e da terminali I/O analogici e digitali presenti sul Driver. Il Driver è un servoinverter che effettua sia l'azionamento che il controllo posizione, velocità e corrente, comandando il motore mediante un'applicazione creata tramite il software LE installato sul PC e caricata attraverso un cavo di diagnostica (DIAGN) nel modulo di memoria del Driver, ottenendo un sistema integrato senza necessità di interfaccia utente. Il motore è comandato tramite il cavo motore e il controllo nel Driver elabora il segnale in arrivo dal cavo resolver.

2. Componenti

I componenti del banco prova si suddividono in componenti montati sul banco prova servocomandi di volo, facenti parte dell'EMA e quindi presenti all'interno del box vetrato messo in sicurezza, e in componenti di comando che consistono nell'elettronica e cablaggi elettrici posti all'esterno di tale box e disponibili all'utente.

2.1 Componenti montati sul banco prova

2.1.1 Attuatore elettro-meccanico lineare con servomotore LEMC

L'elemento da testare consiste in un cilindro elettro-meccanico lineare (LEMC) la cui parte elettrica è costituita da un servomotore *Lenze* (sincrono AC) ad interfaccia parallela azionato da un driver, mentre la parte meccanica è formata da un organo di trasmissione a cinghia, il quale trasmette il moto rotatorio con un rapporto di trasmissione unitario ad un sistema vite-madrevite che lo converte da rotativo a lineare grazie ad una vite a rulli planetari SKF.

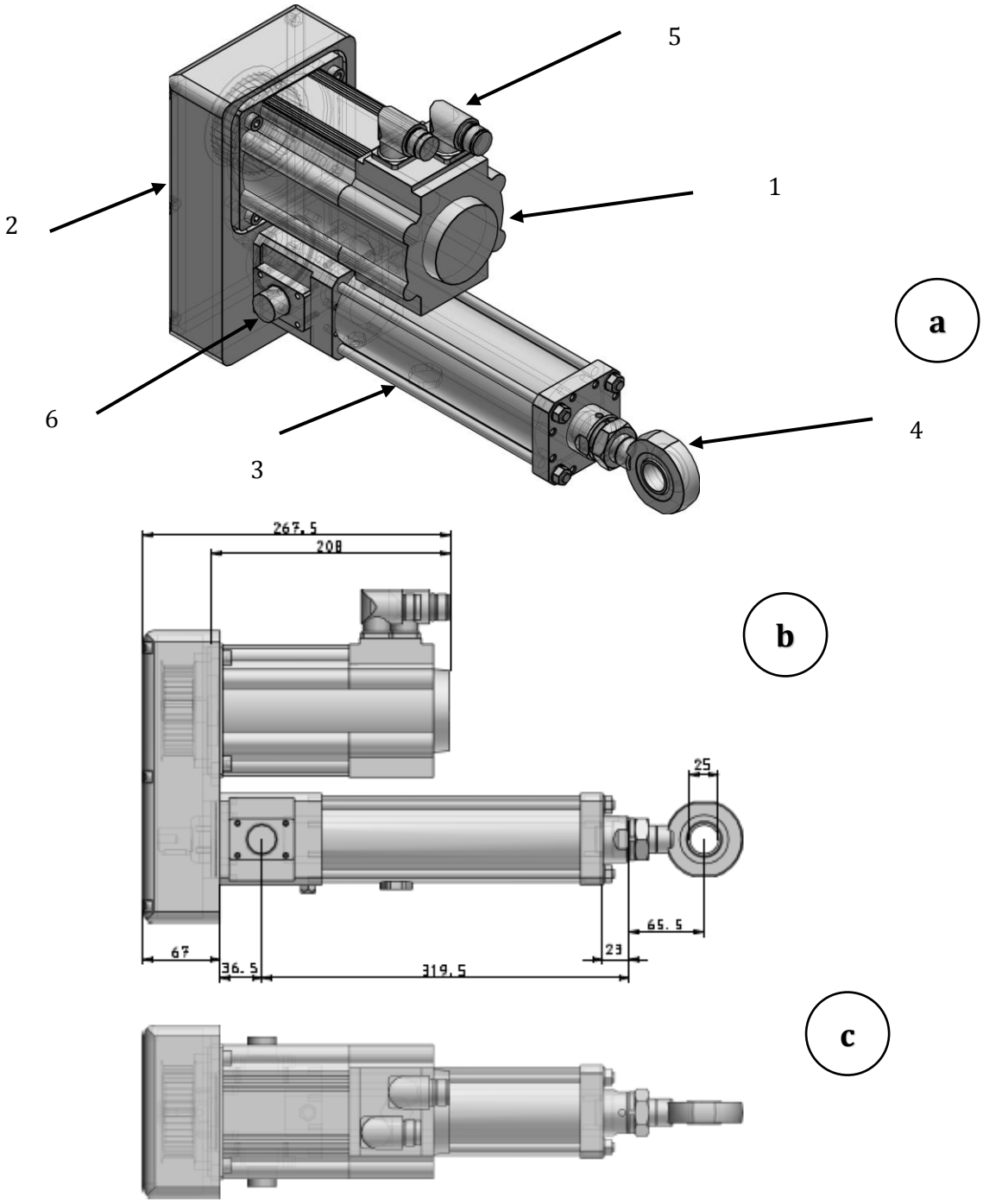


Figura 3: Attuatore elettro-meccanico lineare con servomotore EMA a) in vista 3D b) in vista laterale c) in vista dall'alto.

Numero	Descrizione
1	Servomotore Lenze
2	Trasmissione a cinghia dentata
3	Attuatore lineare a vite con rulli a satelliti SKF
4	*Rod end*
5	*Plug* cavo motore e resolver
6	Perni appoggio LEMC

Tabella 1: Elementi principali LEMC

Come detto in precedenza, l'EMA è montato sul BPSV e presenta i componenti essenziali mostrati in Figura 3 ed elencati in Tabella 1. Inoltre è stata progettata una struttura che permetta l'inserimento del LEMC su tale banco prova che consiste in un supporto fissato sulla testa del banco, il quale sorregge il l'attuatore mediante due boccole accoppiate con i perni di appoggio e dall'altro lato una struttura collegata al *rod end* la quale, oltre a sostenere l'attuatore, svolge anche la funzione di dispositivo anti-rotazione, fondamentale poiché l'attuatore scelto non dispone di tale sistema integrato. I dispositivi in questione sono visualizzabili in Figura 4.

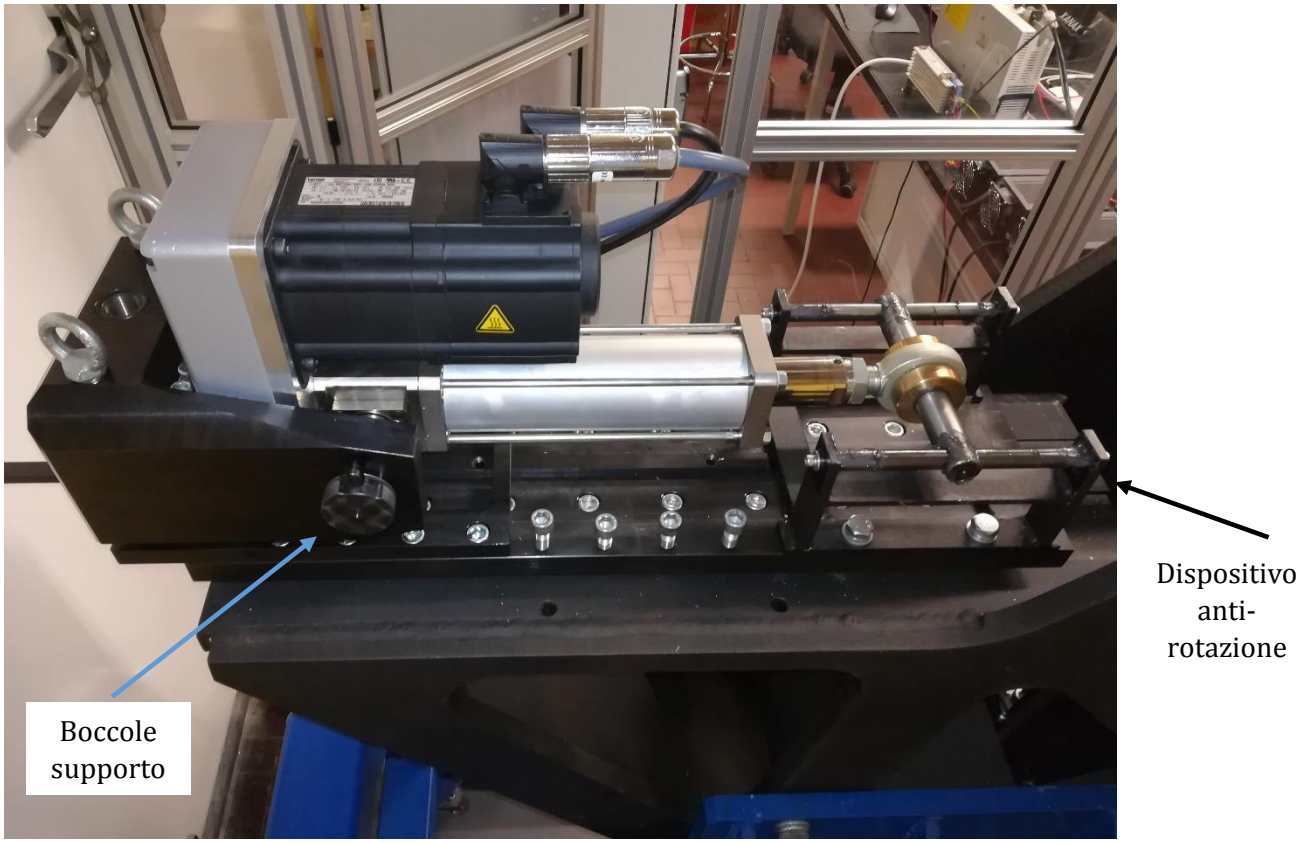
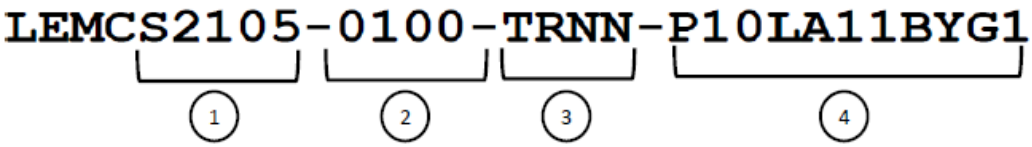


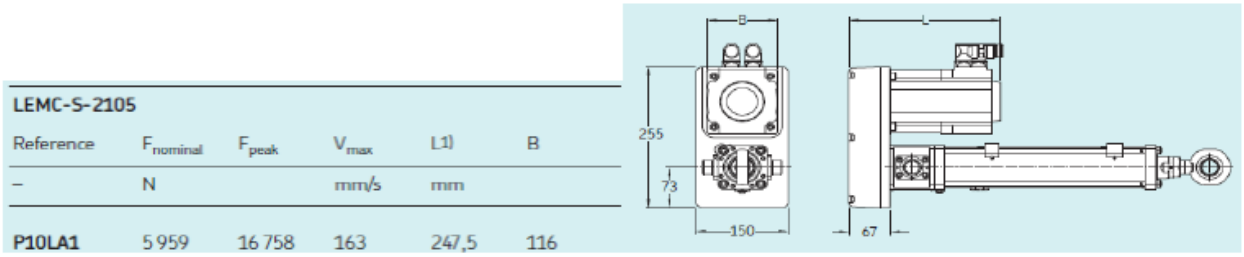
Figura 4: LEMC e supporti montati sul banco prova

L'identificativo di targa del LEMC è presentato in Tabella 2.



N.°	Descrizione
1	S=servomotore , 21=diametro [mm] vite , 05= passo vite 5 [mm].
2	100= corsa dell'attuatore [mm].
3	T= perni come supporto inferiore , R= presenza del *rode end* , N=dispositivo anti-rotazione non incorporato , N= sensori di fine corsa non presenti.
4	P= LEMC in configurazione ad interfaccia parallela , 10= rapporto 1:1 della trasmissione a cinghia , LA1= vedi Figura 5 , 1= resolver , B= freno a DC 24V , Y= driver compreso , G= comunicazione Profibus come fieldbus , 1= lunghezza dei cavi motore e resolver 5 [m] .

Tabella 2: Identificativo di targa del LEMC.



LEMC-S-2105

Reference	$F_{nominal}$	F_{peak}	V_{max}	L1)	B
–	N		mm/s	mm	
P10LA1	5 959	16 758	163	247,5	116

Figura 5: LA1.

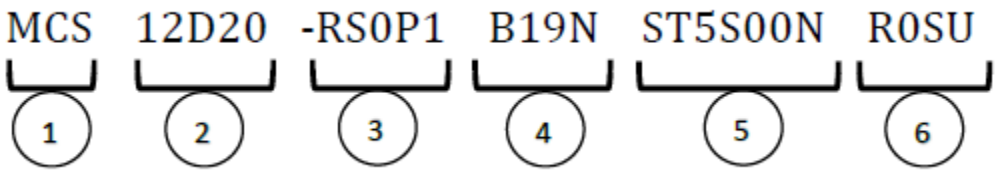
Il LEMC adottato nel progetto è della serie 2105 e di cui i principali dati sono riportati in Tabella 3.

Descrizione	Valore	U.M.
Tipo di vite	Vite a rulli	/
Diametro vite	21	mm
Passo vite	5	mm
Massima forza assiale	40	kN
Massima coppia in ingresso	41.7	Nm
Massima velocità lineare	500	mm/s
Massima velocità di rotazione	6000	rpm
Massima accelerazione	6	m/s^2
Corsa attuatore	100	mm
Inerzia vite	$1.45x10^{-4}$	Kgm^2

Tabella 3: Parametri generali LEMC 2105.

2.1.1.1 Servomotore

Il servomotore utilizzato è un sincrono AC della *Lenze* con sistema di retroazione affidato ad un resolver *Lenze*.
L'identificativo di targa è presentato in Tabella 4.



N.°	Descrizione
1	Tipologia motore (MCS=servomotore sincrono).
2	Taglia motore (12D) e velocità di targa (2000 rpm=20).
3	Voltaggio principale (-=400V), trasduzione (RSO=resolver) e freno (P1=24V DC).
4	Design dell'albero (B=albero con chiavetta , 19=diametro), livello vibrazione (N=normale).
5	Design plug cavo resolver e motore (ST= plug separati), grado di protezione 5, ventilazione (S00=senza ventola), volano (N=senza volano).
6	Sensori (R=sensori di temperatura KTY), targa (0=targa non elettronica), colore (S=nero), specifiche (U=UL/CSA).

Tabella 4: Identificativo della targa del motore elettrico.

I dati di targa del motore sono presentati nella Tabella 5: Dati di targa motore elettrico.Tabella 5 e la caratteristica di coppia in Figura 6.

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 $\eta_{100\%}$	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 5: Dati di targa motore elettrico.

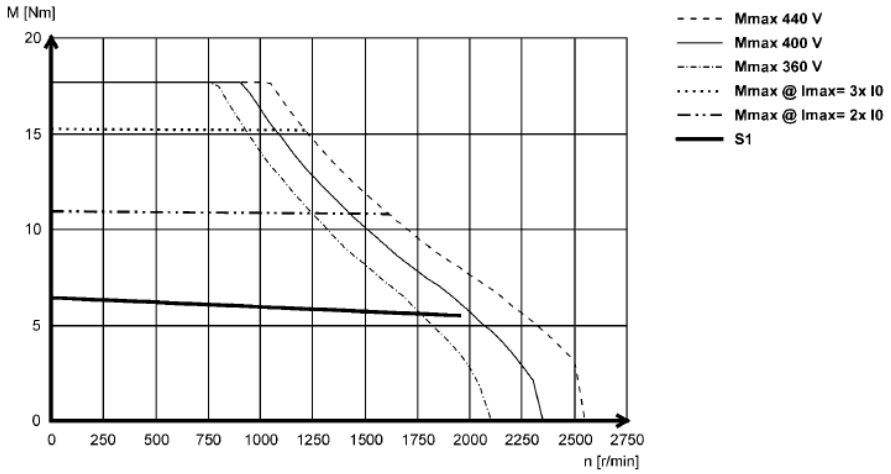


Figura 6: Caratteristica di coppia motore elettrico

Il resolver scelto (identificativo RV03) presenta le caratteristiche in Tabella 6.

Dato	**Valore**	**U.M.**
Risoluzione	0.8	‘
Accuratezza	±10	‘
Posizionamento assoluto	1	giro
Massimo voltaggio input	10	V
Massima frequenza input	4	kHz
Numero paia poli	1	-

Tabella 6: Dati di targa resolver.

2.1.1.1.1 Principio di funzionamento

Il funzionamento dei motori elettrici si basa generalmente sull’attrazione tra campi magnetici del rotore e dello statore. Per la legge della circuitazione una corrente che fluisce in un solenoide genera un campo magnetico, il quale interagendo con un altro campo genera una coppia secondo la relazione (eq.1):

$$C_M \propto B_{rot} B_{stat} \sin(\theta) \quad , \quad \theta: angolo\ di\ carico \tag{1}$$

L’angolo di carico corrisponde all’angolo compreso tra i due campi magnetici e si nota come la coppia sia massima quando i due campi sono disposti a 90°.
Il motore sincrono è un motore AC brushless trifase a magneti permanenti. I motori brushless hanno la caratteristica di essere privi di spazzole e dunque la commutazione (ovvero la variazione delle correnti nelle fasi del motore) è svolta elettronicamente.
Il motore sincrono viene accostato a quello DC brushless per via delle loro similarità, infatti le principali differenze consistono nella forma della forza contro elettro-motrice e nei *ripple* di coppia: nei DC brushless la *back EMF* è di forma trapezoidale mentre negli AC è di forma sinusoidale, a causa di una diversa distribuzione degli avvolgimenti sullo statore, da ciò deriva anche una commutazione di corrente più fluida nei motori sincroni, che quindi risultano meno rumorosi; a causa della commutazione continua si ottengono anche minori *ripple* di coppia.

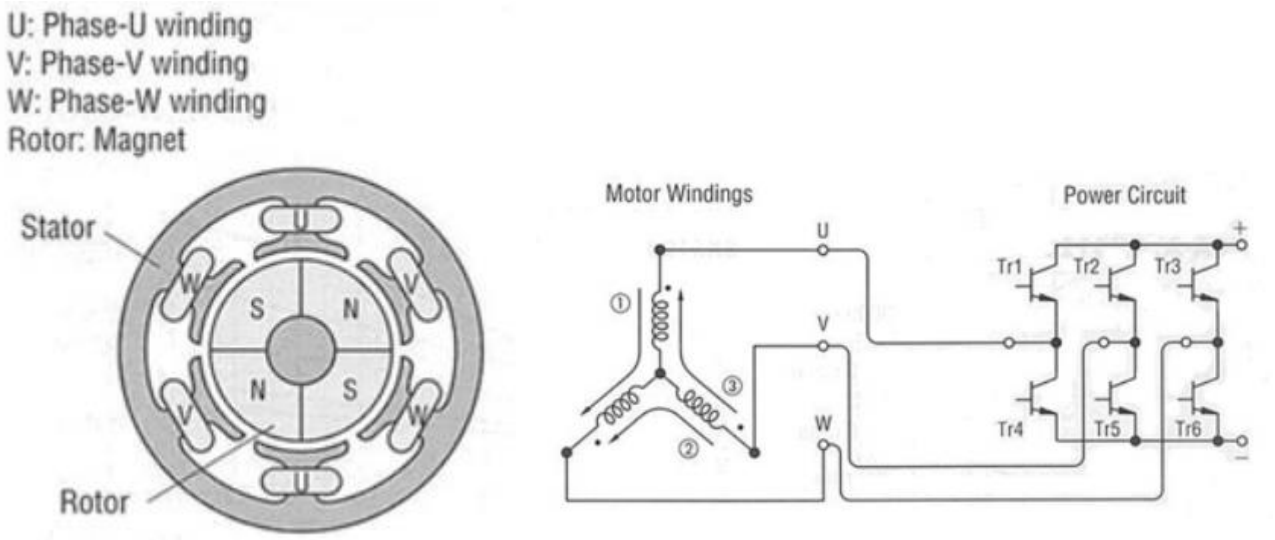


Figura 7: Schema in sezione e circuito di un motore trifase.

Come rappresentato in Figura 7 il circuito elettrico del motore è collegato al circuito di potenza presente nel driver, il quale secondo la logica del controllo determina la modulante sinusoidale per ciascun transistor (Tr) creata tramite la modulazione PWM, la quale determina tre correnti sfasate di 120° per le altrettante fasi, collegate attraverso una disposizione a stella degli avvolgimenti statorici del motore, il tutto rappresentato dall'(eq.2).

$$\begin{cases} i_U = i\sin(\alpha + 90) \\ i_V = i\sin(\alpha + 90 + 120) \\ i_W = i\sin(\alpha + 90 + 240) \end{cases} \quad ,\alpha: posizione\ angolare\ rotore \tag{2}$$

Le variabili che stabiliscono le correnti sulle tre fasi sono la posizione angolare α del rotore e la corrente i ottenuta dal controllo del *loop* di corrente; in uscita dal driver ottengo le tre correnti sinusoidali che scorrono negli avvolgimenti statorici, le quali stabiliscono un campo magnetico rotante che permette la rotazione sincrona del rotore a magneti permanenti trascinato per l'appunto dal campo magnetico dello statore. La Figura 8 illustra il funzionamento sopra descritto tramite un esempio inerente ad una rotazione del motore da 0° a 60°, dove a sinistra sono presentati gli andamenti delle correnti mentre a destra è indicata la direzione e intensità del campo magnetico per ciascuna componente.

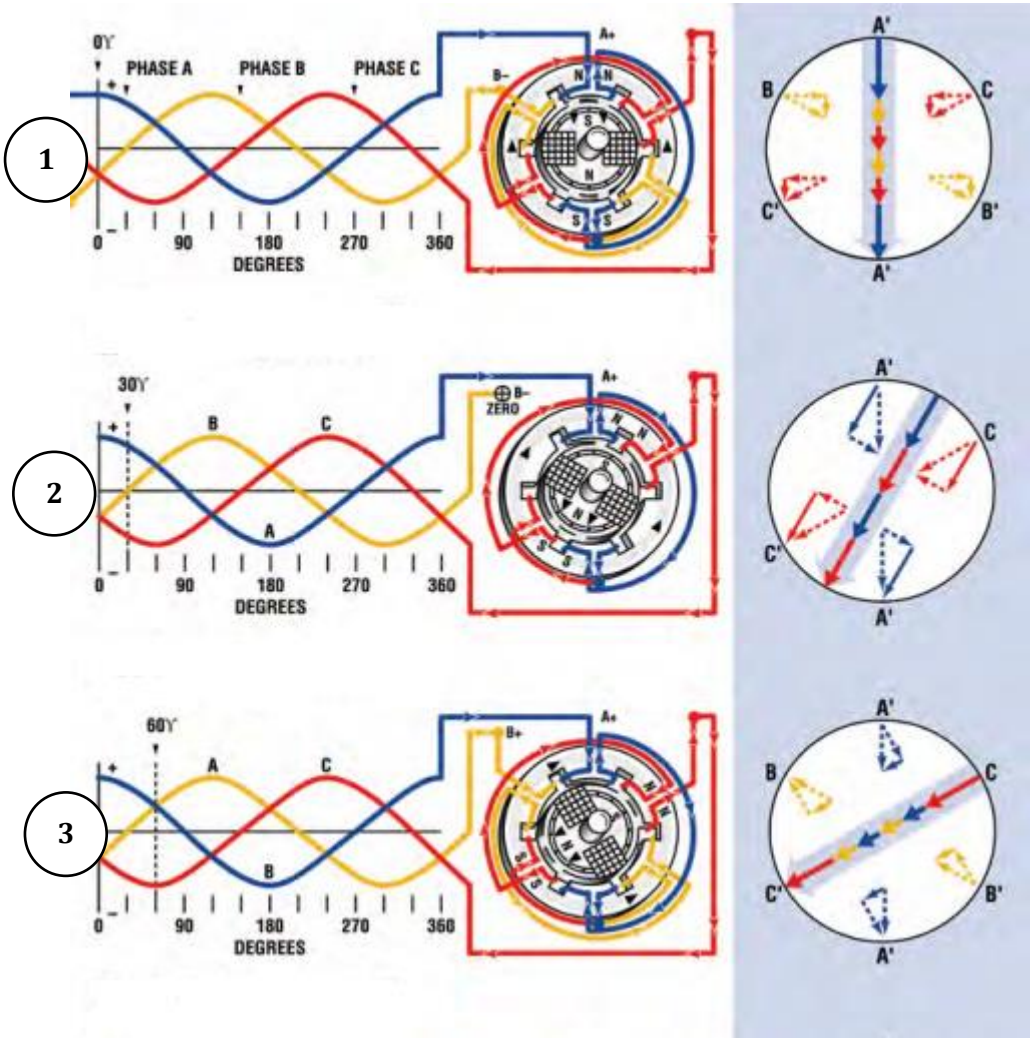


Figura 8: Funzionamento di un motore sincrono trifase a 2 poli a 0°(1), 30° (2) e 60°(3).

In realtà, da quanto si evince sopra, è la posizione del rotore che stabilisce la rotazione del campo magnetico rotante: grazie alla trasduzione del resolver, che manda in retroazione il segnale di posizione del rotore α, si modula la corrente in modo che il campo magnetico rotante ruoti ad una determinata frequenza che corrisponde alla velocità desiderata per il motore. Il controllo del driver verrà discusso in seguito in maniera approfondita.

Per quanto riguarda il resolver, questo è formato da due avvolgimenti sullo statore, denominati *avvolgimento seno* (sine) ed *avvolgimento coseno* (cosine), ed un avvolgimento solidale al rotore denominato *avvolgimento di riferimento* (*reference*). L'ampiezza relativa del segnale sugli avvolgimenti di seno e coseno (fissi e ortogonali tra di loro) determina la posizione angolare del rotore relativa allo statore.

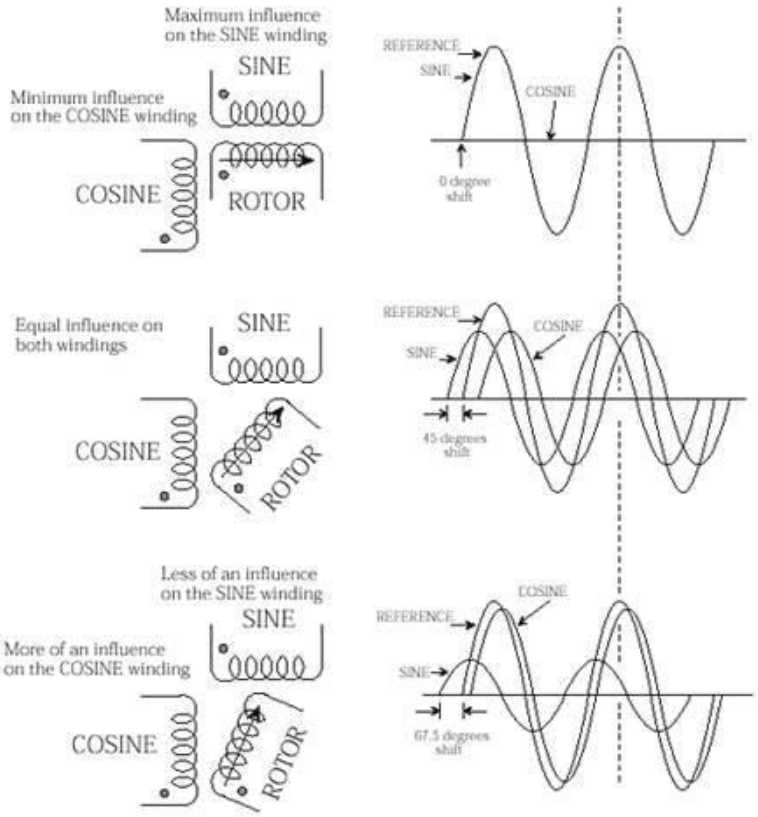


Figura 9: Principio di funzionamento del resolver.

L'avvolgimento di riferimento è eccitato con una tensione alternata, in modo da indurre anche negli avvolgimenti seno e coseno le correnti e quindi delle tensioni alternate come riportato in Figura 9; a seguito delle curve di inviluppo e del convertitore digitale a valle del resolver, è possibile misurare sia la posizione che la velocità del rotore.

2.1.1.2 Trasmissione a cinghia dentata

Il motore elettrico trascina una puleggia la quale trasferisce il moto mediante una trasmissione a cinghia dentata, adatta per applicazioni di precisione, il cui rapporto di trasmissione è unitario.

Figura 10: Trasmissione a cinghia dentata.

La seconda puleggia è accoppiata con la madrevite dell'attuatore lineare mediante una linguetta in modo da trasmettere il momento al sistema vite/madrevite.

2.1.1.3 Attuatore lineare a vite con rulli satelliti

L'attuatore a vite con rulli satelliti SKF è una trasmissione meccanica che trasforma il moto da lineare a rotatorio (o anche viceversa) proveniente dalla trasmissione a cinghia.

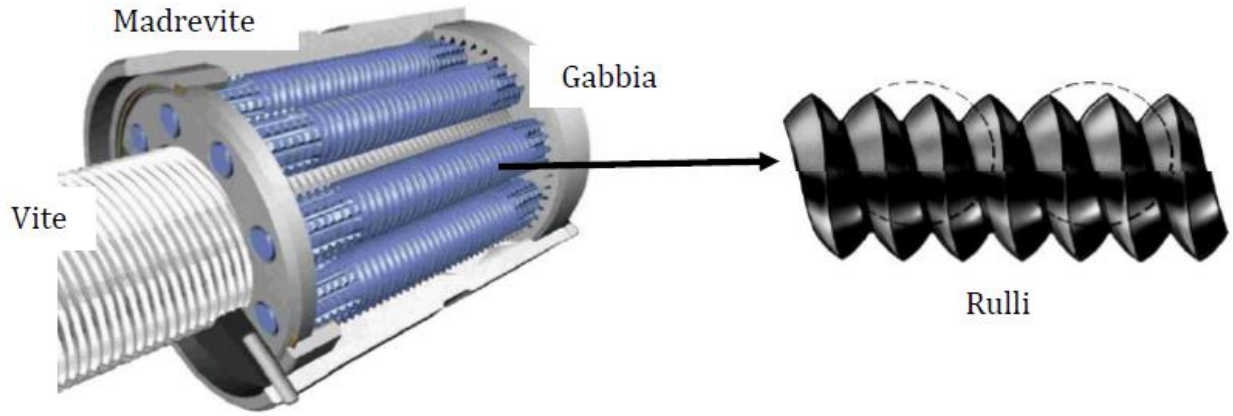


Figura 11: Attuatore lineare a vite con rulli satelliti.

La trasmissione è formata da 4 elementi visibili in Figura 11: la *madrevite*, filettata internamente e con filetto complementare a quello della vite, che è il primo organo di trasmissione della catena ed è accoppiata con il sistema di trasmissione a cinghia; i *rulli satelliti* che hanno un filetto arrotondato in modo da trasferire il contatto in modo simile a delle sfere; la *vite* che è l'ultimo elemento della trasmissione, la quale nella nostra configurazione rappresenta l'elemento di attuazione che trasla grazie al meccanismo a rulli satelliti; la *gabbia* che consente di mantenere la posizione relativa tra i rulli satelliti. Il principio di funzionamento risulta analogo a quello di un treno di ingranaggi epicicloidali.
Rispetto al meccanismo sfere, quello a rulli presenta notevoli vantaggi, tra cui quello di avere una maggiore superficie di contatto per la trasmissione del carico, il che permette una maggiore durata della vita del meccanismo, alti carichi ed alte velocità. In confronto al meccanismo a ricircolo di rulli, quello a satelliti presenta rulli con filetto ad elica e quindi un angolo di elica diverso da zero e coincidente con quello della madrevite, in modo da essere privo di relativo movimento assiale e consentendo un accoppiamento senza strisciamento. Quello a ricircolo di rulli invece dispongono di rulli privi di filetto ad elica ma con semplici scanalature anulari con angolo di elica uguale a zero, il che comporta un movimento relativo tra rulli e madrevite in direzione assiale: a causa di ciò risulta necessario un meccanismo di riposizionamento dei rulli simile a quello di ricircolo di sfere.
La soluzione a rulli satelliti risulta essere particolarmente vantaggiosa per le applicazioni come attuatore elettromeccanico lineare poiché è indicato per elevate velocità lineari, grazie al suo passo lungo, carichi elevati e condizioni ambientali critiche.
Al fondo dello stelo è stato progettato un dispositivo anti-rotazione che accoppiato al *rod end* garantisce la corretta trasmissione del carico dal meccanismo allo stelo e quindi permette il passo effettivo della vite ad ogni rotazione della madrevite, garantendo il posizionamento lineare corretto.

I dati di targa dell'attuatore a vite con rulli planetari sono riportati in Tabella 7 mentre i parametri dimensionali sono riportati in Tabella 8.

Descrizione	**Valore**	**U.M.**
Numero principi del filetto vite	4	-
Diametro vite	21	mm
Passo vite	5	mm
Angolo d'elica della vite	4.33	°
Numero di rulli	9	-
Carico dinamico	50.55	kN
Carico statico	81.97	kN
Gioco assiale massimo	0.02	mm
Massa madrevite	0.4	kg
Massa della vite/metro	2.7	kg/m
Inerzia madrevite	141.2	$Kgmm^2$
Inerzia rulli	6.5	$Kgmm^2$

Tabella 7: Dati di targa dell'attuatore a vite con rulli planetari SKF.

Appellativi	d_0	d_1	d_2	D g6/H7	A h12	w	a h9	b	H	Q	B	D_2	D_3
	mm	mm	mm	mm	mm	mm	mm	mm	mm	mm	mm	mm	mm
SRC 21x5	21	21,4	20,3	45	64	0,5	5	20	47,0	5	4	40,5	26

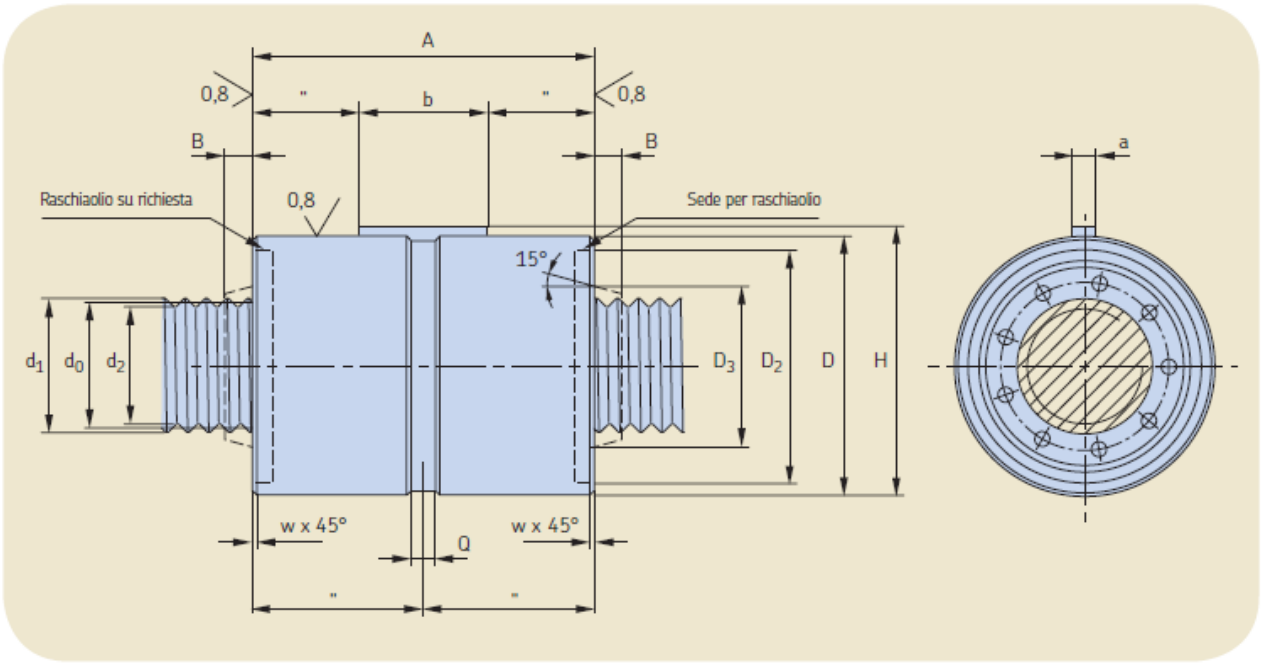


Tabella 8: Parametri dimensionali attuatore a vite con rulli planetari SKF.

2.2 Componenti di comando

In questa sezione si considerano i componenti montati all'esterno del banco prova servocomandi di volo, ovvero il driver, il c-Rio, l'elettronica di supporto e i cablaggi; si omette la UI del PC, la quale sarà trattata in seguito con il dovuto dettaglio.

2.2.1 Driver

Il driver utilizzato per l'azionamento ed il controllo è il *Servo-Drive Lenze 9400*, che consiste in un servo inverter che gestisce sia l'azionamento che il controllo del motore.
L'identificativo di riferimento è E94ASHE0044 e i principali dati elettrici di targa sono presentati in Tabella 9.

	Valore	**U.M.**
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 9: Dati di targa elettrici del driver Lenze 9400 E94ASHE0044.

Il servoinverter è costituito da due elementi da collegare dopo aver cablato la parte elettrica, mostrati in Figura 12: il *driver* e la *base di montaggio*. I cablaggi relativi alla rete elettrica sono collegati ai terminali presenti sulla base di montaggio, la quale funge da componente di potenza per il driver, il quale invece svolge il ruolo di controllore.

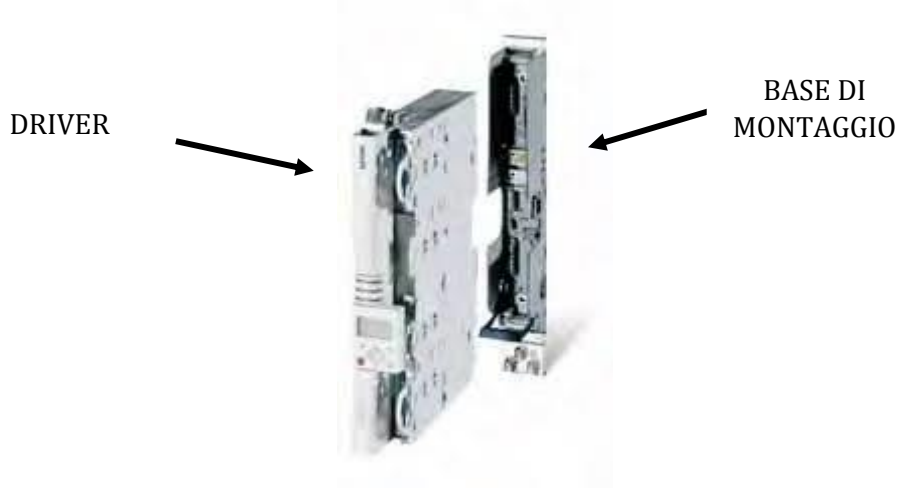


Figura 12: Driver e base di montaggio.

2.2.1.1 Principio di funzionamento

In Figura 13 è rappresentato lo schema elettrico di collegamento.

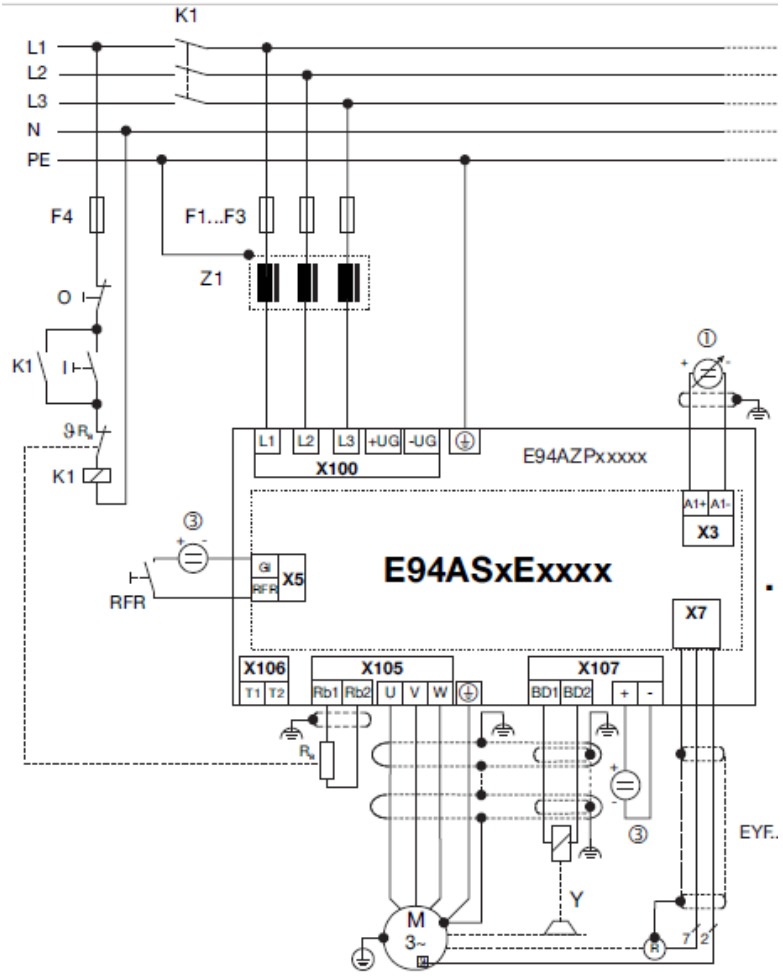


Figura 13: Schema elettrico di collegamento del driver.

Dallo schema si nota che viene prelevata la corrente alternata trifase dalla rete elettrica a 50 Hz tramite una presa tetra-polare industriale, corrente che è modulata grazie alla logica di controllo e dai transistor presenti nel driver, consentendo in questo modo di comandare il motore a velocità differenti.

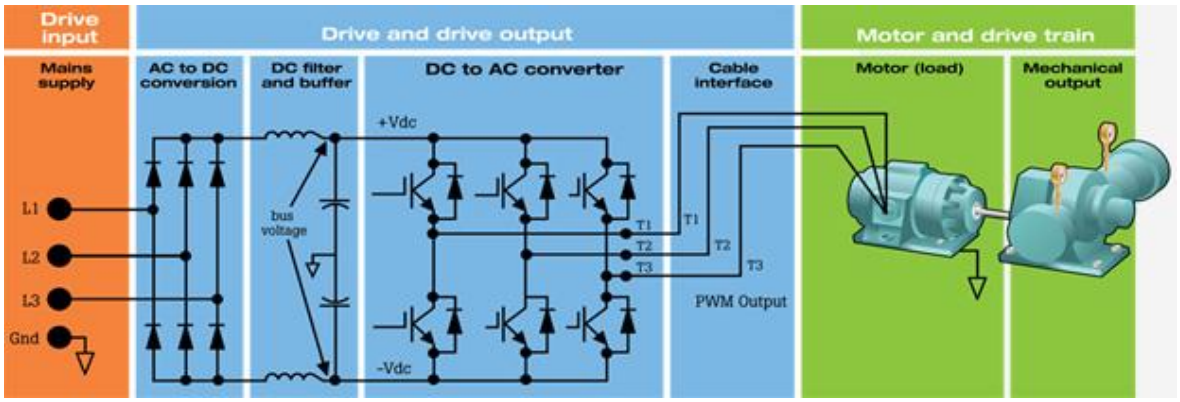


Figura 14: Schema elettronico dell'azionamento del motore elettrico mediante un Driver (inverter).

Come mostrato in Figura 14, prima di comandare il motore il driver deve svolgere una doppia conversione AC-DC/DC-AC. Innanzitutto il driver, rappresentato dai blocchi blu, riceve l'alimentazione dalla rete elettrica (blocco arancio) che è alternata AC e trifase; questa viene convertita in corrente continua DC mediante un raddrizzatore (primo blocco blu) il quale converte la corrente AC in DC e la manda al DC bus link (secondo blocco blu) dove tramite delle resistenze e capacità viene filtrata (evita i *ripples* che si sono creati) in modo da ottenere un voltaggio in corrente continua ai capi dell'inverter (terzo blocco blu), dove tramite un ponte ad H a 6 transistor, comandati dalla logica di controllo, si ottiene nuovamente una corrente alternata AC con frequenza e ampiezza desiderata sulle tre diverse fasi del motore. Quest'ultimo step è ottenuto grazie alla tecnica di modulazione PWM molto utilizzata nell'elettronica, la quale permette di poter comandare in corrente alternata il motore ad una frequenza diversa da quella della rete. Ciò è reso possibile grazie a dei tasti elettronici chiamati *transistor*, la cui frequenza di commutazione raggiungibile risulta dell'ordine di qualche kHz, valore impossibile da raggiungere per degli *switches* meccanici.

Infine il driver manda la corrente agli avvolgimenti delle tre fasi del motore (blocco verde) tramite il cavo motore e riceve il segnale di trasduzione del resolver dal cavo resolver.

È possibile inserire dei moduli esterni opzionali nel driver: sono stati inseriti un modulo di memoria, un modulo di sicurezza e un modulo *profibus*.

Il modulo di memoria MMI3 risulta necessario per allocare la memoria per le applicazioni da inserire una volta progettate e inviate nel Driver per un funzionamento *embedded*; è possibile impostare l'indirizzo e il *baud rate* attraverso dei DIP *switches* mostrati in Figura 15.

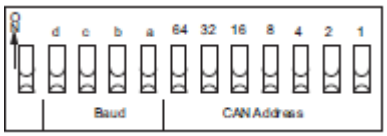


Figura 15: DIP switches del modulo di memoria.

L'indirizzo è calcolato come sommatoria degli *swtiches* in posizione ON, mentre il *baud rate*, ovvero la velocità bit/s di trasmissione dati, è scelto secondo la logica della Tabella 10.

d	c	b	a	Baud rate
OFF	ON	ON	OFF	10 kbps
OFF	ON	OFF	ON	20 kbps
OFF	OFF	ON	ON	50 kbps
OFF	OFF	ON	OFF	125 kbps
OFF	OFF	OFF	ON	250 kbps
OFF	OFF	OFF	OFF	500 kbps
ON	ON	ON	OFF	800 kbps
OFF	ON	OFF	OFF	1000 kbps
OFF	ON	ON	ON	Automated recognition

Tabella 10: Logica DIP switches.

Il modulo di sicurezza SMO è necessario per il funzionamento del driver ma non presenza alcuna funzione di sicurezza.
Il modulo profibus per la comunicazione non è utilizzato ma potrebbe essere utilizzato in futuro come alternativa via di comunicazione.

Inoltre il driver è dotato di una morsettiera con ingressi e uscite analogiche e digitali, attraverso le quali l'utente invia i segnali e quindi comanda il motore; saranno trattati successivamente nel capitolo inerente ai cablaggi.

Per quanto riguarda la frenatura, è presente una resistenza di frenatura esterna e il relativo comando di freno motore inseribile come modulo nella base di montaggio del driver.

2.2.2 c-Rio 9074

La serie c-RIO 907x della National Instruments racchiude dei sistemi integrati che combinano un controller RT e un processore FPGA (*Field Programmable Gate Array*) all'interno di un *chassis*, per applicazioni di controllo industriali e monitoraggio. In questo progetto si è utilizzato il c-RIO 9074, dotato di un chassis equipaggiabile con un massimo di 8 moduli e racchiude all'interno un processore RT da 400 MHz e un chip FPGA spartan-3 2M. Una memoria non volatile da 256 MB consente inoltre al c-RIO di lavorare come un sistema *embedded*: il codice scritto in fase di programmazione viene caricato direttamente sulla memoria consentendone il funzionamento autonomo. Presenta porte ethernet sul *chassis* consentendo la connessione con un PC *host*, che svolge il ruolo di monitoraggio, acquisizione dati e comando. Uno dei punti di forza del sistema c-RIO consiste nella sua elevata configurabilità, assicurata dal circuito FPGA definibile dall'utente e dalla vasta disponibilità di moduli della serie NI I/O che consentono di equipaggiare il c-RIO a seconda della necessità. Sono disponibili moduli per l'acquisizione e generazione di segnale analogico in tensione e in corrente, per l'acquisizione di segnali provenienti da termocoppie, accelerometri e celle di carico, moduli digitali del tipo 5V/TTL, contatori, temporizzatori, generatori di impulsi e relè per l'alta tensione e corrente.
Come accennato, la tecnologia FPGA concorre nell'implementare la riconfigurabilità del c-RIO. Un FPGA è a tutti gli effetti un circuito elettrico programmabile la cui struttura, schematizzata in Figura 16, presenta una suddivisione reticolare che individua un numero finito di blocchi detti celle logiche (nel caso specifico 46080 celle logiche). Tra un blocco e l'altro esistono delle interconnessioni riconfigurabili in base alle necessità (dipendono dai dati scambiati tra un blocco e l'altro). Esistono inoltre dei blocchi I/O che fungono da input per il circuito e da output rispettivamente per acquisizione dati da un modulo e generazione di un segnale.

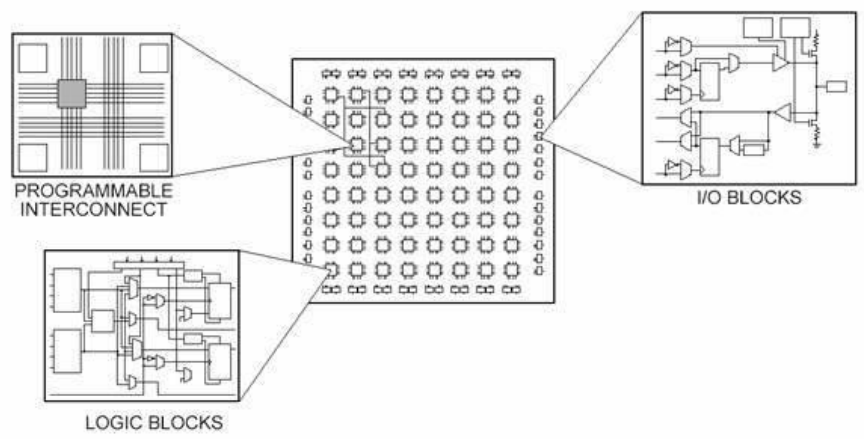


Figura 16: Struttura di un FPGA.

Ciascun blocco logico svolge una funzione (es. somma e sottrazione), assegnata in base al codice scritto dal programmatore: l'assegnazione di tale funzione e la scelta dei blocchi viene gestita autonomamente dal software LabVIEW in fase di compilazione, dunque il programmatore può sviluppare il codice con le tipiche modalità previste dall'ambiente LabVIEW, con l'aggiunta di nuove funzioni relative esclusivamente all'FPGA (modulo LabVIEW FPGA). Tuttavia, per come è definito, un FPGA non può ospitare un codice che richieda un numero di celle logiche superiore a quelle disponibili; di conseguenza, occorre prestare attenzione durante la sviluppo del codice scegliendo architetture che richiedano un minor uso di celle logiche rispetto ad altre.
Un altro notevole vantaggio dell'FPGA è la possibilità di lavorare a frequenze notevoli (fino all'ordine dei MHz). Il passaggio da un blocco logico all'altro viene scandito dalla frequenza di *clock*: maggiore è la frequenza di *clock*, minore sarà il tempo richiesto per l'esecuzione del codice, essendo quest'ultimo dato dall'insieme dei singoli blocchi logici interconnessi.
Inoltre, un FPGA, per via della sua architettura, lavora nativamente in RT. Per chiarire questo punto, occorre spiegare cosa significa un sistema Real-Time prendendo come esempio uno che non lo è: il PC. Un qualsiasi programma, è composto da porzioni di codice il cui funzionamento o scansione avviene ad intervalli regolari definiti dal programmatore. Tuttavia, il carico del processore dipende da diversi fattori, come ad esempio la presenza di programmi che lavorano in background, e tutto ciò non assicura che il tempo definito in fase di programmazione venga esattamente rispettato. Questo spiega la necessità di passare a sistemi RT: oltre alla necessità di elevate frequenze di scansione del codice (2.5 kHz), è importante il rispetto delle tempistiche, essendo il tempo un parametro fondamentale nell'acquisizione e generazione di un segnale. Oltre a ciò, esiste la possibilità di inserire nel programma parti di codice che lavorano in parallelo: nei sistemi RT, mediante opportuni accorgimenti e la scelta di specifiche funzioni, è possibile assicurare il vero parallelismo tra le due parti, ovvero che l'inizio della scansione di entrambi i codici avvenga nello stesso istante temporale. Questi prerequisiti, sono importanti anche in qualsiasi sistema di acquisizione dato che la ricostruzione della storia temporale di un segnale è legata alla frequenza d'acquisizione; si aggiunge che, più la frequenza di lavoro di un programma è alta, maggiore è il peso relativo dell'errore dovuto ad un timing non perfetto e dipendente dal carico del processore. Un FPGA, lavorando come un circuito elettrico, non è soggetto a carichi esterni. Inoltre, il parallelismo è assicurato dal fatto che i codici paralleli vengono fisicamente disposti in modo tale da risultare paralleli sul circuito individuati dai blocchi logici e dalle interconnessioni programmabili.
I vantaggi dell'FPGA possono essere ottenuti anche nell'ambiente LabVIEW Real-Time attraverso specifiche funzioni; nel nostro caso il determinismo è dato dall'utilizzo del FPGA. Come già notato, il c-RIO è composto da un processore RT da 400 MHz controllato e gestito da un SO nativo NI, chiamato per l'appunto Real-Time. Difatti, un sistema c-RIO, grazie al SO al suo interno, è in grado

di lavorare autonomamente (sistema *embedded*) una volta che sia stato caricato un programma scritto in LabVIEW Real-Time. Questo sistema operativo, assieme al processore, è ottimizzato per la gestione delle temporizzazioni del codice, ottenendo risultati, a basse frequenza, simili a quelle che si otterrebbero con l'FPGA. Tuttavia, per codici maggiormente complessi, le frequenze di lavoro non possono avvicinarsi a quelle del concorrente: infatti, con il c-RIO 9074, non è possibile superare i circa 500 Hz di frequenza di scansione, rendendo il Real-Time più vincolato a causa di questo fattore, per cui si è preferito utilizzare sia il processore RT, per ciò che riguarda il codice non deterministico, sia l'FPGA, per ciò che è necessario sia deterministico.

2.2.2.1 Modulo di Output analogico NI 9263

L'NI 9263 è un modulo di uscita in tensione analogico con aggiornamento simultaneo a quattro canali da 100 kS/s. Il range di uscita nominale è di ±10 V, permettendo di ricoprire l'intero range d'ingresso della morsettiera analogica del driver. Il segnale generato ha una risoluzione di 16 bit, sufficientemente elevata data il tipo di applicazione. Su ciascun canale è ammessa una corrente massima di 1 mA, condizione ampiamente. Inoltre, il modulo NI 9263 è dotato di una protezione da sovratensioni di ±30 V e una barriera a doppio isolamento del canale di terra per aumentare la sicurezza e l'isolamento dal rumore.
In Tabella 11 sono riportati i principali dati tecnici reperiti dal catalogo.

Dati tecnici		
Modulo		Output in tensione
Range segnale	Nominale	±10V
	Minimo	±10.3V
	Tipico	±10.7V
	Massimo	±11V
Corrente massima		±1mA/canale
Numero canali		4
Frequenza di campionamento		100 kS/s/canale
Aggiornamento simultaneo		Si
Risoluzione		16 bit
Isolamento		250 Vrms su canale di terra
Connettore		Morsetti

Tabella 11: Dati tecnici NI 9263.

2.2.2.2 Modulo I/O digitale 5V/TTL NI 9401

L' NI 9401 è un modulo di input/output digitale bidirezionale a 8 canali con un tempo d'aggiornamento minimo di 100 ns dato dalla frequenza di clock di 10 MHz. Ciascun canale può essere programmato in ambiente LabVIEW per funzionare come ingresso o come uscita, in base alle necessità. A prescindere dalla funzione, tutti i canali lavorano con un segnale in tensione a 0 o 5 V; lo *switch* tra uno stato e l'altro segue la logica TTL (Transistor-transistor logic).
Il modulo è stato utilizzato per l'abilitazione della funzione di posizionamento e per la scelta da parte dell'utente del secondo segnale di feedback. Deriva che, in caso di mancato segnale di consenso del modulo, gestito dal software, il comando di posizionamento al motore non si attiva e il motore rimane fermo.
In Tabella 12 vengono riportati i principali dati tecnici reperiti dal catalogo.

Dati tecnici	
Modulo	I/O digitale
Intervallo di input	0-5V
Intervallo di output	0-5V
Livello di logica	TTL
Numero di canali	8
Massima frequenza di clock	10 MHz (100ns)
Isolamento	250 Vrms su canale di terra
Connettore	25 pin D-sub

Tabella 12: Dati tecnici NI 9401.

2.2.2.3 Modulo di input analogico NI 9201

L'NI 9201 è un modulo di ingresso in tensione analogico a 8 canali single-ended ed una frequenza di campionamento aggregata di 500 kS/s. Questo modulo permette l'acquisizione dei segnali provenienti dal driver, accettando come input un segnale in tensione nel range ±10 V. L'acquisizione non è simultanea e ciascun canale può essere acquisito ad un massimo di 62.5 kS/s, valore più che sufficiente data la frequenza di lavoro del controllo (5 kHz).
Il canale di terra è isolato fino a 250 Vrms o 60 Vcc, dunque nessuna tensione dannosa all'interno delle specifiche di isolamento può danneggiare gli altri moduli del sistema, lo chassis oppure gli dispositivi connessi al PC. Il modulo è dotato inoltre di una protezione verso eventuali picchi di tensione fino a 2300 Vrms.
In Tabella 13 vengono riportati i principali dati tecnici reperiti dal catalogo.

Dati tecnici	
Modulo	Input in tensione
Range segnale	±10V
Numero di canali	8
Frequenza di campionamento	500 kS/s (62.5 kS/s/canale)
Aggiornamento simultaneo	No
Risoluzione	12 bit
Isolamento	250 Vrms su canale di terra
Connettore	Morsettiera

Tabella 13: Dati tecnici NI 9201.

2.2.3 Cablaggi

I cablaggi dell'intero sistema sono stati eseguiti secondo buona norma e seguendo quanto riportato nelle istruzioni di montaggio per ciascun componente del banco prova. Ogni cablaggio a seconda delle istruzioni riporta la dimensione corretta del conduttore da collegare e le schermature quando necessarie.
Schematicamente si rappresentano i cablaggi e i componenti del banco prova in Figura 17. Successivamente si analizzerà nel dettaglio ogni connessione.

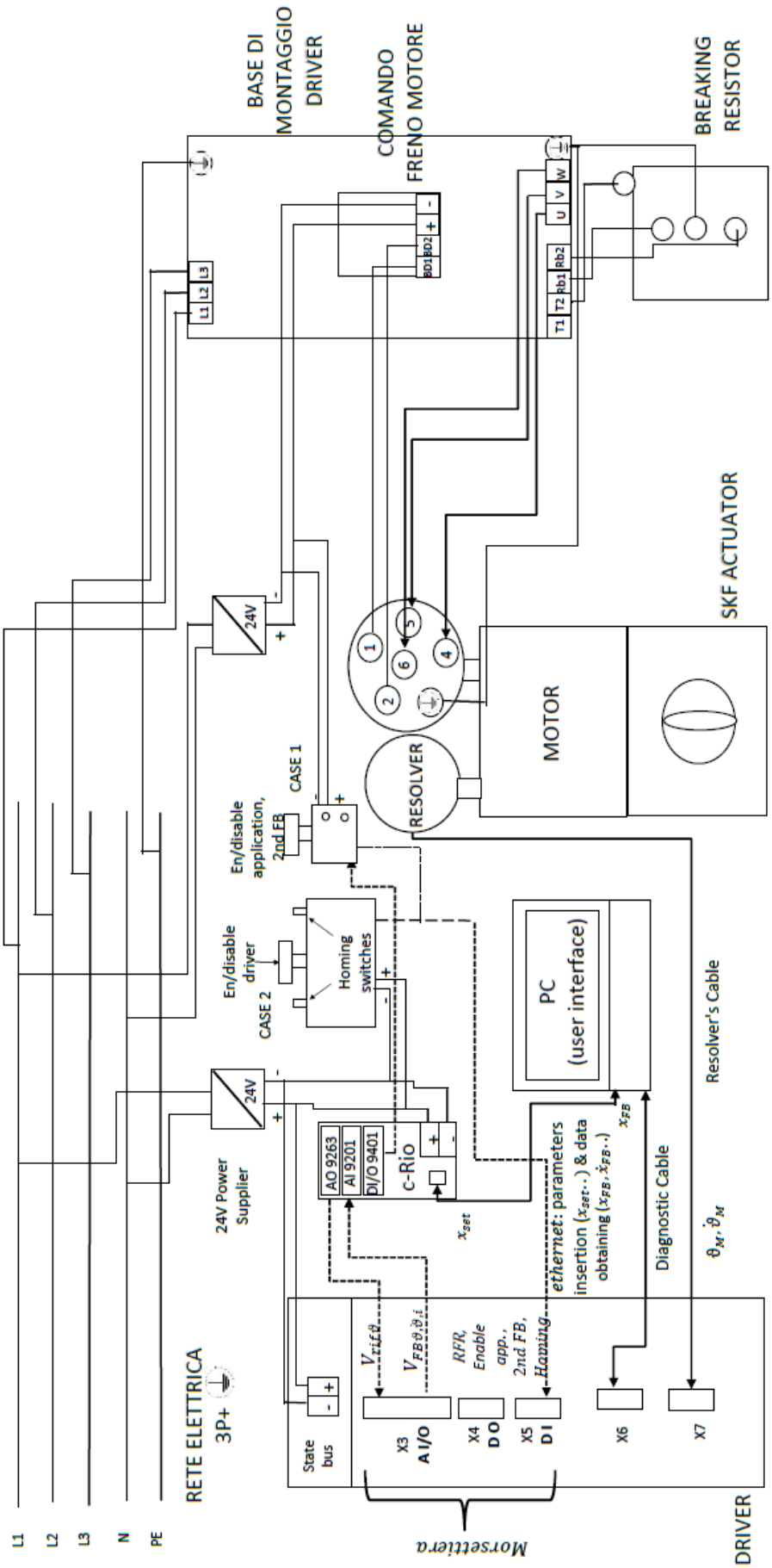


Figura 17: Schema di collegamento dei cablaggi.

La fonte di alimentazione del driver è la rete elettrica prelevata mediante una presa industriale, in Figura 17 rappresentata come pentapolare ma in realtà il driver riceve tramite i morsetti L1, L2, L3 le tre fasi e collega il conduttore di protezione PE alla terra sul *case* metallico del driver, risultando tetrapolare. Il neutro insieme ad una delle tre fasi simboleggia un collegamento di una spina a due poli. L'utenza trifase è quella che alimenta l'azionamento elettrico a transistor nel driver, come discusso in precedenza. Per un collegamento corretto si fa riferimento alla Figura 18.

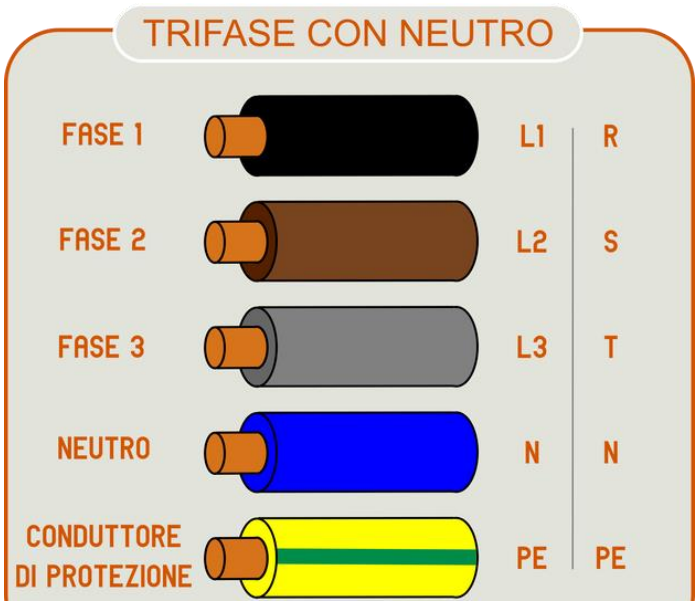
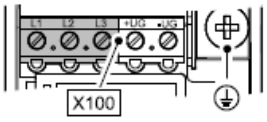


Figura 18: Nomenclatura e colori per un cablaggio industriale.

Si collega una presa tetrapolare alla rete elettrica e il relativo cablaggio con all'interno i conduttori delle fasi 1,2,3 e conduttore di protezione. Ad una adeguata distanza il cavo viene tagliato in modo da spelare i singoli fili e collegarli ai terminali L1, L2, L3 presenti sulla base di montaggio; il conduttore di protezione è collegato al terminale presente sulla base di montaggio mediante l'utilizzo di un occhiello cilindrico (Figura 19 a sinistra) e di una pinza apposita chiamata crimpatrice. Le dimensioni dei fili variano a seconda della corrente che circolerà in essi e sono scelti sapendo che il driver sul catalogo è denominato di taglia 2 come riporta la Tabella 14.

Terminal X100 (left part)	Labelling	Description
 SSP940X100	L1 L2 L3	Connection of the mains phases L1, L2, L3
	⏚	Connection for the supply-side PE conductor with M5 ring cable lug

	Max. conductor cross-section		Tightening torque	
	[mm²]	[AWG]	[Nm]	[lb-in]
Device sizes 1 + 2: flexible with wire end ferrule	2.5	12	0.5 ... 0.6	4.5 ... 5.3

Tabella 14: Collegamento fasi cavo motore e dimensione conduttore.

Figura 19: Capicorda a occhiello cilindrico (sinistra) e piatto (destra).

Il cavo motore collega driver e motore, manda le tre correnti alternate ottenute in uscita dal driver alle tre fasi UVW sul motore, il conduttore PE e i due cavi di comando freno motore BD1 e BD2, anch'essi spelati e collegati secondo quanto riportato nelle istruzioni in Tabella 15 e Tabella 16.

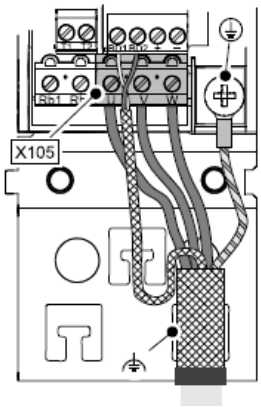
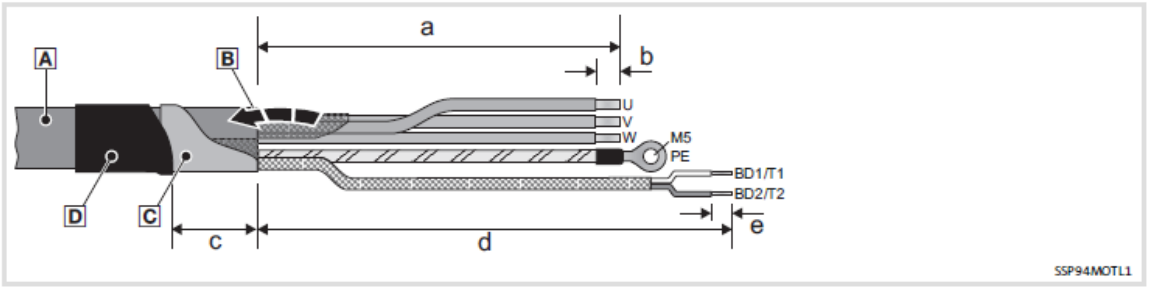
Terminal X105 (right part)	Labelling	Description
	U V W	Connection of the motor phases
	⏚	Functional earth Connect the shields of the motor phases and of the optional motor brake control separately and with a surface as large as possible to the shield sheet. Use EMC wire clamp or EMC shield clamp for fixing.
	⏚	Connection for the PE conductor on the motor side with M5 ring cable lug

Tabella 15: Collegamento cavo motore.

I cavi motore UVW in uscita e i cavi del comando freno motore BD1 e BD2 sono schermati mediante una tessitura metallica, inoltre i cavi sono pinzati mediante un meccanismo a molla, creando una terra funzionale. Le dimensioni dei conduttori sono presentate in Tabella 16.

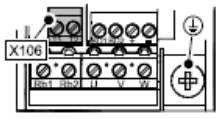
	Max. conductor cross-section		Tightening torque	
	[mm²]	[AWG]	[Nm]	[lb-in]
Device sizes 1 + 2: flexible with wire end ferrule	2.5	12	0.5 ... 0.6	4.5 ... 5.3



Type	Dimensions [mm] a	b	c	d	e
Device size 1	80	8	25	150	8
Device size 2	90	8	30	160	8

Tabella 16: Dimensione conduttori cavo motore.

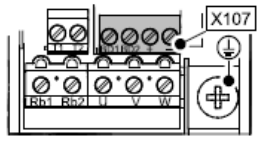
Sempre dalla base di montaggio sono collegati i cavi del sensore di temperatura della resistenza di frenatura T1 e T2 e la relativa messa a terra. Per il collegamento del PE sulla resistenza di frenatura si utilizza un capocorda piatto, mentre sulla base di montaggio un capocorda ad occhiello (Figura 19).

Terminal X106	Labelling	Description
 SSP940X106	T1 T2	Motor temperature monitoring with PTC element (type-A sensor, switching performance according to EN 60947-8 for type-A tripping units) or thermostat (NC contact).

	Max. conductor cross-section		Tightening torque	
	[mm²]	[AWG]	[Nm]	[lb-in]
Flexible With wire end ferrule	2.5	12	0.5 ... 0.6	4.5 ... 5.3

Tabella 17: Collegamento cavi sensore di temperatura motore e dimensione conduttori.

Il cablaggio del controllo freno motore consiste in due cavi provenienti dal cavo motore WH e BN, più i cavi di alimentazione provenienti da un alimentatore 24V, questo perché il comando freno motore deve possedere un alimentazione esterna e separata da quella del driver. Questi terminali sono disposti sul modulo freno motore, il quale è un modulo aggiuntivo che deve essere montato sulla base di montaggio.

Terminal X107	Labelling	Description
 SSP940X107	BD1 BD2	Connection of the motor holding brake + (Lenze: WH) - (Lenze: BN) E94AZHX0051: 24 V DC, max. 2.5 A Observe correct polarity!
	+ / -	Supply voltage for the motor holding brake (18 ... 30 V DC) Observe correct polarity!

	Max. conductor cross-section		Tightening torque	
	[mm²]	[AWG]	[Nm]	[lb-in]
Flexible With wire end ferrule	2.5	12	0.5 ... 0.6	4.5 ... 5.3

Tabella 18: Collegamento cavo freno motore e dimensione conduttori.

Infine, dopo aver inserito il comando freno motore sulla base di montaggio e aver cablato tutti i cavi elettrici, si può collegare la base di montaggio al driver come presentato in Figura 20.

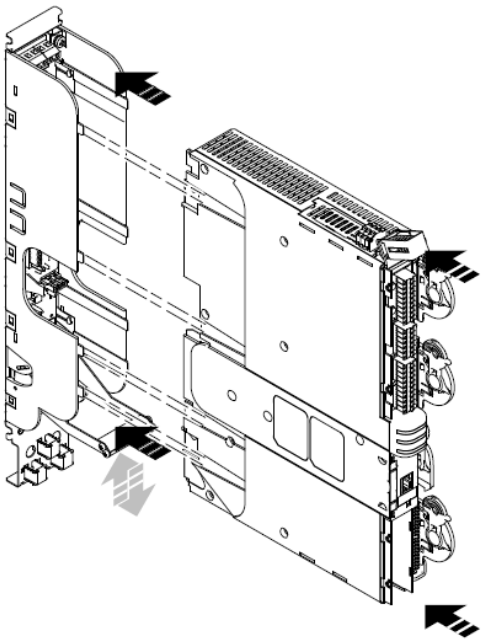


Figura 20: Collegamento driver-base di montaggio.

Sulla testa del motore sono presenti due *plug,* mostrati in Figura 3, il primo per l'ingresso del cavo motore e l'altro per il cavo resolver. Quest'ultimo è collegato sul Driver mediante il terminale X7 tramite un connettore D-sub, il quale schema di cablaggio è presentato in Figura 21.

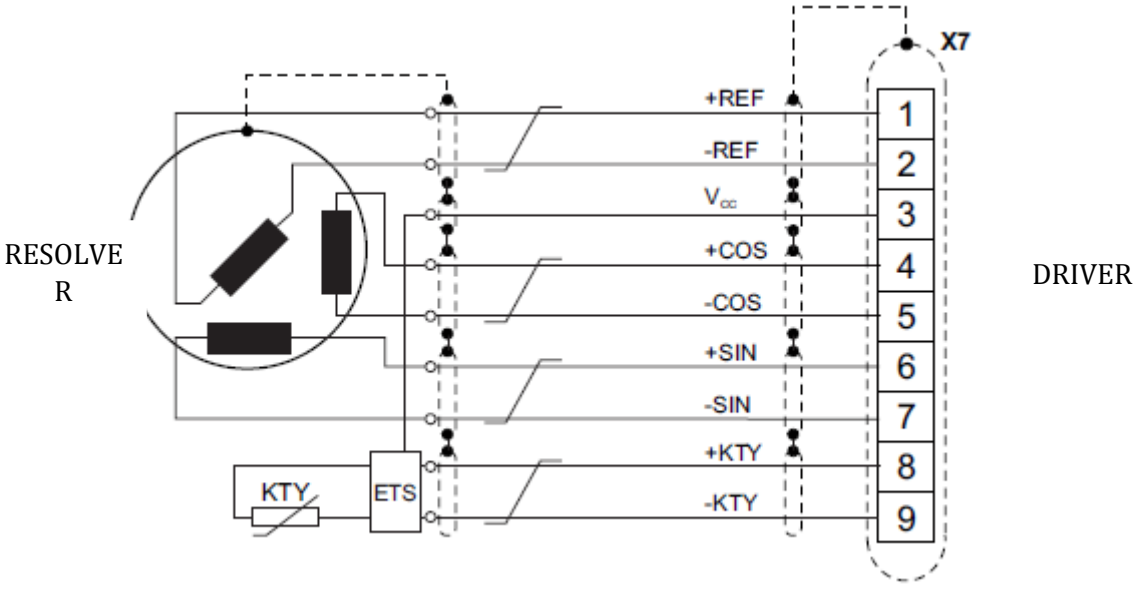


Figura 21: Schema del collegamento resolver-Driver.

Dati tecnici		
V_{cc}, −KTY(GND)	Alimentazione	5V
Range segnale	Massima corrente output	110 mA
+REF,-REF	Frequenza di input	Max 250 kHz
+COS,-COS	Voltaggio di eccitazione	10 V_{ss}
+SIN,-SIN	Frequenza portante	4 kHz fissa
+KTY,-KTY	Tipo	KTY 83-110
Connettore		9 pin D-sub

Tabella 19: Dati tecnici del collegamento resolver.

Il cavo resolver è formato da due cavi per l'avvolgimento di riferimento (+REF,-REF), due cavi per gli avvolgimenti seno e coseno (+COS,-COS, +SIN,-SIN), due cavi per il sensore di temperatura (+KTY,-KTY), un cavo per l'alimentazione di tale sensore (V_{cc}). I dati tecnici del cavo resolver sono presentati in Tabella 19.

Per monitorare lo stato dei parametri e, durante la fase di progettazione, per poter creare l'applicazione ed inserire i parametri necessari e caricare il tutto nel modulo di memoria del driver, si utilizza un cavo di diagnostica che collega PC e driver tramite due porte USB.

2.2.3.1 Relè e cablaggi

I segnali DI in ingresso alla morsettiera del Driver non possono essere elaborati solamente dal c-Rio poiché, come si evidenzierà successivamente nella sezione Terminali I/O, lo stato HIGH dei segnali digitali nel servoinverter è compreso nel range 15-30V e, dato il fatto che il modulo NI 9401 elabora segnali 0-5V, è necessario disporre di relè in modo da attivare gli stati digitali del Driver attraverso il consenso utente da software LabVIEW.
I relè utilizzati sono dei relè a stato solido (*State Solid Relay*-SSR): sono dei relè elettronici che si differenziano da quelli elettromeccanici per la loro compattezza e per la lunga durata di vita (non presenta il problema del degrado di parti meccaniche).
I relè sono quindi attivati attraverso un consenso a 5V in modo da ottenere in uscita un segnale a 24V diretto al terminale digitale del driver, ottenuto grazie all'ausilio di un alimentatore 24V.

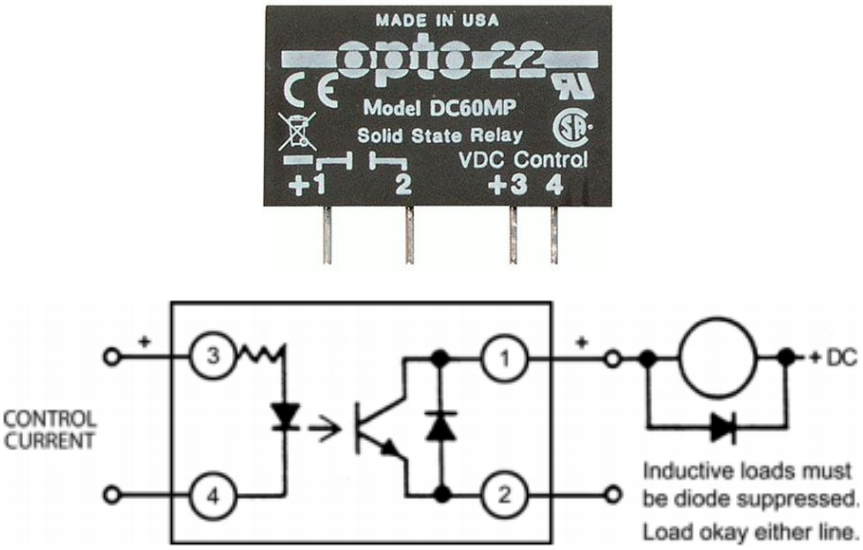


Figura 22: Relè DC60MP Opto 22 e schema elettrico.

In Figura 22 è mostrato il relè impiegato e il suo schema elettrico. Ai capi dei morsetti 3 e 4 sono collegati i cavi del consenso 5V proveniente dal modulo digitale NI 9401 (cavo segnale e comune), il polo positivo dell'alimentatore è collegato al pin 1 e il segnale digitale a 24V in uscita dal relè è prelevato dal pin 2. I due relè utilizzati per il banco prova sono racchiusi all'interno di un *case* (*case 2*) e permettono il consenso digitale mediante software LabVIEW rispettivamente dei segnali *Enable Application* e *2nd FB* visibili in Figura 17. Il primo abilita e disabilita l'applicazione del driver ed essendo un operazione a rischio, per permettere un intervento rapido di reset, si è disposto un pulsante a fungo a contatti normalmente chiusi a monte del relè montato sul *case* (Figura 23).

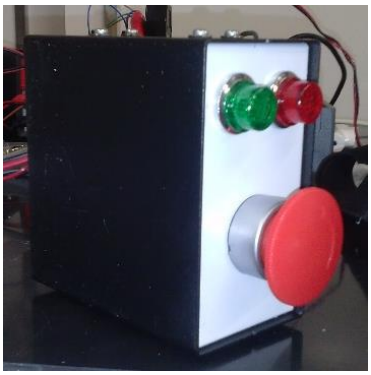


Figura 23: Case contenente i relè (case 2) e pulsante a fungo.

Negli schemi successivi si presentano le diverse configurazioni del circuito con le quali viene gestito il segnale digitale *Enable Application (Enable Follower)*. Vi sono tre possibili configurazioni:

- Nella prima il modulo del c-Rio dispone in uscita il segnale di consenso 5V, permettendo la chiusura del relè di interfaccia (R1); nel frattempo il pulsante di emergenza permane nello stato normalmente chiuso, per cui si ottiene una spia verde accesa e il segnale HIGH 24V in ingresso al terminale X5 del Driver in modo da attivare l'applicazione.

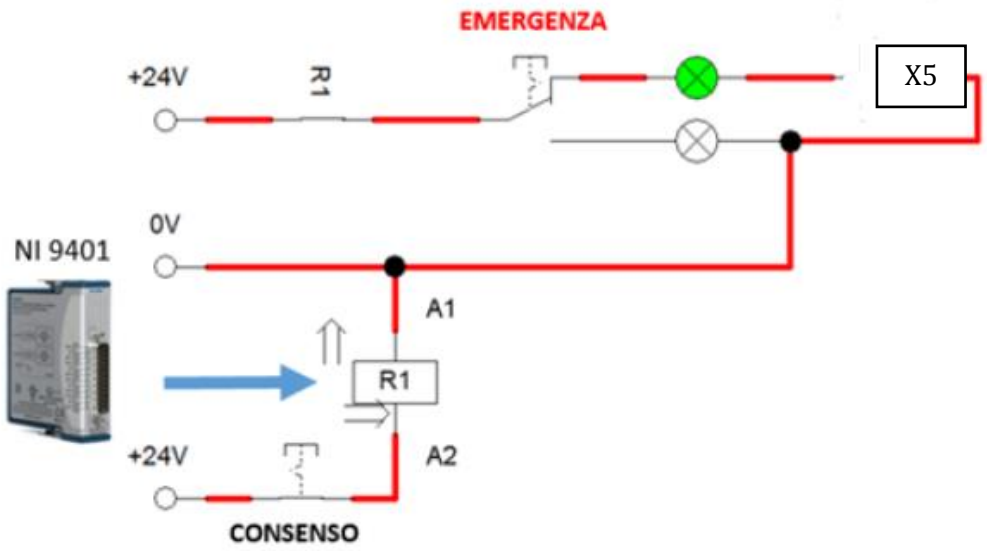


Figura 24: Schema configurazione per abilitazione applicazione: consenso software.

- Nella seconda non si dispone del consenso mediante il modulo digitale, per cui il relativo ingresso digitale nel driver vedrà lo stato LOW 0-5V e, non essendoci voltaggio ai capi di alcuna spia, non si illumina alcuna spia, per cui l'applicazione nel driver è nello stato disabilitata.

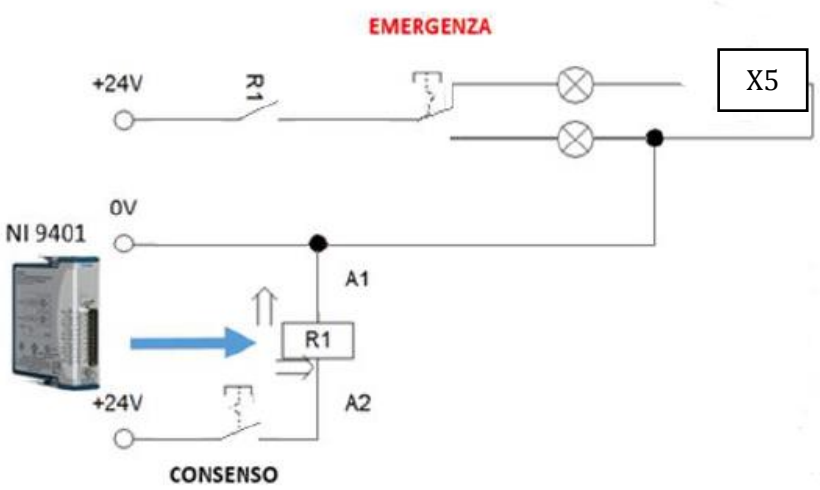


Figura 25: Schema configurazione per abilitazione applicazione: nessun consenso software.

- Nella terza configurazione il relè d'interfaccia è attivato mediante consenso 5V ed è premuto il pulsante di emergenza, per cui si accende la spia rossa e il terminale digitale in ingresso nel driver vedrà lo stato LOW 0-5V, per cui l'applicazione è disabilitata.

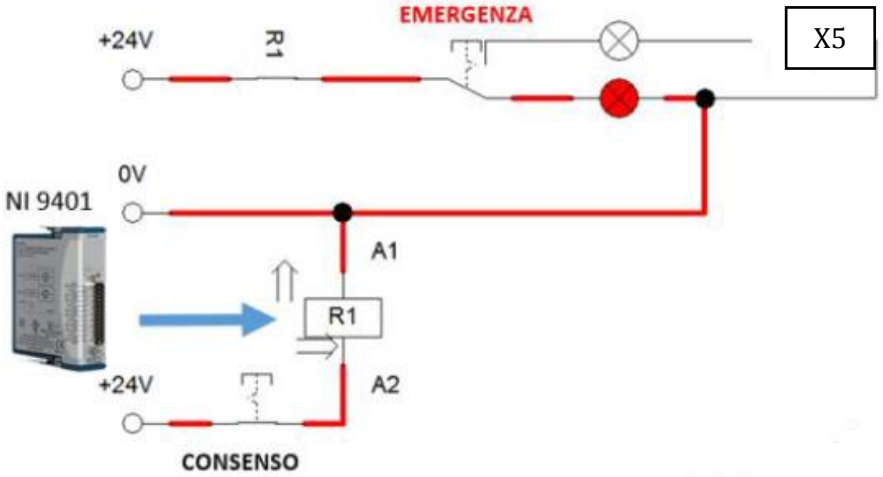


Figura 26. Schema configurazione per abilitazione applicazione: consenso software ed emergenza attiva.

Per quanto riguarda la gestione dei segnali digitali denominati *RFR*, attivazione e abilitazione *Homing,* questi sono gestiti in un secondo *case* (*case 1*). L'abilitazione RFR del Driver è ottenuta semplicemente mediante un pulsante a fungo normalmente chiuso collegato ad un alimentatore a 24 V, mentre le due funzioni di *Homing* sono gestite da due *switches* On/Off collegati anch'essi all'alimentazione a 24 V. Per l'attivazione dell'*Homing* è necessario che il Driver sia *offline* per questioni di logica di controllo nel driver, per cui la sequenza di attivazione consiste dapprima nel premere il pulsante a fungo disabilitando il segnale *RFR* e quindi il servoinverter, poi a quel punto si attivano gli *switches* attivando le funzioni *Homing*. I segnali utilizzati nel progetto e visibili in Figura 17, sono presentati in Tabella 24.

2.2.3.2 Terminali I/O e cablaggi

Il driver presenta quattro morsettiere X2, X3, X4, X5 disposte lungo il lato corto del driver, come presentato in Figura 27.

Il terminale X2 è utilizzato per lo *state bus* e permette di collegare più driver insieme nel caso di gestione *multi drives*; nel nostro caso non serve poiché si utilizza un unico driver. Il terminale X3 è quello relativo agli input e output analogici del Driver.

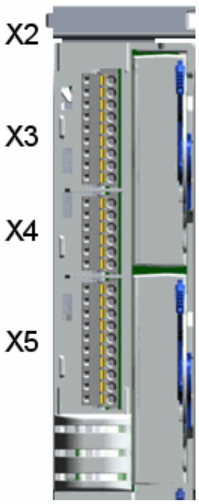
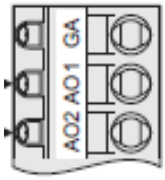


Figura 27: Morsettiere presenti sul driver.

Nelle Tabella 20 e Tabella 21 sono presentati i dati tecnici per la morsettiera X3, la quale comprende i terminali di input analogico e di output analogico.



Terminal	Use	Electrical data	
X3/AO1	Voltage output 1	Level:	-10 V ... +10 V (max. 2 mA)
		Resolution:	11 bits + sign
		Scaling:	$\pm 2^{30} = \pm 10$ V
		Conversion rate:	1 kHz
X3/AO2	Voltage output 2	Level:	-10 V ... +10 V (max. 2 mA)
		Resolution:	11 bits + sign
		Scaling:	$\pm 2^{30} = \pm 10$ V
		Conversion rate:	1 kHz
X3/GA	Reference potential (analog ground)		

Tabella 20: Morsettiera X3, dati tecnici output analogici.

Terminal	Use	Electrical data	
X3/A1- X3/A1+	Differential voltage input 1 (no jumper between A1R and A1-	Level:	-10 V ... +10 V
		Resolution:	11 bits + sign
		Scaling:	When C00034 = "0": ±10 V ≡ $\pm 2^{30}$
		Conversion rate:	1 kHz
	Current input (jumper between A1R and A1-)	Level:	-20 mA ... +20 mA
		Resolution:	10 bits + sign
		Scaling:	When C00034 = "1": -20 mA ... -4 mA = -2^{30} ... 0 +4 mA ... +20 mA = 0 ... 2^{30}
			When C00034 = "2": ±20 mA ≡ $\pm 2^{30}$
		Conversion rate:	1 kHz
X3/A2- X3/A2+	Differential voltage input 2	Level:	-10 V ... +10 V
		Resolution:	11 bits + sign
		Scaling:	±10 V ≡ $\pm 2^{30}$
		Conversion rate:	1 kHz

Tabella 21: Morsettiera X3, dati tecnici input analogici.

Da questi dati si ricava che, utilizzando un segnale in volt e non in corrente, il segnale analogico in ingresso e in uscita è compreso nel range ±10V, che lo *scaling* del segnale analogico in ingresso è di ±10V≡$\pm 2^{30}$ (vedi capitolo Software *Lenze Engineer*) e la frequenza di lavoro del Driver, ovvero quella del convertitore A/D-D/A, è di 1kHz. Nel capitolo successivo si discuteranno ulteriori dati di questa tabella.

Vi sono diversi metodi di cablaggio dei cavi segnale analogico in ingresso e in uscita dal driver a seconda dell'utilizzo desiderato; questi metodi sono schematizzati in Figura 28.

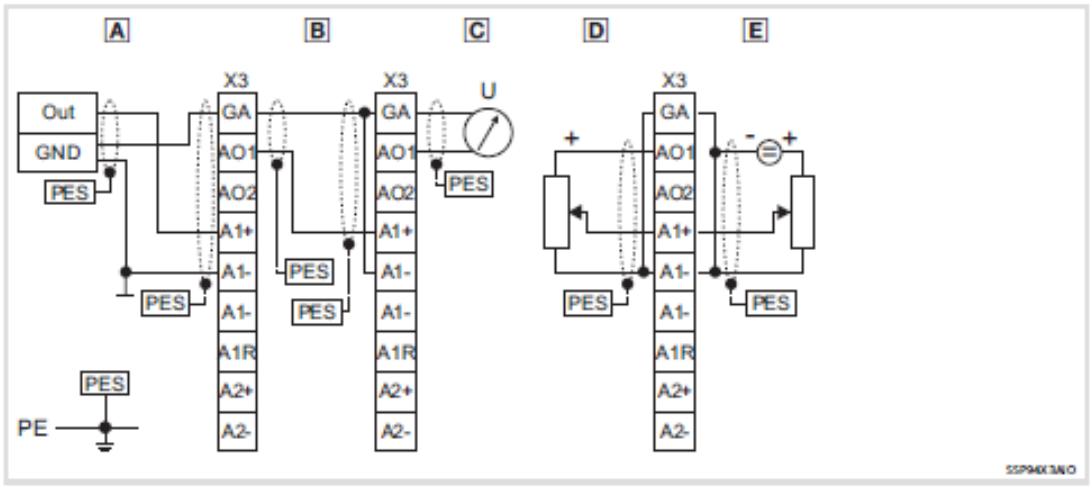


Fig. 3-30 Wiring principle

A	Wiring of an external analog signal
B	Wiring with a slave drive
C	Wiring with a measuring device
D	Potentiometer supplied by analog output 1
E	Potentiometer with external supply
Out	Analog output signal, e.g. of a control
GND	Earth reference potential
X3	Terminal for the analog inputs and outputs
PES	EMC shield connection
PE	Protective earth
U	Measuring device

Figura 28: Metodi di collegamento dei cavi segnali analogici

Il collegamento tipo A per i segnali analogici in ingresso e C per quelli analogici in uscita sono quelli scelti per cablare i cavi segnale del Driver: il segnale in arrivo dal modulo AO del c-Rio presenta tre fili, uno in ingresso al pin +, uno in ingresso al pin – e l'ultimo che collega il *ground* della morsettiera analogica AI a quello del modulo AO su c-Rio. In questo modo si ottiene un segnale differenziale in ingresso al driver. In uscita dal terminale AO i segnali sono *single-ended*, ne consegue un cavo per ogni segnale di feedback su cui varia il voltaggio e un secondo che collega i *ground* del terminale Driver e del modulo AI su c-Rio.

Si apre una parentesi sulla differenza tra segnale *single-ended* e differenziale.
Un input *single-ended* misura il voltaggio tra il canale HIGH e quello LOW, che in questo caso è il *ground* comune per tutti gli input; questo tipo di configurazione permette di avere più spazio per ulteriori canali ma dall'altro lato è suscettibile ai rumori legati a diversi fattori. Un segnale può subire un rumore di *background* dovuto alla EMF causata dalla corrente circolante in altri cavi o apparecchiature, oppure dai *ground loops* dovuti a più di un *ground* collegato tra due apparecchiature.

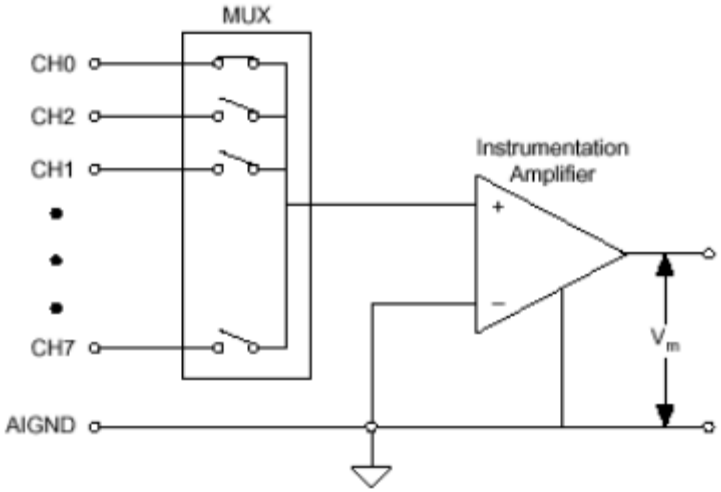


Figura 29: Schema di un input single-ended.

Prendendo in esempio la Figura 30 (EMF come causa del disturbo), nel cavo sono presenti il segnale desiderato (viola) e il segnale "rumore" della EMF (rosso) ad una frequenza diversa. Una volta entrato nell'amplificatore il rumore non può essere rimosso.

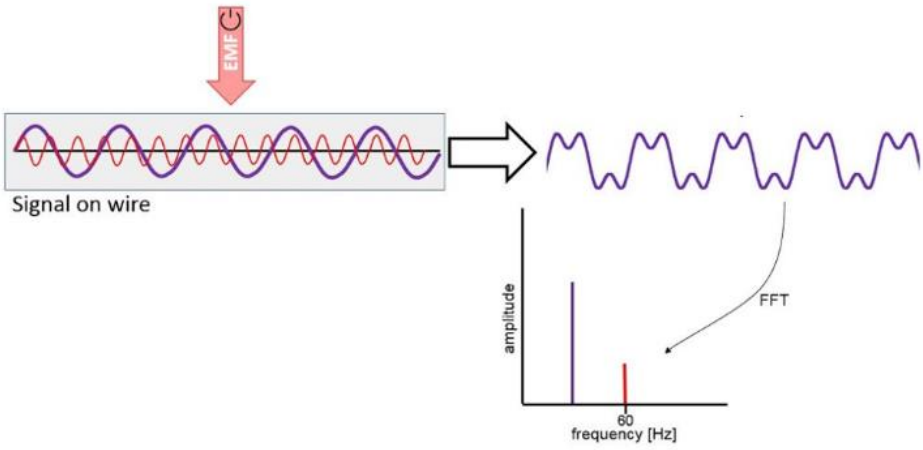


Figura 30: Disturbo EMF in collegamento single-ended

Un input differenziale misura il voltaggio tra due input individuali:

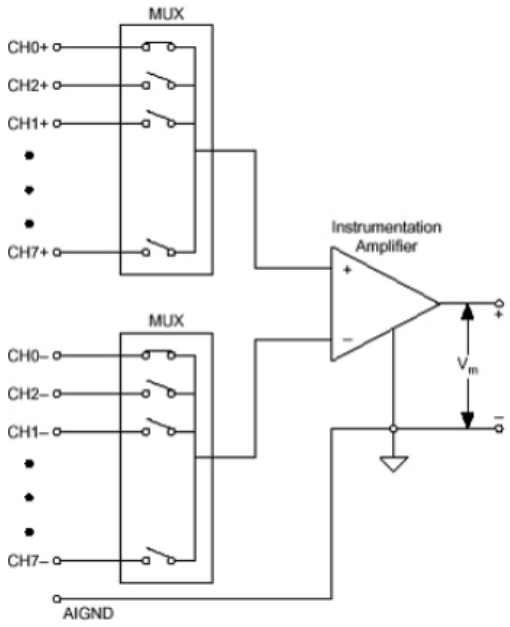


Figura 31: Schema di un input differenziale.

Il segnale (Figura 32) è trasmesso mediante due cavi in cui è stato introdotto il rumore dovuto alla EMF. Entrando nell'amplificatore operazionale il segnale LOW viene moltiplicato per -1 e viene sottratto al segnale HIGH: oltre a raddoppiare il segnale in uscita dal blocco, tende ad eliminare anche il rumore della EMF grazie alla sottrazione dei due segnali. Perciò risulta intuibile il vantaggio di un collegamento differenziale, il quale oltre ad essere indipendente dal *ground*, permette di ottenere un segnale non suscettibile al rumore.

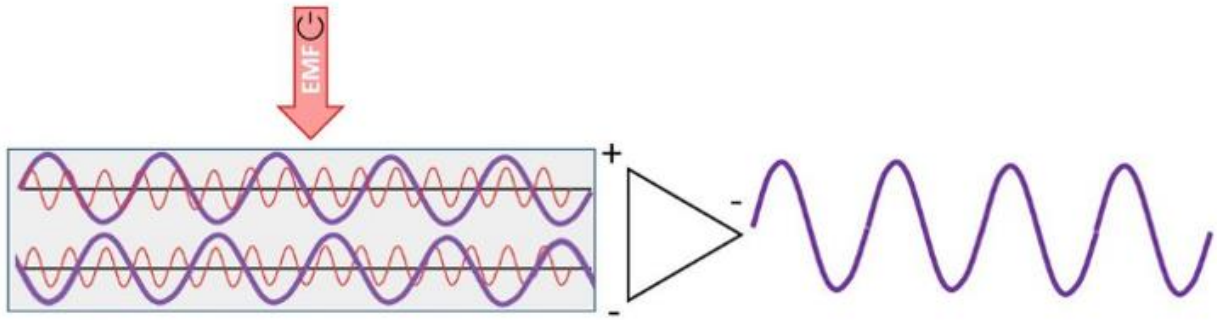
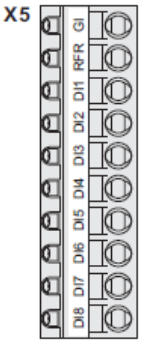


Figura 32: Disturbo EMF in collegamento differenziale.

I terminali della morsettiera X4 sono quelli predisposti ai segnali DO dal Driver ma che nella nostra applicazione non sono utilizzati.

I terminali della morsettiera X5 sono utilizzati per i segnali DI al driver. Le caratteristiche del segnale in ingresso a tale morsettiera sono presentate in Tabella 22.



Terminal	Use	Electrical data	
X5/DI1 X5/DI8	Digital input 1 ... 8	LOW level:	0 ... +5 V
		HIGH level:	+15 ... +30 V
		Input current:	8 mA per input (at 24 V)
		External-voltage protection:	Max. ±30 V
		Conversion rate:	1 kHz
X5/RFR	Controller enable	See digital inputs	
X5/GI	Reference potential (digital ground)		

Tabella 22: Morsettiera X5, dati tecnici input digitali

Si osserva come nel controllo del driver un segnale digitale in ingresso compreso tra 0 e 5V corrisponda allo stato LOW, mentre il driver assumerà lo stato HIGH quando il segnale è compreso tra 15V e 30V. Il campionamento è svolto nuovamente a 1kHz. Il pin RFR rappresenta il terminale di comando fondamentale che abilita e disabilita il Driver.

In

Terminal data				
	Conductor cross-section		Tightening torque	
	[mm²]	[AWG]	[Nm]	[lb-in]
Flexible With wire end ferrule	0.2 ... 2.5	24 ... 12	Spring terminal	

Tabella 23 si riportano le caratteristiche dei conduttori da utilizzare per i terminali I/O.

Terminal data				
	Conductor cross-section		Tightening torque	
	[mm²]	[AWG]	[Nm]	[lb-in]
Flexible With wire end ferrule	0.2 ... 2.5	24 ... 12	Spring terminal	

Tabella 23: Dimensione conduttori per i terminali I/O del driver.

I segnali analogici e digitali in uscita e in ingresso al Driver utilizzati nel progetto sono esposti in Tabella 24, con riferimento al *pin* e alla morsettiera.

Morsettiera	Pin	Segnale
X5 ±	RFR	Abilitazione Driver (RFR)
	DI2	Abilitazione applicazione driver (*Enable Application*)
	DI3	Abilitazione *Homing*
	DI4	Attivazione *Homing*
	DI6	Selezione *Feedback* Corrente/Velocità (*2nd FB*)
X3	AI1	*Setpoint* Posizione
	AO1	*Feedback* Posizione
	AO2	*Feedback* Corrente/Velocità

Tabella 24: Legenda dei segnali utilizzati per le rispettive morsettiere.

Per riassumere quanto detto in precedenza si riporta in Figura 33 i collegamenti tra i moduli del c-Rio, driver, i relè, i pulsanti, gli *switches* e gli alimentatori. Il c-Rio, in cui sono presenti i moduli di comunicazione analogici e digitali, è alimentato a 24V e comunica con il PC nelle due direzioni attraverso un cavo ethernet. Il modulo analogico AO 9263 invia segnali analogici alla morsettiera X3 scelti mediante l'interfaccia utente su PC. Il modulo analogico AI 9201 riceve segnali dalla morsettiera X3 che saranno visualizzati sulla interfaccia utente su PC. Il modulo digitale DI/O 9401 comunica con la morsettiera X5 e utilizza un segnale 0-5V, per cui (come approfondito in precedenza) è necessario utilizzare due relè di interfaccia presenti nel *case 2* in modo da inviare al driver un segnale compreso tra 0-24V; nel *case1* si notano i due *switches* utilizzati per attivare le funzioni di *Homing* e il pulsante a fungo di abilitazione *RFR.*

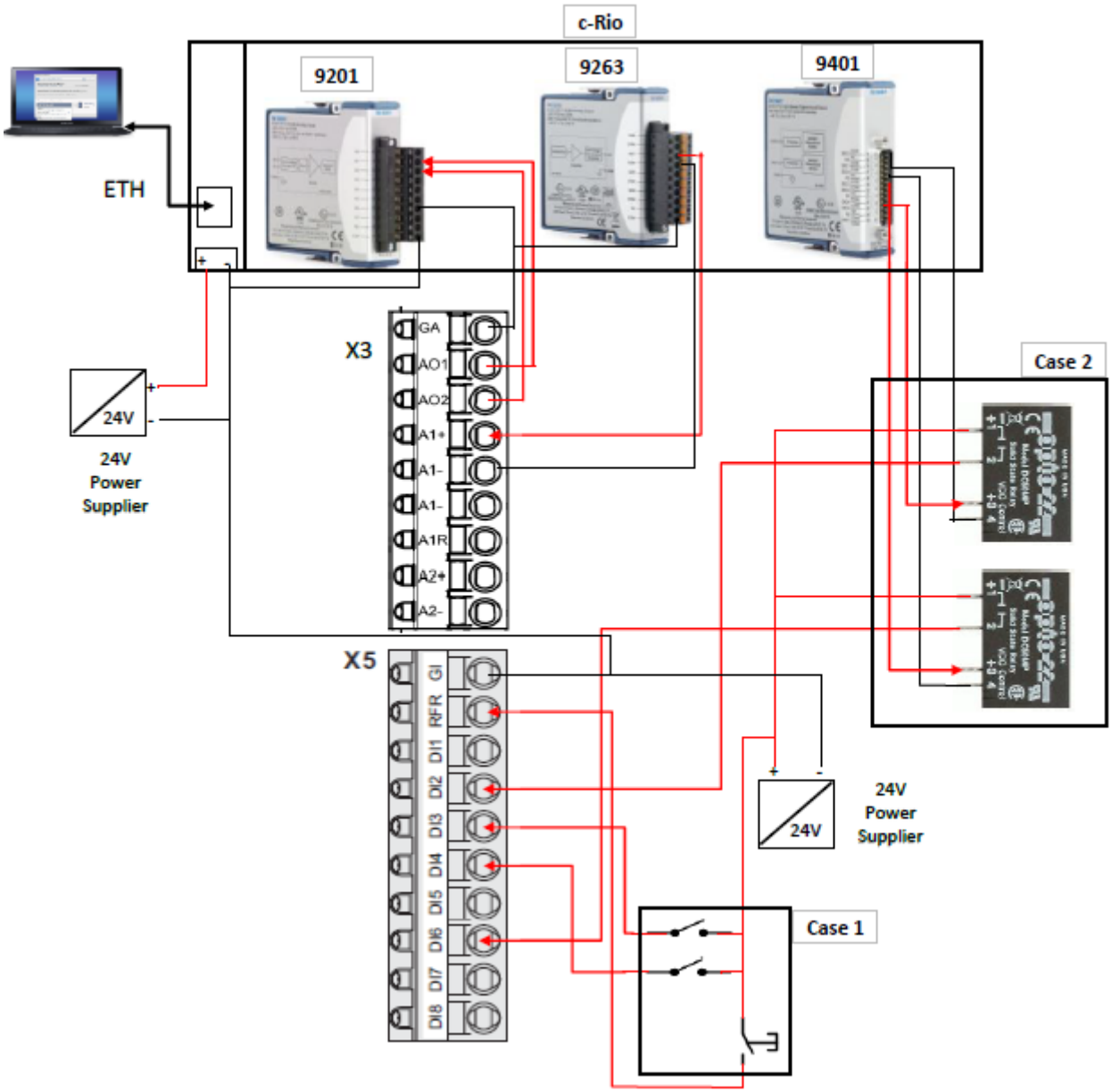


Figura 33: Schema di cablaggio tra c-Rio e morsettiera Driver.

Si nota come il *ground* dei moduli analogici e degli alimentatori siano collegati insieme in modo da evitare i *GL*: consistono in un non intenzionale *loop* di *feedback* causato dalla condivisione del *GND* elettrico comune. Nel nostro caso si possono formare *GL* nel collegamento tra *ground* dei moduli analogici del c-Rio, i quali presentano due *GND* differenti, i *ground* degli alimentatori e quelli della morsettiera del driver. I *GL* si formano solitamente dove vi è l'installazione di più apparecchiature elettroniche che scambiano segnali tra loro e connesse a diversi alimentatori; se c'è una differenza di voltaggio tra i diversi *GND*, allora una corrente fluirà attraverso i cavi portatori di segnale dal *GND* a potenziale maggiore a quello minore. Come approfondito in precedenza, i *GL* possono essere causa del 'rumore' in un segnale. Dunque per evitare ciò si sono collegati i *GND* degli alimentatori a quelli dei moduli analogici in modo da avere un unico potenziale di riferimento ed evitare il disturbo nei segnali.

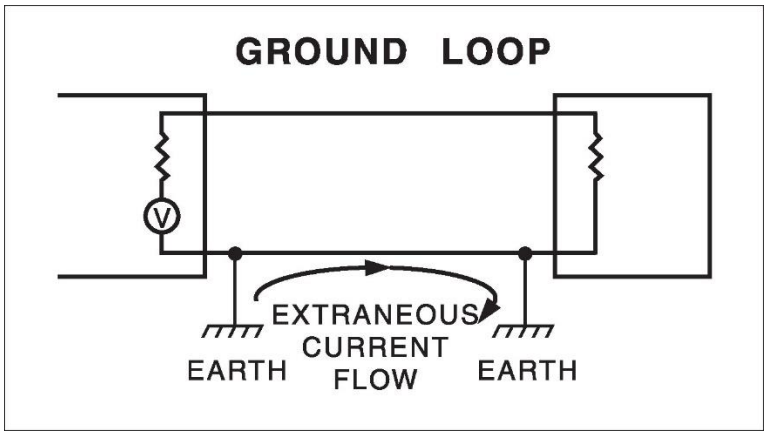


Figura 34: Esempio di schema di un ground loop.

3. Software *Lenze Engineer*

Questo capitolo mira a riassumere a grandi linee il funzionamento del software *Lenze Engineer* (la cui guida completa consta di alcune migliaia di pagine) ed è inteso come una guida di supporto per chi abbia intenzione ai futuri utilizzatori di tale software e dell'applicazione progettata in questa tesi. Risulta inoltre fondamentale, per poter comprendere appieno la progettazione dell'applicazione creata, accompagnare la lettura dei successivi capitoli alla navigazione dell'applicazione creata sul software.

3.1 Introduzione al Software *LE*

Come accennato nel capitolo precedente, il Driver svolge il ruolo di azionamento elettrico, per cui movimenta e regola l'EMA in base a controlli implementati in esso. Il Driver Lenze 9400 è accompagnato da un software installabile sul PC denominato *Lenze Engineer*, mediante il quale è possibile personalizzare il controllo del driver mediante l'inserimento di parametri tecnici relativi all'EMA, effettuare una programmazione a blocchi, monitorare il sistema e altre funzioni che verranno analizzate nella seguente trattazione. Tutto ciò rende la programmazione del Driver un punto fondamentale affinchè l'EMA possa essere azionato e quindi controllato in maniera opportuna.
Lo scopo della programmazione mediante il software è la creazione di una applicazione che, una volta ultimata, viene caricata sul modulo di memoria del Driver in modo da ottenere un sistema integrato che lavori autonomamente nel Driver. Per programmare il Driver mediante il software sul PC non è necessario l'utilizzo del cavo di diagnostica per connettere il PC al driver poiché il software funziona autonomamente sul PC; il cavo di diagnostica è utilizzato per rilevare automaticamente i componenti del driver nelle prime fasi di programmazione, per monitorare su interfaccia utente il funzionamento della applicazione durante i test sul motore elettrico, per caricare l'applicazione e i parametri sul Driver ogni qualvolta avvenga una modifica e per rendere 'online' il progetto sul Driver.
Nei successivi capitoli si analizzerà ogni sezione inerente al software *Lenze Engineer*, come l'interfaccia utente, le funzioni utilizzate del software, la creazione dell'applicazione e il metodo di controllo del motore implementato nel driver.

3.2 Interfaccia utente

L'interfaccia utente nella sua pagina iniziale del progetto svolto sul Driver è presentata in Figura 35.

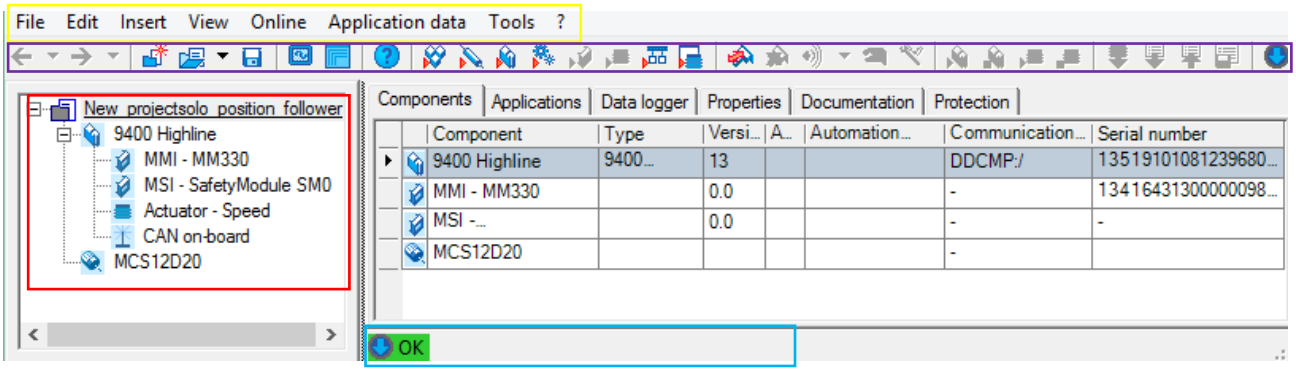


Figura 35: Interfaccia utente: Componenti.

Nella prima riga è disponibile la *toolbar* di menù (riquadro giallo), le cui funzioni principali sono quelle di salvare, aprire o creare un nuovo progetto, modificarlo, inserire nuovi componenti nell'albero di progetto e mandare online il progetto.
Appena sotto invece è riportata la *toolbar* dei comandi (riquadro viola), le cui funzioni principali sono l'inserimento rapido di nuovi moduli, assi, componenti, applicazioni e funzioni come il caricamento del programma da PC a driver, il salvataggio del programma e la sua costruzione (*Build)*.
A sinistra è posto l'albero di progetto (riquadro rosso), ovvero la presentazione gerarchica dell'architettura del progetto attraverso l'elenco dei componenti; cliccando sui singoli elementi si apre la relativa pagina in cui sono riportate informazioni e/o possibilità di interazione e modifica dei componenti selezionati.
A destra sono elencati il servo inverter, i moduli che lo compongono e il motore selezionato: l'inserimento di tali componenti è la prima operazione svolta quando si crea un nuovo progetto e consiste in una procedura guidata che mira ad inserire i componenti attraverso la loro nomenclatura a disposizione sulle relative schede tecniche e sulle targhe dei componenti.
In basso è riportato lo status attuale in cui si trova il Driver mediante una stringa (riquadro azzurro), quindi permette all'utente di visualizzare comodamente se il programma è stato costruito, se è online, se il controllore presente nel driver è stato abilitato o inibito.
Il software LE permette l'inserimento e la visualizzazione di parametri in apposite caselle. Tali caselle sono identificate attraverso diversi colori che consentono di associarle alla funzione a cui sono adibite (Tabella 25).

Colour	Example
Offline representation	
White	1,2345
Light grey	1,2345
Online representation	
Yellow	1,2345
Pale yellow	1,2345
Red	1,2345

Tabella 25: Rappresentazione dei parametri numerici.

Bianco e grigio rappresentano parametri attualmente *offline*, il bianco per i parametri a scrittura e lettura, il grigio per i parametri a sola lettura. Giallo e rosso parametri attualmente *online*, il giallo per parametri modificabili e che sono caricati all'interno del driver (in caso contrario, se i parametri inseriti nel driver sono diversi da quelli inseriti nel programma il valore è preceduto dal simbolo ≠), giallo spento per i parametri di sola lettura e in rosso nel caso di interruzione di comunicazione tra PC e Driver.
Il monitoraggio dei parametri può essere fatto andando a ricercare il parametro desiderato nell'apposita sezione, oppure i parametri di interesse possono essere selezionati e inseriti nella schermata principale (riquadro rosso Figura 36).

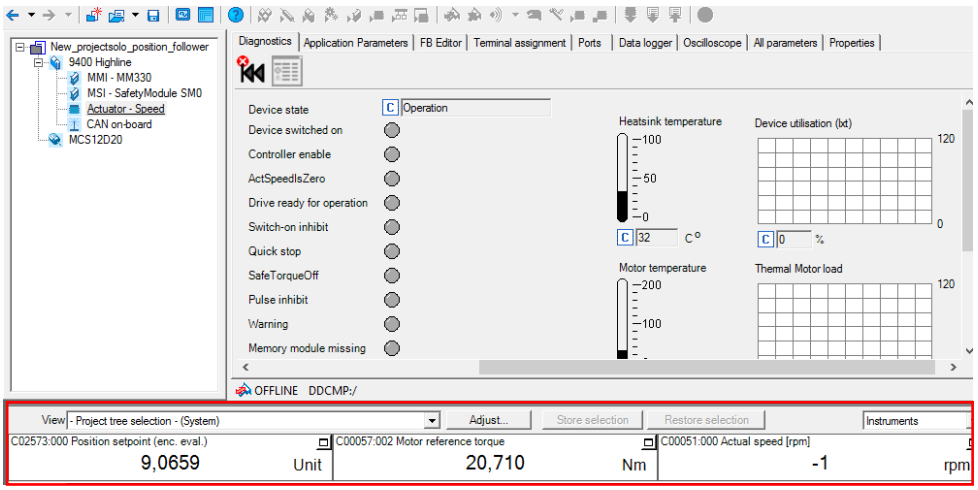


Figura 36: Monitoraggio rapido dei parametri selezionati su interfaccia utente.

3.2.1 Creazione di un progetto

La creazione di un nuovo progetto consiste nel selezionare i componenti facenti parte del sistema da un catalogo virtuale parte (metodo *offline*), mentre alcuni componenti possono essere inseriti attraverso la rilevazione automatica mediante cavo di diagnostica (metodo *online*). La pagina di inserimento risulta quella in Figura 37, dove in sequenza si inseriscono (seguendo il riquadro giallo) dapprima il controllore (Driver Lenze 9400) in possesso, successivamente i rispettivi moduli di sicurezza e di memoria, l'applicazione da inserire nel controllore, il motore sincrono, il resolver e ulteriori componenti opzionali. È inoltre consentito mantenere i valori presettati per i componenti inseriti oppure sostituirli. Tutti gli elementi sopra citati possono essere aggiunti o modificati anche in seguito. Il sistema di I/O nel caso in questione non è inserito poiché gestito dal c-Rio, il quale non è un sistema riconosciuto dal driver Lenze.

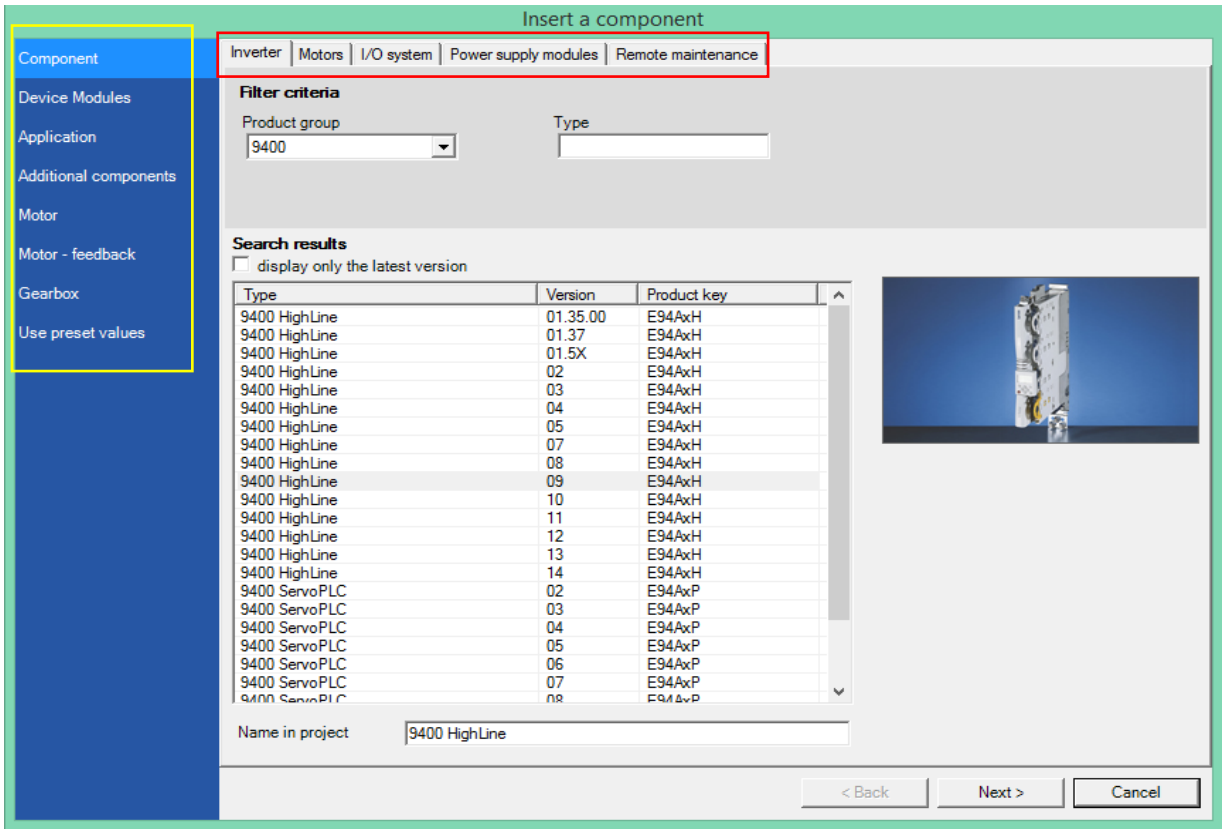


Figura 37: Inserimento dei componenti nel progetto.

A questo punto l'albero di progetto è creato in modo automatico (Figura 38).
In progetti di grosse dimensioni si può strutturare l'albero di progetto in modo da suddividerlo in diversi assi e per ognuno di essi gestire diversi motori e applicazioni, inoltre se si dispone di diversi servoinverter *Lenze* è possibile gestire più di un Driver attraverso la creazione di un unico progetto, configurazione presente in molti contesti indstriali dove è necessario il controllo di diversi motori elettrici su differenti assi. Nel caso in questione il progetto è semplice, per cui non è necessario suddividerlo in ulteriori moduli e assi, per cui è inserito solamente un controllore Lenze, al cui interno sono inseriti i moduli e l'applicazione, adibito al controllo di un unico il motore elettrico MCS12D20.

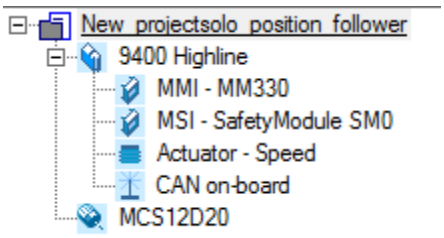


Figura 38: Albero di progetto.

Per stabilire una connessione tra PC e driver e poter svolgere le funzioni di monitoraggio, progettazione e inserimento parametri è necessario un sistema di comunicazione: i possibili metodi sono mediante cavo diagnostica, CAN, Ethernet, Profibus e Profinet. Nel caso in questione si è in possesso di due metodi di comunicazione, tramite cavo di diagnostica e Profibus: si è preferito scegliere il primo metodo di comunicazione e tale scelta viene impostata durante la creazione del progetto.

Una volta creato il progetto, mediante la *toolbar* si utilizza la funzione *Build* per consolidare il progetto, si attiva la comunicazione attraverso il comando *Go Online*. Al termine della programmazione, mediante *toolbar*, si può effettuare il download del programma dal PC al driver, il download soltanto dei parametri inseriti *offline* da PC a driver e l'upload dei parametri nella direzione opposta; è infine presente un pulsante di salvataggio di tali parametri.

3.2.2 Creazione di una applicazione

Durante la fase guidata iniziale di creazione del progetto, dopo aver caricato il driver nell'albero di progetto, viene associata un'applicazione predefinita che può essere una ed una sola per ogni controller presente in progetto; nel caso progettato si utilizza un unico Driver e quindi l'applicazione sarà una sola. Le applicazioni disponibili sono quelle presentate in Figura 39.

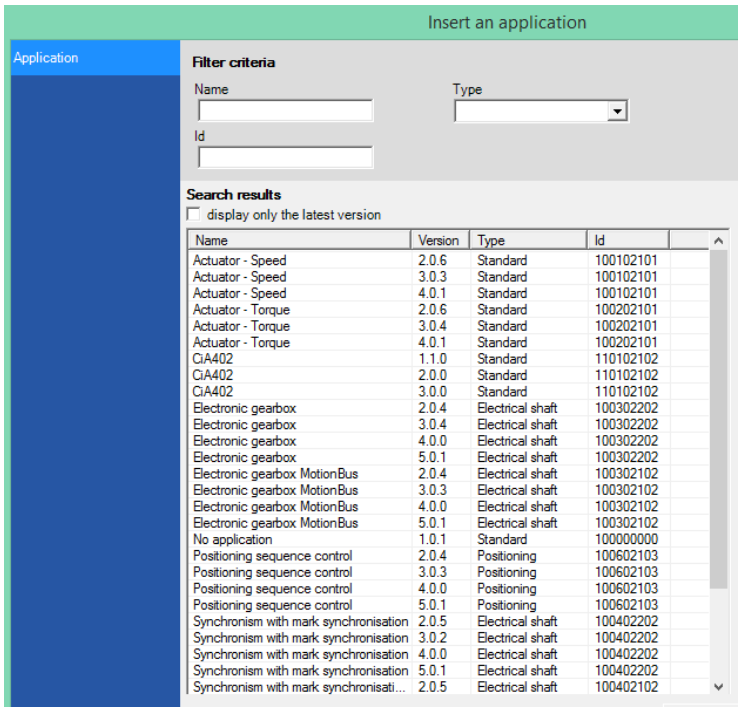


Figura 39: Inserimento dell'applicazione durante la creazione del progetto.

Per il progetto si è optata un'applicazione di attuatore-velocità (*Actuator-Speed*), poiché il controllo più vicino alle nostre necessità svolge un controllo velocità sul motore. Così facendo nella finestra del *FB Editor* appare un diagramma a blocchi predefinito apparentemente pronto per l'uso. In alternativa è possibile partire da un'applicazione vuota e iniziare la programmazione a blocchi dal principio. Per ottenere l'applicazione progettata nella tesi si è modificato in buona parte il diagramma a blocchi predefinito poiché, oltre ad essersi riscontrate numerose incoerenze in fase di programmazione nella configurazione predefinita della *Lenze*, il controllo adatto al servosistema non è un controllo velocità ma un controllo posizione. Inoltre a seconda delle caratteristiche dell'EMA è necessario effettuare lo *scaling* per tutti i dati presenti nel *signal flow*. Il diagramma a blocchi completo verrà analizzato successivamente.
Facendo riferimento alla Figura 40, l'applicazione può essere vista come un scatola al cui interno vi è l'applicazione e ai lati vi sono le porte di ingresso ed uscita collegate a tale applicazione, il tutto configurabile mediante il *FB editor*. Tramite le porte di input a sinistra l'applicazione riceve il *setpoint* e altri comandi analogici e digitali in arrivo dai terminali di input del Driver, mentre a

destra, tramite le porte di output, fornisce i segnali di *feedback* all'uscita dei terminali di output e messaggi di stato (indicatori booleani, limitazioni raggiunte, messaggi di errore etc..).

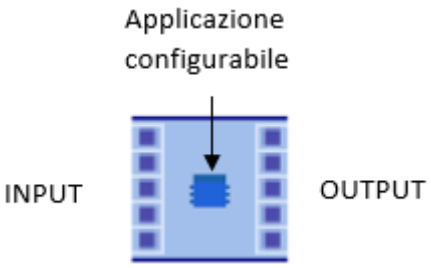


Figura 40: Schema logico di una applicazione all'interno del driver.

Il progetto si basa su un unico controllore e quindi una sola applicazione a se stante; nel caso invece di più driver interconnessi, e dunque di più applicazioni presenti, si adotta un sistema denominato *Machine Application*, il quale interconnette le applicazioni tra di loro mediante porte in ingresso e uscita (Figura 41).

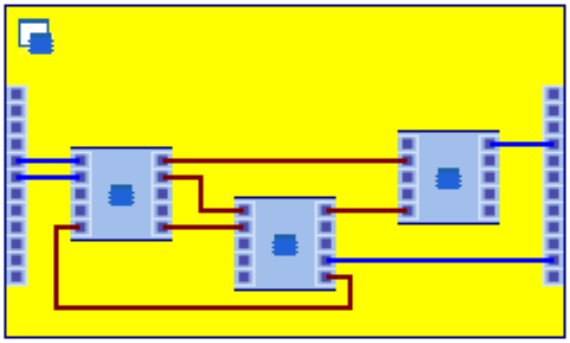


Figura 41: Schema logico di una Machine Application.

Essendo una funzionalità non utilizzata nel progetto, non si entra nella spiegazione di questa configurazione.

3.2.3 Le funzioni dell'applicazione

Entrando all'interno dell'albero di progetto e cliccando sull'icona del driver o dell'applicazione scelta, nell'interfaccia utente appare un elenco a finestra come in Figura 42 (riquadro rosso).

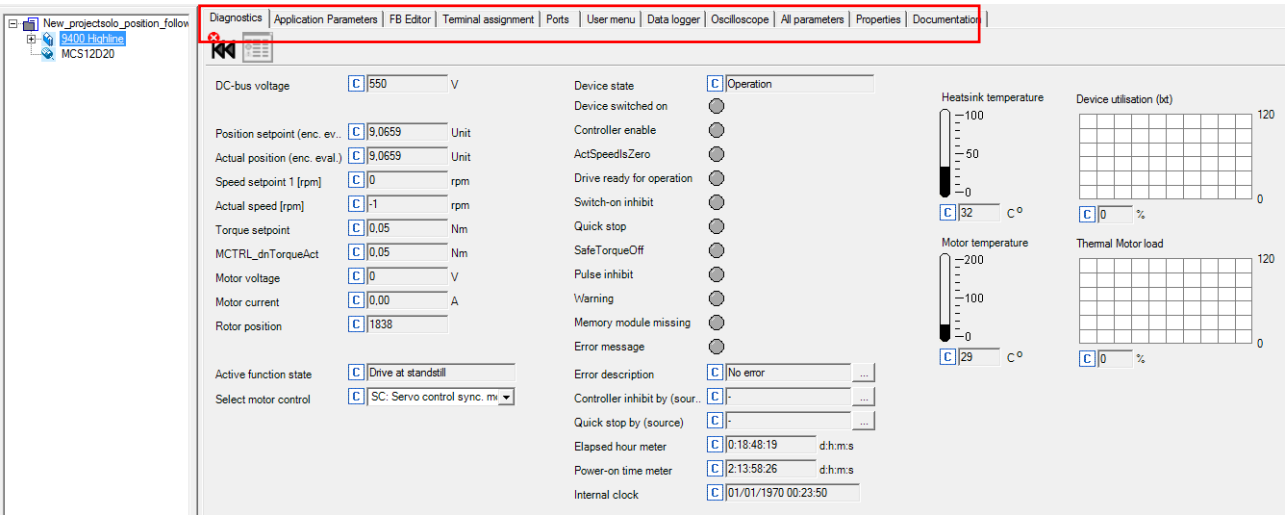


Figura 42: Interfaccia utente: finestre dell'interfaccia utente (cerchiate in rosso) e finestra Diagnostica.

Queste finestre rappresentano la parte fondamentale del progetto, su cui l'utente lavora in fase di progettazione, inserisce i parametri e controlla lo stato del sistema
La prima finestra (*Diagnostic*), visualizzata in Figura 42, è quella di diagnostica in cui si visiona lo stato del driver attraverso i principali parametri mostrati in tempo reale, gli status del driver tramite una stringa o tramite un booleano, la segnalazione di errori, la temperatura e l'utilizzo della resistenza esterna e la temperatura del motore.

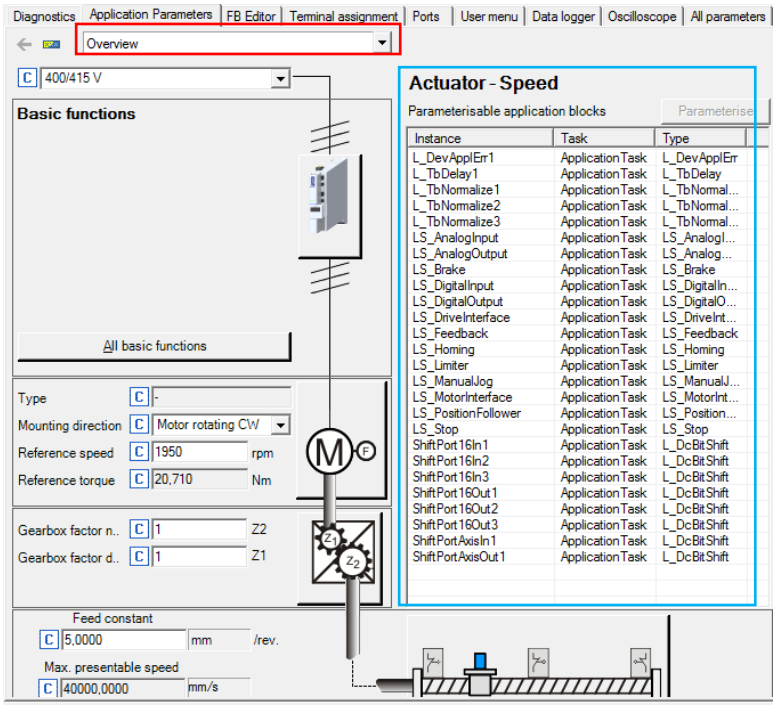


Figura 43: Finestra Application Parameters.

La seconda finestra (*Application Parameters* Figura 43) riporta le funzioni inserite nella programmazione a blocchi del *FB editor* (vedi capitolo *FB editor*) e quindi facenti parte dell'applicazione (riquadro blu) e una finestra a tenda (riquadro rosso); nel caso di Figura 43 è selezionata la finestra *Overview,* la quale mostra le caratteristiche generali dell'EMA cliccando sulle icone di interesse. Navigando su tale finestra si può accedere in modo dettagliato alle sottoclassi del sistema, come riportato in Figura 44.

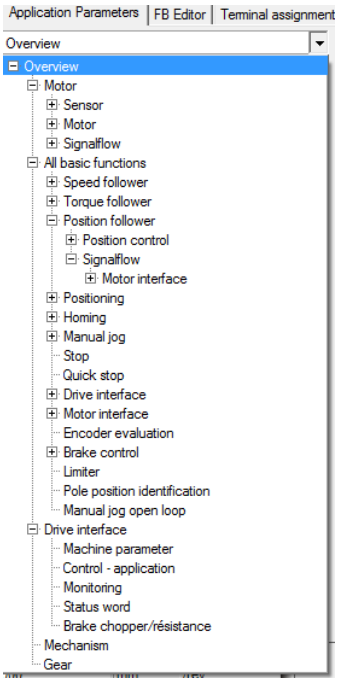


Figura 44: Funzioni disponibili nella finestra a tenda in Application Parameters.

In modo gerarchico sono visualizzate le funzioni principali del programma che si deve creare. Ognuna di queste si tradurrà in funzioni a blocchi che sono il linguaggio di programmazione con cui lavora il Driver e che verranno analizzate in seguito. Selezionando le finestre/funzioni è possibile parametrizzare tali funzioni, in modo che nella programmazione a blocchi (vedi capitolo *FB Editor*) un dato che entra nella funzione venga manipolato in modo diverso a seconda dei parametri inseriti all'interno di essa, in modo da ottenere in uscita un dato elaborato in un certo modo.

Selezionando la funzione motore (*Motor,* Figura 45) si ha la possibilità di leggere a sinistra gli input alla funzione, immettere la corrente massima e selezionare il tipo di controllo del motore in centro (riquadro rosso, nel progetto si è optato per un servocontrollo), e di visualizzare il suo stato sulla destra; cliccando su *Motor* (riquadro azzurro) si apre la finestra dati relativa al motore (Figura 46), mentre cliccando su *Sensor* si apre la finestra inerente al sistema di trasduzione (Figura 47) dove si seleziona come sistema di trasduzione il resolver. Il pulsante *Signal flow (*riquadro giallo*)* apre il diagramma a blocchi del motore di cui si discuterà in seguito nel capitolo Controllo Driver.

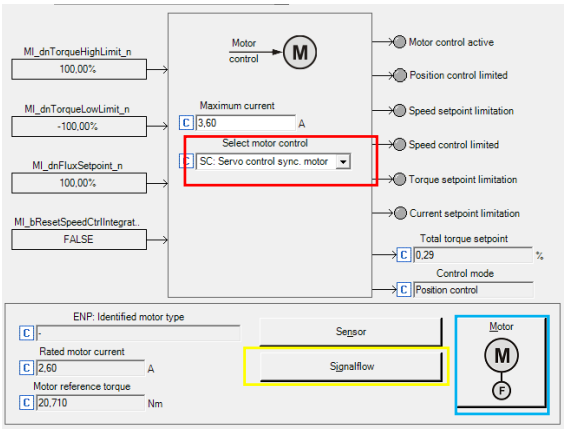


Figura 45: Finestra Motore.

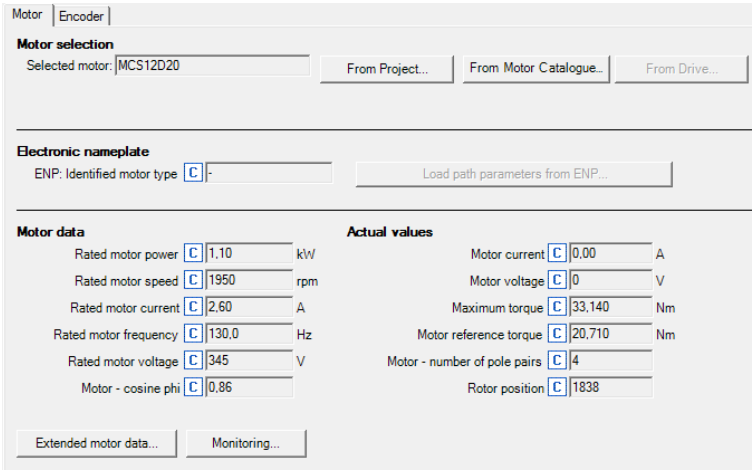


Figura 46: Finestra dati motore.

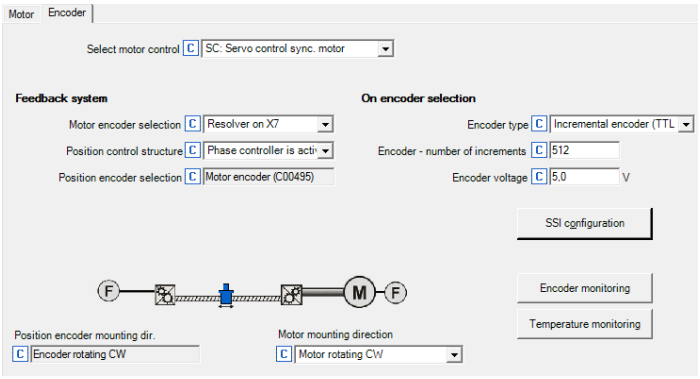


Figura 47: Finestra Resolver.

Continuando a scorrere la finestra di Figura 44, ad un livello gerarchico inferiore del motore si trovano le funzioni strutturate, di cui quelle che interessano sono *Position Follower* (poiché è attuato un controllo in posizione mediante il Driver), che coincide con l'applicazione da abilitare attraverso il segnale digitale discussa nel capitolo 2 che permette all'applicazione intera di essere attivata, e *Homing* che consente di fissare il sistema di riferimento dell'EMA.
La funzione *Position Follower* apre una schermata (Figura 48) simile a quella del motore, in cui si leggono gli input della funzione a sinistra, si immette il valore del guadagno proporzionale in centro e si visualizza lo stato e i principali parametri del controllo posizione a destra. Discorso analogo a quello del motore per il pulsante *Signal Flow (*riquadro rosso*)*.

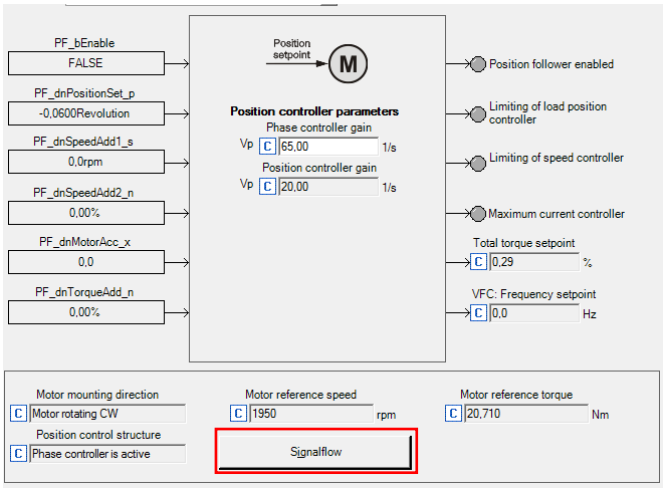


Figura 48: Finestra Position Follower.

La funzione *Homing* permette di trasferire il sistema di misura della macchina al controllore, ovvero di fissare lo zero del sistema con quello del controllore, cosa che sarebbe impossibile da fare non avendo sensori di fine corsa sull'attuatore. L'unico sistema di rilevazione di posizione che si ha a disposizione è il resolver, il quale è assoluto solamente in un giro del motore. Per cui, dopo aver portato l'attuatore nella posizione di 'zero macchina' desiderata, si attiva la funzione *Homing* in modo da salvare nel modulo di memoria del Driver la posizione di zero di quello che sarà il sistema di riferimento per il controllore, facendo sì che per le successive rotazioni orarie e antiorarie del resolver rimanga sempre in memoria la posizione di partenza per la quale si è fissata lo zero, anche dopo lo spegnimento del driver.Di conseguenza, è necessario impostare solamente la prima volta la posizione di zero del controllore, a meno di volerla modificare in seguito al cambiamento dello zero del sistema. Grazie al salvataggio della posizione di zero, il Driver ricava in modo relativo da tale posizione l'attuale angolo di rotazione del motore e ne deriva la traslazione dell'attuatore considerando il passo selezionato.
L'interfaccia della funzione *Homing* è presentata in Figura 49.

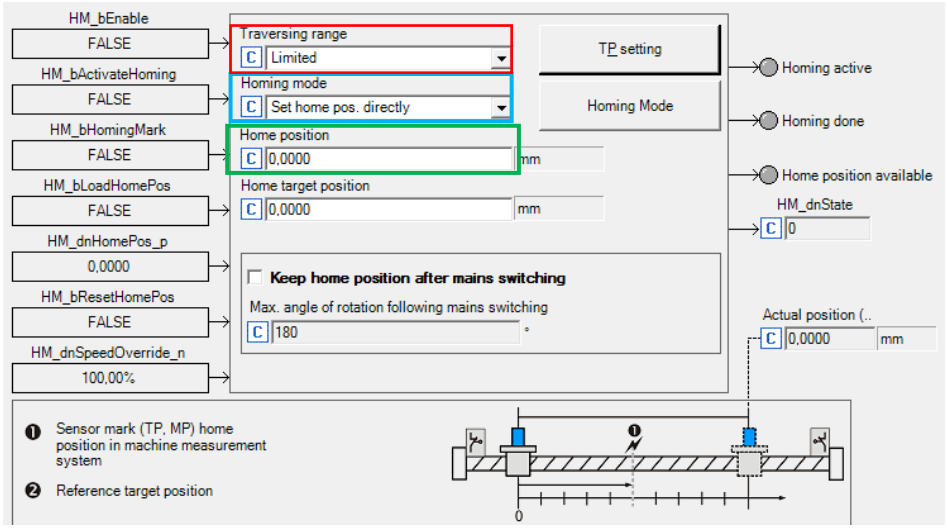


Figura 49: Finestra Homing.

In modo analogo alle altre funzioni studiate, sono presenti a sinistra gli input alla funzione, in centro il settaggio di tale funzione e a destra il suo status. Per *Traversing Range* (riquadro rosso) si intende la tipologia di attuazione adottata per il carico e ve ne sono tre diversi tipi. Si sceglie il metodo *Limited* il quale consiste nello schema di Figura 50.

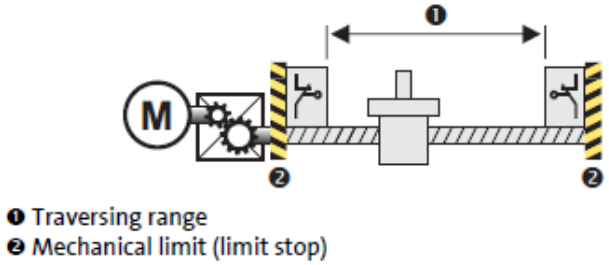


Figura 50: Schema del metodo di attuazione Limited.

Si tratta semplicemente di scegliere il tipo di movimento che esegue l'attuatore, che nel nostro caso è lineare e limitato, ed è quindi assimilabile ad una madrevite che trasla e la vite che ruota, anche se in realtà nel sistema in studio la madrevite ruota e la vite trasla; l'alternativa è un attuatore rotativo, il quale non rientra nei nostri interessi.

Nel riquadro azzurro è selezionata la modalità *Homing Mode* di settaggio manuale dello zero di riferimento, in modo che il valore inserito nella casella *Home Position* sia quello di zero. Per cui, facendo riferimento al capitolo Relè e Cablaggi, quando si attivano i due *switches* del *case* 1 verrà impostato come riferimento di zero nel controller il valore immesso nella casella *Home Position* (riquadro verde), che rimarrà tale fino ad una diversa impostazione dello zero comandata dall'utente.
La finestra *FB editor* consiste nell'ambiente di programmazione a blocchi, necessario per poter gestire i segnali del Driver. Verrà trattata in seguito nel capitolo *FB Editor*.
Nella successiva finestra in Figura 51, dal nome *Terminal Assignments,* sono visualizzati gli I/O analogici e digitali. Si può osservare per i terminali digitali, in tempo reale, lo status segnalato da un led che indica i segnali digitali attivati e la possibilità di invertire la polarità del segnale digitale (a fronte positivo o a fronte negativo, riquadro rosso); per i terminali analogici si visualizza il valore in volt e vi è la possibilità di modificare le impostazioni del segnale (riquadro azzurro). Per entrambi è visibile l'assegnazione tramite il nome conferito nel *FB Editor*. Al centro è rappresentata la disposizione dei pin nella morsettiera e la sigla del segnale.

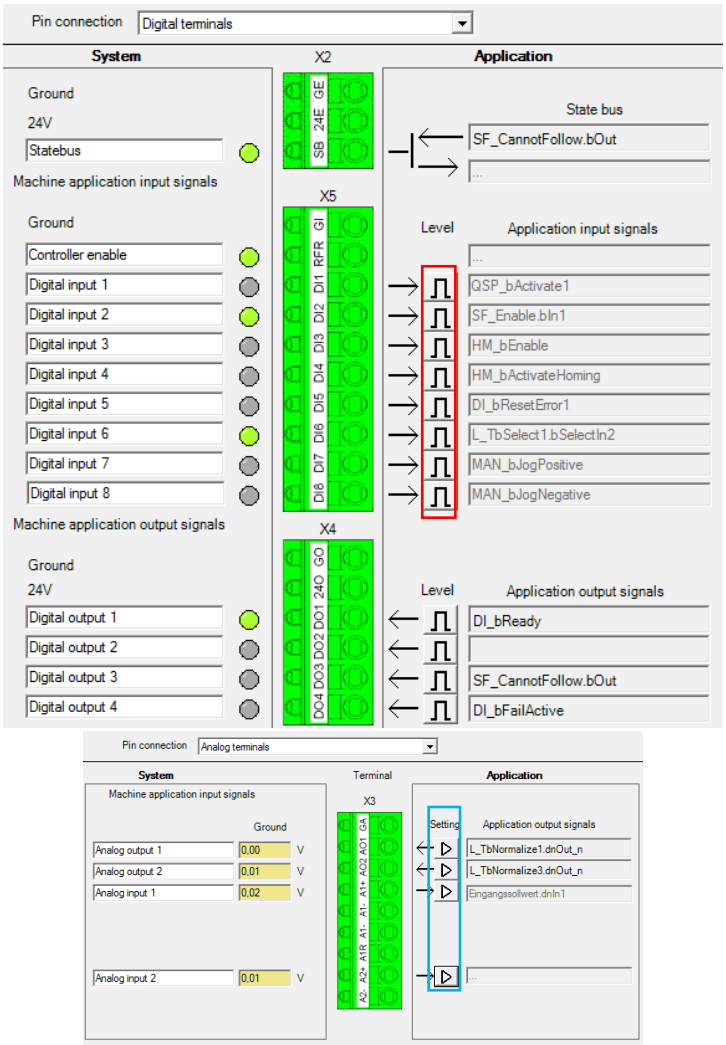


Figura 51: Finestra Terminal Assignments: assegnazione terminali digitali (sinistra) e terminali analogici a (destra).

I segnali utilizzati sono presentati in Tabella 26.

Sigla	Segnale
RFR	Abilitazione Driver
DI2	Abilitazione Position Follower
DI3	Abilitazione Homing
DI4	Attivazione Homing
DI6	Selezione Feedback Corrente/Velocità
AI1	Setpoint Posizione
AO1	Feedback Posizione
AO2	Feedback Corrente/Velocità

Tabella 26: Segnali utilizzati nel progetto e relative sigle.

Premendo il pulsante *Setting* (riquadro azzurro) nella finestra terminali analogici si apre una finestra in cui è possibile impostare l'*offset*, la banda morta, il guadagno e il range del segnale prima che questo venga trasmesso al controllo (Figura 52).

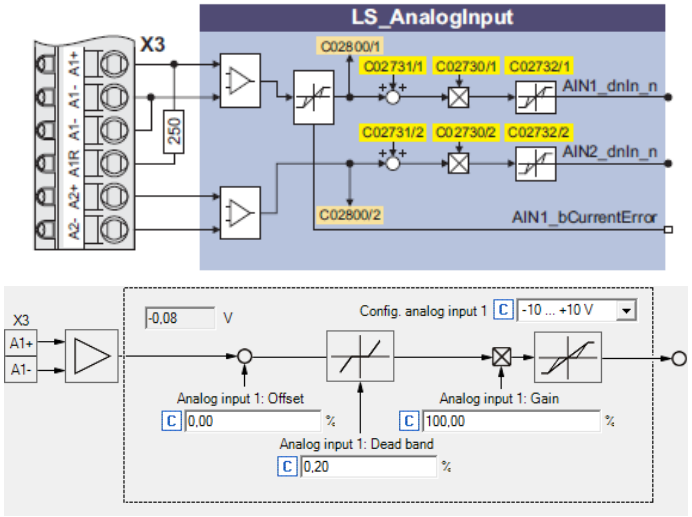


Figura 52: Schema della funzione Analog Input (a sinistra) e setting in un input analogico (a destra).

È possibile notare un certo valore di banda morta che è stato inserito per l'input di posizione: in questo modo il segnale in ingresso è inibito e risulta uguale a zero nell'intorno della percentuale inserita.
Attraverso una configurazione simile, per il segnale analogico in uscita è possibile impostare un offset e il guadagno prima che il segnale sia mandato in uscita alla morsettiera (Figura 53).

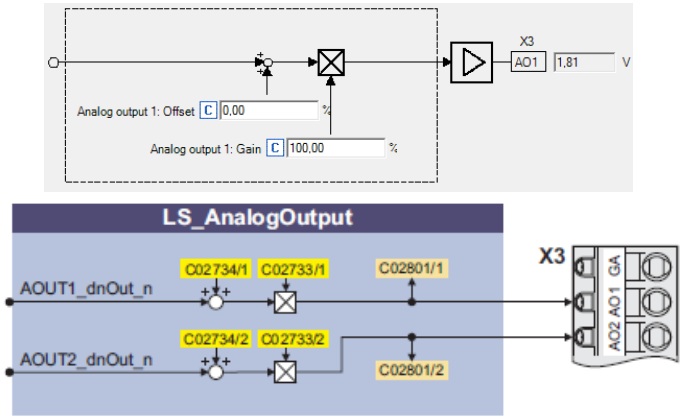


Figura 53: Schema della funzione Analog output (a destra) e setting in un input analogico (a sinistra).

Per il progetto creato, l'acquisizione e il monitoraggio sono svolti mediante il software *LabVIEW*. Ciononostante è possibile sia acquisire che monitorare i segnali tramite il software *Lenze Engineer*.
La finestra successiva è il *Data Logger* (Figura 54), il quale permette di selezionare la variabili di interesse, avviare l'EMA e visualizzarle su un grafico non in *Real Time*, poiché registra il valore dei parametri solo con l'ausilio del PC e non con un sistema *embedded* in grado di acquisire in modo deterministico.

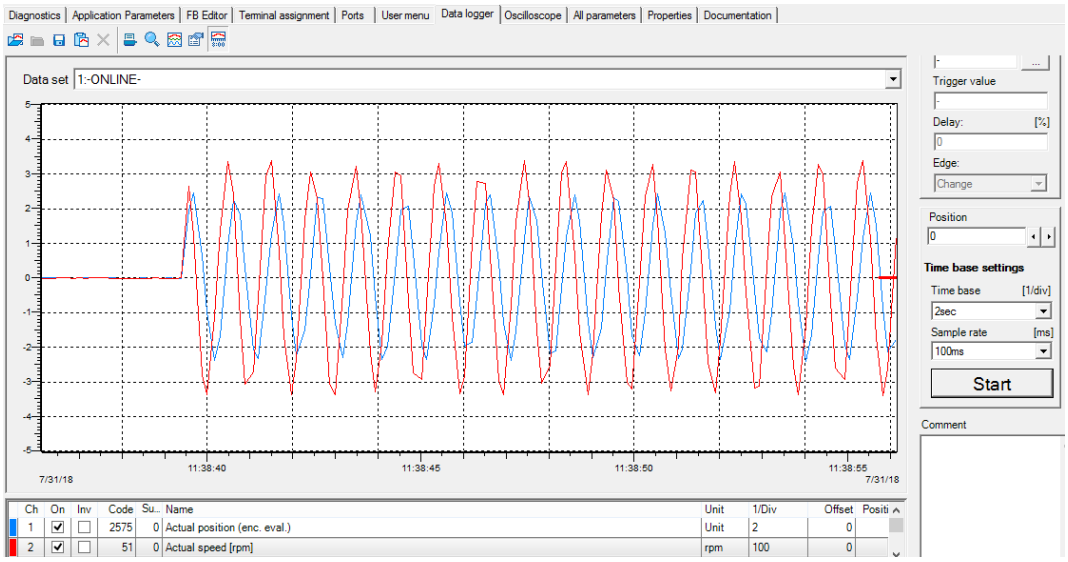


Figura 54: Finestra Data Logger.

Se invece si vuole effettuare una acquisizione in *Real time* si può utilizzare l'oscilloscopio integrato nel controllore mediante l'utilizzo della finestra *Oscilloscope* (Figura 55*)*.

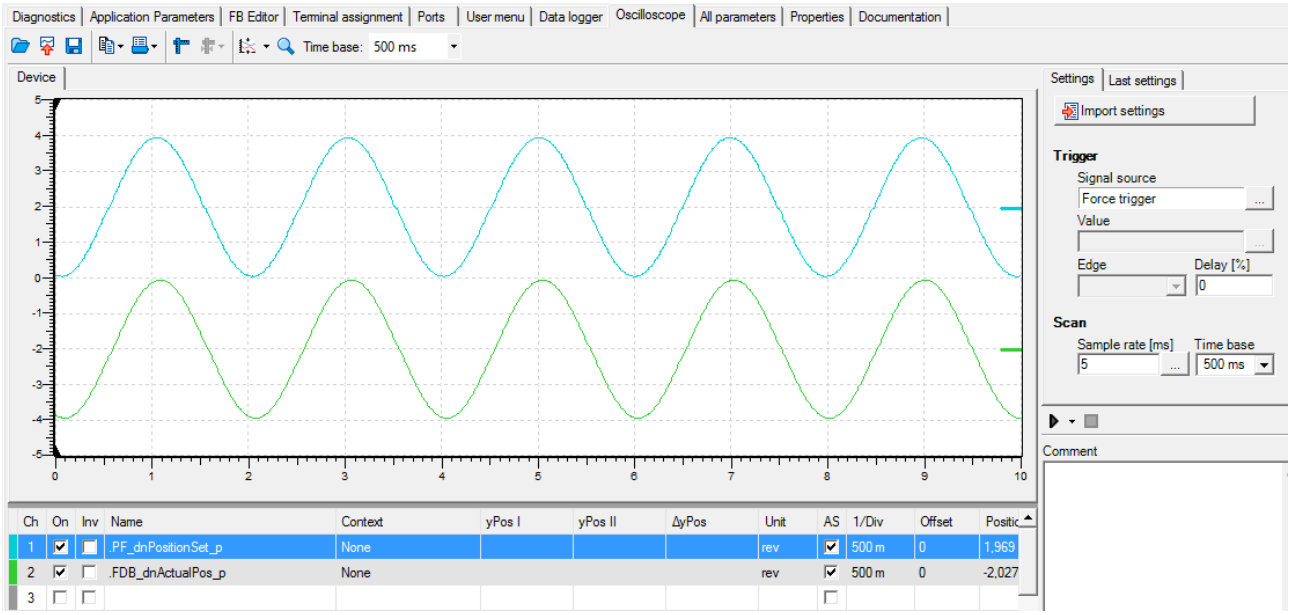


Figura 55: Finestra Oscilloscope.

L'ultima finestra di interesse è denominata *All Parameters* (Figura 56), inerente a tutti i parametri di lettura e scrittura presenti nel Driver utili per la programmazione: permette di effettuare la ricerca sia inserendo il nome o il codice del parametro, oppure ricercando all'interno delle categorie. Risulta comodo poiché il software *LE* possiede un'elevata quantità di finestre per cui, a volte, trovare un determinato parametro risulta alquanto complesso.

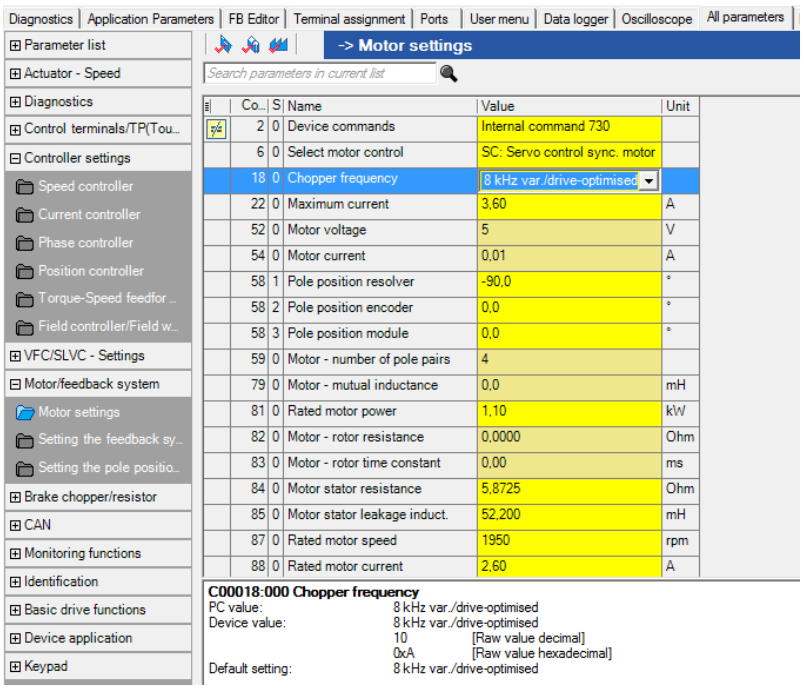


Figura 56: Finestra All Parameters.

3.3 Programmazione mediante funzioni a blocchi

La programmazione a blocchi sul software della *Lenze* permette di personalizzare l'applicazione scelta (attuatore di velocità nel progetto adottato) attraverso un flusso di dati costituito da blocchi: la programmazione a blocchi è raggiungibile attraverso la finestra utente *FB Editor* (*Function Block Editor*). L'ambiente di programmazione consiste in blocchi inseribili, che possono essere del tipo funzioni, variabili o costanti, e frecce che indicano la direzione di tale flusso. Inoltre permette di manipolare le porte di ingresso ed uscita analogiche e digitali, in modo da scegliere quale segnale (o segnali) utilizzare come input nella programmazione e quali segnali rendere disponibili in uscita al Driver e in quale pin. Infine consente di utilizzare le funzioni, che consistono in quelle parametrizzabili e analizzate nel capitolo precedente, il cui utilizzo personalizza l'applicazione manipolando il flusso di dati a seconda della funzione utilizzata e di come questa è stata parametrizzata nella finestra *Application Parameters*, rendendo l'applicazione idonea all'utilizzo ricercato. Riassumendo, nelle finestre utente si parametrizzano le funzioni da inserire nel *FB editor*, mentre nel *FB editor* si inseriscono tali funzioni e si programma il flusso di dati in modo da ottenere un *signal flow,* elaborato mediante funzioni, variabili e costanti, che parte dalle porte di ingresso e giunge alle porte di uscita.

3.3.1 Il Function Block Editor

Come accennato in precedenza, il *FB editor* è a disposizione dell'utente e risulta fondamentale per la personalizzazione delle funzioni utilizzate nell'applicazione e per elaborare i dati che saranno disponibili in uscita al Driver. La logica di programmazione del *FB Editor* consiste in un flusso di dati di svariata natura (incrementi, percentuali, rpm…) avente inizio dai blocchi di input sulla sinistra, per poi essere manipolato dalle funzioni inserite nell'applicazione e reso disponibile ai blocchi di output sulla destra. Tale programmazione, da certi punti di vista, è simile a quella utilizzata dal software *LabVIEW* (vedi capitolo Software *LabVIEW*). In questo modo l'applicazione da caricare nel controllore, parametrizzata tramite le funzioni disponibili nelle finestre dell'interfaccia utente studiate in precedenza, è programmata mediante un flusso di blocchi che possono essere del tipo funzioni, logiche aritmetiche, variabili, costanti, input ed output.
Entrando nel dettaglio del diagramma a blocchi progettato si presenta quest'ultimo in forma completa in Figura 57.

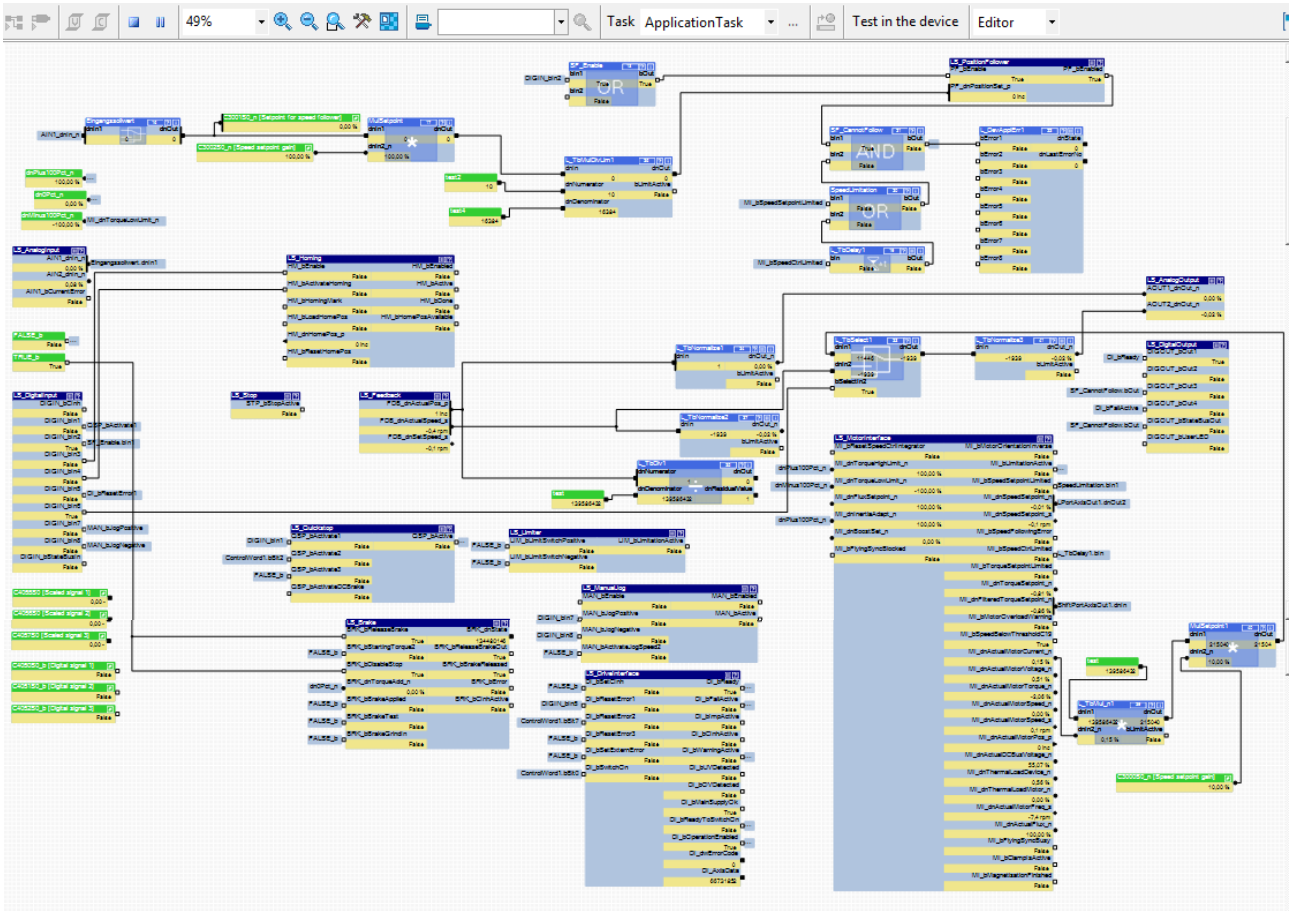


Figura 57: Diagramma a blocchi completo progettato per l'applicazione di progetto nel FB editor.

Il *FB editor* dispone di una *toolbar,* mediante la quale è possibile svolgere diverse operazioni.

Figura 58: Toolbar del FB editor.

Riferendoci alla Figura 58, permette di:

1. Inserire rispettivamente *funzioni base* come quelle necessarie alla conversione da una tipologia di dato ad un'altra, e *funzioni strutturate* come ad esempio *Position Follower*, *Homing* e *Motor*, discusse in precedenza;
2. Inserire delle costanti a blocchi fondamentali per la programmazione;
3. Monitorare il programma creato e svolgere il *debugging*;
4. Zoommare il *FB editor* a piacimento;
5. Ricercare un determinato blocco nell'ambiente di programmazione.

Ingrandendo diverse zone dell'ambiente di programmazione mostrato in Figura 57, si analizza ciascuna sezione in dettaglio.

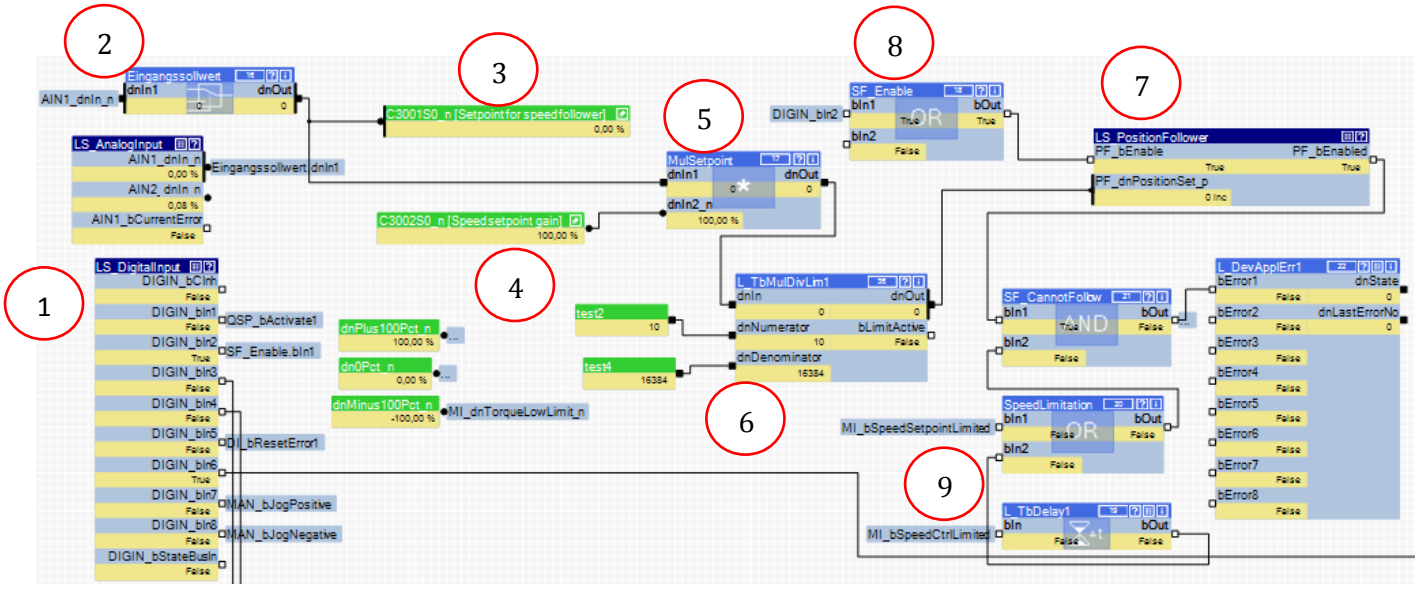


Figura 59: FB Editor: Ingrandimento 1.

Per quando riguarda la sezione in Figura 59, a sinistra vi sono le porte A/D di ingresso (1), il *setpoint* (2) viene trasmesso ad un blocco *switch* per il momento inutilizzato e ne viene visualizzata la percentuale rispetto al range (3), in seguito è moltiplicato da una funzione base (5) per un guadagno pari a 1 (100%) (4). In uscita il valore di riferimento posizione è un numero del tipo DINT ($\pm 2^{30} = \pm 1073741824$) e deve essere tradotto in incrementi (INC) mediante una funzione base (6) per poter entrare con il dato corretto (in incrementi) nella funzione a blocco *Position Follower* (7). Si mostrano in Tabella 27 le scalature (lo *scaling*) per i diversi segnali e dati trattati.

Signal type	Connection symbol in the FB editor	Resolution	Scaling External value	≡ internal value
Scaled (INT)	○	16 bits	100 %	$\equiv 2^{14} \equiv 16384$
Scaled (DINT)	●	32 bits	100 %	$\equiv 2^{30} \equiv 1073741824$
Speed (INT)	◁/▷	16 bits	15000 rpm	$\equiv 2^{14} \equiv 16384$
Speed (DINT)	◆	32 bits	15000 rpm	$\equiv 2^{26} \equiv 67108864$
Position/angle (DINT)	◀/▶	32 bits	1 encoder revolution	$\equiv 2^{16}$ increments
Acceleration (DINT)	■	32 bits	15000000 rpm/s	$\equiv 2^{22} \equiv 4194304$

Tabella 27: Conversioni da effettuare a seconda del tipo di dato e segnale trattato.

Detto ciò, si inserisce una funzione base e due costanti (6) in modo da convertire setpoint e renderlo disponibile alla funzione *Position Follower* secondo il seguente ragionamento: una rivoluzione dell'encoder corrisponde a 2^{16}= 64536 incrementi, a cui corrisponde un passo dello stelo di 5 millimetri, per cui schematizzando la corsa dell'attuatore:

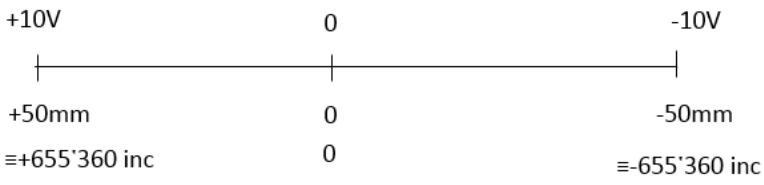


Figura 60: Schema della corsa dell'attuatore per effettuare lo scaling DINT<->INC del Set di posizione.

A causa della ridotta disponibilità di funzioni e in aggiunta che i blocchi 'costante numerica' possono contenere solamente numeri interi, si inserisce una funzione a blocco a due variabili, 10 a numeratore e 16384 a denominatore, in modo che lo *scaling* funzioni secondo le seguenti equazioni (eq.3).

$$\frac{1073741824}{655360} = 1638{,}4 \rightarrow fattore\ di\ scala \qquad \frac{10}{16384} = 1638{,}4 \qquad (3)$$

In seguito il valore corretto di *Setpoint* entra nel blocco *Position Follower* (7), blocco che deve essere attivato tramite il consenso digitale DI2 (8) (Enable Position Follower≡Abilitazione Applicazione); in basso (9) sono inseriti alcuni blocchi logici in modo che se il motore superi il limite di velocità imposto nella funzione, allora appaia nella schermata di diagnostica un messaggio di errore.

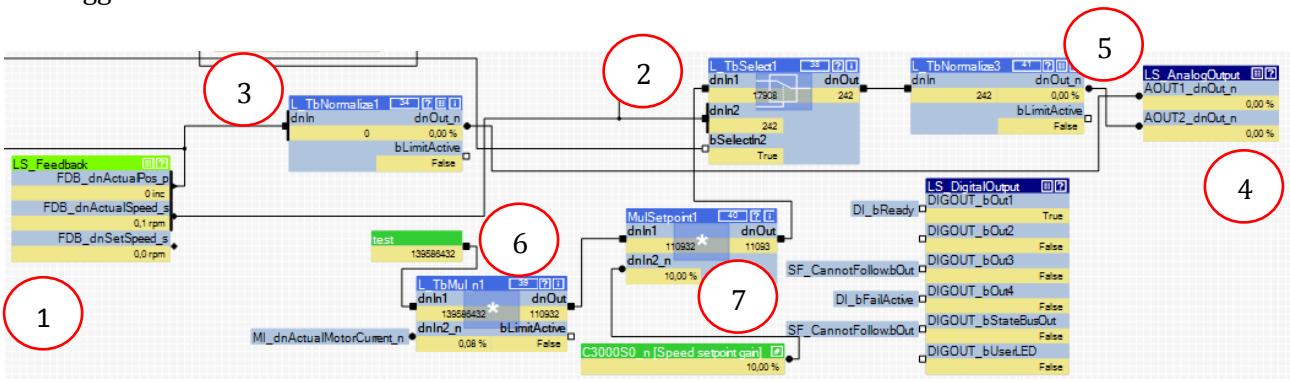


Figura 61: FB editor: Ingrandimento 2.

Analizzando il ramo di retroazione in Figura 61, dalla funzione *Feedback (1)* giungono i segnali di *feedback* come dato di posizione in incrementi, e come dato di velocità in *rpm*; in modo analogo al ramo di setpoint, è necessario convertire i segnali in modo che questi possano essere accettati dalla porta di uscita analogica, la quale accetta solo dati in percentuale.
Inoltre si dispone solamente di due uscite analogiche, per cui, oltre al *feedback* di posizione, si incrementa il numero di opzioni di segnali trasdotti mediante una funzione base a blocco *select* (2) comandata dal DI6 (Selezione Feedback corrente/velocità ≡ *2nd Feedback*), che permette di retroazionare alternativamente la corrente o la velocità, oltre alla posizione che è indipendente da tale blocco. Il dato di posizione è convertito da incrementi a percentuale mediante una funzione base a blocco *Normalize* (3), la quale, cliccando su tale blocco in alto a destra, permette di introdurre una costante da immettere a denominatore e di moltiplicare per 100, in modo da convertire il dato in percentuale e mandarlo alla porta analogica di uscita (4). Elaborando lo stesso discorso effettuato per il *setpoint* si ottiene che tale costante è di 655360 (eq.4).

$$FB_{x\,\%} = \frac{FB_{x\,INC}}{655360} \cdot 100 \qquad (4)$$

Riguardo al secondo segnale retroazionato da acquisire, nel blocco *select* (2) entrano il dato di corrente e il dato di velocità. La corrente è prelevata in uscita dalla funzione a blocco *Motor Interface* (Figura 63) ed è espressa in percentuale. Il problema è che il blocco *select* (2) non riceve in ingresso dati in percentuale, per cui si effettua una conversione. La corrente massima del dispositivo risulta 16 Ampere, sapendo che l'uscita analogica presenta un range ±10 Volt ne consegue che il fattore di correzione per la corrente è di 1.6 A/V (si otterrà eseguendo nelle eq. 3-4 l'operazione $\cdot\frac{16}{10}$, i blocchi accettano solo costanti intere). Inoltre nel blocco *Normalize* (5) è necessario dividere per la costante 8724152 (eq.6: *scaling* di velocità) per convertire la velocità

in percentuale nel caso in cui, attraverso il blocco *select* venga scelta quest'ultima. Di conseguenza, nel caso venisse scelta la corrente, questa costante che non deve essere considerata si deve elidere (eq.5-7). Per cui, secondo quanto detto sopra, nel primo blocco moltiplicativo (6) del ramo *feedback* di corrente si effettua l'operazione (eq.5):

$$FB_{i\,,} = FB_{i\,\%} \cdot 16 \cdot 8724152 \qquad (5)$$

Mentre nel secondo blocco moltiplicativo (7) si effettua:

$$FB_{i\,INC} = \frac{FB_{i\,,}}{10} \qquad (6)$$

In seguito nel blocco *Normalize* (5) si divide per 8724152 (eq.7), tale valore si elide e si ottiene il valore in percentuale sulla porta analogica in uscita.
Per il segnale di velocità, il dato in uscita dal blocco *Feedback* (1) è in giri al minuto, per cui si entra direttamente nel blocco *select (2)* e si normalizza (3) solamente in uscita da esso:

$$FB\ v_{\,\%} = \frac{FB\ v_{\,rpm}}{8724152} \cdot 100 \qquad (7)$$

Il valore 8724152 si ottiene dallo *scaling* di velocità (Tabella 27), secondo cui 15000 giri al minuto $\equiv 2^{26}$=67108864 e quindi:

$$\frac{15000}{67108864} = \frac{1950}{x} \rightarrow x = 8724152 \qquad (8)$$

La velocità di targa del motore è di 1950 giri al minuto.
Dividendo per tale costante (eq.8) ottengo il valore in percentuale della velocità da utilizzare per la porta analogica di uscita.
Le uscite digitali non sono utilizzate nel nostro progetto per cui sono di scarso interesse.
Per ciò che riguarda le funzioni strutturate, si rappresentano nelle Figura 62 e Figura 63.

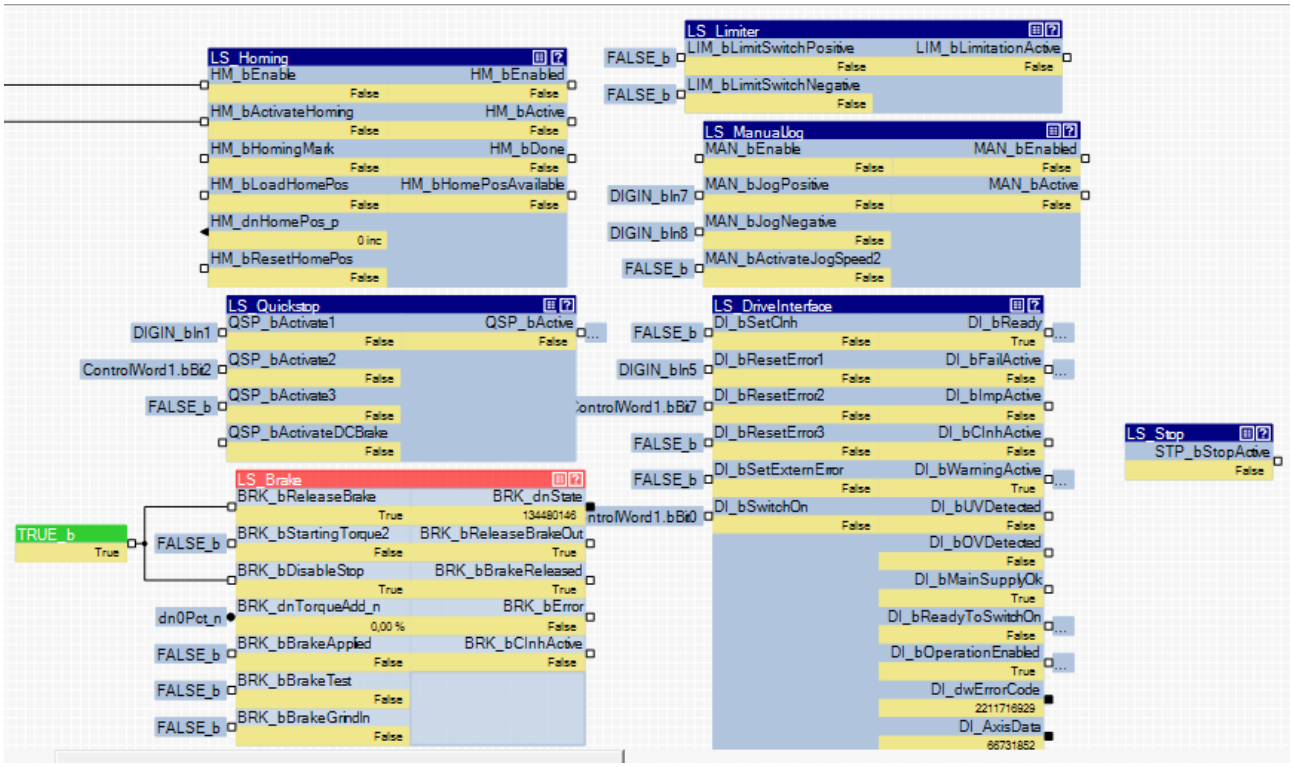


Figura 62: Funzioni Homing, Limiter, Manual Jog, Quick Stop, Drive Interface, Brake e Stop.

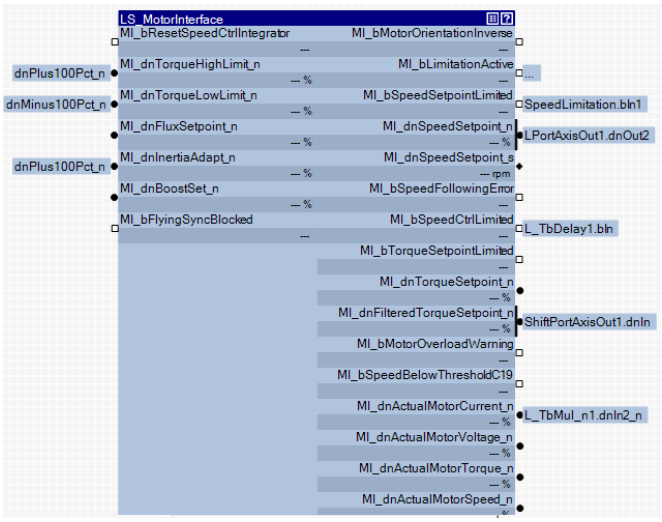


Figura 63: Funzione Motor Interface.

La funzione *Homing* necessita la contemporanea presenza dei due input digitali per poter fissare lo zero nella memoria del Driver (attivazione e abilitazione); la funzione *Quickstop* per il momento non è utilizzata cosi come la *Stop*; la funzione *Brake* viene impostata su vero in modo da by-passarla; la funzione *Limiter* e *Manual Jog* non sono utilizzate per il momento; la funzione *Driver Interface* permette di ottenere lo stato in cui si trova il driver e di valutare eventuali errori mediante stringhe sull'interfaccia utente.
Sono inoltre presenti ulteriori porte inter-applicazione ma che non vengono utilizzate poiché, come detto in precedenza, si utilizza un unico driver e quindi una sola applicazione.
Il *FB Editor,* oltre ad essere fondamentale per la progettazione del controllo, risulta anche importante in fase di *debugging* dell'applicazione poiché permette di visualizzare in tempo "reale" nei blocchi gli stati delle funzioni e il valore dei dati durante il funzionamento.

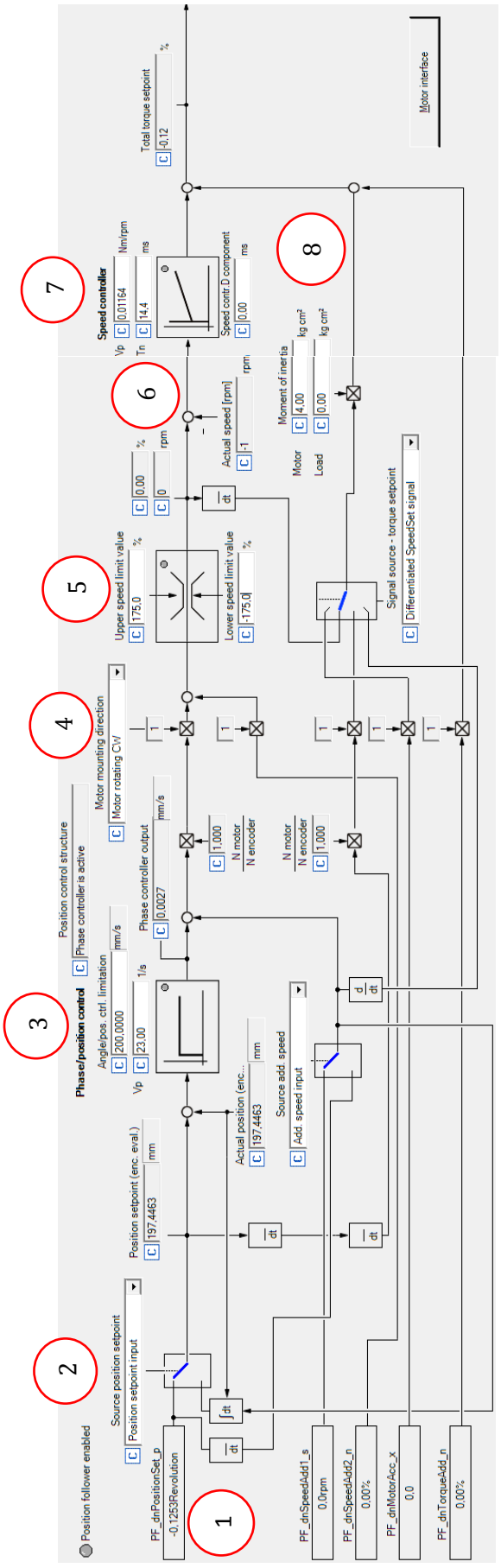
La programmazione a blocchi mediante il *FB editor* intrecciata al controllo interno del driver, di cui si discuterà in seguito, permette di comandare e monitorare il servosistema in maniera comoda ed efficace, sia in fase di progettazione che durante le verifiche sperimentali.

3.4 Controllo implementato nel Driver

Il controllo all'interno del driver può essere di diversi tipi a seconda della scelta utente, il quale può scegliere di controllare il motore in posizione inserendo la funzione a blocco *Position Follower* nella programmazione, oppure controllo velocità o coppia inserendo le rispettive funzioni *Speed Follower* e *Torque Follower*. Inserendo questi blocchi automaticamente il software richiede la creazione di una nuova applicazione o la sovrascrittura dell'attuale applicazione, poiché non può gestire diversi tipi di controllo nella stessa applicazione.
Inoltre si può scegliere tra diversi metodi di controllo motore (Figura 45) tra i quali servocontrollo, controllo vettoriale e controllo V/f. All'interno di ciascun controllo sono disponibili altre sotto-metodologie di controllo tra cui ad esempio il controllo di fase o di posizione.
Per quanto riguarda il controllo del motore, questo dipende anche dalla scelta del motore utilizzato e caricato all'interno dell'applicazione.
L'applicazione progettata per il driver prevede un controllo posizione (*Position follower*), con il metodo servocontrollo di fase. A questo punto l'applicazione carica automaticamente in una sezione denominata *Signal Flow* dei diagrammi a blocchi opportuni per il controllo e non modificabili, a parte l'inserimento di alcuni parametri e alcune scelte che verranno discusse in seguito. Queste sezioni sono raggiungibili tramite la finestra *Application Parameters*, selezionando la funzione *Position Follower* e *Motor,* premendo sul tasto *Signal Flow* (riquadro rosso Figura 48). Unendo i diagrammi a blocchi contenuti nelle sezioni *Signal Flow* delle funzioni *Position Follower* e *Motor* si ottiene l'intero diagramma a blocchi di controllo driver: trattandosi di un servocontrollo di un attuatore elettromeccanico, presenta la tipica struttura a tre anelli di posizione, di velocità e di corrente.

3.4.1 Diagrammi a blocchi del controllo

Il controllo posizione implementato nel driver è strutturato con tre anelli annidati chiusi in retroazione: anello posizione, anello velocità e anello corrente. Lo studio del controllo è riassunto in due diagrammi a blocchi: il primo ingloba il controllo dell'anello di posizione e di velocità (funzione *Position Follower*), mentre il secondo contiene l'anello di corrente (funzione *Motor*).
Nei diagrammi a blocchi alcune unità di misura sono fisse mentre altre sono scelte dall'utente navigando nelle finestre delle relative sezioni utente di impostazione parametri. Il controllo all'interno del driver è altamente personalizzabile mediante aggiunte di *Feedforward*, disturbi, pulsanti di *switch* etc.. che possono rendere il controllo più accurato in base alla necessità dell'utente. Inoltre i parametri con il riquadro a sfondo bianco o giallo (se il driver è *offline* o *online*) sono impostabili direttamente dall'utente, questo vale ad esempio per i guadagni, le saturazioni e altri parametri. Questa opzione risulterà fondamentale in fase di taratura dei guadagni del controllo del driver. In seguito verrà commentato il diagramma a blocchi progettato per l'applicazione scelta.



Facendo riferimento alla

Figura 64, il diagramma a blocchi della funzione *Position Follower* comprende l'anello di posizione e l'anello di velocità. La variabile di riferimento in ingresso al controllo (1) è la stessa in ingresso alla funzione a blocco *Position Follower* presente nel *FB Editor*, tradotta da incrementi a giri. In seguito è presente uno *switch* (2) che permette di modificare il controllo e di dare una posizione di riferimento derivata dall'integrazione di un *set* di velocità. In seguito, il *Set* di posizione è confrontato con il *Feedback* di posizione in retroazione dal resolver e l'errore ottenuto viene gestito da un controllore proporzionale (3), il quale possiede inoltre una limitazione legata alla velocità in [mm/s]. Si può notare come la scelta della direzione di montaggio del motore (4) influisca sul controllo.

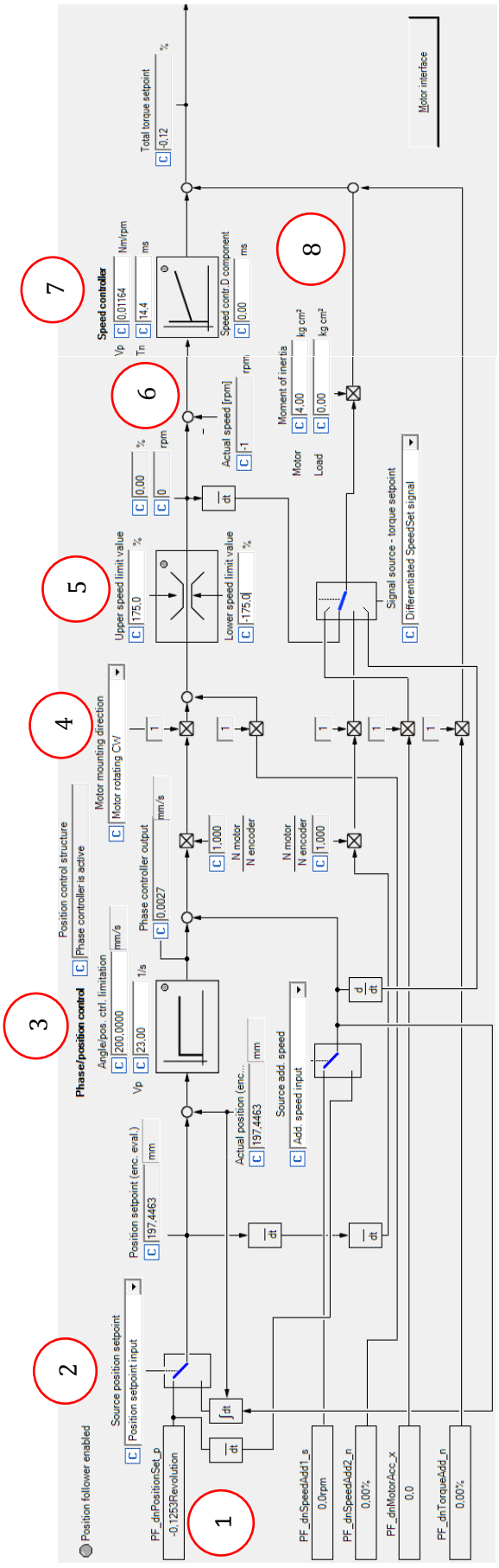


Figura 64: Diagramma a blocchi della funzione Position Follower: anello di posizione e anello di velocità.

In seguito è presente una saturazione di velocità (5): le saturazioni nella logica di controllo del Driver sono impostate come percentuale del valore di targa del motore. A questo punto è chiuso l'anello di velocità con il segnale di *Feedback* di velocità retroazionato dal resolver (6) e l'errore in uscita è compensato attraverso un controllore di velocità PID (7), che converte tramite i guadagni l'errore di velocità in un segnale di coppia, che viene ulteriormente sommato da un contributo di coppia che si ottiene moltiplicando l'accelerazione attuale per l'inerzia riportata sull'albero motore (vedi capitolo modellazione) (8). Si ottiene così la coppia di riferimento che si trasforma nell'input alla funzione *Motor*.
Il diagramma a blocchi del motore rappresenta l'anello di corrente del controllo, ed è presentato in Figura 65.

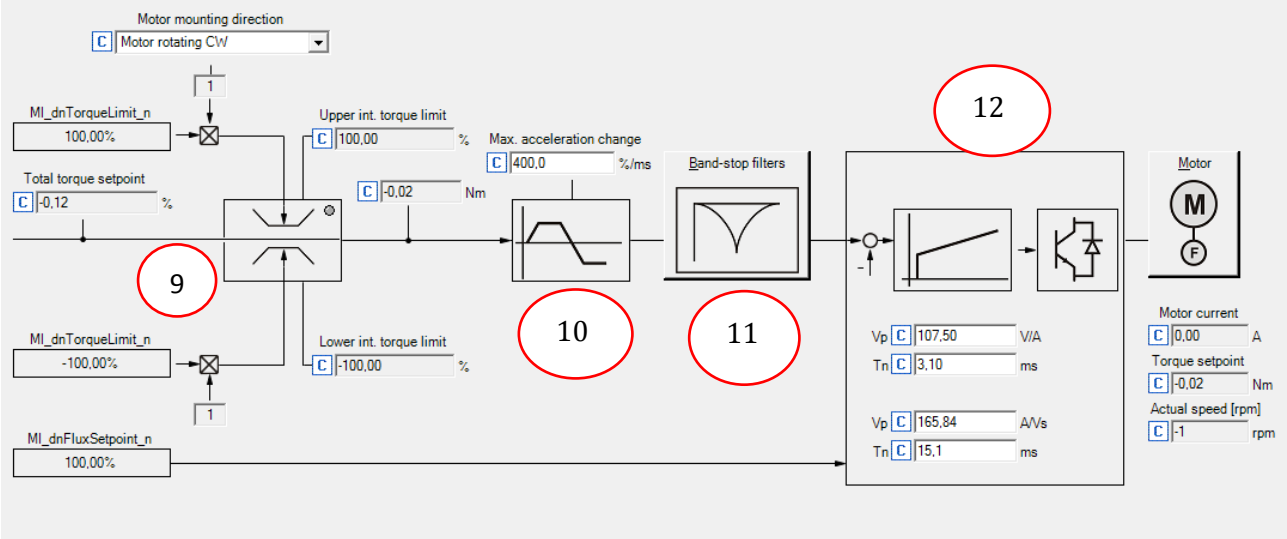


Figura 65: Diagramma a blocchi funzione Motor: Anello di corrente.

La coppia risulta la variabile di riferimento in ingresso: viene saturata in coppia da un primo blocco inserendo come limiti le percentuali rispetto al valore di targa (9) e saturata da un secondo blocco in base alla velocità con cui può variare la coppia (10) (valore impostato alla sua massima capacità). In seguito è posto un filtro di banda (11), all'interno del quale è possibile impostare i valori. A questo punto si chiude l'anello di corrente e si compensa l'errore di corrente con un controllore PI (12), i cui guadagni trasformano il segnale di corrente in un segnale di tensione, il quale è inviato all'inverter che attraverso la sua elettronica manda le correnti alle tre fasi del motore.
Per poter comprendere meglio il funzionamento del diagramma a blocchi della funzione *Motor* è necessario ricorrere alla guida del driver, in cui è presente un diagramma maggiormente dettagliato, presentata in Figura 66.

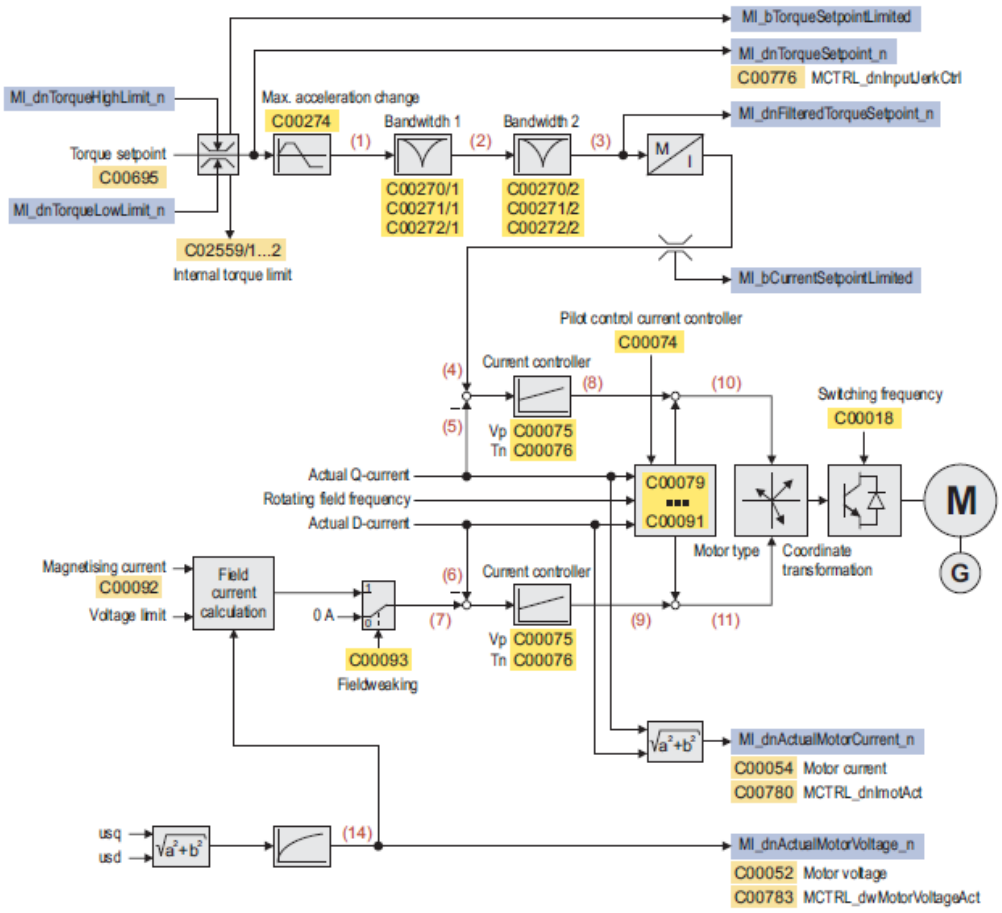


Figura 66: Diagramma a blocchi controllo motore (guida del driver Lenze 9400).

Si nota dalla Figura 66 come il controllo motore all'interno del driver sia più articolato rispetto al diagramma a blocchi mostrato nel *Signal Flow* della funzione *Motor* in Figura 65.

3.4.1.1 *Field Oriented Control* FOC

A questo proposito, è necessario introdurre il controllo motore del tipo FOC (*Field Oriented Control*), il quale si basa sul comando sinusoidale in corrente delle tre fasi con un ritardo rispettivamente di 120° ognuna. Questo metodo consiste in un algoritmo il quale consente di controllare la corrente nelle tre fasi del motore e quindi le componenti vettoriali del campo magnetico statorico che ne conseguono. La Figura 67 mostra la sezione di un motore e tre diversi sistemi cartesiani.

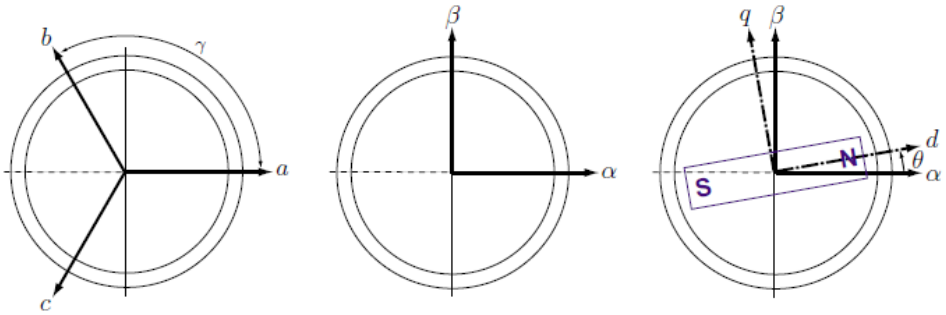
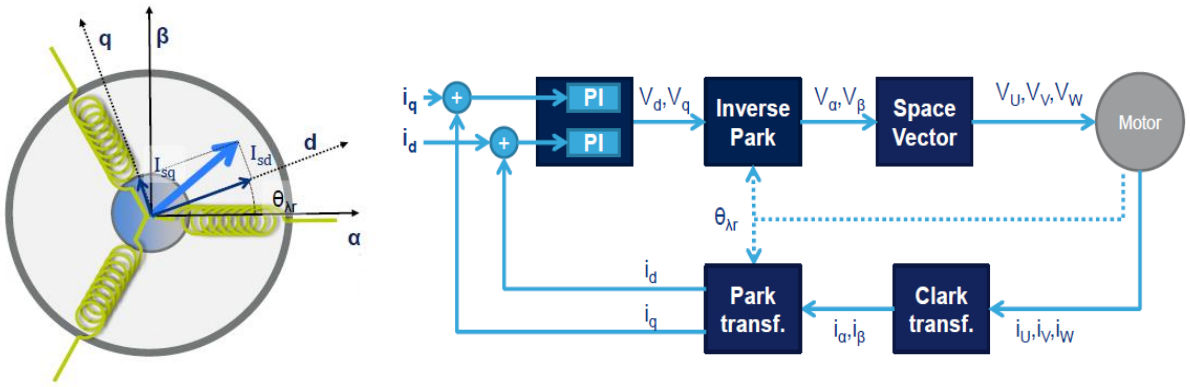


Figura 67: FOC: Sistemi cartesiani a differenti coordinate.

Nella prima si rappresenta un sistema a tre assi *a b c* sfasato di 120 gradi che è il sistema trifase fisso AC sullo statore, nella seconda invece l'equivalente α e β statorico fisso per un sistema a due fasi AC, mentre nella terza si rappresenta un sistema equivalente DC a due fasi *d* e *q* che ruotano insieme al flusso del rotore. Per passare da un sistema cartesiano all'altro si effettua dunque una trasformazione da coordinate *a b c* alle coordinate *d* e *q* e viceversa (una trasformata ad orientazione di campo) in modo da poter esprimere la forza magneto-motrice e i campi magnetici nel motore attraverso due componenti *d* e *q* ortogonali che ruotano insieme agli assi di flusso del rotore (avente in figura i poli S ed N). Questi due assi prendono il nome di asse diretto (*d*), quello allineato con il flusso del rotore, e asse di quadratura (*q*), quello ortogonale al diretto. Grazie a questo metodo, si possono esprimere le correnti transitanti nelle tre fasi dello statore in un sistema a due coordinate *d* e *q*. La componente diretta *d* dà informazioni sull'angolo di carico ϑ e risulta di massima efficienza per un valore nullo di tale angolo; la componente di quadratura esprime invece la coppia di riferimento per il motore.
Il controllo si basa sul concetto di trasformate, ed è evidenziato in Figura 68.



ϑ_r: *posizione angolare rotore*

Figura 68: Rappresentazione motore elettrico a coordinate diretta ***d*** *e di quadratura* ***q*** *(a sinistra) e relativo diagramma a blocchi per il controllo FOC in corrente del motore (a destra).*

Facendo riferimento alla Figura 68, il regolatore di corrente dà in output le tensioni di quadratura e diretta, e tramite le trasformate di Clark dirette (eq.9) e inverse (eq.10) e di Park dirette (eq.11) e inverse (eq.12) si riesce a gestire il controllo FOC del motore.

$$\begin{cases} i_\alpha = i_U \\ i_\beta = \frac{1}{3} i_U + \frac{1}{\sqrt{3}} i_V \end{cases} \qquad (9)$$

$$\begin{cases} i_a = \frac{2}{3} i_\alpha \\ i_b = -\frac{1}{3} i_\alpha + \frac{1}{\sqrt{3}} i_\beta \\ i_c = -\frac{1}{3} i_\alpha - \frac{1}{\sqrt{3}} i_\beta \end{cases} \qquad (10)$$

$$\begin{cases} i_d = i_\alpha \cos\theta_r - i_\beta \sin\theta_r \\ i_q = -i_\alpha \sin\theta_r + i_\beta \cos\theta_r \end{cases} \qquad (11)$$

$$\begin{cases} i_\alpha = i_d \cos\theta_r - i_q \sin\theta_r \\ i_\beta = i_d \sin\theta_r + i_q \cos\theta_r \end{cases} \qquad (12)$$

Il vantaggio consiste nell'avere un controllore in corrente analogo a quello di un motore DC, dove il flusso della componente diretta *d* rappresenta il flusso principale e i_q la corrente di armatura in un motore DC, per cui quest'ultima risulta proporzionale alla coppia di riferimento. In un motore AC a magneti permanenti, il flusso del rotore è fisso e non si può variare, per cui si può assumere $i_d = 0$.
In questo modo il controller dell'anello di corrente di un motore AC può essere progettato come se si disponesse di un motore DC, ed è proprio quello che fa il Driver in questione.
Una volta ottenute le tensioni sinusoidali per le tre fasi UVW ai capi del motore, si ottengono tre correnti su tre fasi sinusoidali in uscita sfasate di 120° (eq.11), tali da generare un campo magnetico sfasato di 90° elettrici rispetto al rotore e di intensità proporzionale alla coppia da generare.

$$\begin{cases} i_U = i\sin(\alpha + 90) \\ i_V = i\sin(\alpha + 90 + 120) \\ i_W = i\sin(\alpha + 90 + 240) \end{cases} \qquad (13)$$

Nella modellazione dell'EMA, discussa nel capitolo Modello Lineare, le trasformate dirette ed inverse si elidono tra di loro in modo da ottenere un modello di controllo per un motore AC analogo a quello tipico per un motore DC.

3.4.1.2 Pulse-Width Modulation PWM

A questo punto è necessario spiegare come vengono gestite le tre correnti sinusoidali (eq.13), che sono in uscita dalla logica di controllo del Driver, all'interno dell'elettronica del Driver in modo da comandare il motore. Nel circuito dell'inverter sono presenti degli interruttori elettronici chiamati *transistor*, i quali permettono di comandare 'analogicamente', attraverso una logica ON/OFF digitale, il motore alla velocità desiderata, diversa da quella fissa ottenibile dalla frequenza di rete (50Hz). Questo si ottiene attraverso una tecnica denominata PWM (*Pulse-Width Modulation*). Seguendo la Figura 69, l'inverter è costituito da due ponti-H e da sei transistor accoppiati; ciascuna coppia è comandata da un comparatore il quale comanda l'apertura dei due transistor con una logica ON/OFF, di cui uno dei due è preceduto da un blocco NOT in modo tale da ricevere un segnale complementare ON e OFF evitando che il circuito vada in corto circuito.
Il comparatore riceve due segnali: *modulante* (*m*) e *portante (p)*. Il primo consiste in una sinusoide, che sono proprio le tre correnti UVW in uscita dalla logica di controllo e che costituiscono la modulante per ognuna delle tre diverse coppie di transistor, mentre la seconda è un onda triangolare con una frequenza che è quella denominata *switching frequency* nel catalogo del Driver. Le due onde vengono comparate e laddove la modulante risulti maggiore della portante, il comparatore invia un segnale 1 (impulso positivo), viceversa invia un segnale 0 (impulso negativo).

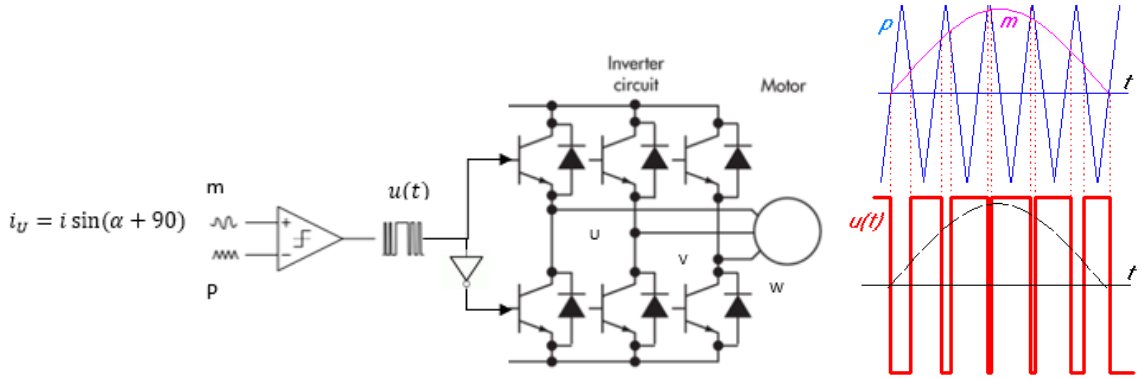


Figura 69: Schema di funzionamento di un inverter per un motore trifase (a sinistra), segnale modulante, portante per la tecnica PWM (a destra).

Tramite questa tecnica viene prodotto un segnale digitale ON/OFF provocato dall'apertura e chiusura dei transistors, di ampiezza definita dal *Duty Cycle*, che rappresenta un impulso di corrente che se sommato di volta in volta crea una sinusoide digitalizzata (Figura 70). Da notare come la frequenza di *switch* dei transistor, e quindi del *Duty Cycle,* coincida con la frequenza della portante e quindi con la *switching frequency*, mentre la frequenza del segnale sinusoidale in uscita digitalizzato coincida con quella della modulante elaborata dal controllo. L'ampiezza d'onda dipende dalla sommatoria dell'ampiezza dei *Duty Cycle*, definito come il rapporto tra il segnale ON e il periodo della portante.
Questo discorso viene ripetuto per le tre fasi, in modo da ottenere tre sinusoidi digitalizzate e sfasate di 120°.

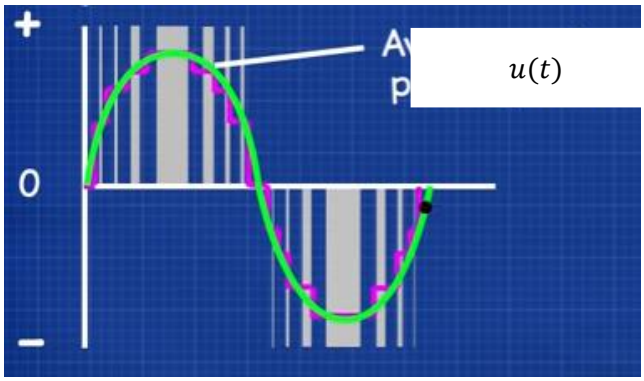


Figura 70: Creazione della sinusoide digitalizzata attraverso la tecnica PWM.

Questa tecnica è una delle più utilizzate in elettronica poiché permette di ottenere un comando variabile in corrente alternata AC mediante una logica di controllo gestita in corrente continua DC.

4. Software LabVIEW

Come riportato in Figura 2, la gestione dei segnali è effettuata dal c-Rio 9074 tramite l'ausilio dei moduli equipaggiati nel *chassis* e di una applicazione implementata in esso, programmata e gestita attraverso l'interfaccia utente del PC. In questo capitolo si introduce la programmazione in ambiente LabVIEW che si è utilizzata per la realizzazione dell'applicazione e successivamente si entra nel dettaglio dell'applicazione progettata.

4.1 Introduzione a LabVIEW

In questa sezione si descrivono le caratteristiche fondamentali dell'ambiente di sviluppo LabVIEW e si daranno alcune informazioni generali sul suo ambito di utilizzo e sulla sua interfaccia utente.
LabVIEW (*Laboratory Virtual Instrument Engineering Workbench*) è un ambiente di sviluppo per applicazioni principalmente orientate:

- All'acquisizione di dati e alla gestione di strumentazione elettronica;
- All'analisi ed elaborazione dei segnali.

LabVIEW fornisce un ambiente di programmazione di tipo grafico ad oggetti denominato "G language", il quale consente di realizzare programmi in forma di diagrammi a blocchi.
LabVIEW conserva comunque molte similitudini con gli ambienti di programmazione tradizionali: presenta tutti i tipi di dati e gli operatori predefiniti di uso comune, permette di generare nuovi tipi di dati combinando tra loro i tipi di dati elementari e di controllare l'esecuzione dei programmi ricorrendo a strutture di controllo di flusso, come ad esempio cicli e costrutti per l'esecuzione condizionale di codice. Mette inoltre a disposizione del programmatore una serie di librerie di funzioni che possono essere richiamate ed utilizzate all'interno dei programmi: le librerie comprendono funzioni di uso comune (funzioni aritmetiche e statistiche, funzioni per la manipolazione di stringhe, …) ed inoltre funzioni specializzate per l'acquisizione e l'elaborazione dei segnali, la trasmissione di dati mediante l'uso di porte seriali oppure mediante il protocollo di comunicazione TCP/IP. E' possibile inoltre per l'utente definire nuove funzioni ed arricchire le librerie in dotazione a LabVIEW.
Infine, il programma consente di effettuare il *debug* delle applicazioni create in linguaggio G attraverso opportune modalità di esecuzione dei programmi, come ad esempio il metodo *highlight execution* o *single step*.
LabVIEW presenta notevoli vantaggi rispetto ad un linguaggio di programmazione tradizionale:

- Permette di dare al codice una struttura modulare che consente di suddividere programmi complessi in sottoprogrammi più semplici che possono essere riutilizzati (subVI).
- Consente di raccogliere i VI in librerie, ovvero in un insieme di subVI utilizzabili da altri VI e velocemente inseribili nel codice sorgente dal programmatore.
- Fornisce un considerevole insieme di librerie per lo sviluppo di applicativi, tra le quali si trovano funzioni di tipo matematico e statistico, controllo di dispositivi per mezzo di alcuni tipi di interfaccia, comunicazione tra PC, etc...

4.1.1 I Virtual Instruments (VI)

L'ambiente di sviluppo consente di costruire programmi che prendono il nome di strumenti virtuali (*Virtual Instrument*, VI). Un *Virtual Instrument* permette l'interazione tra PC e strumentazione fornendo contemporaneamente all'utente un opportuno pannello frontale grafico per il dialogo con il VI stesso. In questo modo l'utente interagisce con un nuovo dispositivo (*Instrument*), costituito da PC, interfacce, strumenti e programma il quale presenta una realtà (*Virtual*) diversa dai singoli oggetti fisici che compongono il sistema stesso. Tale fatto spiega il nome di *Virtual Instrument* dato ad un programma LabVIEW.
L'utilizzatore può modificare il valore di alcune grandezze agendo su manopole o interruttori visualizzati dal programma e può osservare il risultato delle elaborazioni condotte internamente al VI su display grafici molto simili a quelli che si trovano sulla strumentazione numerica.
Un VI è composto da tre parti fondamentali:

- Pannello frontale (*Front Panel*);
- Diagramma a blocchi funzionale (*Block diagram*);
- Icona/connettore (*Icon/connector*).

Il *Front Panel* è la finestra che rappresenta l'interfaccia tra il programma e l'utilizzatore. Nel pannello frontale trovano posto tutti i controlli e gli indicatori dello strumento virtuale: per controllo si intende una variabile di ingresso che può essere modificata agendo sul pannello frontale, per indicatore si intende una variabile di uscita il cui valore può essere modificato dal programma e non dall'utente. Alcune delle tipologie di controlli ed indicatori messi a disposizione da LabVIEW sono mostrati in Figura 71.

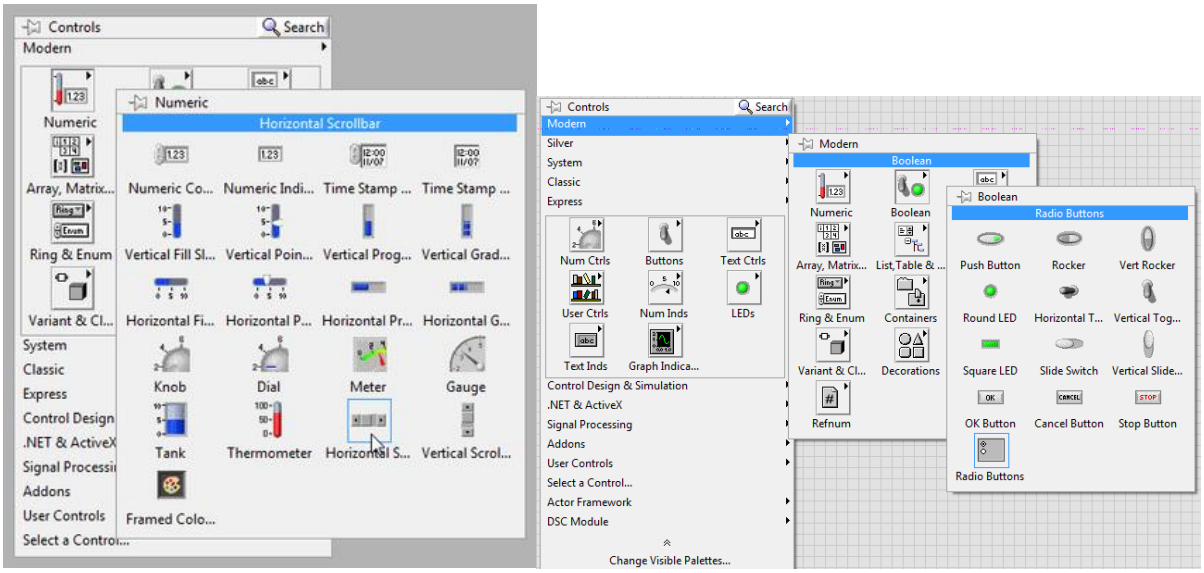


Figura 71: Esempi di controlli e indicatori del Front Panel.

Il *Block Diagram* contiene il codice vero e proprio, ed è costituito da:

- *Nodi*: sono degli elementi di elaborazione.
- *Collegamenti*: uniscono i nodi e permettono lo scambio di informazioni. Le informazioni passano da un nodo all'altro del pannello frontale per mezzo dei connettori che uniscono i nodi stessi (Figura 72).

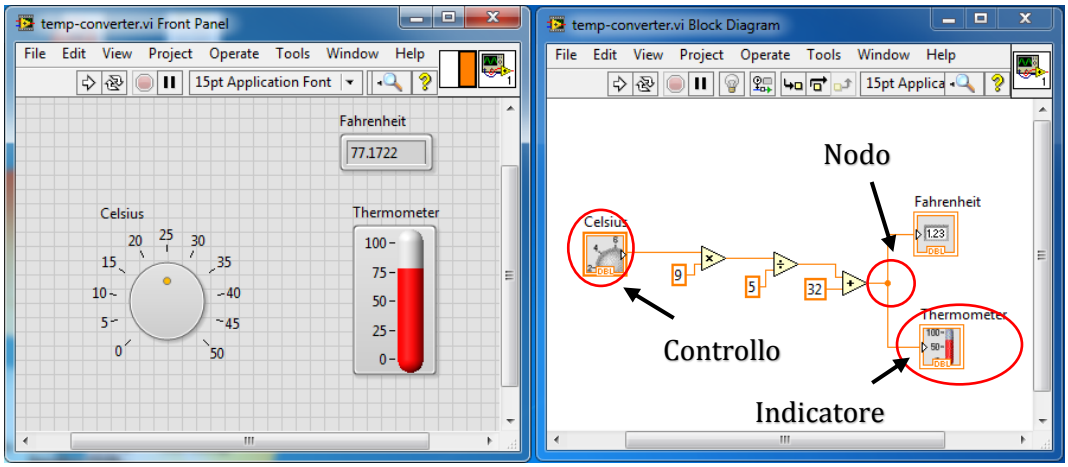


Figura 72: Esempio di un Front Panel (a sinistra) e di un Block Diagram (a destra).

- La coppia *Icon/connector* è il terzo elemento fondamentale di un programma LabVIEW (Figura 73). L'icona è un simbolo grafico di piccole dimensioni presente sul *Block Diagram* e che rappresenta simbolicamente il VI stesso e che permette di trasformare il programma in un oggetto (o SubVI). Il connettore stabilisce la corrispondenza tra aree dell'icona e controllori/indicatori del pannello frontale (il suo utilizzo risulta fondamentale per le SubVI).

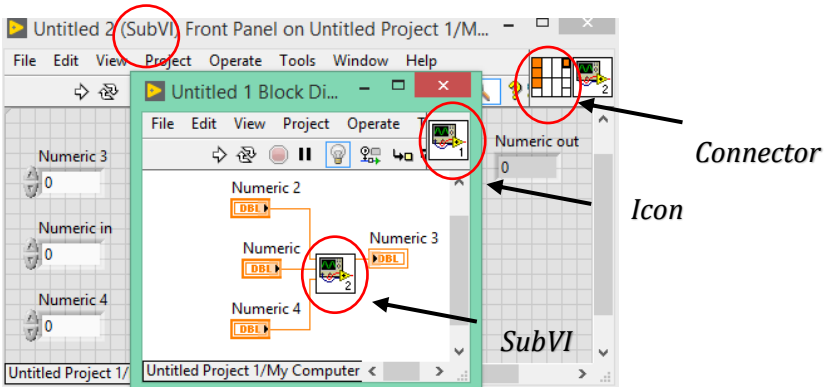


Figura 73: Esempio di Icon e Connector in una SubVI.

4.1.2 Tipologie di dato e strutture

LabVIEW dispone di un'ampia gamma di tipologie di dati, visualizzabili in Tabella 28. Per ogni dato corrisponde uno specifico colore sia per i controlli e gli indicatori, sia per i collegamenti che uniscono funzioni che operano con dati dello stesso genere.

Controllo	Nome	N. Bit	Controllo	Nome	N. Bit
SGL	Single-precision, floating-point	32	I32	Long signed integer	32
DBL	Double-precision, floating-point	64	I64	Quad signed integer	64
EXT	Extended-precision, floating-point	128	U8	Byte unsigned integer	8
FXP	Fixed-point	64 o 72	U16	Word unsigned integer	16
I8	Byte signed integer	8	U32	Long unsigned integer	32
I16	Word signed integer	16	U64	Quad unsigned integer	64

Tabella 28: Tipologie di dato.

Si può controllare il flusso di dati di un VI attraverso le strutture. LabVIEW mette a disposizione principalmente sei tipi di strutture: il *While Loop*, il *For Loop*, la *Case Structure*, la *Sequence Structure, la MathScript Structure* e la *Event Structure*.
Il *While Loop* ripete una parte di programma più volte. Il codice al suo interno viene eseguito fino a che il valore booleano passato al *Terminal Condition* (Stop) non cambia stato. L'*Iteration Terminal*, invece fornisce il numero di iterazioni eseguite dal loop partendo da zero. La struttura *While Loop* è illustrata in Figura 74.

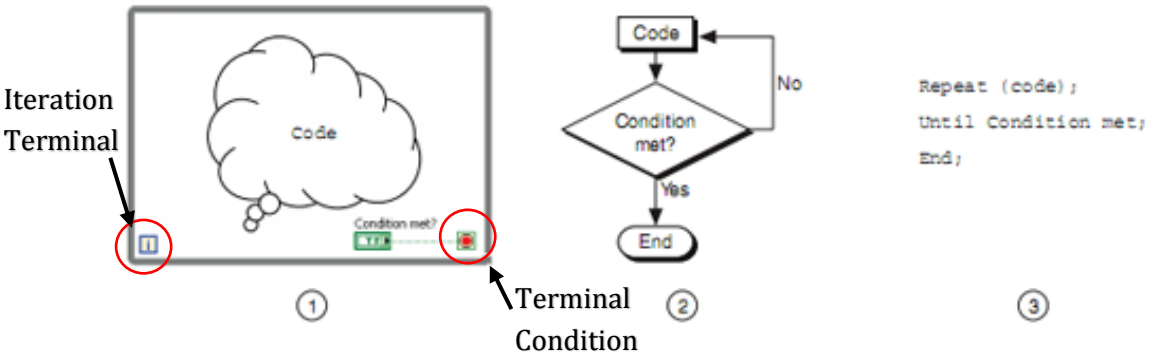


Figura 74: While Loop in ambiente Labview (1), in diagrammi a blocchi (2) e in codice C (3).

Dopo la creazione del *While Loop*, è possibile usare registri di trasferimento (*Shift Register*), mostrati in Figura 75, per passare valori da una iterazione alla successiva, oppure utilizzare un tunnel per trasferire dati all'esterno del ciclo dopo aver raggiunto la *Terminal Condition*.

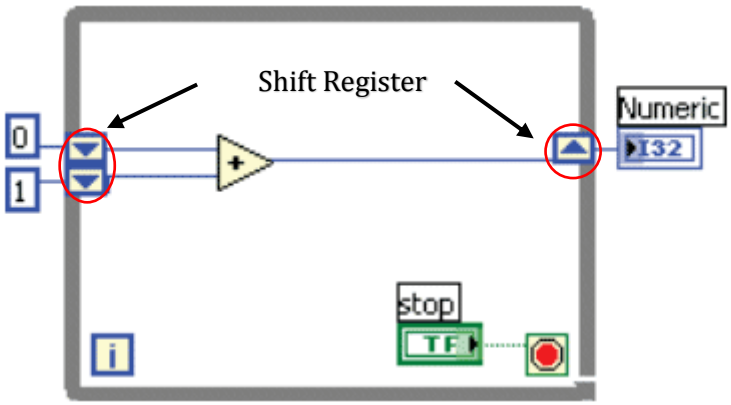


Figura 75: Shift Register in un While Loop.

Il For Loop esegue il codice al suo interno per un numero definito di volte scelto dal programmatore attraverso il *Count Terminal* (Figura 76). Come si può notare, il For Loop ha due terminali: il *Count Terminal* e *l'Iteration Terminal*. Il primo identifica il numero di volte che il ciclo verrà eseguito, mentre il secondo tiene traccia del numero delle iterazioni eseguite fino a quel momento. A differenza del *While Loop* il *For Loop* viene eseguito un numero predefinito di volte, mentre un *While Loop* termina quando il *Terminal Condition* cambia di stato.

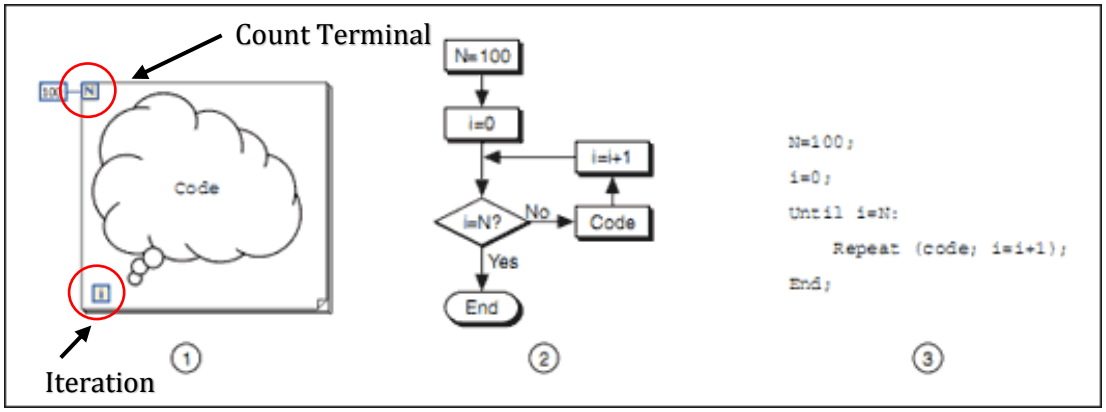


Figura 76: For Loop in ambiente Labview (1), in diagrammi a blocchi (2) e in codice C (3).

La struttura *Case Structure* è analoga alle istruzioni SWITCH CASE…, IF…THEN…ELSE… dei classici linguaggi di programmazione. Quindi esegue un codice, accedendo alla desiderata finestra di codice, piuttosto che un altro a seconda dello stato della variabile collegata al *Selector Terminal* mostrato in Figura 77.

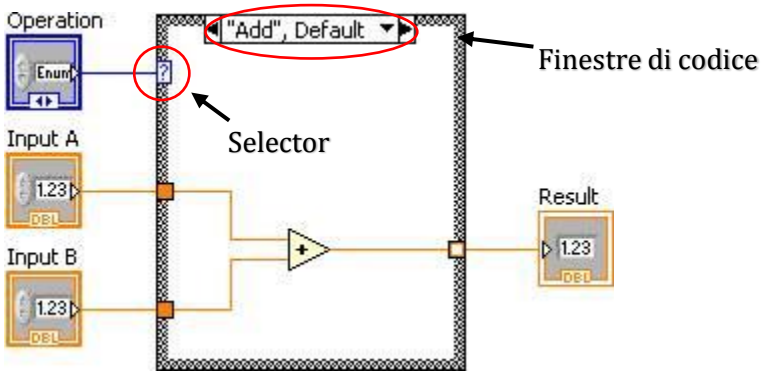


Figura 77: Esempio di una Case Structure.

La *Sequence Structure* (Figura 78) appare graficamente come una serie di riquadri sequenziali con un bordo simile a quello di una pellicola cinematografica ed è utilizzata per eseguire il codice in una sequenza ordinata. Nei linguaggi di programmazione convenzionali le righe di programma vengono eseguite nell'ordine in cui appaiono nel codice. Nel linguaggio grafico G di LabVIEW un nodo viene eseguito soltanto quando tutti i dati in ingresso sono disponibili ma a volte è necessario forzare e controllare l'ordine di esecuzione e questo si ottiene con la *Sequence Structure*. La struttura di default presenta un solo *frame*: si utilizza l'opzione "*Add frame after/before*" dal menù pop-up associato alla struttura stessa per aggiungere altri *frames*.

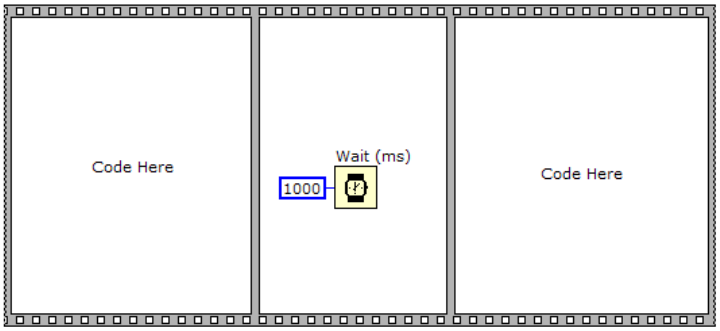


Figura 78: Esempio di Sequence Structure.

La struttura *MathScript* consente di implementare la matematica testuale di *MATLAB* all'interno dell'ambiente di sviluppo LabVIEW tramite un compilatore per *file ".m"* (Figura 79).

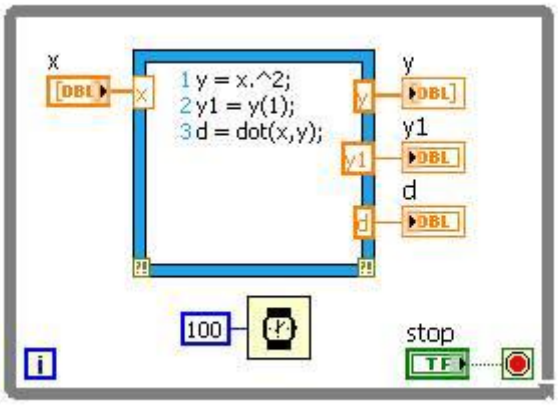


Figura 79: Esempio di una MathScript Structure all'interno di un While Loop.

Come ultimo, la struttura *Event Structure* è una struttura gestita ad eventi che, a differenza della *Case Structure*, esegue il codice desiderato solo quando avviene un certo evento, ovvero quando avviene un cambiamento di stato della variabile di processo (Figura 80). L'uso di questo strumento permette di generare algoritmi *event driven* come avviene, ad esempio, nel Visual Basic. In altre parole questa struttura non interroga continuamente il codice (*polling*) ma reagisce esclusivamente quando avviene un evento, ovvero un cambiamento dello stato della variabile di processo.

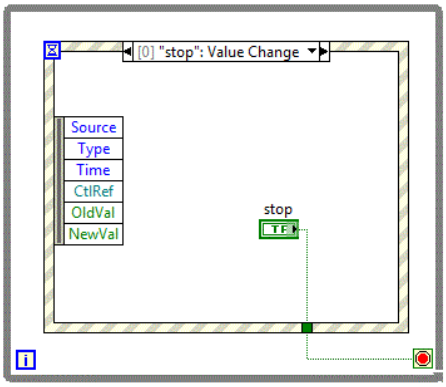


Figura 80: Esempio di Event Structure all'interno di un While Loop.

Risulta interessante analizzare la differenza tra una *Case Structure* ed una *Event Structure* mediante l'interrogazione di uno stesso codice come in Figura 81.

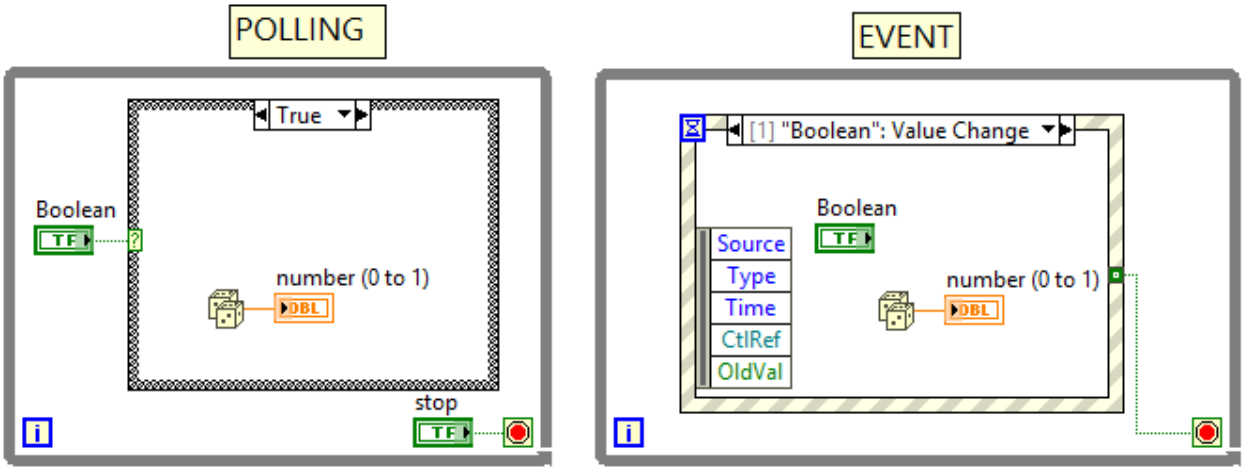


Figura 81: Case Structure VS Event Structure.

Utilizzando una *Case Structure* (POLLING), il codice è eseguito continuamente anche quando il tasto booleano non cambia il proprio stato, mentre utilizzando una *Event Structure* (EVENT) il codice rimane in attesa fino a quando lo stato logico del controllo booleano non cambia rendendo così il codice più leggero dal punto di vista computazionale.

4.2 Applicazione Motor Signal Management & Acquisition

L'applicazione *Motor Signal Management & Acquisition* (MSMA) è il software progettato in ambiente *LabVIEW* che permette gestire, mediante un'interfaccia utente, il comando del Driver e quindi del motore attraverso segnali analogici e digitali, come l'immissione del valore di riferimento di posizione dell'attuatore e il consenso dei segnali digitali, oppure la selezione del tipo di comando (sinusoidale o a rampa) e i parametri necessari a definirlo. Oltre alla gestione dei segnali per l'EMA, consente di visualizzare e di acquisire i segnali di *feedback* in uscita dal driver. L'architettura dell'applicazione è strutturata su tre livelli che presentano una comunicazione bidirezionale mediante diverse metodologie di trasmissione del segnale. I livelli e le comunicazioni sono riassunti in Figura 82.

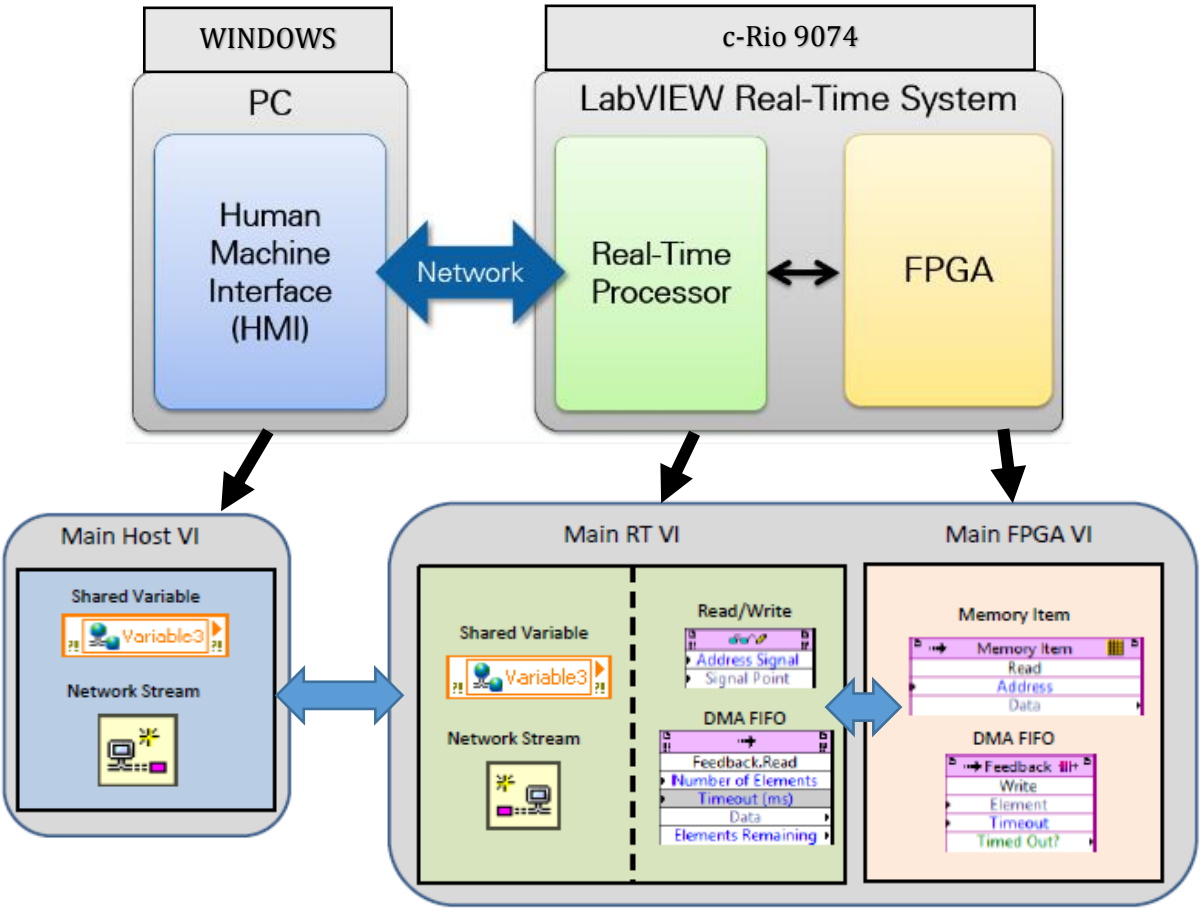


Figura 82: Architettura dell'applicazione e metodi di comunicazione.

Questi livelli sono rappresentati da:

- *Host PC*: è il livello in cui l'utente può interagire attivamente con il programma mediante un'interfaccia grafica che permette di avviare il programma, gestire il controllo, salvare i risultati e visualizzare i segnali provenienti dai trasduttori. A questo livello non vengono eseguite operazioni deterministiche, ovvero operazioni le cui tempistiche di esecuzione risultano critiche come ad esempio la generazione del segnale di riferimento e l'acquisizione del *feedback*. Il non-determinismo è implicato dal fatto che il PC presenta numerose operazioni in background atte a garantire il suo funzionamento, per cui potrebbe esserci la saturazione della CPU la quale causerebbe dei ritardi nel trattamento dei segnali, causati dall'impossibilità di stabilire con certezza quanto tempo sarà necessario per reperire l'informazione utile alla corretta esecuzione del codice. Di conseguenza tutte le operazioni che richiedono determinate tempistiche (determinismo) vengono affidate all'Hardware del c-RIO;
- *Real Time*: è gestito dal controllore Real Time a bordo del c-RIO ed ha il compito di supervisionare la tempistica e la sequenza nello scambio dei dati dal PC all'FPGA e viceversa;
- *FPGA*: rappresenta la parte dell'applicazione dove i segnali vengono acquisiti e inviati tra moduli NI e morsettiera del Driver Lenze.

In ognuno dei seguenti livelli è presente una *VI,* la quale, programmata dall'utente in ambiente *LabVIEW,* consente di adempire alle operazioni sopra citate.
Nei successivi capitoli, verrà dapprima introdotta la *UI*, in seguito si passerà alla descrizione dell'architettura e delle scelte progettuali.

4.2.1 Interfaccia utente

L'interfaccia utente dell'applicazione (mostrata in Figura 83) permette all'operatore di interagire facilmente con il PC, consentendo la scelta del tipo di prova e dei *feedback*, l'immissione dei parametri, il monitoraggio delle grandezze d'interesse, l'abilitazione del driver, il metodo di salvataggio dei dati e il controllo e la connessione dell'intera architettura.
Il quadro comandi è collocato a sinistra dell'interfaccia ed è stato suddiviso in diverse finestre raggruppando i comandi tra loro attinenti, facilitando così il lavoro dell'utente.
Tramite un *Tab Control* posizionato in alto, si possono inoltre selezionare due diverse finestre, ognuna delle quali consente di focalizzare il monitoraggio su grandezze differenti. L'interfaccia adottata è il risultato di una serie di scelte stilistiche con il fine di conferire al programma un aspetto quanto più possibile *user-friendly.*

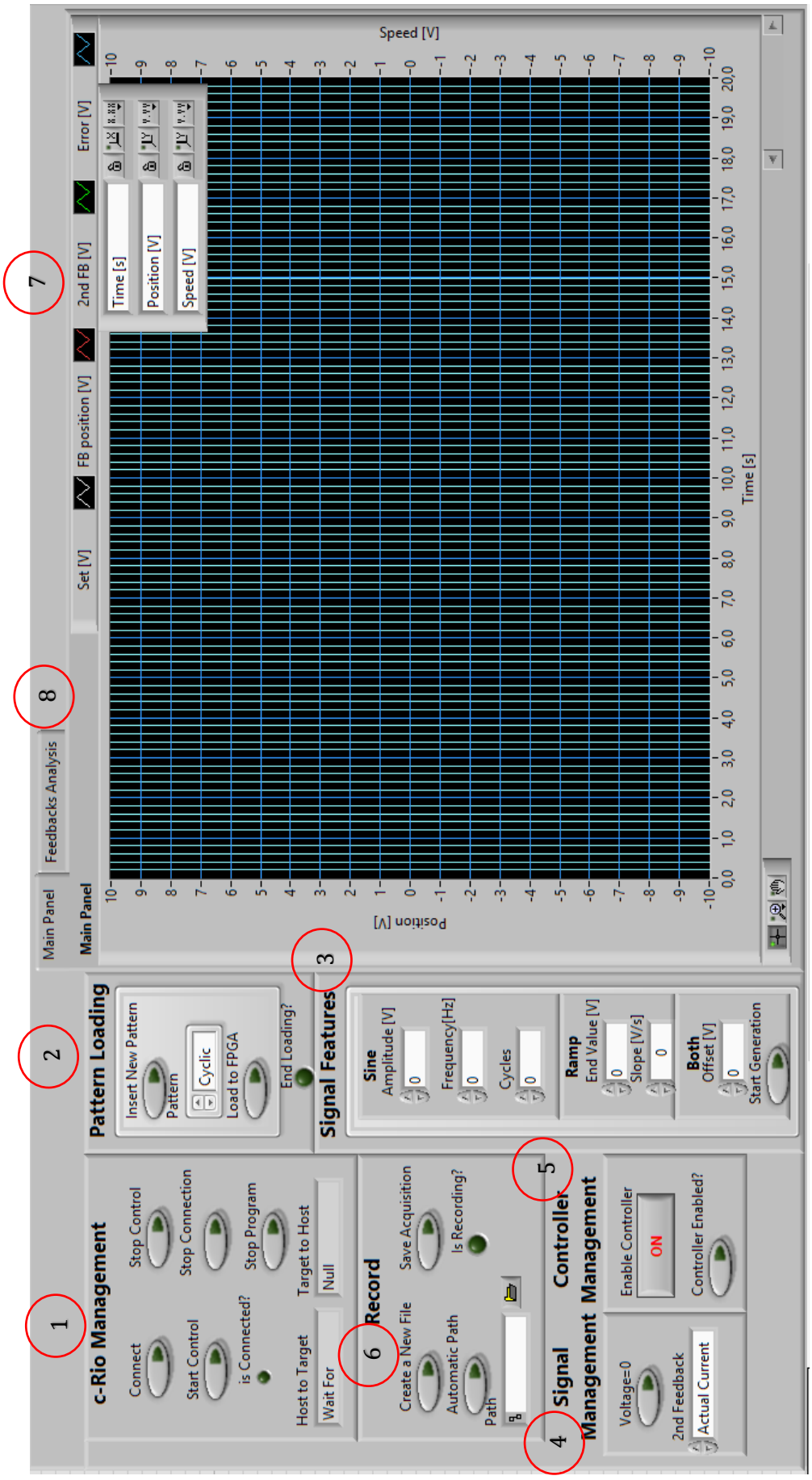


Figura 83: Interfaccia Utente dell'applicazione MSMA.

Nella seguente Tabella 29 sono descritti sinteticamente gli elementi dell'interfaccia utente indicati in Figura 83. Nei paragrafi seguenti verranno descritti in dettaglio i principali elementi elencati.

Numero	Descrizione
1	Stabilisce la connessione tra PC e c-Rio e avviare la gestione dei segnali.
2	Seleziona il tipo di segnale SET e di caricarlo sulla memoria FPGA.
3	Si immettono i parametri e si avvia la generazione del segnale.
4	Permette di ottenere un segnale nullo in modo rapido e di selezionare il secondo feedback desiderato.
5	Consente di attivare l'applicazione di controllo del driver (terminale X5, pin DI2: *Enable Application*).
6	Sono raggruppati tutti i controlli che permettono di salvare su un file ".txt" i dati relativi alla prova svolta.
7	In quest'area è possibile gestire il plot dei segnali, nascondere o rendere visibile un segnale in particolare, modificare lo stile del plot oppure modificare parametri relativi alle scale.
8	Il *Tab Control* permette di accedere a discrezione dell'utente alla finestra *Main Panel*, in cui sono mostrati gli andamenti delle acquisizioni, e alla finestra *Feedback Analysis*, in cui viene analizzato lo spettro di frequenza dei segnali acquisiti.

Tabella 29: Descrizione interfaccia utente.

4.2.1.1 c-Rio Management

Il sottogruppo c-RIO Management raggruppa gli strumenti necessari alla gestione della connessione tra il PC e il c-RIO 9074. Per creare la via di comunicazione dati tra target (PC) ed Host (c-Rio) è necessario stabilire una connessione mediante *Network Streams*: per attivare la sezione di codice che permette tale collegamento l'utente deve premere sul pulsante *Connect*. Fatto ciò, il LED verde si accende ad indicare che la via di comunicazione è stata stabilita.
Il pulsante *Start Control* permette di avviare la comunicazione nei due versi necessaria per la gestione dei segnali dell'EMA. Dal momento in cui viene premuto, il c-RIO riceve il consenso dall'utente per eseguire il codice presente sul RT prima e sull'FPGA dopo e resta in attesa dei successivi comandi di generazione del *pattern*. Di default, il segnale di SET viene portato a zero subito dopo aver premuto il pulsante.
Il pulsante *Stop Control* avvia la procedura d'arresto del codice presente su RT e su FPGA, riportando il SET a zero, leggendo i dati rimasti in coda e riportando la comunicazione allo stato *Connect*.
Il pulsante *Disconnect* consente di mettere il c-RIO in uno stato di stand-by. Consiste nel distruggere la comunicazione creata in precedenza mediante *Network Stream* e di riportare quindi

la connessione allo stato precedente a quando è stato premuto Connect. Il codice del Real-time si porta in uno stato di attesa fino a quando non viene ripremuto il pulsante *Connect*. Dopo aver premuto il pulsante, il LED verde si spegne.
Il pulsante Stop Program permette la chiusura dell'applicazione MSMA.
Nell'indicatore *Host To Target* vengono mostrati, sotto forma di stringa di testo, i comandi che il PC (Host) invia al c-RIO (Target) per sincronizzare il processo di connessione. Viceversa, l'indicatore *Target To Host* mostra i comandi che il c-RIO invia al PC. Questo permette all'utente di capire a che punto di sezione della programmazione si è giunti attraverso i comandi effettuati.

4.2.1.2 Pattern Loading

Questo sottogruppo permette di gestire il *pattern* che si intende utilizzare per il comando, ovvero la forma di comando desiderata. Sono stati utilizzati due *pattern*: quello sinusoidale e quello a rampa.
Oltre a tale scelta, il sottogruppo gestisce l'invio di tale segnale: attraverso il pulsante *Insert New Pattern* si rende disponibile al *target* il tipo di segnale selezionabile attraverso la finestra *Pattern* (sinusoidale o a rampa) in modo da inserire i parametri di tale segnale (vedi sottogruppo Signal Features); attraverso il pulsante *Load to FPGA* si dà il consenso al segnale in attesa su RT di essere inviato al FPGA.
La sequenza si generazione pattern è: *Insert New Pattern, Pattern,* definizione dei parametri su sottogruppo *Signal Features*, *Load to FPGA.*

4.2.1.3 Signal features

In questo sottogruppo sono definiti dall'utente i parametri caratteristici del *pattern* scelto in *Pattern Loading.* A seconda che sia selezionato un pattern sinusoidale o a rampa, il codice automaticamente rende disponibile all'utente solo i parametri correlati a tale segnale. A questo punto l'utente seleziona il valore da immettere, osservando le unità di misura riportate su ciascuna finestra.
Una volta premuto il pulsante *Load to FPGA,* il segnale è pronto per essere trasmesso in uscita al modulo AO, trasmissione che avviene quando si preme il pulsante *Start Generation*.

4.2.1.4 Signal Management

In questo riquadro è possibile selezionare il secondo *feedback* da acquisire scegliendo tra la corrente attuale e la velocità attuale mediante la finestra *2nd FB.* Inoltre, tramite il pulsante *Voltage=0,* consente di mandare un set di 0V al motore in modo da riottenere la posizione di riferimento zero in caso di emergenze.

4.2.1.5 Controller Management

Il sottogruppo permette di abilitare l'applicazione caricata all'interno del driver (segnale *Enable Application*) mediante un segnale digitale in uscita dal modulo del c-Rio; quando il LED verde è acceso l'applicazione è abilitata.

4.2.1.6 Data Logging

Nel riquadro *Data Logging* sono presenti i controlli che consentono il salvataggio dei dati sperimentali. Innanzitutto, prima dell'inizio della prova, si determina la selezione del *Folder*, il quale può essere scelto automaticamente premendo il pulsante *Automatic Path* e selezionando il percorso del *Folder* nel riquadro *Path*, oppure si preme il pulsante *Create New File* in modo da selezionare manualmente il percorso ad ogni acquisizione. La scrittura dei dati sul nuovo file inizierà soltanto dopo aver premuto il pulsante *Save Acquisition*. Quando il programma sta registrando, il LED verde è acceso. Per interrompere il salvataggio basterà premere di nuovo il pulsante *Save Acquisition*.

4.2.1.7 Tab Control

La finestra principale *Main Panel* consente di visualizzare, dopo che è stato premuto il pulsante *Start Generation*, i segnali coinvolti nella gestione del motore e quindi il *SET* di posizione, il *FEEDBACK* di posizione, il *2nd FEEDBACK* di velocità o di corrente, *l'ERRORE* tra set e feedback di posizione. Ciascun segnale può essere reso visibile o meno cliccando sull'icona del segnale nella legenda ed accedendo ad un menù che consente di gestire diverse opzioni di plottaggio (ad esempio, lo spessore ed il colore delle linee).
La seconda finestra *Feedback Analysis* consente di visualizzare lo spettro di frequenza dei segnali acquisiti.

4.2.2 Codice PC

LabVIEW offre una vasta gamma di metodi per il trasferimento dati tra c-RIO e un PC. Il ruolo del VI Host (Figura 84) è quello di creare una comunicazione interattiva tra utente e applicazione tramite un interfaccia grafica su cui è possibile impostare parametri, visualizzare le acquisizioni e salvarle.

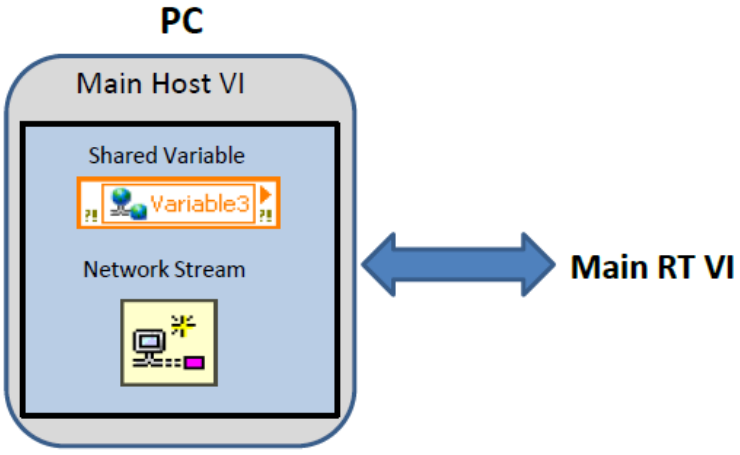


Figura 84: Livello PC.

4.2.2.1 Strumenti di comunicazione tra PC e RT

4.2.2.1.1 Le *Shared Variable*

Le *Shared Variable* sono particolarmente adatte quando non è necessario uno streaming di dati (es. dati provenienti dai trasduttori), ma occorre trasportare dati che non variano con una frequenza elevata, come:

- Parametri del segnale di set come ampiezza, frequenza e offset nel caso di un set sinusoidale e ampiezza e pendenza nel caso di un segnale a rampa;
- Controlli booleani di vario genere come l'avvio, lo stop, l'emergenza etc.. ;
- Stringhe di comunicazione tra Host e Target.

In Figura 85 è presente un esempio di *Shared Variable*. In questo caso il dato viene scritto sul c-RIO (RT) e letto dal Host PC, ma può verificarsi anche il caso opposto.

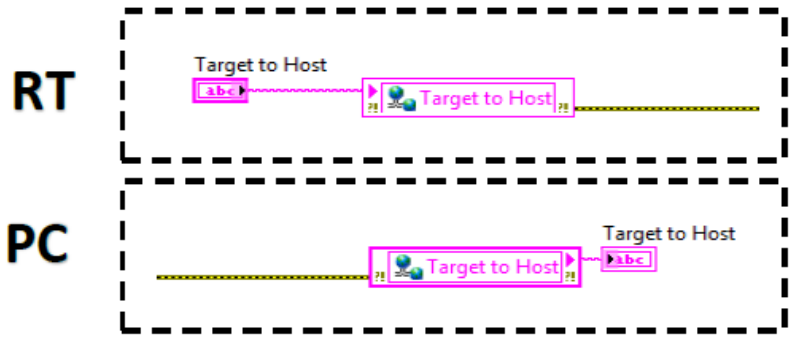


Figura 85: Esempio di comunicazione PC-RT tramite Shared Variables.

Le *Shared Variable* possono essere configurate in base alle esigenze dell'utente tramite l'apposita finestra di configurazione delle proprietà mostrata in Figura 86.

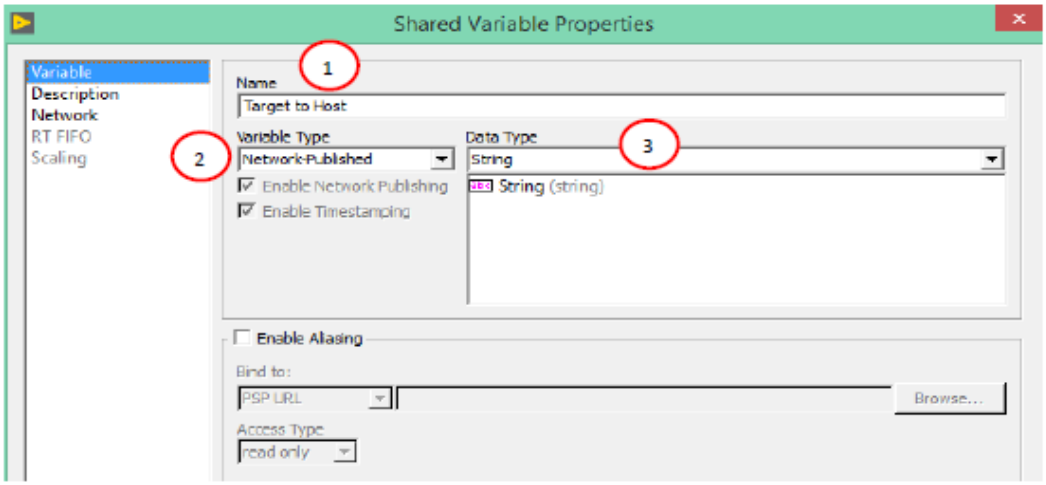


Figura 86: Finestra di configurazione di una Shared Variable.

Tramite questa finestra di dialogo è possibile impostare:
1) Nome della variabile;

2) Tipologia della variabile: LabVIEW mette a disposizione principalmente due tipologie differenti di variabile:

- *Single Process*: sono utilizzate per trasferire dati fra due punti del diagramma di uno stesso VI che non possono essere connessi con un filo, come ad esempio accade in due cicli paralleli.
- *Network Published*: utilizzano il NI *Publish-Subscribe Protocol* (NI-PSP) e sono state utilizzate per la condivisione dati tra c-RIO e PC tramite cavo Ethernet.

3) Tipologia del dato:

- Dati tipo stringa: una stringa è una sequenza di caratteri ASCII. Le stringhe forniscono un formato per informazioni e dati indipendente dalla piattaforma, e sono utilizzate per controllore gli strumenti inviando comandi di testo e restituendo i dati nella forma o di stringhe ASCII o di stringhe binarie, che si potranno poi convertire in valori numerici. All'interno del VI questo tipo di dato è individuato dal colore rosa.
- Dati tipo numerico: LabVIEW rappresenta i dati numerici come numeri in virgola mobile, in virgola fissa, interi con segno, interi senza segno e numeri complessi. La precisione *Double* e *Single* e i dati numerici *Complex* sono rappresentati con il colore arancione. Tutti i dati numerici interi sono individuati dal colore blu.
- Dati di tipo booleano: LabVIEW memorizza i dati booleani come valori a 8 bit. Un booleano può essere utilizzato in LabVIEW per rappresentare 0 o 1, oppure TRUE o FALSE. I dati di tipo booleano sono indicati in colore verde.

4.2.2.1.2 Le *Network Stream*

Le *Network Stream* sono state progettate per la comunicazione dati ad alta produttività, come ad esempio i dati acquisiti, e utilizzano un modello di comunicazione a senso unico per trasmettere dati tra piattaforme diverse.
Il primo step è creare un *Endpoint*, ovvero una connessione tra il Main Host VI e il Main RT VI; dato che la comunicazione è unilaterale, è necessario creare due tipologie di *Endpoint* (*Writer Endpoint* e *Reader Endpoint*). Di conseguenza, se si desidera trasmettere dati dal Real Time al PC sarà necessario creare un *Writer Endpoint* sul Real time e un *Reader Endpoint* sul PC, proprio come nella applicazione MSMA.

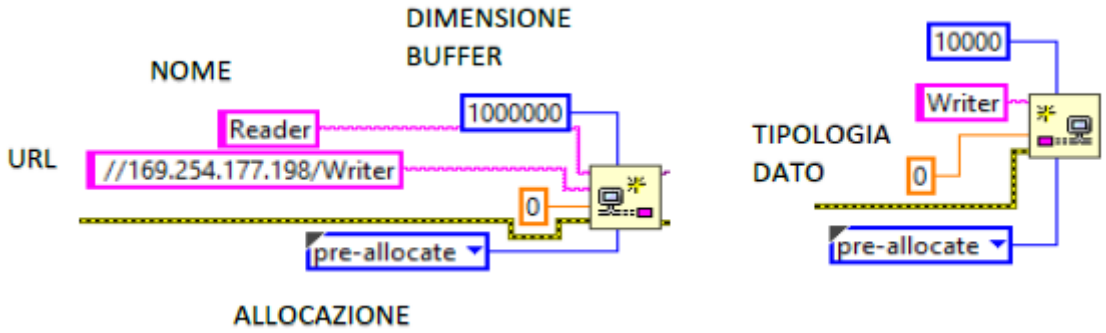


Figura 87: Reader Endpoint e Writer Endpoint.

E' necessario distinguere due tipologie di *endpoint*: *l'endpoint* attivo e quello passivo. L'*endpoint* attivo, su cui è specificato l'URL, è quello che origina la prima connessione, mentre *l'endpoint* passivo, quello senza l'URL, aspetta la richiesta di connessione dell'*endpoint* attivo. Nella terminologia *client/server*, l'*endpoint* attivo è sinonimo di *client* mentre quello passivo è sinonimo di *server*. Nel progetto realizzato, il lettore presente nel VI del PC è l'*endpoint* attivo, mentre lo scrittore all'interno del VI del Real Time è l'*endpoint* passivo. In altre parole, quando l'utente vuole stabilire la connessione il PC invia la richiesta al c-RIO.

Una volta stabilita la connessione, si passa alla lettura e scrittura dei dati con le funzioni mostrate in Figura 88.

Figura 88: Lettura e scrittura dei dati mediante Network Stream.

Infine per eliminare la connessione stabilita si utilizza la funzione *Destroy Stream Endpoint*; tuttavia, se si vuole avere la certezza che tutti i dati scritti siano stati letti prima della chiusura, si può inserire la funzione *Flush Stream* (Figura 89).

Figura 89: Flush Stream e Destroy Endpoint

Si riporta in Figura 90, a titolo di esempio, il processo di scrittura e lettura mediante *Network Stream.*

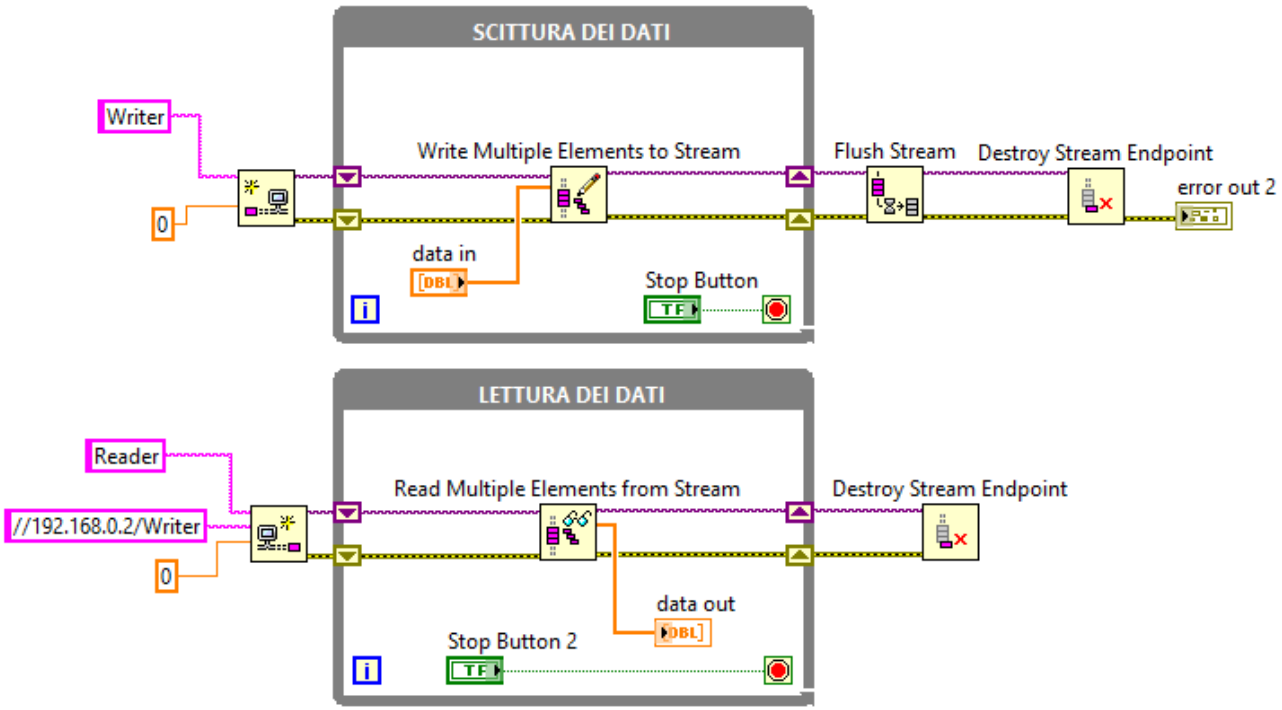


Figura 90: Esempio lettura e scrittura dati mediante Network Stream.

Nell'applicazione MSMA si avrà una *Network Stream* per il flusso dati dal c-Rio al PC. In particolare è stata usata per il trasporto di sei segnali:
1) Segnale di Set di posizione;
2) Segnale di Feedback di posizione;
3) segnale di Feedback di corrente o velocità;
4) Errore di posizione.

Il trasporto utilizza un buffer FIFO (acronimo di *first in-first out*) con una capacità di 1'000'000 elementi. Il processo di bufferizzazione multi-elemento è schematizzato dall'esempio in Figura 91.

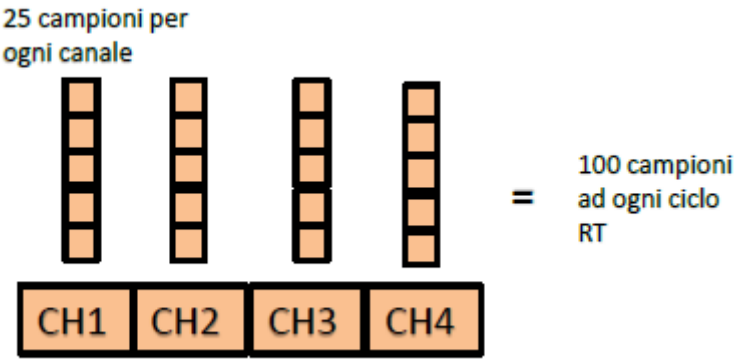


Figura 91: Schema bufferizzazione multi-elemento.

Considerando che questa serie di dati è acquisita dall'FPGA con una frequenza di 2500 Hz e che i dati vengono scritti sull'Endpoint Writer presente sul Real Time con una frequenza pari a 100 Hz, ad ogni ciclo del RT vengono scritti 25 elementi per ogni canale. Dopo la scrittura, i dati vengono mandati al lettore sul PC, che trasforma il flusso di dati da un array 2D a quattro array 1D, per poter essere manipolati singolarmente. I dati provenienti dal RT vengono spacchettati e letti dall'Endpoint Reader sul Main Host VI con una frequenza di 5 Hz.
E' molto importante impostare correttamente il tempo di processamento dei dati su tutti e tre i livelli (PC, Real Time e FPGA) e la dimensione dei buffer per evitare l'insorgere *dell'overflow*, che avviene quando lo scrittore processa più dati di quanto il lettore è fisicamente in grado di leggere, con conseguente perdita di dati.

4.2.2.2 Main Host VI

In Figura 92 si riporta il codice implementato sul VI Host. È costituito da 5 *While Loop* in parallelo, ciascuno con una funzione specifica. Il parallelismo è fondamentale durante la definizione dell'architettura del VI in quanto ogni operazione impiega un tempo diverso per essere eseguita. Inserendo cicli in parallelo ogni porzione di codice viene eseguita in maniera indipendente dalle altre, arginando così rallentamenti ed eventuali colli di bottiglia.

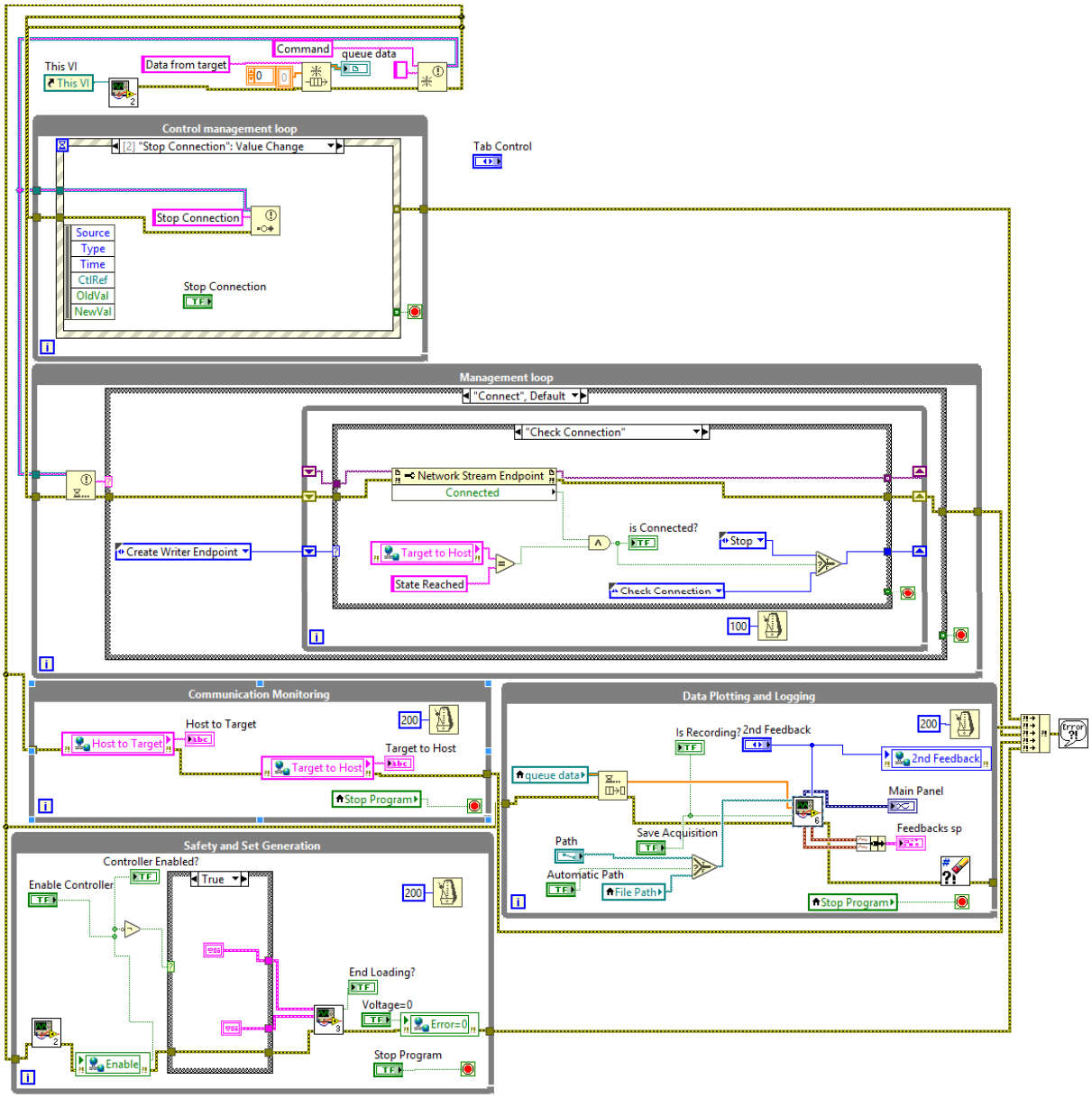


Figura 92: Block Diagram del Main Host VI.

4.2.2.2.1 Comunicazione con il c-Rio

La comunicazione e il trasporto dati tra RT e PC è gestita dalla porzione di codice mostrata in Figura 93.

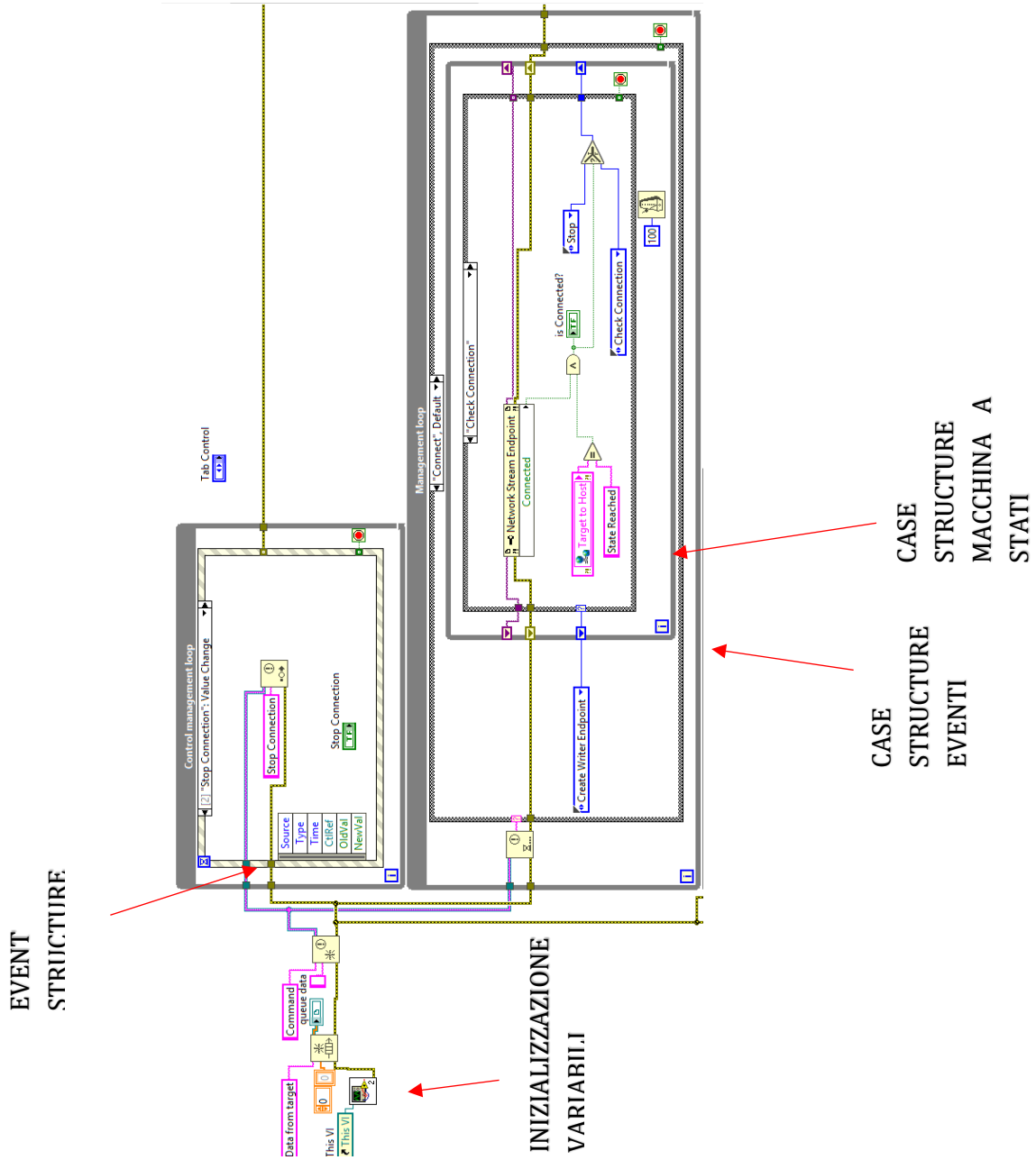


Figura 93: Inizializzazione variabili, Control Management loop e Management Loop.

Prima di tutto, le variabili vengono inizializzate ponendo a zero quelle di tipo numerico e a falso quelle di tipo booleano. In questo modo, tra un avvio e l'altro del programma, si evita di scrivere nella variabile valori rimasti dall'avvio precedente.
Successivamente, si entra nella *Event Structure* all'interno del *Control Management Loop*. Si tratta di una struttura ad eventi che definisce particolari azioni da svolgere in base ai tasti premuti sull'interfaccia utente. Quando viene attivato un particolare evento nella *Event Structure*, l'azione da svolgere viene notificata alla *Case Structure Eventi*, che seleziona la macchina a stati appropriata all'interno del *Management Loop.* Di conseguenza, in base all'operazione da svolgere, possono essere attivate quattro differenti macchine a stati: *Create Writer Endpoint*, *Create Reader Endpoint*, *Check Connection* e *Stop,* le quali permettono di ottenere la connessione di PC e c-Rio mediante *Network Stream*.
Inoltre nel Main Host VI è presente il *Comunication Monitoring loop* (Figura 94), che consente all'utente di visualizzare, tramite una finestra di dialogo, le fasi delle macchine a stati. Il

monitoraggio è molto importante poiché permette all'utente di osservare lo stato raggiunto dal ciclo (utile in caso di debug), individuando quindi facilmente la fase in cui il ciclo si è arrestato.

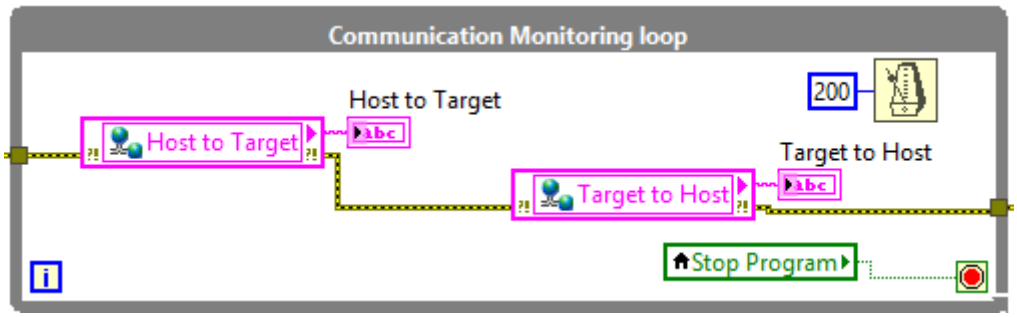


Figura 94: Communication Monitoring loop.

4.2.2.2.2 Data Plotting & Logging

Nel *Data Plotting & Logging loop* le acquisizioni vengono plottate e salvate. Nel plot, visualizzabile sull'interfaccia utente, i dati vengono rappresentati con le rispettive unità di misura. I dati sono inoltre salvati in formato ".txt", sono espressi in Volt e hanno come separatore decimale il punto, in modo da poter essere facilmente manipolati all'interno del software Matlab. I loop e i SubVI annidati contenuti al suo interno sono visibili in Figura 95.

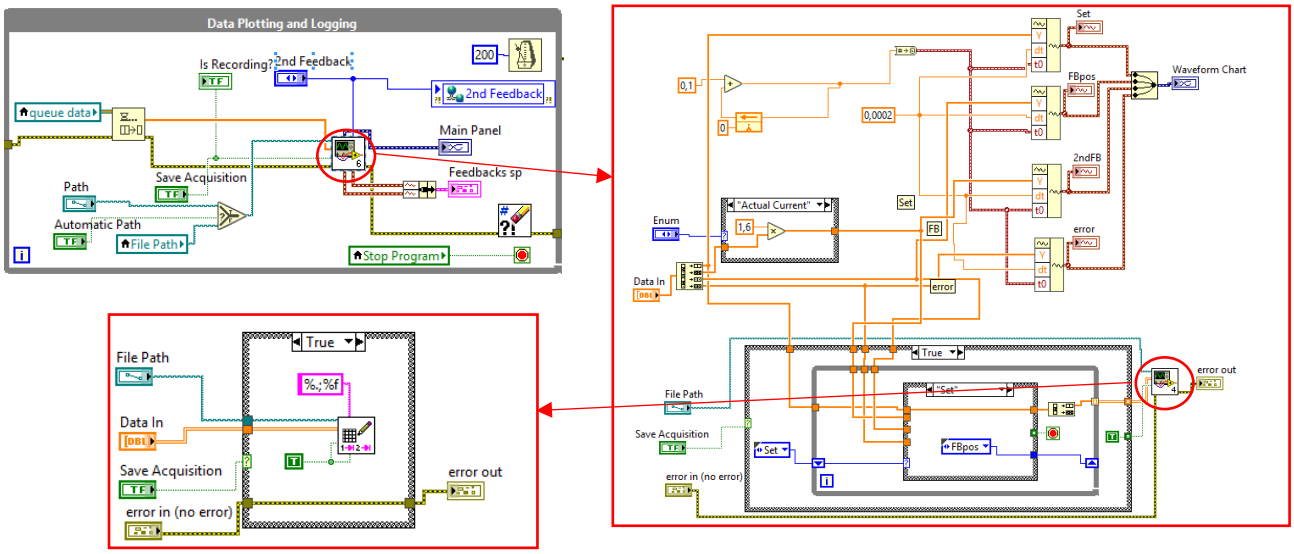


Figura 95: Data Plotting and Logging loop e SubVI annidati.

Figura 96: la prima colonna è il Set di posizione (1), la seconda è il Feedback di posizione(2), la terza è il Feedback di velocità o corrente (3) e la quarta è l'errore di posizione (4).

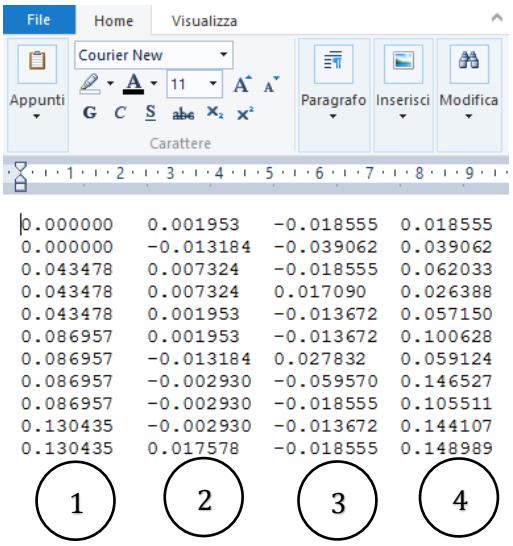


Figura 96: File '.txt' e riferimenti colonne.

4.2.2.2.3 Safety & Set Generation

All'interno del loop *Safety & Generation loop* (Figura 97) vengono gestiti i parametri del segnale di set da generare e i seguenti strumenti immessi dall'utente:

- Insert New Pattern: caricamento di un nuovo segnale di SET.
- Pattern: set sinusoidale o a rampa.
- Load to FPGA: caricare il Set desiderato a bordo dell'FPGA.
- End Loading: indicatore che informa l'utente che il pattern è stato caricato correttamente ed è pronto per la generazione.
- Start Generation: permette la generazione del segnale di Set;
- 2nd Feedback: la scelta di avere come secondo feedback la velocità o la corrente.

Questi parametri e strumenti sono trasmessi al c-Rio attraverso *Shared Variable.*

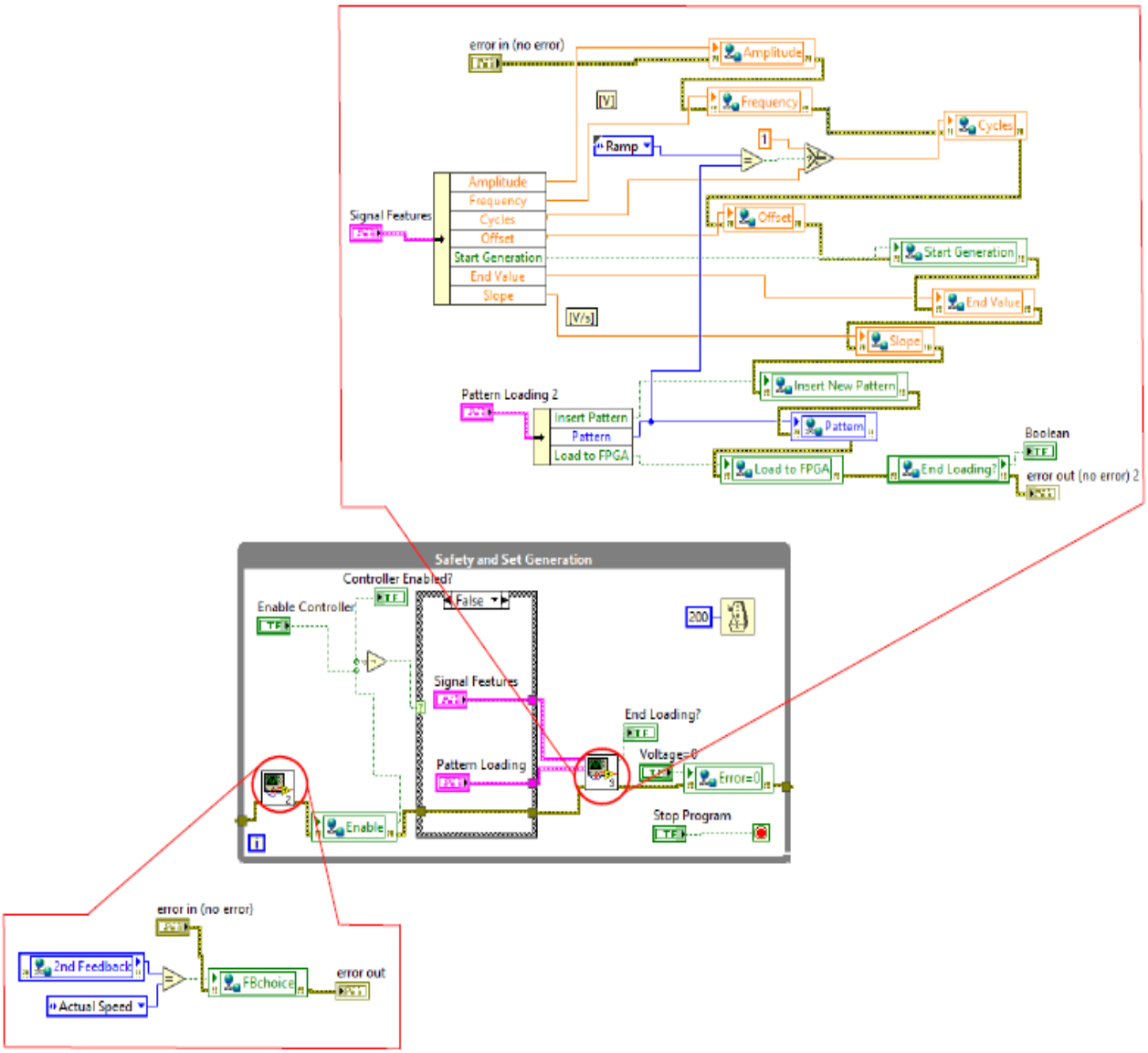


Figura 97: Safety & Set Generation loop e SubVI.

4.2.3 Codice RT

Il livello Real Time è disposto tra il PC e l'FPGA e nell'applicazione svolge il ruolo di intermediario, per cui fornisce all'FPGA i parametri impostati dall'utente sul PC e rende disponibili a quest'ultimo i dati provenienti dai trasduttori. Per questo in tale VI sono presenti due interfacce (Figura 98): una per la comunicazione con il VI presente nel PC Host, tramite i metodi visti in precedenza, e l'altra per la comunicazione con il VI caricato nell'FPGA.

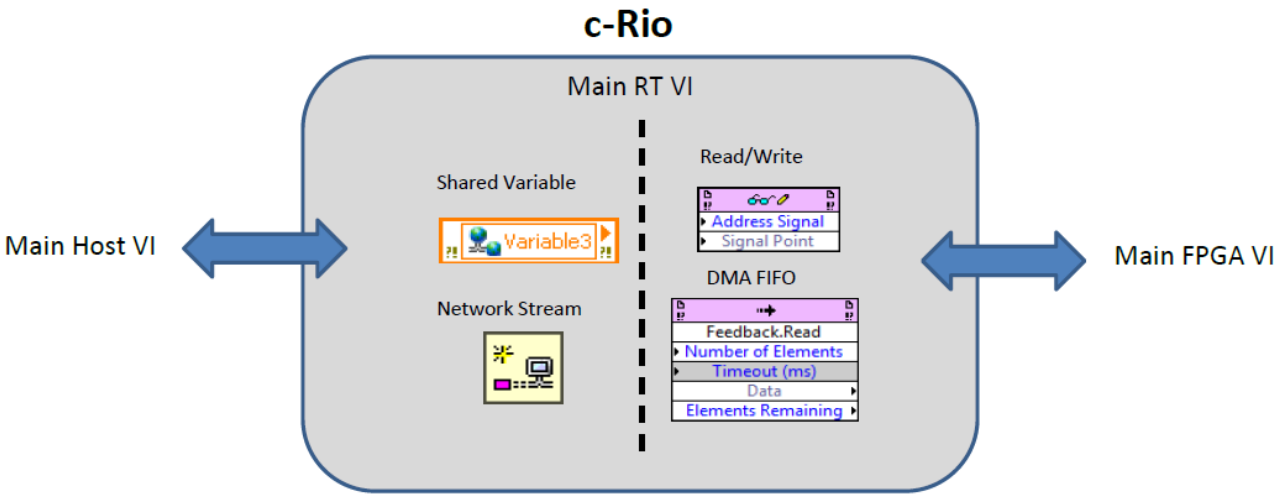


Figura 98: Livello Real Time.

4.2.3.1 Strumenti di comunicazione tra PC e FPGA

Anche in questo caso i metodi di comunicazione adottati dipendono dalla natura stessa del dato. Per quanto riguarda la comunicazione RT verso PC, i metodi sono quelli discussi in precedenza, ovvero tramite *Shared Variables* nel caso di dati che non variano con frequenza elevata e *Network Stream* per dati che variano con frequenza elevata. Facendo riferimento all'interfaccia RT verso FPGA, per le attività caratterizzate da vincoli temporali rigidi, come lo streaming dei dati provenienti dai moduli I\O, si utilizzano le *DMA FIFO* (funzione *Invoke Method*), mentre per la lettura dei controlli si utilizzano le funzioni *Read/Write Control*. In Figura 99 sono mostrate le funzioni per l'interfacciamento e lo scambio dati con il livello FPGA

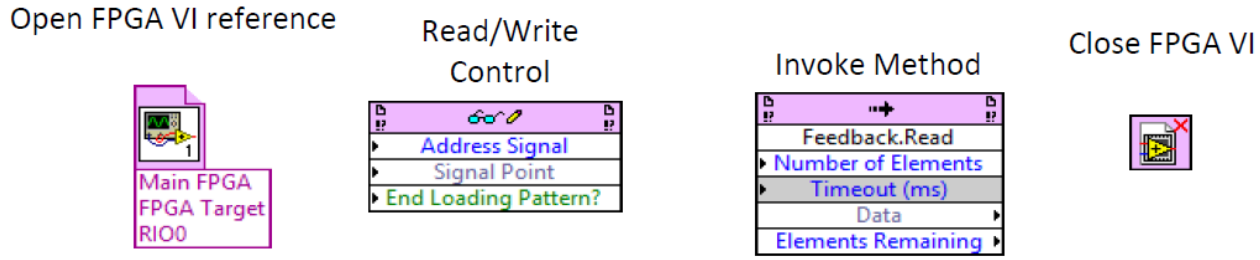


Figura 99: Funzioni di comunicazione su Real Time.

Per stabilire la connessione tra i due livelli si utilizza all'interno del VI sul RT la funzione *Open FPGA VI Reference*. Le funzioni *Read/Write Control* permettono, di leggere i valori dagli indicatori o di scrivere dei valori sui controlli, presenti nel Front Panel dell'FPGA. Per invocare un metodo o un'azione dal VI sul RT, si utilizza la funzione *Invoke Method*: i metodi utilizzati sono il *Run*, l'*Abort*, il *Reset* e le operazioni di lettura e scrittura sulle *FIFO DMA*. Con la funzione *Close FPGA VI Reference* si chiude il riferimento, si ferma l'esecuzione dell'FPGA e se ne effettua il reset riportando i valori alle condizioni iniziali (se è richiesto dal programmatore).
Riguardo alla comunicazione tramite *DMA FIFO* (Figura 100), sul lato FPGA è presente il DMA Writer che riceve i dati dai moduli di input del c-Rio, li bufferizza e li manda sul *DMA Channel* verso il RT. Sul lato Real-Time, i dati ricevuti dalla DMA vengono ulteriormente bufferizzati, fino a quando non vengono letti attraverso la *FIFO Read Method*. L'intero processo di scambio dati viene gestito dall'*NI DMA Engine*, che coordina tra loro i due buffer e la DMA. Per porre rimedio alle diverse velocità con cui i dati vengono scritti e letti, è sufficiente sul RT leggere più elementi contemporaneamente: dunque, mentre sul lato FPGA i dati vengono scritti singolarmente (si ha

un singolo numero in formato *Fixed-Point* in ingresso), sul lato Real-Time vengono letti un numero di elementi (in formato array) pari ad un valore impostato durante la programmazione. Utilizzando dati Fixed-Point, la lunghezza massima di una coda del tipo *Target to Host DMA* è pari a 1023 elementi.

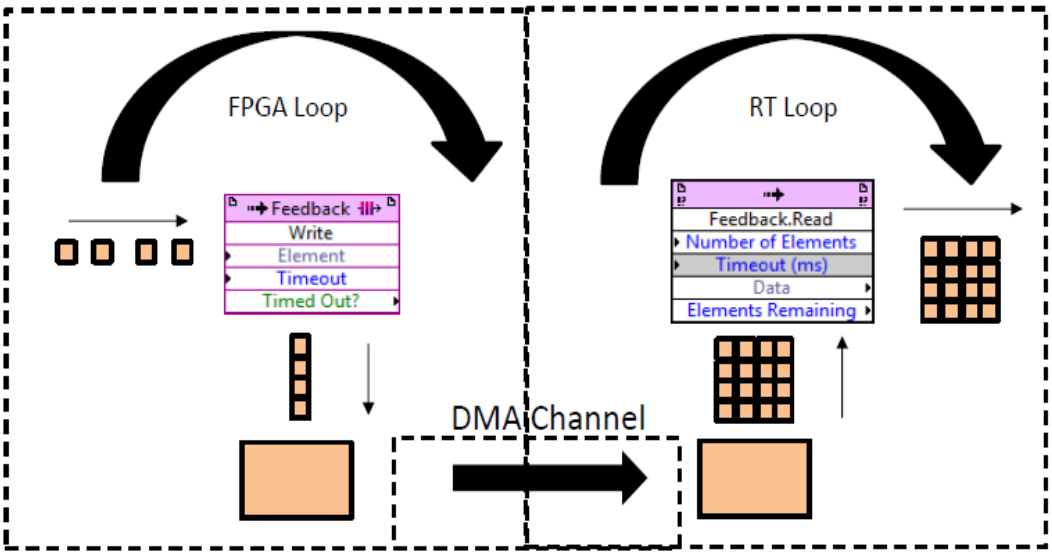


Figura 100: Principio di funzionamento di una DMA FIFO.

4.2.3.2 Main RT VI

Il VI posto sul livello RT ha il compito principale di chiudere l'anello di comunicazione tra il PC e l'FPGA. Con un'architettura hardware del tipo PC/c-RIO non è possibile comunicare direttamente con l'FPGA, cosa che sarebbe stata invece possibile disponendo di una scheda PCI-Express con chip FPGA integrato. La struttura utilizzata è del tipo PC/c-Rio, per cui è stato necessario lo sviluppo della "Main RT VI", riportata in Figura 101.

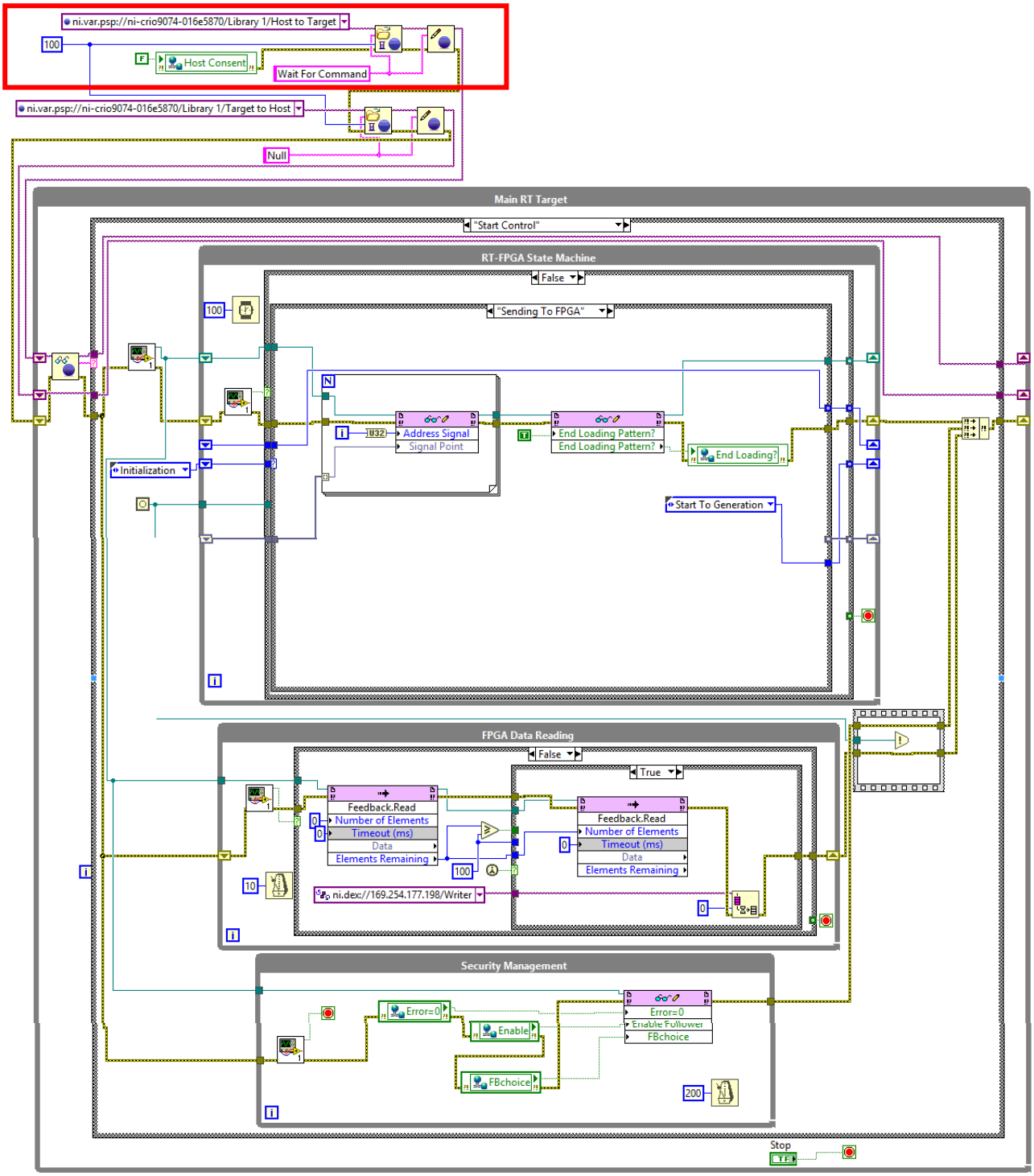


Figura 101: Block Diagram della Main RT VI.

Nella *Case Structure* più esterna, all'interno del *While Loop* denominato *Main RT Target*, sono presenti i casi che permettono di ottenere il percorso di comunicazione tra PC e c-RIO mediante *Network Stream* e lo *start* e *stop* della gestione segnali. I casi di tale Case Structure sono:

1) *Wait For Command (*Figura 102*);*
2) *Create Writer Endpoint (*Figura 103*);*
3) *Reader Endpoint Created (*Figura 104*);*
4) *Stop Control (*Figura 105*);*
5) *Close Writer Endpoint (*Figura 106*);*
6) *Start Control (*Figura 108*).*

Al primo avvio della VI, vengono inizializzate le due *Shared Variable* di tipo *String* all'esterno del loop, necessarie a coordinare lo scambio dati tra i due sistemi in base ai comandi dell'utente. Tramite i comandi dell'utente, si attivano delle strutture automatizzate, situate nel *Main Host VI,* che avviano i casi a livello RT nell'ordine necessario al compimento di una determinata azione mediante un uso combinato di *Shared Variable* e macchine a stati. Per stabilire la comunicazione tra livello PC e livello RT risulta fondamentale la temporizzazione delle macchine a stati in entrambi i livelli che è stata stabilita attraverso il *debugging*.
La porzione di codice nel riquadro rosso in Figura 101 inizializza una struttura di trasporto dati di tipo *Shared Variable* che movimenta stringhe dal PC verso il RT allo scopo di sincronizzare la creazione della *Network Stream.* L'altra porzione svolge lo stesso compito ma riguarda il flusso di dati opposto.

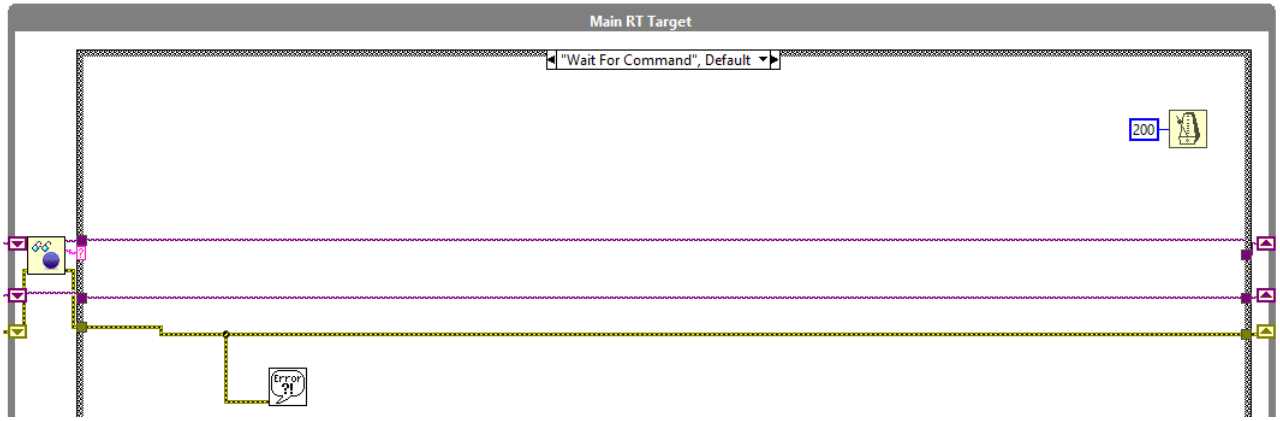


Figura 102: Caso Wait For Command.

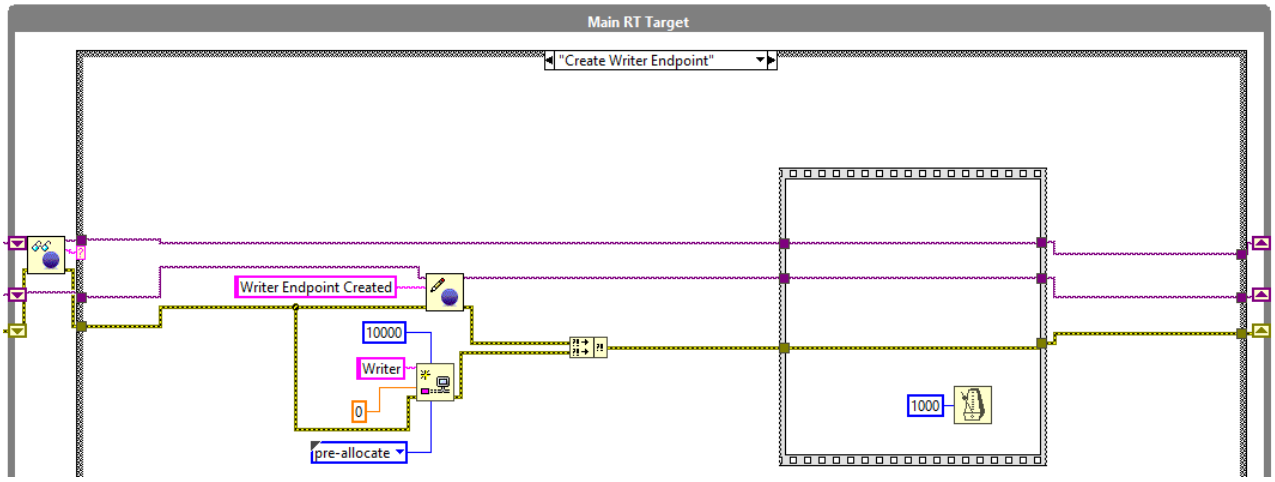


Figura 103: Caso Create Writer Endpoint.

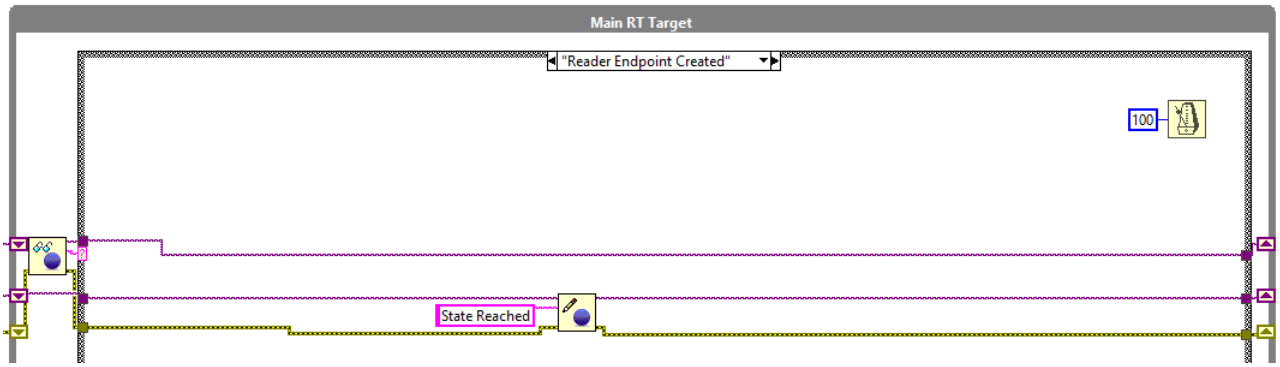


Figura 104: Caso Reader Endpoint Created.

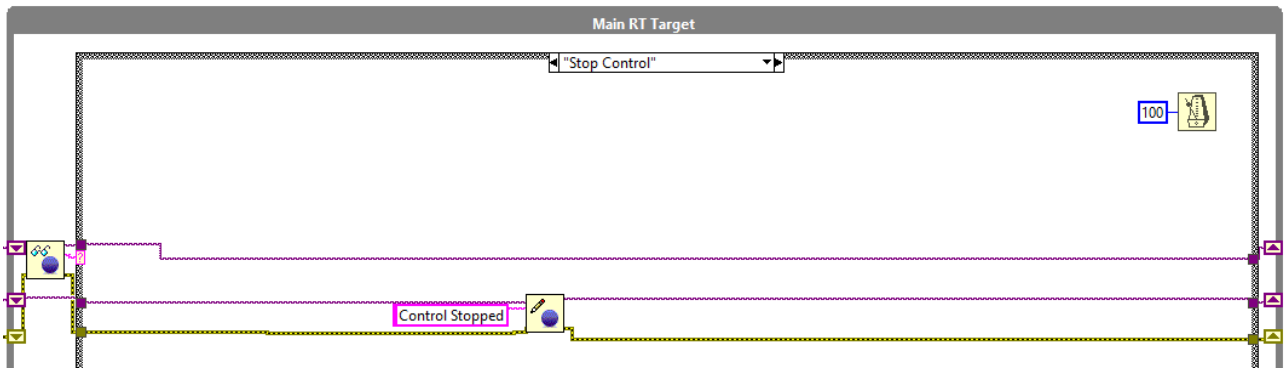


Figura 105: Caso Stop Control.

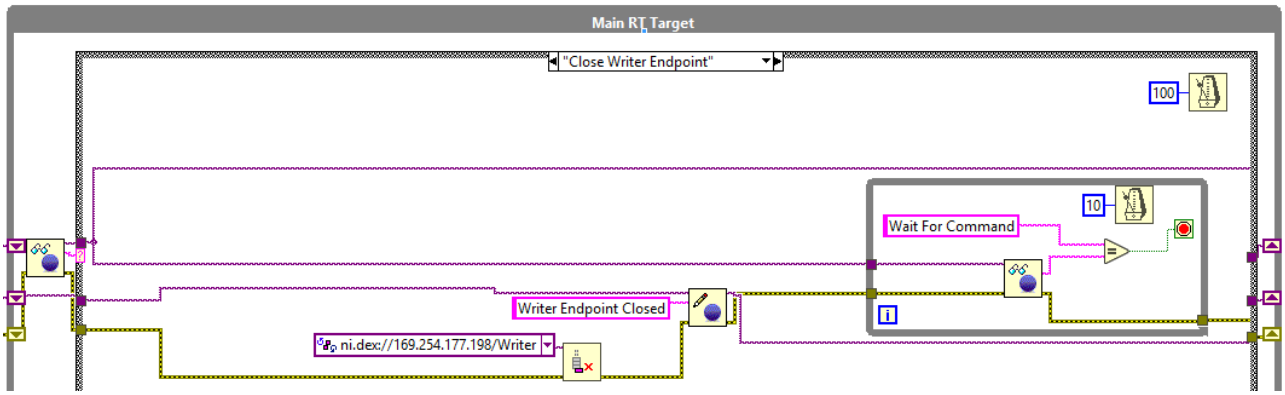


Figura 106: Caso Close Writer Endpoint.

Per quanto riguarda il caso Start Control, questo costituisce il caso più interessante dal punto di vista della programmazione essendo tutti gli altri casi destinati unicamente all'instaurazione e al termine delle connessioni, per cui si entra nel dettaglio dell'architettura. Premendo il tasto *Start Control* sull'interfaccia utente, il Real-Time inizia a lavorare secondo il compito prefissatogli in fase di pre-sviluppo dell'intero software, ovvero chiudere l'anello di comunicazione bilaterale tra PC ed FPGA. All'interno della Case Structure, sono presenti i seguenti loop:

1) RT-FPGA State Machine;
2) FPGA Data Reading;
3) Security & Management.

4.2.3.2.1 RT-FPGA State Machine

All'interno del loop *RT-FPGA State Machine* vi è una macchina a stati che gestisce la comunicazione con l'FPGA. In Figura 107 è riassunto il funzionamento della macchina a stati mediante un diagramma di flusso.

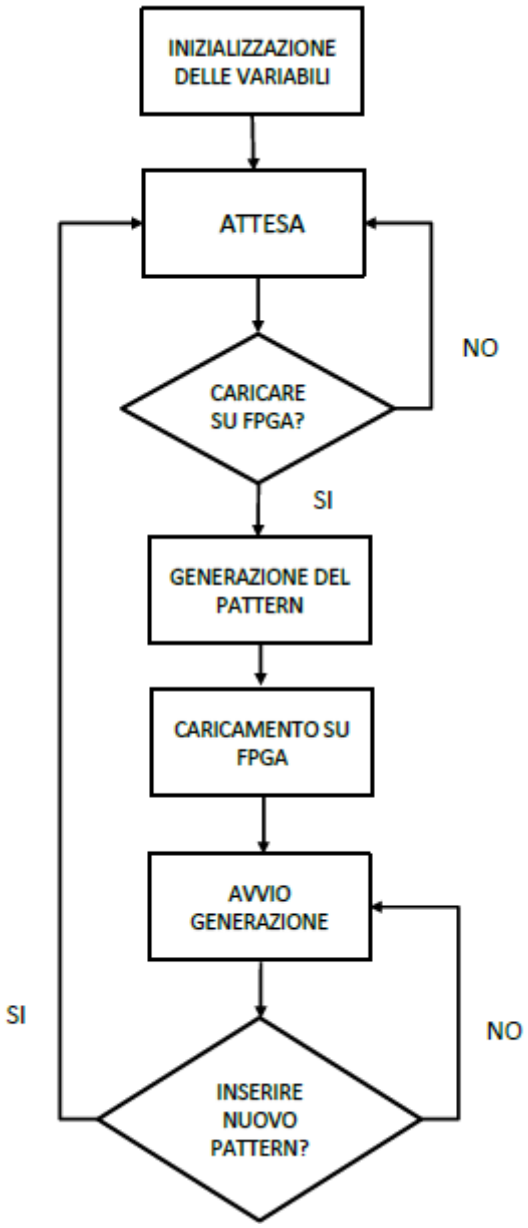


Figura 107: Diagramma di flusso della macchina a stati RT-FPGA State Machine.

Nella prima fase della macchina a stati vengono inizializzati tutti i comandi, compresi i parametri del segnale di set e i comandi booleani (Figura 108). In questo modo, ad un nuovo avvio dell'applicazione, si evita di lavorare con valori rimasti in memoria dall'esecuzione precedente del programma. Inoltre si crea il riferimento con l'FPGA in modo da utilizzare le relative funzioni; in aggiunta si dispone di una SubVI che riporta su RT il consenso in arrivo da Host in modo da selezionare la *Case Structure* adeguata.

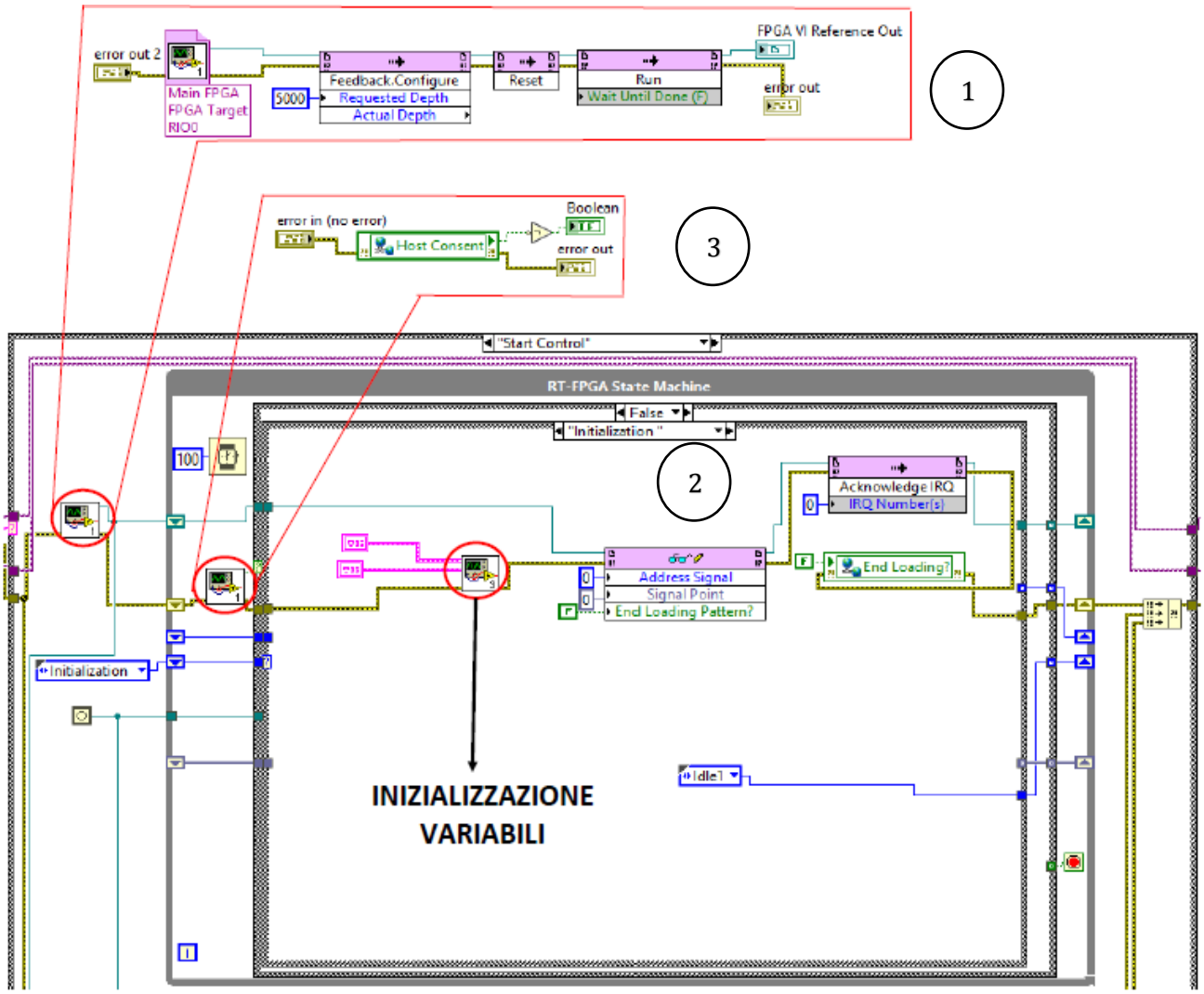


Figura 108: SubVI creazione del FPGA Reference(1), Macchina a stati RT-FPGA Machine con caso Initialization (2) e SubVI del consenso da parte dell'Host (3).

Successivamente alla seconda fase d'attesa, vi è la terza fase *Generating Pattern* (Figura 109) dove si genera un segnale di set sinusoidale o a rampa. Si tratta di una sinusoide di ampiezza unitaria formata da 100 punti e di una rampa di ampiezza unitaria e di 70 punti di salita. Questo modello verrà modificato nell'FPGA in base ai parametri definiti dall'utente come ampiezza, offset, frequenza e pendenza. Il pattern generato, essendo un array 1D, non possiede alcuna informazione temporale: di conseguenza, la frequenza del segnale di SET e la durata della rampa verranno definite in un secondo momento.

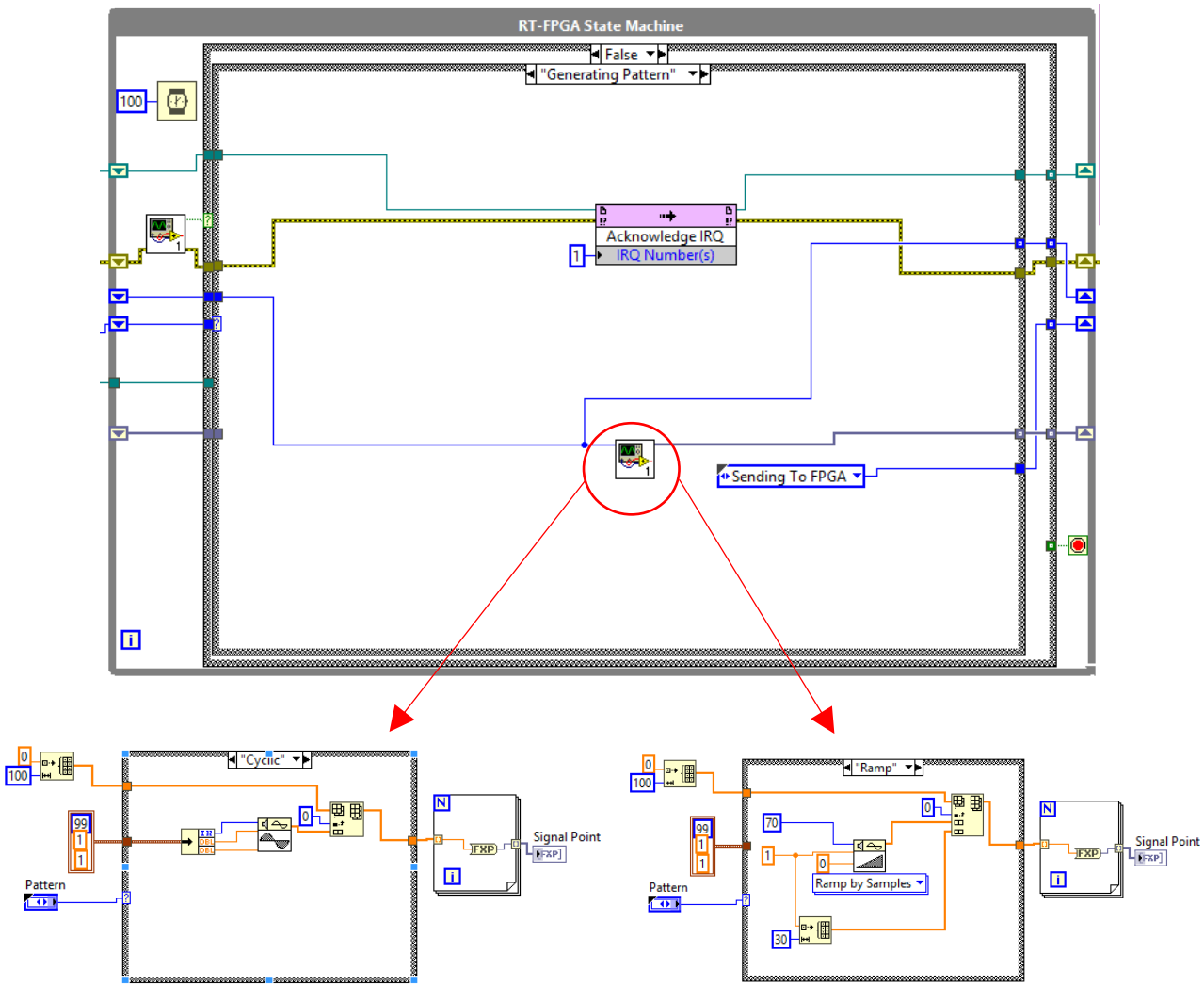


Figura 109: Caso Generating Pattern e SubVI Cyclic/Ramp.

Nella quarta fase il pattern è inviato al livello FPGA come dato Fixed-Point ed ogni dato è accoppiato ad un indirizzo (Figura 110).

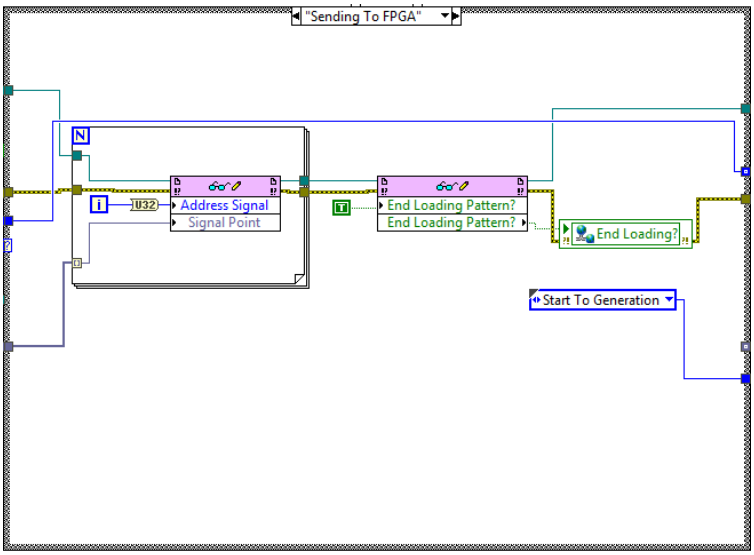


Figura 110: Caso Sending to FPGA.

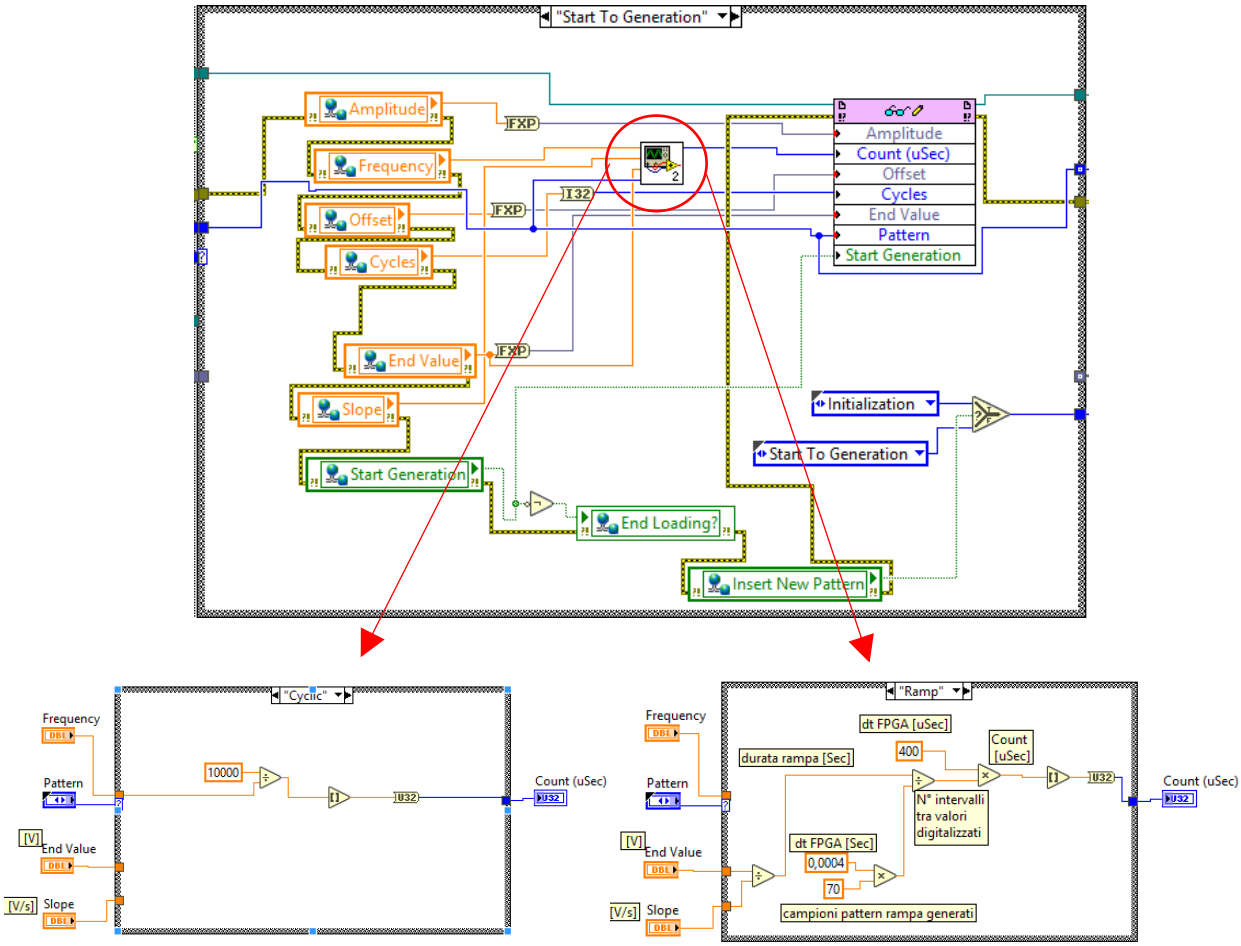


Figura 111: Caso Start to Generation, con SubVI di calcolo temporizzazione per sinusoide e per rampa.

Nella quinta fase *Start to Generation* si invia al livello FPGA i parametri a bassa cadenza dalle *Shared Variable* in arrivo da *Host* al livello FPGA mediante la funzione *Read/Write Control*. Inoltre si esegue l'operazione di ottenere un parametro fondamentale (Count µsec) per temporizzare un determinato loop in FPGA che determina la corretta forma del segnale.
Prendendo in esame il caso di segnale a rampa (Figura 111), si digitalizza la rampa in modo da ottenere una struttura a gradini, l'intervallo di tempo di ciascun gradino è ulteriormente suddiviso in campioni di durata pari al periodo del loop primario dell'FPGA, mentre ciascuna durata del gradino rappresenta il periodo del loop secondario dell'FPGA ed è esattamente il Count µsec ottenuto (Figura 112). Discorso analogo per il pattern sinusoidale. Si rimanda al capitolo inerente al Main FPGA VI.

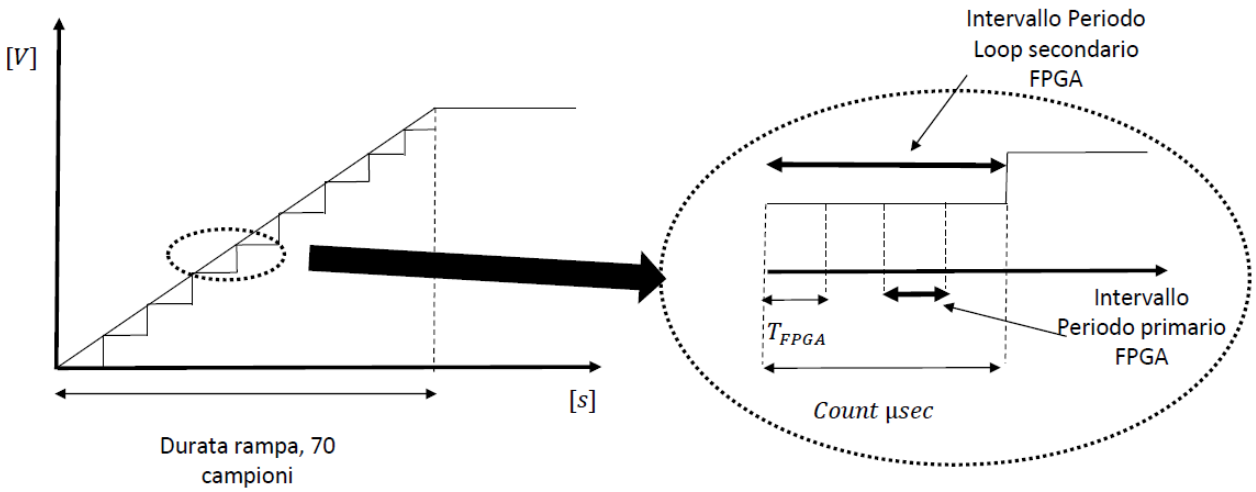


Figura 112: Logica di generazione della rampa.

Nel caso in cui il consenso proveniente da Host fosse negativo, si attiva la *Case Structure 'True'*, il che disabilita l'applicazione del Driver e chiude il riferimento con il livello FPGA (Figura 113).

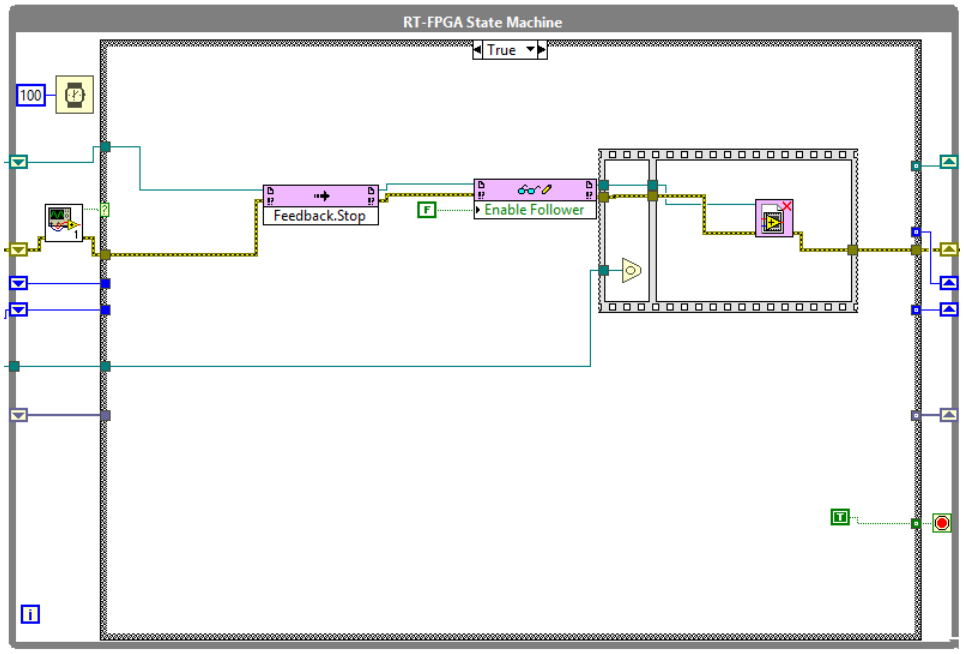


Figura 113: Case structure Host Consent 'True'

4.2.3.2.2 FPGA Data Reading

In questo loop i dati provenienti dall'FPGA vengono letti dalla DMA FIFO e scritti sulla N*etwork Stream.* Al primo avvio del programma, per essere sicuri che non siano rimasti dei dati dal precedente avvio, è stata inserito un *First Call* collegato alla *Case Structure* in Figura 114. Questo operatore booleano diventa vero solo al primo ciclo dopo l'avvio del programma, in questo modo tutti gli eventuali dati rimasti in coda verranno eliminati tramite la funzione *Flush Stream*.

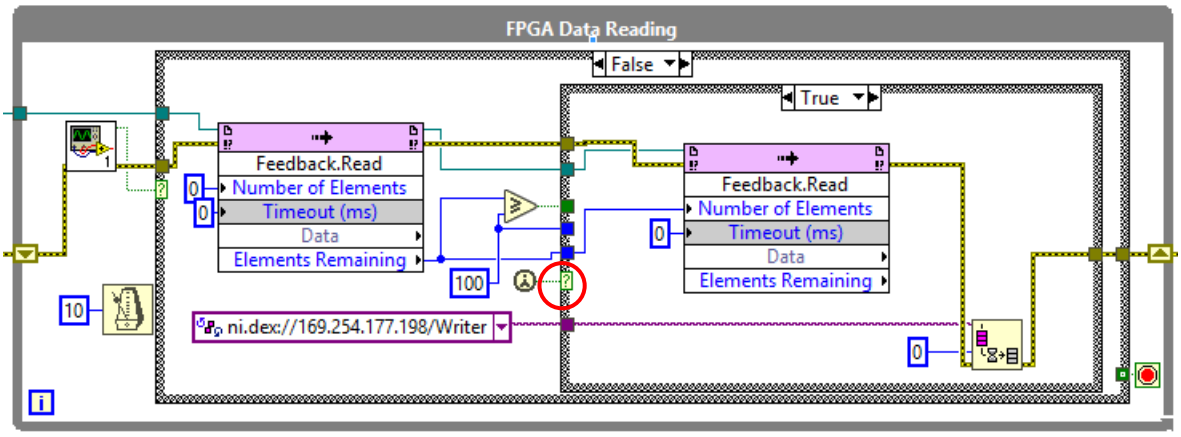


Figura 114: FPGA Data Reading First Call.

Come detto in precedenza, ad ogni ciclo il Real-Time leggerà 25 campioni per canale per un totale di quattro canali. Quindi, per evitare che l'ordine dei canali cambi durante il processo di scrittura nella *Network Stream*, la scrittura non si avvia fino a quando non si raggiungono i 100 elementi cioè 25 campioni per ogni canale (Figura 114).

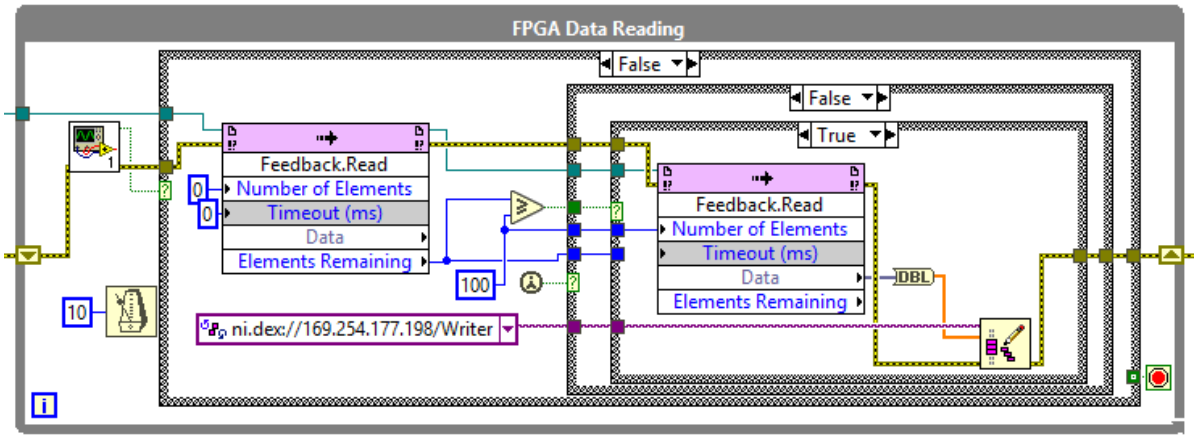


Figura 115: FPGA Data Reading Lettura.

Quando è raggiunto il numero di elementi richiesto, i dati vengono convertiti da *Fixed Point* a *Double Precision* per essere poi scritti sulla *Network Stream* pe essere trasmessi al *Main Host VI* (Figura 115).

4.2.3.2.3 Security & Management

L'ultimo loop della macchina a stati nella fase *Start Control è Security & Management* (Figura 116), nel quale, attraverso tre Shared Variable, si inviano tramite una funzione *Read/Write Control* i booleani che attivano un voltaggio uguale a zero, abilitano/disabilitano il driver e scelgono il *2nd Feedback.*

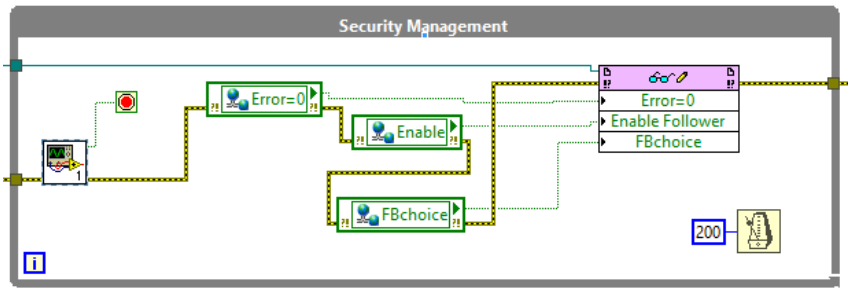


Figura 116: Security & Management Loop.

4.2.4 Codice FPGA

Il codice creato a livello FPGA (Figura 117) costituisce l'ultimo atto nella gestione del segnale. In esso sono implementati i moduli I/O per l'acquisizione e l'invio del segnale di comando, nonché numerose strutture ausiliarie che si occupano di scambio dati e sincronizzazione con il RT.

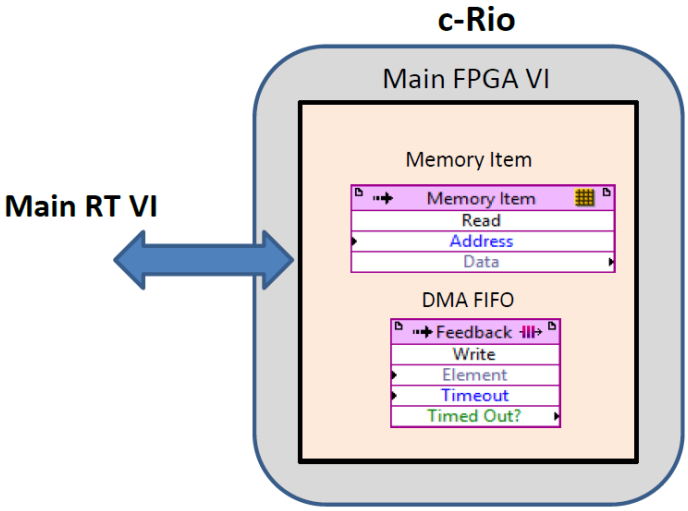


Figura 117: Livello FPGA.

In questo VI, a differenza dei precedenti, è utilizzata una nuova tipologia di dato, contraddistinto dal colore grigio: il *Fixed Point.* Si adotta questa tipologia sulle piattaforme FPGA perché nella loro progettazione il numero di bit di un dato è un parametro non trascurabile. Di conseguenza, per ottenere un corretto funzionamento e le prestazioni desiderate, è necessario utilizzare una rappresentazione a virgola fissa, ovvero una tipologia di dato che mantenga sempre lo stesso numero di bit. Rappresentando un certo dato con un numero elevato di bit aumenta la precisione ma aumenta inevitabilmente anche il numero di risorse da allocare nell'FPGA. Solitamente, è buona norma configurare il dato in base alla sua natura, ad esempio i parametri del controllo non necessitano di un numero alto di bit come invece succede per i segnali provenienti dai trasduttori.

4.2.4.1 Main FPGA VI

Nel Main FPGA VI (Figura 118) sono presenti tre *While Loop*, operanti in parallelo, di cui i primi due consentono la gestione dei segnali analogici, mentre il restante gestisce i segnali digitali diretti al Driver.

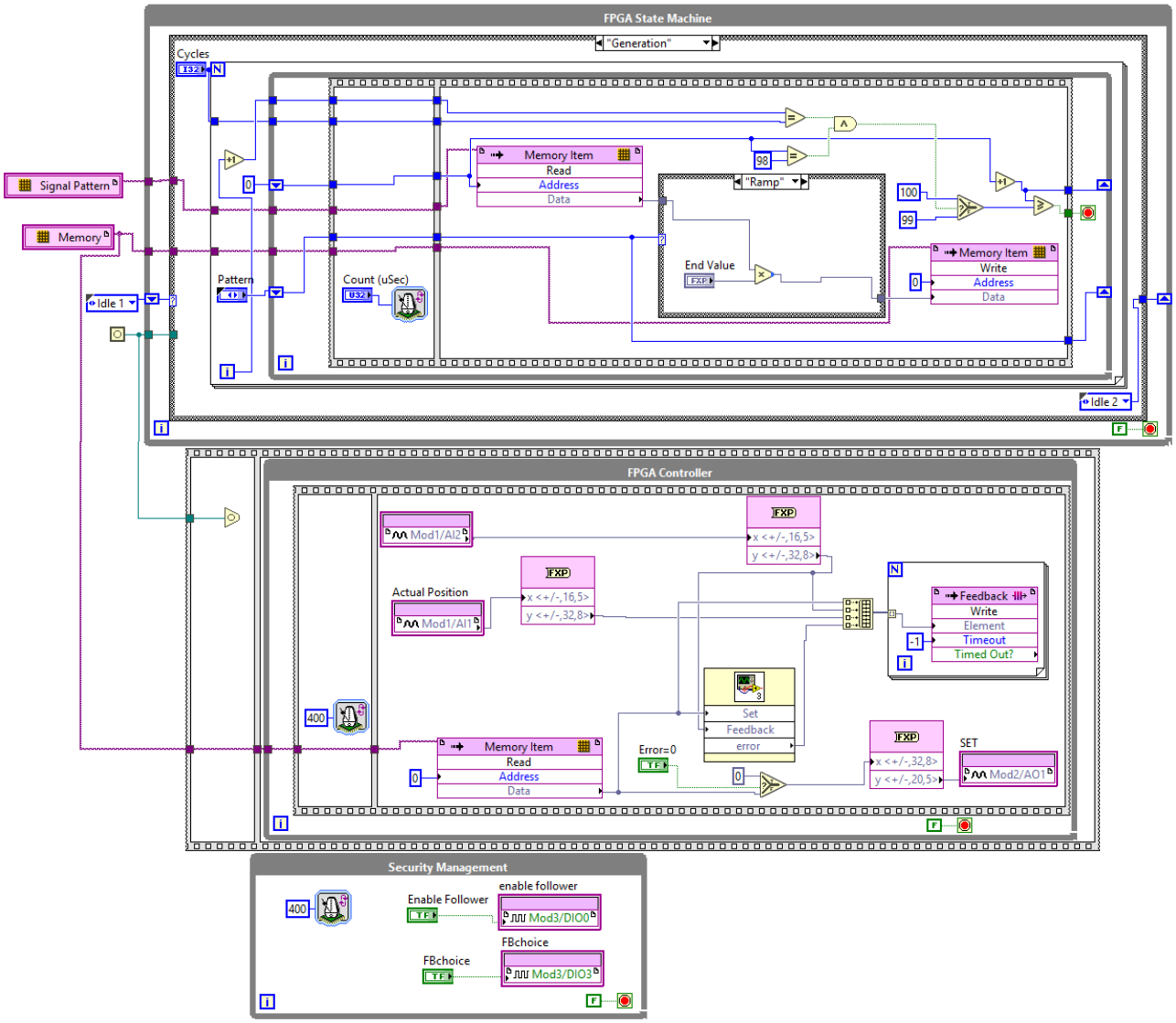


Figura 118: Block Diagram del Main FPGA VI.

4.2.4.1.1 FPGA State Machine

Il primo loop (chiamato in precedenza loop secondario del FPGA), *FPGA State Machine*, sincronizza l'intero VI con il RT ed attinge a tutte le informazioni (scelte dall'utente sul PC host) necessarie alla gestione dei segnali. Tali informazioni sono il *pattern* e le *feature* del segnale scelte dall'utente che giungono dal livello RT.
Per facilitare la comprensione di tale loop, strutturato con una macchina a stati, si riporta il diagramma di flusso (Figura 119).

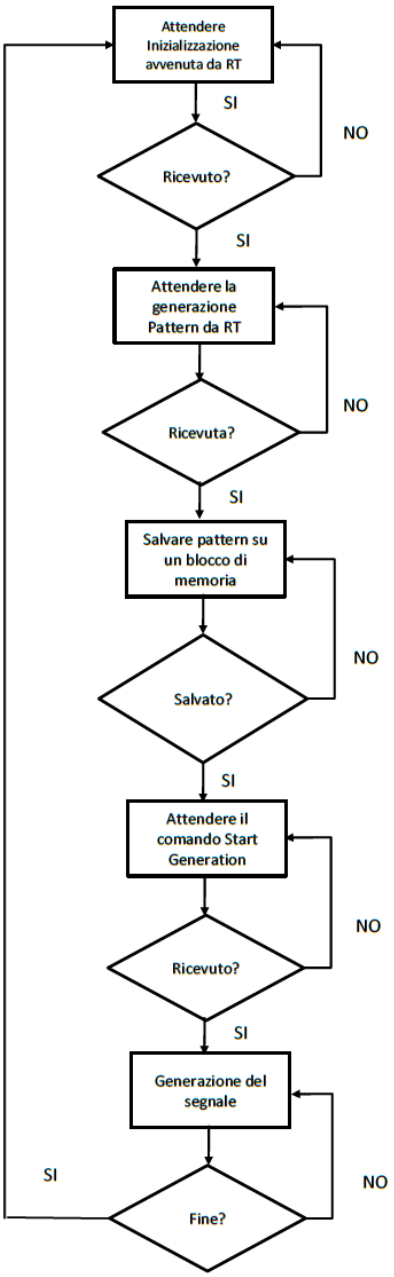


Figura 119: Diagramma di flusso loop FPGA State Machine.

Le fasi di attesa sono gestite mediante *l'Invoke Method* su RT *Acknowledge IRQ* e il relativo blocco *Interrupt* su FPGA posto all'interno di una *Flat Sequence Structure*. Quando FPGA riceve il consenso da RT si passa al frame successivo.
Successivamente il pattern è caricato su un blocco di memoria chiamato *Memory Item* che tiene in memoria il dato e l'indirizzo attribuito (Figura 120).

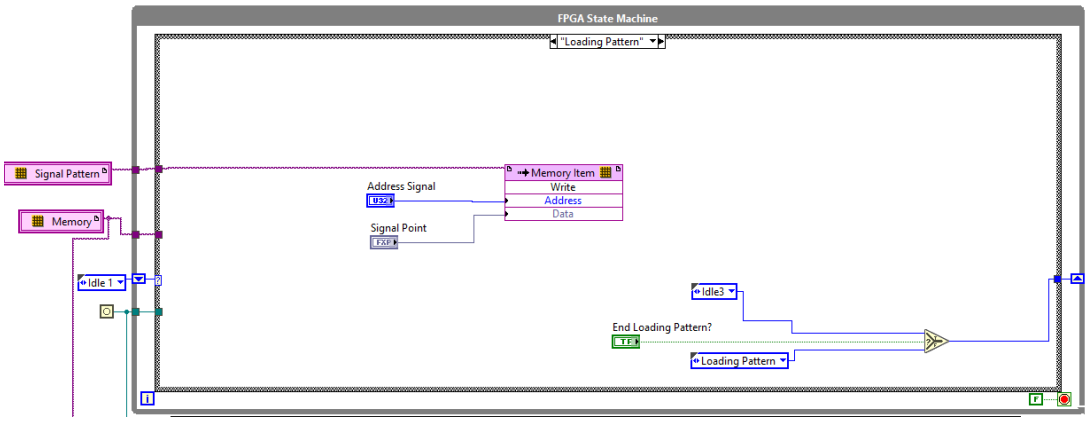


Figura 120: Fase di Salvataggio del pattern sul blocco Memory Item.

Infine, l'ultima macchina a stati è quella di generazione del segnale ed è quella su cui si focalizza l'interesse (Figura 121).

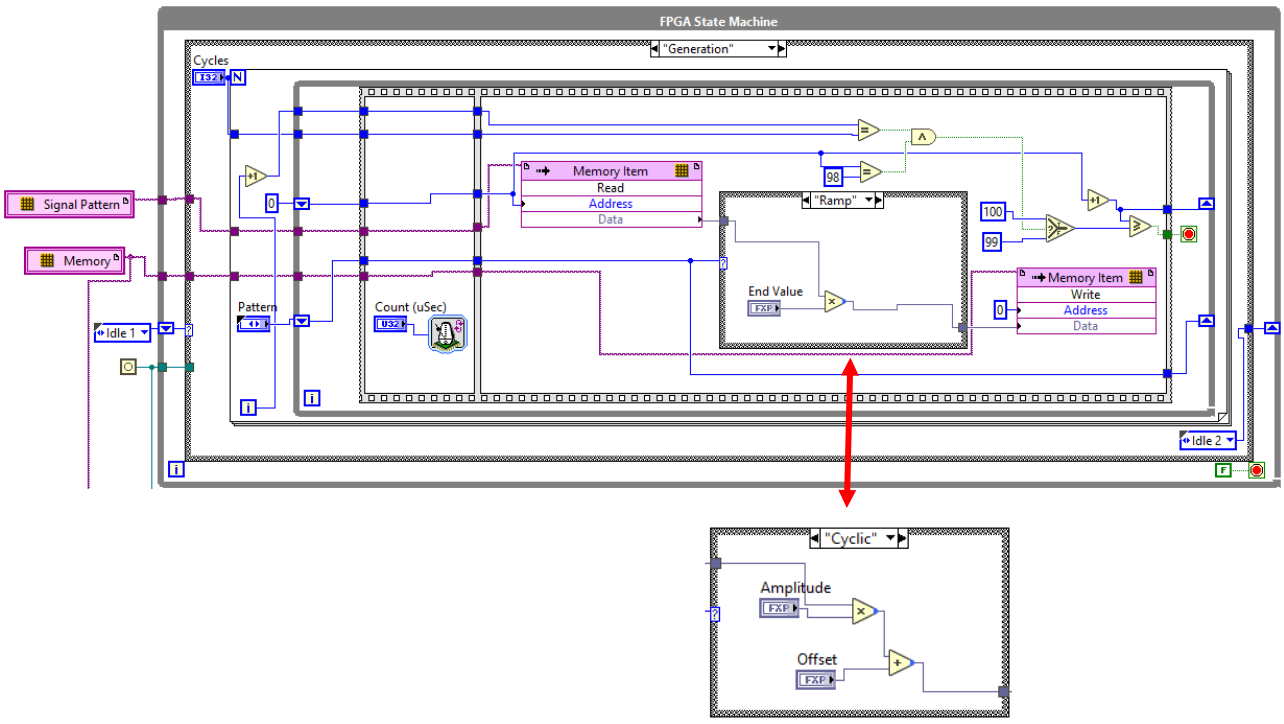


Figura 121: Fase Generazione del segnale a rampa e sinusoidale.

Questo stato contiene la parte di codice che consente di generare un segnale di SET partendo dal *pattern* caricato in precedenza (Figura 121). Il pattern stabilito a livello RT consiste in un array di 100 punti ottenuti campionando una sinusoide o una rampa. Prendendo in esempio il pattern sinusoidale, la frequenza del segnale in uscita dal generatore dipende dunque dalla frequenza con cui questi punti vengono letti in successione. Grazie alla temporizzazione ottenuto mediante il parametro *Count µsec,* il loop secondario ha una durata ciclo maggiore rispetto al loop primario, che presenta una frequenza di acquisizione fissata a 2500 Hz (0.0004 s). In questo modo il loop primario leggerà più volte gli elementi del pattern generato dal loop secondario, 'digitalizzando' il pattern sinusoidale.
Ad esempio se si intende ottenere una sinusoide con frequenza 1Hz, *Count µsec* risulterà quello di (eq.14).

$$Count\ \mu sec = \frac{10000}{frequenza} = 0.01\ s \qquad (14)$$

Che è il periodo del loop secondario.

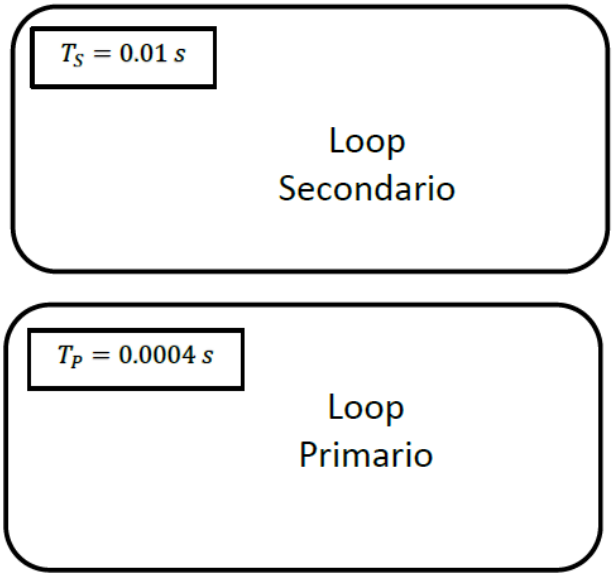


Figura 122: Schema di esempio temporizzazione in FPGA.

Per cui facendo riferimento alla Figura 122 e ricordando che i campioni del pattern per ogni ciclo sono N=100 (eq.):

$$\frac{T_S}{T_P} \cdot N \cdot T_P = Periodo\ sinusoide\ [s] \rightarrow \frac{0.01}{0.0004} \cdot 100 \cdot 0.0004 = 1s \rightarrow \mathbf{1Hz}$$

$$dove\ \frac{T_S}{T_P} : numero\ di\ volte\ che\ il\ loop\ primario\ legge\ il\ dato\ generato\ dal\ loop\ secondario$$

È importante sottolineare inoltre come nel caso di *pattern* a rampa il parametro *Cycles* sia fissato pari ad uno, grazie al codice su RT: questo parametro stabilisce il numero di cicli per la sinusoide con l'ausilio di un ciclo *For*.

4.2.4.1.2 FPGA Manager e Driver Management

Per quanto riguarda il secondo loop (loop primario in Figura 123), oltre ad ottenere il *pattern* 'digitalizzato', in tale loop avviene la trasmissione dei segnali mediante i moduli del c-Rio inseriti nel *chassis.* I segnali, sia in uscita che in ingresso, vengono configurati attraverso dei blocchi di configurazione *Fixed-Point* in modo da ottenere un dato più leggero e facilmente gestibile dal FPGA.

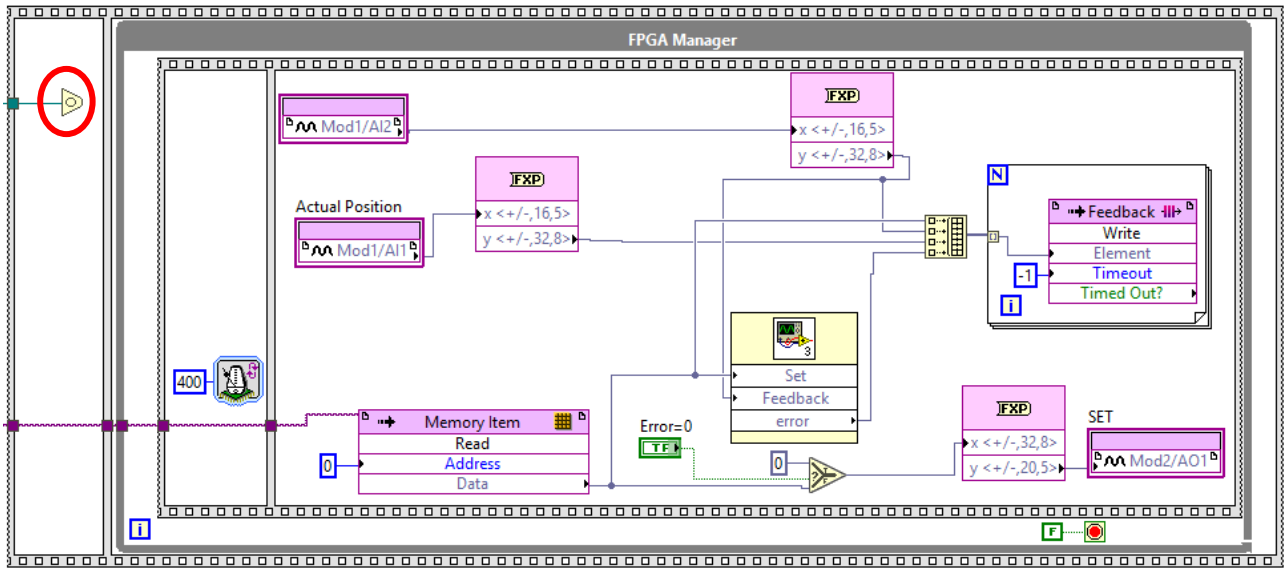


Figura 123: Loop primario in Main FPGA VI.

Il loop è inserito in un frame che viene attivato soltanto in seguito alla abilitazione della funzione *Wait on Occurrence* (cerchiato in rosso) da parte della funzione *Set Occurrence* gestita dal loop secondario durante la seconda fase d'attesa.
I quattro segnali *Set di posizione*, *Feedback di posizione*, *Feedback di velocità o corrente* e *Errore* vengono inviati alla *DMA FIFO Feedback Write,* mediante un *For Loop,* al livello RT in modo da chiudere l'anello di gestione segnali trattato finora.
Infine il terzo ed ultimo loop è quello di gestione del Driver (Figura 124), dove attraverso due blocchi di moduli digitali si gestiscono l'abilitazione/disabilitazione dell'applicazione del driver e la scelta del secondo feedback da acquisire.

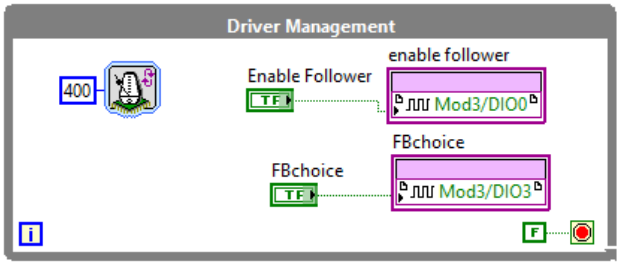


Figura 124: Driver Management Loop.

Riassumendo, i segnali gestiti in FPGA e i relativi moduli e pin utilizzati sono esposti in Tabella 30.

Modulo	**Pin**	**Segnale**
DI/O	0	Abilitazione applicazione driver (*Enable Application*)
DI/O	3	Selezione *Feedback* Corrente/Velocità (*2nd FB*)
AO	1	*Setpoint* Posizione
AI	1	*Feedback* Posizione
AI	2	*Feedback* Corrente/Velocità

Tabella 30: Legenda segnali gestiti da FPGA.

5. Modello Lineare

Per caratterizzare il funzionamento dell'EMA risulta fondamentale elaborare un modello matematico dell'intero servosistema da affiancare alle acquisizioni sperimentali. Lo studio preliminare è effettuato con un modello lineare che, nonostante i suoi limiti e la presenza non trascurabile di non linearità, permette di prevedere il comportamento del sistema reale in particolari condizioni e con variazioni di piccola entità dei parametri in ingresso.
Il primo passo nella creazione del modello consiste nell'individuare le equazioni che caratterizzano ogni componente del sistema, il cui schema generale è rappresentato in Figura 125.

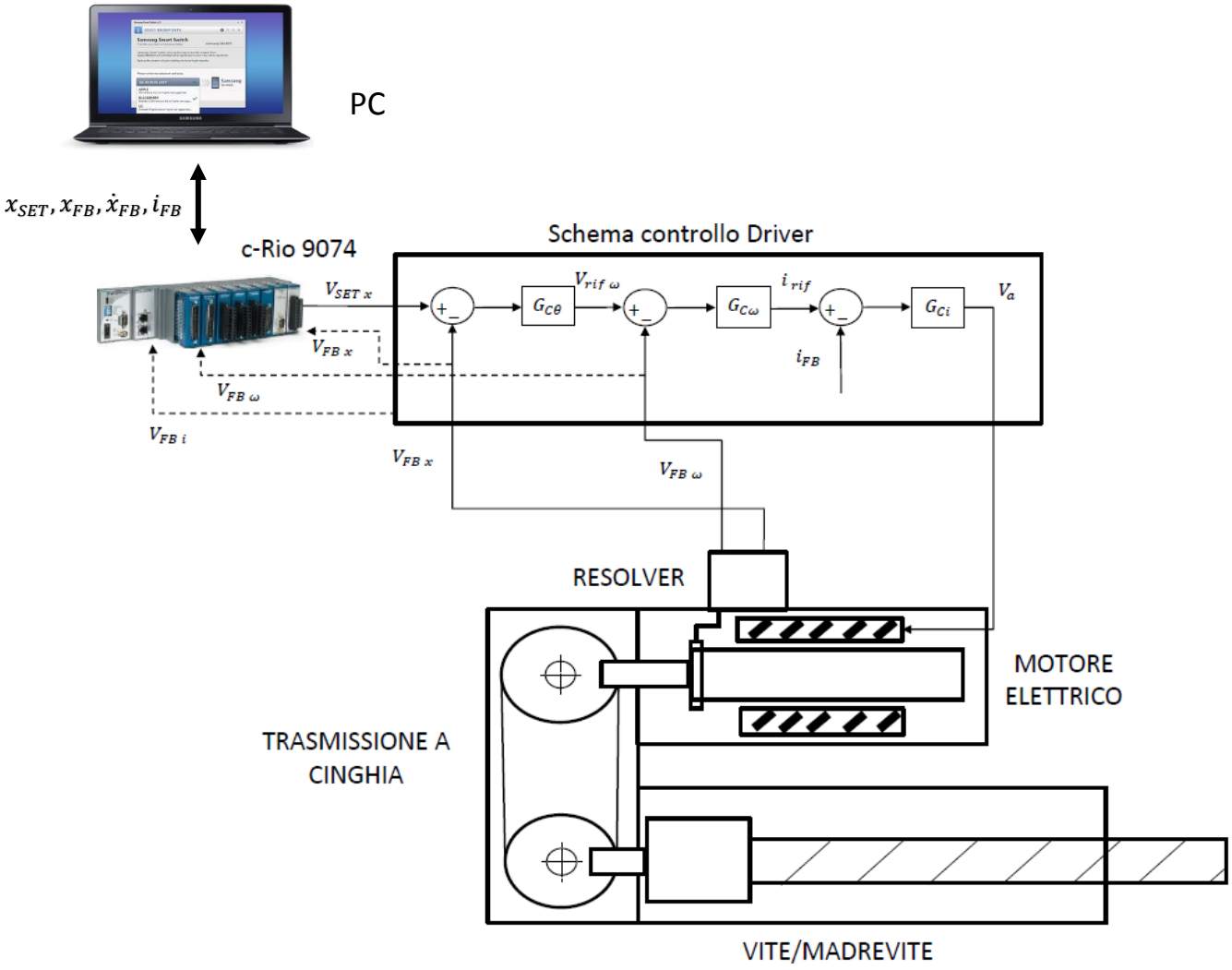


Figura 125: Schema meccatronico del sistema.

5.1 Modellazione dei componenti

La modellazione dell'EMA è effettuata in base a quanto riportato secondo la letteratura e, per quanto riguarda la parte inerente all'azionamento, in base a quanto implementato nel controllo del Driver in modo tale da ottenere un modello il più possibile confrontabile con quest'ultimo.
Per ciascuno dei componenti costituenti l'EMA se ne individuano le equazioni caratterizzanti, in modo da poter modellizzare il servosistema.

5.1.1 Motore elettrico

Il motore elettrico è modellizzato come un motore DC, nonostante il motore sia a corrente alternata AC. Dal punto di vista pratico questo facilita notevolmente la modellazione e tale semplificazione non causa errori rilevanti rispetto al caso in cui si fosse adottato un modello notevolmente più complesso AC.
Per il motore elettrico, supposto a corrente continua a magneti permanenti, si possono considerare le equazioni alla maglia del circuito elettrico e della generazione di coppia magnetica.

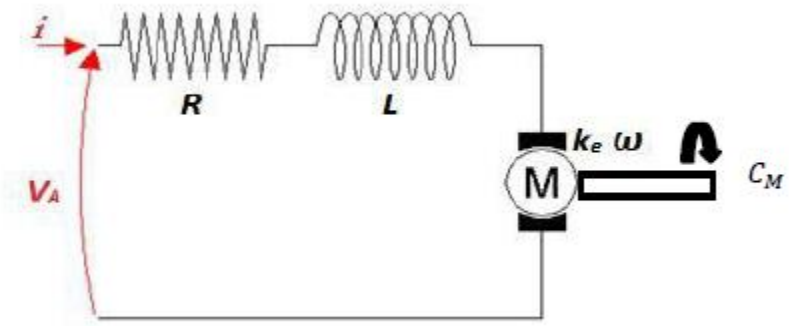


Figura 126: Rappresentazione grafica del modello del motore elettrico.

Il circuito, esposto in Figura 126, è caratterizzato da:

- tensione di armatura V_A [V]
- corrente elettrica di armatura i [A]
- resistenza elettrica R [Ω]
- induttanza L [H]
- costante di tensione del motore elettrico k_e [V/rad/s]
- velocità angolare del rotore ω [rad/s]

Dunque è possibile scrivere l'equazione alla maglia del circuito elettrico (eq.15):

$$V_A = Ri + L\frac{di}{dt} + k_e\omega \qquad (15)$$

$$\bar{\imath} = \frac{1}{(Ls + R)}(V_A - k_e\bar{\omega}) \qquad (16)$$

La corrente di comando del motore genera una coppia magnetica C_M espressa da (eq.17).

$$C_M = k_c \cdot i \qquad (17)$$

Dove k_c è la costante di coppia del motore.
La forza contro elettromotrice $k_e\bar{\omega}$ (espressa in Volt) presentata in eq.16 è dovuta alla rotazione del rotore il quale rotando, per la legge di Lorentz, induce nell'avvolgimento una corrente nel verso opposto a quella di armatura che si traduce in una tensione elettro-motrice ai capi del circuito opposta a quella di armatura.

5.1.2 Riduttore a cinghia

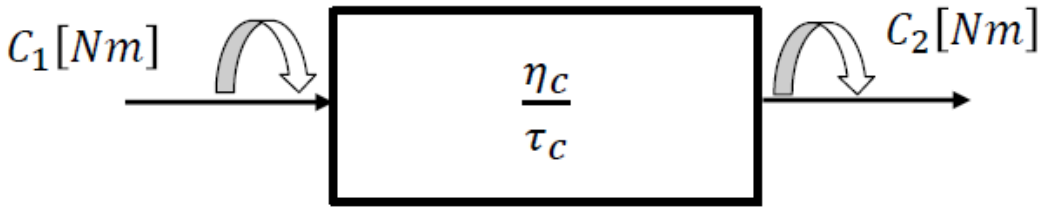


Figura 127: Schema di riporto della coppia a valle del riduttore a cinghia.

$$\eta_c = \frac{C_1\,\dot{\vartheta}_1}{C_2\,\dot{\vartheta}_2} \qquad (18)$$

$$C_2 = \frac{C_1\eta_c}{\tau_c} \qquad (19)$$

Dove:

- η_c rendimento del riduttore a cinghia;
- $\tau_c = \frac{\dot{\vartheta}_1}{\dot{\vartheta}_2}$ rapporto di trasmissione del riduttore a cinghia.

5.1.3 Equilibrio alla rotazione sull'albero motore

Si riportano le coppie sull'albero motore in modo da scrivere l'equazione di equilibrio alla rotazione (eq.20).

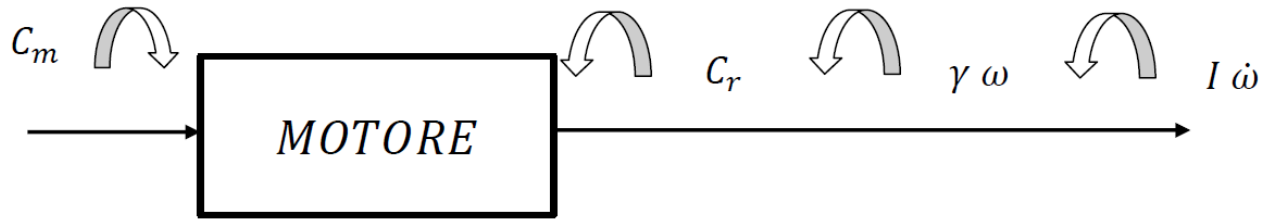


Figura 128: Equilibrio alla rotazione.

$$C_m - C_r - \gamma\omega - I\dot{\omega} = 0 \qquad (20)$$

$$C_m - C_r - \gamma\bar{\omega} - I\bar{\omega}s = 0 \qquad (21)$$

$$\bar{\omega} = \frac{1}{(\gamma + Is)}(C_m - C_r) \qquad (22)$$

Dove:

- C_m è la coppia motrice disponibile in uscita dal blocco motore e riduttore a cinghia [Nm];
- C_r è la coppia resistente applicata dal carico a valle [Nm];
- I è l'inerzia totale riportata sull'asse dell'albero motore (Riporto dell'inerzia sull'albero motore) [$kg\ m^2$];
- γ è l'attrito viscoso dei componenti rotanti $\left[\frac{Nm}{\frac{rad}{s}}\right]$;.

5.1.4 Vite e madrevite

Nel processo, rappresentato dalla trasformazione da moto rotatorio a lineare nel blocco vite/madrevite, entra come disturbo la forza applicata dall'esterno F_e, che corrisponde alla coppia resistente C_r.

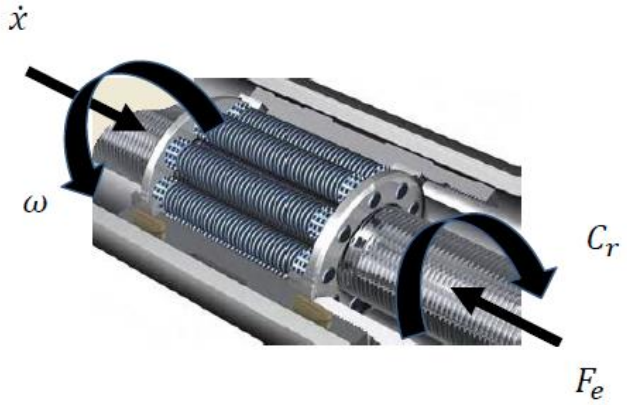


Figura 129: Trasmissione Vite/Madrevite nell'attuatore SKF.

Il *rendimento diretto* della vite con rulli identifica il rendimento della trasmissione a rulli planetari presente nell'attuatore SKF ed è utilizzato in caso di motore rotativo che aziona una traslazione (viceversa si parlerebbe di *rendimento indiretto*). La sua formulazione è espressa dalla eq.23.

$$\eta_v = \frac{F_e\,\dot{x}}{C_r\,\omega} = \frac{F_e}{C_r}\frac{p}{2\pi} \qquad (23)$$

$$C_r = F_e\frac{p}{2\pi}\frac{1}{\eta_v} \qquad (24)$$

Seguendo quanto riportato nella guida SKF per l'attuatore a vite con rulli planetari, il rendimento diretto della vite η_v è calcolato secondo l'(eq.25):

$$\eta_v = \frac{1}{1+\pi\,d_0\frac{\mu}{p}} \qquad (25)$$

$$\begin{cases} \mu = 0.010 & ,per\ \alpha \le 7° \\ \mu = 0.007\,\alpha - 0.04 & ,per\ \alpha > 7° \end{cases} \qquad (26)$$

Dove:

- d_0 è il diametro nominale della vite;
- p è il passo della vite;
- α è l'angolo dell'elica della filettatura della vite.

A valle del blocco trasmissione Vite/Madrevite si ottiene la velocità di rotazione della madrevite ω che,una volta integrata, si traduce nella posizione angolare (eq.27). La vite, grazie al dispositivo anti-rotazione di supporto, trasla solamente, per cui considerando il passo della vite e applicando il rapporto di trasmissione (eq.28) si ricava la posizione lineare della vite x (eq.29).

$$\vartheta = \frac{\overline{\omega}}{s} \qquad (27)$$

$$\frac{x}{\vartheta} = \frac{p}{2\pi} \qquad (28)$$

$$x = \frac{p}{2\pi}\vartheta \qquad (29)$$

Dove:

- ϑ è la posizione angolare della madrevite [rad]
- ω è la velocità angolare della madrevite [rad/s]
- x è la posizione lineare della vite [m]
- p è il passo della vite [m]
- $\frac{p}{2\pi}$ è il rapporto di trasmissione da moto rotatorio a lineare della vite [m/rad]

5.1.5 Riporto dell'inerzia sull'albero motore

Per facilitare la modellizzazione del servosistema, essendo l'EMA in questione un sistema ad interfaccia parallela, si riporta l'inerzia totale I sull'albero motore tramite alcuni accorgimenti partendo dallo schema in Figura 130.

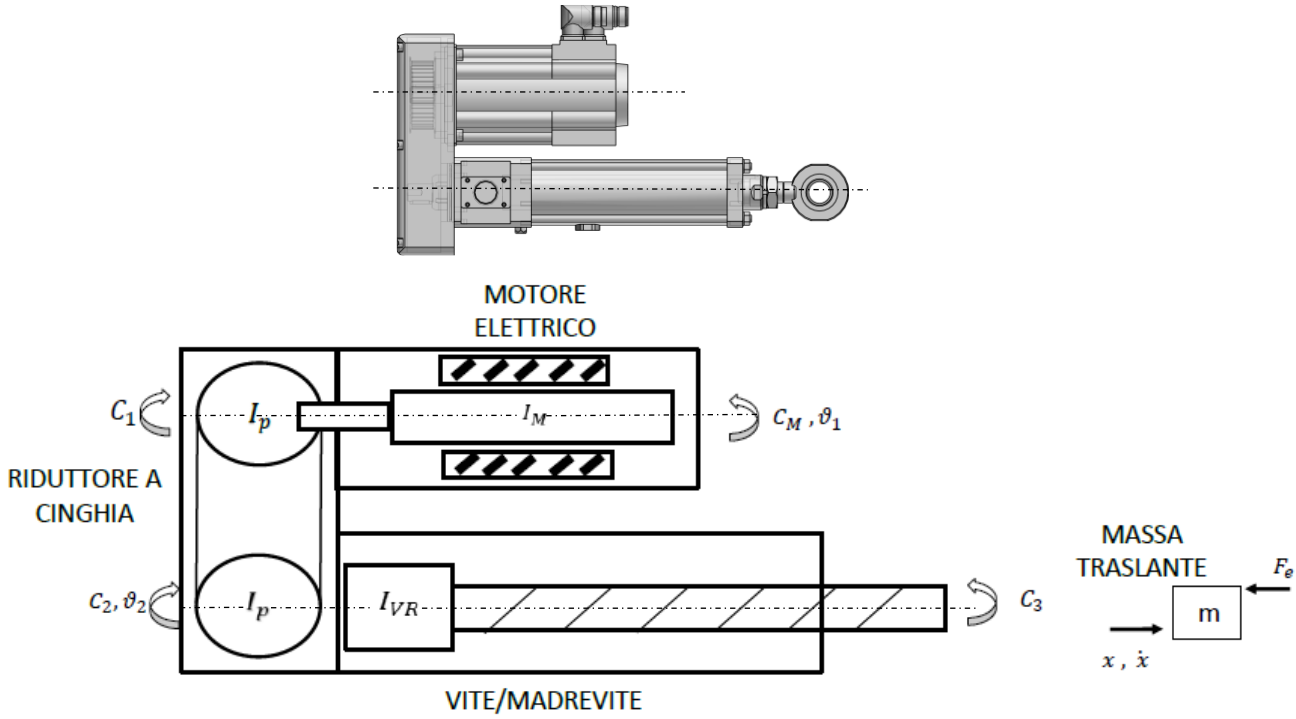


Figura 130: Schema EMA per il riporto dell'inerzia sull'albero motore.

Dove:

- I_M è l'inerzia del motore $[kg\ m^2]$;
- I_p è l'inerzia della puleggia $[kg\ m^2]$;
- I_{VR} è l'inerzia della vite a rulli planetari $[kg\ m^2]$;
- C_M è la coppia motrice del motore $[Nm]$;
- C_1 è la coppia resistente applicata dalla prima puleggia $[Nm]$;
- C_2 è la coppia resistente applicata dalla seconda puleggia $[Nm]$;
- C_3 è la coppia agente sulla vite a rulli $[Nm]$;
- m è la massa traslante $[kg]$;
- F_e è la forza resistente che rappresenta il carico applicato sull'attuatore $[N]$.

Detto ciò, si raggruppano le inerzie per ciascun asse (eq.30-31), si esprime il rapporto di trasmissione della vite τ_v (eq.32), il rapporto di trasmissione del riduttore τ_c (eq.33) e il rendimento della trasmissione a cinghia η_c (eq.34).

- $I_1 = I_p + I_M$ (30)
- $I_2 = I_p + I_{VR}$ (31)

- $\tau_v = \frac{p}{2\pi} = \frac{x}{\vartheta_2} = \frac{\dot{x}}{\dot{\vartheta}_2} = \frac{\ddot{x}}{\ddot{\vartheta}_2}$ (32)
- $\tau_c = \frac{\vartheta_2}{\vartheta_1}$ (33)
- $\eta_c = \frac{C_2\,\vartheta_2}{C_1\vartheta_1},\ C_1 = C_2\frac{\tau_c}{\eta_c}$ (34)

Scrivendo l'equazione di equilibrio per ciascun asse, utilizzando le relazioni precedenti e svolgendo alcuni passaggi si ottiene l'equazione per il calcolo dell'inerzia riportata sull'albero motore (eq.35)

$$C_M - C_1 = I_1\ddot{\vartheta}_1$$

$$C_2 - C_3 = I_2\ddot{\vartheta}_2$$

$$F - F_e = m\ddot{x}\ ,\ F = F_e + m\ddot{x}$$

$$\eta_v = \frac{F\,\dot{x}}{C_2\,\vartheta_2},\ C_3 = F\frac{\dot{x}}{\vartheta_2}\frac{1}{\eta_v} = F\frac{\tau_v}{\eta_v}$$

$$C_2 = I_2\ddot{\vartheta}_2 + F\frac{\tau_v}{\eta_v} = I_2\ddot{\vartheta}_2 + \frac{\tau_v}{\eta_v}(F_e + m\ddot{x}) = \frac{\tau_v}{\eta_v}F_e + \ddot{\vartheta}_2\left(I_2 + \frac{\ddot{x}}{\ddot{\vartheta}_2}\frac{\tau_v}{\eta_v}\right) = \frac{\tau_v}{\eta_v}F_e + \ddot{\vartheta}_2\left(I_2 + \frac{{\tau_v}^2}{\eta_v}m\right)$$

$$C_M = I_1\ddot{\vartheta}_1 + C_2\frac{\tau_c}{\eta_c} = \frac{\tau_c}{\eta_c}\frac{\tau_v}{\eta_v}F_e + \ddot{\vartheta}_2\left(I_2\frac{\tau_c}{\eta_c} + \frac{\tau_c}{\eta_c}\frac{{\tau_v}^2}{\eta_v}m\right) + I_1\ddot{\vartheta}_1$$
$$= \frac{\tau_c}{\eta_c}\frac{\tau_v}{\eta_v}F_e + \ddot{\vartheta}_1\left(I_1 + I_2\frac{{\tau_c}^2}{\eta_c} + \frac{{\tau_c}^2}{\eta_c}\frac{{\tau_v}^2}{\eta_v}m\right)$$

$$I = I_1 + I_2\frac{{\tau_c}^2}{\eta_c} + \frac{{\tau_c}^2}{\eta_c}\frac{{\tau_v}^2}{\eta_v}m \qquad (35)$$

La relazione ottenuta permette di riportare le inerzie dei componenti dell'EMA sull'asse dell'albero motore, ottenendo un'unica espressione di inerzia totale I da inserire nell'equazione di equilibrio alla rotazione (eq.20).

5.2 Modellazione del controllo

Come accennato in precedenza, una sezione fondamentale nella modellazione dell'EMA risulta essere quella del controllo. Per poterlo modellizzare correttamente e in seguito confrontarlo con i valori sperimentali ottenuti risulta essenziale che il controllo modellizzato rispecchi quanto implementato all'interno del Driver.
Di conseguenza, facendo riferimento al capitolo Controllo implementato nel Driver in cui è esposta l'architettura del controllo reale, il modello del controllo è costituito da tre anelli annidati e altrettanti regolatori, la cui tipologia è esposta in Tabella 31.

Anello	Posizione	Velocità	Corrente
Tipo controllo	P	PI	PI

Tabella 31: Tipologie regolatori nel modello dell'EMA.

5.2.1 Regolatore di posizione

Nel blocco regolatore di posizione si individua in ingresso un errore di posizione in metri e in uscita un segnale di velocità di riferimento in metri al secondo. Il regolatore è di tipo proporzionale.

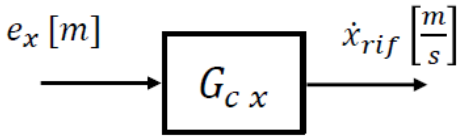


Figura 131: Regolatore anello di posizione.

La funzione di trasferimento che realizza la compensazione P è indicata in (eq.36).

$$(P) \quad G_{C\,x} = kp_x \;\; , \qquad kp_x : \left[\frac{1}{s}\right] \qquad\qquad (36)$$

L'ingresso e l'uscita sono legate algebricamente dal coefficiente kp_x chiamato guadagno proporzionale. Al crescere di questo parametro cresce l'uscita, per cui aumenta la velocità. In un anello in controllo di posizione puramente proporzionale un aumento di tale parametro diminuisce l'errore (differenza tra segnale di Set e di Feedback) ma ci sarà sempre un certo offset tra il segnale di set e quello di feedback, chiamato errore a regime, in quanto il contributo proporzionale man mano che l'errore diminuisce è meno intenso. All'aumento del guadagno esiste però un limite dovuto al fatto che il sistema potrebbe diventare instabile. La risposta in frequenza di un contributo proporzionale è presentato in Figura 132.

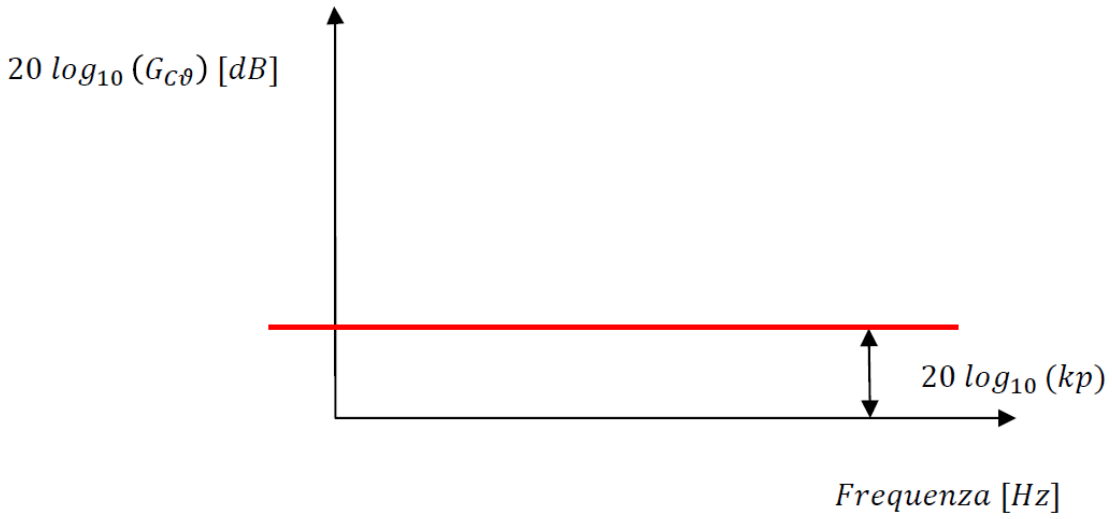


Figura 132: Risposta in frequenza del contributo proporzionale del regolatore di posizione.

5.2.2 Anello di velocità

Nel blocco regolatore di velocità si individua in ingresso un errore di velocità in metri e in uscita un segnale di coppia di riferimento in Newton per metro. Il regolatore è di tipo proporzionale-integrativo.

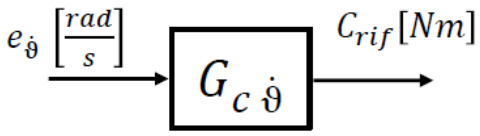


Figura 133: Regolatore anello di velocità.

La funzione di trasferimento che realizza la compensazione PI è indicata in (eq.37).

$$(PI) \quad G_{C\dot{\vartheta}} = kp_{\dot{\vartheta}} + \frac{ki_{\dot{\vartheta}}}{s} = \left(\frac{kp_{\dot{\vartheta}}}{ki_{\dot{\vartheta}}}s + 1\right)\frac{ki_{\dot{\vartheta}}}{s} = (\tau_1 s + 1)\frac{1}{\tau_I s} \qquad (37)$$

$$kp_{\dot{\vartheta}} : \left[\frac{Nm}{\frac{rad}{s}}\right] \quad , \quad ki_{\dot{\vartheta}} : \left[\frac{Nm\ s}{\frac{rad}{s}}\right]$$

In aggiunta al contributo proporzionale, vi è l'azione integrativa, la quale permette di annullare l'errore a regime e di azzerarlo tanto più velocemente quanto più è alto *ki*. Una tale maggiore velocità comporta però una diminuzione della banda passante e delle forti oscillazioni di fronte ad una brusca variazione del segnale di set, con conseguente pericolo di instabilità.
La risposta in frequenza del contributo proporzionale-integrativo è presentata in Figura 134.

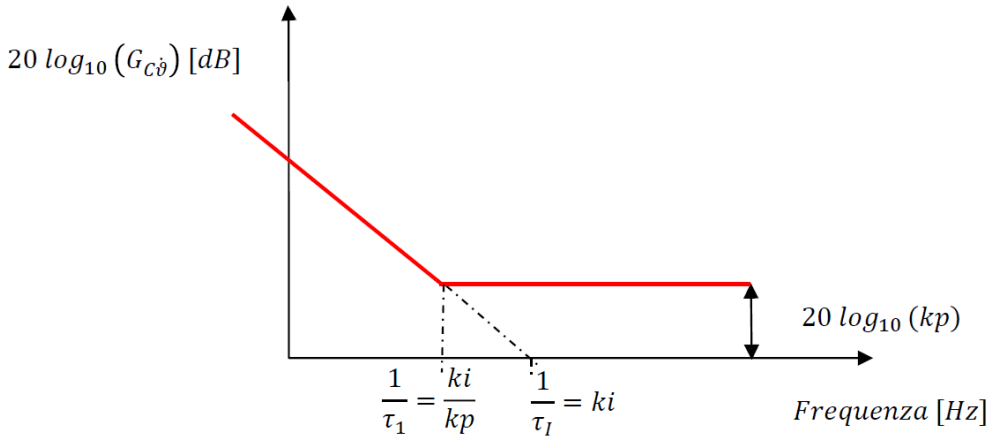
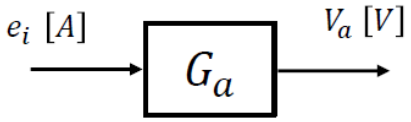


Figura 134: Risposta in frequenza del contributo proporzionale-integrativo del regolatore di velocità.

5.2.3 Anello di corrente

Nel blocco regolatore di corrente si individua in ingresso un errore di corrente in Ampere e in uscita un segnale di tensione di armatura in Volt. Il regolatore è di tipo proporzionale-integrativo.



La funzione di trasferimento che realizza la compensazione PI è indicata in (eq.38).

$$(PI) \quad G_a = \left(\frac{kp_i}{ki_i}s + 1\right)\frac{ki_i}{s} = (\tau_1 s + 1)\frac{1}{\tau_I s} \qquad (38)$$

$$kp_i : \left[\frac{V}{A}\right] \quad , \quad ki_i : \left[\frac{V\,s}{A}\right]$$

La risposta in frequenza del contributo proporzionale-integrativo è presentata in Figura 135.

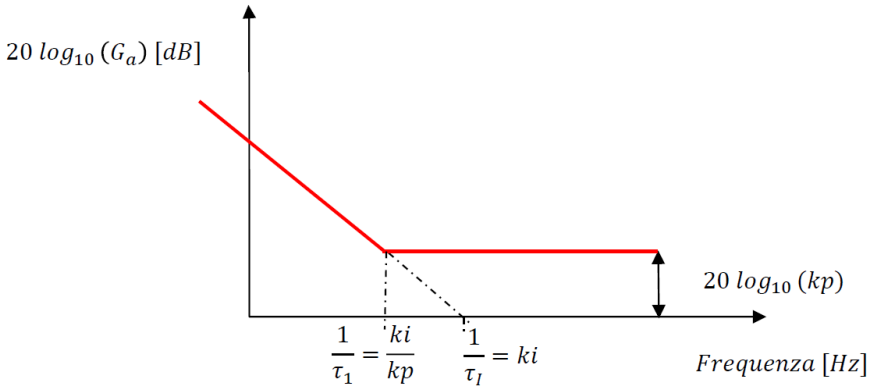


Figura 135: Risposta in frequenza del contributo proporzionale-integrativo del regolatore di corrente.

5.2.4 Ritardo dell'elettronica

Un controllo di tipo digitale presenta un ritardo aggiuntivo tra variabile di riferimento e variabile di processo causato dal tempo di campionamento e dal tempo di calcolo del microprocessore, definito come ritardo dell'elettronica. Nel modello ciò si traduce in un blocco di ritardo di trasporto, il quale introduce uno sfasamento tra la variabile compensata ed il riferimento in tensione nell'anello.
La funzione di trasferimento di ritardo dell'elettronica è presentata in (eq.39) dove il parametro τ è il tempo di ritardo, definito dalla (eq.40).

$$D_{(s)} = e^{-s\tau} \qquad (39)$$

$$\tau = \frac{T_S}{2} + T_C \qquad (40)$$

- T_S è il tempo di campionamento;
- T_C è il tempo di calcolo del microprocessore.

Analizzando la risposta armonica (eq.41) si ottiene la risposta in frequenza da riportare come ampiezza (eq.42) e fase (eq.43) sul diagramma di bode (Figura 136).

$$D(j\omega) = e^{-j\omega\tau} = M(\omega)\,e^{j\varphi(\omega)} \qquad (41)$$

$$M(\omega) = 1 \;,\; M_{dB}(\omega) = 0 \qquad (42)$$

$$\varphi(\omega) = \omega\tau \qquad (43)$$

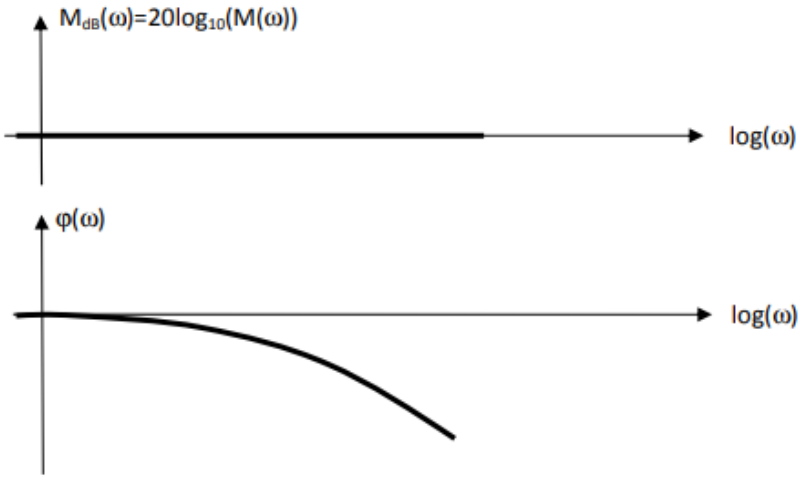


Figura 136: Diagramma di bode per il ritardo dell'elettronica.

Il ritardo puro introduce solamente uno sfasamento, mentre non altera l'ampiezza di risposta del segnale.
Nei capitoli successivi si ingloba il ritardo di trasporto nel blocco del regolatore di corrente G_a (Figura 137).

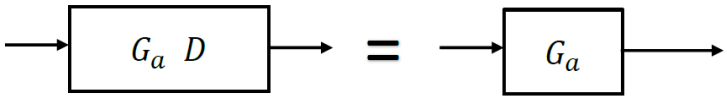


Figura 137: Blocco regolatore di corrente + ritardo di trasporto.

5.3 Diagrammi a blocchi del modello

5.3.1 Azionamento e motore elettrico

L'anello più interno presente nel controllo del Driver è un anello di corrente. In un motore elettrico DC semplicemente regolato in velocità non si riesce ad ottenere una coppia solamente proporzionale alla tensione di alimentazione, a causa della presenza della forza contro elettromotrice $k_e\overline{\dot{\vartheta}_M}$.
Inserendo al suo interno un anello di regolazione di corrente con un elevato valore di guadagno, la corrente risulta quasi indipendente dalla velocità angolare e quindi anche la coppia fornita dal motore, ottenendo un preciso e veloce controllo della coppia (Jacazio, 'Regolazione e servomeccanismi'). Inoltre l'aggiunta di un anello annidato ad altri favorisce la stabilità del sistema, evitando la propagazione di errori sugli anelli più esterni.
Considerando per la modellazione un motore DC a magneti permanenti, lo schema meccatronico dell'azionamento e del motore elettrico può essere riassunto dalla Figura 138.

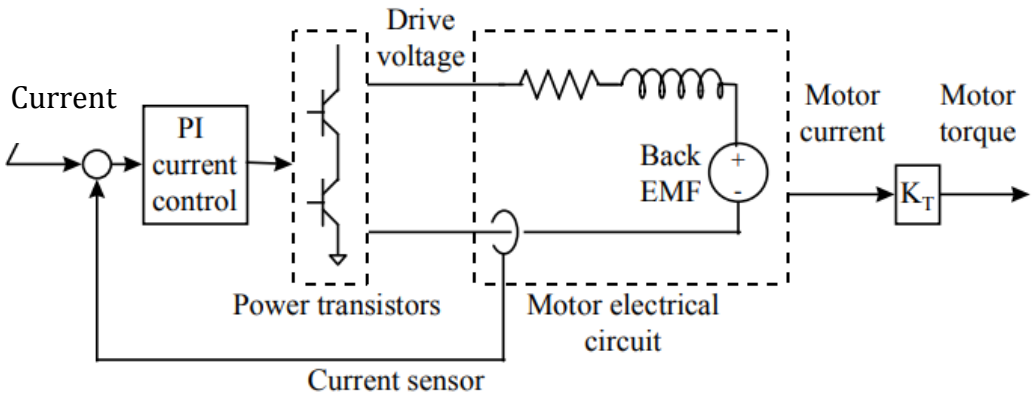


Figura 138: Schema meccatronico dell'azionamento e del motore elettrico.

Tale schema, facendo riferimento alle equazioni scritte in precedenza, si traduce nel diagramma a blocchi del modello per l'azionamento elettrico ed il motore (Figura 139).

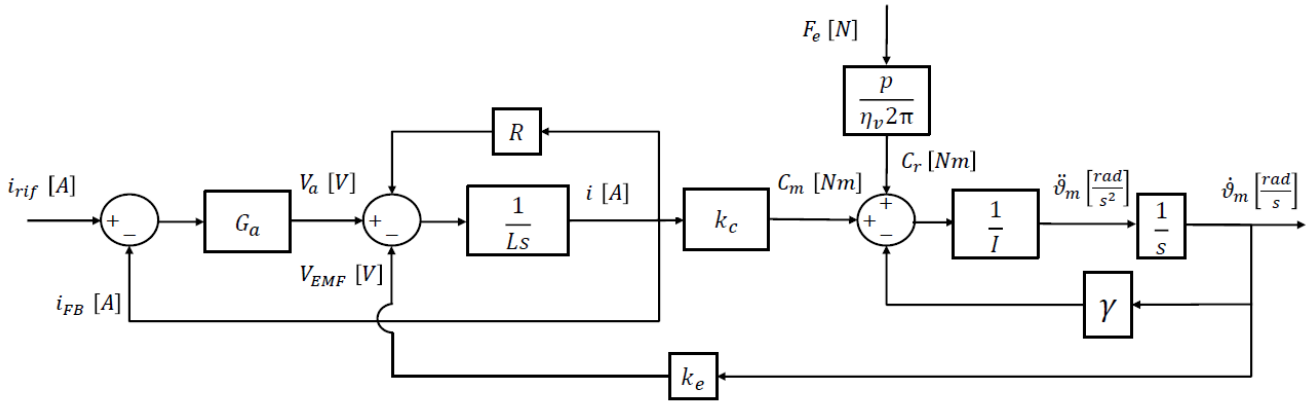


Figura 139: Diagramma a blocchi del modello Driver e motore elettrico.

La formulazione della corrente di armatura in ingresso al motore è espressa in (eq.44).

$$\bar{\imath} = \frac{G_a \overline{\imath_{rif}} - k_e \overline{\dot{\vartheta}_M}}{R + G_a + Ls} \qquad (44)$$

Elaborando tale espressione si nota come in presenza di un elevato valore di guadagno del regolatore, il contributo di velocità tende a zero per cui la corrente risulta quasi indipendente dalla velocità.
Considerando una forza resistente nulla sull'attuatore e semplificando il diagramma si ottiene la sua forma semplificata.
Si effettua un ulteriore passaggio per inglobare alcuni parametri in due blocchi per facilitare la comprensione dove si raggruppano i parametri attraverso i le seguenti diciture (eq.45-eq.46).

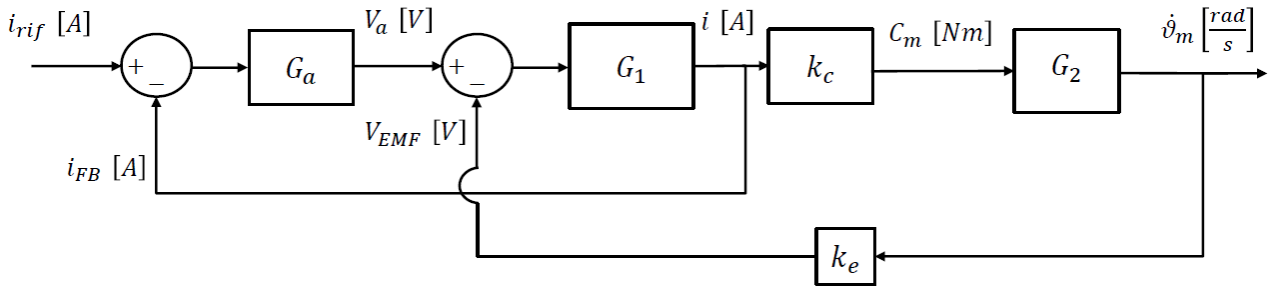


Figura 140: Semplificazioni diagramma a blocchi 1.

$$G_1 = \frac{1}{R}\frac{1}{\frac{L}{R}s + 1} \qquad (45)$$

$$G_2 = \frac{1}{\gamma}\frac{1}{\frac{I}{\gamma}s + 1} \qquad (46)$$

A questo punto si possono sfruttare le proprietà dei diagrammi a blocchi per semplificare il diagramma.

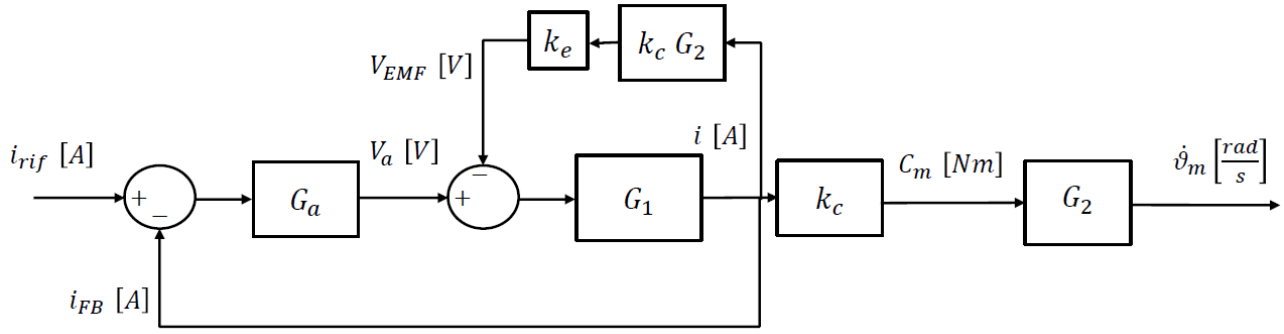


Figura 141: Semplificazioni diagramma a blocchi 2.

$$G_3 = \frac{G_1}{1 + K_c\, G_2 G_1 K_e} \qquad (47)$$

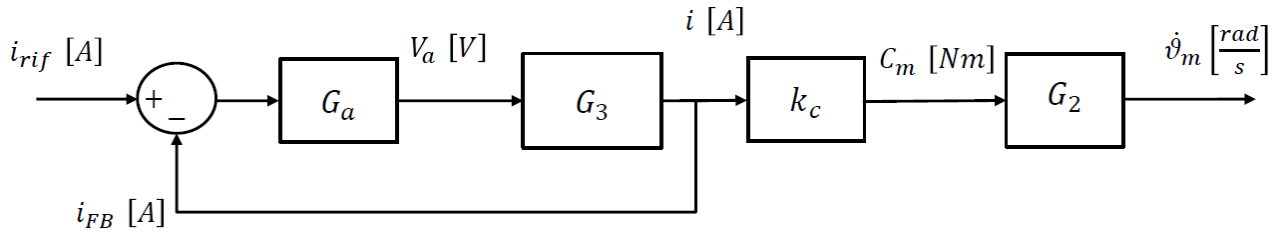


Figura 142: Semplificazioni diagramma a blocchi 3.

Sostituendo le variabili ottenute (eq.47) ed dopo alcuni passaggi si ottengono la funzione di trasferimento in anello aperto e chiuso del loop di corrente $G_{CL\,i}$ (eq.48-49).

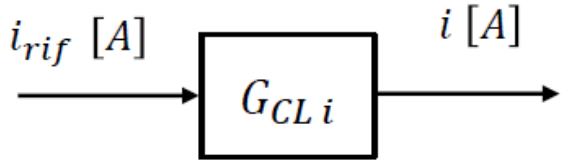


Figura 143: Diagramma a blocchi della funzione di trasferimento dell'anello chiuso di corrente.

$$G_{OL\,i} = \frac{i_{FB}}{e} = \frac{G_a\,(Is+\gamma)}{(Is+\gamma)(Ls+R)+k_e\,k_c} = \frac{k_{I\,i}\;\gamma}{s\,(\gamma R+k_e k_c)}\;\frac{\left(\frac{k_{p\,i}}{k_{I\,i}}s+1\right)\left(\frac{I}{\gamma}s+1\right)}{s^2\,\frac{IL}{\gamma R+k_e k_c}+s\,\frac{IR+L\gamma}{\gamma R+k_e k_c}+1} \quad (48)$$

$$G_{CL\,i} = \frac{i}{i_{rif}} = \frac{G_a\,(Is+\gamma)}{(Is+\gamma)(Ls+R)+k_e\,k_c+G_a\,(Is+\gamma)} = \frac{\left(\frac{k_{p\,i}}{k_{I\,i}}s+1\right)\left(\frac{I}{\gamma}s+1\right)\,k_{I\,i}\;\gamma}{(s^2\,\frac{IL}{\gamma R+k_e k_c}+s\,\frac{IR+L\gamma}{\gamma R+k_e k_c}+1)\,(\gamma R+k_e k_c)+\left(\frac{k_{p\,i}}{k_{I\,i}}s+1\right)\left(\frac{I}{\gamma}s+1\right)\,k_{I\,i}\;\gamma} \quad (49)$$

La seconda parte delle funzioni di trasferimento evidenzia i poli del sistema: dalla funzione *open loop* si ottengono due zeri finiti, due poli finiti ed un polo nell'origine, da cui deriva un eccesso di poli su zeri pari ad uno. Di conseguenza, si può intuire l'andamento in risposta di frequenza della funzione: per frequenze tendenti a $\pm\infty$ la funzione avrà un inclinazione di $\pm$20dB ogni decade di frequenza. Riguardo alla funzione *Closed Loop*, questa presenterà lo stesso numero poli e zeri ma questi risulteranno diversi dalla funzione *Open Loop* e soprattutto non vi sarà più il polo nell'origine.

5.3.2 Anello di velocità

Un anello in regolazione di velocità è posizionato all'esterno del *loop* di corrente ed è modellato seguendo il metodo introdotto nel Driver (Figura 144).

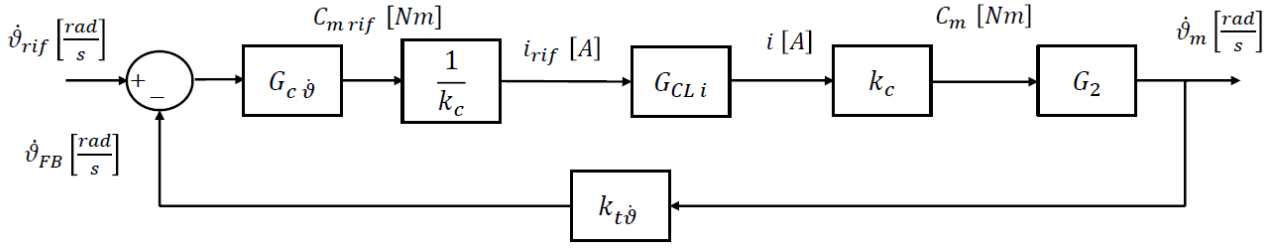


Figura 144: Diagramma a blocchi del modello: anello di velocità.

Semplificando il diagramma a blocchi e sostituendo le variabili si ottengono le funzioni di trasferimento in anello aperto e chiuso in forma estesa del loop di velocità (eq.50-51).

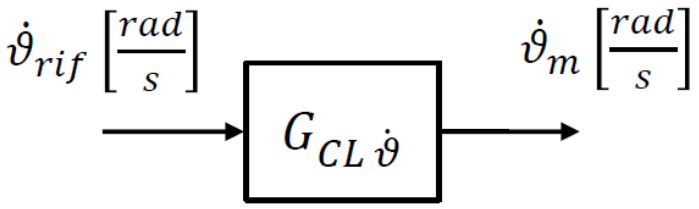


Figura 145: Diagramma a blocchi della funzione di trasferimento in anello chiuso di velocità.

$$G_{OL\,\dot{\vartheta}} = \frac{\dot{\vartheta}_{FB}}{e} = \frac{G_{c\dot{\vartheta}}\, G_a\, k_{t\dot{\vartheta}}}{(Is+\gamma)(Ls+R+\ G_a\ ) + k_e\, k_c} \qquad (50)$$

$$G_{CL\,\dot{\vartheta}} = \frac{\dot{\vartheta}_m}{\dot{\vartheta}_{rif}}$$

$$= \frac{G_{c\dot{\vartheta}}\, G_a}{(Is+\gamma)(Ls+R+G_a\ ) + k_e\, k_c} \frac{(Is+\gamma)^2(Ls+R+G_a\ ) + (Is+\gamma)k_e\, k_c}{(Is+\gamma)^2(Ls+R+G_a\ ) + (Is+\gamma)k_e\, k_c + G_{c\dot{\vartheta}}\, G_a\, k_{t\dot{\vartheta}}} \qquad (51)$$

5.3.3 Anello di posizione

Il *loop* più esterno consiste in un anello in regolazione di posizione ed è modellato seguendo quanto implementato nel Driver (Figura 146).

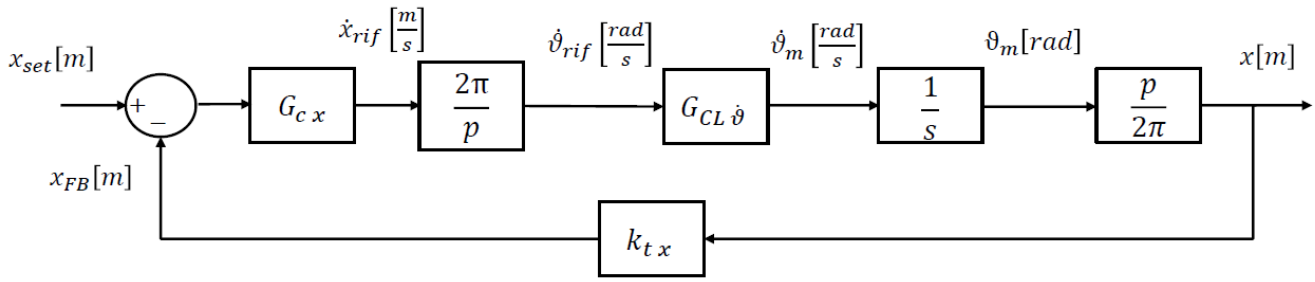


Figura 146: Diagramma a blocchi del modello: anello di posizione.

Semplificando il diagramma a blocchi e sostituendo le variabili si ottengono le funzioni di trasferimento in anello aperto e chiuso in forma estesa del *loop* di posizione.

$$G_{OL\,x} = \frac{x_{FB}}{e} =$$

$$= \frac{G_{c\dot{\vartheta}}\, G_a}{(Is+\gamma)(Ls+R+G_a\ ) + k_e\, k_c} \frac{(Is+\gamma)^2(Ls+R+G_a\ ) + (Is+\gamma)k_e\, k_c}{(Is+\gamma)^2(Ls+R+G_a\ ) + (Is+\gamma)k_e\, k_c + G_{c\dot{\vartheta}}\, G_a\, k_{t\dot{\vartheta}}}\ G_{c\,x}\, \frac{1}{s}\, k_{t\,x}$$

$$G_{CL\,x} = \frac{x}{x_{set}} =$$

$$\frac{[G_{c\dot{\vartheta}}\, G_a]\,[(Is+\gamma)^2(Ls+R+G_a\ ) + (Is+\gamma)k_e\, k_c]\, G_{c\,x}}{[(Is+\gamma)(Ls+R+G_a\ ) + k_e\, k_c]\,[(Is+\gamma)^2(Ls+R+G_a\ ) + (Is+\gamma)k_e\, k_c + G_{c\dot{\vartheta}}\, G_a\, k_{t\dot{\vartheta}}]\, s + [G_{c\dot{\vartheta}}\, G_a]\,[(Is+\gamma)^2(Ls+R+G_a\ ) + (Is+\gamma)k_e\, k_c]\, G_{c\,x}\ k_{t\,x}}$$

Il diagramma a blocchi del modello completo in forma estesa è presentato in Figura 147 ed i parametri utilizzati sono elencati in Tabella 32.

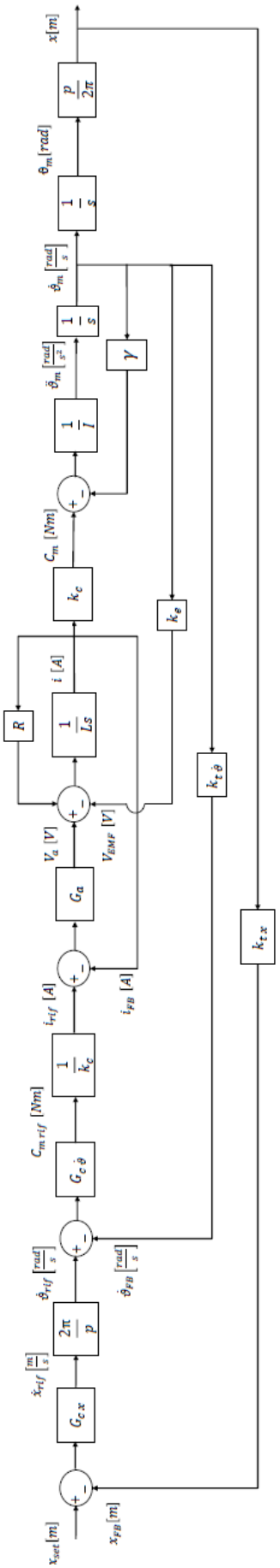


Figura 147: Diagramma a blocchi del modello EMA in forma estesa.

Parametro	Descrizione	Valore
$\boldsymbol{p}$	*Passo della vite*	$0.005\ m$
$\boldsymbol{k_e}$	*Costante di tensione del motore*	$1.31\ \frac{V}{rad/s}$
$\boldsymbol{k_c}$	*Costante di coppia del motore*	$2.34\ \frac{Nm}{A}$
$\boldsymbol{L}$	*Induttanza*	0.0522 H
$\boldsymbol{I}$	*Inerzia totale riportata sul motore*	$6.29 \cdot 10^{-4}\ kg\ m^2$
$\boldsymbol{R}$	*Resistenza*	5.87 Ω
$\boldsymbol{\gamma}$	*Coefficiente di attrito viscoso*	$0.08 \frac{Nm}{rad/s}$
$\boldsymbol{k_{t\,x}}$	*Guadagno del ramo posizione di retroazione*	$1 \frac{m}{m}$
$\boldsymbol{k_{t\,\dot{\vartheta}}}$	*Guadagno del ramo velocità di retroazione*	$1\ \frac{rad/s}{rad/s}$

Tabella 32: Parametri utilizzati nel modello: valore e descrizione.

5.4 Analisi in risposta del modello

Ottenute le funzioni di trasferimento del modello nel precedente capitolo, si possono implementare in un codice Matlab, appositamente studiato per poter ottenere le curve di risposta in frequenza, risposta ad un segnale a rampa e risposta ad un gradino.
I guadagni dei regolatori utilizzati per la simulazione sono quelli utilizzati nel Driver, corretti con l'unità di misura opportuna al modello (Tabella 33).

Guadagno	Descrizione	Valore
$\boldsymbol{kp_i}$	*Guadagno proporzionale del regolatore corrente*	$107.5\ \frac{V}{A}$
$\boldsymbol{k_{I\,i}}$	*Guadagno integrativo del regolatore corrente*	$34677\ \frac{V\,s}{A}$
$\boldsymbol{kp_{\dot{\vartheta}}}$	*Guadagno proporzionale del regolatore velocità*	$0.111 \frac{Nm}{rad/s}$
$\boldsymbol{k_{I\,\dot{\vartheta}}}$	*Guadagno integrativo del regolatore velocità*	$7.723\ \frac{Nm\,s}{rad/s}$
$\boldsymbol{kp_x}$	*Guadagno proporzionale del regolatore posizione*	$60\ \frac{1}{s}$

Tabella 33: Guadagni dei regolatori per il modello EMA.

Il modello deve rispondere a delle specifiche che sono la stabilità, l'accuratezza a regime e l'insensibilità ai disturbi.
L'insensibilità ai disturbi nel sistema in controllo di posizione è garantita dal controllo retroazionato che per definizione è insensibile ai disturbi. L'accuratezza a regime consiste nell'ottenere un errore statico di posizionamento nullo in seguito ad un transitorio ed è garantita dalla parte integrativa nel regolatore di velocità.
La stabilità è la specifica più interessante nel progetto, poiché se un sistema è instabile può danneggiarsi e provocare danni all'esterno. Un sistema è stabile se è possibile governarlo: in un

sistema di controllo in anello chiuso l'errore è la grandezza fisica che permette di far convergere la variabile di processo (*Feedback*) con la variabile di riferimento (*Set*). Nel caso in cui tra errore e Feedback vi sia uno sfasamento maggiore di 180°, il sistema risulta instabile, ovvero sono in controfase. Di conseguenza è fondamentale restare distanti dalle condizioni di instabilità, per cui devono essere rispettati dei margini di fase e margini di guadagno nello studio della risposta in frequenza in anello aperto del modello. Il margine di guadagno deve essere maggiore di $|-7 \div 9\,|dB$, mentre il margine di fase deve essere maggiore di $|-40 \div 45\,|°$.
Ulteriori specifiche che è necessario rispettare in un sistema di controllo sono la velocità di risposta del sistema, la precisione dinamica e la banda passante negli anelli del modello. Secondo i testi di riferimento, le bande passanti da rispettare in un modello di questo tipo sono:

- Anello di corrente : $Banda\ passante\ [Hz] \geq \frac{1}{10} f_s = 250\ Hz$;
- Anello di velocità : $10\ Hz \leq Banda\ passante\ [Hz] \leq \frac{1}{10} f_s = 250\ Hz$
- Anello di posizione : $Banda\ passante\ [Hz] \cong 10\ Hz$

Dove f_s è la frequenza di campionamento. Ciò significa che gli anelli più interni debbano avere una dinamica migliore di quelli esterni, ovvero una banda passante maggiore. Detto questo, il modello è realizzato rispettando tali specifiche.
Lo studio parte dagli anelli interni per poi passare a quelli esterni, poiché la modifica di un anello interno si ripercuote su quelli all'esterno mentre una modifica di quelli esterni non comporta una variazione in quelli interni.
Per riassumere si presenta il diagramma a blocchi che evidenzia i tre anelli per cui sono state formulate le tre coppie di funzioni di trasferimento.

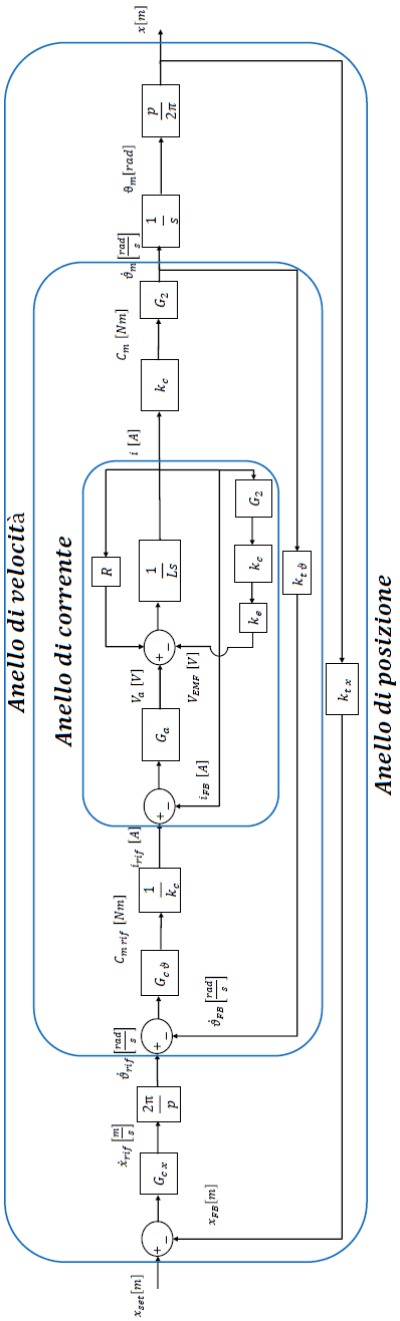


Figura 148: Diagramma a blocchi: anelli delle funzioni di trasferimento.

Per cui si presentano in sequenza le curve di risposta in frequenza partendo dall'anello di corrente, passando per l'anello di velocità ed infine per l'anello di posizione.

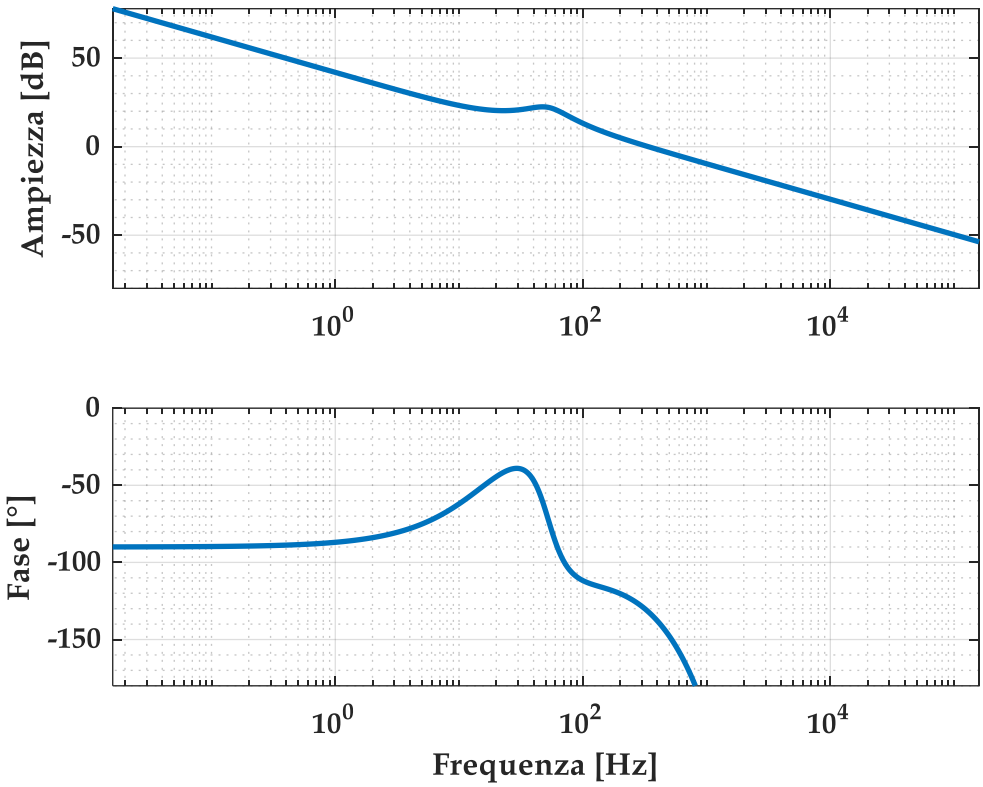


Figura 149: Risposta in frequenza Open Loop dell'anello di corrente.

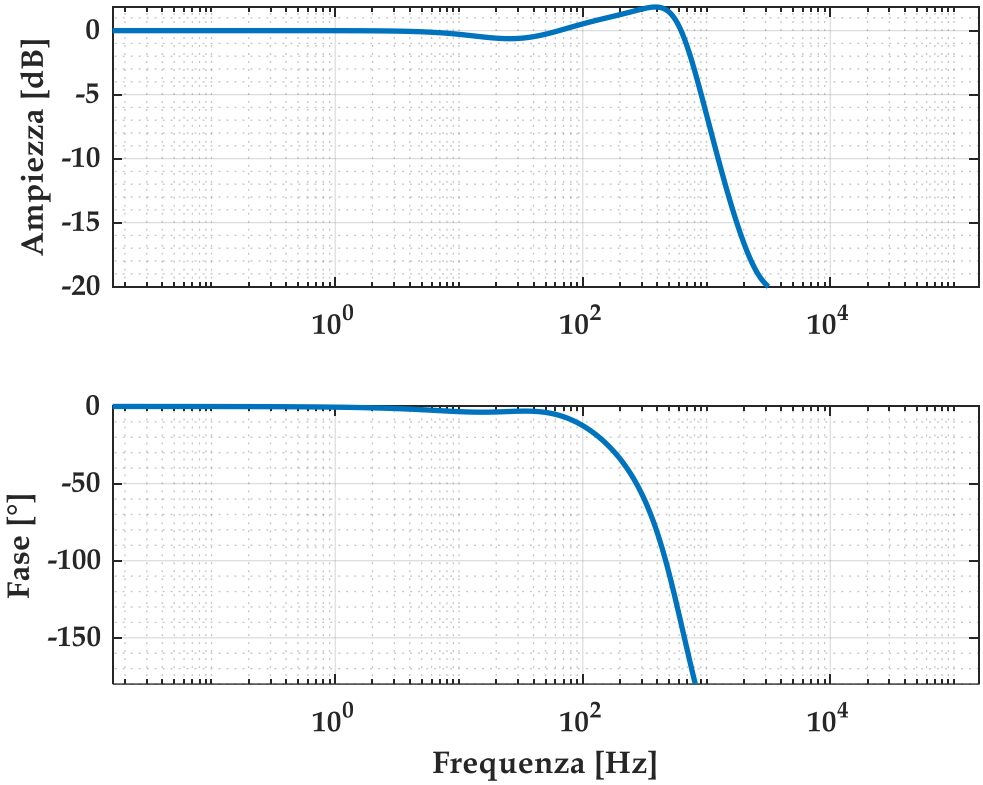


Figura 150: Risposta in frequenza Closed Loop dell'anello di corrente

I margini di fase e di guadagno risultano rispettati (Figura 149), così come la larghezza di banda dell'anello di corrente ($\cong 700\ Hz$) (Figura 150).

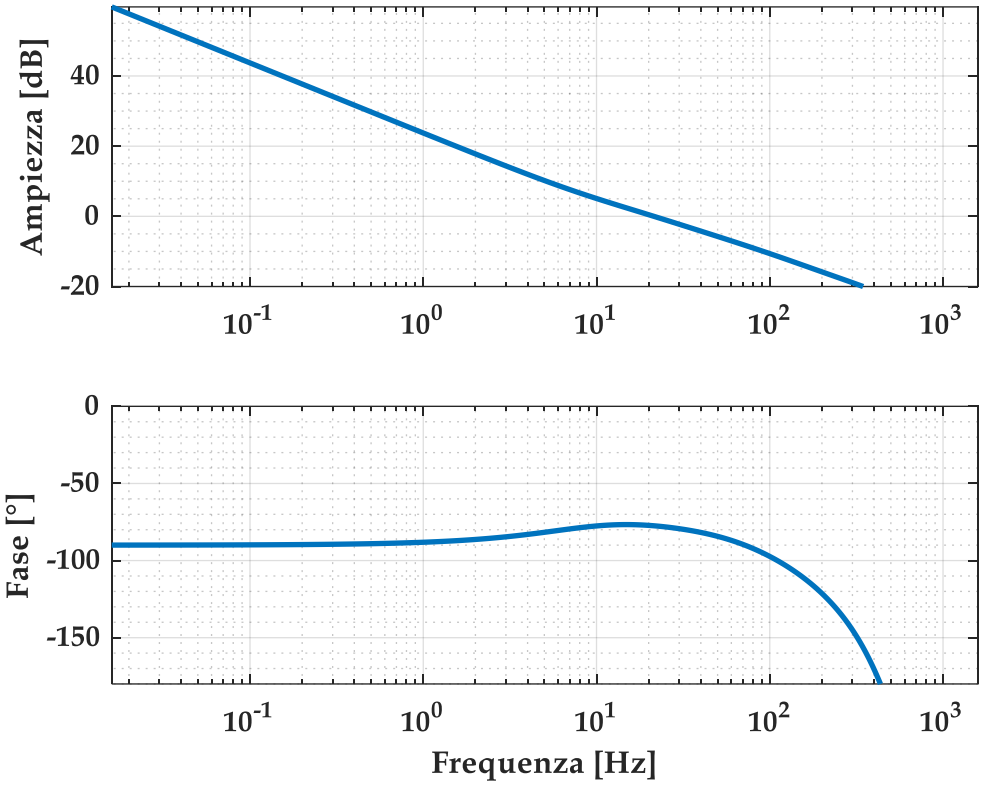


Figura 151: Risposta in frequenza Open Loop dell'anello di velocità.

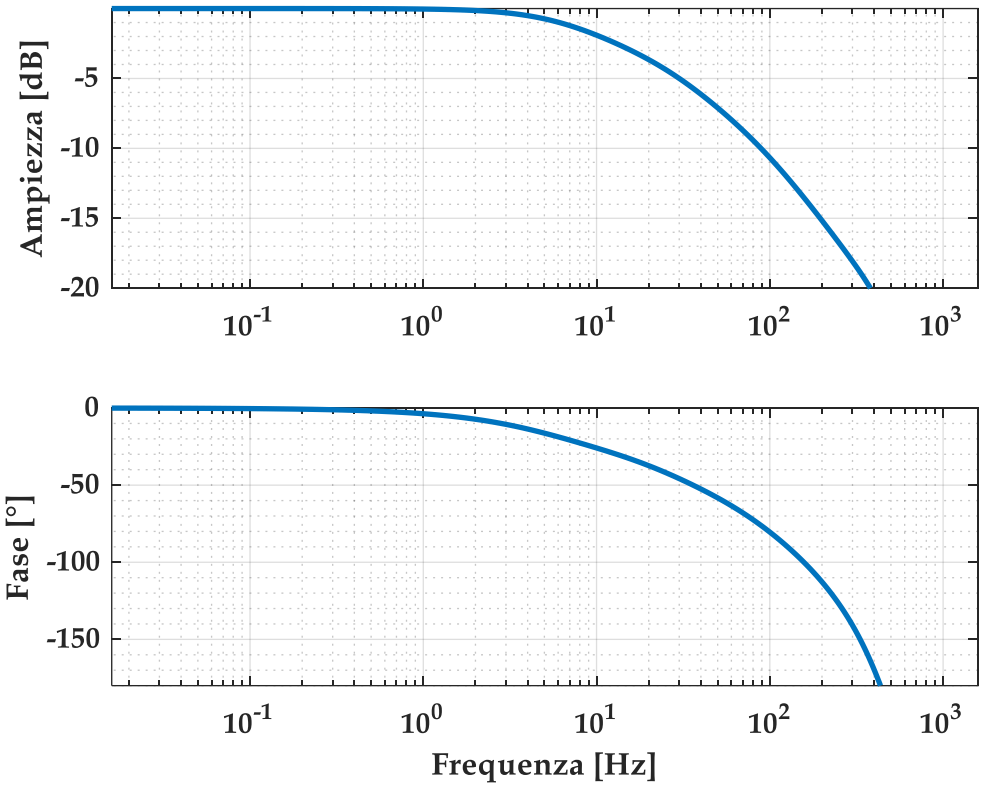


Figura 152: Risposta in frequenza Closed Loop dell'anello di velocità.

I margini di fase e di guadagno risultano rispettati (Figura 151), così come la larghezza di banda dell'anello di velocità ($\cong 16\ Hz$) (Figura 152).

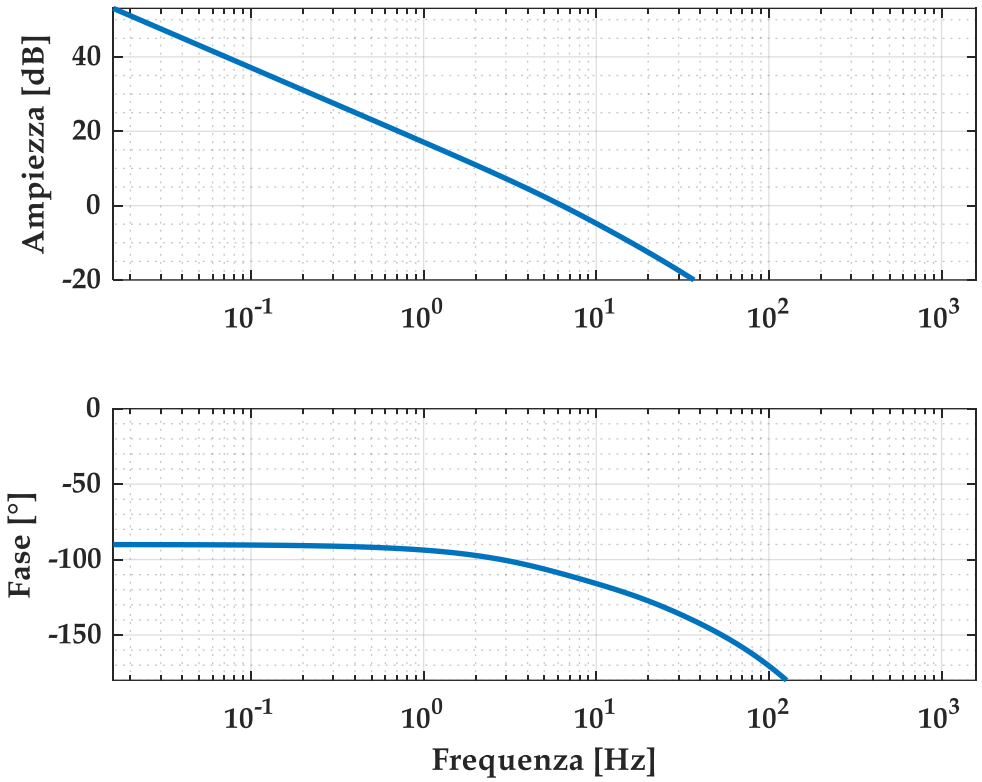


Figura 153: Risposta in frequenza Open Loop dell'anello di posizione.

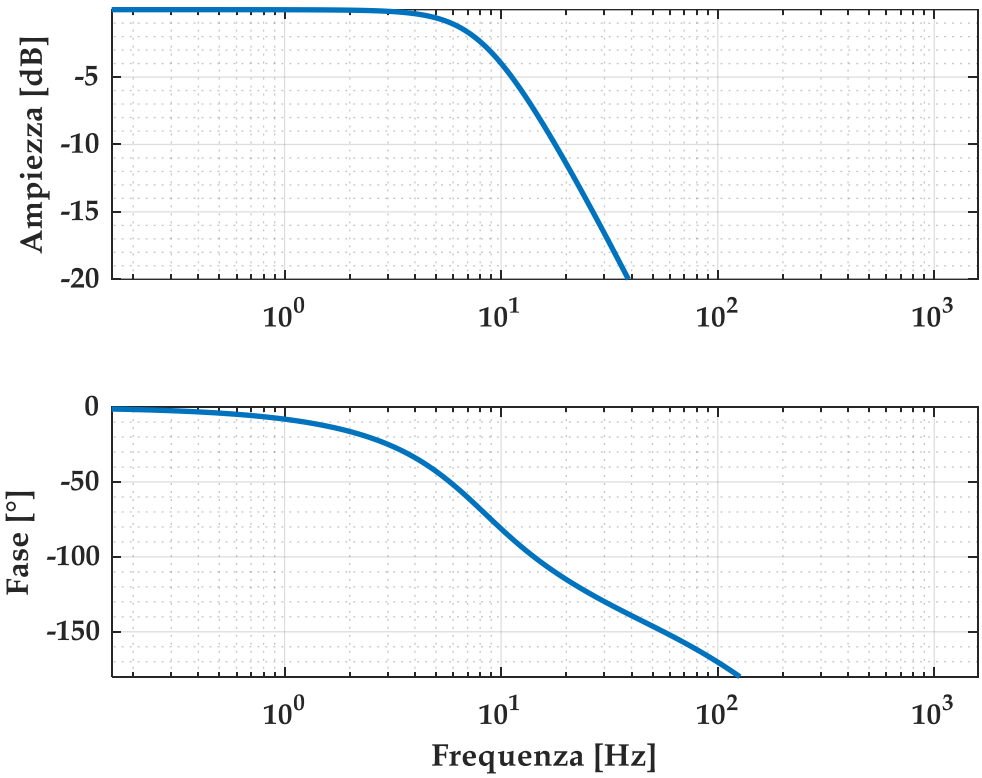


Figura 154: Risposta in frequenza Closed Loop dell'anello di posizione.

I margini di fase e di guadagno risultano rispettati (Figura 153), così come la larghezza di banda dell'anello di posizione ($\cong 8\ Hz$) (Figura 154).
Considerando la risposta ad un comando a gradino ed a rampa, il sistema risponde con una precisione, sia statica che dinamica, opportuna (Figura 155 e Figura 156).

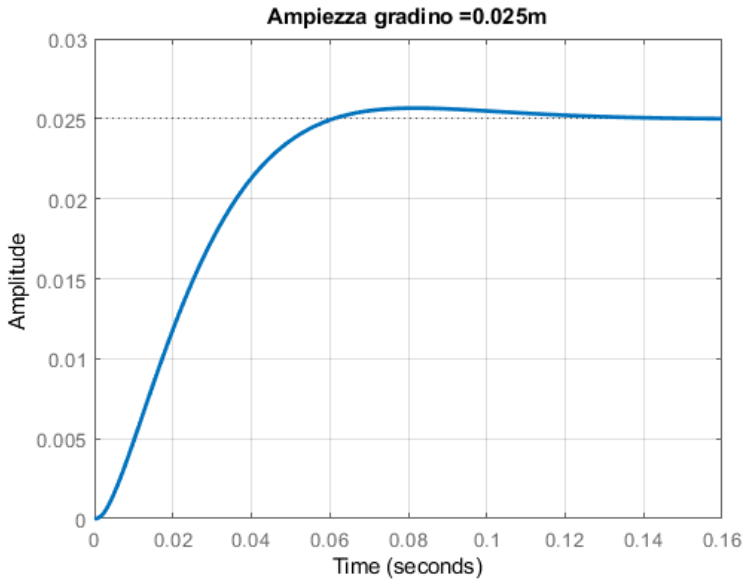


Figura 155: Risposta del modello ad un comando a gradino.

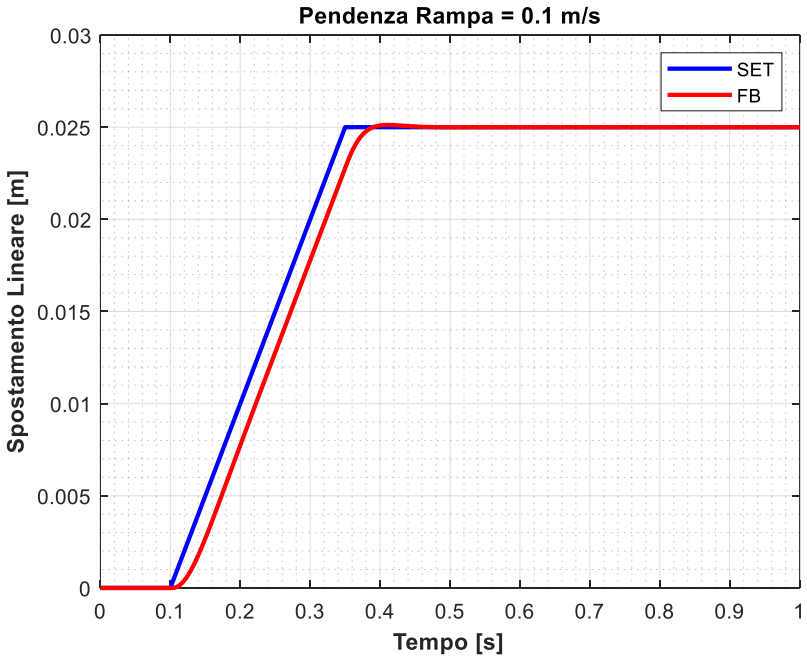


Figura 156: Risposta del modello ad un comando a rampa.

Infine, facendo riferimento al controllo implementato nel Driver, si valuta il modello lineare nel caso di inserimento del *Feedforward* di coppia (Figura 157).

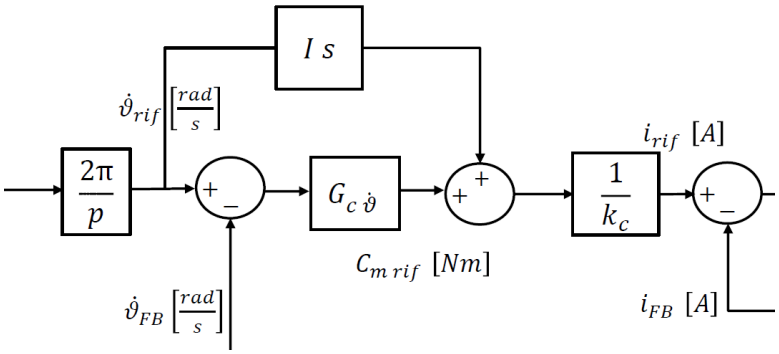


Figura 157: Diagramma a blocchi del Modello lineare nel caso di inserimento del Feedforward di coppia.

Per valutare l'effetto di tale aggiunta si studia la risposta ad un comando di rampa (Figura 158). Come ci si aspetterebbe, il *Feedforward* permette di anticipare il riferimento di coppia rendendo la risposta più pronta ma nel caso in esame non altera di molto le caratteristiche di risposta del modello (dettaglio Figura 158).

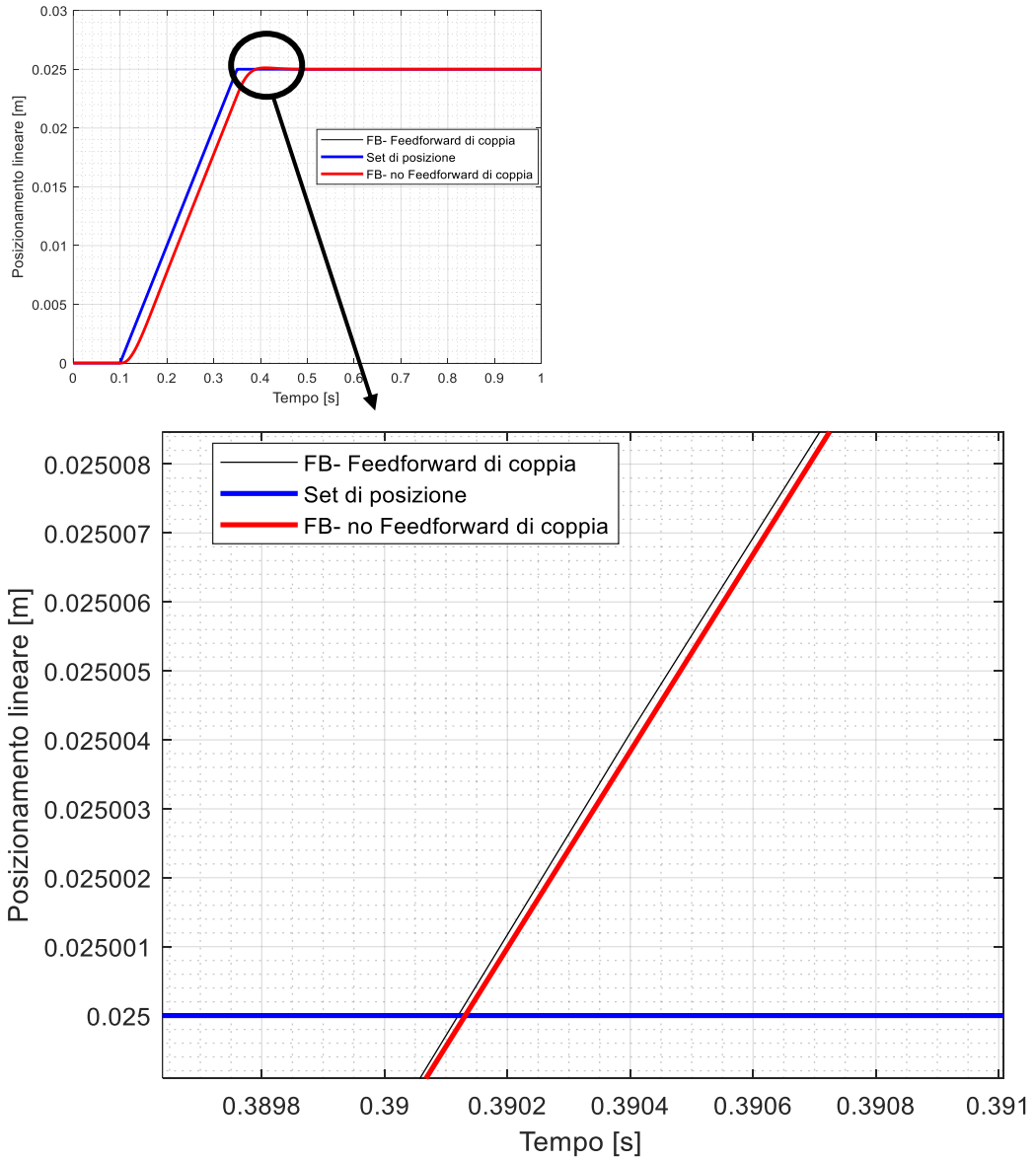


Figura 158: Effetto del Feedforward di coppia nel modello lineare in un segnale a rampa.

6. Verifiche sperimentali

Grazie alle applicazioni progettate e illustrate nei capitoli precedenti (Software *Lenze Engineer* e Software LabVIEW), è possibile testare l'EMA e quindi ottenere le acquisizioni per diversi segnali di comando scelti dall'utente. I file '*.txt'* ottenuti tramite l'applicazione sono manipolati tramite lo script MATLAB in appendice.
È importante presentare in modo dettagliato i parametri inseriti nel driver al momento delle acquisizioni, per cui si discuteranno i valori dei guadagni dei controllori, delle saturazioni e dei filtri.
Successivamente si analizzano le tipologie di prove effettuate e si presentano i risultati ottenuti.

6.1 Parametri del diagramma a blocchi del Driver

Seguendo quanto riportato nel capitolo Controllo implementato nel Driver, si riassume il diagramma a blocchi del controllo implementato (Figura 159). Il controllo è ovviamente non lineare, poiché vi sono saturazioni e filtri. Come nel modello, presenta un'architettura a tre anelli di controllo annidati, i cui regolatori sono analoghi ai primi ma con il valore dei guadagni scalati per l'unità di misura utilizzata.
Sempre come nel modello, entra una variabile di riferimento che è la posizione lineare dell'attuatore ed esce una variabile di processo che è la corrente che va al blocco motore.

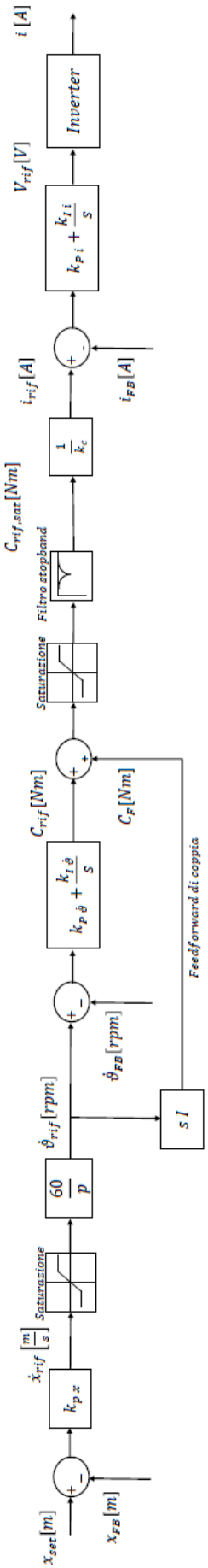


Figura 159: Diagramma a blocchi nel Driver riassuntivo.

Il regolatore di posizione è un proporzionale $k_{p\,x}$ in cui l'utente inserisce il valore desiderato.

$$k_{p\,x}\ \left[\frac{1}{s}\right]$$

In uscita dal controllo posizione si ottiene la velocità di riferimento, la quale è saturata attraverso un blocco di saturazione, dove il limite inferiore e superiore sono impostati dall'utente; si è scelto di saturare la velocità al valore di targa indicata dal motore, per cui (eq.52)

$$1950\ rpm \equiv 162.5 \frac{mm}{s} \qquad (52)$$

In seguito l'errore di velocità è compensato da un regolatore PI i cui guadagni sono:

$$k_{p\,\dot{\vartheta}}\ \left[\frac{Nm}{rpm}\right]$$

$$k_{I\,\dot{\vartheta}}\ \left[\frac{Nm}{rpm}\ s\right]$$

In questo modo si ottiene un riferimento di coppia, il quale è sommato ad un *Feedforward* di coppia che permette di ottenere una risposta più pronta ad un riferimento di velocità, dove:

$$I : inerzia\ globale\ riportata\ sull'albero\ motore = 6.29 \cdot 10^{-4}\ [kg\ m^2]$$

La coppia di riferimento ottenuta è saturata al valore di targa di coppia del motore e poi filtrata da due filtri *bandstop*, impostati con una profondità pari a zero dB.
Per ottenere la corrente di riferimento si divide la coppia di riferimento per la costante di coppia k_c.

$$k_c = 2.34\ \left[\frac{Nm}{A}\right]$$

L'errore di corrente è compensato da un regolatore PI i cui guadagni $k_{p\,i}$ *e* $k_{I\,i}$, secondo quanto riportato dalla guida, risultano automaticamente ottimizzati una volta che sono inseriti i parametri principali del motore durante la definizione dei componenti costituenti l'albero di progetto:

$$k_{p\,i}\ \left[\frac{V}{A}\right]$$

$$k_{I\,i}\ \left[\frac{V}{A}\ s\right]$$

Il Driver, dopo aver caricato i dati di targa del motore e altri parametri durante la definizione dell'albero di progetto, oltre ai guadagni del regolatore corrente PI, inserisce dei guadagni preimpostati per il regolatore velocità PI e per il regolatore posizione P.
I parametri preimpostati risultano:

- Regolatore posizione:

$$k_p = 20\frac{1}{s}$$

- Regolatore velocità:

$$k_p = 0.01164\ \frac{Nm}{rpm}$$

$$k_I = 0.808\ \frac{Nm}{rpm}\ s$$

$$k_d = 0\ \frac{Nm}{rpm\ s}$$

- Regolatore corrente:

$$k_p = 107.5\ \frac{V}{A}$$

$$k_I = 34667\ \frac{V}{A}\ s$$

Si svolgono le acquisizioni per un segnale sinusoidale e per un segnale a rampa, rilevando Set, Feedback di posizione e Feedback di velocità.

6.2 Prove con guadagni preimpostati

Le prove effettuate per valutare i guadagni preimpostati nel controllo del Driver sono presentati in Tabella 34 e i risultati sono esposti in Figura 160, Figura 161, Figura 162.

Sinusoidale		
Prova	*Ampiezza* $[m]$	*Frequenza* $[Hz]$
1	0.01	1
2	0.02	1

Rampa		
Prova	*Valore finale* $[m]$	*Pendenza* $[m/s]$
1	0.015	0.09

Tabella 34: Prove effettuate per i comandi sinusoidale e a rampa

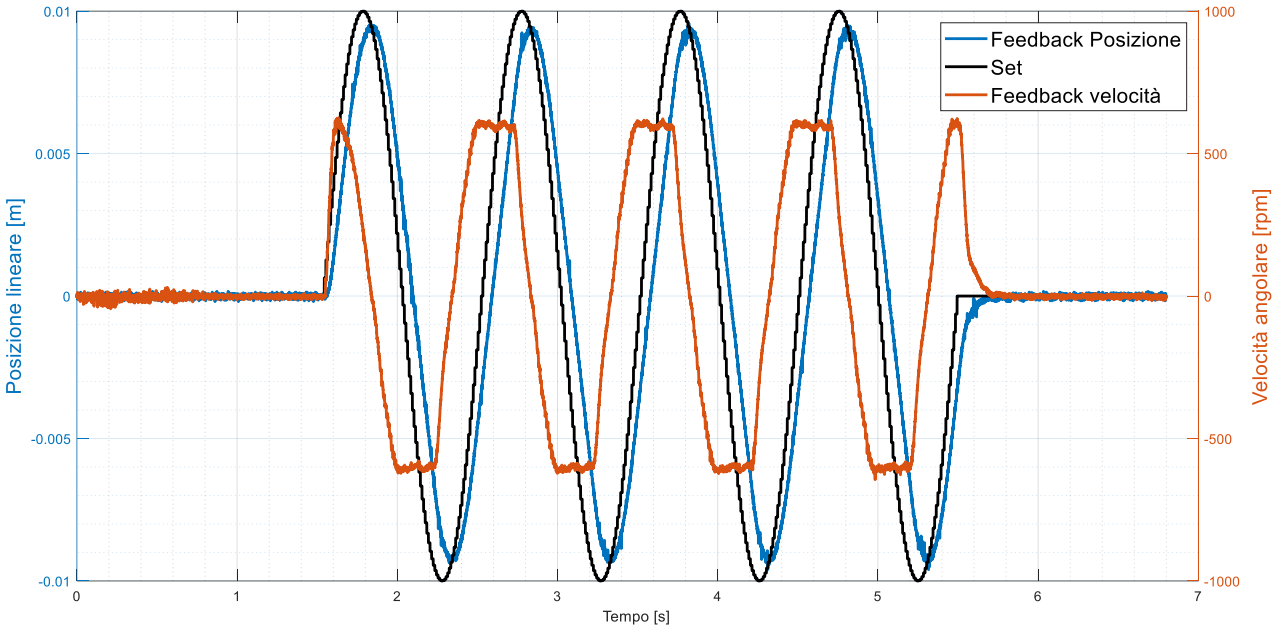


Figura 160: Segnale di comando sinusoidale 1.

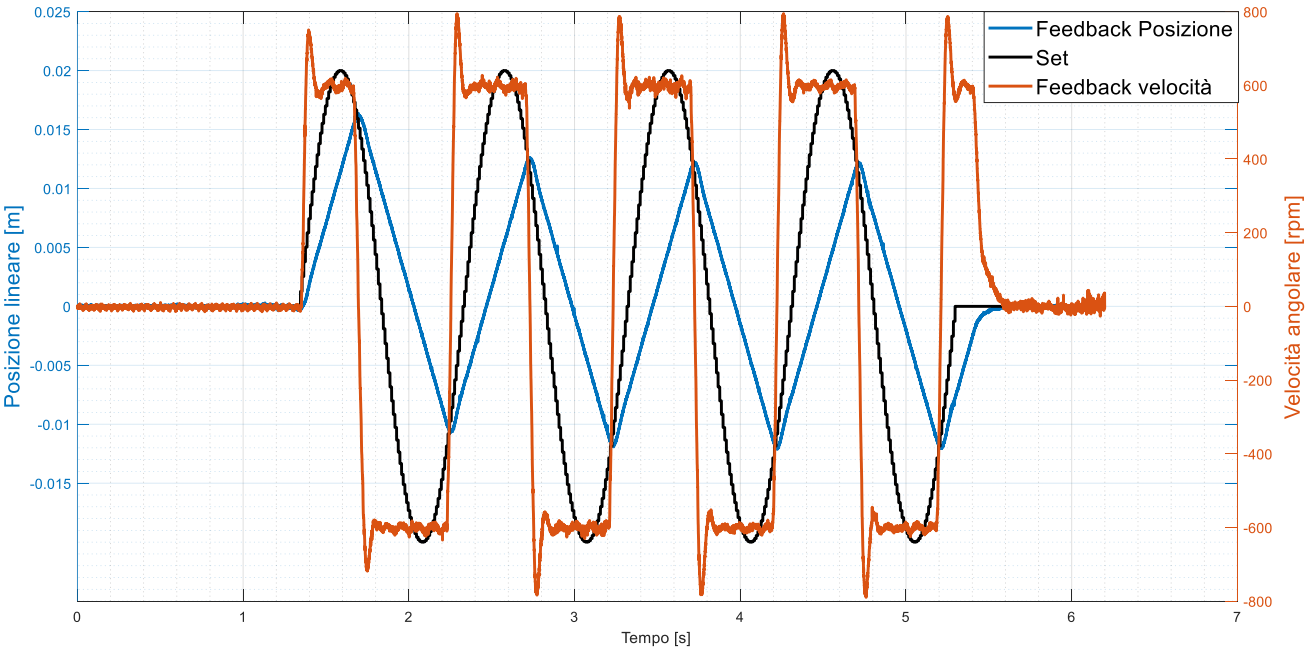


Figura 161: Segnale di comando sinusoidale 2.

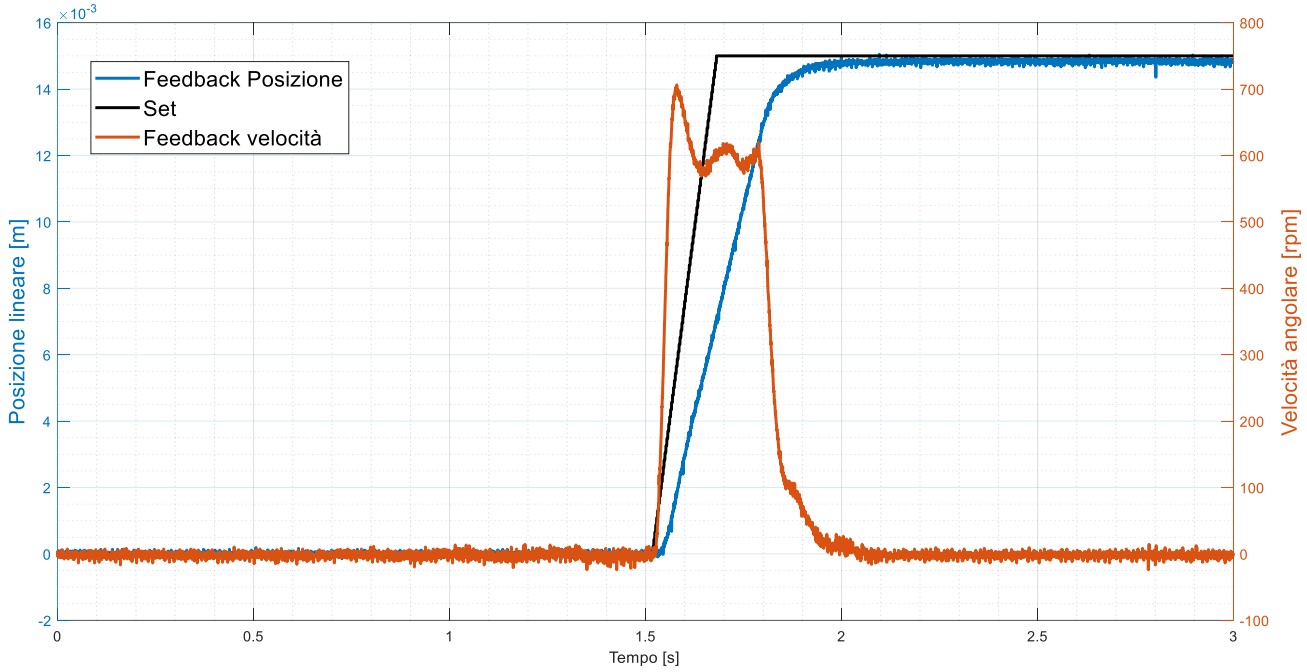


Figura 162: Segnale di comando a rampa 1.

Si intuisce che il controllo implementato nel Driver non lavori in maniera ottimale, per cui è necessario effettuare il *tuning* dei guadagni; partendo dal presupposto che il regolatore di corrente risulti ottimizzato (come riporta la guida di progettazione dell'applicazione), si regola dapprima il regolatore velocità e poi il regolatore posizione.

6.3 Prove con guadagni ottimizzati

L'ottimizzazione dei guadagni per un servoattuatore elettrico non può essere svolta provando per tentativi fino a giungere alla combinazione opportuna, poiché è molto probabile che il sistema raggiunga l'instabilità, rischiando di danneggiare l'EMA. Per cui ci si deve servire di un metodo che permetta di tarare i guadagni in sicurezza

Uno dei metodi di taratura potrebbe essere quello di affidarsi al modello e simulare le condizioni di prova su di esso; non avendo a disposizione un modello validato attraverso le prove sperimentali, non risulta possibile utilizzare questa via.

La guida del Driver riporta una procedura di taratura dei guadagni: deve essere svolta mantenendo un segnale in ingresso al Driver nullo, partendo dal valore di guadagno proporzionale k_p preimpostato del regolatore di velocità e aumentandolo fino a che il controllo del Driver inizia ad avvicinarsi all'instabilità. Tale condizione è monitorata attraverso l'interfaccia utente dell'applicazione *Lenze Engineer* attraverso i parametri attuali di coppia e corrente di riferimento. Si aumenta il k_p fino a che i parametri monitorati iniziano ad oscillare nell'intorno di un punto percentuale, a quel punto si abilita il controllo di posizione (*Enable Position Follower*) e si aumenta tale valore al punto che il motore rumoreggi e la coppia e la corrente inizino ad oscillare in modo marcato. A quel punto è necessario diminuire il valore del guadagno k_p fino a al valore per cui il motore si ristabilizza e dimezzare ancora per due tale valore.

In modo analogo, si effettua la procedura di ottimizzazione per il guadagno integrativo k_I , o meglio per la relativa costante di tempo integrativa T_I (eq.53) , la quale invece di essere aumentata viene diminuita con la stessa metodologia sopra elencata.

$$T_I = \frac{k_p}{k_I} \qquad (53)$$

Si ottengono così i guadagni ottimizzati del regolatore di velocità.

	$K_p \left[\frac{Nm}{rpm}\right]$	$K_I \left[\frac{Nm}{rpm} s\right]$
Preimpostati	0.01164	0.808
Ottimizzati	0.045	3.866

Tabella 35: Guadagni preimpostati e ottimizzati del regolatore di velocità.

Si svolgono le acquisizioni per gli stessi segnali sinusoidali e a rampa presentati in Tabella 34 (Figura 163 Figura 164 Figura 165).

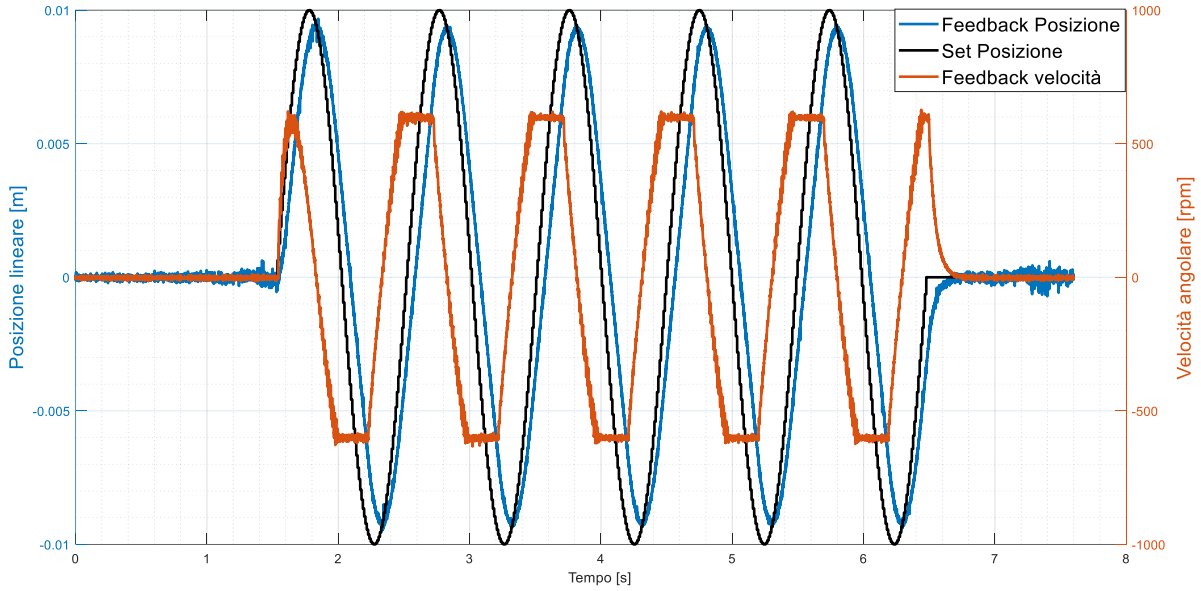


Figura 163: Segnale di comando sinusoidale 1 con guadagni di velocità ottimizzati.

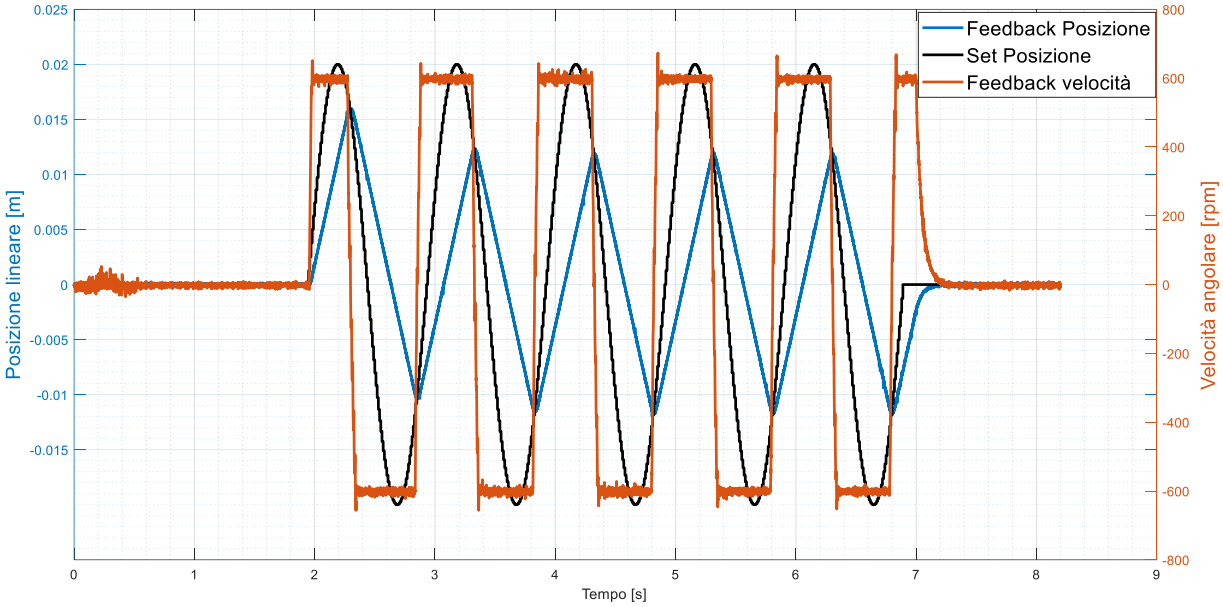


Figura 164: Segnale di comando sinusoidale 2 con guadagni di velocità ottimizzati.

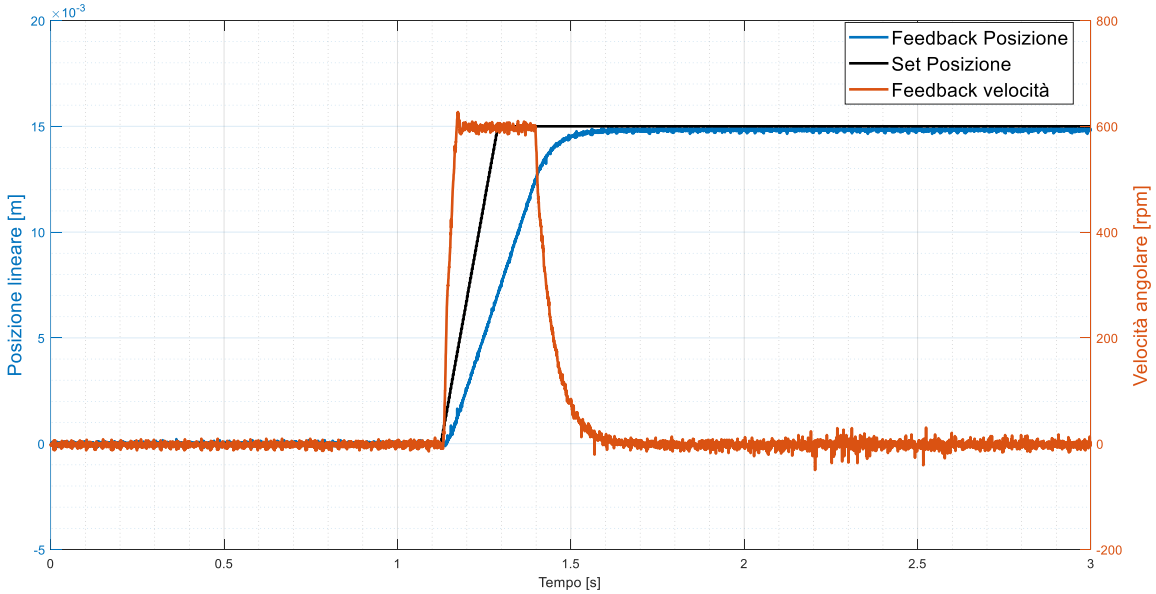


Figura 165: Segnale di comando a rampa con guadagni di velocità ottimizzati.

Con l'ottimizzazione del regolatore di velocità si nota come la velocità del motore risulti decisamente più pronta rispetto alle prove con i guadagni preimpostati (Figura 166 e Figura 167). Ad un comando di posizione a rampa corrisponde un segnale di velocità di riferimento a gradino, per cui si studia la risposta di velocità come un segnale di comando a gradino. Utilizzando i guadagni preimpostati (Figura 166) il processo transitorio di velocità è adeguato, infatti la risposta di velocità è caratterizzata da bassa precisione dinamica e bassa velocità di risposta; ottimizzando i guadagni di velocità si ottiene invece il soddisfacimento di tali specifiche (Figura 167). Tale miglioria nella risposta di velocità non si ripercuote visibilmente sull'anello di posizione.

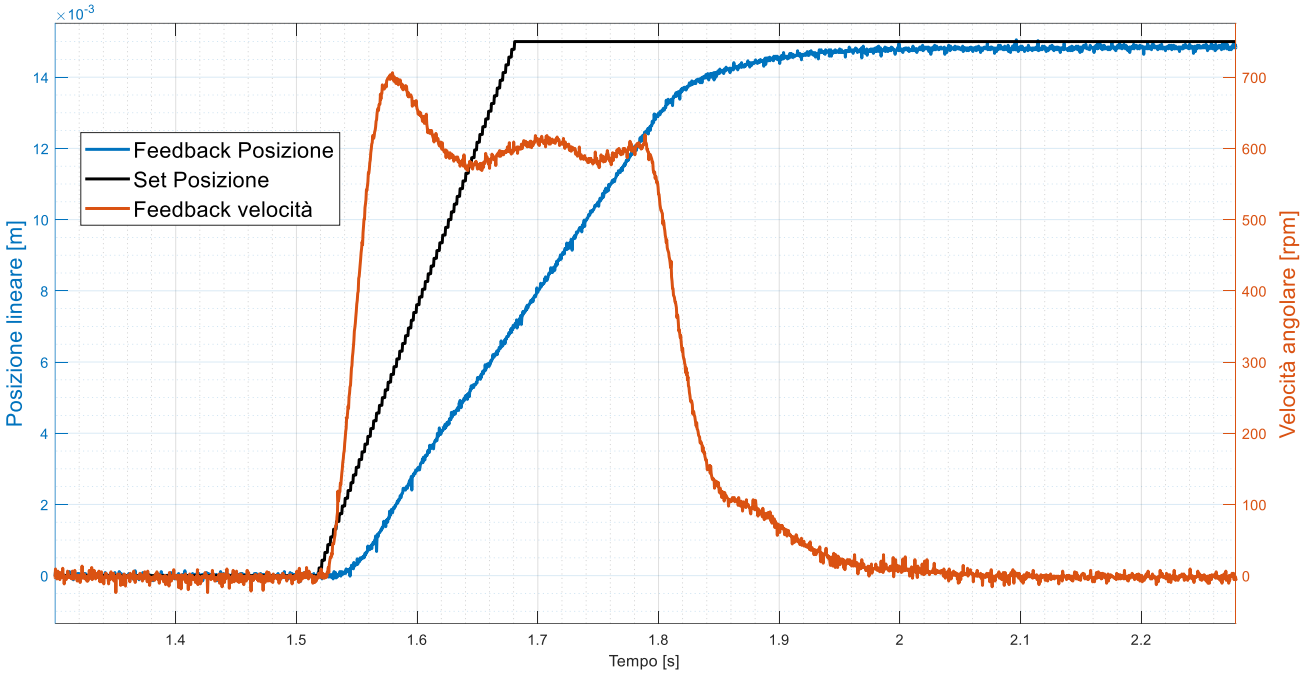


Figura 166: Dettaglio rampa: guadagni di velocità preimpostati.

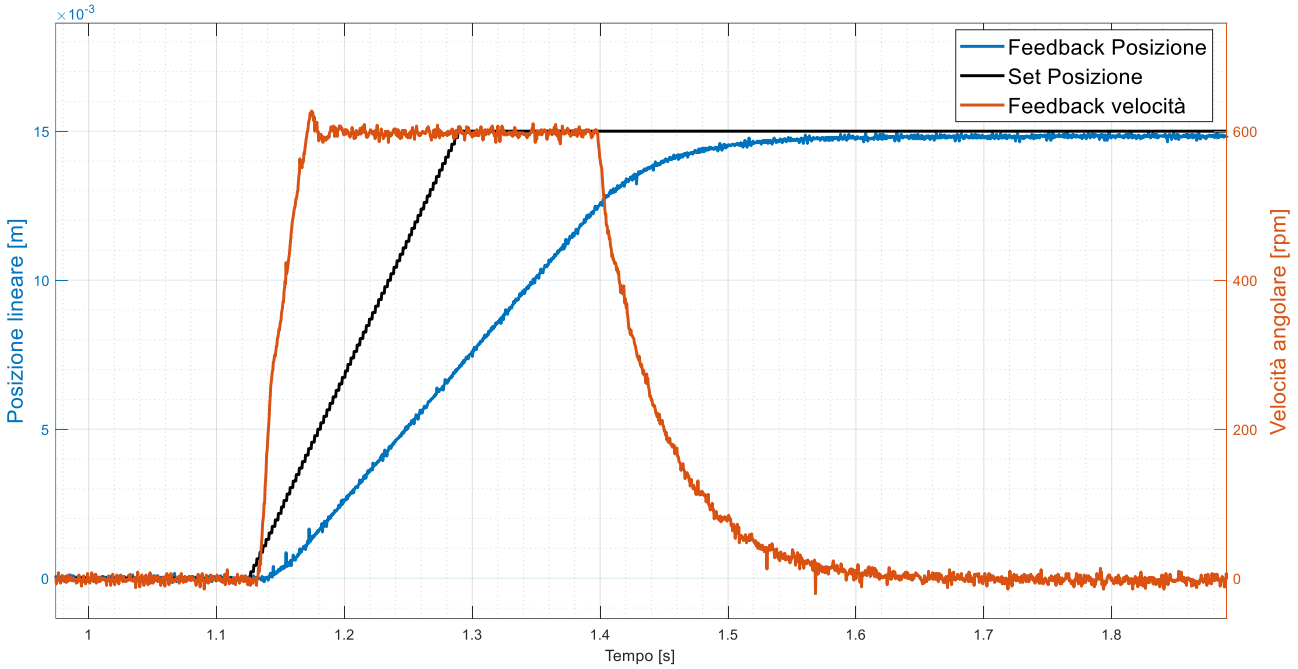


Figura 167: Dettaglio rampa: guadagni di velocità ottimizzati.

Osservando le curve di risposta ad una sinusoide, si intuisce come il controllo del driver non riesca ad ottenere come variabile di riferimento una velocità degna tale da raggiungere il set di posizione durante il test; in modo analogo, dalle curve di risposta ad una rampa, si osserva come la pendenza della risposta, che coincide con la velocità angolare attuale, sia tale da far divergere il *Feedback* di posizione dal segnale in comando di posizione, ovvero la costante di tempo diverge. Di conseguenza, risulta necessario aumentare il guadagno proporzionale in modo da incrementare la pendenza della curva e quindi la velocità, che al momento è limitata attorno ai 600 giri al minuto. Si aumenta il guadagno proporzionale del regolatore di posizione monitorando i parametri da interfaccia utente fino ad ottenere un sistema stabile e pronto.

	$K_{p\,x}\left[\frac{1}{s}\right]$	$K_{p\,\dot{\vartheta}}\left[\frac{Nm}{rpm}\right]$	$K_{I\,\dot{\vartheta}}\left[\frac{Nm}{rpm}s\right]$
Preimpostati	20	0.01164	0.808
Ottimizzati	60	0.045	3.866

Tabella 36: Guadagni preimpostati e ottimizzati del regolatore di velocità e di posizione.

Si effettuano le prove con i segnali di comando precedenti (Tabella 34) e con i guadagni ottimizzati (Tabella 36).

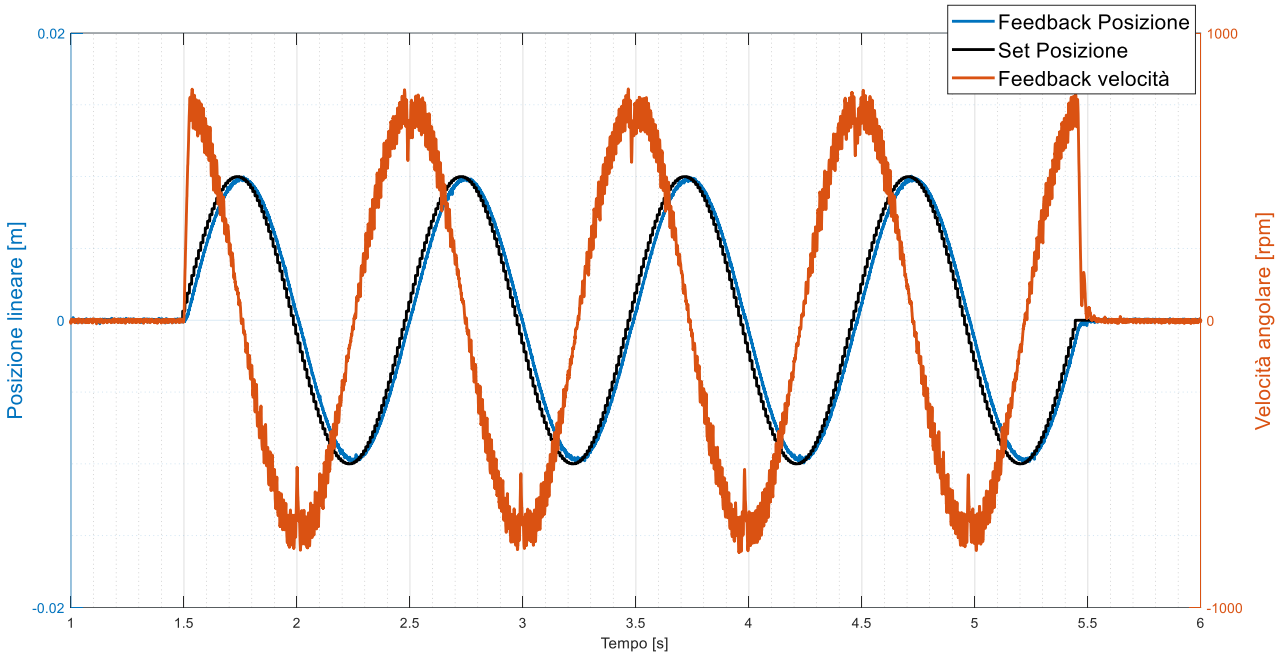


Figura 168: Segnale di comando sinusoidale 1 con guadagni di posizione e velocità ottimizzati.

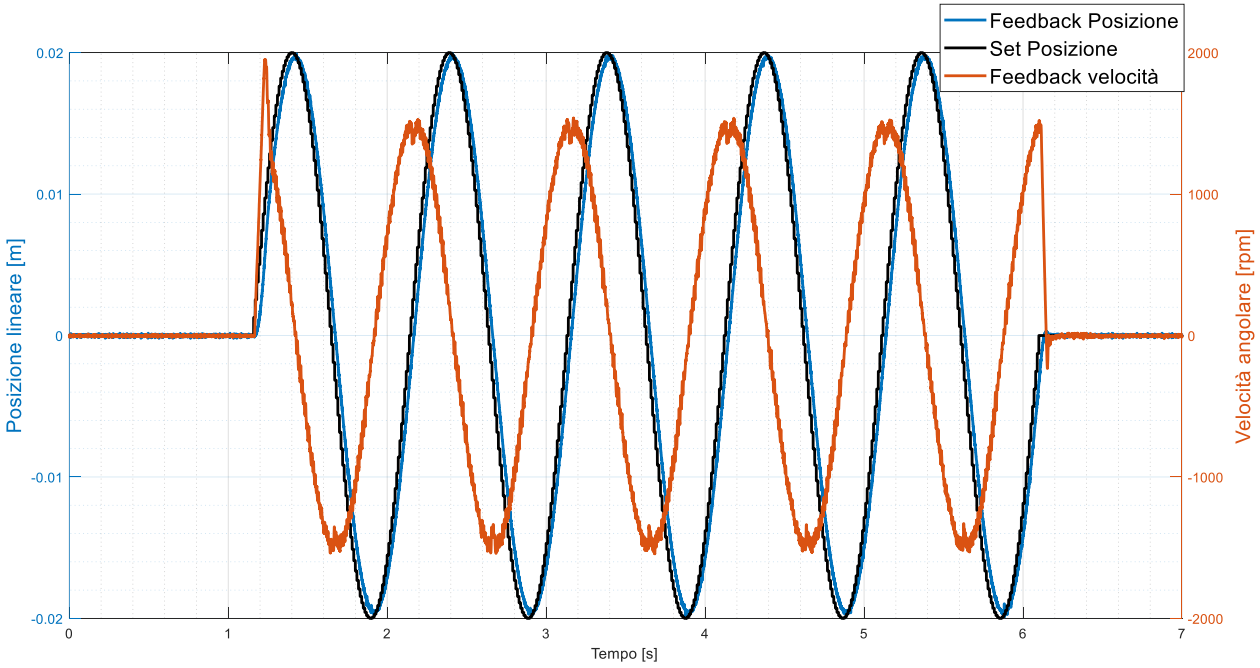


Figura 169: Segnale di comando sinusoidale 2 con guadagni di posizione e velocità ottimizzati.

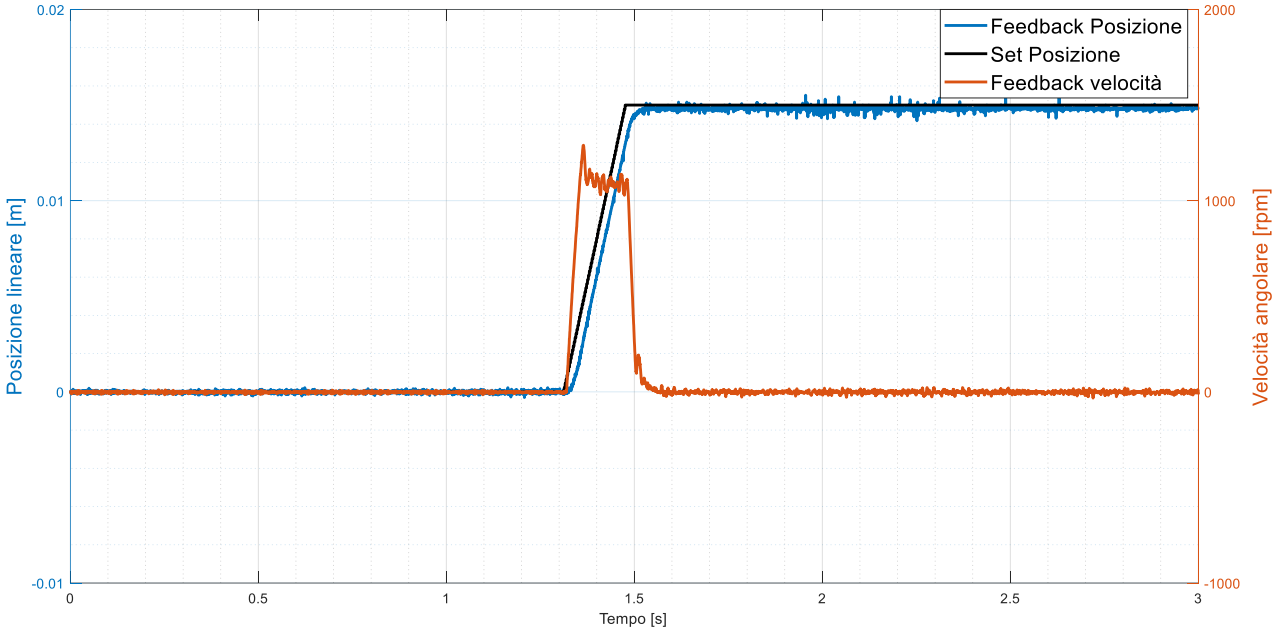


Figura 170: Segnale di comando a rampa con guadagni di posizione e velocità ottimizzati

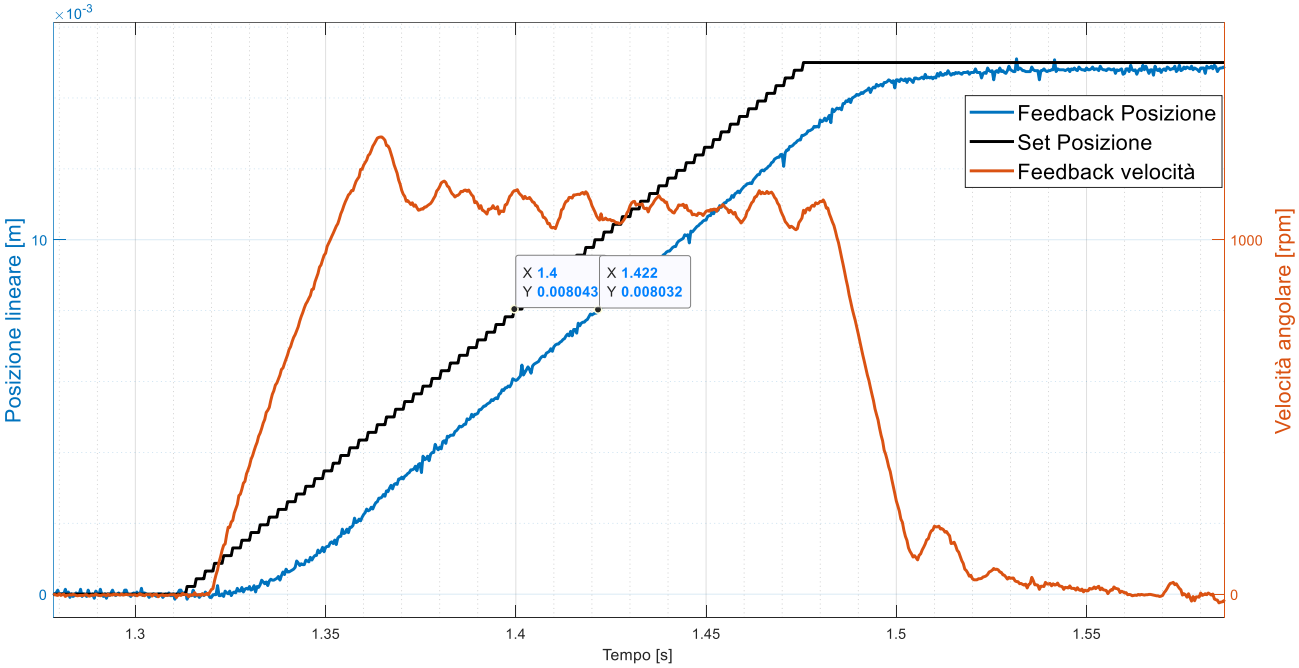


Figura 171: Ingrandimento.

I guadagni dei regolatori ottimizzati consentono alla velocità di raggiungere un valore superiore, così da rispondere alle specifiche di posizionamento (Figura 168, Figura 169, Figura 170). In dettaglio (Figura 171) è presentata la risposta ad una rampa con la valutazione della costante di tempo, che per tale segnale di comando risulta $\tau = 0.022\ s$.
A questo punto, ottimizzati i guadagni, è possibile studiare nel dettaglio le caratteristiche statiche e dinamiche del sistema attraverso storie temporali e diagrammi di risposta in frequenza.

6.3.1 Sinusoidale

Le prove effettuate attraverso comandi sinusoidali con guadagni ottimizzati sono presentate in Tabella 37.

SINUSOIDALE (N.° prova)	**Ampiezza [m]**	**Frequenza [Hz]**
1	0.0025	1
2	0.0025	3
3	0.0025	8
4	0.01	1
5	0.01	3
6	0.01	6
7	0.035	0.5
8	0.035	1
9	0.035	2

Tabella 37: Prove sinusoidali.

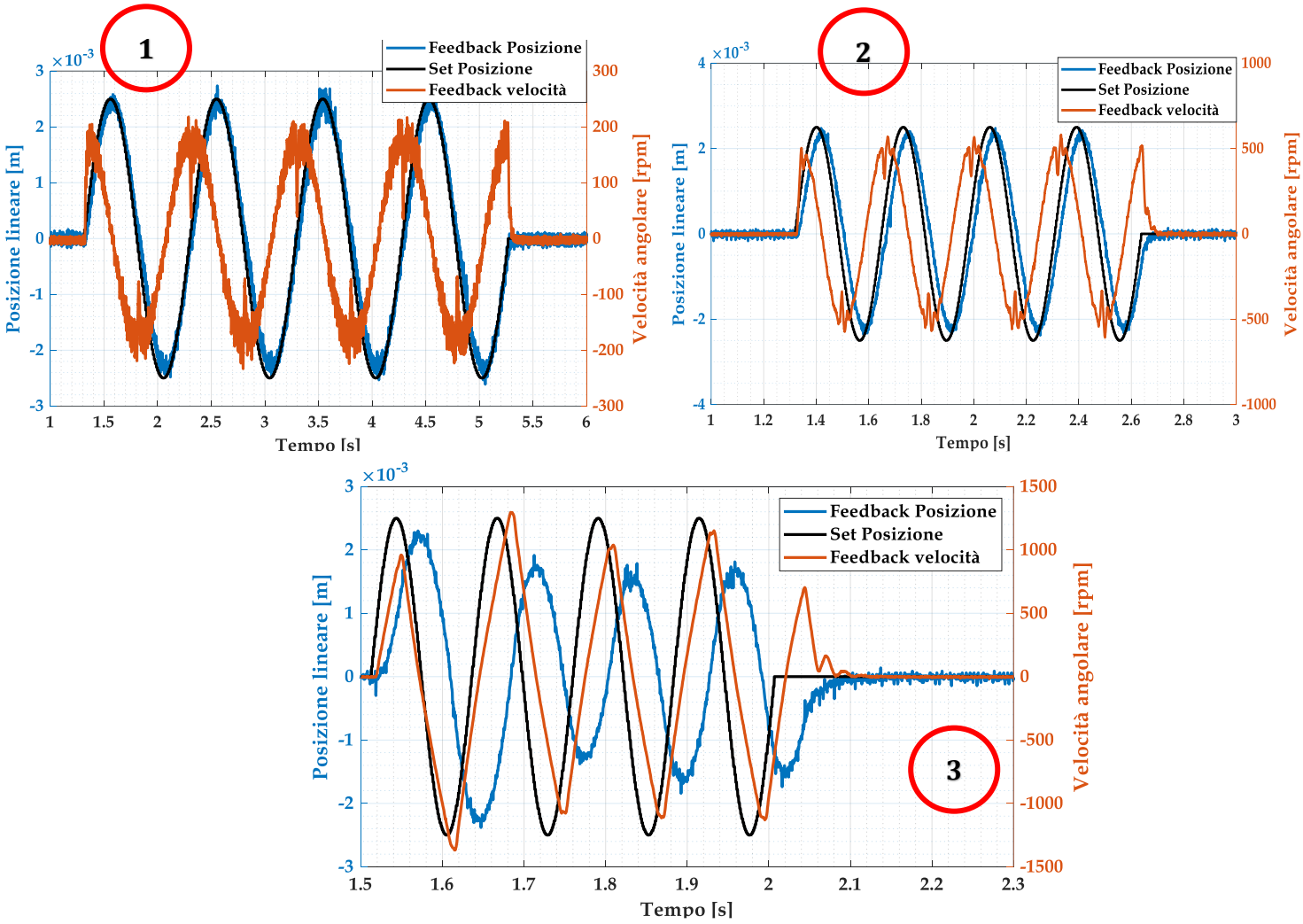


Figura 172: Prova1-2-3.

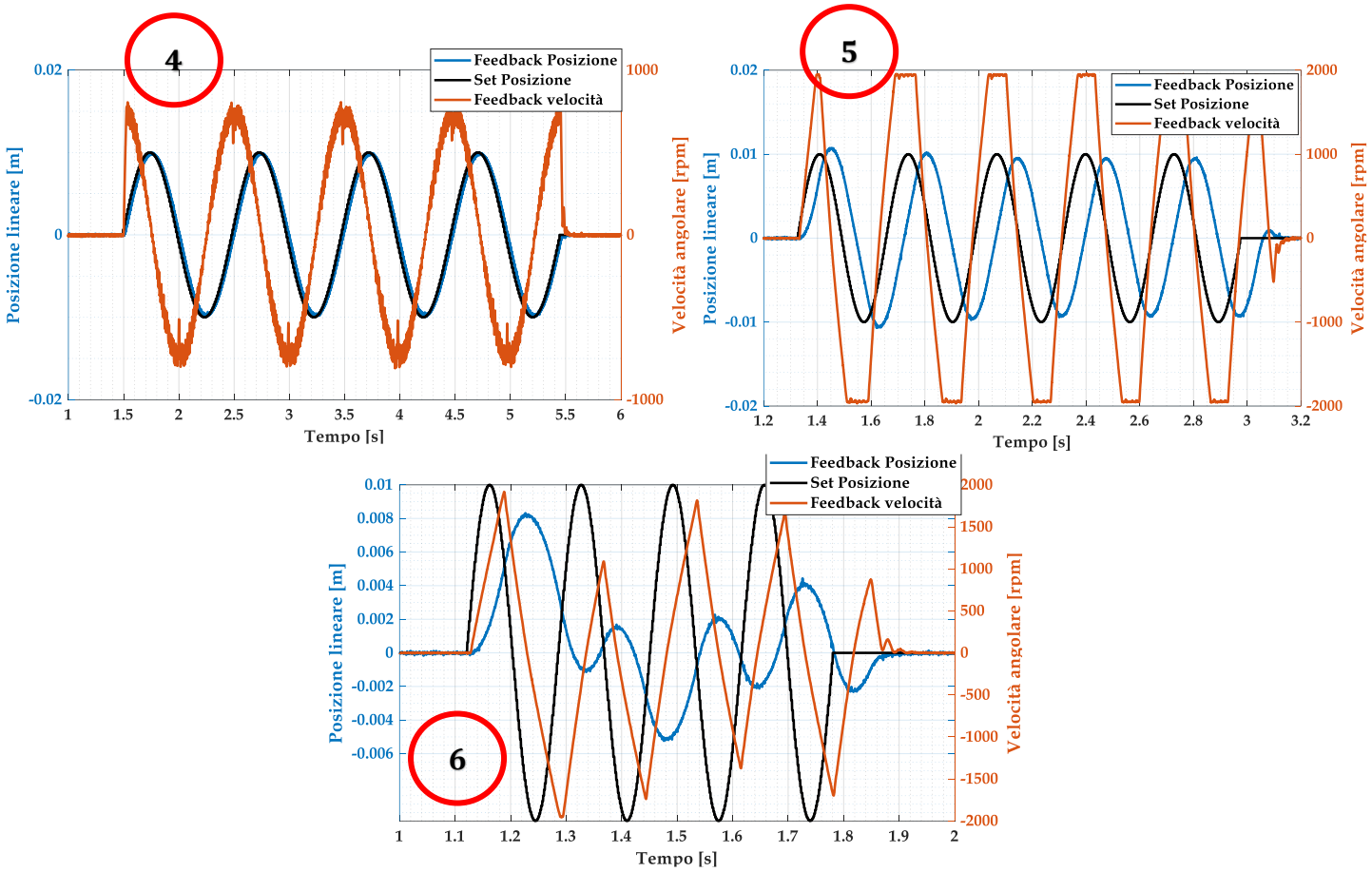


Figura 174: Prova 4-5-6.

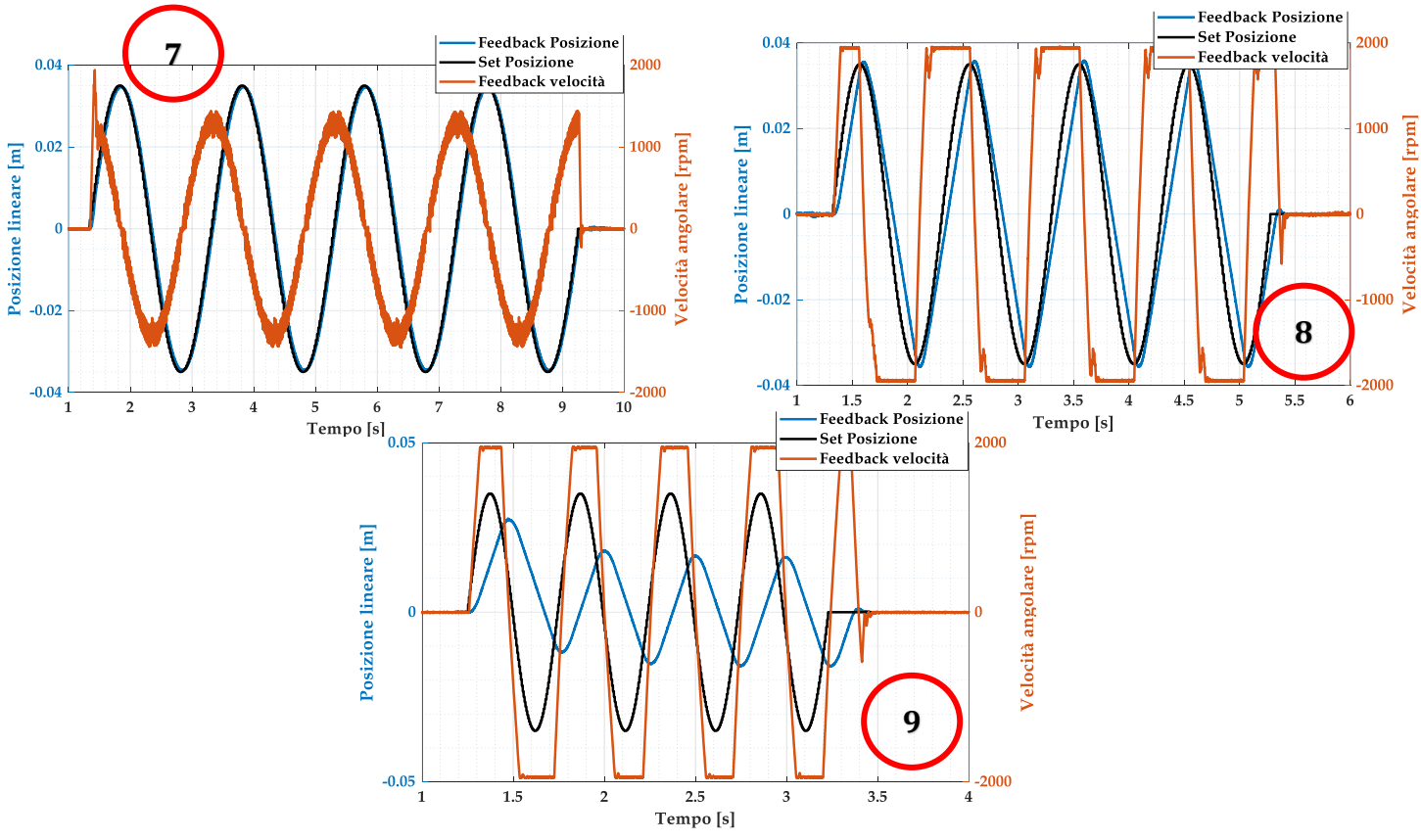


Figura 173: Prova 7-8-9.

Analizzando le prove, per basse frequenze il sistema risponde in modo opportuno, mentre aumentando la frequenza il sistema tende dapprima ad attenuare e a sfasare, poi a derivare a causa delle non linearità.

6.3.2 Rampa

Attraverso dei test effettuati con comandi a rampa (Tabella 38) è possibile analizzare il sistema dal punto di vista della velocità, partendo dal fatto che la derivata della posizione è la velocità, che coincide con il coefficiente angolare della curva di posizione.

RAMPA (N.° prova)	**Valore finale [m]**	**Pendenza [m/s]**
1	0.015	0.09
2	0.015	0.12
3	0.015	0.16
4	0.015	0.3

Tabella 38: Prove a rampa.

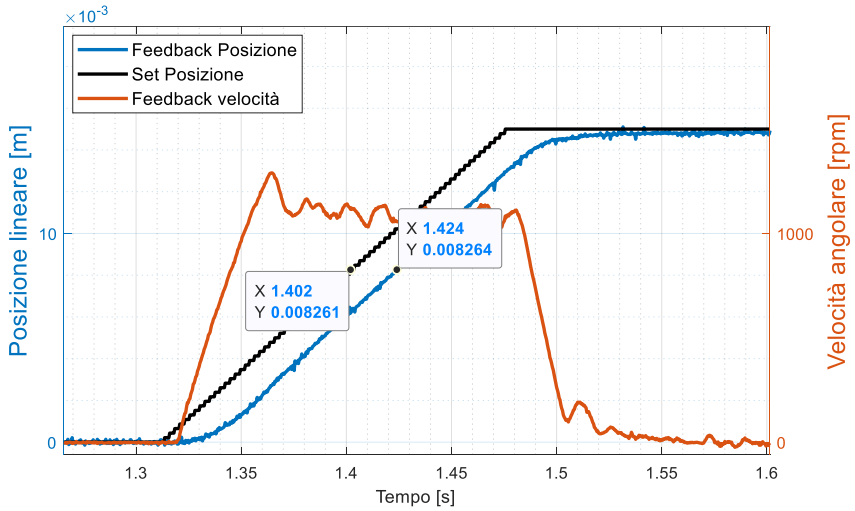


Figura 175: Prova 1.

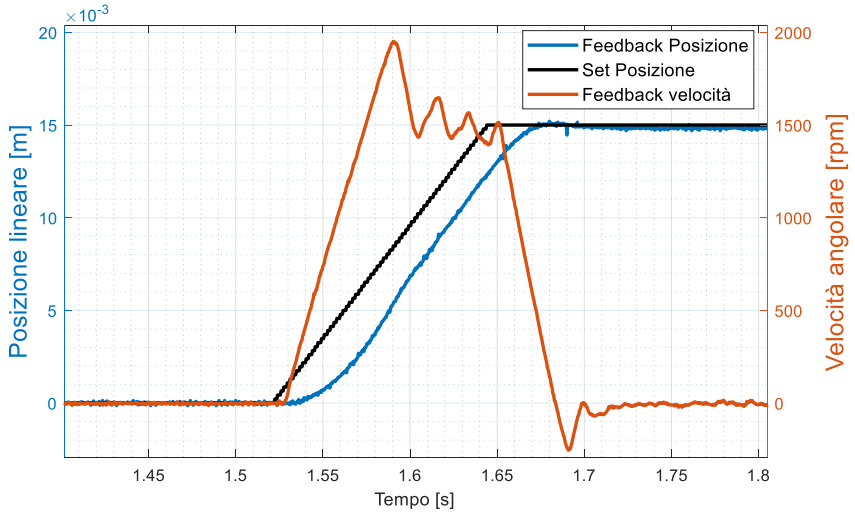


Figura 176: Prova 2.

Nelle prime due prove (Figura 175 e Figura 176) il sistema risponde bene al comando a rampa e, una volta che la velocità raggiunge le condizioni di regime in un determinato intorno, è mantenuto il ritardo ($\tau = 0.022\ s$). Si sottolinea come la velocità che si raggiunge in quel determinato intorno coincida con la pendenza di comando.

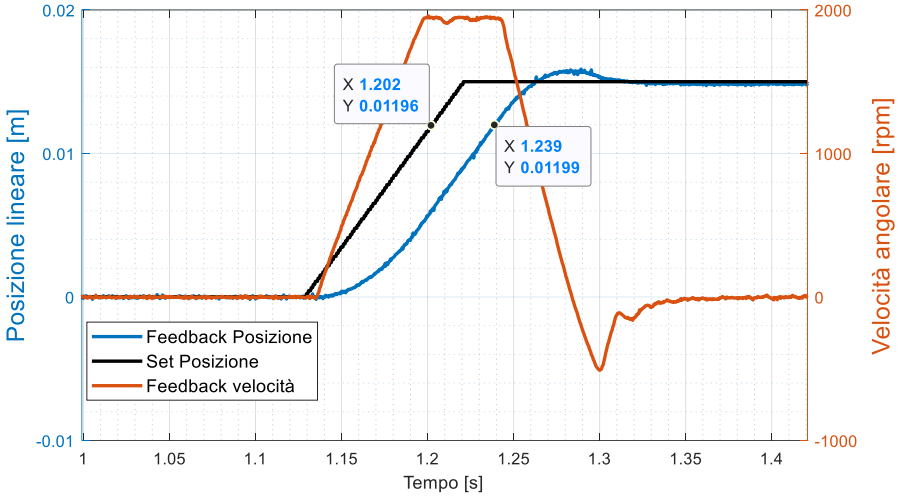


Figura 177: Prova 3.

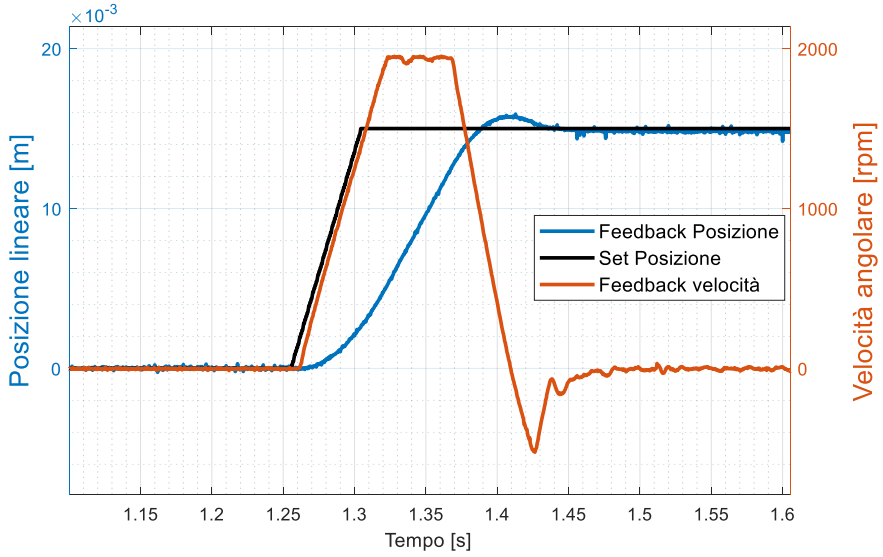


Figura 178: Prova 4.

All'aumentare della pendenza di comando (Figura 177) inizia a presentarsi un *overshoot* nell'attuale posizionamento lineare e si raggiunge il limite di saturazione della velocità (1950 giri al minuto). Il raggiungimento di tale limite comporta un ritardo maggiore della curva *Feedback* di posizione ($\tau = 0.037\ s$).
Per valori ancora maggiori di pendenza di comando (Figura 178), la risposta di posizione diverge in fase di salita, poichè il comando di riferimento sale con una pendenza che non può essere raggiunta dal controllo (a causa della saturazione di velocità) e quindi il ritardo tende ad aumentare.
Gli *overshoot* rilevati risultano inferiori al 20% del segnale di comando, per cui il processo transitorio è accettabile.

6.3.3 Risposta in frequenza

La risposta in frequenza identifica le caratteristiche dinamiche di un sistema e la sua stabilità attraverso rispettivamente lo studio della frequenza di banda passante e i margini di stabilità.
Ottenute le acquisizioni per segnali di comando sinusoidale ad una determinata ampiezza e frequenza, raggruppando le acquisizioni caratterizzate dal medesimo comando di ampiezza e per differenti frequenze su MATLAB e utilizzando lo *script* posto in appendice, è possibile ottenere il diagramma di bode in funzione della frequenza in *Hertz*.
L'EMA, come tutti i sistemi fisici, a differenza del modello esposto nel capitolo precedente, è influenzato dalle non linearità come ad esempio le saturazioni. Tali sistemi possono essere riconducibili a sistemi lineari solamente per bassi segnali di comando, in modo da non coinvolgere le non linearità.
Di conseguenza, attraverso l'analisi della risposta in frequenza, è inoltre possibile determinare la dipendenza della caratteristica dinamica con le non linearità, diagrammando la risposta in frequenza per diversi segnali di comando. I segnali di comando utilizzati per l'analisi di risposta in frequenza sono presentati in Tabella 39, considerando che il comando di ampiezza massimo corrisponde a quello dell'intera corsa dell'attuatore ($100\% \equiv 0.05\ m$).

Diagramma N.°	**Comando [%]**	**Ampiezza [m]**	**Frequenze [Hz]**								
1	5%	0.0025	0.5	1	2	3	4	5	6	7	8
2	10%	0.005	0.5	1	2	3	4	5	6	/	8
3	20%	0.01	0.5	1	2	3	4	5	6	/	8
4	30%	0.015	0.5	1	2	3	4	5	6	/	/
5	50%	0.025	0.5	1	2	3	4	5	6	/	/
6	70%	0.035	0.5	1	2	3	4	5	6	/	/

Tabella 39: Comandi utilizzati nell'analisi di risposta in frequenza.

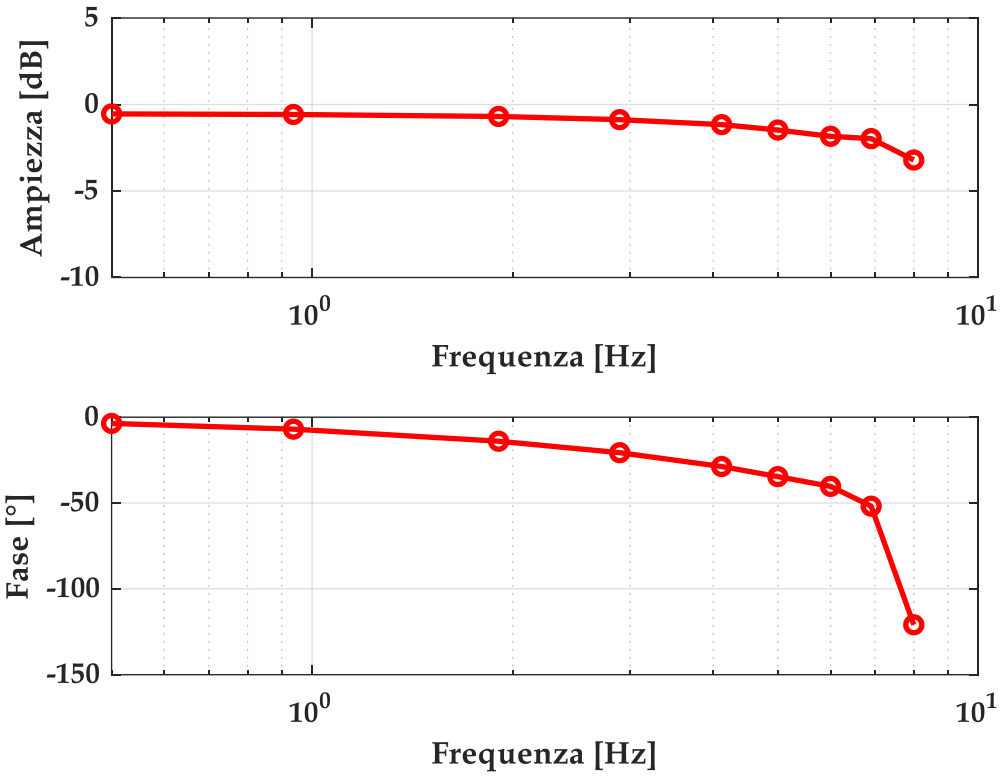


Figura 179: Diagramma 1, 5% del comando.

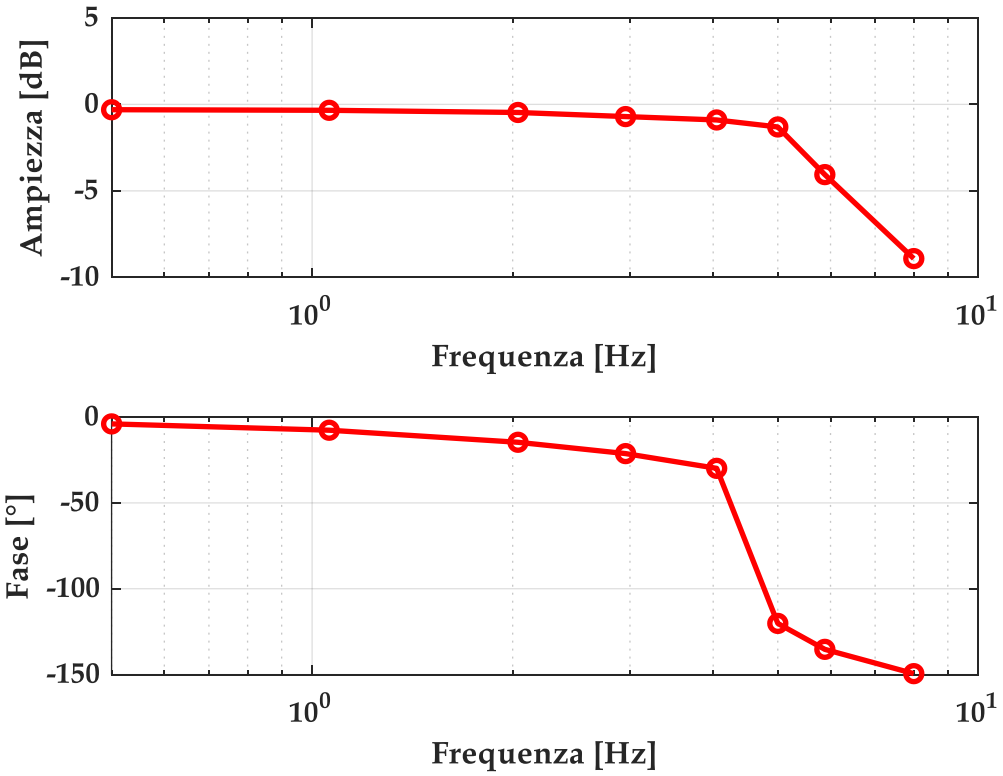


Figura 180: Diagramma 2, 10% del comando.

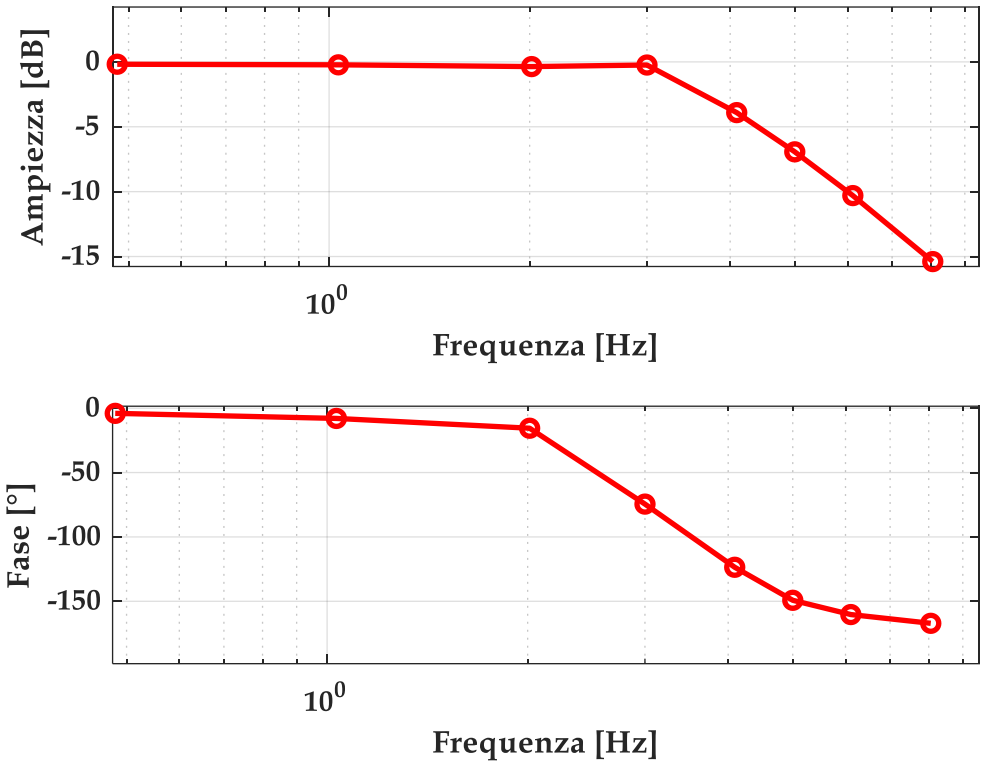


Figura 181: Diagramma 3, 20% del comando.

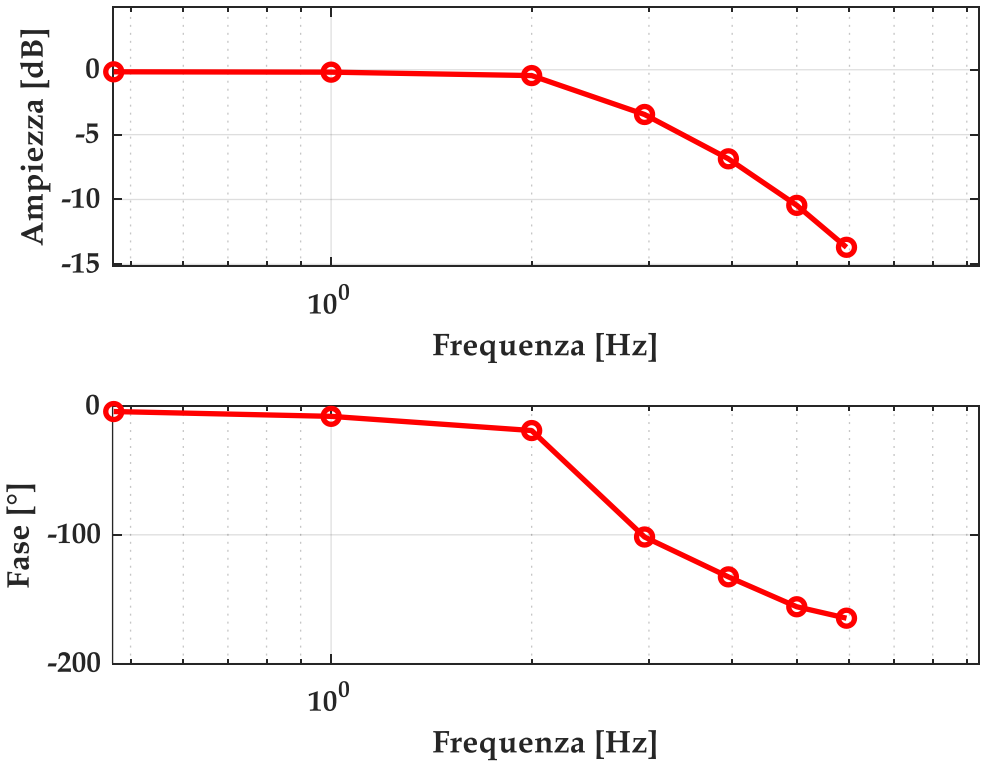


Figura 182: Diagramma 4, 30% del comando.

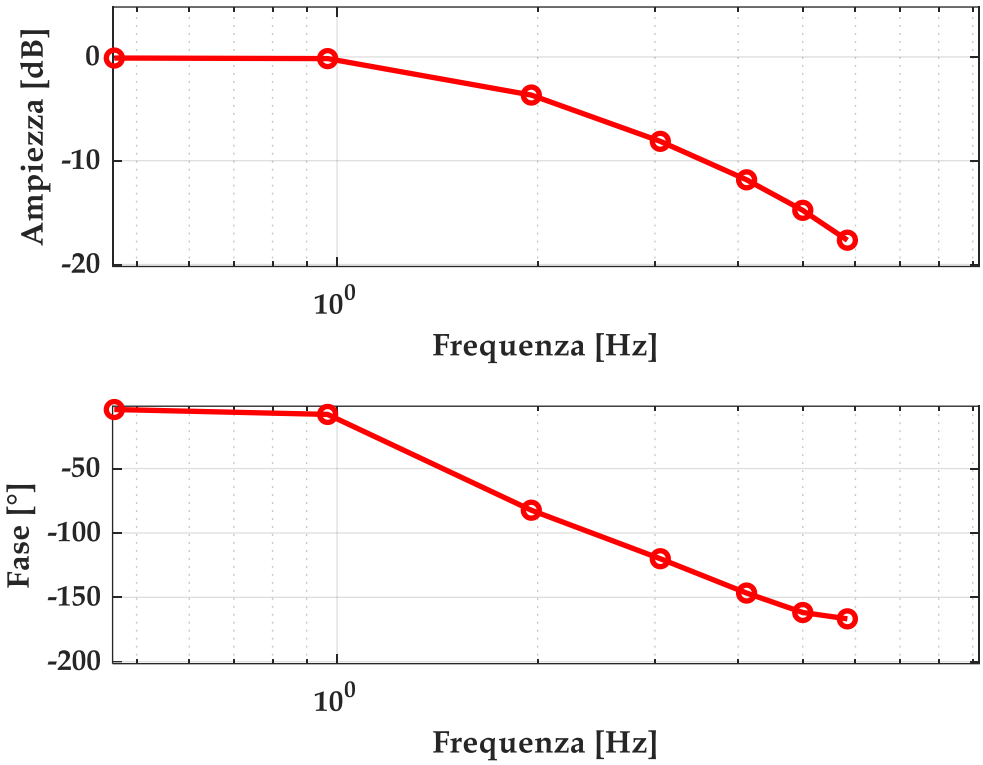


Figura 183: Diagramma 5, 50% del comando.

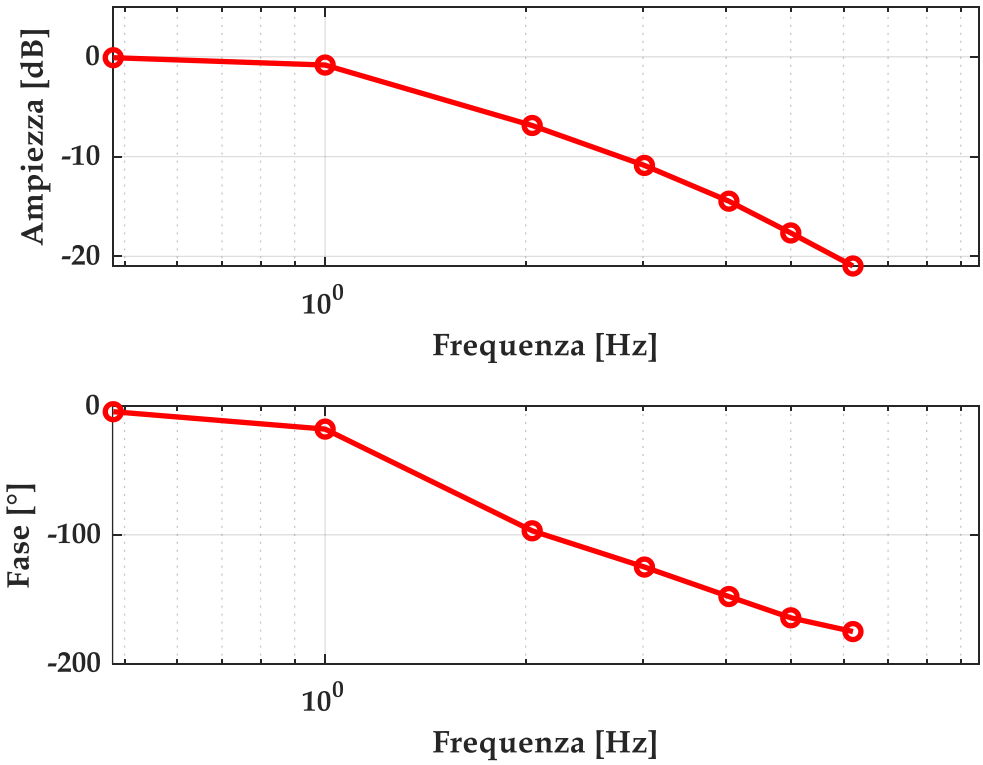


Figura 184: Diagramma 6, 70% del comando.

Analizzando i diagrammi riportati (da Figura 179 a Figura 184), è possibile valutare la larghezza di banda del sistema reale a diverse percentuali di comando. La larghezza di banda è definita come

la frequenza alla quale il modulo della risposta in frequenza scende al 70.7% (-3dB) del valore che aveva a bassa frequenza, ovvero fino a quale frequenza il sistema segue il comando con una accettabile fedeltà. È una misura della velocità di risposta del sistema.
Inserendo le curve su uno stesso diagramma (Figura 185) si nota come la larghezza di banda per basse percentuali di comando in ingresso si attesta intorno agli 8 Hz, diminuisce per comandi sempre maggiori fino a giungere a 1.5 Hz per un comando del 70%.

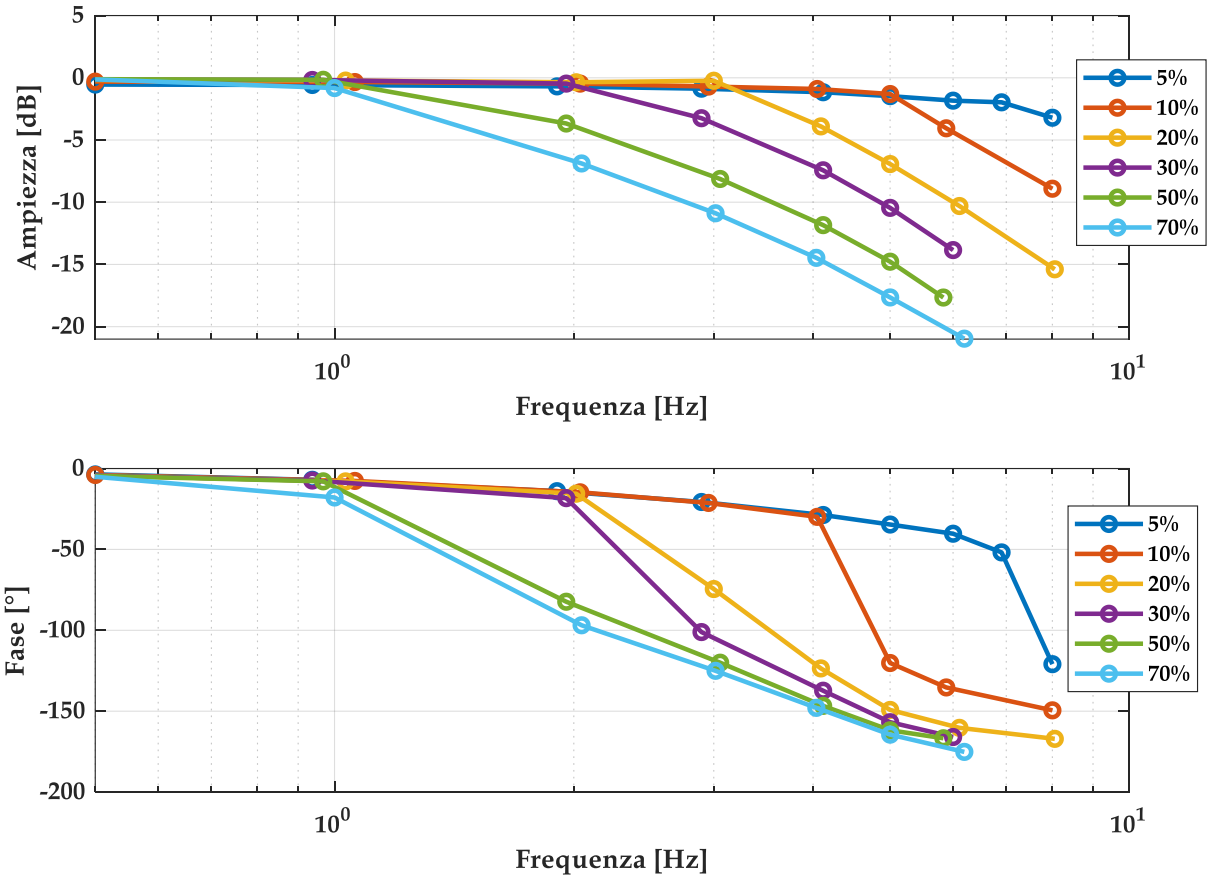


Figura 185: Diagramma di risposta in frequenza per comandi 5%, 10%, 20%, 30%, 50%, 70% del valore massimo di comando.

7. Confronto tra il modello lineare e il sistema reale

Il modello progettato, che caratterizza l'intero sistema dell'EMA, deve essere sufficientemente rappresentativo del sistema reale, nonostante si tratti di un modello lineare.
Di conseguenza, dopo aver ricavato sia il modello rispettante delle specifiche, sia le acquisizioni del sistema reale, è necessario valutare che il modello sia rappresentativo del sistema reale.

7.1 Risposta in frequenza

Per confrontare i diagrammi di bode del modello e del sistema reale si fa riferimento al diagramma di risposta in frequenza in CL di posizione del modello lineare e ai diagrammi di risposta in frequenza sperimentali, riportandoli sul medesimo grafico (Figura 186).
Il grafico mostra che, come ci si aspettava, il modello lineare rappresenta in modo opportuno il sistema reale EMA per basse percentuali di comando e per frequenze contenute, mentre all'aumentare della percentuale di comando insorgono non linearità causate principalmente da saturazioni che il modello lineare non considera. Per cui tale effetto delle saturazioni è tanto più marcato quanto all'aumentare della percentuale di comando e della frequenza.

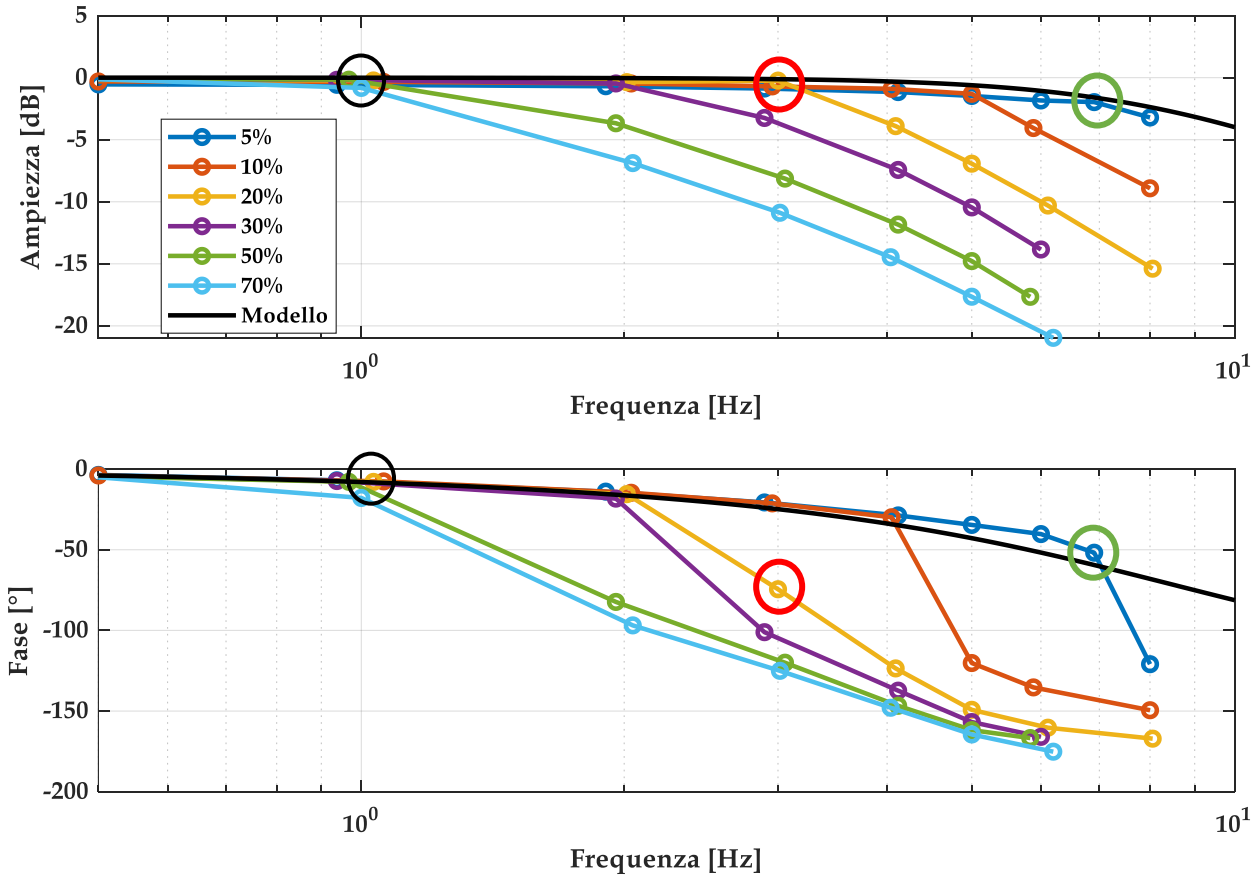


Figura 186: Confronto tra modello e sperimentale: risposta in frequenza.

Il diagramma di bode mostra come il modello lineare rispecchi l'andamento delle prove sperimentali per bassi comandi fino a circa 7-8 Hz, il quale è il campo di frequenze in cui il motore dovrebbe rispondere con un'ottima dinamica ad un comando in ingresso.
A basse frequenze e moderati comandi si ottiene un'ottima approssimazione del sistema reale; si riporta in Figura 187 uno di tali casi in risposta nel dominio del tempo (cerchiato in nero in Figura 186).

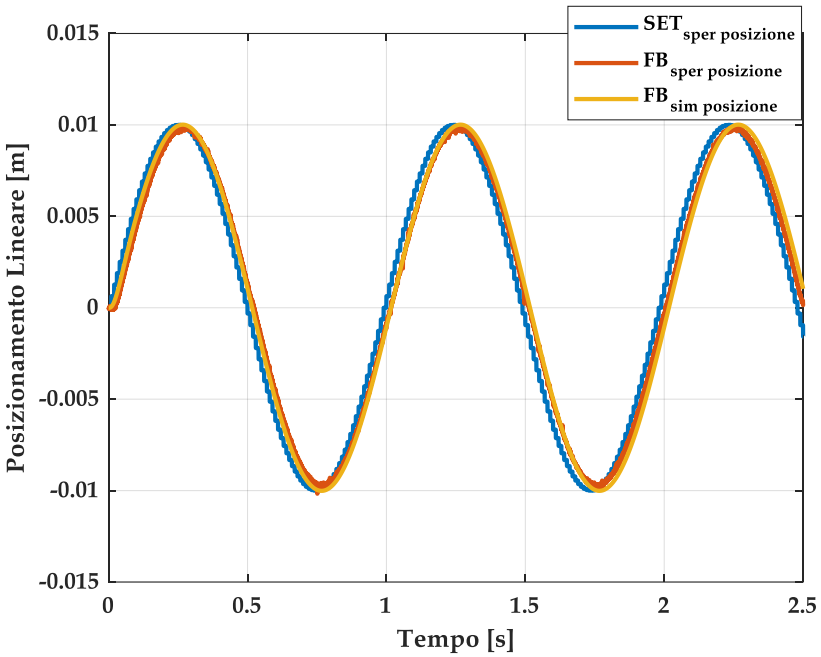


Figura 187: Confronto tra prova sinusoidale sperimentale e modello: comando 20%, frequenza 1Hz.

Analizzando un altro caso limite di risposta nel dominio del tempo (cerchiato in verde in Figura 186), si confrontano la prova per un comando del 5% e frequenza 7 Hz e la relativa simulazione (Figura 188) utilizzando il modello lineare implementato su SIMULINK, il cui diagramma a blocchi è riportato in appendice. Il grafico dimostra quanto riportato sul diagramma di bode, ovvero che il modello approssima in modo opportuno lo sperimentale sia in fase che in ampiezza e nel dettaglio attenua in modo minore e presenta un leggero ritardo rispetto allo sperimentale.

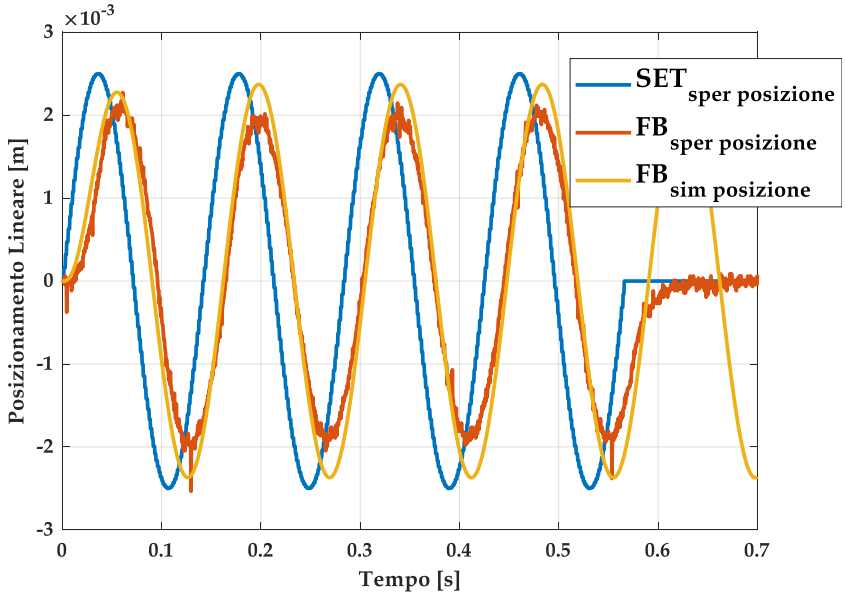


Figura 188: Confronto tra prova sinusoidale sperimentale e modello: comando 5%, frequenza 7Hz.

Prendendo in esempio la prova sinusoidale con 20% di segnale di comando e frequenza 3Hz (cerchiata in rosso in Figura 186), questa rappresenta un caso particolare, in cui è presente un accentuato ritardo di fase non corrispondente ad una attenuazione di ampiezza. Ciò è spiegabile attraverso il fatto che ad elevate frequenze insorgono non linearità, che il modello lineare non considera.

Tale prova sperimentale si confronta con l'analoga prova ottenuta simulando il modello lineare in SIMULINK, il cui diagramma a blocchi è riportato in appendice.

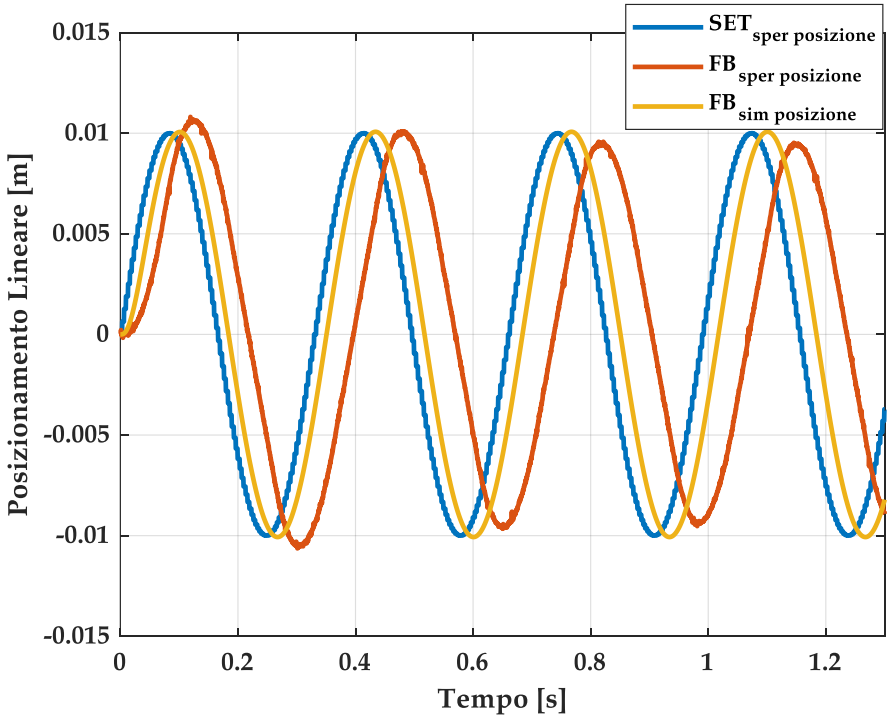


Figura 189: Confronto tra prova sinusoidale sperimentale e modello: 20% comando, 3Hz frequenza

7.2 Risposta ad una rampa

Oltre alla risposta in frequenza e ad un segnale sinusoidale nel dominio del tempo, il modello lineare deve approssimare con una certa affidabilità anche la riposta per un comando a rampa per diversi coefficienti angolari. Per analizzare la risposta si fa nuovamente riferimento alle prove riportate in Tabella 38.

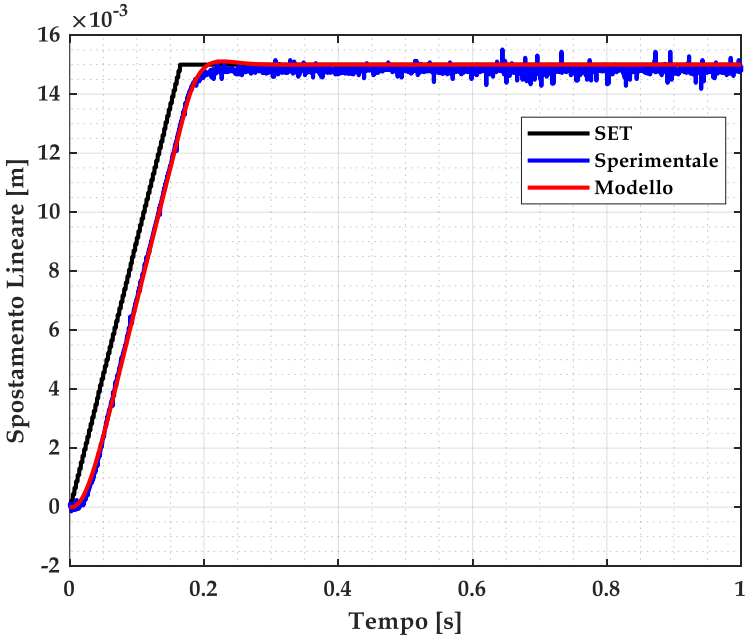


Figura 190: Confronto tra prova a rampa sperimentale e modello: pendenza 0.09 m/s, valore finale 0.015m

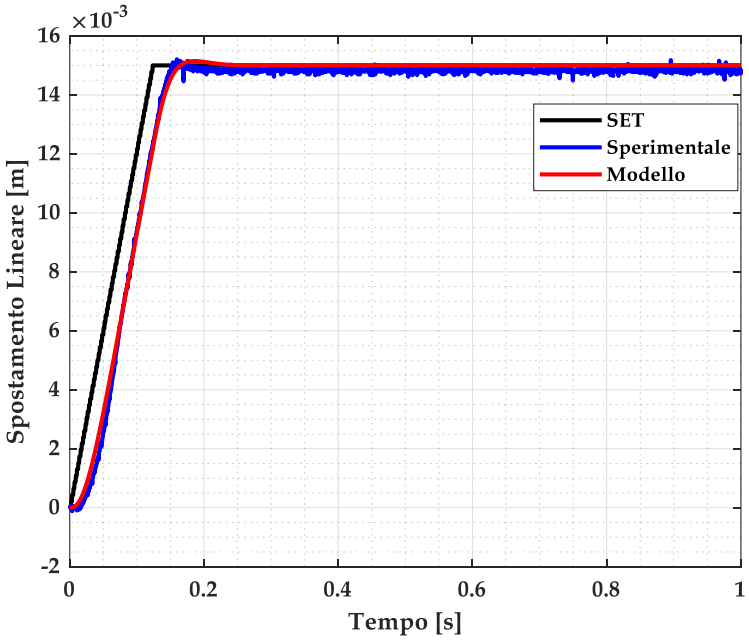


Figura 191: Confronto tra prova a rampa sperimentale e modello: pendenza 0.12 m/s, valore finale 0.015m.

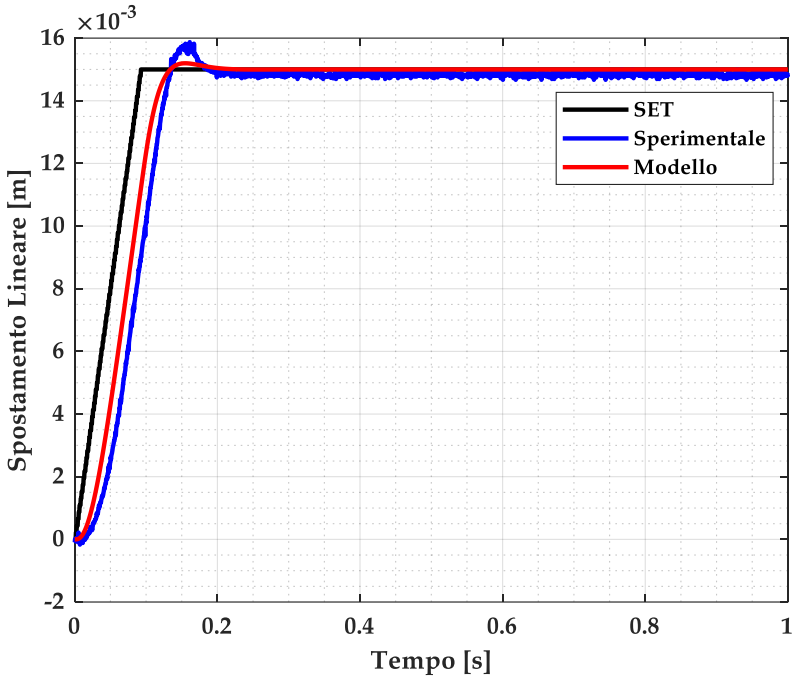


Figura 192: Confronto tra prova a rampa sperimentale e modello: pendenza 0.16 m/s, valore finale 0.015m

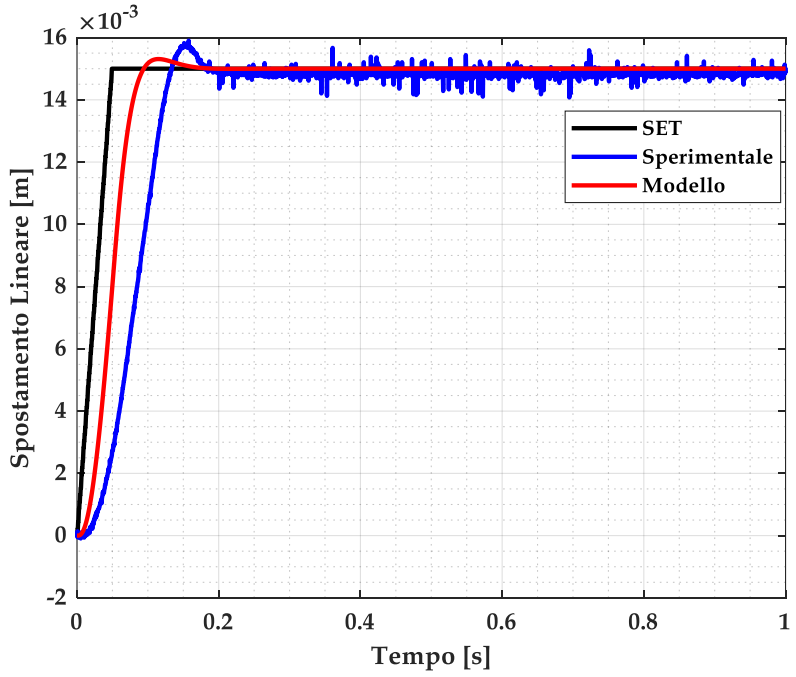


Figura 193: Confronto tra prova a rampa sperimentale e modello: pendenza 0.3 m/s, valore finale 0.015m.

Per bassi coefficienti angolari della rampa di comando, il modello lineare segue con affidabilità il sistema reale, mentre aumentando il coefficiente angolare (e quindi la velocità di riferimento) insorgono non linearità che causano un ritardo nella risposta, di cui il modello lineare non tiene conto (da Figura 190 a Figura 193).

8. Conclusione e sviluppi futuri

8.1 Conclusione

L'elaborato svolto riassume il lavoro eseguito durante il percorso di tesi, la cui conclusione ha necessitato la durata di un anno. In tale periodo è stato richiesto un notevole impegno dal punto di vista della multidisciplinarietà dei campi trattati e l'approfondimento di nuovi temi non trattati nel percorso di studi, come la progettazione di applicazioni di comando e di controllo e il funzionamento di servo inverter. Inoltre, sono stati frequentati tre corsi di formazione presso la *National Instruments* ad Assago (MI), per un totale di 80 ore:

- LabVIEW Core 1;
- LabVIEW Core 2;
- Embedded Control and Monitoring Using LabVIEW.

La progettazione dei software ha richiesto un arduo studio preliminare della logica di programmazione del Driver, attraverso la consultazione di differenti manuali di riferimento, poiché le applicazioni necessitavano di essere definite in base al suo funzionamento. In aggiunta, si sono definiti molteplici modelli del sistema prima di ottenere quello definitivo, poiché anche tale modellazione era influenzata dalle logiche implementate nel Driver. Nonostante tale percorso difficoltoso, rimane la soddisfazione di avere cominciato un cammino svolto in autonomia dalle sue basi, ovvero dallo spacchettamento dei singoli componenti e dalle operazioni di cablaggio, fino a giungere ad un controllo ottimale dell'EMA.
Gli obiettivi inizialmente prefissati per il progetto sono stati portati a termine: l'EMA è stato tarato in modo ottimale per differenti segnali di comando, con una banda passante di circa 8 Hz, ed in sicurezza grazie alla progettazione delle due applicazioni e dell'elettronica di supporto; in aggiunta si è definito un modello lineare affidabile che rispecchiasse il controllo nel Driver. Oltre a ciò, con il seguente elaborato, si è presentato in modo esaustivo il progetto permettendo a chi proseguirà il lavoro di avere una guida dettagliata che agevolerà i successivi studi.

8.2 Sviluppi futuri

In seguito allo studio svolto in questa tesi, si effettuerà il collegamento tra lo stelo dell'EMA e dell'attuatore idraulico mediante il giunto rotante del banco prova, per poi testare l'attuatore idraulico con il disturbo di velocità introdotto dall'EMA. Se necessario, sarà effettuato il modello non lineare dell'EMA. Una volta assicurato il funzionamento ottimale del sistema, verrà installato sulla culla del BPSV l'attuatore di volo del *winglet* e collegato all'attuatore idraulico mediante il giunto rotante.
A questo punto, si disporrà di un attuatore idraulico che, secondo delle leggi di forza in accordo ad un comando di Set, simula le forze aerodinamiche e che agisce come disturbo di forza sul servocomando di volo da testare, il quale verrà controllato in posizione durante una determinata condizione di volo. In questo modo si potranno condurre prove con l'obiettivo di ottimizzare le performance dell'AUT. Inoltre sarà possibile valutare lo stato di salute del comando di volo

degradato artificialmente o sbarcato dopo un percorso di vita operativa effettuando test su di esso e di conseguenza applicare e verificare algoritmi di prognostica per attuatori di volo.

Appendice

A-Listati MATLAB

Parametri e Risposte in frequenza del modello

```
% Da runnare per primo
clc
%clear all
close all
%% Parametri

%geometrici

p=0.005;                                %[m]
gamma=0.08;                            %[Nm/(rad/s)]
L_c=0.05;                           % [m] limite inferiore e superiore della
corsa dell'attuatore

%rendimenti e rapporti di trasmissione

eta_v=0.7;                          %rendimento diretto vite a rulli
eta_c=0.98;                         %rendimento riduttore a cinghia
eta_m=0.79;                         %rendimento motore
tau_c=1;                            %rapp. trasmissione cinghia
tau_v=p/2*pi;                       %rapp. trasmissione vite

%inerzie

Im=0.0004;                        % [kg*m^2] inerzia del motore
Ic=0.1*Im;                        % [kg*m^2] inerzia riduttore a cinghia
Iv=1.45e-4;                       % [kg*m^2] inerzia albero vite;
m=0;                              % [kg] massa parti traslanti (che non sono
incluse in Iv, come esempio occhiello)
I=Im+Ic+tau_c^2/eta_c*(Ic+Iv)+tau_c^2*eta_v/eta_c/eta_v*m;     % [kg*m^2]
inerzia globale riportata sull'albero motore

%dati motore elettrico

L=0.0522;                     %[Ohm*s] induttanza
R=5.8725;                          %[Ohm] resistenza
tau_e=0.0058;                 %[s] costante elettrica del motore
ke=137/(2*pi*(1000/60));      %[V/rad/s] costante di tensione motore
kc=2.34;                      %[Nm/A] costante di coppia motore
kw=1/ke;                      %[rad/s/V] costante di velocità motore
tau_m=(I*R)/(ke*kc);          %[s] costante meccanica del motore
sigma_e=sqrt(1/(tau_e*tau_m));     %[rad/s] pulsazione naturale motore
elettrico
eps_e=0.5*sqrt(tau_m/tau_e);       % fattore di smorzamento elettrico
```

```
imax=10;                        %[A] corrente massima
Cmax=18;                        %[Nm] coppia massima
Vn=345;                         %[V] rated voltage
In=2.6;                         %[A] rated current
Cn=5.5;                         %[Nm] rated torque
tetarif=1950;                   %[rpm] rated speed
fn=130;                         %[Hz] rated frequency
Vmax_d=10;                      %[V] limite superiore e inferiore dei segnali
in arrivo dal driver
wn=1950;                        %[rpm] rated speed

%controllo e trasduttori

kp_i=107.5;                % [V/A] guadagno proporzionale anello di corrente
ki_i=34677;                % [V/A/s] guadagno integrale anello di corrente
kp_w=0.111;               % [Nm/rad/s] guadagno rete proporzionale del
controllo  coppia
ki_w=7.723;                   % [Nm*s/rad/s] guadagno rete integrativa del
controllo  coppia
kd_w=0;                    % [Nm/((rad/s)*s)] guadagno rete derivativa del
controllo velocità angolare
kp_teta=60;                % [(rad/s)/rad] guadagno rete proporzionale del
controllo posizione angolare
ki_teta=0;                 % [(rad/s)/(rad*s)] guadagno rete integrativa del
controllo posizione angolare
kd_teta=0;                 % [(rad/s)/(rad/s)] guadagno rete derivativa del
controllo posizione angolare
kt_teta=1;                 % [rad/rad] guadagno del trasduttore di posizione
angolare
kt_tetad=1;                % [(rad/s)/(rad/s)] guadagno del trasduttore di
velocità angolare
Ha=1;                      % [-] guadagno feedback anello di corrente
s=tf('s');
%kt_tetad=1/(0.001*s+1)

%Ritardo dell'elettronica
fs=2500;                       % [Hz] frequenza di campionamento
t_mp=1e-4;                     % [s] tempo di calcolo
tau=t_mp+1/(2*fs);             % [s] ritardo elettronica
D=tf(1,'inputdelay',tau);% funzione trasferimento ritardo %D = exp(-tau*s);

%Funzioni di trasferimento anello corrente+motore e controlli velocità
%posizione

Gc_teta=kp_teta+(kd_teta*s); %kp_teta*(1+(kd_teta/kp_teta)*s);   %
controllo globale di posizione (PD)
Gc_w=kp_w+(ki_w/s); %((kp_w/ki_w)*s+1)*(ki_w/s);                 %
controllo locale di velocità  (PI)
Ga=kp_i+(ki_i/s); %(((kp_i/ki_i)*s+1)*(ki_i/s));                 %
controllo locale di corrente  (PI)
G_1=feedback((1/(L*s)),(R));  %((1/R)*(1/((L/R)*s+1));
G_2=feedback((1/(I*s)),(gamma));
%(1/gamma)*(1/((I/gamma)*s+1));G_2=1/(I*s)%
G_3=feedback((G_1),(ke*kc*G_2));
%(kc*eta_c*eta_m*G_1*G_2)/(1+(kc*eta_c*eta_m*G_2*G_1*ke));
G_4=feedback((Ga*G_3*D),(1));
%(Ga*G_3)/(1+(Ga*G_3*Ha/(kc*eta_c*eta_m*G_2)));
%(Ga*G_1*G_3*kc*eta_c*eta_m*G_2)/((kc*eta_c*eta_m*G_2)+(Ga*G_1*G_3*Ha))
```

```
%% TF

%G_ol_m=G_1*kc*eta_c*eta_m*G_2*ke
%G_cl_m=feedback((G_1*kc*eta_c*eta_m*G_2),(ke))
% anello corrente+motore elettrico
G_ol_curr=(Ga*G_3*D);                                          % G*H
G_cl_curr=G_4;
% anello velocità
G_ol_tetad=G_cl_curr*Gc_w*G_2*kt_tetad;                        %
(G_4*Gc_w*kt_tetad)/(eta_c*eta_m*kc);
G_cl_tetad=feedback((G_cl_curr*Gc_w*G_2),(kt_tetad));
%((G_4*Gc_w)/(eta_c*eta_m*kc))/(1+((G_4*Gc_w*kt_tetad)/(eta_c*eta_m*kc)));
% anello posizione
G_ol_teta=Gc_teta*G_cl_tetad*(1/s)*kt_teta;
%(G_4*Gc_w*Gc_teta*kt_teta)/(s*(eta_c*eta_m*kc+G_4*Gc_w*kt_tetad));
G_cl_teta=feedback((Gc_teta*G_cl_tetad*(1/s)),(kt_teta));
%(G_4*Gc_w*Gc_teta)/(s*(eta_c*eta_m*kc+(G_4*Gc_w*kt_tetad))+(G_4*Gc_w*Gc_te
ta*kt_tetad*kt_teta));

% funzione rigidezza
G_R=-(1-((Gc_w*kw*sigma_e^2*eta_c*eta_m*((-kt_tetad*s/Gc_teta)-
kt_teta)*(2*pi/p))/(s^2+2*eps_e*sigma_e*s+sigma_e^2*eta_c*eta_m))/((p/2*pi)
*(1/eta_v)*(s^2+2*eps_e*sigma_e*s+sigma_e^2*eta_c*eta_m)*(1/s)*(p/2*pi)))/(
(1+tau_e*s)*(1/I*tau_e));

%% Plot

figure('name','OPEN LOOP POSITION')
w=logspace(log10(0.1),log10(10000),1e3); %vettore (10^-1->10^3 di 1000
campioni)
[mag_ol,Phase_ol]=bode(G_ol_teta,w);
mag_ol=20*log10(squeeze(mag_ol)); Phase_ol=(squeeze(Phase_ol));
subplot(2,1,1)
semilogx(w/(2*pi),mag_ol,'linewidth',2),grid, grid minor
title(['G ol \theta,        ','Gc \thetad(PI): kp_w=',num2str(kp_w),'
Nm/rad/s ,  ki_w=',num2str(ki_w),' Nm*s/rad/s;','  Gc \theta(PD):
kp_\theta=',num2str(kp_teta),' s^-1, kd_\theta=',num2str(kd_teta)])
ylabel('Ampiezza [dB]','fontweight','bold')
set(gca,'FontName','Book Antiqua')
set(gca,'FontSize',11)
set(gca,'FontWeight','bold')
xlim([w(1)/(2*pi) w(end)/(2*pi)])
ylim([-20 inf])
subplot(2,1,2)
semilogx(w/(2*pi),Phase_ol,'linewidth',2),grid, grid minor
xlabel('Frequenza [Hz]','fontweight','bold')
ylabel('Fase [°]','fontweight','bold')
set(gca,'FontName','Book Antiqua')
set(gca,'FontSize',11)
set(gca,'FontWeight','bold')
xlim([w(1)/(2*pi) w(end)/(2*pi)])
ylim([-180 0])
[Gm,Pm,Wcg,Wcp] = margin(G_ol_teta);
Gm_dB = 20*log10(Gm);

figure('name','CLOSED LOOP POSITION')
w=logspace(log10(1),log10(10000),1e3);
[mag_cl,Phase_cl]=bode(G_cl_teta,w);
mag_cl_CL=20*log10(squeeze(mag_cl)); Phase_cl_CL=(squeeze(Phase_cl));
subplot(2,1,1)
semilogx(w/(2*pi),mag_cl_CL,'linewidth',2),grid, grid minor
```

```
title(['G cl \theta,      ','Gc \thetad(PI): kp_w=',num2str(kp_w),'
Nm/rad/s ,  ki_w=',num2str(ki_w),' Nm*s/rad/s;','  Gc \theta(PD):
kp_\theta=',num2str(kp_teta),' s^-1, kd_\theta=',num2str(kd_teta)])
ylabel('Ampiezza [dB]','fontweight','bold')
set(gca,'FontName','Book Antiqua')
set(gca,'FontSize',11)
set(gca,'FontWeight','bold')
xlim([w(1)/(2*pi) w(end)/(2*pi)])
ylim([-20 inf])
subplot(2,1,2)
semilogx(w/(2*pi),Phase_cl_CL,'linewidth',2),grid, grid minor
xlabel('Frequenza [Hz]','fontweight','bold')
ylabel('Fase [°]','fontweight','bold')
set(gca,'FontName','Book Antiqua')
set(gca,'FontSize',11)
set(gca,'FontWeight','bold')
xlim([w(1)/(2*pi) w(end)/(2*pi)])
ylim([-180 0])

figure('name','OPEN LOOP SPEED')
w=logspace(log10(0.1),log10(10000),1e3);
[mag_ol_d,Phase_ol_d]=bode(G_ol_tetad,w);
mag_ol_d=20*log10(squeeze(mag_ol_d)); Phase_ol_d=(squeeze(Phase_ol_d));
subplot(2,1,1)
semilogx(w/(2*pi),mag_ol_d,'linewidth',2),grid, grid minor
title(['G ol \thetad,      ','Gc \thetad(PI): kp_w=',num2str(kp_w),'
Nm/rad/s ,  ki_w=',num2str(ki_w),' Nm*s/rad/s'])
ylabel('Ampiezza [dB]','fontweight','bold')
set(gca,'FontName','Book Antiqua')
set(gca,'FontSize',11)
set(gca,'FontWeight','bold')
xlim([w(1)/(2*pi) w(end)/(2*pi)])
ylim([-20 inf])
subplot(2,1,2)
semilogx(w/(2*pi),Phase_ol_d,'linewidth',2),grid, grid minor
xlabel('Frequenza [Hz]','fontweight','bold')
ylabel('Fase [°]','fontweight','bold')
set(gca,'FontName','Book Antiqua')
set(gca,'FontSize',11)
set(gca,'FontWeight','bold')
xlim([w(1)/(2*pi) w(end)/(2*pi)])
ylim([-180 0])
[Gmd,Pmd,Wcgd,Wcpd] = margin(G_ol_tetad);
Gmd_dB = 20*log10(Gmd);

figure('name','CLOSED LOOP SPEED')
w=logspace(log10(0.1),log10(10000),1e3);
[mag_cl_d,Phase_cl_d]=bode(G_cl_tetad,w);
mag_cl_d=20*log10(squeeze(mag_cl_d)); Phase_cl_d=(squeeze(Phase_cl_d));
subplot(2,1,1)
semilogx(w/(2*pi),mag_cl_d,'linewidth',2),grid, grid minor
title(['G cl \thetad,      ','Gc \thetad(PI): kp_w=',num2str(kp_w),'
Nm/rad/s ,  ki_w=',num2str(ki_w),' Nm*s/rad/s'])
ylabel('Ampiezza [dB]','fontweight','bold')
set(gca,'FontName','Book Antiqua')
set(gca,'FontSize',11)
set(gca,'FontWeight','bold')
xlim([w(1)/(2*pi) w(end)/(2*pi)])
ylim([-20 inf])
subplot(2,1,2)
semilogx(w/(2*pi),Phase_cl_d,'linewidth',2),grid, grid minor
```

```
xlabel('Frequenza [Hz]','fontweight','bold')
ylabel('Fase [°]','fontweight','bold')
set(gca,'FontName','Book Antiqua')
set(gca,'FontSize',11)
set(gca,'FontWeight','bold')
xlim([w(1)/(2*pi) w(end)/(2*pi)])
ylim([-180 0])

figure('name','OPEN LOOP CURRENT+MOTOR')
w=logspace(log10(0.1),log10(1000000),1e3);
[mag_ol_curr,Phase_ol_curr]=bode(G_ol_curr,w);
mag_ol_curr=20*log10(squeeze(mag_ol_curr));
Phase_ol_curr=(squeeze(Phase_ol_curr));
subplot(2,1,1)
semilogx(w/(2*pi),mag_ol_curr,'linewidth',2),grid, grid minor
title(['G ol curr+motor,      ','Gc (PI): kp_i=',num2str(kp_i),' V/A ,
ki_i=',num2str(ki_i),' V/A/s' ])
ylabel('Ampiezza [dB]','fontweight','bold')
set(gca,'FontName','Book Antiqua')
set(gca,'FontSize',11)
set(gca,'FontWeight','bold')

xlim([w(1)/(2*pi) w(end)/(2*pi)])
ylim([-80 inf])
subplot(2,1,2)
semilogx(w/(2*pi),Phase_ol_curr,'linewidth',2),grid, grid minor
xlabel('Frequenza [Hz]','fontweight','bold')
ylabel('Fase [°]','fontweight','bold')
set(gca,'FontName','Book Antiqua')
set(gca,'FontSize',11)
set(gca,'FontWeight','bold')
xlim([w(1)/(2*pi) w(end)/(2*pi)])
ylim([-180 0])
[Gmcurr,Pmcurr,Wcgcurr,Wcpcurr] = margin(G_ol_curr);
Gmcurr_dB = 20*log10(Gmcurr);

figure('name','CLOSED LOOP CURRENT+MOTOR')
w=logspace(log10(0.1),log10(1000000),1e3);
[mag_cl_curr,Phase_cl_curr]=bode(G_cl_curr,w);
mag_cl_curr=20*log10(squeeze(mag_cl_curr));
Phase_cl_curr=(squeeze(Phase_cl_curr));
subplot(2,1,1)
semilogx(w/(2*pi),mag_cl_curr,'linewidth',2),grid, grid minor
title(['G cl curr+motor,       ','Gc (PI): kp_i=',num2str(kp_i),' V/A ,
ki_i=',num2str(ki_i),' V/A/s' ])
ylabel('Ampiezza [dB]','fontweight','bold')
set(gca,'FontName','Book Antiqua')
set(gca,'FontSize',11)
set(gca,'FontWeight','bold')
xlim([w(1)/(2*pi) w(end)/(2*pi)])
ylim([-20 inf])
subplot(2,1,2)
semilogx(w/(2*pi),Phase_cl_curr,'linewidth',2),grid, grid minor
xlabel('Frequenza [Hz]','fontweight','bold')
ylabel('Fase [°]','fontweight','bold')
set(gca,'FontName','Book Antiqua')
set(gca,'FontSize',11)
set(gca,'FontWeight','bold')

xlim([w(1)/(2*pi) w(end)/(2*pi)])
ylim([-180 0])
```

```
% figure('name','OPEN LOOP motor')
% w=logspace(log10(0.1),log10(10000),1e3); %vettore (10^-1->10^3 di 1000
campioni)
% [mag_mol,Phase_mol]=bode(G_ol_m,w);
% mag_mol=20*log10(squeeze(mag_mol)); Phase_mol=(squeeze(Phase_mol));
% subplot(2,1,1)
% semilogx(w/(2*pi),mag_mol,'linewidth',2),grid, grid minor
% title(['G ol \theta,       ','Gc \thetad(PI): kp_w=',num2str(kp_w),'
Nm/rad/s ,  ki_w=',num2str(ki_w),' Nm*s/rad/s;','  Gc \theta(PD):
kp_\theta=',num2str(kp_teta),' s^-1, kd_\theta=',num2str(kd_teta)])
% ylabel('Ampiezza [dB]')
% xlim([w(1)/(2*pi) w(end)/(2*pi)])
% ylim([-20 inf])
% subplot(2,1,2)
% semilogx(w/(2*pi),Phase_mol,'linewidth',2),grid, grid minor
% xlabel('Frequenza [Hz]')
% ylabel('Fase [°]')
% xlim([w(1)/(2*pi) w(end)/(2*pi)])
% ylim([-180 0])
% [Gmm,Pmm,Wcgm,Wcpm] = margin(G_ol_m);
% Gmm_dB = 20*log10(Gmm);
%
% figure('name','CLOSED LOOP motor')
% w=logspace(log10(0.1),log10(10000),1e3);
% [mag_mcl,Phase_mcl]=bode(G_cl_m,w);
% mag_mcl=20*log10(squeeze(mag_mcl)); Phase_mcl=(squeeze(Phase_mcl));
% subplot(2,1,1)
% semilogx(w/(2*pi),mag_mcl,'linewidth',2),grid, grid minor
% title(['G cl \theta,       ','Gc \thetad(PI): kp_w=',num2str(kp_w),'
Nm/rad/s ,  ki_w=',num2str(ki_w),' Nm*s/rad/s;','  Gc \theta(PD):
kp_\theta=',num2str(kp_teta),' s^-1, kd_\theta=',num2str(kd_teta)])
% ylabel('Ampiezza [dB]')
% xlim([w(1)/(2*pi) w(end)/(2*pi)])
% ylim([-20 inf])
% subplot(2,1,2)
% semilogx(w/(2*pi),Phase_mcl,'linewidth',2),grid, grid minor
% xlabel('Frequenza [Hz]')
% ylabel('Fase [°]')
% xlim([w(1)/(2*pi) w(end)/(2*pi)])
% ylim([-180 0])
% figure('name','G_R [N/m]')
% w=logspace(log10(0.1),log10(10000),1e3);
% [mag_r,Phase_r]=bode(G_R,w);
% mag_r=20*log10(squeeze(mag_r)); Phase_r=(squeeze(Phase_r));
% % subplot(2,1,1)
% semilogx(w,mag_r,'linewidth',2),grid, grid minor
% title(['G cl \theta,       ','Gc \thetad(PI): kp_w=',num2str(kp_w),',
ki_w=',num2str(ki_w),'; ','Gc \theta(PD): kp_\theta=',num2str(kp_teta),',
kd_\theta=',num2str(kd_teta)])
% ylabel('Ampiezza [dB]')
% xlabel('Frequency (rad/s)')
% xlim([w(1) w(end)])
% ylim([-20 inf])

%% Visualizzazione dei dati nella Command Window
fprintf('    \n    Dati servosistema \n\n')
fprintf('  Gain Margin - Current+motor     [dB]            Gm_curr  =
%7.3g\n',Gmcurr_dB)
fprintf('  Phase Margin - Current+motor     [°]            Pm_curr  =
%7.3g\n',Pmcurr)
```

```
fprintf('  Gain Margin - Speed             [dB]            Gm_w  =
%7.3g\n',Gmd_dB)
fprintf('  Phase Margin - Speed             [°]            Pm_w  =
%7.3g\n',Pmd)
fprintf('  Gain Margin - Position          [dB]            Gm_teta  =
%7.3g\n',Gm_dB)
fprintf('  Phase Margin - Position           [°]            Pm_teta  =
%7.3g\n',Pm)

% step
u=0.025;
figure('name','Step response')
step(G_cl_teta*u), grid, grid minor
title(['Ampiezza gradino =', num2str(u),'m'])

 %clear all %,'y-o','linewidth',2,'m-o','linewidth',2,'c-
o','linewidth',2,'r-o','linewidth',2,'g-o','linewidth',2,'b-
o','linewidth',2

figure
subplot(2,1,1)
p=semilogx(FREQ05,MAG05,FREQ1,MAG1,FREQ2,MAG2,FREQ3,MAG3,FREQ5,MAG5,FREQ7,M
AG7)
hold on
w=logspace(log10(1),log10(10000),1e3);
[mag_cl,Phase_cl]=bode(G_cl_teta,w);
mag_cl_CL=20*log10(squeeze(mag_cl)); Phase_cl_CL=(squeeze(Phase_cl));
m1=semilogx(w/(2*pi),mag_cl_CL)
%m1=semilogx(w/(2*pi),mag_cl_CL)
p(1).LineWidth = 2;
p(1).Marker = 'o';
p(2).LineWidth = 2;
p(2).Marker = 'o';
p(3).LineWidth = 2;
p(3).Marker = 'o';
p(4).LineWidth = 2;
p(4).Marker = 'o';
p(5).LineWidth = 2;
p(5).Marker = 'o';
p(6).LineWidth = 2;
p(6).Marker = 'o';
m1.Color='black'
m1.LineWidth = 2;
grid on
xlabel('Frequenza [Hz]')
ylabel('Ampiezza [dB]')
set(gca,'FontName','Book Antiqua')
set(gca,'FontSize',11)
set(gca,'FontWeight','bold','ylim',[-21 5],'xlim',[0.5 10])
legend('5%','10%','20%','30%','50%','70%','Modello')
subplot(2,1,2)
t=semilogx(FREQ05,PHASE05,FREQ1,PHASE1,FREQ2,PHASE2,FREQ3,PHASE3,FREQ5,PHAS
E5,FREQ7,PHASE7)
hold on
m2=semilogx(w/(2*pi),Phase_cl_CL)
%m2=semilogx(w/(2*pi),Phase_cl_CL)
t(1).LineWidth = 2;
t(1).Marker = 'o';
t(2).LineWidth = 2;
t(2).Marker = 'o';
t(3).LineWidth = 2;
```

```
t(3).Marker = 'o';
t(4).LineWidth = 2;
t(4).Marker = 'o';
t(5).LineWidth = 2;
t(5).Marker = 'o';
t(6).LineWidth = 2;
t(6).Marker = 'o';
m2.Color='black'
m2.LineWidth = 2;
grid on
xlabel('Frequenza [Hz]')
ylabel('Fase [°]')
set(gca,'FontName','Book Antiqua')
set(gca,'FontSize',11)
set(gca,'FontWeight','bold','xlim',[0.5 10])
legend('5%','10%','20%','30%','50%','70%','Modello')

% ramp
figure('name','Ramp response')
rs.t=0:1e-3:1;
% [s] vettore tempi
rs.xd_ramp=0.1;
% [m/s] pendenza della rampa di posizione lineare (velocità lineare)
rs.xtarget=0.025;
% [m] posizione lineare raggiunta al termine della rampa
rs.tetad_ramp=rs.xd_ramp*2*pi/p;
% [rad/s] pendenza della rampa di posizione angolare del motore (velocità
angolare)
rs.tetatarget=rs.xtarget*2*pi/p;
% [rad] posizione angolare raggiunta dall'asse motore al termine della
rampa
rs.time2tetatarget=rs.tetatarget/rs.tetad_ramp;
% [s] tempo necessario per raggiungere la posizione (angolare/lineare)
desiderata
rs.u_fun=@(x)0*(x<0.1)+(x-0.1).*rs.tetad_ramp.*(x>=0.1 &
x<=rs.time2tetatarget+0.1)+rs.tetatarget*(x>rs.time2tetatarget+0.1);
rs.u=rs.u_fun(rs.t);
[rs.y,rs.tsim,rs.x]=lsim(G_cl_teta,rs.u,rs.t);     %Sys=U*t
plot(rs.t,rs.u*p/2/pi,'b',rs.tsim,rs.y*p/2/pi,'r','linewidth',2), grid,
grid minor
xlabel('Tempo [s]','fontweight','bold')
ylabel('Spostamento Lineare [m]','fontweight','bold')
legend('SET','FB')
title(['Pendenza Rampa = ',num2str(rs.xd_ramp),' m/s'])
```

Importazione dati sperimentali

```
clc
clear all
close all

filesCRio=dir('CRio\*.txt');
TStep=1/(5000/2);
% ciclo sui files per l'estrazione dei dati
for kk=1:length(filesCRio)

    % CRio
    filename = filesCRio(kk).name;

    file = importdata (['CRio\',filename],'\t',1);
    file = file.data;

    disp('Dati caricati da:')
    disp(['CRio\',filename]);

    Set = file(:,1).*(0.05/10);
    FB_position = file(:,2).*(0.05/10);
    FB_speed = file(:,3).*(1950/10);
    Error=Set-FB_position;

    Motor.Acq{1,kk}=[Set FB_position FB_speed Error];
    Motor.Time{1,kk} = (0:size(Motor.Acq{kk},1))' * TStep;


save([cd,'\CRio\',filename(1:end),'.mat'],'Set','FB_position','FB_speed','E
rror');

    disp('Dati processati salvati in:')
    disp(['DATA\',filename(1:end),'.mat'])
    disp(' ')
    end
L=length(Set);
freq=5000/2;
in_tempo=(0:L-1)'*1/freq;
%FB_speed=FB_speed.*sign(Set-FB_position);
figure()
% plot(in_tempo,[Set FB_position FB_speed],'linewidth',2)
% grid
% %title(sprintf(filename))
% xlabel('Tempo [s]')
% ylabel('Ampiezza [m]')
% %yyaxis right
% %plot(in_tempo,FB_speed,'linewidth',2);
% %ylabel('current [A]')
% legend('Set','Feedback Posizione','Feedback velocità')
[axis,FB_pos,FB_sp]=plotyy(in_tempo,FB_position,in_tempo,FB_speed)
hold on,plot(in_tempo,Set,'k','LineWidth',2)
grid on
grid minor
set(axis(1),'FontName','Book
Antiqua','FontSize',13,'FontWeight','bold','xlim',[1,4])
set(axis(2),'FontName','Book
Antiqua','FontSize',13,'FontWeight','bold','xlim',[1,4])
```

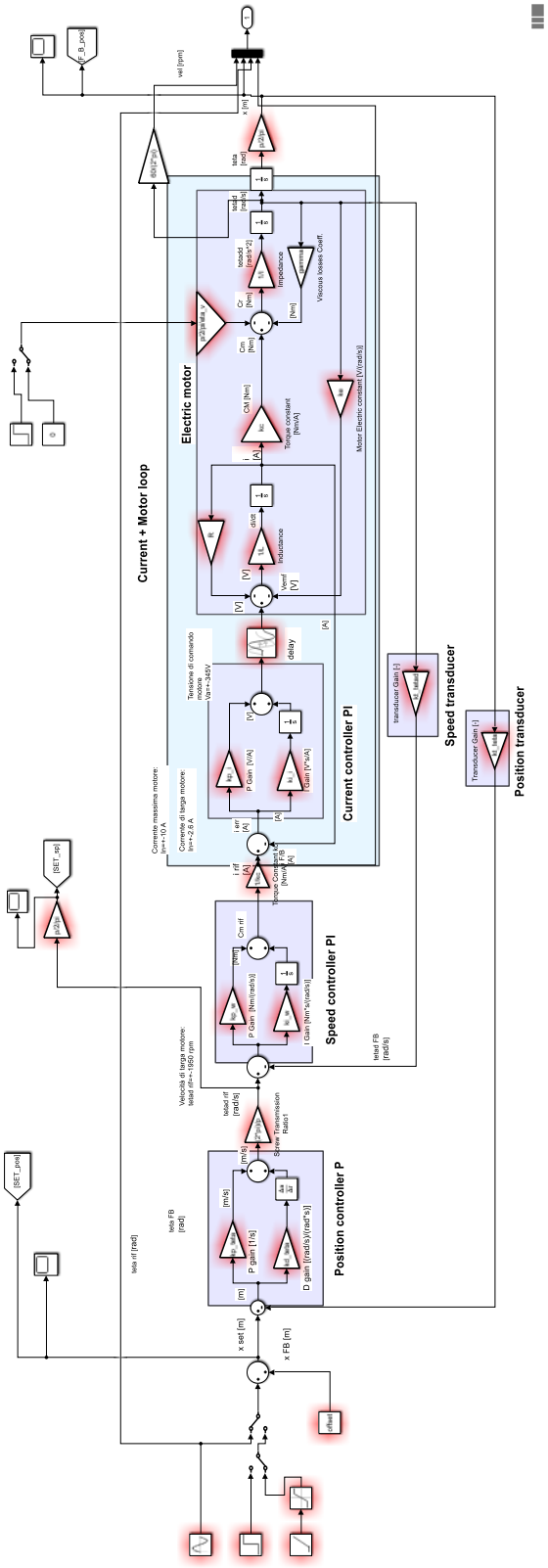
```
ylabel(axis(1),'Posizione lineare [m]','FontSize',15);
ylabel(axis(2),'Velocità angolare [rpm]','FontSize',15);
set(FB_pos,'LineWidth',2)
set(FB_sp,'LineWidth',2)

xlabel('Tempo [s]');
legend('Feedback Posizione','Set Posizione','Feedback
velocità','FontSize',13)
```

Risposta in frequenza dei dati sperimentali

```
clc
clear all
 close all

filesCRio=dir('CRio\0.5V\*.txt');
TStep=1/(5000/2);
% ciclo sui files per l'estrazione dei dati
for kk=1:length(filesCRio)

    % CRio
    filename = filesCRio(kk).name;

    file = importdata (['CRio\0.5V\',filename],'\t',1);
    file = file.data;

    disp('Dati caricati da:')
    disp(['CRio\0.5V',filename]);

    Set = file(:,1);
    FB_position = file(:,2);
    FB_speed = file(:,3);
    Error = file(:,4);

    Motor.Acq{1,kk}=[Set FB_position FB_speed Error];
    Motor.Time{1,kk} = (0:size(Motor.Acq{kk},1))' * TStep;

    save
([cd,'\CRio\0.5V\',filename(1:end),'.mat'],'Set','FB_position','FB_speed');

    disp('Dati processati salvati in:')
    disp(['Crio\0.5V',filename(1:end),'.mat'])
    disp(' ')
end

% d = fdesign.lowpass('Fp,Fst,Ap,Ast',3,5,0.5,40,100);
%     Hd = design(d,'equiripple');
%     Motor.Acq{ii}(:,1)= filter(Hd,Motor.Acq{ii}(:,1));
%     Motor.Acq{ii}(:,2)= filter(Hd,Motor.Acq{ii}(:,2));
figure
for ii = 1:numel(Motor.Acq)

    u = Motor.Acq{ii}(:,1); %set
```

```
    y = Motor.Acq{ii}(:,2);    %FB

    Fs = 5000/2;
    L = numel(u);
    % Eseguo DFT, con algoritmo FFT
    U = fft(u);
    Y = fft(y);
    % Valuto PSD (Power Spectral Density)
    P2U = abs(U/L);
    P1U = P2U(1:L/2+1);
    P1U(2:end-1) = 2 * P1U(2:end-1);
    P2Y = abs(Y/L);
    P1Y = P2Y(1:L/2+1);
    P1Y(2:end-1) = 2 * P1Y(2:end-1);

    f = Fs * (0:L/2)/L;
    subplot(3,3,ii)
    plot(f,P1U)
    hold all
    plot(f,P1Y)
    grid on
    xlabel('f (Hz)')
    ylabel('P1 (f)')
    title('Single-Sided Amplitude Spectrum of X(t)')
    xlim([0 8])

    % bode
    [~,index] = max(P1U);
    MAG(ii) = 20 * log10(Y(index)/U(index));
    PHASE(ii) = angle(Y(index)/U(index))*180/pi;
    FREQ(ii) = f(index);

end

figure
subplot(2,1,1)
semilogx(FREQ,MAG,'r-o','linewidth',2)
grid on
xlabel('Frequenza [Hz]')
ylabel('Ampiezza [dB]')
set(gca,'FontName','Book Antiqua')
set(gca,'FontSize',11)
set(gca,'FontWeight','bold','ylim',[-21 5],'xlim',[0.5 10])
subplot(2,1,2)
semilogx(FREQ,PHASE,'r-o','linewidth',2)
grid on
xlabel('Frequenza [Hz]')
ylabel('Fase [°]')
set(gca,'FontName','Book Antiqua')
set(gca,'FontSize',11)
set(gca,'FontWeight','bold','xlim',[0.5 10])
```

B-Diagramma a blocchi SIMULINK



Bibliografia

Jacazio, G. & Piombo, B., 2015. *Meccanica applicata alle macchine, Torino*: Levrotto & Bella.

Jacazio, G. & Piombo, B., 1994. *Meccanica applicata alle macchine, Vol.3, Regolazione e servomeccanismi,* Torino: Levrotto & Bella.

National Instruments, 2014. *LabVIEW Core 1 - Partecipant Guide.* s.l.:National Instruments.

National Instruments, 2014. *LabVIEW Core 2 - Partecipant Guide.* s.l.:National Instruments.

National Instruments, 2014. Embedded Control and Monitoring Using *LabVIEW - Partecipant Guide.* s.l.:National Instruments.

Sorli, M., 2010. *Dispense del corso di meccatronica.* Torino: Dipartimento di Ingegneria Meccanica e Aerospaziale.

Sorli, M. & Quaglia, G., 2003. *Meccatronica.* I a cura di Torino: Politeko.

Viktorov, V. & Colombo, F., 2013. *Automazione dei sistemi meccanici.* III ed. Torino: Clut.

M. H. Rashid, 2008, Elettronica di Potenza, Prentice Hall.

Naziha Ahmad Azli, 2011, Design of a Current Mode PI Controller for a Single-phase PWM Inverter.

Texas Instruments Europe, February 1998, Field Orientated Control of 3-Phase AC-Motors.

Michael K. Liebman, 2006, AC Motor Control in Precision Machines, Massachusetts Institute of Technology.

Mapelli, 2011, Il motore sincrono a magneti permanenti, Dispense di Azionamenti e Controllo dei Sistemi meccanici.

Doeblin Ernest, O., 2008. *Strumenti e metodi di misura.* II ed. Milano: McGraw-Hill.