UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA TRIENNALE IN INGEGNERIA MECCANICA

IMPLEMENTAZIONE DEL SISTEMA DI AZIONAMENTO DI UN ROBOT PLANARE

Implementation of the driving system of a planar robot

Tesi di laurea di:

Lorenzo Lattanzi

Relatore:

Prof. Ing. Matteo Claudio Palpacelli

Anno Accademico 2018/2019

UNIVERSITA' POLITECNICA DELLE MARCHE

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA TRIENNALE IN INGEGNERIA MECCANICA

IMPLEMENTAZIONE DEL SISTEMA DI AZIONAMENTO DI UN ROBOT PLANARE

Implementation of the driving system of a planar robot

Tesi di laurea di:

Lorenzo Lattanzi

Relatore:

Prof. Ing. Matteo Claudio Palpacelli

Anno Accademico 2018/2019

Università Politecnica delle Marche

Il vero sapiente è colui che sa di non sapere

Università Politecnica delle Marche

Via Brecce Bianche – 60131 Ancona (AN), Italia

Sommario

1 Introduzione

L'obbiettivo del lavoro svolto è l'azionamento di un motore. Nel seguente elaborato in prima analisi vengono descritti il funzionamento e le caratteristiche dell'hardware e del software scelti per lo svolgimento del problema. Successivamente viene fatta un'analisi e un'illustrazione molto generale di segnali analogici e digitali, perché era necessaria per riuscire far funzionare l'hardware scelto con il motore. Nella seconda parte dell'elaborato viene descritto il controllo e l'analisi della traiettoria del motore attraverso i codici scritti in LabVIEW. Per prima cosa viene spiegato il codice scritto nell'FPGA di rotazione del verso dell'encoder collegato al motore. Poi viene illustrata una trattazione generale del controllo PID, dei valori scelti per questo problema specifico e del codice scritto in LabVIEW. In ultima analisi viene definito e spiegato il codice della traiettoria di posizione del motore ed i risultati ottenuti del controllo dell'intero sistema. Nella parte finale dell'elaborato viene anche fatta una trattazione generale del calcolatore MATLAB e viene esposta e commentata la cinematica diretta e indiretta di un robot 2R.

2 Architettura di controllo

Per il controllo della posizione di un motore abbiamo bisogno di una specifica architettura, composta sia da hardware che da software. Ovviamente le tipologie che si possono usare sono molteplici, nel nostro caso in particolare si è deciso di usare un'architettura di questo tipo.

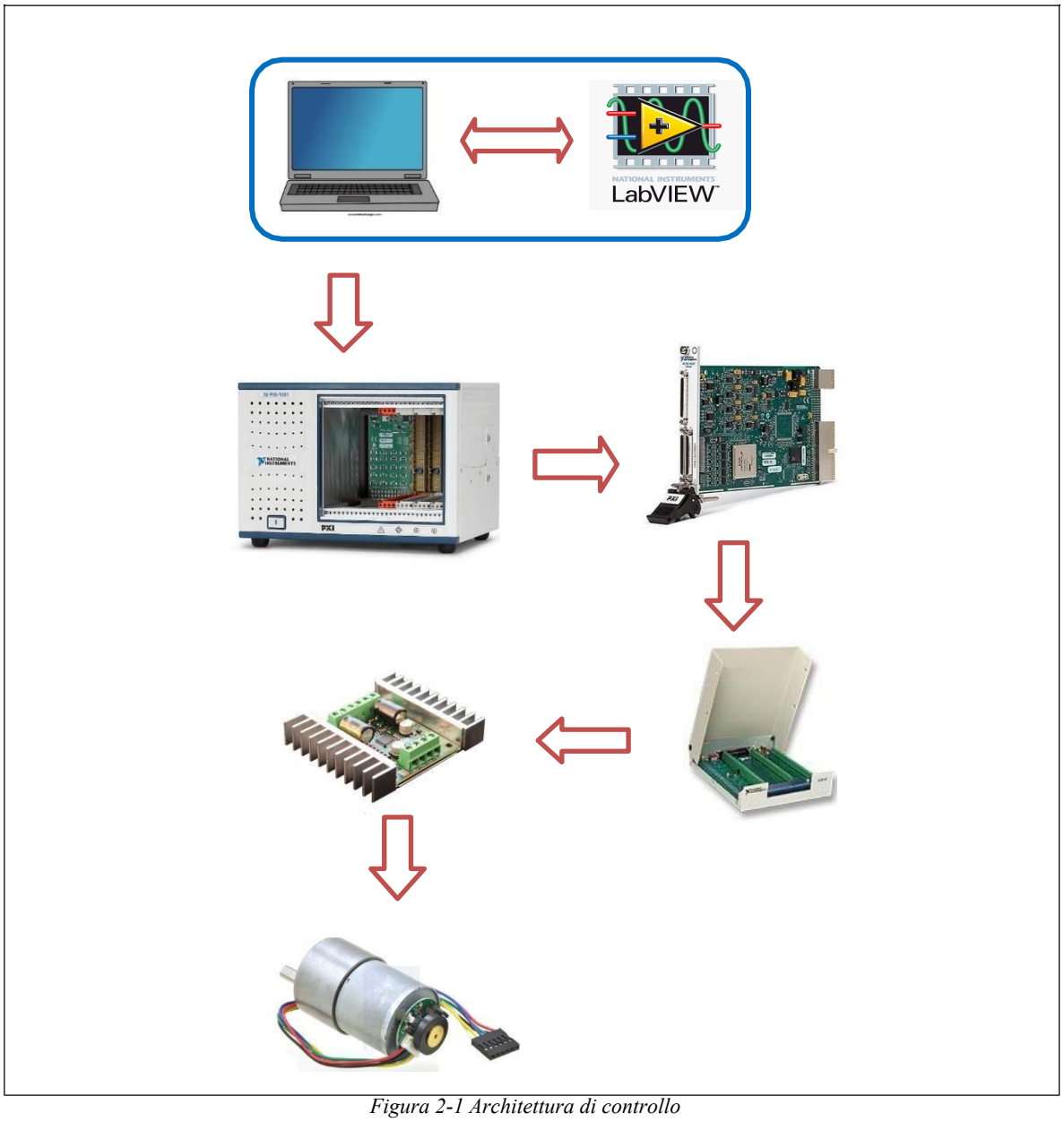


Figura 2-1 Architettura di controllo

2.1 . Hardware

Il cervello di tutto il sistema di controllo è rappresentato dall'elaboratore NI PXI-8187 dotato di sistema operativo Phar Lap che consente di utilizzare il modulo Real-Time di LabVIEW. La macchina nel suo insieme è composta da diversi moduli che assolvono a specifiche funzioni tra cui l'acquisizione dati e la comunicazione con eventuali altri elaboratori; il tutto è racchiuso in uno chassis industriale.

Figura 2-2 chassis PXI 1031

Il sistema implementato è così composto:

- Chassis NI PXI 1031
- Controller NI PXI 8187
- Scheda I/O riconfigurabile NI-PXI 7833 R
- Terminaleria SCB-68
- Sabertooth 2 x 12

2.2 Chassis NI PXI 1031

Lo chassis PXI-1031 prodotto da National Instruments dispone di slot adatti ad accomodare schede PXI o Compact PCI in formato 3U. È progettato per assolvere i compiti richiesti da una grande famiglia di applicazioni, dalla misura al controllo. I bus PXI presentano uno standard maturo ed elevate prestazioni in campo sia scientifico che industriale, con la possibilità di scegliere tra due diversi clock, 1 KHz e 1 MHz. Questo consente di ottenere una risoluzione massima di 1 μs e di eseguire software con frequenza di ciclo fino 1 MHz. Inoltre, sono presenti 8 linee di interrupt hardware disponibili su tutte le schede collegate al bus, necessarie per sincronizzare tutte quelle attività che, come le acquisizioni e le attuazioni, richiedono l'intervento coordinato di più moduli, con sfasamenti inferiori al microsecondo. Infine, da non trascurarsi, i bus PXI garantiscono migliori prestazioni in ambiente Lab VIEW, nel quale è stato implementato il controllo motore. Lo chassis è ottimizzato per fornire adeguato raffreddamento ai moduli in esso contenuti tramite due ventole di sistema che forniscono raffreddamento forzato e filtrato. Le ventole sono comandate secondo due schemi di funzionamento: HIGH o AUTO. In modalità HIGH le ventole vanno alla massima velocità operativa per fornire il massimo potenziale di raffreddamento mentre in posizione AUTO il sistema adatta dinamicamente il regime delle ventole per garantire un raffreddamento ottimale e, al contempo, un buon comfort acustico all'operatore. Il clock generato ha una frequenza di riferimento di 10MHz e un'accuratezza di 25 ppm (parti per milione). Questo si traduce in una fluttuazione inferiore a 5ps e a uno sfasamento slot-to-slot inferiore a 250 ps. Il sistema di alimentazione è rimovibile dotato di una robusta protezione con le sovratensioni e funziona correttamente con le principali reti elettriche, a 110 o 220 V, 50 o 60 Hz. La linea a 12 V che alimenta le ventole è isolata al minimo per ridurre il rumore La linea a 12 V che alimenta le ventole è isolata al minimo per ridurre il rumore elettrico sul bus.

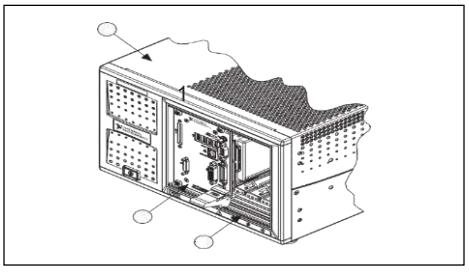


Figura 2-3 chassis PXI 1031

Un sistema di monitoraggio remoto permette di tenere sotto controllo i valori delle tensioni e delle temperature tramite un connettore DB-9 presente sul retro. Un led di stato posto sull'interruttore di alimentazione indica un eventuale mal funzionamento del modulo di alimentazione.

2.3 NI PXI 8178

Il controller PXI-8187 prodotto da National Instruments incorpora un processore Intel Pentium 4 a 2.2 GHz di frequenza, con 1 MB di DDR RAM e 40 GB di disco fisso e può essere usato in qualsiasi sistema PXI o CompactPCI. Esso è particolarmente indicato per applicazioni di analisi o per lo sviluppo di sistemi di controllo su bus PXI in tempo reale poiché capitalizza al meglio le prestazioni del processore Intel ai benefici in termini di temporizzazione spinta e sincronizzazione derivanti dalla tecnologia PXI. Insieme allo chassis dedicato forma infatti una piattaforma compatta ad alte prestazioni per applicazioni modulari di acquisizione, analisi dati e controllo. Un'efficiente razionalizzazione dello spazio concesso al modulo dalle specifiche PXI ha permesso ai progettisti di inserire tutte le periferiche standard di un PC in una sola unità molto compatta. Questo alto livello di integrazione ha come effetto principale quello di lasciare liberi tutti gli slot dello chassis all'utente, ovviando in tal modo alla necessità di provvedere a complessi e costosi cablaggi. Il modulo incorpora tutte le periferiche di I/O standard: Ethernet 10/100 Base TX, tastiera, mouse e video. Inoltre, dispone di una connessione GPIB (GeneraI Purpuse Inerface Bus, IEEE 488.2) che permette la connessione di strumentazione esterna. Due porte USB 2.0 permettono la connessione di lettori e masterizzatori CD/DVD, stampanti, dischi di memoria, ecc. completano la configurazione una porta parallela ECP/EPP (Extended Capabilities Port/Enhanced Parallel Port) e due porte seriali RS232. Il controller include una connessione 5MB esterna utilizzabile come input/output trigger o watchdog timer attraverso cui è possibile trasferire i segnali di interrupt presenti sul bus PXI all'esterno o fornire segnali di triggering al bus stesso. L'interfaccia video integrata è basata sul chipset Intel Extreme Graphics ed è capace di elevate prestazioni sia in 3D che in 2D: questa architettura, basata sulla condivisione della memoria di sistema, richiede il perfetto bilanciamento delle risorse, in termini soprattutto di RAM. La memoria volatile di sistema è del tipo double-data-rate (DDR) SDRAM, che offre un'ampiezza di banda doppia rispetto alle memorie convenzionali. Questa caratteristica rende il controller ideale per applicazioni analisi dati particolarmente intense. La configurazione base prevede 256Mb di RAM espandibili a 1 024Mb. Il modulo è fornito con un set di software preinstallato e pronto all'uso:

- Microsoft Windows XP Professional o Windows 2000;
- Immagine hard drive per il ripristino del sistema;
- Drivers per NI-VISA e NI-488 (GPIB);
- Drivers per le periferiche installate.

Il controller può essere avviato con il comune sistema operativo Windows XP o in modalità Phar Lap per un'esecuzione molto più affidabile e prestante delle applicazioni real time, implementate con Lab VIEW Real Time e successivamente caricate sul controller tramite rete Ethernet. Il controller può essere avviato con il comune sistema operativo Windows XP o in modalità Phar Lap per un'esecuzione molto più affidabile e prestante delle applicazioni real time, implementate con LabVIEW Real Time e successivamente caricate sul controller tramite rete Ethernet.

CARATTERISTICHE		
Computazionali		
Processore	2.2 GHz Pentium 4-M	
Ethernet	10/100 TX. Connettore RJ45	
Video	Intel Extreme Graphics	
Porta seriale	2RS232	
Porta parallela	IEEE 1284 connttore tipo C	
GPIB	PCI-GPIB/TNT, connettore micro D25, IEEE488	
USB	2 versione 2.0	
Tastiera/Mouse	Connettori PS/2	
RAM	1 slot SO-DIMM, 256Mb espandibili a 1024Mb	
Hard drive	30GB minimo, 2.5”, intrfaccia Ultra ATA100	
Assorbimento corrente elettrica		
Tensione (V)	Tipico (A)	Massimo (A)
+3.3	4.0	5.0
+5	6.5	8.0
+12	0.15	0.2
-12	0	0
Fisiche		
Dimensioni	PXI 3U, 8.1x13x21.6 cm	
Peso	1.0 Kg	
MTBF	170248 ore	
Ambiente operativo		
Temperatura	5-40°C	
Umidità relativa	10-90% non condensante	

Tabella 1 caratteristiche PXI 8178

2.4 Scheda di I/O riconfigurabile NI PXI 7833R

Le schede NI PXI-7831R di National Instruments sono equipaggiate con una tecnologia che permette di disporre di hardware I/O riconfigurabile. Il cuore di ognuno dei moduli è un FPGA prodotto da Xylinx, le cui porte, slices e celle assumono una configurazione hardware determinata dal compilatore che traduce il codice prodotto con un modulo speciale di LabVIEW. È possibile personalizzare il comportamento del dispositivo sfruttando le seguenti caratteristiche

Figura 2-4 FPGA 7833R

- controllo sulla temporizzazione e sincronizzazione con risoluzione di 25ns;
- esecuzione di codice con frequenze di ciclo di 40MHz;
- 96 linee di I/O digitali completamente configurabili come input, output, timer, PWM (Pulse Width Modulation), encoder input e protocolli definiti dall'utente;
- disponibilità di 8 ingressi analogici con risoluzione di 16bit e frequenza di campionamento di 200 Ksample/s ognuno;
- presenza di 8 uscite analogiche con risoluzione di 16bit e frequenza di aggiornamento di 1 Msample/s ciascuno.

Unendo queste potenzialità a quelle offerte dalla piattaforma Lab VIEW Real Time è possibile realizzare applicazioni come:

- PWM e protocolli di comunicazione digitali;
- Simulazione Hardware;
- Rapid Control Prototyping;
- controllo in tempo reale in ambito analogico o digitale.

Ogni linea digitale è configurabile indipendentemente dalle altre e memorizzabile nella memoria flash del dispositivo per assicurare un ben definito stato all'avvio. Uno speciale modulo software, LabVIEW FPGA Module, permette la programmazione in ambiente grafico di tutte le funzionalità della scheda, con un approccio molto simile a quello del normale Lab VIEW per PC. Tuttavia, non tutte le funzionalità di LabVIEW sono implementabili (ad esempio sono possibili solo calcoli con numeri interi).

CARATTERISTICHE			
Ingressi analogici		**I/O digitale**	
Num.Canali	8	Num.Canali	96
Modalità acquisizione	Diff/SE	Compatibilità	TTL
Risoluzione	16 bit	Corrente uscita Max.	±5 mA
Tempo conversione	4 µs	Tensione ammessa Max.	-0.5 7V
Freq. Max. campionamento	200 kS/s		
Impedenza ingresso	10 GΩ	**FPGA riconfigurabile**	
Tensione campionata	±10 V		
Accoppiamento	DC	Gates di sistema	1M
Tensione Max. ammessa	±42 V	Num. Slices logici	5120
		Num. Celle equivalenti	11520
		RAM disponibile	80 Kbytes
		Clock	40 MHz
Uscite analogiche		**Fisiche**	
Num.Canali	8	Dimensioni	16x10 cm
Risoluzione	16 bit	Connettori	3
Tempo aggiornamento	1 µs	Pin	68 VHDCI
Freq. Max. aggiornamento	1MS/s		
Tipo di DAC	Enh R-2R	**Ambientali**	
Range	±10 V		
Accoppiamento	DC	Temperatura	0-55°C
Impedenza di uscita	1.25 Ω	Umidità relativa	10-90% n.c.
Corrente Max.	±2.5 mA		

Tabella 1.2 Caratteristiche FPGA 7833

2.5 SCB 68

SCB-68 è un blocco connettore I/O schermato con 68 morsetti a vite per un facile collegamento del segnale a un dispositivo DAQ National Instruments a 68 o 100 pin. L'SCB-68 presenta un'area di breadboard generale per circuiti e prese personalizzati per l'interscambio di componenti elettrici. Queste prese o piastre componenti consentono il filtraggio, la misura dell'ingresso di corrente da 4 a 20 mA, il rilevamento della termocoppia aperta e l'attenuazione della tensione. I pad a componenti aperti consentono di aggiungere facilmente il condizionamento del segnale ai segnali di ingresso analogico (AI), uscita analogica (AO) e PFI 0 di un dispositivo DAQ a 68 o 100 pin.

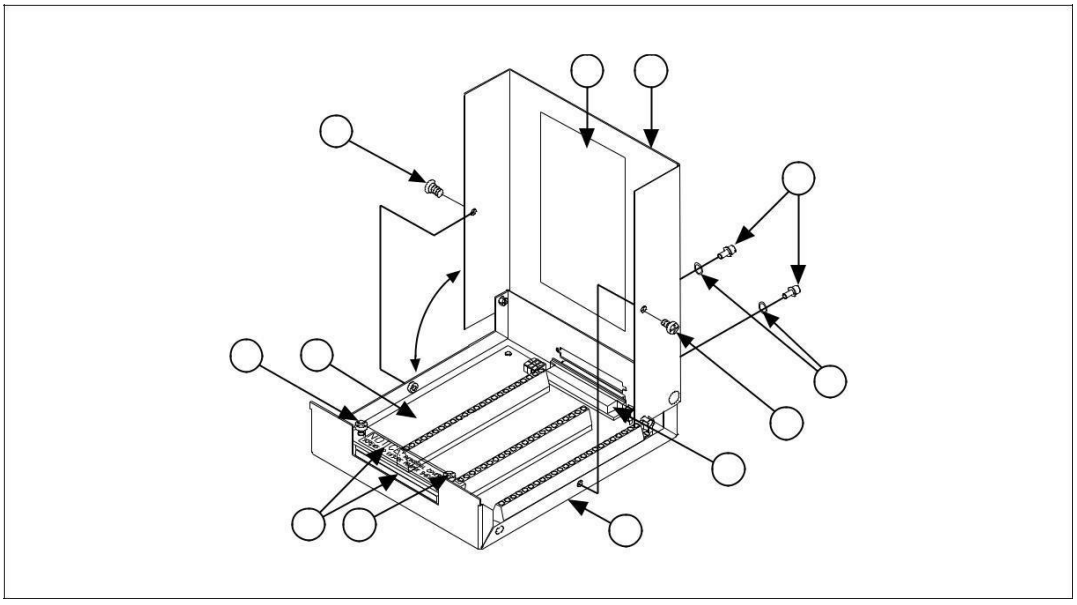


Figura 2-5 SCB-68

L'SCB-68 ha 68 terminali a vite per connessioni di segnale I/O. Per utilizzare l'SCB-68 con NI 78xxR, è necessario configurare l'SCB-68 come blocco connettore generico.

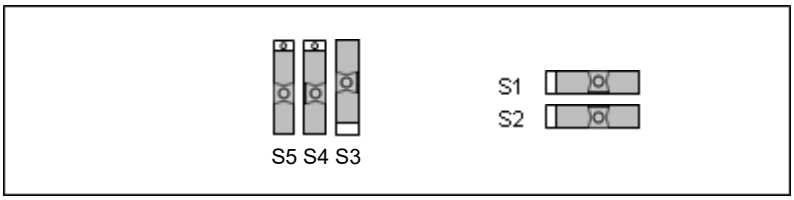


Figura 2-6 switch configurazione SCB-68

Fare riferimento alla *figura 2-6* per la configurazione dello switch generico. Dopo aver configurato gli interruttori SCB-68, è possibile collegare i segnali I/O ai terminali a vite SCB-68. Dopo aver collegato i segnali I/O ai terminali a vite SCB-68, è possibile collegare l'SCB-68 al Connector 0 per avere a disposizione sia terminali analogici che digitali, oppure posso collegarlo agli altri 2 connettori con i quali si ha a disposizione solamente terminali digitali. Come è rappresentato in figura.

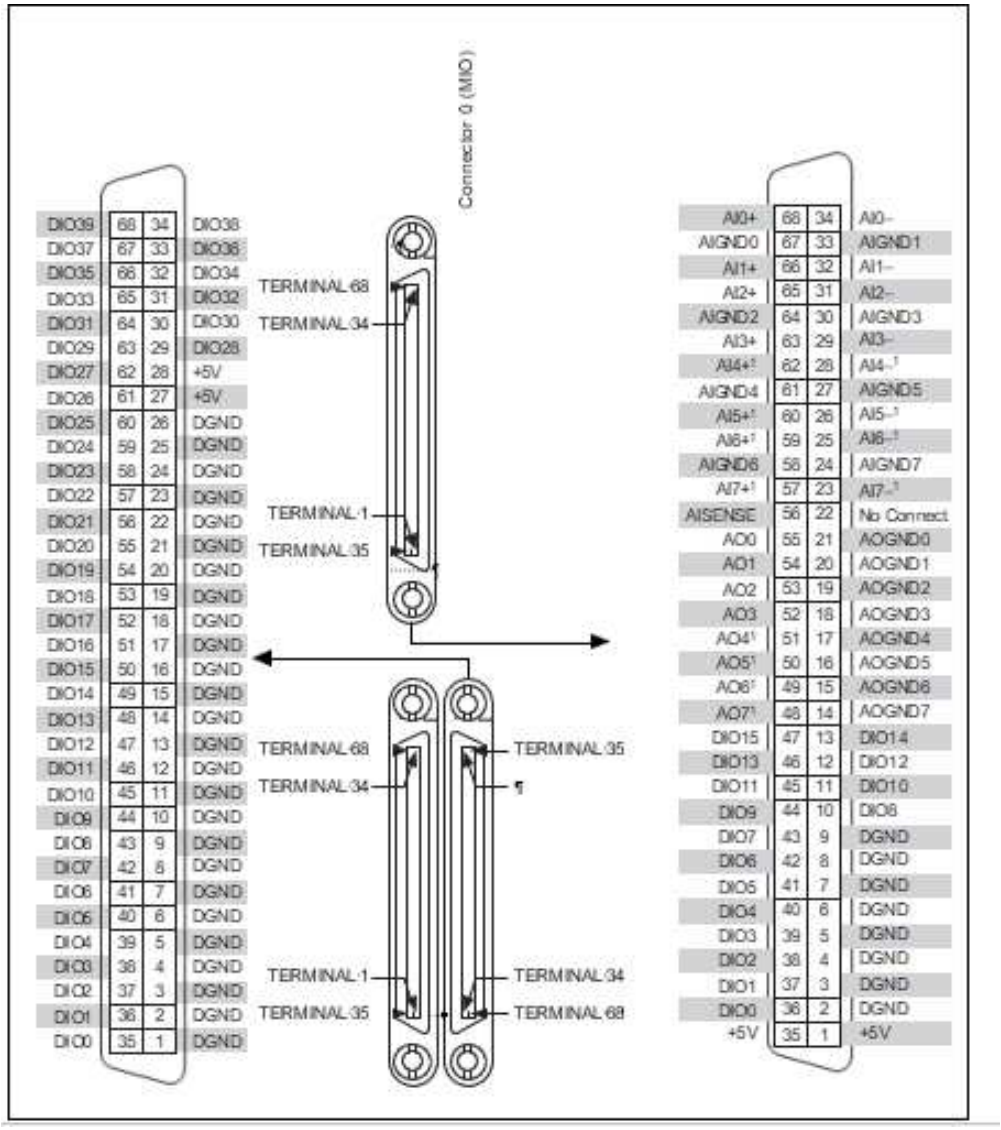


Figura 2-7 connettori e terminali SCB-68

2.6 Sabertooth 2x12

La Sabertooth è una componente hardware fondamentale per il controllo del mio motore. In entrata c'è il voltaggio che esce dall'SCB mentre in uscita c'è il voltaggio necessario per muovore il motore nella posizione desiderata. In seguito viene descritto il funzionamento della sabertooth per il controllo di un motore elettrico.

Motor Terminals

Il motore 1 è collegato ai terminali M1A e M1B, come illustrato di seguito. Se il motore funziona nel modo opposto a quello desiderato, è possibile invertire i fili del motore per invertire la rotazione.

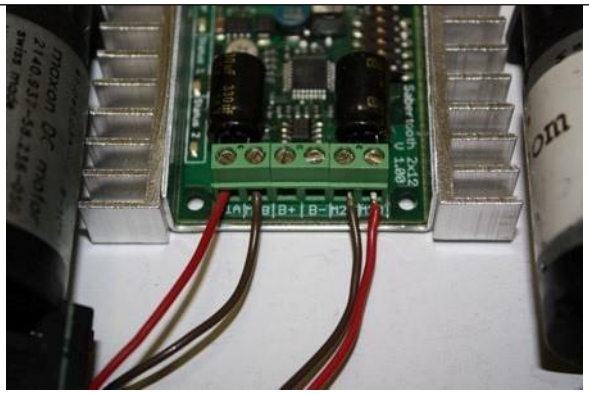


Figura 2-8 sabertooth

:

Signal Input

Terminals S1 S2 0 V

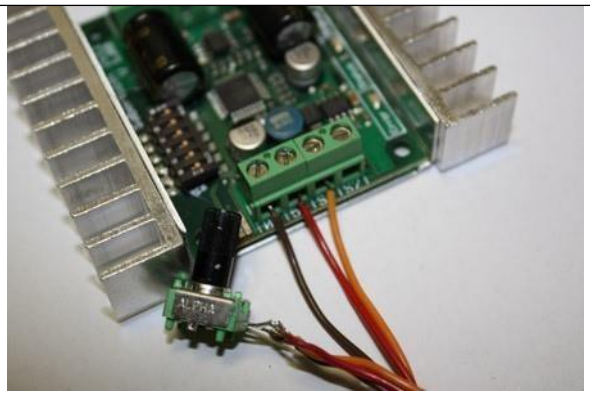


Figura 2-9 sabertooth

I segnali di ingresso che controllano il Sabertooth sono collegati ai terminali S1 e S2, nel nostro caso abbiamo solo un segnale, quindi si utilizza solo S1. È il segnale che esce dalla SCB-68; l'output analogico che varia in base alla posizione che voglio del motore. La connessione 0V è il segnale di terra per il Sabertooth. Per ricevere correttamente i segnali di ingresso, deve essere collegato allo stesso GND del dispositivo che invia i segnali.

:

Terminali Batteria B+ B-

La batteria o l'alimentazione è collegata ai terminali B e B. B- si collega al lato negativo della batteria (di solito nero). B si collega al lato positivo della batteria (di solito rosso o giallo). Di solito è meglio collegare la batteria attraverso un connettore invece che direttamente al motore. L'alimentatore usato in questo caso è un 12 V. In qualsiasi caso il segnale analogico in entrata che corrisponde al non movimento del motore è 2.5 V, per ovviare a questo problema in LabVIEW FPGA si è aggiunto un segnale corrispondente a 2.5 V in bit.

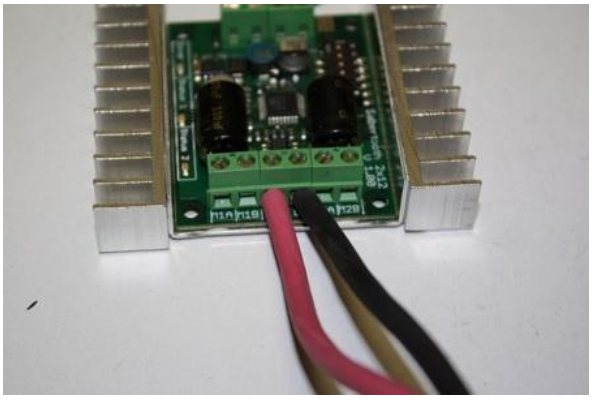


Figura 2-10 sabertooth

3 Software

Il Software che si è scelto di utilizzare per l'implementazione del motore è LabVIEW 16, un linguaggio di programmazione grafico sviluppato appositamente dalla National Instruments, studiato e ottimizzato per facilitare la realizzazione di applicazioni per acquisizione, analisi di segnali e controllo dati, compatibile con gran parte dell'hardware oggi disponibile per questa tipologia di applicazioni. LabVIEW è costituito da tre diversi moduli (Lab IEW for Windows, LabVIEW Real Time Module, LabVIEW FPGA Module) che si differenziano tra loro per il supporto hardware a cui è destinata l'applicazione da realizzare (PC, Controller Real Time, Scheda FPGA) e hanno caratteristiche differenti soprattutto per quanto riguarda le tempistiche con cui vengono eseguite le varie applicazioni e per la complessità delle librerie messe a disposizione.

3.1 LabVIEW for Windows

In contrasto con i linguaggi di programmazione testuali, LabVIEW sfrutta una programmazione grafica ad icone basata sul flusso dei dati, che permette di realizzare elaborati programmi di analisi o di controllo senza scrivere fisicamente alcuna riga di programma. Infatti, i programmi realizzati con LabVIEW sono detti *Virtual Instrument (VI)* in quanto nell' aspetto fisico e nel modo di interagire riproducono strumenti reali come oscilloscopi e multimetri mentre si tratta di *oggetti virtuali*. L'applicazione è

costituita da icone (la cui funzione è espressa nella finestra "Help Context") collegate tra loro da "cavi" virtuali che trasportano le informazioni che assumono una forma e un colore differente a seconda del tipo di dato trasportato (numero, stringa di testo, matrice, booleano, etc.). Da notare che a differenza degli elaborati scritti secondo un codice testuale tipici dei linguaggi di programmazione tradizionali, la programmazione grafica di LabVIEW semplifica enormemente l'apprendimento e la scrittura dei programmi da parte dell'utente; inoltre qui è il flusso dei dati a determinare l'ordine di esecuzione dei vari blocchi nel programma, e non una sequenza di istruzioni. Un blocco di elaborazione non è altro che la rappresentazione di una serie di operazioni che vengono eseguite sui dati in ingresso a seconda della funzione del blocco stesso: può ad esempio essere una semplice operazione matematica o di un intero programma a sé stante, poiché, come nei linguaggi di programmazione tradizionali, un programma può richiamare una *sub*-routine. L'interfaccia con l'utente di ogni VI è il Front Panel in cui sono posizionati controlli e indicatori che rappresentano rispettivamente gli input e gli output interattivi del VI. I controlli sono manopole, potenziometri, quadranti, pulsanti, interruttori, comandi scorrevoli, caselle numeriche o di testo e altri meccanismi di introduzione di dati, mentre gli indicatori sono grafici, LED, tabelle e altri componenti che consentano di visualizzare gli output acquisiti o generati dal *Block Diagram*. Dopo aver realizzato il pannello di controllo, è necessario implementare le funzionalità richieste dall'applicazione nello schema a blocchi, unendo ed elaborando le icone che rappresentano gli oggetti del pannello frontale. Quest'ultima parte è il corpo centrale di un VI in quanto contiene il codice grafico, sotto forma di diagramma a blocchi, che gestisce gli oggetti presenti nel pannello di controllo. Nella realizzazione del codice si possono utilizzare, oltre ai capisaldi della programmazione classica come i cicli while, for e la struttura case, gli elementi messi a disposizione dalle librerie di LabVIEW come le strutture temporizzate (*Timed Structures*) che permettono di stabilire una priorità tra le varie parti del programma o funzione già implementate, le cosiddette *Buit-in functions*, per l'acquisizione, l'analisi e l'esposizione di dati; una volta terminata anche questa operazione è possibile personalizzare l'icona del VI in alto a destra, attribuendo al file stesso il numero di connettori desiderato, per poi poterlo utilizzare in seguito come subVI. L'enorme flessibilità di LabVIEW risiede anche nella possibilità di stabilire una scala gerarchica e una priorità dei VI, consentendo questo modo di realizzare strutture anche molto complicate senza mai perdere di vista il quadro di insieme. rispettivamente gli input e gli output interattivi del VI. I controlli sono manopole, potenziometri, quadranti, pulsanti, interruttori, comandi scorrevoli, caselle numeriche o di testo e altri meccanismi di introduzione di dati, mentre gli indicatori

sono grafici, LED, tabelle e altri componenti che consentano di visualizzare gli output acquisiti o generati dal Block Diagram.

3.2 LabVIEW Real Time

L'applicazione realizzata con Lab VIEW Real Time differisce da una realizzata per Windows per il fatto che l'elaboratore RT comunica con il mondo esterno tramite un'interfaccia di rete oltre alle schede di input-output. Quindi, sfruttando questo canale di trasmissione dati, è possibile modificare tutti i parametri del sistema RT, con la necessità però di utilizzare un computer esterno, definito Host, che visualizza l'interfaccia con l'utente, quindi aggiorna i dati in tempo reale e invia al modulo real time le modifiche effettuate. Questo espediente consente di al1eggerire il processore dal compito di visualizzare e aggiornare l'interfaccia, liberando più risorse di calcolo. In campo RT è possibile inoltre attribuire diverse priorità a ciascun VI a seconda della loro criticità temporale, cioè dell'intervallo di tempo entro i quali le applicazioni devono essere eseguite. In questo modo il processare, nel caso in cui siano richieste risorse maggiori rispetto a quelle disponibili, da precedenza ai VI definiti ad alta priorità, e quindi più critici. Il modulo FPGA utilizza un approccio diverso rispetto al RT, definito hard real time, in cui il programmatore è celio, a priori, che l'esecuzione del programma avvenga entro tempi prestabiliti, in quanto l'algoritmo è implementato in hardware. La scheda utilizzata per questo compito è una NI PXI-7833R, la quale è dotata di un FPGA e nella quale viene codificato in hardware il modello realizzato In LabVIEW, garantendo in questa maniera il determinismo della applicazione. alle schede di input-output. Quindi, sfruttando questo canale di trasmissione dati, è possibile modificare tutti i parametri del sistema RT, con la necessità però di utilizzare un computer esterno, definito Host, che visualizza l'interfaccia con l'utente, quindi aggiorna i dati in tempo reale e invia al modulo real time le modifiche effettuate. Questo espediente consente di al1eggerire il processore dal compito di visualizzare e aggiornare l'interfaccia, liberando più risorse di calcolo. In campo RT è possibile inoltre attribuire diverse priorità a ciascun VI a seconda della loro criticità temporale, cioè dell'intervallo di tempo entro i quali le applicazioni devono essere eseguite. In questo modo il processare, nel caso in cui siano richieste risorse maggiori rispetto a quelle disponibili, da precedenza ai VI definiti ad alta priorità, e quindi più critici. In LabVIEW, garantendo in questa maniera il determinismo della applicazione. Il modulo FPGA mantiene la stessa facilità d'uso e intuitività dell'ambiente LabVIEW, ma è dotato di un nuovo set di VI che consentono di realizzare applicazioni funzionanti sulla scheda utilizzata. Questo dispositivo è limitato però al calcolo intero a 64 bit, quindi non è possibile implementare VI complessi. Ciò è dovuto al fatto che, per rendere il più veloce possibile l'esecuzione delle applicazioni, si

è reso necessario ridurre notevolmente la complessità dei calcoli che vengono effettuati. Inoltre, altre limitazioni nascono dalla limitata disponibilità di RAM e di gales (elementi logici attivi). La realizzazione del VI per il FPGA non è molto differente da quella abituale per uno di Lab VIEW in ambiente Windows, l'unica eccezione consiste nel dover compilare tale VI nel linguaggio del FPGA (binario) e poi caricarlo nella sua memoria. Questo processo può però richiedere anche alcune ore e deve essere eseguito ogni volta che si modifichi il flusso di dati all'interno del VI. Il software del sistema di controllo gestisce la comunicazione tra vari VI. Una parte di questa viene eseguita sull' elaboratore real time, l'altra sulla scheda del FPGA. Inoltre, è necessario un ulteriore VI, definito Host, che gira su di un altro elaboratore dotato di sistema operativo (ad esempio Windows) e di scheda di rete, che fornisce l'interfaccia utente per la gestione del sistema RT.

4 Analisi del Segnale

4.1 Segnale analogico

il segnale è detto *analogico* quando i valori utili che lo rappresentano sono in stretta "analogia" con il fenomeno che li genera e spesso sono *continui* (infiniti). Cioè se prendessimo in esame un intervallo spaziotemporale A - B (tipo quello rappresentato da un potenziometro ed i suoi relativi valori MIN(A) e MAX(B)) si passerebbe da Min. a MAX per una infinità di mutazioni elettriche, non numerabili in R (dal latino *continuum* = congiunto, unito insieme) e ciascuna in diretta "analogia" con la posizione dell'asse del potenziometro. *Analogico* si contrappone a *digitale* (=discreto). Analogico significa "continuo", "non discreto".

È possibile campionare i canali NI 783*x*R contemporaneamente o a velocità diverse. La modalità di input è configurabile dal software e l'intervallo di input è fisso a 10 V. I convertitori restituiscono dati nel formato complemento a due. La tabella 2-1 mostra il codice di uscita ideale restituito per una determinata tensione AI

Input Description	**AI Voltage**	**Output Code (Hex) (Two's Complement)**
Full-scale range –1 LSB	9.999695	7FFF
Full-scale range –2 LSB	9.999390	7FFE
Midscale	0.000000	0000

Tabella 4.1 Caratteristiche segnale analogico FPGA

Input Description	AI Voltage	Output Code (Hex) (Two's Complement)
Negative full-scale range +1 LSB	–9.999695	8001
Negative full-scale range	–10.000000	8000
Any input voltage	$\frac{Output\ Code}{32{,}768} \times 10.0\ V$	—

Tabella 4.2 Caratteristiche segnale analogico FPGA

L'intervallo NI 783*x*R/784*x*R/785*x*RAI è fissato a 10 V. In LabVIEW FPGA il collegamento di un segnale analogico in entrata e in uscita attraverso un filo di rame che connette i due terminali. è come in figura.

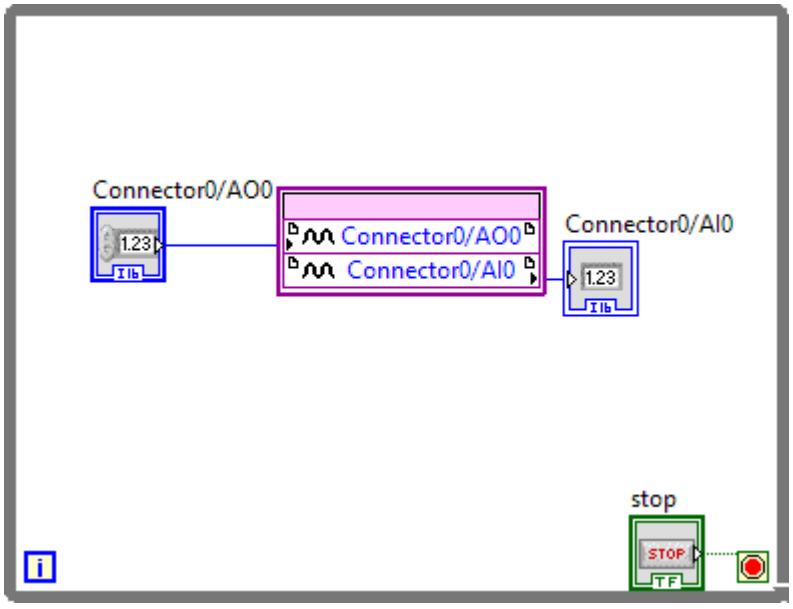


Figura 4-1 Block Diagram segnale analogico

Figura 4-2 Front Panel segnale analogico

4.2 Segnale digitale

Il segnale digitale è un segnale che viene utilizzato per rappresentare i dati come una sequenza di valori discreti; in qualsiasi momento può assumere solo uno di un numero finito di valori. Ciò contrasta con un segnale analogico, che rappresenta continuo valori; in qualsiasi momento rappresenta numero reale all'interno di un intervallo continuo di valori.

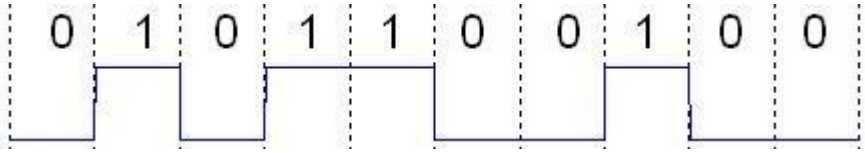


Figura 4-3 segnale digitale

In LabVIEW FPGA al contrario dei segnali analogici, quelli digitali si comportano sia come input che come output.

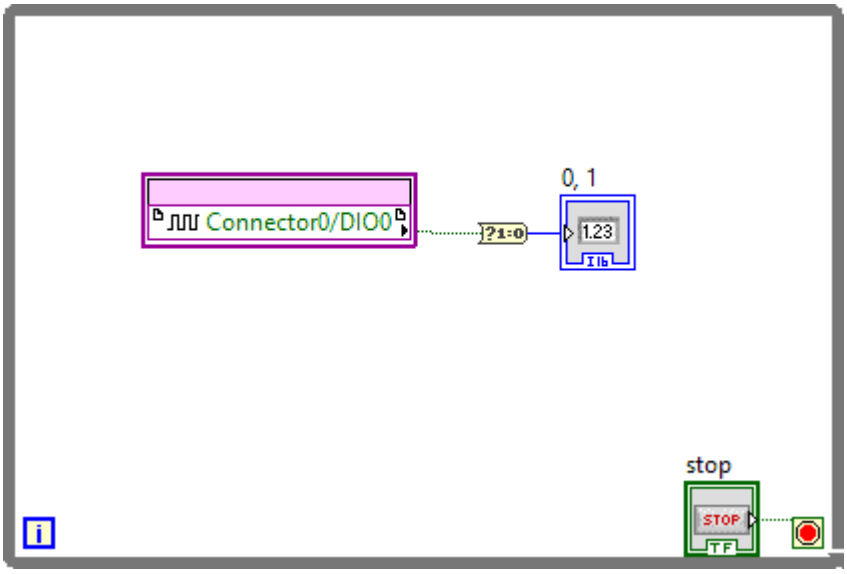


Figura 4-4 Block Diagram segnale digitale

Figura 4-5 Front Panel segnale digitale

5 Controllo e analisi di traiettoria del motore

5.1 50:1 Metal Gearmotor 37Dx70L mm with 64 CPR Encoder

Dopo aver preso confidenza con i vari segnali, prima quelli analogici e poi quelli digitali, si è passati al vero obbiettivo della tesi, ovvero fare il controllo di un motore elettrico, in particolare il "50:1 Metal Gearmotor 37Dx70L mm with 64 CPR Encoder". L'encoder associato al motore è un effetto Hall a due canali (A e B), viene utilizzato per rilevare la rotazione di un disco magnetico su una protrusione posteriore dell'albero del motore. L'encoder quadrato fornisce una risoluzione di 64 conteggi per rivoluzione dell'albero del motore quando si contano entrambi i bordi di entrambi i canali.

Colore	Funzione
Rosso	motore (si collega a un terminale motore)
Nero	motore (si collega all'altro terminale del motore)
Verde	Encoder GND
Blu	encoder Vcc (3.5 – 20 V)
Giallo	encoder A output
Bianco	encoder B output

Figura 5-1 encoder effetto Hall

Dopo aver preso conoscenza del funzionamento e dell'encoder a disposizione, si è scritto il codice del verso di rotazione dell'encoder, associato al motore, in LABIVEW FPGA da far girare a saturazione nell' FPGA NI 7833R.

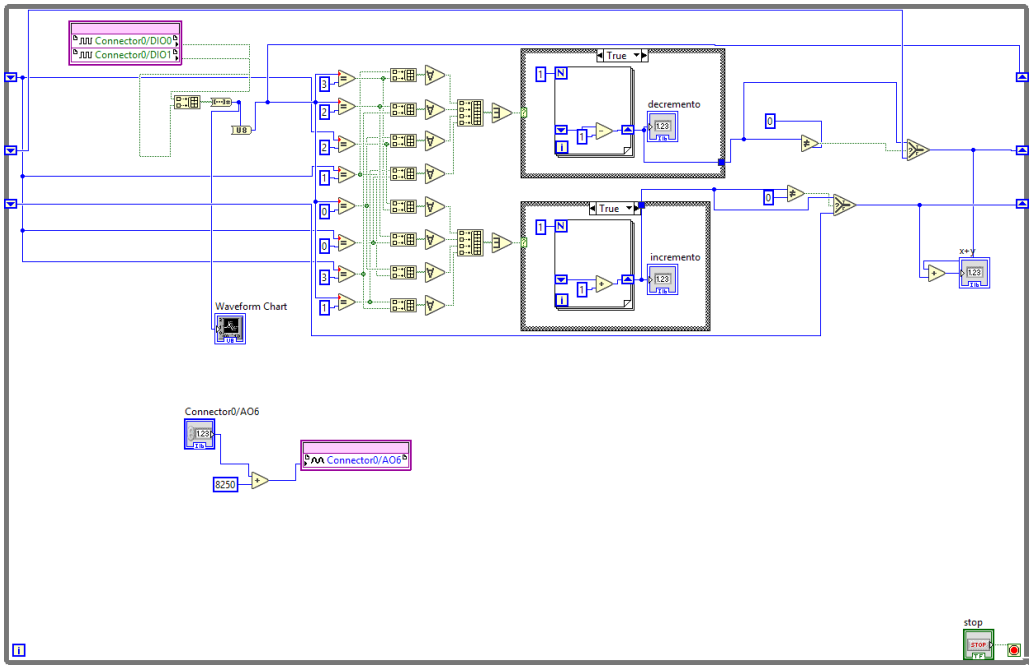


Figura 5-2 Block Diagram rotazione encoder

.

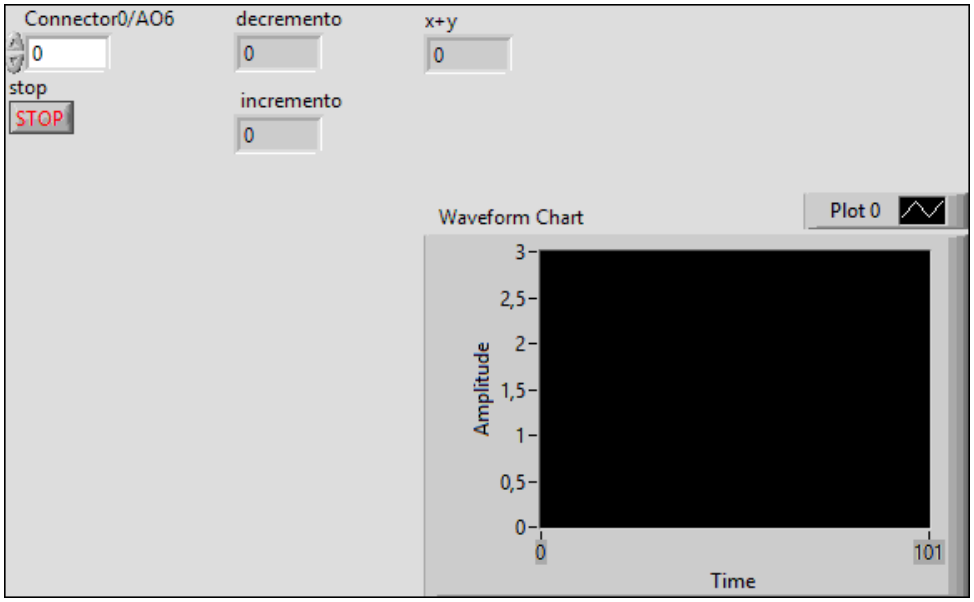


Figura 5-3 Front Panel rotazione encoder

Nelle due immagini è raffigurato il codice rispettivamente nel “Block diagram” e nel “Front Panel” in LabVIEW FPGA. Per scrivere il codice della rotazione (verso destra o verso sinistra) era necessario conoscere le varie posizioni che potessero assumere i due segnali digitali dell’encoder (sfasati di 90°)

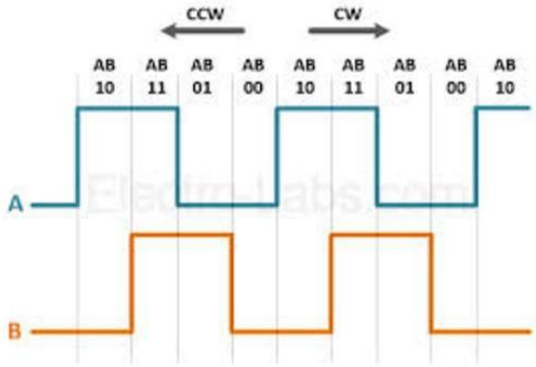


Figura 5-4 schema segnale encoder a due canali

In particolare, conoscendo il segnale attuale e confrontandolo con quello precedente si riesce a capire il verso di rotazione. Per facilitare la scrittura del codice si è passati da 2 segnali (0,1) ad un segnale (0,1,2,3) utilizzando il sistema binario.

SEGNALE DIGITALE ENCODER		SISTEMA BINARIO
A	B	
0	0	0
0	1	1
1	0	2
1	1	3

Tabella 2 sistema binario per segnale encoder a 2 canali

In LabVIEW per confrontare un segnale attuale con uno precedente si utilizza uno shift register all'interno di un ciclo while, senza usare delle condizioni in modo da far girare il ciclo alla frequenza massima di lavoro dell'FPGA.

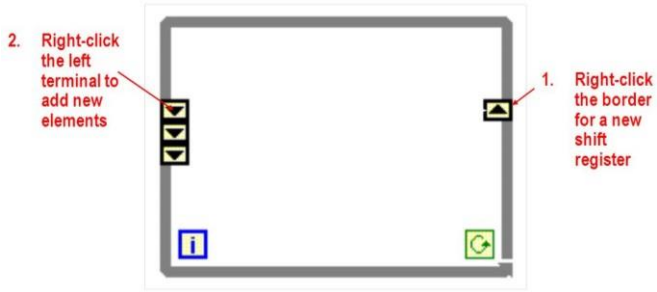


Figura 5-5 shift register LabVIEW

Nel nostro caso per capire se il sistema stesse ruotando da un verso o da un altro abbiamo confrontato i due segnali (il precedente e l'attuale) utilizzando dei booleani (and, or, =)

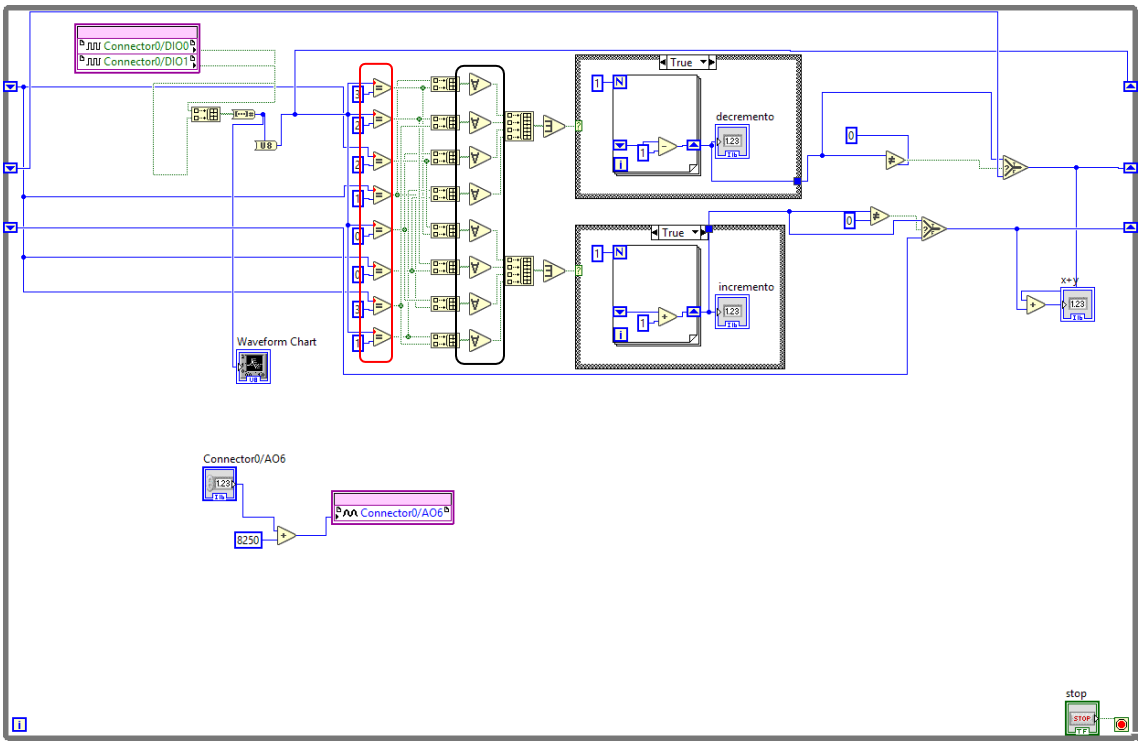


Figura 5-6 Block Diagram encoder rotazione

In primo luogo, bisogna capire quale è il segnale attuale e quello precedente, si è utilizzato il comando "=" per tutte e 4 le possibilità (0,1,2,3). Il segnale può uscire "Vero o Falso", vero se verifica l'uguaglianza, falso se non la verifica. Successivamente si creano vari array 2D in modo da collegare insieme i segnali attuali con quelli precedenti a coppie. L'array è collegato ad un AND array in modo tale che in uscita ho VERO solo se all'interno dell'Array ho entrambi i segnali VERI. In tutti gli altri casi in uscita ho FALSO. Per verificare appunto sto andando verso orario prendo i 4 segnali booleani e li metto in un array, l'uscita entrerà in un case structure, all'interno del quale ho un contatore che aumenterà il valore di 1 ogni volta che il case structure risulta vero. La stessa cosa viene fatta nel caso che il motore stia girando in verso antiorario, in questo caso il case structure risulti vero, avrò un decremento di uno.

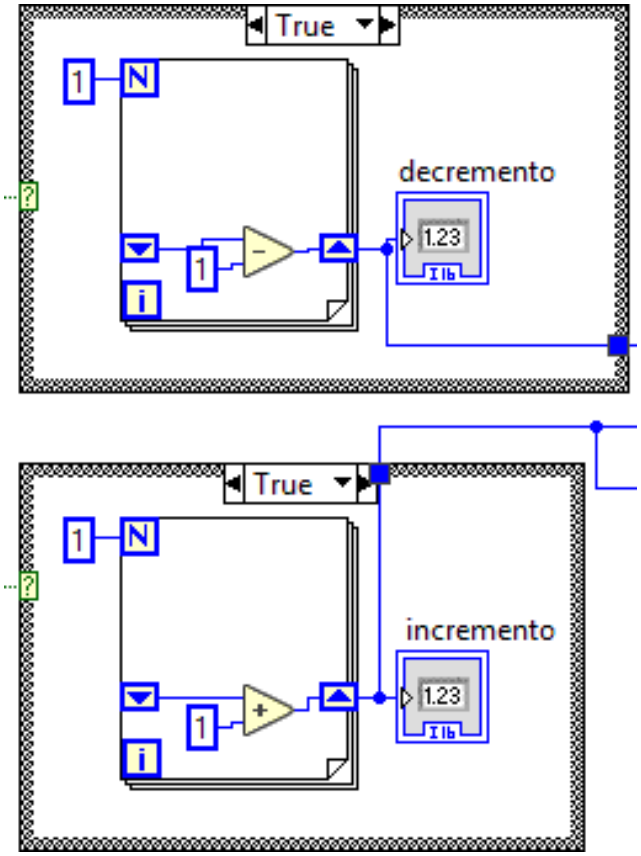


Figura 5-7 Incremento e decremento Block Diagram LabVIEW

Ovviamente l'oggetto del nostro interesse è il valore assoluto, in modo tale che se ho valore positivo il verso è orario, se invece ho valore negativo ho verso antiorario.

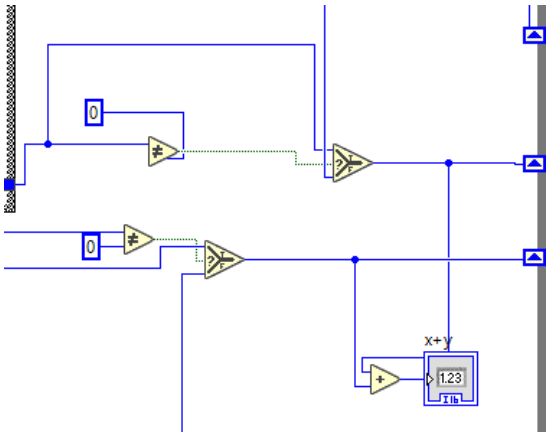


Figura 5-8 codice LabVIEW valore assoluto rotazione

Questo codice, scritto nell'FPGA, viene come già detto letto in maniera continua in modo da conoscere istante per istante, la posizione dell'encoder, cioè quella del motore. Conoscere il verso di rotazione del motore è il primo step per fare il controllo di una traiettoria.

5.2 PID

Il controllo Proporzionale-Integrale-Derivativo, comunemente abbreviato come PID, è un sistema in retroazione negativa ampiamente impiegato nei sistemi di controllo. Può essere schematizzato in modo semplice da questa figura.

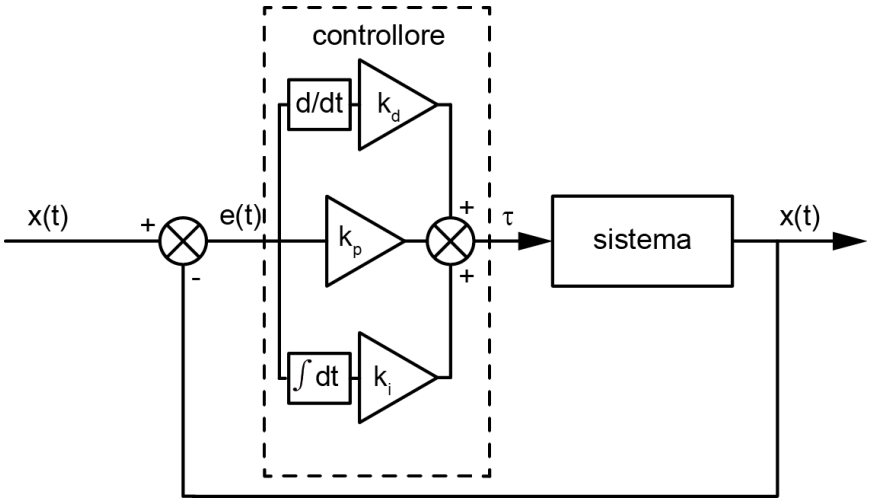


Figura 5-9 PID

Il primo step è dare una posizione o una traiettoria desiderata, conoscendo istante per istante la posizione del motore, che nel nostro caso viene letta dall'FPGA in modo continuo; facendo la differenza tra questi due valori ottengo un errore di posizione/traiettoria in uscita. Il controllore in questo caso il PID, modifica la posizione del motore fornendo il voltaggio necessario utilizzando una Sabertooth. Per avere un buon controllo bisogna definire gli elementi che caratterizzano il PID; I parametri che lo identificano sono K_P, K_I (o T_I) e K_D (o T_D) ed essi sono chiamati anche *gradi di libertà del controllore*.

Azione proporzionale

Quando parliamo di azione proporzionale significa che l'ingresso *e(t)* e l'uscita *u(t)* sono legati algebricamente da un coefficiente K_P, detto anche *coefficiente dell'azione proporzionale,* o semplicemente *guadagno proporzionale.* Lo schema di riferimento è il seguente:

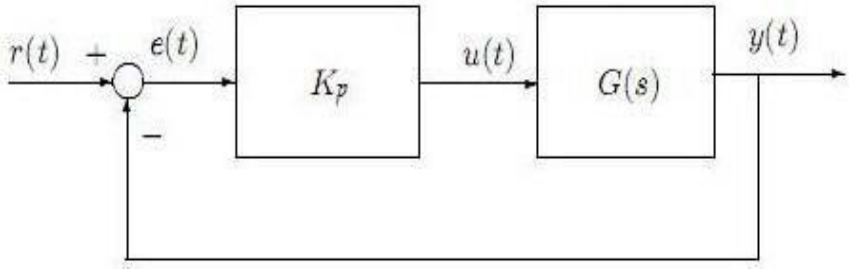


Figura 5-10 schema PID azione proporzionale

Il significato dell'azione proporzionale sta nel fatto che maggiore sarà l'errore *e(t)* all'ingresso del controllore e maggiore sarà l'azione di controllo svolta dallo stesso regolatore. Per riassumere il K_P si può considerare la seguente tabella.

Simbolo	Funzione di regolazione	Scopo principale	Stabilità	Tempo di risposta
K_P	Uno scostamento sull'ingresso (Errore) produce una variazione dell'uscita proporzionale all'ampiezza dello scostamento	Fa variare la grandezza regolante in funzione della grandezza regolata		
Aumenta K_P	-	-	Migliora	Rallenta
Diminuisce K_P	-	-	Peggiora	Accelera

Tabella 3 influenza variazione di Kp

Azione integrale

Veniamo ora ad analizzare la seconda azione introdotta dal PID, ovvero l'azione integrale. Il contributo di questa azione è proporzionale all'integrale dell'errore *e(t)* (e quindi proporzionale al suo valor medio) e il coefficiente dell'azione integrale K_I definisce la costante di tempo integrale T_I (chiamata anche *tempo di reset*).

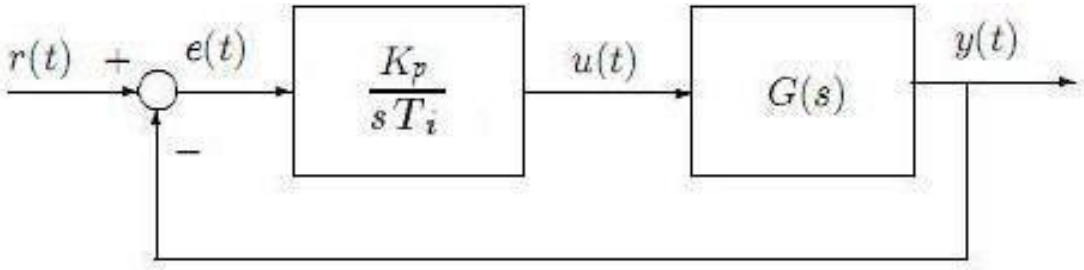


Figura 5-11 schema PID azione integrale

L'azione integrale è particolarmente importante nelle applicazioni, perché assicura un errore nullo a regime per variazioni a gradino del riferimento *r(t)*. L'errore rimane nullo anche in presenza di variazioni del guadagno del processo, purché sia preservata la stabilità del sistema in anello chiuso. I controllori PI (proporzionale e integrativo) vengono utilizzati quando è richiesto un errore a regime di modesta entità unito ad una buona velocità di risposta alle variazioni della sollecitazione; pertanto vengono inseriti soprattutto nei sistemi in cui le variazioni di carico avvengono lentamente. La taratura della costante di integrazione è legata alle sovra elongazioni ed alle oscillazioni che si possono innescare. Inoltre, cambiamenti improvvisi di carico possono portare il sistema verso l'instabilità, quando il coefficiente K_I dell'azione integrale non è scelto in modo opportuno. In ambito industriale oltre il 90% dei controllori è di questo tipo. Nel caso di un segnale di riferimento *r(t)* a gradino, per esempio, l'integrale crescerà sempre di più, come evidenziato in *figura 5-13.*

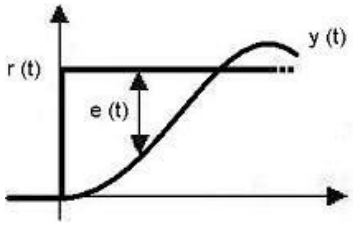


Figura 5-12 grafico influenza Ki sul controllo

La variabile da tarare in base al sistema con cui si ha a che fare è la costante di tempo integrale T_I; da essa infatti dipende l'effetto di integrazione che è tanto più importante quanto più T_I è piccola. A questo riguardo è necessario notare come riducendo il *tempo di reset*, l'integrale dell'errore salirà più velocemente verso il valore *r(t)* in ingresso al blocco pagando però questa velocità di salita con delle forti oscillazioni che richiedono tempo prima di stabilizzarsi. Viceversa, aumentando T_I e facendolo tendere ad infinito, si ha come conseguenza l'eliminazione dell'effetto dell'integrale

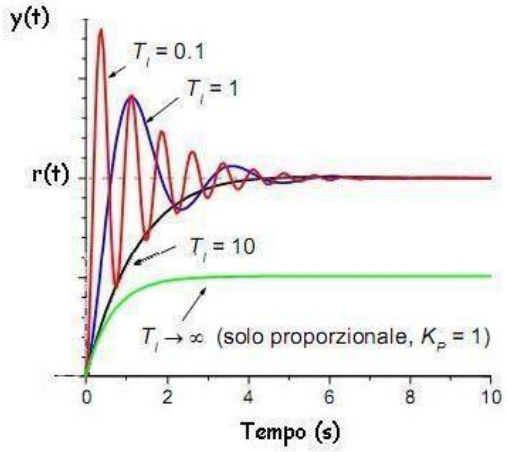


Figura 5-13 influenza Ki in funzione del Ti

Il termine integrale però, a differenza di quello proporzionale, introduce uno sfasamento di 90° in ritardo (dunque aggiunge un polo in termini di trasformata di Laplace) che porta ad un peggioramento dei margini di stabilità del sistema ad anello chiuso, limitando il valore del guadagno integrale (e quindi la banda del sistema). Questa tabella ci fornisce una visione generale del blocco integrale.

Simbolo	Funzione di regolazione	Scopo principale	Stabilità	Tempo di risposta del loop
T_I	Appena si ha uno scostamento sull'ingresso (Errore), si produce una variazione dell'uscita con velocità proporzionale allo scostamento	Fissa il punto di regolazione (Elimina l'offset dato dall'azione proporzionale)		
Aumenta T_I	-	-	Migliora	Rallenta
Diminuisce T_I	-	-	Peggiora	Accelera

Tabella 4 influenza variazione di Ti

Azione derivativa

Come ultima caratteristica di un regolatore ci rimane da analizzare l'azione derivativa che fornisce in uscita la derivata rispetto al tempo dell'errore *e(t)*. In questo caso abbiamo la presenza del coefficiente dell'azione derivativa K_D che definisce la costante di tempo derivativa T_D.

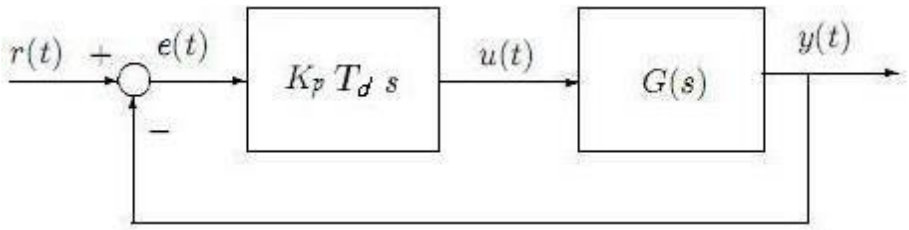


Figura 5-14 schema PID azione derivativa

Un controllore derivativo viene chiamato anche controllore di velocità, oppure anticipatore. Il suo comportamento è marcatamente diverso da quello dei controllori proporzionale e integrale. L'uscita di un controllore derivativo non dipende dall'errore presente o passato, ma dalla velocità con cui varia l'errore. Inoltre, l'azione derivativa è complementare all'azione integrale perché fornisce un anticipo di fase di 90° (in quanto porta all'introduzione di uno zero nell'origine). Il parametro appunto che governa questo blocco è la costante di tempo dell'azione derivativa T_D, il cui valore determina la velocità di salita del segnale di controllo. In questo caso rispetto ai due blocchi visti precedentemente la stabilità peggiora sia aumentando sia diminuendo il valore di T_D.

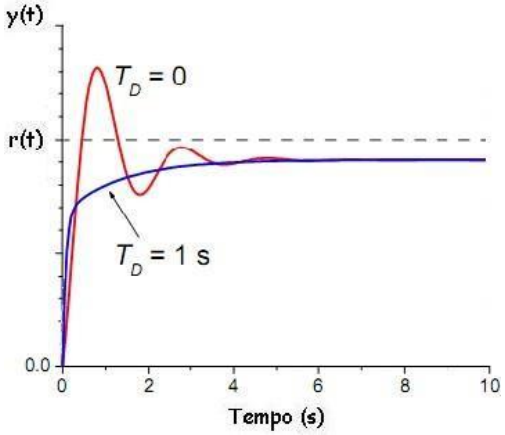


Figura 5-15 grafico influenza Td sul controllo

Nella tabella sottostante possiamo dare una breve visione generale delle azioni che un blocco derivatore introduce

Simbolo	Funzione di regolazione	Scopo principale	Stabilità	Tempo di risposta del loop
T_D	Uno scostamento sull'ingresso (Errore), produce una variazione dell'uscita proporzionale alla velocità di variazione dello scostamento	Dimunuisce il tempo di risposta per il ritorno al punto di regolazione		
Aumenta T_D	-	-	Peggiora	Accelera
Diminuisce T_D	-	-	Peggiora	Rallenta

Tabella 6 influenza variazione Td

Il termine derivativo purtroppo introduce l'inconveniente di amplificare i segnali con contenuto armonico a frequenze elevate. Siccome in genere questo tipo di segnali corrispondono al rumore elettromagnetico sovrapposto al segnale utile, l'utilizzo del termine derivativo deve essere valutato con cautela. Nella realizzazione pratica dei regolatori infatti si adotta l'accorgimento di limitare l'azione derivativa in modo tale che in corrispondenza a transizioni a gradino l'uscita del blocco derivatore non abbia andamento impulsivo.

Metodo della risposta in frequenza (II metodo di Ziegler-Nichols)

Per calcolare i valori di K_P, K_I (o T_I) e K_D (o T_D) da inserire nel controllore è stato utilizzato il metodo di Ziegler-Nichols. Questo metodo opera con il sistema in retroazione unitaria, cioè si considera il sistema in catena chiusa, e in cui è già presente il controllore PID.

Per poter applicare questo metodo si procede mediante una sequenza di passaggi che vengono illustrati nel seguente elenco:

- per prima cosa bisogna annullare, o quantomeno ridurre al minimo livello possibile, le azioni di integrazione e derivazione, facendo lavorare il controllore in modo puramente proporzionale. Per fare ciò si settano le corrispondenti variabili di controllo T_I e T_D al valore nullo.
- Successivamente si aumenta in maniera progressiva (e prudentemente) il valore della variabile di controllo K_P fino a quando si osserva che l'uscita $y(t)$ del sistema oscilla in modo permanente (limite di stabilità).

Indichiamo con K_0 il valore limite di K_P, detto anche *guadagno critico*, e con T_0 il periodo di oscillazione misurato quando viene raggiunto questo valore limite di K_P. Se il sistema non entra mai in oscillazione, il metodo non è applicabile. In *figura 5-16* è riportata l'uscita del sistema entrato in oscillazione permanente.

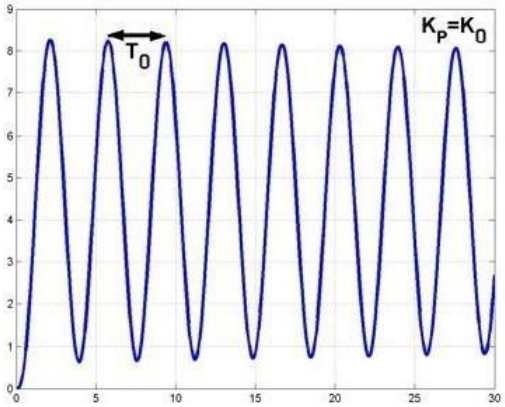


Figura 5-16 sistema in oscillazione permanente

- A tal punto si ricorre ad una tabella di conversione (riportata di seguito) per settare correttamente i parametri di controllo, così da avere una buona reiezione dei disturbi sul carico

	K_p	K_i	K_d	T_i	T_d
P	$0{,}50K_0$				
PI	$0{,}45K_0$	$0.54K_0/T_0$		$T_0/1{,}2$	
PD	$0{,}80K_0$		$K_0T_0/10$		$T_0/8$
PID	$0{,}60K_0$	$1.2K_0/T_0$	$3K_0T_0/40$	$T_0/2$	$T_0/8$

Conoscendo i valori di K_0 e T_0, trovati nel modo descritto precedentemente e utilizzando queste formule si riescono a trovare i valori di K_P, K_I (o T_I) e K_D (o T_D) desiderati per il controllo del sistema. In particolare, i valori usati sono:

- K_p= 13,2
- K_i= 0,43
- T_i= 0,31

Si è scelto inoltre $K_d = 1$ per limitarlo così da non avere un andamento impulsivo.
Si passa alla scrittura del codice PID in LabVIEW sulla PXI.

5.3 Virtual Instrument PXI

Dopo aver scritto il codice di rotazione dell'encoder nell'FPGA si passa alla scrittura del PID nella PXI NI 8187, in modo da ottenere un controllo più efficace. La prima è aprire il codice dell'FPGA, poi viene letto, fuori dal ciclo timed loop in modo tale che la posizione iniziale nella traiettoria del motore risulti sempre uguale a 0.

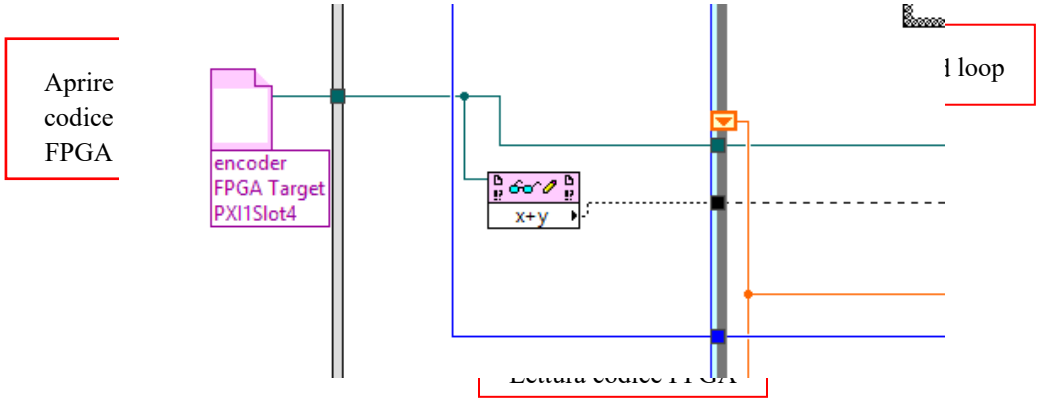


Figura 5-17 Block Diagram

La linea in uscito dalla lettura del codice FPGA è tratteggiata perché in quel momento il pc non era collegato alla PXI. All'interno del timed loop si passa alla scrittura del controllo PID nel Block Diagram

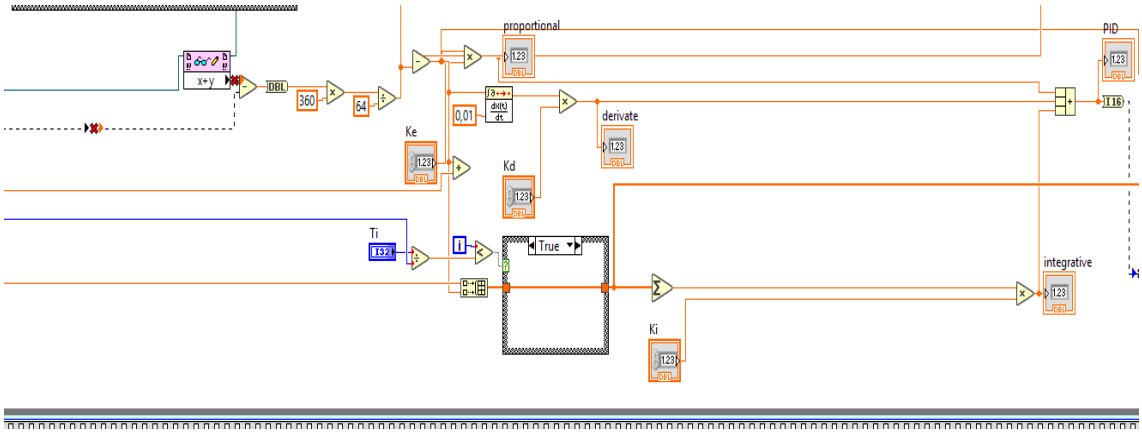


Figura 5-18 Labview Block Diagram PID

Con i valori di Ke (Kp) di Kd e di Ki scelti con le formule di Ziegler-Nichols.

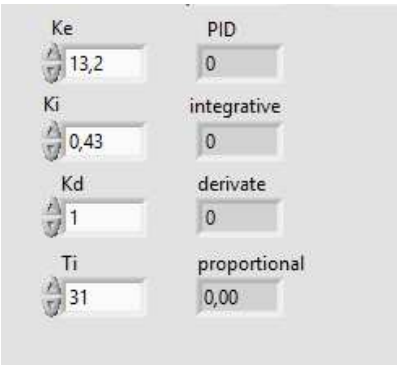
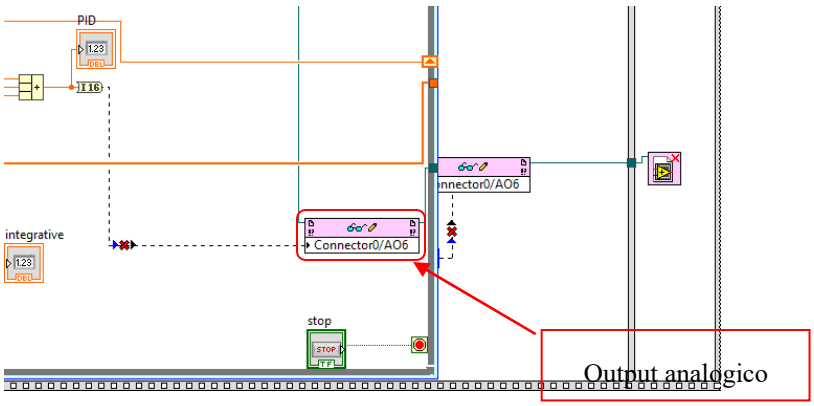


Figura 5-19 Front Panel valori variabili PID usate nel sistema

In uscita dal PID ho un Output Analogico il quale attraverso la Sabertooth2x12 fornirà il voltaggio necessario per far muovere il motore con la traiettoria scelta.



5.4 Analisi e Virtual Instrument traiettoria motore

Con la pianificazione della traiettoria si intende stabilire la modalità con cui si vuole che evolva il movimento del manipolatore (in questo caso il motore), da una posizione iniziale ad una posizione finale. Si tratta di definire sia il percorso geometrico sia la legge di moto da realizzare (ossia la dipendenza temporale di posizioni, velocità ed accelerazioni), nel nostro caso si è definita solamente l'analisi di posizione. La corretta pianificazione della traiettoria ha estrema importanza, perché comporta che la traiettoria possa essere eseguita, in modo corretto, da parte del sistema di controllo del moto in anello chiuso. Il caso più semplice di pianificazione della traiettoria per moto punto-punto si ha quando sono specificate alcune condizioni iniziali e finali sulla posizione,

velocità ed eventualmente anche su accelerazione ed il tempo di percorrenza. Si possono prendere in considerazione funzioni polinomiali del tipo:

$$q(t) = a_0 t^0 + a_1 t^1 + \cdots + a_n t^n$$

Più alto è il grado n del polinomio, più condizioni al contorno si possono soddisfare. Per l'analisi di posizione viene considerato un polinomio di 3° grado.

$$q(t) = a_0 t^0 + a_1 t^1 + a_2 t^2 + a_3 t^3$$

E come condizione al contorno si hanno: velocità iniziali e finali nulle, che nel nostro caso sono le velocità angolari

$$\omega_i = 0$$

$$\omega_f = 0$$

Con queste due condizioni al contorno il polinomio si semplifica

$$q(t) = a_2 t^2 + a_3 t^3$$

Le altre 2 condizioni al contorno che mancano solo quelle relative alla posizione ovvero

$$\theta_i = \theta_i$$

$$\theta_f = \theta_f$$

Il θ_i e il θ_f sono dei valori che conosco perché quello finale lo scelgo io, mentre quello iniziale è la posizione attuale del mio motore. Il codice della traiettoria deve essere scritto in LabVIEW, come prima cosa bisogna definire un tempo assoluto, per avere una traiettoria corretta, poi scrivere il codice della traiettoria. Il tempo LabVIEW è una variabile non deterministica, ci sono i "comandi" di tempo, però tutti dipendono dalla frequenza dell'hardware. Per avere un tempo deterministico come prima cosa bisogna utilizzare una "flat sequence" in modo tale che i codici scritti nelle varie sequenze non vengono eseguiti contemporaneamente, ma in modo successivo.

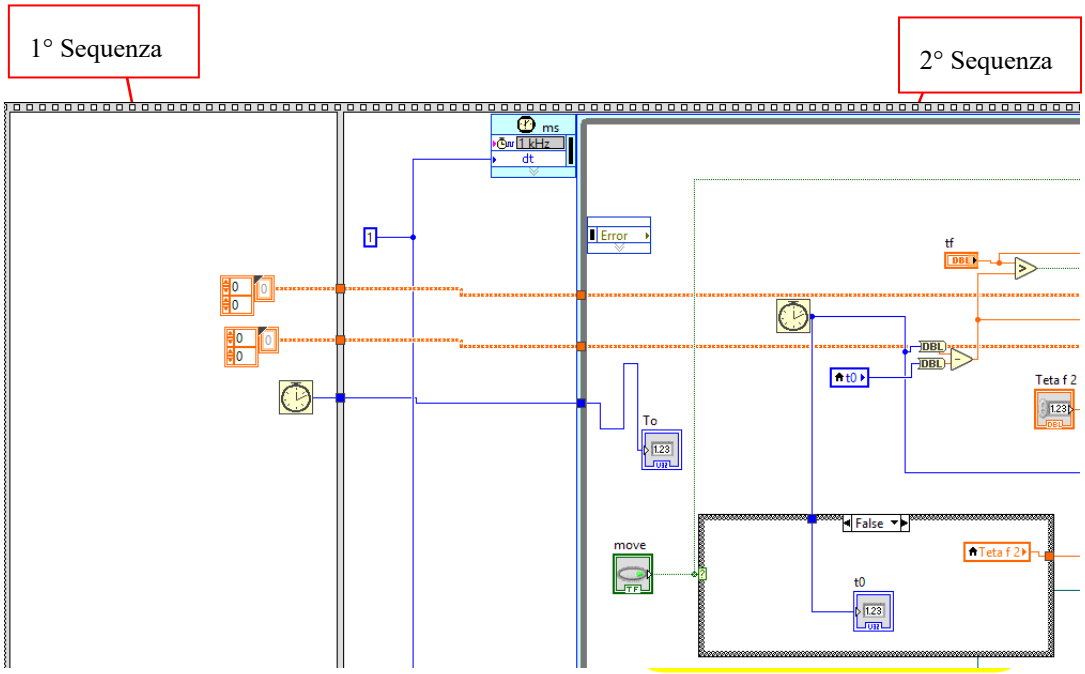
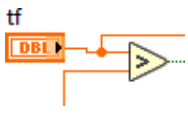


Figura 5-20 Block Diagram parte del codice della PXI

In figura è rappresentato il codice per determinare il tempo preciso. Nella prima sequenza viene fatto partire un tempo di riferimento T_0. Nella seconda sequenza si ha un ciclo temporizzato (timed loop) con frequenza di 1 KHz, per far lavorare la PXI a massimi regimi. Il tempo t_0 all'interno del timed loop cresce rispetto al tempo di riferimento T_0, che essendo uscito dalla 1° sequenza si stoppa. Per poter scegliere il tempo di esecuzione della traiettoria si utilizza un "case structure", controllato da un booleano. Se la struttura è falsa il tempo t_0 aumenta continuamente, mentre quando premo il booleano la struttura diventa vera. nell'instante in cui il "case structure" risulti VERO si ha la differenza di due tempi uno che continua ad aumentare e uno che si stoppa. La differenza tra i due tempi mi darà il valore assoluto del tempo di esecuzione che scelgo per la traiettoria t_f. L' esecuzione del codice della traiettoria viene eseguito solamente quando la condizione di "comparison" risulta verificata.



La comparision" risulta VERA finché la differenza tra i due tempi, la quale parte da un istante zero, è inferiore al valore scelto di t_f, non appena lo supera l'output della "comparison" risulterà FALSO.

Il codice della traiettoria viene eseguito solamente quando la “comparison” risulta VERA perché è scritto all’interno di un “case structure”.

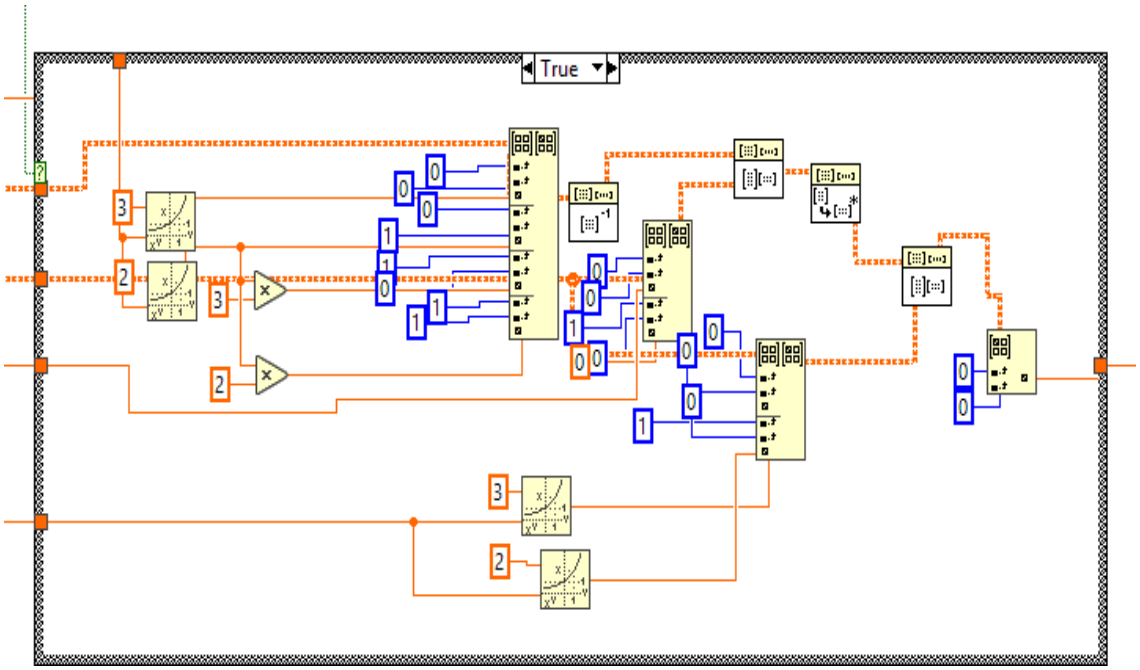


Figura 5-21 Block Diagram codice traiettoria di posizione

In figura è rappresentato il codice del polinomio di 3° grado, semplificato facendo le considerazioni scritte precedentemente, scritto in Labview.

$$q(t) = a_2 t^2 + a_3 t^3$$

Per prima cosa bisogna rappresentarlo in forma matriciale. Le mie incognite in questo caso sono a e b. θ_i lo imposto sempre uguale a zero, θ_f invece è il valore che si vuole ottenere dalla traiettoria. Per ottenere i valori desiderati di a e b bisogna fare la matrice inversa. Tutti questi processi in Labview sono descritti dal seguente codice.

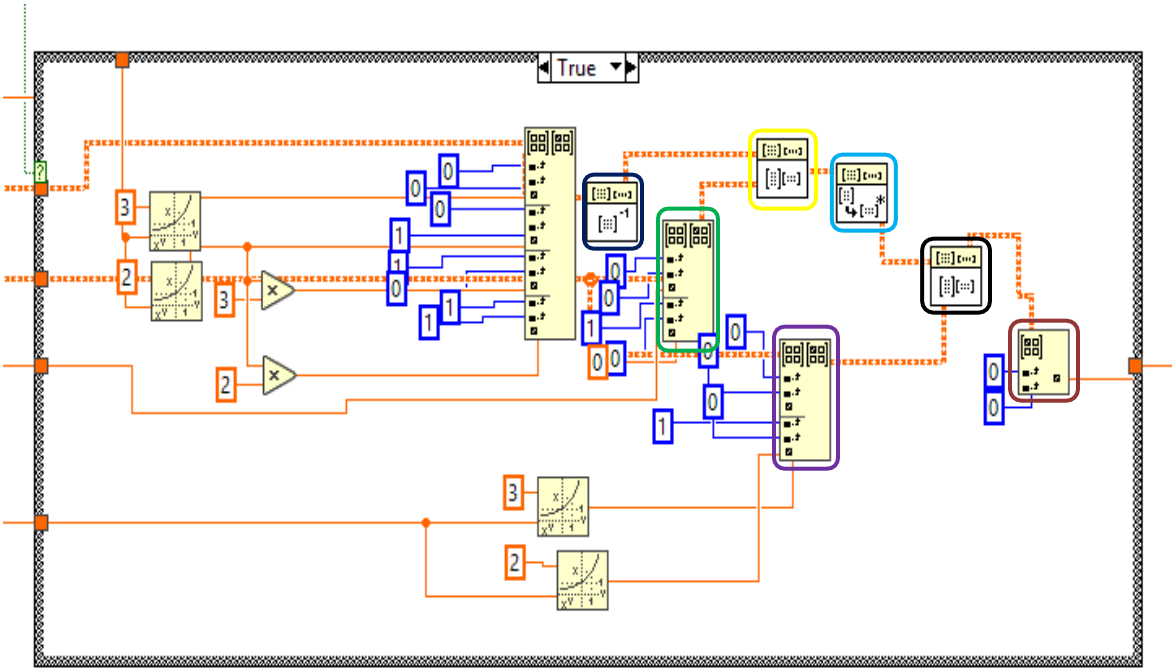


Figura 5-22 codice traiettoria LabVIEW

- 1° step Costruzione matrice 2x2 i numeri riquadrati in blu indicano la posizione all'interno della matrice, mentre le linee in arancione sono collegate al valore desiderato.
- ° step Inversa
- 3° step prodotto tra la matrice inversa per il vettore $\begin{bmatrix} \theta_f \\ 0 \end{bmatrix}$
- 4° step Trasposta del prodotto il risultato è un vettore colonna
- 5° step Prodotto vettoriale tra la trasposta e il vettore $\begin{bmatrix} t^3 \\ t^2 \end{bmatrix}$
- 6° step estrarre i valori di a e b dalla matrice 2x2 ottenuta

Il codice della traiettoria viene eseguito solamente quando il "case structure" risulta VERO. Nel caso in cui l'output della comparison risulta FALSO, cioè quando la differenza dei tempi risulta maggiore di t_f. Il codice all'interno del "case structure" è il seguente

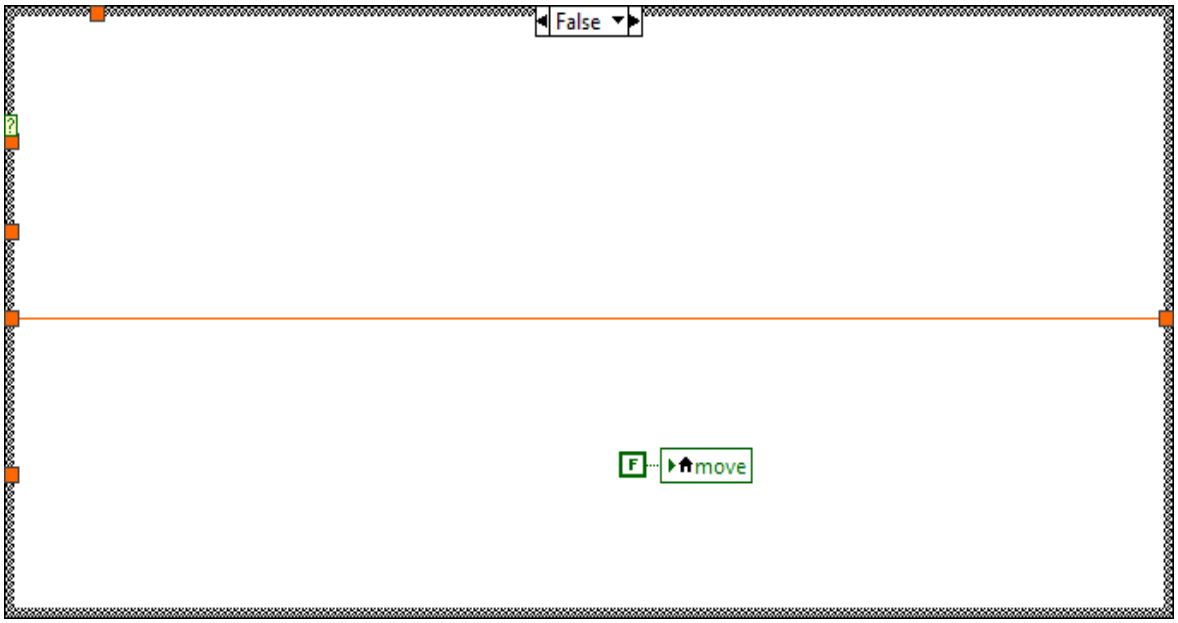


Figura 5-23 Block Diagram case sctructure False

Il quale mi fa diventare falso il “case structure” controllato dal pulsante “move”.

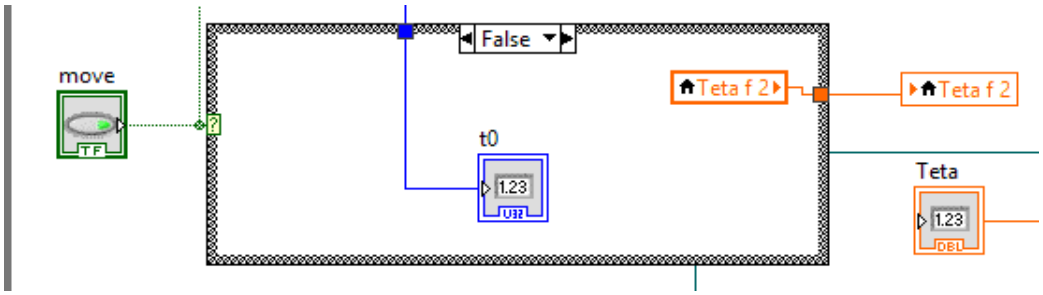


Figura 5-24 Block Diagram case sctructure per iniziare la traiettoria

In questo modo il mio motore rimarrà sempre nella posizione θ_f scelta, finché non si voglia cambiare posizione. In figura è raffigurato il Front Panel del VI completo di controllo del motorescritto nella PXI.

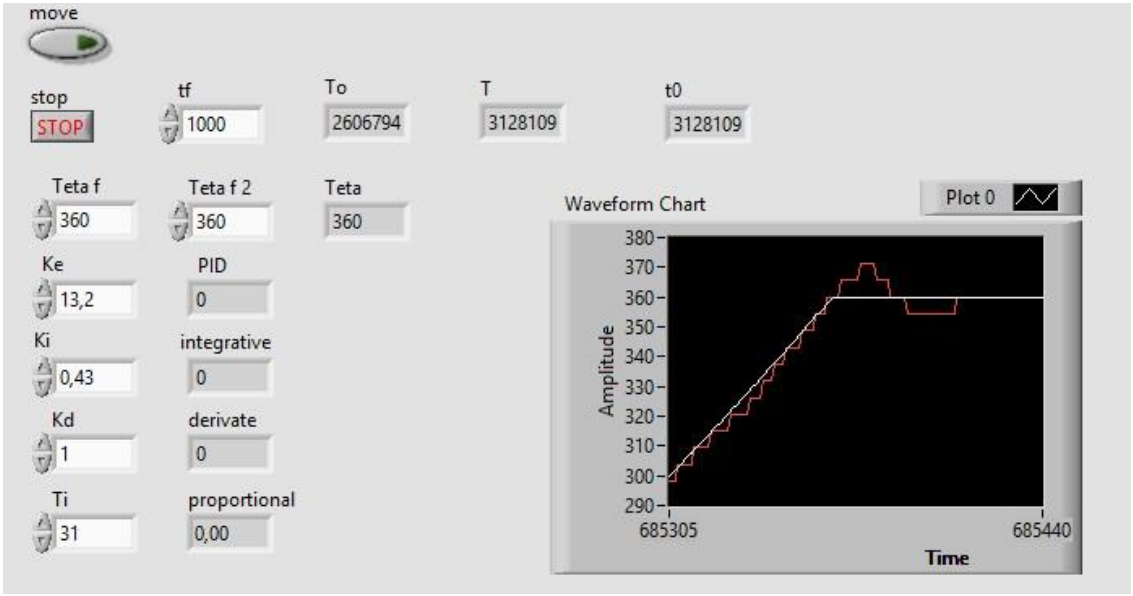


Figura 5-25 Front Panel azionamento e controllo del motore

Nel grafico all' interno della figura la linea bianca rappresenta il valore di posizione finale scelto per la traiettoria θ_f, mentre la linea rossa rappresenta la posizione dell'encoder rispetto alla traiettoria scelta. Le linee dell'encoder sono onde quadre perché il segnale è digitale.

Il Block Diagram completo di controllo del motore all'interno della PXI è il seguente.

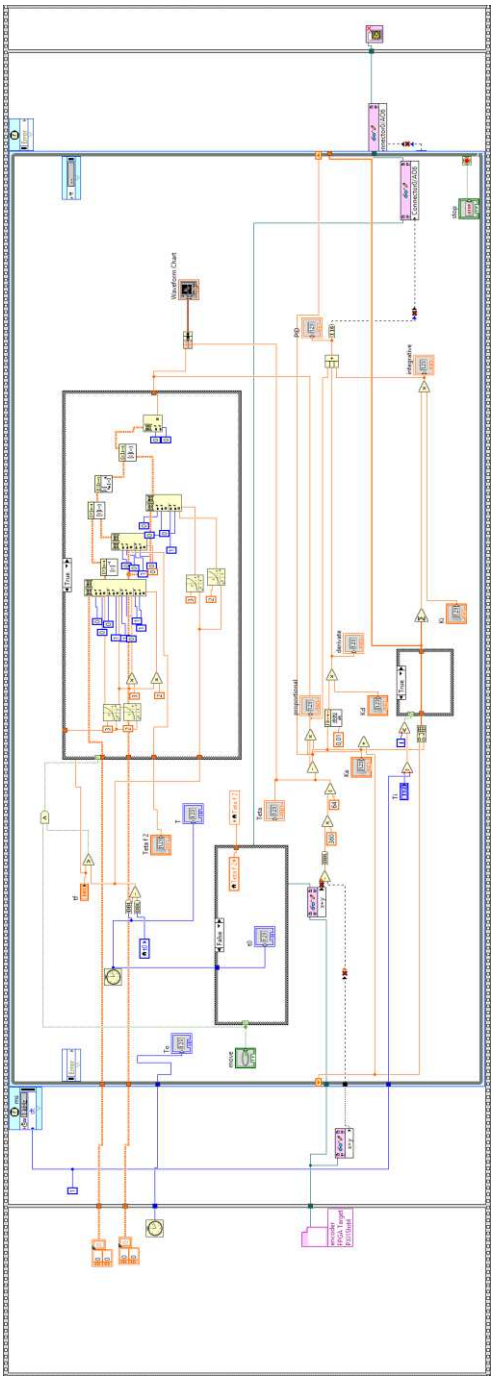


Figura 5-26 codice Block Diagram PXI

La schematizzazione dell'hardware finale è il seguente.

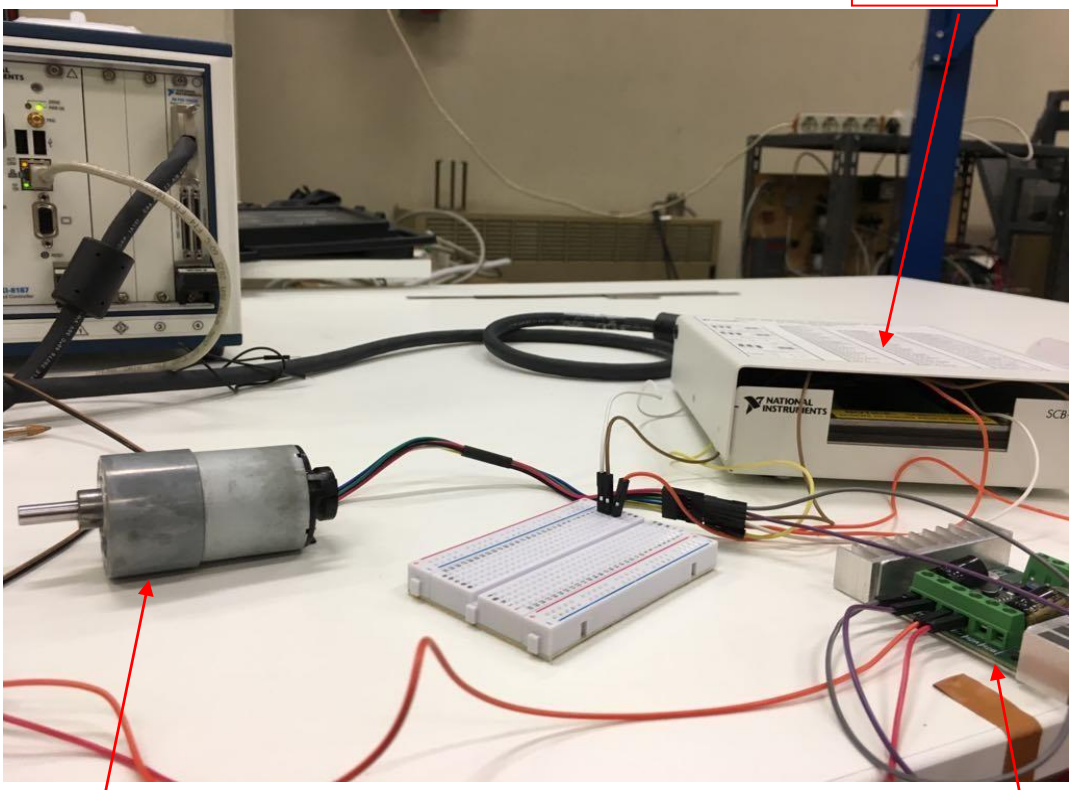


Figura 5-27 Foto banco di lavoro

In figura è rappresentato la composizione finale dell'hardware di controllo. Il motore viene alimentato dalla Sabertooth che a sua volta riceve l'input di voltaggio da dare al motore dall'output analogico nel SCB-68. La quale è collegata alla PXI che attraverso i codici scritti nel software LabVIEW controlla tutto il sistema è il voltaggio in uscita dall'SCB-68 necessario per il controllo della traiettoria del motore. Con questo capitolo si conclude la trattazione del controllo della traiettoria attraverso l'hardware e il software citati precedentemente.

6 MATLAB

Dopo aver affrontato il problema in Labview si è fatta un'analisi di traiettoria (diretta e inversa) di un Robot 2R in MATLAB. MATLAB è un ambiente per il calcolo numerico e l'analisi statistica scritto in C, che comprende anche l'omonimo linguaggio di programmazione creato dalla MathWorks. MATLAB consente di manipolare matrici, visualizzare funzioni e dati, implementare algoritmi, creare interfacce utente, e interfacciarsi con altri programmi. Prima di iniziare ad utilizzare il programma, essendo il primo approccio con questo programma, si è fatta un veloce infarinatura di comandi, cicli e codici per l'utilizzo. Come primo esercizio è stata pianificata

Dati del problema:

- Punto iniziale e finale della traiettoria

  $P_0 = \begin{bmatrix} 0\ \text{m} \\ 0\ \text{m} \end{bmatrix}$ $\theta_1 = 0°;\ \theta_2 = 90°$

- Lunghezza aste

  $l_1 = l_2 = l = 1\ \text{m}$

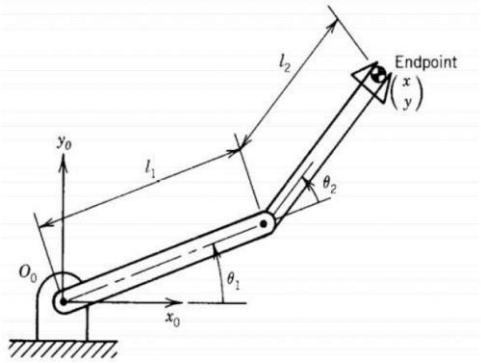


Figura 6-1 robot piano 2R

In MATLAB per avere una corretta pianificazione della traiettoria in cinematica diretta, in primo luogo è stata scritta la function della cinematica diretta, come in figura.

```
function [x,y]= cin_diretta(q1,q2)
l1=1
l2=1
x0=0
y0=0
x=x0+l1*cosd(q1)+l2*cosd(q1+q2)
y=y0+l1*sind(q1)+l2*sind(q1+q2)

end
```

Figura 6-2 codice function cinematica diretta in matlab

Per semplicità di scrittura abbiamo considerato $\theta_1 = q_1$ e $\theta_2 = q_2$. Con i dati dell'esercizio la soluzione trovata in MATLAB $P_f = \begin{bmatrix} 1\ m \\ 1\ m \end{bmatrix}$ è, come si può vedere in figura.

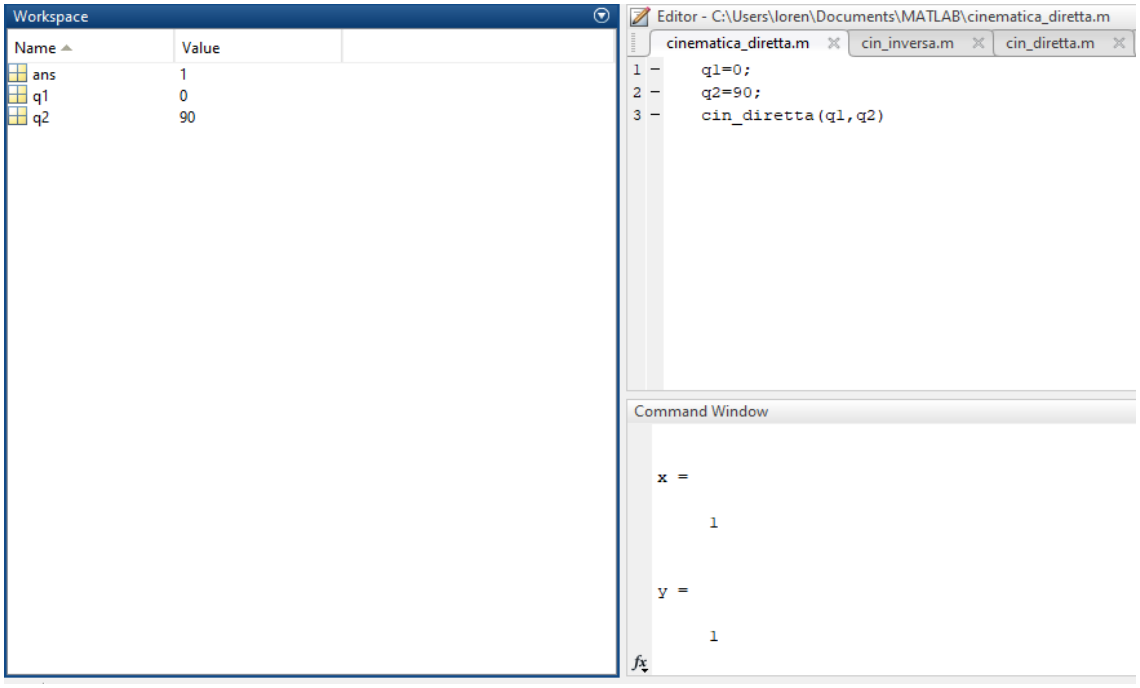


Figura 6-3 schermata matlab esercizio cinematica dirtetta

Dopo aver fatto una semplice pianificazione di cinematica diretta di un robot 2R, si è passati alla cinematica inversa. La differenza sostanziale è che in cinematica inversa l'incognita sono i due angoli, mentre in quella diretta era il punto finale. La cinematica inversa è molto simile al controllo di un motore fatto in precedenza infatti se si considera i punti di intersezione delle aste come due motorini elettrici, avendo prefissato una posizione finale, il motorino si sposterà di una certa angolazione. Per l'esecuzione della cinematica inversa in primo luogo viene scritta la seguente funtion.

```
function [q1,q2] = cin_inversa(x,y)
l1=1;
l2=1;
c2=(x^2+y^2-l1^2-l2^2)/(2*l1*l2);
s2=-sqrt(1-(c2)^2);
c1=((l1+l2*c2)*x+l2*s2*y)/(x^2+y^2);
s1=((l1+l2*c2)*x-l2*s2*x)/(x^2+y^2);
q1= atan2(s1,c1)*180/pi;
q2= atan2(s2,c2)*180/pi;
end
```

Figura 6-4 codice function matlab cinematica inversa

In seguito, ipotizzando i seguenti dati:

$$P_i = \begin{bmatrix} 1\ \text{m} \\ 1{,}5\ \text{m} \end{bmatrix} \qquad P_f = \begin{bmatrix} -1\ \text{m} \\ -1\ \text{m} \end{bmatrix}$$

- Lunghezza aste

$$l_1 = l_2 = l = 1\ \text{m}$$

- Numero di punti per discretizzare la traiettoria

n=50

```
clear
clc
close all


pi=[1,1.5];
pf=[-1,-1];
```

```
l1=1;
l2=1;
n=50;
figure;
   hold on;
    X=linspace(1,-1,n);
    axis equal
    xlim([-2 2]);
    ylim([-2 2]);

 for i=1:n
     Y(i)=(X(i)-pi(1))/(pf(1)-pi(1))*(pf(2)-pi(2))+pi(2);
     [q(1,i),q(2,i)]=cin_inversa(X(i),Y(i));
     A=[0;0];
     B=l1*[cosd(q(1,i));sind(q(1,i))];
     C=B+l2*[cosd(q(1,i)+q(2,i));sind(q(1,i)+q(2,i))];
     robot=[A,B,C];
     if i==1
         PR=line(robot(1,:),robot(2,:),'marker','o');

     else
         PR.XData=robot(1,:);
         PR.YData=robot(2,:);

     end
     pause(0.1)
 end
```

Figura 6-5 codice finale esercizio cinematica inversa

I risultati dei vari angoli, avendo considerato 50 iterazioni sono 50. I primi e gli ultimi 2 valori delle varie iterazioni sono riportati nella seguente tabella, ovviamente più aumento il numero di iterazioni, più avrò una pianificazione precisa.

q_1	79,31114538	84,93213252	-87,74465773	-90
q_2	-51,31781255	-59,35031784	-95,14647554	-90

Tabella 5

Per avere anche una visione grafica dell'andamento della traiettoria nell'editor sono stati scritti anche dei codici per rappresentare graficamente la traiettoria come si può vedere in figura.

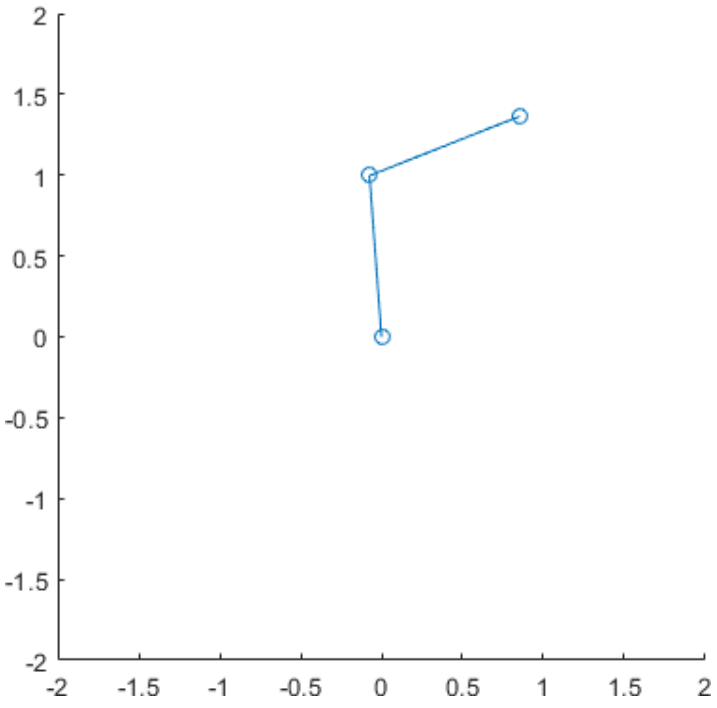


Figura 6-6 grafico movimento robot 2R

7 Conclusioni

In conclusione i risultati ottenuti dallo svolgimento del problema sono ottimali e soddisfano le richieste iniziali dell'argomento di tesi. L'obbiettivo era di riuscire a controllare e modificare la traiettoria di posizione di un motore elettrico. Nella prima fase vengono scritti correttamente in LabVIEW tutti i codici necessari per la riuscita del problema affrontato, dopo aver trovato i valori delle variabili caratteristiche del controllo PID necessarie per il sistema si è passati alla verifica del funzionamento dei codici attraverso la sabertooth 2x12. Il driver è necessario per il corretto funzionamento di tutto il sistema, in input riceve il voltaggio che esce dalla SCB-68, il quale è troppo basso per far muovere il motore, infatti massimo può arrivare a 5V, mentre in uscita fornisce il valore necessario di voltaggio al motore per fargli fare la traiettoria scelta. Cambiando la traiettoria, il driver in base al segnale che riceve in input cambia il voltaggio che fornisce al motore.

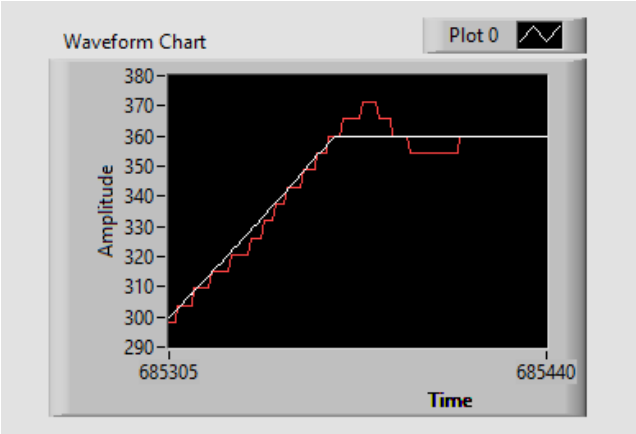


Figura 7-1

In figura si può vedere la risposta del motore rispetto al valore di traiettoria richiesto. Il risultato è ottimale perché considerando che in figura il tempo è rappresentato in millisecondi il motore segue perfettamente la traiettoria scelta e l'errore di posizione finale tende a zero, questo perché i valori delle variabili del PID scelti sono ottimali. Il problema di questi risultati è che non ho una trattazione generale e propriamente specifica del controllo di un motore. Perché nel precedente elaborato viene presa in considerazione solamente la posizione per determinare la traiettoria di un motore, però in realtà è una semplificazione. Per avere una completa visione del problema bisogna

anche considerare la velocità e l'accelerazione del sistema preso in analisi. Nel nostro caso la velocità finale e inziale le abbiamo ipotizzate nulle e la traiettoria si muoveva a velocità costante, per avere una visione un po'più specifica bisogna fare anche un'analisi di velocità e scrivere gli eventuali codici necessari per la realizzazione. Inoltre, non si è nemmeno ipotizzata un'accelerazione, la quale risulta necessaria perché così conosco tutti i parametri necessari per avere una traiettoria precisa del sistema preso in analisi. Ovviamente sia i codici di velocità e accelerazione sono molto più complessi da scrivere in LabVIEW rispetto alla semplice posizione.

7 Bibliografia

[1] NI R Series Multifunction RIO User Manual

[2] Cangini Marco tesi

[3] Manuale NI PXI 1031

[4] https://www.dimensionengineering.com/datasheets/Sabertooth2x12.pdf

[5] https://www.pololu.com/product/2824

8 Elenco Figure

9 Elenco Tabelle

Ringraziamenti

"Impariamo a dire "Grazie" a Dio, agli altri. Lo insegniamo ai bambini, ma poi lo dimentichiamo." Non sapevo, come iniziare i ringraziamenti e quindi ho scelto questa frase di Papa Francesco. Il primo grazie per questo percorso va alla mia famiglia, che mi ha aiutato e sostenuto nei momenti più brutti. A mamma che anche se ci vogliamo bene un po' a modo nostro, mi ha sempre aiutato nelle scelte. A babbo che è sempre una fonte di confronto sempre disponibile. A Giorgio che oramai è entrato nella mia famiglia, un po' allargata, che, anche se ordina sempre pacchi, è stato una spalla su cui appoggiarsi. Ovviamente anche a tutti i parenti un grande grazie. Adesso si passa agli amici. In primis devo ringraziare il gruppo storico e di una vita F.C. Troie 2.0, loro sono stati la mia seconda famiglia, ovviamente con alcuni ho un rapporto più stretto e con altri un po' meno, però vi voglio veramente tanto bene. Un altro grazie va a Carbo, anche se ogni tanto ti sbatterei la testa contro il muro, sei stato sempre una persona disponibile, profonda e piena di interessi. Grazie anche Lori Conti in particolare, ma anche a tutti gli amici con cui ho passato le superiori, tra risate e voti insufficienti sono stati 5 anni indimenticabili. Per concludere gli amici a Cardi, Ferro, Marci, Simo, Monte, Tibe, Tacca, Cru, Nico, Richi, Fra…siete veramente troppi e mi dispiaci non potervi invitare tutti alla festa. Oltre a tutti loro una parte della mia vita è l'azione cattolica ed è li che ho fatto anche tantissime conoscenze, dagli educatori di Regina della Pace, con i quali ho cominciato un percorso bellissimo e che continuerà sicuramente. Anche agli educatori del Kolbe con cui quest'anno abbiamo dovuto faticare insieme a tenere 30 ragazzi e a tutti quelli che in questi anni hanno camminato con me. Non so a quanti ringraziamenti sono arrivato, però adesso tocca ai The Branches, anche se ogni tanto i cantanti stonano siete una cosa bellissima, ciò che facciamo può non piacere a tutti, ma a me piace tutto, dai concerti, anche se alcuni li avrei evitati, alle prove e alle chiacchierate con Dani e Fede. Siete una scintilla all'interno del mio cuore, grazie. In particolare, tra gli amici "fedeli" devo assolutamente ringraziare Letizia, che è sempre una fonte di confronto, di lunghe chiacchierate e di una diponibilità unica. Un'altra è Eleonora, che anche se sei diventata malata di testa con lo studio, le passeggiate lunghe e le chiacchierate ancora più lunghe sono qualcosa di unico. Un super gigante ringraziamento va a tutti gli amici con cui ho condiviso questo percorso, siete stati unici, e avete reso il tutto più spensierato, bello e leggero. Infine, un ultimo ringraziamento, ma non meno importante va a Dorotea, grazie a lei questo momento è ancora più bello. Ti ringrazio perché mi sopporti sempre e ci sei sempre, grazie per avermi reso una persona migliore, sei davvero importante e oramai hai preso un posto grande all'interno del mio cuore, anche se può sembrare freddo. Ti voglio bene, non sono il tipo da cose sdolcinate lo sai. Insomma, un ringraziamento finale a tutti e grazie per aver condiviso con me questo percorso.

Lorenzo Lattanzi