

UNIVERSITÀ DEGLI STUDI DI PADOVA
FACOLTÀ DI INGEGNERIA



UNIVERSITÀ DEGLI STUDI DI PADOVA
FACOLTÀ DI INGEGNERIA

—
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

—
TESI DI LAUREA MAGISTRALE IN BIOINGEGNERIA

CONTROLLO DI SISTEMA
ROBOTICO PLANARE TRAMITE
BUS ETHERCAT

RELATORE: CH.MO PROF. ING. ALDO ROSSI

LAUREANDO: SIMONE MINTO

ANNO ACCADEMICO 2010-2011

alla mia famiglia...

*“One equal temper of heroic hearts,
Made weak by time and fate, but strong in will
To strive, to seek, to find, and not to yield”*

ALFRED TENNYSON, ULYSSES, 1833

Indice

Sommario	XIII
Introduzione	XV
1 Progetto Sophia3	1
1.1 Introduzione ai sistemi cable-based	1
1.2 Stato dell'Arte	2
1.3 Sistemi a cavi in letteratura	4
1.4 Robot a cavi adattativi	7
1.5 Esperienza del gruppo Mechatronics	8
1.6 Obiettivi del lavoro di Tesi	12
2 Real-time Ethernet fieldbus: EtherCAT	15
2.1 Introduzione	15
2.2 Trasferimento dati	16
2.3 Protocollo	18
2.4 Clock Distribuiti	19
2.5 Prestazioni	21
2.6 Topologia	22
2.7 Profili e protocolli supportati	23
2.8 Master Implementation	24
2.9 Slave Implementation	25
3 Modelli ed Algoritmi	27
3.1 Cinematica Diretta	27
3.1.1 Metodo della distanza media	29

3.1.2	Metodo di predizione	33
3.1.3	Compensazione dell'elasticità dei cavi	36
3.2	Analisi dell'accuratezza di posizionamento	37
3.3	Algoritmo di autocalibrazione	41
3.4	Calcolo della distribuzione delle forze	48
3.4.1	Scalatura delle tensioni	50
3.4.2	Compensazione della forza di gravità	50
3.5	Controllo della posizione del motore mobile	51
4	Descrizione dell'Hardware	55
4.1	Controllore Beckhoff CX1020-0111	55
4.1.1	Unità di alimentazione CX1100-0004	57
4.2	Moduli EtherCAT	58
4.2.1	EtherCAT Coupler	58
4.2.2	Input digitali	59
4.2.3	Input analogici	60
4.2.4	Output analogici	61
4.2.5	Output digitali	62
4.2.6	Encoder	63
4.2.7	Collegamenti ingressi-uscite ai dispositivi EtherCAT	64
4.3	Controllo del motore in corrente continua: tecnica PWM	64
4.4	Gestione input/output controllore	69
4.4.1	Layout d'insieme	69
4.4.2	Sensori induttivi	71
5	Descrizione del Software	75
5.1	Windows CE	75
5.1.1	Tecnologie e supporti	76
5.1.2	Caratteristiche di sicurezza e robustezza	78
5.2	TwinCAT System Manager	79
5.2.1	TwinCat I/O	82
5.2.2	Configurazione dei dispositivi EtherCAT	83
5.3	Programmazione driver Cello	88

5.3.1	Controllo in coppia	89
5.3.2	Controllo in velocità	90
5.4	Software di Controllo	91
5.4.1	Metodi di sincronizzazione tra thread	92
5.4.2	Comunicazione tra thread	92
5.4.3	Oggetti implementati	93
5.4.4	I thread attivi	97
5.4.5	Implementazione della macchina a stati finiti	99
5.4.6	Protocollo di comunicazione	101
5.5	Controllo del motore DC: macchina a stati	102
5.6	Revisione automatica del codice: cppcheck	105
6	Risultati sperimentali	107
	Conclusioni	113
	Bibliografia	115

Sommario

Questa tesi si inserisce nel contesto del progetto **Sophia3** (**String Operated Planar Haptic Interface for upper-Arm rehabilitation**), che ha come obiettivo finale lo sviluppo di una macchina ad interfaccia aptica a cavi per la neuroriabilitazione degli arti superiori. Un sistema aptico è un dispositivo studiato per interagire con l'operatore umano, con l'intenzione di fornire la sensazione di essere a contatto con una impedenza meccanica variabile. Nel campo medico si adatta ad operazioni complesse quali la simulazione di interventi chirurgici, la telemanipolazione chirurgica e la riabilitazione. Una apposita interfaccia grafica tridimensionale assicura il coinvolgimento del paziente e facilita la gestione della terapia da parte del fisioterapista. Presso i laboratori del DIMEG (Dipartimento di Innovazione Meccanica e Gestionale) a Padova è stato realizzato un dispositivo per il recupero funzionale degli arti superiori a tre gradi di libertà per pazienti, post-ictus, affetti da deficit motori di origine neurologica. Il lavoro di questa Tesi consiste nel set-up del prototipo, nell'ingegnerizzazione e sviluppo del Sistema di Controllo attraverso fieldbus EtherCAT e la modellizzazione di algoritmi cinematici. Gli strumenti usati per lo sviluppo del software di controllo sono il linguaggio C/C++, il sistema operativo Hard-RT Windows CE ed il controllore Beckhoff CX1020.

Introduzione

Questa tesi si inserisce nel contesto del progetto **Sophia3** (**String Operated Planar Haptic Interface for upper-Arm rehabilitation**), che ha come obiettivo finale lo sviluppo di una macchina ad interfaccia aptica a cavi per la neuroriabilitazione degli arti superiori.

Il **primo capitolo** è introduttivo al lavoro svolto. Vengono illustrate le caratteristiche dei robot a cavi facendo riferimento in particolare alle attrezzature riabilitative realizzate precedentemente nei laboratori del DIMEG e descritti brevemente i robot a cavi adattativi.

Nel **secondo capitolo** viene riportata un'analisi approfondita del fieldbus EtherCAT, descrivendo il protocollo di comunicazione utilizzato e la strategia di trasferimento dei dati. Vengono anche analizzate le caratteristiche peculiari di questa tecnologia come, ad esempio, la flessibilità e le alte prestazioni di utilizzo. Viene brevemente descritta l'implementazione dei dispositivi master e slave EtherCAT.

Il **terzo capitolo** tratta la parte relativa alla modellistica e agli algoritmi sviluppati. In particolare viene descritto l'algoritmo di cinematica diretta, analizzata l'incertezza di calibrazione, descritto il metodo per l'autocalibrazione del prototipo e per il calcolo della distribuzione dei cavi.

Nel **quarto capitolo** vengono descritte le componenti hardware del controllo, i dispositivi di input ed output utilizzati e le modifiche apportate all'impianto

elettrico.

Il **quinto capitolo** contiene la descrizione del software di controllo utilizzato, oltre che ad un' introduzione del sistema operativo hard real-time Windows CE.

Nel **sesto capitolo** sono descritti i primi risultati sperimentali ottenuti con il prototipo.



Figura 1: Prototipo Sophia3

Capitolo 1

Progetto Sophia3

1.1 Introduzione ai sistemi cable-based

I robot azionati a cavi presentano notevoli vantaggi rispetto ai manipolatori tradizionali a membri rigidi, primi fra tutti l'economicità, l'elevata payload-ratio e la possibilità di lavorare in ampi spazi di lavoro. Per questi motivi, l'interesse della ricerca accademica ed industriale in questo campo della robotica è andato via via crescendo nel corso degli ultimi due decenni. A dispetto del design piuttosto semplice, la peculiarità di queste macchine, consistente nell'attuazione unilaterale, comporta una certa complessità nel sistema di controllo, la cui efficace implementazione è subordinata ad una conoscenza dettagliata del modello cinematico e dinamico del robot. Per la stessa ragione, le performance del dispositivo risultano strettamente dipendenti non solo dai parametri geometrici della macchina, ma anche dalla particolare configurazione della stessa, con una conseguente estrema variabilità delle capabilities del dispositivo all'interno dello spazio di lavoro. Poiché le specifiche progettuali richiedono di garantire un determinato livello di performance minimo all'interno di uno spazio di lavoro predefinito, questa variabilità comporta spesso lo sviluppo di sistemi rispondenti ai task richiesti, ma scarsamente adatti ad essi. Per superare questa limitazione, il gruppo di ricerca Mechatronics afferente al DIMEG ed operante nel settore dei sistemi cable-based ormai da 10 anni, ha recentemente presentato un design innovativo legato all'introduzione di passacavi mobili nella struttura del robot, a partire dal quale è stato sviluppato

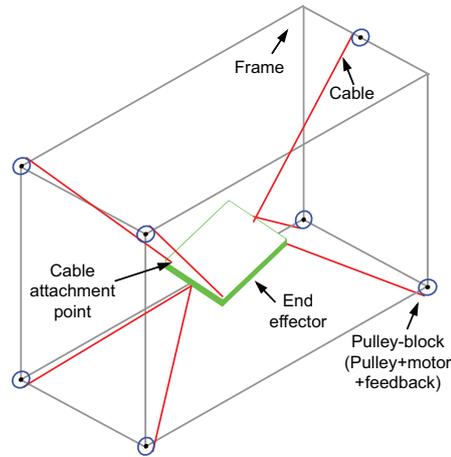


Figura 1.1: Layout di un cable-based system

un nuovo prototipo di interfaccia aptica planare per riabilitazione robot-assistita [1] [2]. Una collaborazione con il Dipartimento di Ing. Meccanica dell'Università del Delaware (Prof. S. K. Agrawal), ha finora portato a formalizzare una nuova metodologia di progettazione per robot a cavi denominati “adattativi”.

1.2 Stato dell'Arte

Un robot cable-based è un sistema ad architettura parallela in cui l'end-effector è sostenuto da cavi, messi in tensione da appositi motori. La differenza fondamentale tra robot tradizionali e robot a cavi risiede nell'attuazione unilaterale (i cavi possono essere sollecitati solo a trazione). In genere, ciascun cavo è avvolto in una puleggia, mentre l'altra estremità è fissata ad un punto dell'end-effector: agendo opportunamente sulle lunghezze e sulle tensioni dei cavi, è possibile far ottenere all'end-effector le pose e forze generalizzate desiderate (Figura 1.1).

Nelle comuni gru, il carico è sostenuto da un unico cavo, a sua volta vincolato ad una struttura rigida seriale, a 2 o più gdl. Movimentare i carichi evitando ondulazioni eccessive degli stessi richiede sistemi di controllo molto complessi, tanto che, ad oggi, non esistono sistemi di questo tipo completamente automatizzati, e le movimentazioni possono considerarsi quasi-statiche. Nei sistemi cable-based, al contrario, il numero di cavi è normalmente superiore al numero di gdl, cosicché il moto del carico può essere controllato finché lo stesso rimane all'interno dello

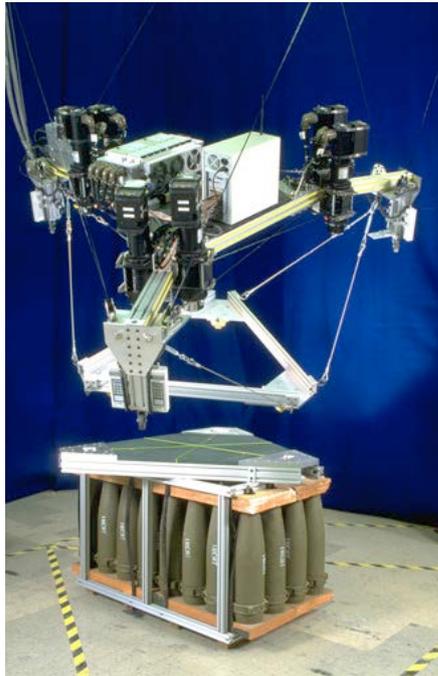


Figura 1.2: Robocrane (NIST)

spazio di lavoro della macchina. In più, l'assenza di link rigidi in movimento permette di raggiungere prestazioni dinamiche elevate, mantenendo nel contempo l'elevata payload-ratio tipica dei sistemi crane-type tradizionali (Figura 1.2).

Altri aspetti positivi sono: la semplicità e l'economicità del design, la facilità di riconfigurazione e trasporto e la possibilità di gestire ampi spazi di lavoro (Figura 1.3).

Inoltre, nel caso di dispositivi interagenti con un operatore, l'attuazione unilaterale rende il sistema più sicuro, perché intrinsecamente cedevole, e l'assenza di membri rigidi in movimento permette di ridurre l'impedenza meccanica percepita dall'operatore (Figura 1.8).

Del resto, non mancano gli aspetti negativi: la presenza dei cavi, se non correttamente gestita, può provocare collisioni con gli ostacoli presenti nello spazio di lavoro, e l'elasticità dei cavi, se non compensata, riduce l'accuratezza di posizionamento. Lo studio dello workspace della macchina (luogo dei punti raggiungibili dal baricentro dell'end-effector, in una data orientazione ed in condizioni di equilibrio statico) e la successiva deduzione delle performance della macchina al suo interno comportano non poche difficoltà, tanto che questi rimangono a tutt'oggi i principali



Figura 1.3: Skycam



Figura 1.4: SACSO-9

temi affrontati dalla letteratura correlata.

1.3 Sistemi a cavi in letteratura

Si deve al contributo di Y. Shen, H. Osumi e T. Arai [4] la prima significativa modellizzazione matematica dei sistemi a cavi. In seguito, diversi autori hanno approfondito design meccanico, control design e design analysis. P. Lafourcade and M. Llibre si sono occupati del design di un sistema a cavi per test sperimentali di modelli in galleria del vento [5] (Figura 1.4).

Gli stessi autori hanno proposto un metodo grafico per determinare facilmente lo spazio di lavoro di sistemi con un numero di gdl pari od inferiore a 3. Tale metodo, estendibile con qualche approssimazione anche a sistemi più complessi,

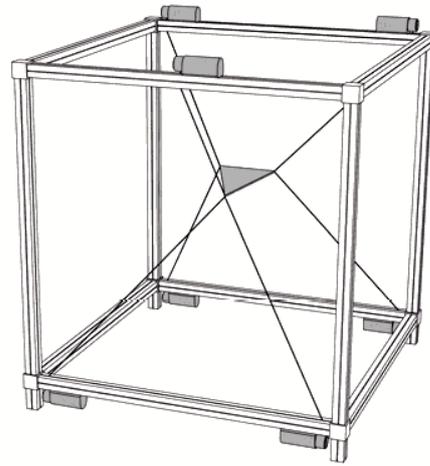


Figura 1.5: SEGESTA

è pensato per agevolare i progettisti nelle prime fasi del design. M. Hiller and R. Verhoeven hanno sviluppato un approfondito studio teorico sulle piattaforme di Stewart *tendon-based* [6] (Figura 1.5), occupandosi di caratterizzazione dello spazio di lavoro, calcolo ottimale delle tensioni dei cavi, cinematica e pianificazione delle traiettorie.

S. Krut, O. Company and F. Pierrot hanno evidenziato l'inadeguatezza degli indici di performance comunemente utilizzati per i manipolatori a membri rigidi - e basati sulla matrice Jacobiana - quando applicati ai sistemi a cavi, proponendo indici alternativi per quantificare le performance di forza e di velocità di queste macchine [7], [5]. Recentemente, M. Gouttefarde e S. Krut [8] hanno proposto un algoritmo numerico per determinare il design ottimale di sistemi completamente vincolati, tali da soddisfare assegnate performance di forza all'interno di un dominio spaziale predefinito. A.T. Riechel, P. Bosscher e I. Ebert-Uphoff si sono occupati di sistemi *cable-suspended sottovincolati* (sistemi per cui $m < (n + 1)$, con m numero di cavi ed n numero dei gdl, (Figura 1.6), sviluppando una serie di tools geometrici di analisi, adottati successivamente anche da altri autori. Oltre allo studio delle caratteristiche di forza e delle caratteristiche dinamiche [9], questi autori hanno affrontato anche altri aspetti, quali la resistenza ad azioni esterne statiche o impulsive [10].

X. Diao and O. Ma hanno studiato algoritmi per la determinazione dello spazio



Figura 1.6: esempio di sistema *cable-suspended*



Figura 1.7: STRINGMAN

di lavoro di sistemi a 6 gdl *completamente vincolati* ($m = (n + 1)$) e *sovra-vincolati* ($m > (n + 1)$) [12]. In [11], D. Surdilovic and R. Bemhardt hanno proposto un robot a cavi per la riabilitazione degli arti inferiori (Figura 1.7).

Nello stesso lavoro, gli autori presentano un metodo di ottimizzazione del design, la cui funzione di costo tiene conto del numero di condizionamento e dell'omogeneità del kernel della matrice Jacobiana.

1.4 Robot a cavi adattativi

L'idea cardine alla base del nuovo metodo di progettazione consiste nell'introduzione di uno o più passacavi mobili (*pulley-block*), per ottimizzare le prestazioni della macchina al variare della posizione assunta dall'EE[3]. Quasi tutte le proprietà locali di un sistema a cavi sono strettamente dipendenti dalla configurazione assunta dai cavi per assegnata posa dell'end-effector. Poiché, all'interno dello spazio di lavoro, questa disposizione è variabile, ne consegue un'estrema variabilità delle performance della macchina, il che si scontra con le usuali specifiche di progetto, che spesso richiedono che un certo livello di performance sia garantito all'interno di un volume di lavoro predefinito. Di conseguenza, ciò porta a progettare sistemi scarsamente adattati alla performance goal.

Un metodo immediato per risolvere il problema consiste nell'aumentare il numero di cavi impiegati, ma ciò comporta un aumento della probabilità di collisione tra gli stessi ed un maggiore assorbimento di potenza. Al contrario, se, per una data performance goal, si deduce dapprima la configurazione ottimale dei cavi, e quindi si progetta un sistema di guide per i pulley-block (resi mobili), in modo che tale configurazione possa essere sempre garantita all'interno dello spazio di lavoro, il risultato è un sistema maggiormente adatto alle specifiche, e quindi meno sovradimensionato. Il sistema ottimizzato così ottenuto è in grado di modificare la disposizione dei pulley-block in funzione della posa dell'end-effector, in modo da garantire costantemente la configurazione migliore dei cavi rispetto ad un determinato parametro di performance. In altre parole, per ciascuna posa dell'end-effector, il sistema si comporta come una macchina a cavi tradizionale, che realizza il massimo valore dell'indice di performance nella posa corrente: di conseguenza, un sistema di questo tipo può essere a ragione identificato come adattativo. In un recente lavoro [2] sono stati sottolineati, con riferimento a semplici sistemi planari, i vantaggi che l'introduzione di uno o più passacavi mobili danno in termini di performance di forza, di potenza ed inerziali rispetto a sistemi tradizionali aventi lo stesso numero di cavi, od anche numero maggiore di cavi.



Figura 1.8: FeRiBa-3, PiRoGa-5

1.5 Esperienza del gruppo Mechatronics

I primi dispositivi a cavi realizzati presso i laboratori DIMEG sono stati dei display aptici general-purpose, uno planare (FeRiBa-3), l'altro 3D (PiRoGa-5). Contestualmente alla progettazione dei singoli dispositivi, sono stati sviluppati degli indici di performance (forza esercitabile, inerzia percepita, potenza assorbita, ecc.) utili per l'ottimizzazione del layout in sede di design preliminare. Inoltre, per tutti i dispositivi planari progettati, sono stati sviluppati algoritmi diretti per il calcolo della distribuzione ottimale delle tensioni dei cavi, che hanno il pregio di ridurre notevolmente i tempi di calcolo. Recentemente, infine, si è proposto un nuovo approccio nel design di sistemi a cavi, basato sull'introduzione di passacavo mobili.

L'esperienza maturata nel campo delle interfacce aptiche a cavi è stata recentemente trasferita al campo della riabilitazione dell'arto superiore di pazienti post-stroke, grazie alla collaborazione nata tra il Gruppo di Robotica del DIMEG e la cattedra di Riabilitazione dell'Università di Padova. È noto, infatti, che uno degli approcci più efficaci per ristabilire le funzionalità dell'arto superiore consiste nell'esecuzione intensiva di movimenti mirati e ripetitivi, passivi o attivi. L'utilizzo di macchine interagenti con l'uomo in questo genere di applicazioni consente di incrementare la produttività del fisiatra e di monitorare più efficacemente i progressi del paziente, attraverso l'elaborazione dei dati registrati dal robot. In



Figura 1.9: NeReBot

quest'ottica sono stati progettati e costruiti due prototipi di robot a cavi, a 3 e 5 gradi di libertà, impiegabili nel trattamento di soggetti con emiplegia in fase sub-acuta.

Il primo prototipo, denominato NeReBot (NEuroREhabilitationroBOT), è stato validato clinicamente presso l'Ospedale S. Antonio di Padova, ed ha fornito risultati molto positivi. Il sistema consiste in una struttura trasportabile atta a supportare l'arto superiore del paziente mediante un'apposita ortesi, collegata ai 3 bracci della struttura principale mediante altrettanti cavi. I 3 bracci superiori possono essere ruotati rispetto alla colonna centrale di sostegno, e la posizione del passacavo in ciascun braccio può essere regolata a piacere, così da permettere l'aggiustamento manuale della configurazione rispetto alle esigenze del paziente, prima dell'inizio delle terapie. Una volta definita la tipologia di esercizio, il fisiatra guida il braccio del paziente attraverso delle locazioni di riferimento: in questa fase la macchina provvede a registrare le locazioni, mantenendo sempre una tensione minima ai cavi. Successivamente essa assiste il paziente nella ripetizione dell'esercizio, assicurando inoltre un continuo feedback visivo ed acustico, volto a mantenerne alto il livello di attenzione.

Dai risultati dei test clinici si nota che i pazienti sottoposti alla terapia assistita da robot in aggiunta alla terapia usuale, mostrano un maggiore recupero della disabilità motoria e delle funzionalità dell'arto. I risultati ottenuti hanno incoraggiato il gruppo di ricerca a proseguire su questa strada, realizzando un secondo robot, denominato MariBot (MARIsaroBOT), nato come evoluzione del precedente, con l'intento di superare le limitazioni del NeReBot. Tra queste si sottolinea la struttura difficilmente trasportabile, la necessità per il fisiatra di impostare manualmente la configurazione della macchina, le prestazioni del sistema di controllo. Il nuovo prototipo a cavi modifica radicalmente la precedente struttura meccanica: i 3 passacavi sono ora vincolati ai due link di un braccio meccanico planare a 2 gdl., azionato in modo da seguire i movimenti del paziente durante l'esecuzione della terapia. La struttura principale è costituita da una colonna regolabile di derivazione commerciale, dotata di una guida lineare motorizzata: in questo modo lo spazio di lavoro della macchina diviene simile a quello dell'arto, inoltre il peso della struttura viene ridotto e il set-up iniziale può essere effettuato più velocemente. Durante la fase di addestramento, i cavi vengono ancora mantenuti ad una tensione minima, mentre il braccio meccanico, a motori spenti, segue liberamente i movimenti del paziente. La parte software del prototipo è attualmente in corso di sviluppo, e nel prossimo futuro l'attenzione verrà posta sul miglioramento dell'interazione macchina-paziente, al fine di poter somministrare terapie più efficaci.

Allo stesso tempo sono stati realizzati due dispositivi planari cable-based denominati Sophia (Sophia4 e Sophia3). Nonostante il loro semplice design, questo genere di dispositivi offre prestazioni paragonabili e talvolta superiori a quelle dei dispositivi equivalenti a membri rigidi. Questi prototipi, la cui struttura deriva dal display aptico FeRiBa3, sono pensati per il trattamento di pazienti cronici all'interno di strutture riabilitative decentralizzate. La struttura portante di Sophia4 è costituita da un tavolo per ufficio: l'EE può essere traslato sulla superficie del tavolo, scivolando su dischi in PTFE. I 4 cavi sono vincolati ad uno stesso punto sull'asse dell'EE e, passando attraverso appositi passacavo, sono avvolti in pulegge poste al di sotto del tavolo.

Il design di Sophia3 condivide molte delle soluzioni adottate in Sophia4, con

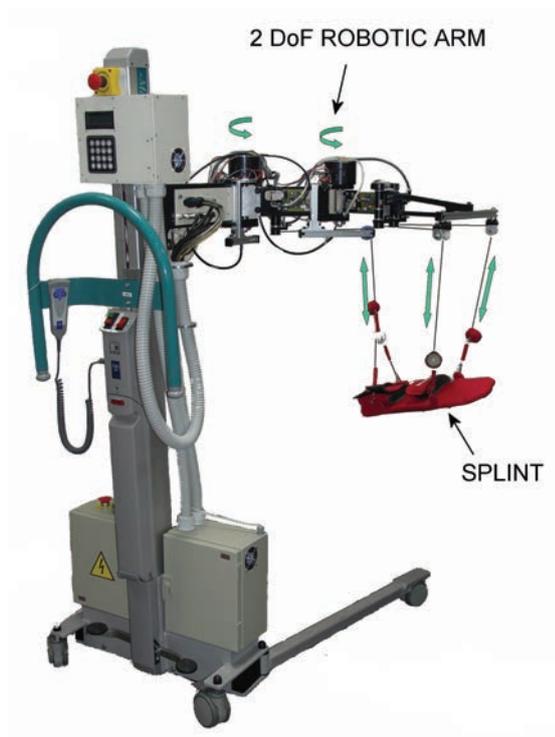


Figura 1.10: MariBot



Figura 1.11: Sophia4, Sophia3

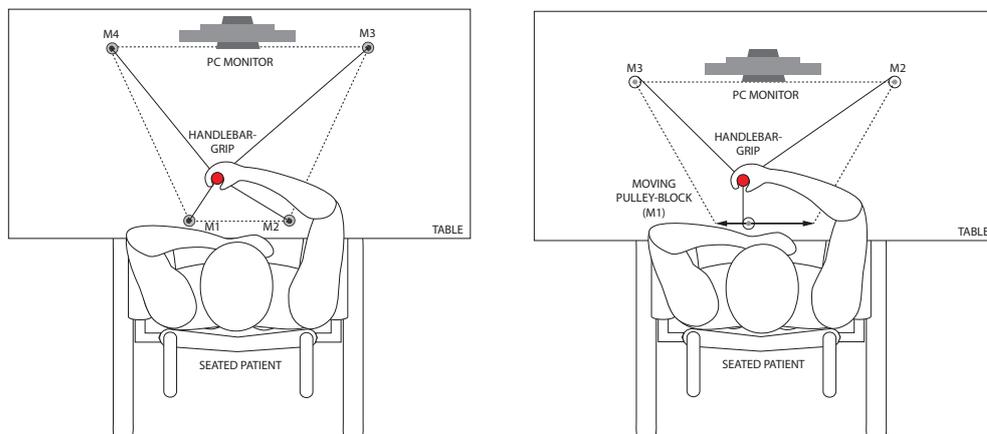


Figura 1.12: Top view: Sophia-4, Sophia-3

alcune significative modifiche: una struttura modulare ad-hoc sorregge il piano di lavoro, che ora può ruotare attorno ad un asse longitudinale, posto vicino alle ginocchia del paziente. Al fine di ridurre l'attrito, i passacavo sono stati eliminati, e sul piano di lavoro è applicato un foglio di PTFE. Il moto traslatorio del carrello porta-puleggia, ottenuto tramite vite a ricircolo ed appositamente studiato per incrementare le caratteristiche inerziali e le performance di forza dell'interfaccia, rappresenta il principale contributo di questo dispositivo alla ricerca sui dispositivi a cavi. Il controllo della macchina deve interagire con il Sistema Gestione Task, ovvero il software che funge da interfaccia con il fisiatra per la definizione dei compiti motori proposti al paziente, e con il paziente stesso per il feedback grafico sull'esercizio in corso. L'esercizio consiste nell'esecuzione di compiti motori di reaching punto-punto, attraverso percorsi liberi oppure entro percorsi predefiniti (canali): il robot riceve in real-time le posizioni previste ed assiste in modo semi-attivo il paziente nell'esecuzione del task. Esso sposta l'EE nella direzione dettata dalla forza applicata all'EE, intervenendo con azioni correttive qualora il paziente stenti ad eseguire l'esercizio oppure esca dal canale.

1.6 Obiettivi del lavoro di Tesi

La Tesi prevede lo sviluppo del sistema di controllo di un robot planare aptico a cavi a tre gradi di libertà, facendo utilizzo di un PC industriale e del bus di

campo Ethercat. Si dovrà studiare un algoritmo di cinematica diretta maggiormente robusto, un algoritmo di auto-calibrazione ed analizzare l'incertezza di posizionamento.

Capitolo 2

Real-time Ethernet fieldbus: EtherCAT

2.1 Introduzione

Negli ultimi anni la tecnologia dei Fieldbus è divenuta parte integrante e fondamentale per la realizzazione di un sistema di automazione, aumentando in modo significativo la possibilità di integrare e di realizzare applicazioni di controllo basati su hardware generico (PC). Mentre le performance dei controllori CPU, in particolare IPC e embedded PC, sono aumentate rapidamente, i fieldbus tradizionali tendevano a rappresentare un limite di prestazione non facilmente superabile per le realizzazioni di impianti di automazioni [17].

Un ulteriore fattore negativo era rappresentato dall'architettura del sistema di controllo che consisteva in una serie di sottosistemi subordinati (usualmente ciclici). Il passo fondamentale per superare tali problematiche è stato quello di utilizzare collegamenti Ethernet non solo per creare una rete tra i controllori, ma per sostituire i fieldbus nelle aree di applicazione dei driver e a livello I/O. Le caratteristiche principali per questo tipo di applicazione sono la capacità di supportare caratteristiche temporali deterministiche “real-time” per piccole quantità di dati e la diminuzione dei costi di impianto.

Lo standard EtherCAT (Ethernet Control Automation Technology) è un protocollo di comunicazione ad elevate prestazioni per connessioni Ethernet deter-

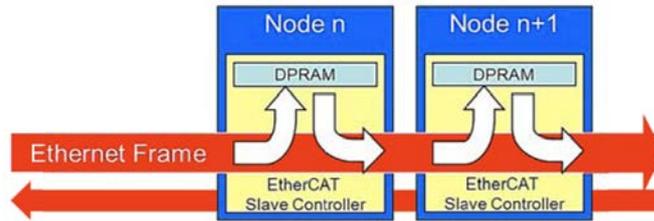


Figura 2.1: Telegramma modificato “on the fly” [21]

ministiche. Esso estende lo standard Ethernet IEEE 802.3 al trasferimento dati con una temporizzazione prevedibile ed esatta sincronizzazione. Questo standard aperto è stato rilasciato come parte integrante dello standard IEC 61158 e viene comunemente utilizzato in applicazioni quali progettazione di macchine e controllo assi.[19] .

2.2 Trasferimento dati

Il principio fondamentale che sta alla base del sistema di comunicazione EtherCAT è quello di utilizzare un “Pass-through telegram”, ovvero i dati da inviare vengono inseriti in un telegramma che attraversa i vari dispositivi inseriti nella rete, ritornando successivamente al controllore. Il telegramma viene modificato “on the fly” durante il passaggio da ogni dispositivo EtherCAT slave (come illustrato in figura 2.1), che legge l’indirizzo contenuto nelle informazioni di processo, e inserisce, se necessario, i dati di input. Le dimensioni delle informazioni di processo utilizzate da ogni slave sono quasi illimitate e vanno da un minimo di 1 bit al massimo di 60 kbyte (vengono usati più frame se necessario) [20] .

La compilazione delle informazioni di processo viene modificata in modo sincrono ad ogni ciclo di controllo ed è possibile in aggiunta modificarla in modo asincrono tramite “event triggered communication” (comunicazione attivata da eventi). Ogni EtherCAT slave riceve il pacchetto Ethernet inviato dal master e lo ritrasmette automaticamente al dispositivo successivo. L’intero processo avviene sia in fase di lettura che in fase di scrittura con un leggero ritardo per ogni nodo non superiore ai 60 ns [17] .

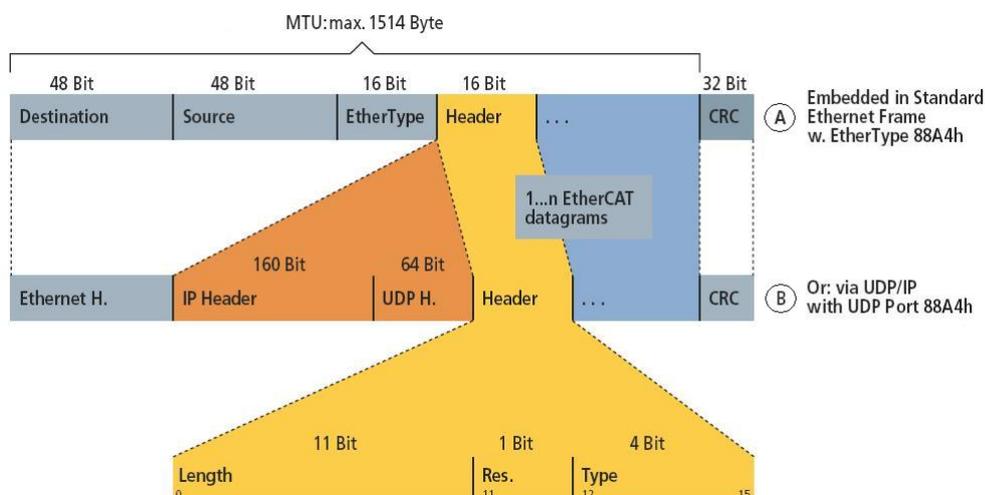


Figura 2.2: Protocollo di trasmissione EtherCAT : A) Via Ethernet (EtherType 0x88A4); B) Via UDP (User Datagram Protocol, UDP port 0x88A4); [17]

Il telegramma spedito dal controllore master passa da un dispositivo all'altro fino ad arrivare all'ultimo della configurazione, il quale rispedisce le informazioni modificate lungo il passaggio al master. Il fatto che il pacchetto venga trasmesso continuamente da uno slave all'altro, ne garantisce la continua presenza su tutti i dispositivi. Con questa tecnica di comunicazione si riesce così ad ottimizzare la velocità di trasferimento e l'ampiezza di banda garantendo un certo grado di determinismo.

Questo tipo di strategia di trasferimento dati è possibile grazie all'aumento della miniaturizzazione dei componenti elettronici, in particolare al miglioramento dell'hardware di comunicazione EtherCAT e all'utilizzo di terminali dotati di ESC (EtherCAT Slave Controller). Questo ha permesso di sostituire le interfacce tipicamente presenti nell'IPC con interfacce intelligenti implementate nei terminali del sistema EtherCAT. L'architettura degli slave è stata rivisitata e semplificata, demandando il compito di lettura e scrittura dei telegrammi "on the fly" all'ESC, con il vantaggio di operare direttamente sugli I/O digitali, in modo indipendente dalle performance dei μC installati e senza ritardi introdotti dai firmware locali.

2.3 Protocollo

Il protocollo EtherCAT implementa un'architettura master/slave usufruendo di una connessione Ethernet in accordo con lo standard IEEE 802.3 che consente l'utilizzo di ogni tipo di controllore con interfaccia Ethernet (master), e di applicazioni che supportano tale standard (es. qualsiasi tipo di monitoraggio disponibile in commercio) [17]. Il protocollo EtherCAT prevede due tipi di implementazioni. Il primo tipo è ottimizzato per processare dati e trasportarli direttamente all'interno del frame Ethernet grazie ad un speciale Ethertype (0x88A4h).

Esso consiste in diversi sotto-telegrammi, definiti *Datagram*, ognuno dei quali serve una particolare area di memoria dell'immagine processo logico che può arrivare fino a 4GB di grandezza. Ciascun *Datagram* ha un indirizzo verso uno o più slave; questa combinazione di "dati e indirizzi" (con l'aggiunta del conteggio di validazione WC) forma un telegramma EtherCAT. Il secondo tipo di implementazione consente la comunicazione via UDP/IP prevista quando si ricorre ad un router per interfacciarsi con subnet diverse. In questo caso al protocollo vengono aggiunti i campi IP Header e UDP header. Le operazioni di codifica e la presenza di più subnet richiedono un tempo di risposta della rete maggiore, con conseguente perdita di determinismo.

Entrambe le implementazioni prevedono la rilevazione dei bit errati durante il trasferimento attraverso la valutazione del checksum CRC: il CRC a 32 bit ha un polinomio con una distanza di hamming minima di 4 [19]. Il protocollo EtherCAT garantisce una piena trasparenza per la comunicazione TCP/IP e rende disponibili tutte le tecnologie Internet (HTTP, FTP, Webserver, ...) senza restrizioni alle prestazioni real-time [20]. L'ordine dei dati è indipendente dalla posizione fisica dei terminali nella rete e l'inserimento dei dati di input da parte di un qualsiasi terminale viene fatto nel primo datagram vuoto. Tale procedura è garantita dalla Fieldbus Memory Management Unit (FMMU) presente nei slave EtherCAT della rete, con compito di trasferimento dei Datagram e la verifica degli indirizzi. I dati vengono copiati dai terminali I/O nell'allocazione desiderata direttamente senza aver bisogno di un'ulteriore mappatura. La memoria disponibile per gli indirizzi è molto ampia e può arrivare fino ai 4 Gbytes.

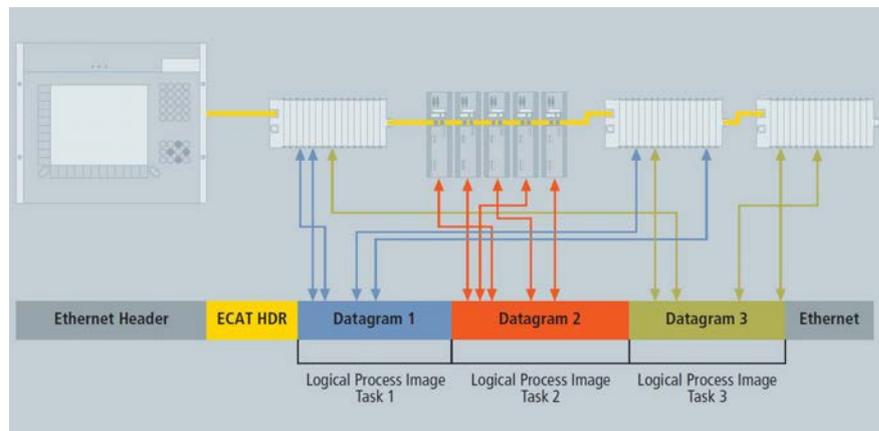


Figura 2.3: Le informazioni di processo sono inserite nei Datagram. [17]

EtherCAT può contenere più telegrammi, e spesso sono necessari più frames per gestire tutti i telegrammi in un ciclo di controllo. Il protocollo EtherCAT garantisce anche la comunicazione Broadcast, Multicast, slave-to-slave, master-to-master. Per quanto riguarda la comunicazione fra EtherCAT slave esistono due meccanismi utilizzati. Il primo dipende dalla topologia della rete e prevede che se abbiamo due dispositivi disposti in serie uno prima dell'altro, la comunicazione possa avvenire tra il primo slave EtherCAT e il successivo nello stesso ciclo e molto velocemente. In alternativa la comunicazione può avvenire con uno scambio di informazioni tramite master, impiegando così almeno due cicli [17].

2.4 Clock Distribuiti

Nel caso di processi spazialmente distribuiti, che richiedono azioni simultanee, una sincronizzazione accurata è particolarmente importante. L'approccio utilizzato dalla rete EtherCAT è l'allineamento accurato dei clock distribuiti, come descritto nello standard IEEE 1588, che prevede un alto grado di tolleranza rispetto a possibili ritardi dovuti ad errori di comunicazione.

La sincronizzazione dei dispositivi è completamente basata su una macchina hardware. Dal momento che la comunicazione si avvale di una struttura logica ad anello, ogni nodo della rete può misurare la differenza tra il passaggio di arrivo e quello di ritorno di un frame. Il clock principale, avvalendosi di tali misurazioni,

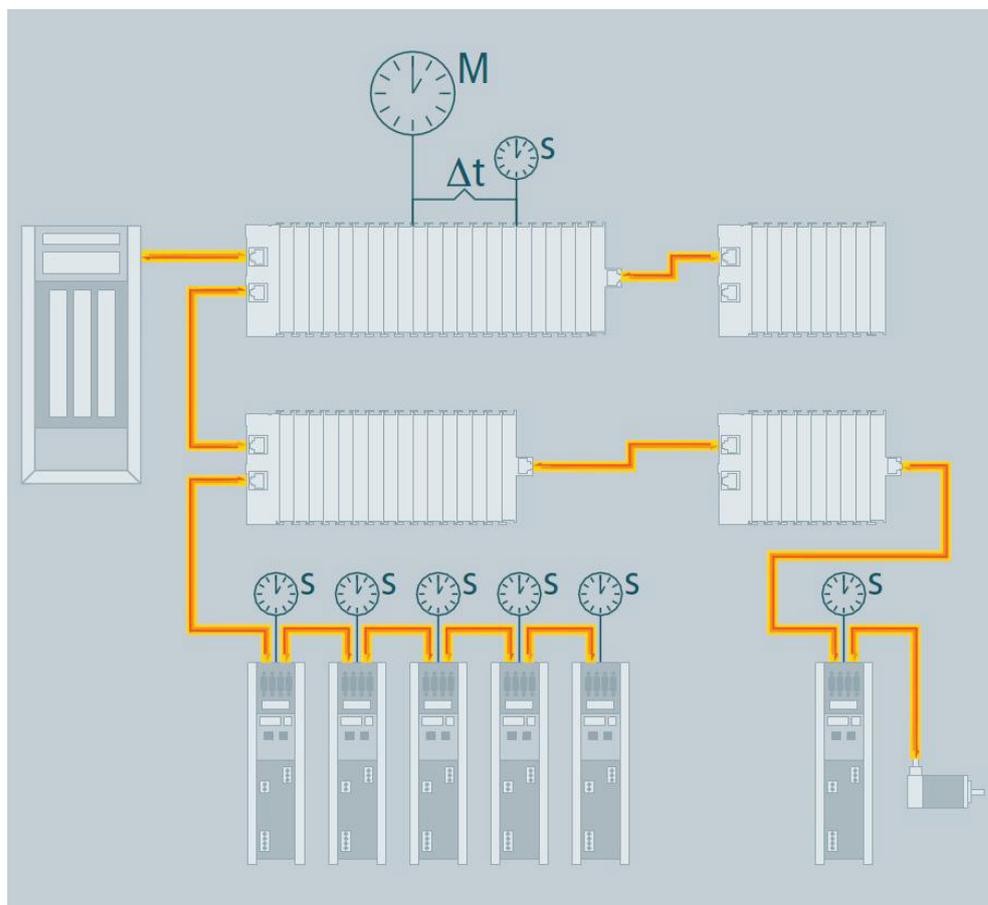


Figura 2.4: Layout di un cable-based system

Figura 2.5: Distributed Clock :I clock di ogni singolo dispositivo vengono impostati sulla base dei tempi a livello di rete. [17]

può determinare il ritardo di propagazione di ogni singolo slave in modo semplice ed accurato. I clocks distribuiti sono corretti in base a questo valore, tale approccio permette di avere un preciso tempo di base, con un jitter minore di $1 \mu s$ [17] .

Tuttavia, l'alta risoluzione dei clock distribuiti non viene solo usata per la sincronizzazione, ma risulta molto importante soprattutto nell'ambito della misurazione della posizione, consentendo l'aumento della velocità di calcolo del controllore, provvedendo ad un'accurata informazione circa il timing locale del dato da acquisire.

2.5 Prestazioni

Il protocollo EtherCAT è progettato per raggiungere elevate prestazioni e per gestire un alto numero di canali in applicazioni di controllo. Poiché la fase di lettura e scrittura dello slave può avvenire nello stesso frame, la struttura del telegramma EtherCAT è ottimizzata per gestire sistemi di I/O decentralizzati. L'utilizzo di frame Ethernet, che comprende i dati di molte periferiche sia in ingresso che in uscita, fa aumentare il data rate utilizzabile di oltre il 90%. Sono pienamente utilizzate le caratteristiche full duplex di 100BaseTX.

Fino a 1486byte di dati di processo possono essere scambiati con un unico frame. Inoltre, la completa elaborazione del protocollo avviene all'interno dell'hardware e pertanto è indipendente dalle prestazioni della CPU e da ogni componente software. Ad esempio, tramite l'accesso diretto alla memoria (DMA), è possibile trasferire i dati tra la scheda di rete e il processore master, oppure gli I/O dello slave, con un coinvolgimento minimo della CPU. Gli slave, che dispongono di unità fieldbus per la gestione della memoria (FMMU) dedicati al trasferimento dei PDO e la verifica degli indirizzi, sono incaricati di mappare i telegrammi appropriati riducendo in questo modo il lavoro del master.

La tecnica dei clock distribuiti consente di sincronizzare gli assi con una deviazione minore di 1 $m\mu s$. Tali caratteristiche consentono di raggiungere determinate prestazioni come [17] :

- 256 I/O digitali in 11 μs
- 1000 I/O digitali distribuiti su 100 nodi in 30 μs
- 200 I/O analogici (16 bit) in 50 μs , corrispondenti a 20 kHz di sampling rate
- 100 "Servo-axes" (con 8 Byte In + Out) ogni 100 μs
- 12000 I/O digitali in 350 μs

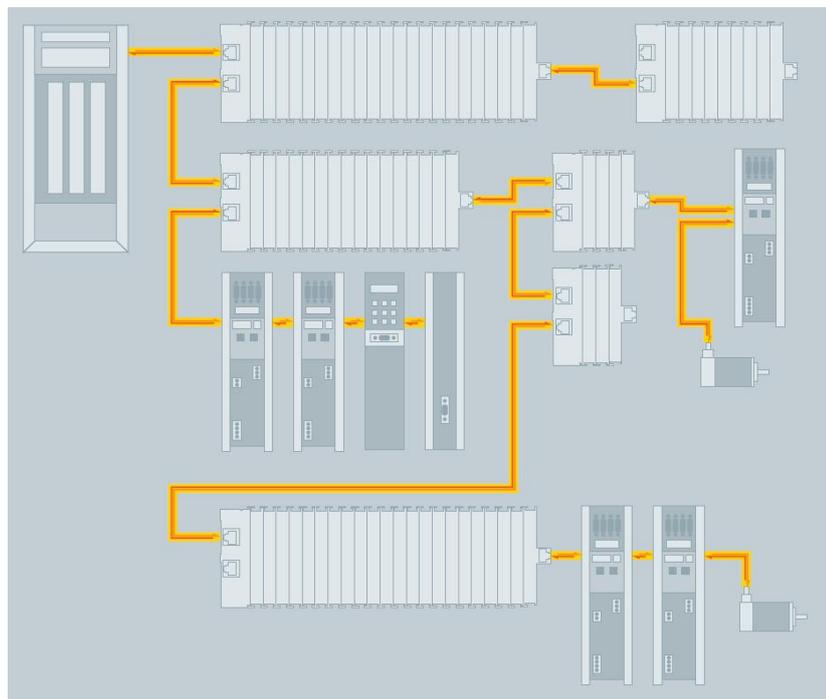


Figura 2.6: Topologia flessibile : linea, albero o stella [17]

2.6 Topologia

EtherCAT supporta numerosi tipi di topologia di rete : linea, albero o stella (vedi figura 2.6). La struttura di linea viene riconosciuta dal fieldbus e diventa così disponibile anche per Ethernet, senza le limitazioni quantitative dovute alla presenza di switch e di hub [17] . Particolarmente utile per il cablaggio del sistema è la possibilità di avere una combinazione di linee e di rami: le interfacce richieste sono già predisposte su molti dispositivi (ad esempio sui moduli I/O), senza bisogno di ulteriori switches.

La flessibilità di collegamento e il basso costo di cablaggio sono ulteriormente ottimizzati attraverso l'utilizzo di cavi differenti : cavi industriali Ethernet standard per trasferire segnali in modalità 100BASE-TX (consente l'utilizzo di un cavo di lunghezza 100 m tra due dispositivi, fino a 65535 dispositivi possono essere collegati assieme e la dimensione della rete può superare i 500 Km); fibre ottiche 100BASE-FX o Plastic Optical Fibres per integrare il sistema con applicazioni speciali (consente l'utilizzo di un cavo di lunghezza variabile da 50 m a 2000 m tra due dispositivi).

2.7 Profili e protocolli supportati

L'utilizzo di dispositivi con profili differenti o di altri protocolli viene garantito grazie a semplici interfacce che aiutano, sia gli utenti che i produttori di dispositivi, nel gestire il passaggio da bus di campo esistenti a EtherCAT (vedi figura 2.7). Allo stesso tempo, la specificazione EtherCAT si mantiene semplice perché tutti i protocolli sono opzionali. Il produttore del dispositivo deve solo implementare il protocollo che l'applicazione del dispositivo necessita. Diversi profili di dispositivi e protocolli possono essere supportati su rete EtherCAT :

- CAN application protocol over EtherCAT (CoE) : dispositivi CANopen e profili di applicazione sono disponibili (componenti di I/O, azionamenti, encoder, valvole proporzionali idraulici, per esempio). EtherCAT è in grado di fornire gli stessi meccanismi di comunicazione di CANopen (object Dictionary, PDO Processo Data Objects e SDO Service Data Objects).
- Servo drive profile in accordo con IEC 61800-7-204 (SERCOS over EtherCAT SoE) : l'interfaccia SERCOS è riconosciuto come un'interfaccia di comunicazione real-time ad alto rendimento. Il profilo SERCOS per servo unità e la tecnologia di comunicazione sono coperti dallo standard IEC 61800-7-204. Il canale di servizio, e quindi l'accesso a tutti i parametri e le funzioni, si basa sulla mailbox EtherCAT. I dati di processo, con SERCOS in forma di AT e dati MDT, vengono trasferiti utilizzando il meccanismo e il protocollo EtherCAT.
- Ethernet over EtherCAT (EoE) : la tecnologia EtherCAT non solo è compatibile con Ethernet, ma è caratterizzata anche da apertura particolare "by design". Il protocollo tollera infatti altri servizi e protocolli basati su Ethernet sulla stessa rete fisica (con minima perdita di prestazioni). Non vi è alcuna restrizione sul tipo di dispositivo Ethernet che può essere collegato all'interno della rete EtherCAT attraverso una switchport. I frame Ethernet sono inseriti tramite protocollo EtherCAT, rappresentando l'approccio standard per le applicazioni Internet (Ad esempio VPN, PPPoE, DSL, ecc.). La rete EtherCAT è completamente trasparente per i dispositivi Ethernet, e

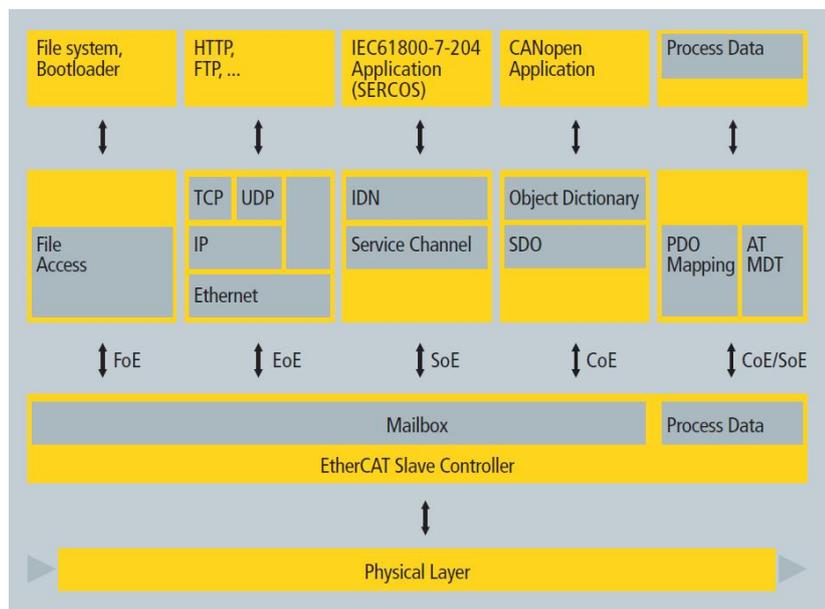


Figura 2.7: Diversi profili di dispositivi e protocolli possono coesistere fianco a fianco [17]

le caratteristiche real-time non vengono compromesse. Tutte le tecnologie Internet possono dunque anche essere utilizzate in ambiente EtherCAT : web server integrato; e-mail; trasferimento FTP ecc.

- File Access over EtherCAT (FoE) : questo protocollo molto semplice, simile al TFTP, consente l'accesso a qualsiasi struttura dati del dispositivo. E' possibile quindi caricare sul dispositivo firmware standardizzati, a prescindere dalla presenza di supporto TCP / IP.

2.8 Master Implementation

EtherCAT comunica un massimo di 1486 *byte* di dati di processo distribuiti con un solo frame Ethernet. Il sistema EtherCAT tipicamente ha bisogno solo di uno o due frame per ogni ciclo per la comunicazione con tutti i nodi. Pertanto il master EtherCAT non necessita di un apposito processore per la comunicazione. Il basso costo computazionale della comunicazione EtherCAT consente di sfruttare la CPU

in maniera minimale per gestire questo compito lasciando maggior spazio per elaborare le applicazioni di controllo.

L'implementazione di un master EtherCAT può essere molto facile, in particolare per sistemi di controllo di piccole e medie dimensioni (ad esempio un PLC con una singola immagine di processo che non supera i 1486 byte). L'immagine di processo non viene mappata nel master EtherCAT ma negli slaves, gravando ulteriormente meno la CPU. Gli EtherCAT masters possono essere implementati su una vasta gamma di RTOS (Real-Time Operating System), inclusi : eCos, INtime, MICROWARE OS-9, On Time RTOS-32, Proconos OS, Real-Time Java, RT Kernel, RT Linux, RTCX Quadros, RTAI Linux, PikeOS, Linux with RT-Preempt, QNX, VxWin + CeWin, VxWorks, Windows CE, Windows XP/XPE con CoDeSys SP RTE, Windows XP/XPE con TwinCAT RT e XENOMAI Linux [17] . Servizi di implementazione sono disponibili da una varietà di fornitori e per una varietà di piattaforme hardware. In un ambiente embedded l'implementazione del Master EtherCAT deve avere alcuni requisiti speciali: l'ambiente hardware comprende una variazione massima di micro controllori disponibili (8, 16 o 32 bit); l'utilizzo di diversi chip MAC per affrontare diversi metodi di accesso ai dati (accesso DMA, accesso IO, accesso Dual Port alla RAM); l'utilizzo di differenti sistemi operativi o, in alcuni casi, di nessun sistema operativo.

Il beneficio di EtherCAT in un ambiente embedded è dato dalla possibilità di avere un master scalabile, utilizzando alcune caratteristiche o protocolli speciali, per sfruttare in modo ottimale la memoria e la CPU. Questo è possibile grazie al fatto che un master non necessita di un hardware speciale ma può essere implementato in software.

2.9 Slave Implementation

Lo slave EtherCAT non ha bisogno di un microcontrollore integrato, in quanto la sua architettura prevede una semplice interfaccia I/O implementata solo con un ESC (EtherCAT Slave Controller) , un PHY (Physical Layer) e i connettori RJ45. La process data interface (PDI) per l'applicazione di uno slave è un'interfaccia I/O 32 bit. Lo slave senza parametri configurabili non ha bisogno di software o

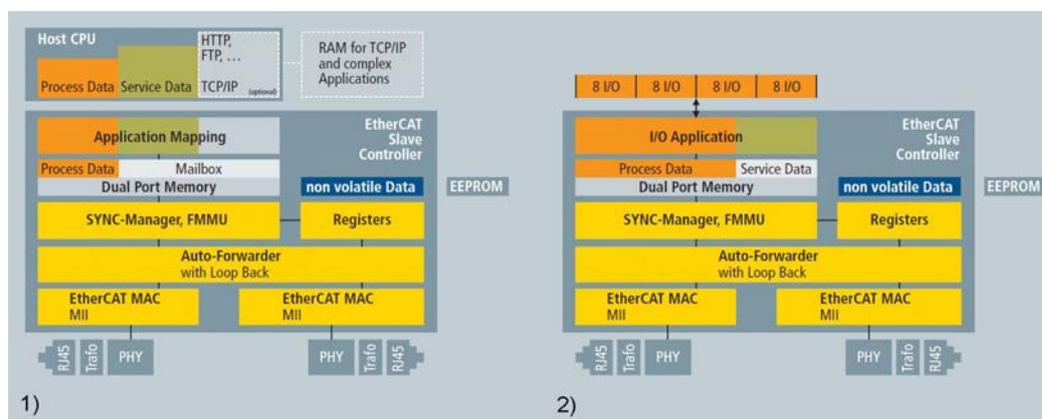


Figura 2.8: Implementazione Hardware dello slave : 1) con host CPU; 2) con interfaccia I/O diretta [17]

di mailbox protocol. L'EtherCAT State Machine è gestito nel ESC che prende le informazioni di boot-up dalla EEPROM, che supporta anche le informazioni di identità dello slave.

Gli slave più complessi, che sono configurabili, montano una CPU host a bordo, collegata all'ESC con un interfaccia (8-16 bit) in parallelo o tramite una connessione seriale (SPI). Diversi fornitori offrono servizi di implementazione degli slave, comprensivi di una integrazione hardware e software. Le varie EtherCAT Controller Slave sono supportate da corrispondenti evaluation kits, rendendo tutte le interfacce del controllore facilmente accessibili allo sviluppatore. Dal momento che con EtherCAT non sono necessari potenti processori per la comunicazione, il kit sopra indicato contiene un microprocessore a 8 bit che può essere anche utilizzato come host CPU. Il kit viene fornito con uno slave host software, l'equivalente di uno stack di protocolli in codice sorgente, e un pacchetto di riferimento software per il master.

Capitolo 3

Modelli ed Algoritmi

Il prototipo Sophia-3 è composto da due motori fissi (M2 e M3) e da uno mobile (M1), il quale è movimentato attraverso un ulteriore motore (M4) mediante una vite a ricircolo. Il motore mobile ha la funzionalità di seguire la posizione dell'EE al fine di aumentare le prestazioni del controllo.

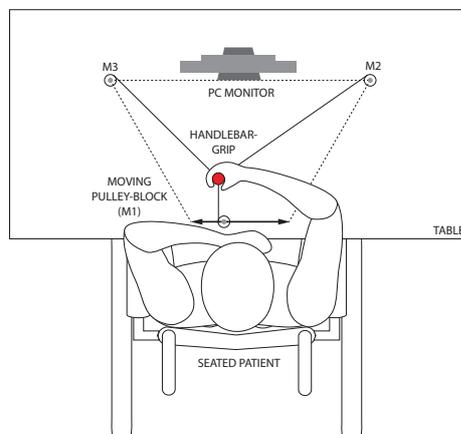


Figura 3.1: Immagini descrittiva della disposizione spaziale delle varie pulegge e e della conformazione del motore mobile

3.1 Cinematica Diretta

La cinematica diretta del Sophia-3 era stata sviluppata a partire dal concetto di intersezione di evolventi.

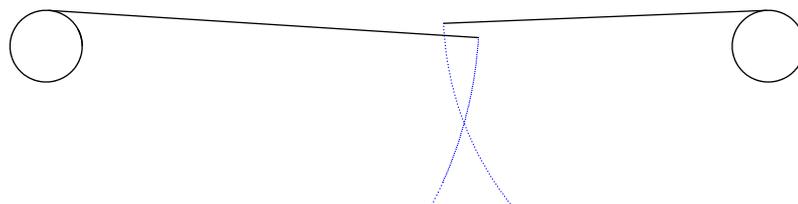


Figura 3.2: Esempio di intersezione di evolventi

I cavi delle pulegge fisse sono considerati inestensibili, pertanto la figura geometrica che l'estremità del cavo descrive, durante lo svolgimento, è un'evolvente. Il punto in cui si troverà l'end effector appartiene nello stesso tempo alle due distinte curve, quindi le coordinate della posizione cercata si ricaveranno calcolando l'intersezione delle due evolventi, che hanno le circonferenze di base coincidenti rispettivamente con la puleggia 2 e con la puleggia 3, secondo l'esempio di figura 3.3. Il punto di partenza per sviluppare la trattazione è la definizione di 3 sistemi di riferimento per ciascuna puleggia. La convenzione per i segni di tutti gli angoli che verranno introdotti, prevede di considerare positivi gli angoli concordi al verso di svolgimento del cavo su ciascuna puleggia. Si considerino ora le pulegge 2 e 3. Si individuano due sistemi di riferimento fissi $\{x_{01}, y_{01}, z_{01}\}$ e $\{x_{02}, y_{02}, z_{02}\}$ in cui ciascun asse y_0 è diretto verso il centro della puleggia contigua, gli assi z_0 e y_0 completano la terna ricordando che l'asse z individua il verso di svolgimento dei cavi (z_{01} sarà uscente dal foglio, z_{02} sarà entrante). Il secondo sistema di riferimento $\{x_t, y_t, z_t\}$ è quello che individua il punto di tangenza sulla puleggia. L'asse x_t sarà perpendicolare al cavo in qualsiasi configurazione durante la rotazione. Gli assi z e y saranno posizionati seguendo le stesse considerazioni fatte in precedenza. In figura 3.3 è raffigurato quanto sopra esposto.

In precedenza per risolvere il problema della cinematica diretta si utilizzava un metodo iterativo, *Newton's Method for Unconstrained Optimization*, ma in talune condizioni non riusciva a fornire il risultato corretto. Questa problematica era da imputare al fatto che tale algoritmo, in alcuni casi, andava in direzione di un'altra soluzione, non accettabile per la zona di lavoro del prototipo. In figura 3.4 è possibile trovare un esempio di tale problematica. Pur confinando la soluzione, il metodo non riusciva a convergere alla soluzione desiderata, indicata nel grafico con il simbolo *. Per la descrizione completa di tale metodo si faccia riferimento a

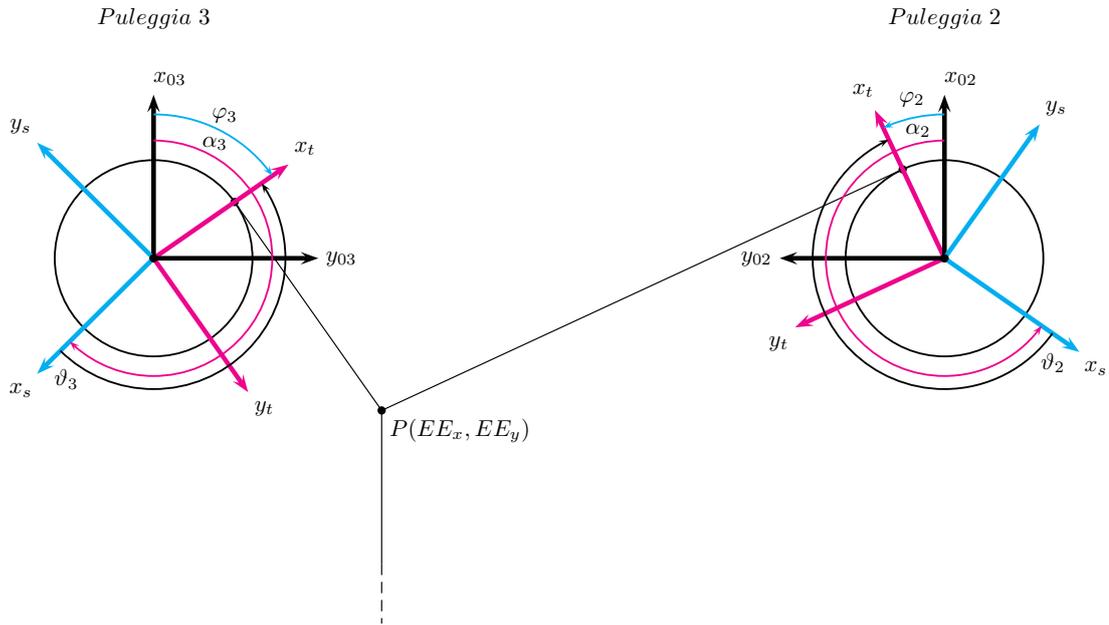


Figura 3.3: Sistemi di riferimento pulegge

[15].

Per superare tali problematiche, è stato sviluppato un nuovo metodo iterativo, che si basa sull'utilizzo di due algoritmi distinti. Per le prime iterazioni si utilizza l'algoritmo "metodo della distanza media", per le altre prima di arrivare a convergenza si utilizza l'algoritmo chiamato, in questa tesi, "metodo di predizione".

3.1.1 Metodo della distanza media

È possibile pensare l'intersezione di due evolventi, come la condizione in cui, le due estremità dei cavi abbiano distanza nulla. Tale metodo, opera in questo modo: data la lunghezza iniziale l_0 di un cavo ed un valore di "inclinazione" iniziale φ , l'estremità del cavo si troverà in una determinata posizione (x_l, y_l) del piano. Ripetendo lo stesso ragionamento per l'altro cavo considerato è possibile stimare il punto di intersezione, facendolo coincidere con il baricentro tra i due punti estremi dei cavi. In figura 3.5 è possibile trovare una descrizione grafica del concetto esposto.

Al fine di fornire una spiegazione dettagliata del metodo, si considera la figura 3.6 per poter descrivere la simbologia ed i versi degli angoli utilizzati. In tabella

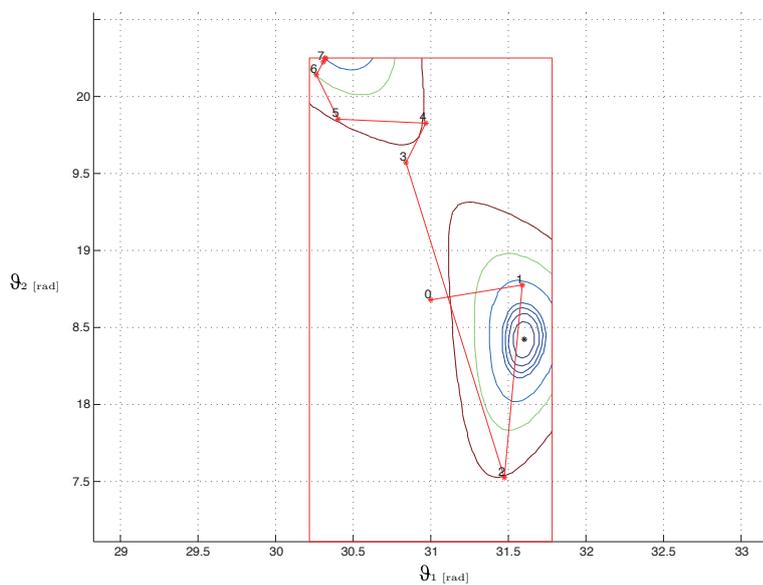


Figura 3.4: Problematiche riscontrate con il precedente algoritmo per il calcolo della cinematica diretta, nel grafico è riportato il valore della funzione $f_1(\theta_1, \theta_2)^2 + f_2(\theta_1, \theta_2)^2$

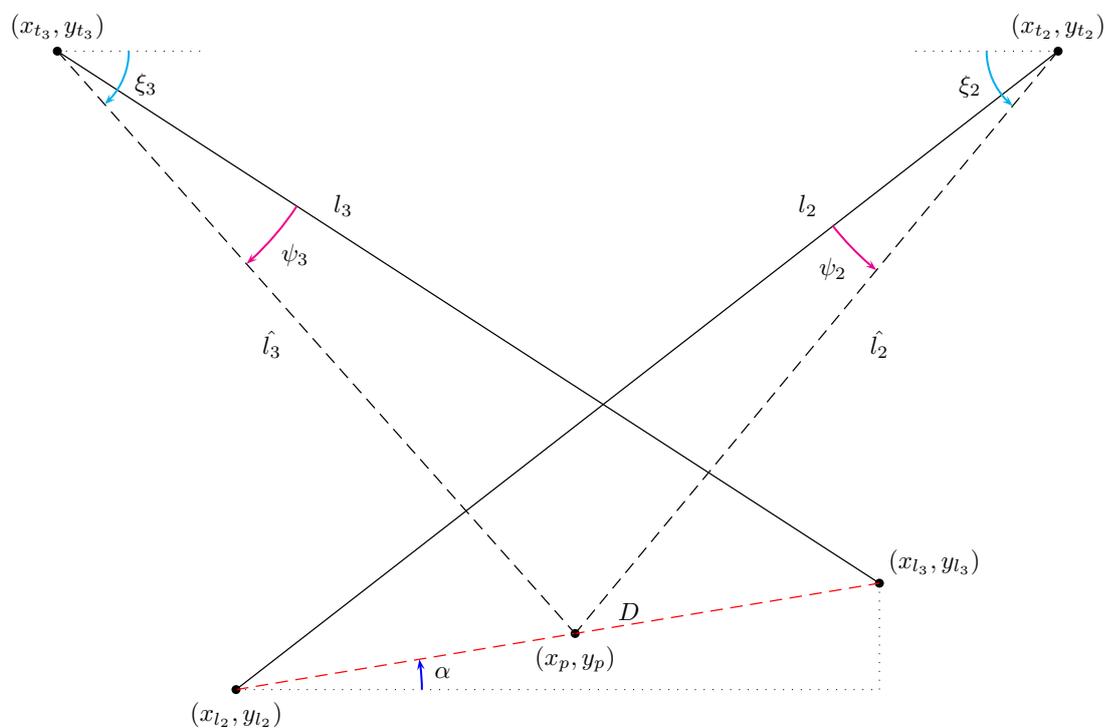


Figura 3.5: Rappresentazione grafica del metodo di media distanza.

3.1 è possibile trovare l'assegnazione delle coppie delle pulegge al numero di combinazione, oltre che i valori delle costanti utilizzate in ogni combinazione.

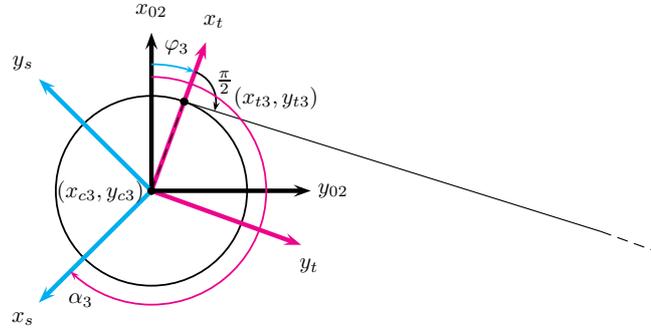


Figura 3.6: Raffigurazione dei punti di tangenza alle pulegge

combinazione	u	v	a	b	φ_b
1 (puleggia 1-2)	1	1	2	1	$\pi/2$
2 (puleggia 1-3)	-1	-1	3	1	$\pi/2$
3 (puleggia 2-3)	-1	1	3	2	$\pi/4$

Tabella 3.1: Tabella raffigurante i valori da assegnare alle costanti nelle varie combinazioni di pulegge utilizzate. In tutti i casi $\varphi_a = \pi/4$

Conoscendo le letture encoder è possibile definire la lunghezza iniziale del cavo mediante la seguente espressione:

$$l_0 = \alpha r_c \quad (3.1)$$

Per la configurazione iniziale dell'angolo φ la lunghezza del segmento l_i sarà pari a:

$$l = l_0 - \varphi r_c \quad (3.2)$$

Si calcola il punto di tangenza del cavo alla puleggia con le seguenti formule:

$$\begin{aligned} x_{ta} &= x_{ca} - u r_{ca} \sin \varphi_a \\ y_{ta} &= y_{ca} + r_{ca} \cos \varphi_a \end{aligned} \quad (3.3)$$

$$\begin{aligned}x_{tb} &= x_{cb} - r_{cb} \sin \varphi_b \\y_{tb} &= y_{cb} + v r_{cb} \cos \varphi_b\end{aligned}\tag{3.4}$$

Ora è possibile calcolare le coordinate del punto estremo di ciascun cavo, come:

$$\begin{aligned}x_{la} &= x_{ta} - u l_a \cos \varphi_a \\y_{la} &= y_{ta} - l_a \sin \varphi_a\end{aligned}\tag{3.5}$$

$$\begin{aligned}x_{lb} &= x_{tb} - l_b \cos \varphi_b \\y_{lb} &= y_{tb} - v l_b \sin \varphi_b\end{aligned}\tag{3.6}$$

Per calcolare la previsione di spostamento, si calcola la distanza D dei punti (x_{la}, y_{la}) e (x_{lb}, y_{lb}) come:

$$D = \sqrt{(x_{la} - x_{lb})^2 + (y_{la} - y_{lb})^2}\tag{3.7}$$

L'angolo α sarà pari a:

$$\alpha = \text{atan2}(y_{la} - y_{lb}, x_{la} - x_{lb})\tag{3.8}$$

Con D e α , il calcolo del punto del baricentro è immediato:

$$\begin{aligned}x_p &= x_{l1} - D/2 \cos \alpha \\y_p &= y_{l1} - D/2 \sin \alpha\end{aligned}\tag{3.9}$$

Dato il punto di tangenza alla puleggia (x_t, y_t) e le coordinate del punto P è possibile ottenere la nuova lunghezza del cavo:

$$\hat{l} = \sqrt{(x_t - x_p)^2 + (y_t - y_p)^2}\tag{3.10}$$

L'angolo ψ , utile al fine di poter condizionare in modo corretto l'angolo φ può essere calcolato attraverso il teorema di Carnot come:

$$\psi = \arccos\left(\frac{l^2 + \hat{l}^2 - D^2}{2l\hat{l}}\right)\tag{3.11}$$

Gli angoli ξ_a e ξ_b sono espressi con la seguente relazione:

$$\xi_a = u \arctan\left(\frac{y_{ta} - y_p}{x_{ta} - x_p}\right) \quad (3.12)$$

$$\xi_b = v \arctan\left(\frac{y_{tb} - y_p}{x_{tb} - x_p}\right) \quad (3.13)$$

Per conoscere il verso virtuale della rotazione che la puleggia dovrebbe fare, è necessario preventivamente condizionare gli angoli come segue. E' possibile ottenere tali condizioni mediante analisi di tipo geometrico.

$$\xi_b = \begin{cases} \xi_b & \text{per } \xi_b \geq 0 \\ \xi_b + \pi & \text{per } \xi_b < 0 \end{cases} \quad (3.14)$$

$$\psi = \begin{cases} \psi & \text{per } \psi \geq \xi \\ -\psi & \text{per } \psi < \xi \end{cases} \quad (3.15)$$

Il risultato della $k + 1$ iterazione può essere espressa come:

$$\varphi_{k+1} = \varphi_k + \psi \quad (3.16)$$

Tale algoritmo viene iterato per tre volte; sarebbe in grado di arrivare a convergenza in modo autonomo, anche se in alcuni punti del piano sarebbero necessarie oltre 70 iterazioni.

3.1.2 Metodo di predizione

Tale metodo sfrutta una predizione del punto di intersezione delle due evolventi, basandosi sul punto di intersezione di due rette. Ciascuna retta viene calcolata basandosi sui due punti precedenti. In figura 3.7 vi è una rappresentazione grafica di quanto sopra esposto. I punti * (rossi) sono calcolati con il metodo della media distanza. I punti * (neri) rappresentano il punto di intersezione delle due rette, mentre i punti * (viola) rappresentano il punto (x_p, y_p) calcolato dall'algoritmo. Si arriva a convergenza quando la distanza tra P_a e P_b è minore di un opportuno Δ .

In figura 3.8 è presente la rappresentazione grafica del metodo, riportando tutta la simbologia utilizzata nella trattazione.

I parametri della retta vengono calcolati nel seguente modo:

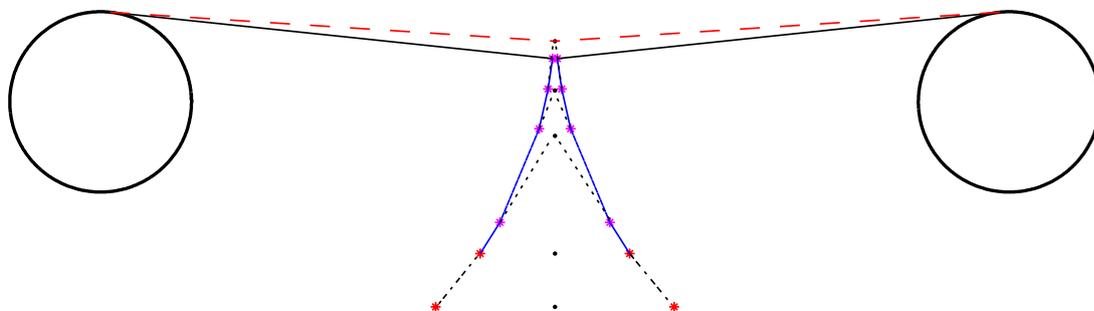


Figura 3.7: Evoluzione della stima di predizione del punto di intersezione tra evolventi

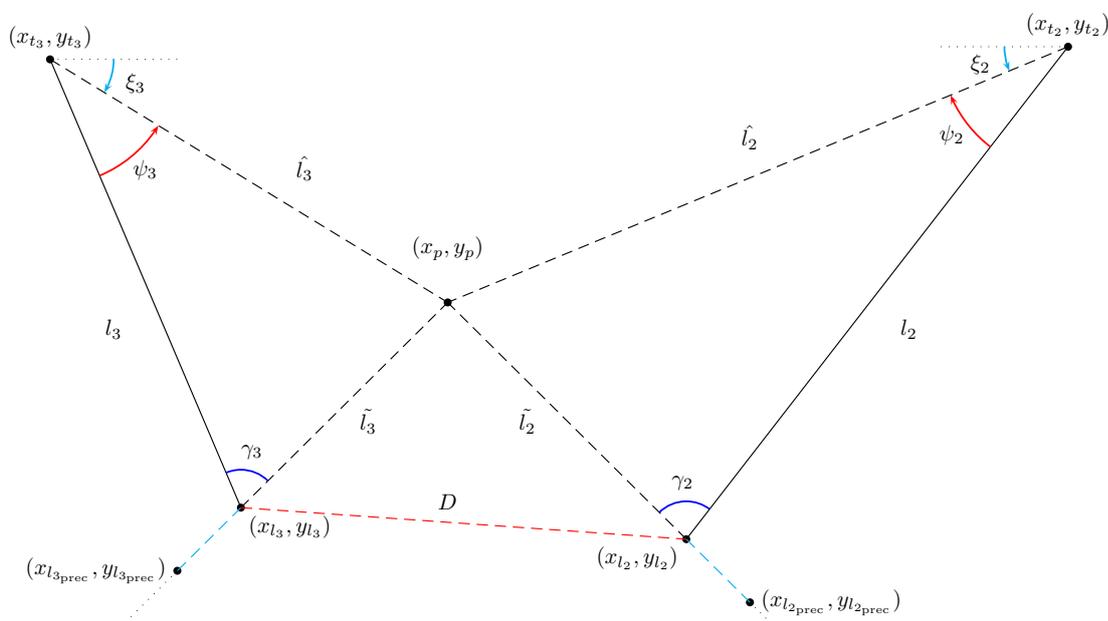


Figura 3.8: Rappresentazione grafica del metodo di predizione

$$\begin{aligned} m &= \frac{y_{l_{\text{prec}}} - y_l}{x_{l_{\text{prec}}} - x_l} \\ q &= \frac{x_{l_{\text{prec}}} y_l - x_l y_{l_{\text{prec}}}}{x_{l_{\text{prec}}} - x_l} \end{aligned} \quad (3.17)$$

mentre il punto di intersezione delle due rette è:

$$\begin{aligned} x_p &= \frac{q_a - q_b}{m_b - m_a} \\ y_p &= m_a x_p + q_a \end{aligned} \quad (3.18)$$

Dato il punto di tangenza alla puleggia (x_t, y_t) e le coordinate del punto P è possibile ottenere la nuova lunghezza del cavo:

$$\hat{l} = \sqrt{(x_t - x_p)^2 + (y_t - y_p)^2} \quad (3.19)$$

ed il valore di \tilde{l} :

$$\tilde{l} = \sqrt{(x_p - x_l)^2 + (y_p - y_l)^2} \quad (3.20)$$

Mediante considerazioni di tipo geometrico si calcola il valore di ψ e ξ , quest'ultimo importante al fine di condizionare l'angolo ψ , mediante le seguenti espressioni:

$$\gamma_i = \arccos\left(\frac{l^2 + \tilde{l}^2 - \hat{l}^2}{2l\tilde{l}}\right) \quad (3.21)$$

$$\xi = -\arctan\left(\frac{y_t - y_p}{x_t - x_p}\right) \quad (3.22)$$

$$(3.23)$$

$$\psi = \arcsin\left(\frac{\sin \gamma \tilde{l}}{\hat{l}}\right) \quad (3.24)$$

Il condizionamento degli angoli avviene secondo gli stessi criteri del metodo precedente, per questo motivo non vengono riportati.

Il risultato dell'iterazione sarà, quindi:

$$\varphi_{k+1} = \varphi_k + \psi \quad (3.25)$$

In figura 3.9 è presente il grafico relativo all'andamento degli scarti di convergenza relativo ai due metodi illustrati e al metodo di Newton. E' possibile notare come la linea continua blu e verde si sovrappongono per le prime tre iterazioni. Questo è dovuto dal fatto che durante il calcolo della cinematica diretta per le prime tre iterazioni si utilizza il *metodo della media distanza*. Dalla quarta iterazione in poi si utilizza il *metodo di predizione*, in corrispondenza del quale è possibile notare un leggero picco.

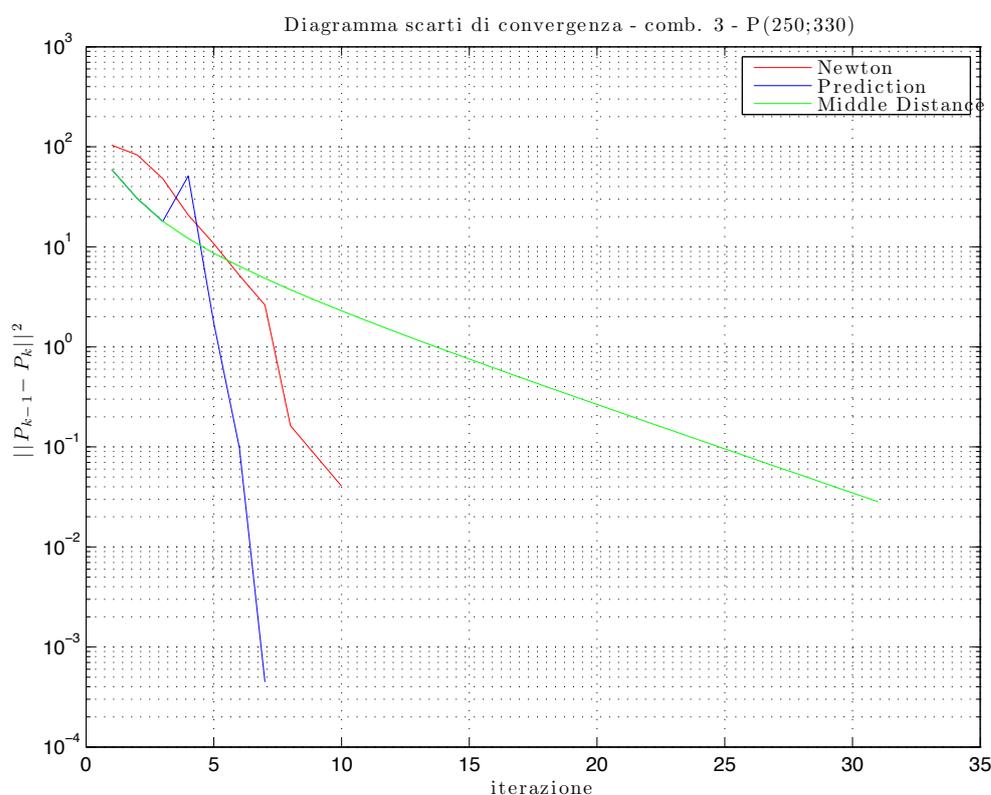


Figura 3.9: Andamento degli scarti di convergenza per i tre metodi di calcolo della cinematica diretta

3.1.3 Compensazione dell'elasticità dei cavi

Il collegamento tra End-Effector e puleggia è stato pensato per l'impiego con cavi di nylon o d'acciaio per questioni di sicurezza, questi saranno sottoposti ad un carico di trazione e, quindi, ad un allungamento. La relazione che lega sforzi e deformazioni è:

$$\sigma = E\epsilon = E \cdot \ln\left(\frac{l}{l_0}\right)$$

da cui si può ricavare:

$$\frac{F}{EA} = \ln\left(\frac{l}{l_0}\right) \implies e^{\frac{F}{EA}} = \frac{l}{l_0} \implies l = l_0 \cdot e^{\frac{F}{EA}} \quad (3.26)$$

dove:

E = modulo di elasticità del cavo [N/mm^2]

A = sezione del cavo [mm^2]

F = tensione del cavo [N]

l = lunghezza attuale del cavo (sottoposto ad una tensione F) [mm]

l_0 = lunghezza del cavo quando la tensione F è nulla [mm]

Si assume che durante la deformazione l'area A rimanga costante e che l'allungamento sia relativo solo al tratto di filo svolto (il filo ancora avvolto sulla puleggia viene considerato indeformabile).

Quindi sostituendo nei calcoli di cinematica diretta il valore della lunghezza iniziale del cavo 3.1, otterremo che:

$$l_0 = \alpha r_c e^{f/EA} \quad (3.27)$$

Con questa modifica e seguendo la procedura di calcolo in precedenza descritta si ricavano le nuove relazioni che tengono conto dell'elasticità dei cavi.

3.2 Analisi dell'accuratezza di posizionamento

In questa sezione viene trattata l'accuratezza di posizione. Dato \mathbf{x} il vettore che indica la posizione dell'End-Effector, il valore \bar{n}_i ottenuto dalla lettura encoder può differire dal valore n_i , puramente teorico, andando a sua volta ad influenzare l'accuratezza della cinematica diretta. Le principali fonti di errori sono da imputare alle tolleranze meccaniche e agli errori di misura. Mediante l'utilizzo di macchine NC ad alta precisione, gli errori dimensionali possono essere trascurati rispetto a quelli di misura. I fattori fondamentali, quindi, saranno:

$$\Delta \mathbf{n} = \Delta \mathbf{n}_{str} + \Delta \mathbf{n}_{res} + \Delta \mathbf{n}_{cal} \quad (3.28)$$

$\Delta \mathbf{n}_{str}$ è dovuto all'allungamento dei cavi: essendo un termine deterministico potrà essere facilmente compensato conoscendo la rigidità dei cavi stessi. Il secondo termine è dovuto alla risoluzione degli encoder, e può essere calcolato come:

$$\Delta n_{res,i} = \pm 0.5 \quad (3.29)$$

L'ultimo termine è relativo all'errore di calibrazione, ovvero all'inesattezza di posizionamento dell'EE in una posizione nota del piano di lavoro, nella quale gli encoder andranno settati in modo opportuno.

Riscrivendo l'equazione 3.28 in termini di α , trascurando il contributo d'errore dato dall'elasticità dei cavi si ottiene:

$$\Delta \boldsymbol{\alpha} = \Delta \boldsymbol{\alpha}_{res} + \Delta \boldsymbol{\alpha}_{cal} \quad (3.30)$$

Il termine dovuto alla risoluzione degli encoder è pari a:

$$\alpha_{res,i} = \pm \frac{\pi r_{c,i}}{N_{Tenc}} \quad (3.31)$$

dove il termine N_{enc} rappresenta la risoluzione dello stesso (tacche encoder/giro). Definito $\Delta \mathbf{x}_{cal}$ l'accuratezza di posizionamento durante il processo di calibrazione, esso produrrà differenti valori di $\boldsymbol{\alpha}_{cal}$ a seconda della collocazione dell'EE. Con l'assunzione di piccoli $\Delta \mathbf{x}_{cal}$, è possibile esprimere $\Delta \boldsymbol{\alpha}_{cal}$ con la seguente relazione linearizzata:

$$\Delta \boldsymbol{\alpha}_{cal} = \mathbf{J} \Delta \mathbf{x}_{cal} \quad (3.32)$$

dove $\mathbf{J} \in \mathbf{M}_{3 \times 2}$, nello specifico:

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \alpha_1}{\partial x} & \frac{\partial \alpha_1}{\partial y} \\ \frac{\partial \alpha_2}{\partial x} & \frac{\partial \alpha_2}{\partial y} \\ \frac{\partial \alpha_3}{\partial x} & \frac{\partial \alpha_3}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\cos \gamma_1 \tan \beta_1 + \sin \gamma_1}{d_1} & \frac{\sin \gamma_1 \tan \beta_1 - \cos \gamma_1}{d_1} \\ -\frac{\cos \gamma_2 \tan \beta_2 + \sin \gamma_2}{d_2} & -\frac{\sin \gamma_2 \tan \beta_2 - \cos \gamma_2}{d_2} \\ \frac{\cos \gamma_3 \tan \beta_3 + \sin \gamma_3}{d_3} & \frac{\sin \gamma_3 \tan \beta_3 - \cos \gamma_3}{d_3} \end{bmatrix} \quad (3.33)$$

con:

$$\begin{aligned}
 d_i &= \sqrt{(x_p - x_{c,i})^2 + (y_p + y_{c,i})^2} \\
 l_i &= \sqrt{d_i^2 - r_{c,i}^2} \\
 \beta_i &= \arctan \frac{l_i}{r_{c,i}} \\
 \gamma_i &= \arctan \frac{y_p - y_{c,i}}{x_p - x_{c,i}}
 \end{aligned} \tag{3.34}$$

Al fine di quantificare il peggior effetto di $\Delta \mathbf{x}_{cal}$ sulle letture degli encoder, si procede prendendo il modulo da entrambi i lati dell'equazione 3.32 ed assumendo $\Delta x_{cal} = \Delta y_{cal} = \Delta s = 1mm$, ottenendo la seguente relazione:

$$\Delta \alpha_{cal,i} = \left(\left| \frac{\partial \alpha_i}{\partial x} \right| + \left| \frac{\partial \alpha_i}{\partial y} \right| \right) \Delta s \tag{3.35}$$

Analizzando singolarmente una componente del vettore $\Delta \alpha_{cal}$, è possibile affermare che dato un punto iniziale (x, y) , se ci si sposta lungo il vettore punto di tangenza alla puleggia-EE, il suo valore sarà massimo, minimo invece se il movimento risulta essere normale ad esso. Tale conclusione può essere giustificata analizzando la figura 3.10, ricavando i risultati riportati in 3.36, specifici, per questo esempio, alla puleggia 3.

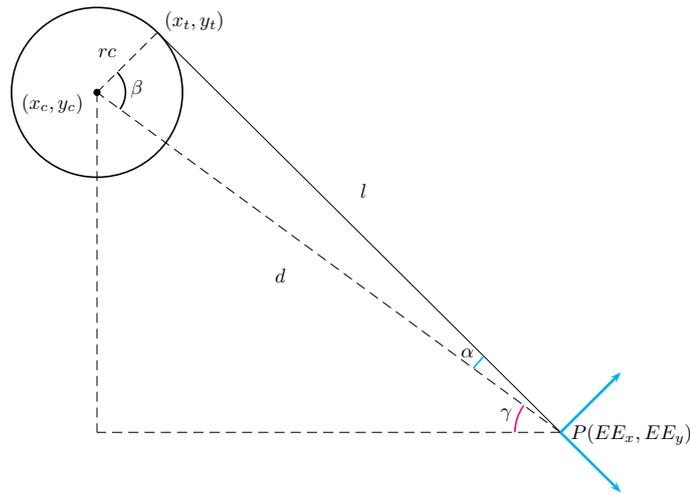


Figura 3.10: Rappresentazione geometrica al fine di evidenziare gli angoli di interesse

$$\begin{aligned} & \frac{\cos \gamma_3 \tan \beta_3 + \sin \gamma_3}{d} \cos\left(\frac{3\pi}{2} + \beta_3 + \gamma_3\right) + \\ & + \frac{\sin \gamma_3 \tan \beta_3 - \cos \gamma_3}{d} \sin\left(\frac{3\pi}{2} + \beta_3 + \gamma_3\right) = \frac{1}{\cos \beta_3 d} = \frac{1}{rc} \end{aligned} \quad (3.36)$$

$$\begin{aligned} & \frac{\cos \gamma_3 \tan \beta_3 + \sin \gamma_3}{d} \cos(\beta_3 + \gamma_3) + \\ & + \frac{\sin \gamma_3 \tan \beta_3 - \cos \gamma_3}{d} \sin(\beta_3 + \gamma_3) = 0 \end{aligned}$$

Gli effetti di $\Delta \boldsymbol{\alpha}$ sulla cinematica diretta possono essere stimati nel seguente modo: sia \mathbf{x}_i la soluzione calcolata trascurando la i -esima lettura encoder, $\Delta \boldsymbol{\alpha}_i$ il vettore delle letture encoder deprivato della i -esima componente, e $\mathbf{J}_i^{-1} \in \mathbf{M}_{2 \times 2}$ l'inversa della matrice quadrata ottenuta da \mathbf{J} eliminando la i -esima riga. L'errore associato con \mathbf{x}_i è dato da:

$$\Delta \mathbf{x}_i = \mathbf{x}_i - \mathbf{x}_{act} = \mathbf{J}^{-1} \Delta \boldsymbol{\alpha}_i \quad (3.37)$$

Una stima dell'attuale posizione può essere calcolata prendendo la media di una combinazione di vettori \mathbf{x}_i . Dal momento che possono essere calcolati tre vettori, sono possibile sette combinazioni. Definito k il numero di soluzioni \mathbf{x}_i prese in considerazione, ne consegue che l'errore associato con \mathbf{x} è dato da:

$$\begin{aligned} \Delta \mathbf{x} &= \mathbf{x} - \mathbf{x}_{act} = \frac{1}{k} \sum_{i=1}^k \mathbf{x}_i - \mathbf{x}_{act} \\ &= \frac{1}{k} \sum_{i=1}^k (\mathbf{x}_i - \mathbf{x}_{act}) = \frac{1}{k} \sum_{i=1}^k \Delta \mathbf{x}_i \\ &= \frac{1}{k} \sum_{i=1}^k \mathbf{J}_i^{-1} \Delta \mathbf{n}_i = \mathbf{H} \Delta \mathbf{n} \end{aligned} \quad (3.38)$$

dove $\mathbf{H} \in \mathbf{M}_{2 \times 3}$ è ottenuta assemblando matrici \mathbf{J}_i^{-1} in accordo con la combinazione di cavi utilizzata, e dividendo il risultato ottenuto per k . Eventualmente, l'accuratezza di posizione può essere stimata dalla norma unitaria di $\Delta \mathbf{x}$:

$$\begin{aligned} \Delta x &= \pm \sum_{j=1}^3 |H_{1,j}| \Delta n_j \\ \Delta y &= \pm \sum_{j=1}^3 |H_{2,j}| \Delta n_j \end{aligned} \quad (3.39)$$

In figura 3.11 sono presenti i grafici relativi all'incertezza di posizione per tutte le possibili combinazioni delle coppie di cavi da utilizzare per effettuare la cinematica diretta. In figura 3.12 è raffigurata l'accuratezza di posizionamento, considerando per ogni punto del workspace, la combinazione che rende l'errore. In figura 3.13 sono presenti le regioni di delimitazione dove le differenti combinazioni di coppie di cavi sono state usate per ottenere la scelta ottima. Il numero della combinazione corrisponde alla traduzione decimale del numero binario a tre bit $[b_{12} b_{13} b_{23}]$, dove b_{ij} è pari a 1 se la coppia $i - j$ è stata utilizzata.

3.3 Algoritmo di autocalibrazione

A livello teorico per conoscere la posizione dell'end effector basterebbe effettuare la cinematica diretta con un'unica coppia di cavi. Praticamente, ciò non è veritiero. La componente di errore maggiore è dovuta all'errore di calibrazione, ovvero all'inesattezza di posizionamento dell'EE in una posizione nota del piano di lavoro. In figura 3.14 è possibile osservare tale problematica: l'errore di calibrazione (simulato) comporta che il risultato della cinematica diretta per le tre combinazioni di cavi sia differente l'una dall'altra. Il pallino viola indica la reale posizione, sempre simulata, dell'end effector mentre gli * stanno ad indicare la soluzione del problema di cinematica diretta con le tre coppie di cavi. E' facile osservare l'errore commesso. Nel caso in cui non vi sia alcun errore di calibrazione, i punti calcolati con le varie coppie di cavi e quello reale sarebbero perfettamente sovrapponibili. Da questa considerazione nasce l'idea di considerare come funzione costo una relazione legata all'area formata dal triangolo, i cui vertici sono le coordinate calcolate attraverso le tre coppie di cavi. Ovvero, con riferimento alla figura 3.15, vale la seguente formula:

$$A = \frac{1}{2} \det \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} \quad (3.40)$$

L'area così calcolata è dotata di segno, positivo se la numerazione dei nodi è antioraria rispetto al sistema di coordinata prescelto.

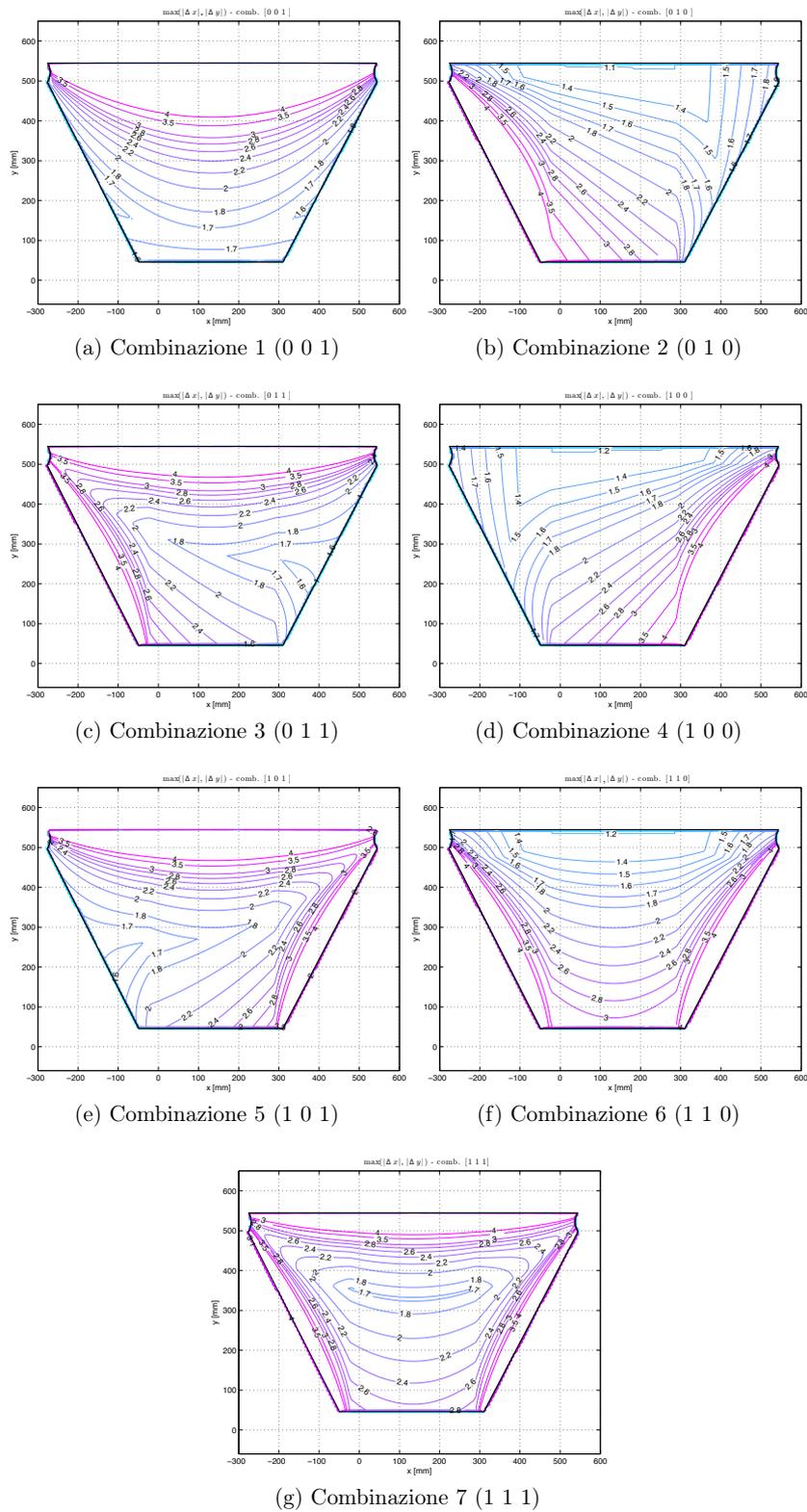


Figura 3.11: Grafici relativi all'incertezza di posizione per tutte le combinazioni possibili

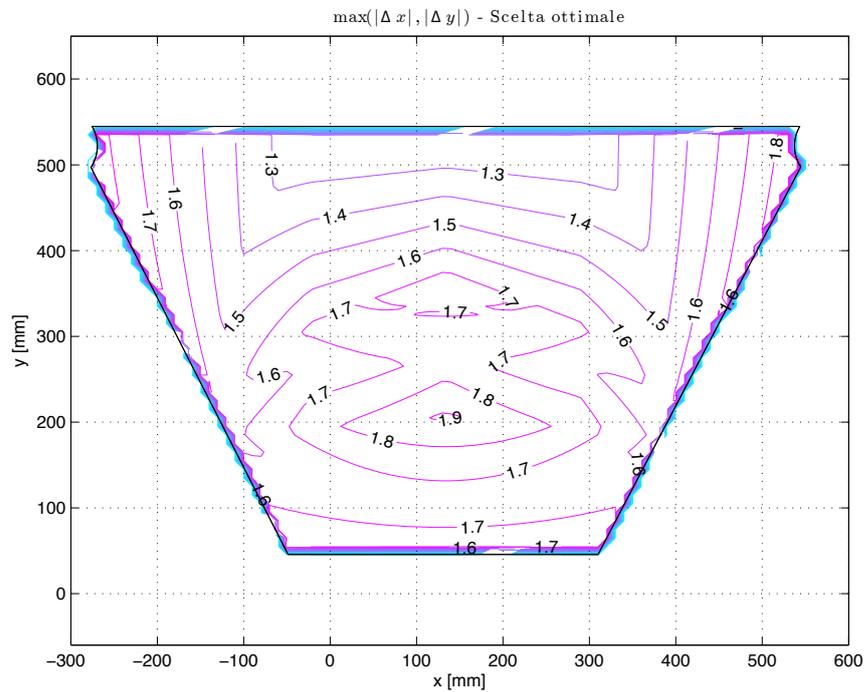


Figura 3.12: Incertezza di posizione considerando per ogni punto dello spazio di lavoro la migliore combinazione

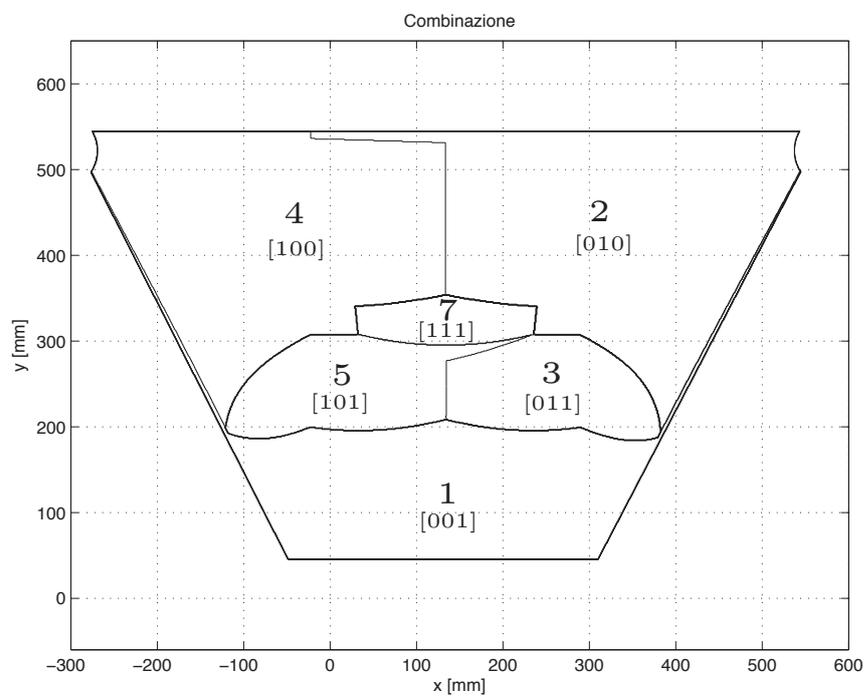


Figura 3.13: Delimitazione delle regioni del piano di lavoro con la scelta ottima delle letture encoder

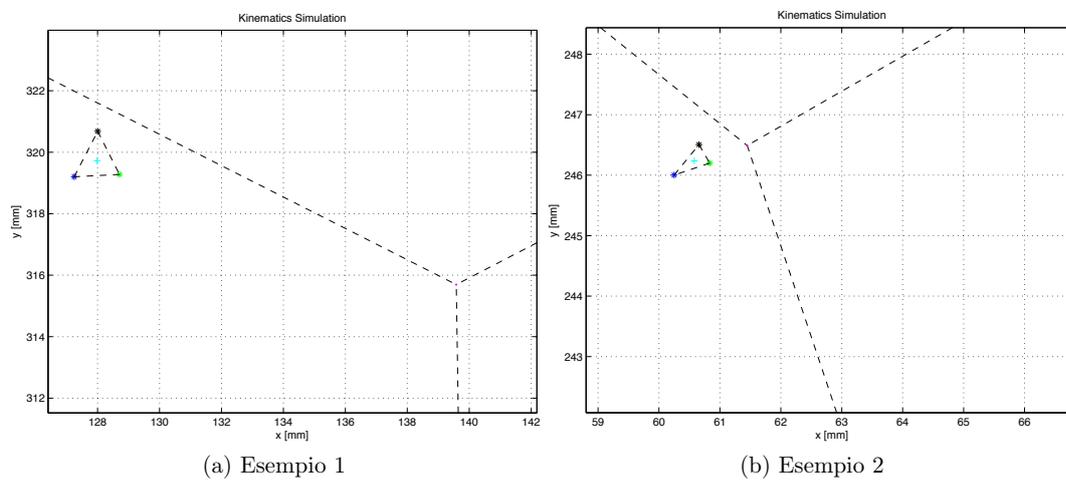


Figura 3.14: Esempi di errore nel calcolo della cinematica diretta dovuto, prevalentemente, all'errore di calibrazione

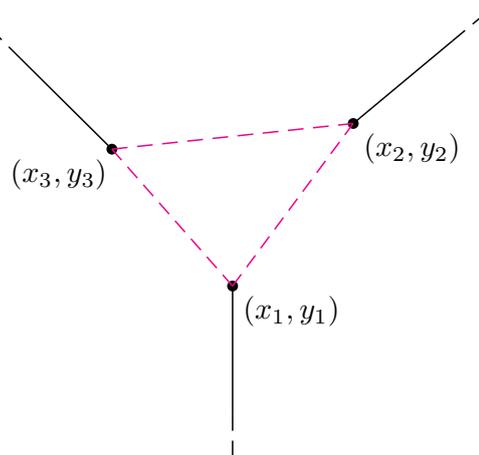


Figura 3.15: Errore triangolo

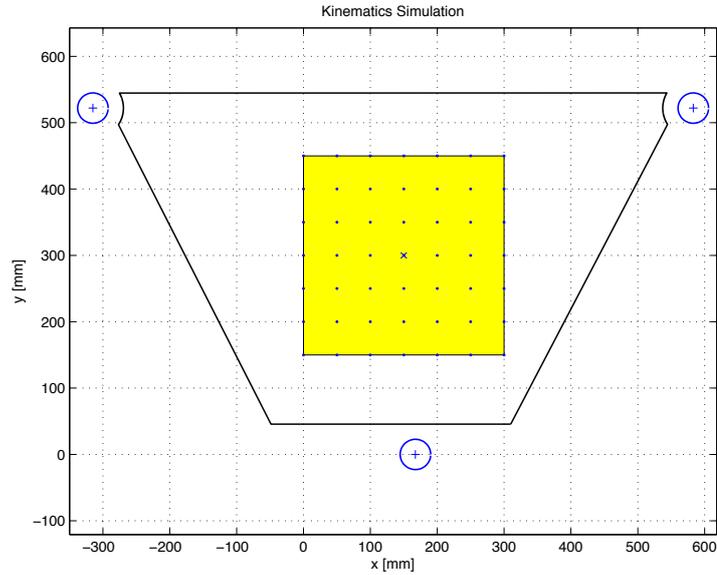


Figura 3.16: Griglia virtuale nel workspace

E' possibile modellizzare le lettura degli encoder in questo modo:

$$\boldsymbol{\alpha}_k = \boldsymbol{\alpha}_0 + \Delta\boldsymbol{\alpha}_k \quad (3.41)$$

dove $\boldsymbol{\alpha}_0$ rappresenta il valore degli encoder nel punto di calibrazione, quindi fisso una volta determinato, mentre il termine $\Delta\boldsymbol{\alpha}_k$ la differenza da quest'ultimo. Il termine $\Delta\boldsymbol{\alpha}_k$ è noto con sufficiente accuratezza.

Con la notazione $\mathbf{x}_{j,i} = [x_{j,i} y_{j,i}]^T$ indicheremo la posizione dell'EE calcolata con la j -esima combinazione e con l' i -esima misura, mentre con \mathbf{x}_m il punto del piano di lavoro in cui si andrà a calcolare la funzione costo di interesse.

Si definisca n il numero di punti presi in considerazione per effettuare l'auto-calibrazione, la funzione $G(\boldsymbol{\alpha})$ che determina la cinematica diretta e la funzione $G^{-1}(\mathbf{x}_m)$ che permette di stabilire la relazione di cinematica inversa.

Indichiamo con $C(x, y)$ la funzione costo:

$$\begin{aligned}
 C(x, y) &= \sum_{i=1}^n A_i^2 \\
 &= \frac{1}{2} \sum_{i=1}^n \det \begin{bmatrix} 1 & 1 & 1 \\ x_{1,i} & x_{2,i} & x_{3,i} \\ y_{1,i} & y_{2,i} & y_{3,i} \end{bmatrix}^2 \\
 &= \frac{1}{2} \sum_{i=1}^n \det \begin{bmatrix} 1 & 1 & 1 \\ G(\boldsymbol{\alpha}_0(\mathbf{x}_m) + \Delta\boldsymbol{\alpha}_i) \end{bmatrix}^2 \\
 &= \frac{1}{2} \sum_{i=1}^n \det \begin{bmatrix} 1 & 1 & 1 \\ G(G^{-1}(\mathbf{x}_m) + \Delta\boldsymbol{\alpha}_i) \end{bmatrix}^2
 \end{aligned} \tag{3.42}$$

Il metodo implementato per determinare il valore $\boldsymbol{\alpha}_0$ è qui di seguito descritto, supponendo l'EE all'interno di una determinata area di lavoro. Il valore iniziale del vettore $\boldsymbol{\alpha}_0$ è il baricentro dell'area di lavoro considerata per l'autocalibrazione.

- si determina la griglia virtuale;
- si calcola per ciascun punto della griglia (figura 3.16) mediante cinematica inversa il valore temporaneo di $\boldsymbol{\alpha}_0$. Con tale valore si calcola la funzione costo in quel punto;
- si sceglie come stima più probabile di $\boldsymbol{\alpha}_0$, il punto in cui la funzione costo presenta valore minimo, centrando la nuova griglia virtuale proprio su questo punto.

Si reitera il procedimento fintanto che il valore della funzione costo scende al di sotto di un valore preimpostato. Praticamente si trova il vero punto in cui la macchina è stata calibrata; per passare ai valori del vettore $\boldsymbol{\alpha}_0$, basterà utilizzare la funzione di cinematica inversa. La rappresentazione grafica della funzione costo, nell'area di interesse, durante le varie iterazioni, la si può trovare in figura 3.17.

In figura 3.18 è rappresentata per ogni iterazione la distanza tra la stima della posizione di autocalibrazione e la posizione vera di autocalibrazione (simulata) in scala logaritmica. I tempi di computazione dell'algoritmo, nel controllore CX 1020, è $t_{\text{comp}} \approx 60 \text{ ms}$.

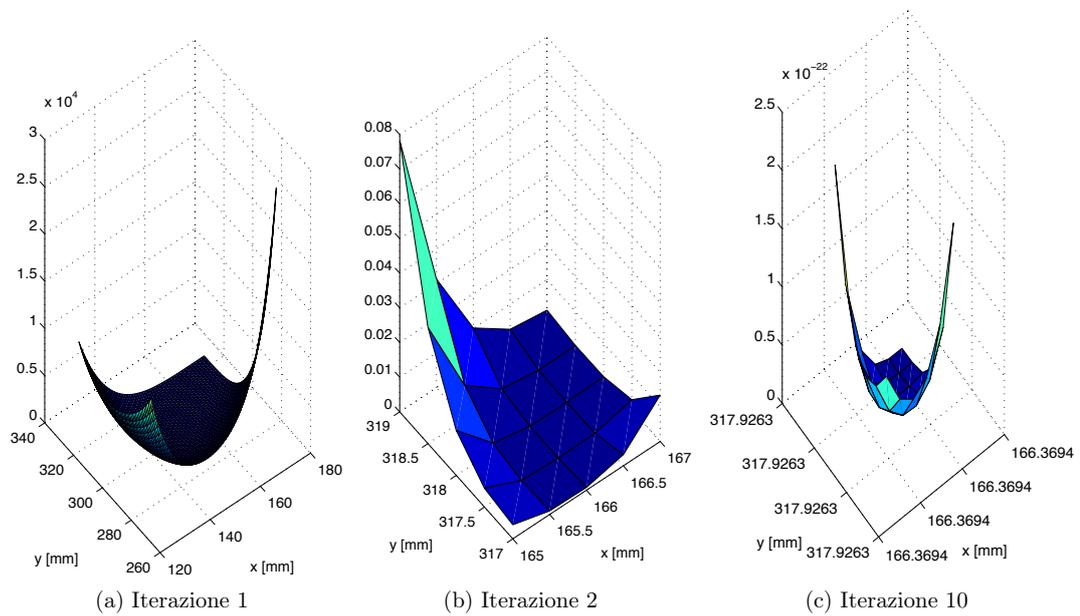


Figura 3.17: Rappresentazione del paraboloide a diverse iterazioni dell'algoritmo

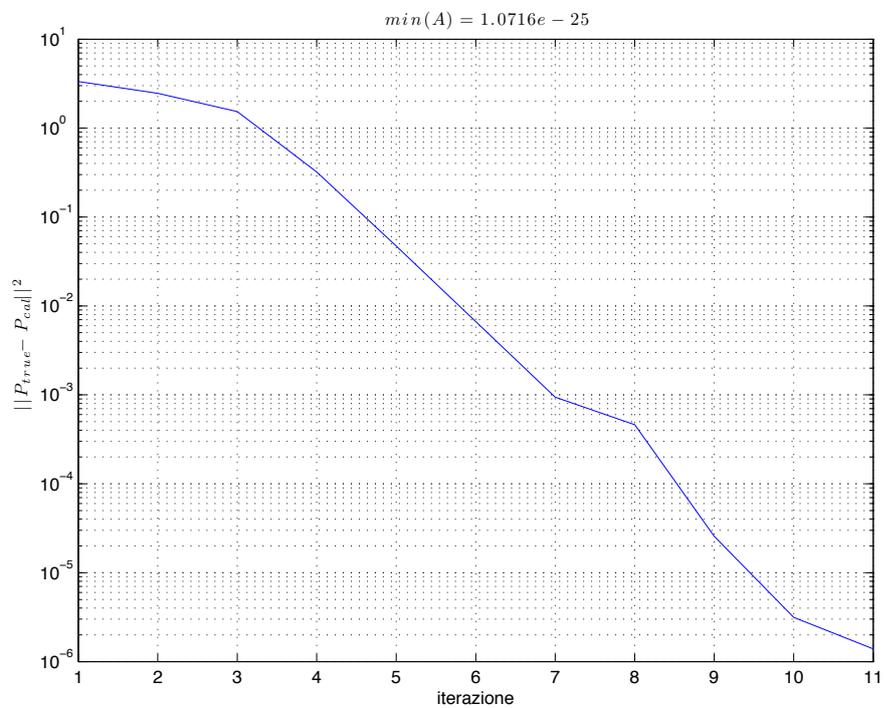


Figura 3.18: Distanza tra la stima della posizione di autocalibrazione e la posizione vera di autocalibrazione (simulata) ad ogni iterazione

3.4 Calcolo della distribuzione delle forze

In fase di terapia, sulla manopola si deve effettuare un controllo di forza, con ciò si intende che per ogni spostamento dell'End-Effector si deve produrre, in ogni condizione, una risultante che permetta il corretto proseguimento dell'esercizio. Dal momento in cui il robot dovrà essere utilizzato per la riabilitazione di pazienti post-stroke, sull'End-Effector si aspettano solamente bassi valori di velocità ed accelerazioni. Per tale motivo è possibile trascurare gli effetti dinamici ed assumere l'ipotesi di pseudo-statica. Al fine di espletare tale compito, si utilizza l'algoritmo in seguito illustrato.

Si consideri la matrice di struttura $\mathbf{B} \in \mathbf{M}_{2 \times 3}$, a rango pieno. Data una forza \mathbf{F} all'EE, si cerca un set di tensioni \mathbf{f} che soddisfa la relazione di equilibrio:

$$\mathbf{B}\mathbf{f} = \mathbf{F} \quad (3.43)$$

La soluzione ottimale è quella che rende minima la potenza dissipata dai motori:

$$P_{diss} = \sum_{j=1}^m R_{aj} \cdot i_j^2 = \sum_{j=1}^m R_{aj} \cdot \left(\frac{r_{pj} \cdot f_j}{k_{tj}} \right)^2 \quad (3.44)$$

$$= \sum_{j=1}^m \left(\frac{R_{aj} \cdot r_{pj}^2}{k_{tj}^2} \right) \cdot f_j^2 \quad (3.45)$$

Dove:

- R_{aj} resistenza di armatura del j-esimo motore.
- i_j corrente del j-esimo motore.
- r_{pj} raggio della puleggia del j-esimo motore.
- f_j tensione al cavo j-esimo.
- k_{tj} costante di coppia del motore j-esimo.

Se i motori presentano caratteristiche simili:

$$\min(P_{diss}) \iff \min(\|\mathbf{f}\|^2)$$

L'algoritmo di calcolo, che deriva dalla trattazione matematica svolta in [13] prevede il calcolo della seguente *matrice di struttura*:

$$B = \begin{bmatrix} \cos\beta_1 & \cos\beta_2 & \cos\beta_3 \\ \sin\beta_1 & \sin\beta_2 & \sin\beta_3 \end{bmatrix} \quad (3.46)$$

I valori degli angoli β sono calcolati nel seguente modo:

$$\begin{aligned} \gamma_i &= \text{atan2}((y_p - y_{m,i}), (x_p, x_{m,i})) \\ \alpha_i &= \arccos\left(\frac{rc}{\sqrt{(y_p - y_{m,i})^2 + (x_p, x_{m,i})^2}}\right) \\ \beta_i &= \gamma_i + (-1)^{i+1}\alpha_i + k_i\pi \end{aligned} \quad (3.47)$$

con $\mathbf{k} = [1/2 \ 3/2 \ 1/2]$.

La matrice \mathbf{B} può essere riscritta nel seguente modo:

$$\mathbf{B} = (\mathbf{v}_1 \ \mathbf{v}_2 \ \mathbf{v}_3) \quad (3.48)$$

dove $\mathbf{v}_i = (\cos \beta_i \ \sin \beta_i)^T$ è il vettore di lunghezza unitaria lungo l' i -esimo cavo, diretto dal centro dell'EE al punto tangente dell' i -esima puleggia. Si definisce \mathbf{B}_i la sotto-matrice ottenuta da \mathbf{B} eliminando la i -esima colonna, e preso h_i come l' i -esimo minore algebrico di ordine 3 della matrice \mathbf{B} :

$$h_i = (-1)^{i+1} \det(\mathbf{B}_i) \quad (3.49)$$

Il vettore \mathbf{h} forma una base del $\ker(\mathbf{B})$, e può essere espresso come:

$$\mathbf{h} = \begin{pmatrix} \sin(\beta_3 - \beta_2) \\ \sin(\beta_1 - \beta_3) \\ \sin(\beta_2 - \beta_1) \end{pmatrix} = \begin{pmatrix} \sin(\chi_2) \\ \sin(\chi_3) \\ \sin(\chi_1) \end{pmatrix} \quad (3.50)$$

Sia $\tilde{\mathbf{f}}_i$ una soluzione dell'equazione 3.43, dove f_i è stata arbitrariamente posta a zero, e preso \mathbf{h} un vettore strettamente positivo della base del kernel definito nell'equazione 3.50. Tutte le soluzioni possono essere riscritte nella forma:

$$\bar{\mathbf{f}} = \tilde{\mathbf{f}}_i + \beta_f \mathbf{h} \quad (3.51)$$

Se β_f , positivo, è scelto come:

$$\beta_f = \max_{t=1\dots 3} \left(\frac{f_{\min} - \tilde{\mathbf{f}}_i(t)}{\mathbf{h}(t)} \right) \quad (3.52)$$

l'equazione 3.51 fornisce la soluzione a norma minima tra quelle accettabili (ovvero $\tilde{\mathbf{f}}_i \geq f_{\min}$).

3.4.1 Scalatura delle tensioni

Può accadere che in alcune zone del piano di lavoro, per produrre la risultante desiderata sull'end-effector, vengano computate combinazioni di tensioni sui cavi che hanno uno o due valori superiori al limite consentito.

In questa particolare situazione deve avvenire la scalatura delle tensioni. Con ciò si intende che tutte le forze calcolate vengano ridotte proporzionalmente affinché tutte diventino più piccole della forza massima. Il coefficiente di scalatura k_s si calcola con la formula:

$$k_s = \min \left(\frac{f_{max}}{f_i} \right), \quad i = 1, 2, 3 \quad (3.53)$$

e i valori di forza da applicare ai cavi diventano, nel solo caso in cui k_s sia ≤ 1 :

$$\mathbf{f}_{new} = \mathbf{f}_{old} \cdot k_s \quad (3.54)$$

Con questa operazione, se la risultante da produrre è nulla, non si modifica affatto l'effetto finale sul paziente. Invece, nel caso la risultante sia in modulo maggiore di zero, questa con la scalatura viene modificata. Infatti il modulo della forza viene ridotto di un fattore pari a k_s . In ogni modo ne viene preservata la direzione e questo è sufficiente ad evitare che la macchina faccia dei movimenti inaspettati e dannosi per il paziente.

3.4.2 Compensazione della forza di gravità

Per compensare la gravità è bastato calcolare la componente della forza peso dell'end-effector parallela al piano inclinato di figura 3.19:

$$F_{parallela} = m_{EE} g \sin \delta \quad (3.55)$$

dove:

- m_{EE} è la massa dell'end-effector.
- g è l'accelerazione di gravità.
- δ è l'inclinazione del tavolo rispetto al piano orizzontale.

Il valore così ricavato (dipendente dalla lettura del potenziometro) sarà sommato alla componente F_y della forza esterna computata al fine di compensare l'effetto della forza di gravità.

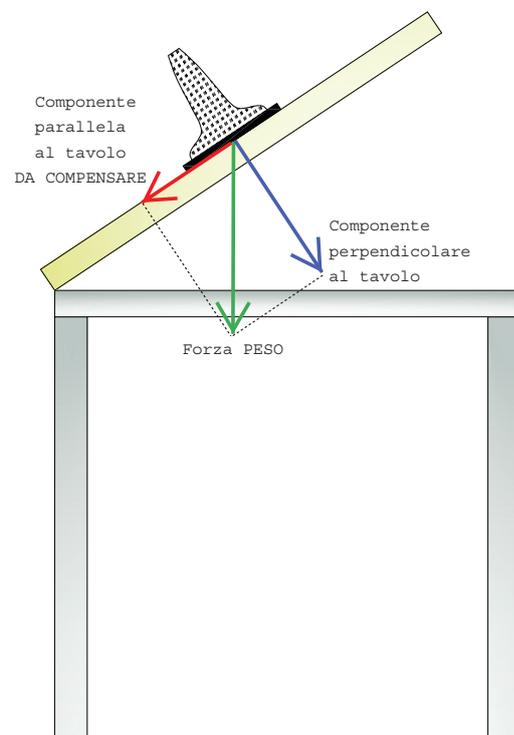


Figura 3.19: Scomposizione della forza peso

3.5 Controllo della posizione del motore mobile

Il moto traslatorio del carrello porta-puleggia, ottenuto tramite vite a ricircolo è movimentato dal motore 4. Si è implementato un controllore PID con anti-windup con derivazione dell'uscita. Il motore viene controllato in velocità, mediante riferimento di posizione. Tale scelta implementativa può essere giustificata attraverso le considerazioni in seguito riportate.

Un inconveniente della realizzazione classica [24] [25] [26], è quello che la derivazione è effettuata direttamente sull'errore e ; in questo caso, in presenza di uno scalino del segnale di riferimento r , l'uscita del derivatore, e di conseguenza la variabile di controllo u , hanno un andamento di tipo impulsivo; questa brusca variazione può provocare la saturazione dell'attuatore. Per queste ragioni l'azione derivativa è esercitata sulla sola variabile di uscita y . Poiché y è l'uscita di un sistema che usualmente ha le caratteristiche di un filtro passa basso, le sue variazioni istantanee (e quindi la sua derivata) sono in genere contenute e la presenza dell'azione derivativa non provoca l'andamento impulsivo di u .

La presenza combinata dell'azione integrale e di una saturazione dovuta all'attuatore provoca un effetto di tipo non lineare che può deteriorare significativamente le prestazioni del sistema di controllo. L'attuatore con ingresso w e uscita u , con u_M e u_m rispettivamente il massimo ed il minimo valore prima di saturare l'attuatore, è descritto dalla relazione:

$$u(t) = \begin{cases} u_M, & w(t) \geq u_M \\ w(t), & -u_m \leq w(t) \leq u_m \\ u_m, & w(t) < u_m \end{cases} \quad (3.56)$$

Si consideri, per semplicità espositiva, il caso di un sistema di controllo con il regolatore dato dalla sola parte integrale I . Quando l'errore $e = r - y$, dove r è il segnale di riferimento ed y la variabile di uscita, si mantiene dello stesso segno per un certo periodo, lo stato dell'integratore, che coincide con la sua uscita, w cresce in modulo sempre più. Ciò avviene anche se l'effettiva variabile di ingresso w viene limitata al valore u_M o u_m dalla saturazione dovuta all'attuatore. Quando questo accade, se l'errore cambia segno è necessario attendere che lo stato w dell'integratore torni ad assumere valori in modulo inferiori a u_M prima che l'attuatore riprenda ad operare in zona lineare, cioè si abbia $w(t) = u(t)$. In altri termini bisogna attendere la scarica dell'azione integrale. Il fenomeno descritto prende il nome di carica integrale o integral wind-up. Di conseguenza, i controllori che contengono un termine integrale provocano una sovraelongazione transitoria quando l'attuatore satura se non si adottano opportuni accorgimenti.

Per la realizzazione del controllore PID si faccia riferimento alla figura 3.20. Trascurando la parte derivativa, si è implementata la versione discreta dello schema 3.20. Considerando la funzione di trasferimento:

$$\frac{Z(s)}{M(s)} = \frac{1}{1 + sT_i} \quad (3.57)$$

Definendo T_s il passo di campionamento e t_k il generico istante di campionamento, utilizzando il metodo di Eulero in avanti, è stato possibile ottenere la seguente antitrasformata:

$$\frac{z(t_k) - z(t_{k-1})}{T_s} T_i = m(t_k - 1) - z(t_k - 1) \quad (3.58)$$

da cui è possibile ottenere la relazione:

$$\begin{aligned} z(t_k) &= \frac{T_s}{T_i} m(t_{k-1}) + (1 - T_s/T_i) z(t_{k-1}) \\ z(t_k) &= z(t_{k-1}) + \frac{T_s}{T_i} (m(t_{k-1}) - z(t_{k-1})) \end{aligned} \quad (3.59)$$

dove, analizzando lo schema a blocchi è possibile riscrivere il termine $z(t_{k-1})$ nel seguente modo:

$$z(t_{k-1}) = u(t_{k-1}) - q(t_{k-1}) = u(t_{k-1}) - k_p e(t_{k-1}) \quad (3.60)$$

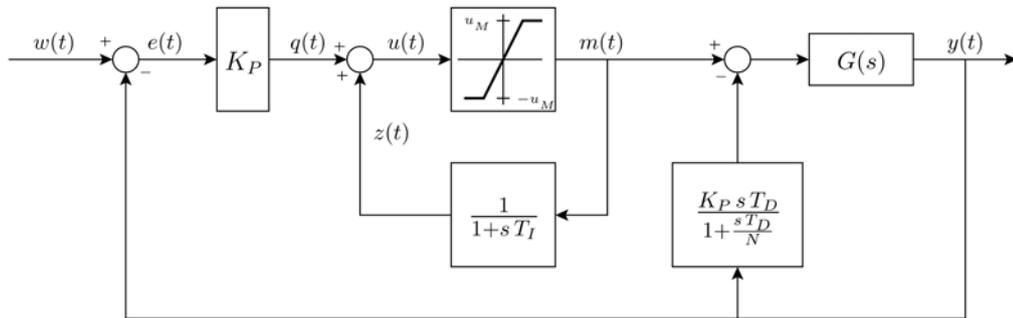


Figura 3.20: Schema generale di anti wind-up per un controllore PID

L'equazione finale sarà nella forma:

$$z(t_k) = z(t_{k-1}) + \frac{k_p}{T_i} e(t_{k-1}) T_s + \frac{m(t_{k-1}) - u(t_{k-1})}{T_i} T_s \quad (3.61)$$

in cui il primo termine rappresenta l'integrale dell'errore, mentre il secondo il termine di compensazione dell'effetto di saturazione.

Capitolo 4

Descrizione dell'Hardware

4.1 Controllore Beckhoff CX1020-0111

L'embedded PC di Beckhoff CX1020 fa parte della serie di dispositivi denominati CX che uniscono la tecnologia dei sistemi di controllo basati su PC al livello hardware dei terminali I/O per formare un'unità modulare da inserire in un qualsiasi pannello di controllo. Il CX1020 estende la famiglia dei prodotti CX con una versione con performance maggiormente elevate e permette la connessione diretta dei Bus Terminals e dei terminali EtherCAT.

Questo dispositivo è equipaggiato con un Intel Celeron M CPU a 1000 MHz, garantendo da una parte un'elevata capacità di calcolo e dall'altra un minor dispendio di energia, grazie all'utilizzo di un processore "ultra-low core voltage" (il Thermal Design Power del processore è pari a 7 W, il consumo complessivo del controllore è inferiore ai 31 W complessivi). Per tale scelta di progettazione, il controllore non dispone di una ventola di raffreddamento aggiuntiva. Altre caratteristiche sono la presenza di 256 MB di memoria RAM DDR2 ed una compact flash da 64 MB contenente il sistema operativo.

Il dispositivo è strutturato in maniera modulare, esso infatti è formato da diversi componenti che possono essere assemblati dall'utente in modo da ottenere ed utilizzare la configurazione più adatta ad ogni specifico impiego. Il CX1020 Embedded PC è stato sviluppato per supportare e rendere ottimale l'integrazione con EtherCAT consentendo però diverse opzioni di connessione con gli altri tipi

di fieldbusses. E' installato sia il sistema operativo, Microsoft Windows CE, che il programma di Beckhoff TwinCAT, che svolge sia funzioni di configurazione che funzione più avanzate di PLC e Motion Control.

Per un elenco esaustivo delle caratteristiche del dispositivo, è possibile analizzare la tabella 4.1.

Dati tecnici	CX1020
Processore	Intel® Celeron® M ULV, 1 GHz clock
Memoria flash	64 MB Compact Flash card
Memoria interna	256 MB DDR RAM (espandibile fino ad 1 GB)
Interfacce	2 x RJ 45 (Ethernet, internal switch)
LED diagnostici	1 x power, 2 x LAN, TC status, 1 x flash access
Slot di espansione	1 x Compact Flash type I+II
Clock	batteria interna RTC (battery exchangeable)
Sistema Operativo	Microsoft Windows CE
Software di controllo	TwinCAT PLC, NC PTP, NC I run-time
System bus	16 bit ISA (PC/104)
Alimentazione	system bus
Max. dissipazione termica	11 W (including CX1020-N0xx system interfaces)
Dimensioni (W x H x D)	96 mm x 112 mm x 98 mm
Peso	96 mm x 112 mm x 98 mm
Temperature di utilizzo	0...+50 °CC/-25...+85 °CC
Umidità relativa	95 %
Vibrazioni/resistenza agli urti	conforms to EN 60068-2-6/EN 60068-2-27/29
Resistenza alla rottura EMC/ESD	conforms to EN 61000-6-2/EN 61000-6-4
Classe di protezione	IP 20

Tabella 4.1: Caratteristiche tecniche del controllore Beckhoff CX1020

4.1.1 Unità di alimentazione CX1100-0004

L'unità di alimentazione CX1100-0004 è collegata al sistema tramite bus interno PC104 e dispone di caratteristiche aggiuntive che vanno oltre al semplice compito di unità di potenza. Essa infatti integra una NOVRAM che permette il salvataggio delle informazioni di processo qualora ci fosse un problema di alimentazione, un display LCD con due linee di 16 caratteri, entrambi usati per visualizzare messaggi. CX1100-0004 power supply unit è equipaggiata con un'interfaccia I/O che permette di connettere i terminali EtherCAT in modo diretto al Embedded PC. Le informazioni provenienti dagli I/O vengono mappate direttamente nella memoria principale della CX1100-0004 CPU, senza il bisogno di una DPRAM. La potenza fornita alimenta il sistema CX con un voltaggio di 24 V DC ($-15\%/+20\%$) e viene distribuita a tutti i terminali per mezzo dell'E-Bus.

Modulo CX1020-N010

Il modulo CX1020-N010 DVI / USB, alimentato e connesso al modulo di base per mezzo del bus PC104, permette di utilizzare pannelli di controllo o monitor standard DVI o VGA attraverso le uscite DVI o USB. Come per un normale PC possono essere connessi, tramite porta USB, altri dispositivi, quali ad esempio stampante, scanner, mouse e tastiera.

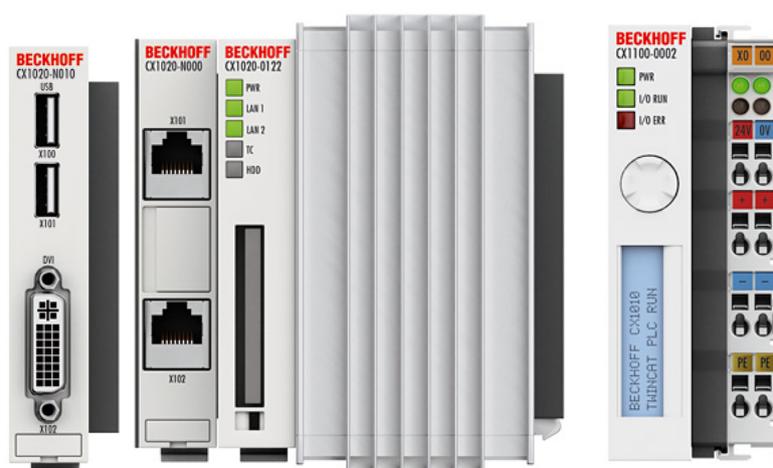


Figura 4.1: Modulo CX1020-N010 con controllore CX1020 ed unità di alimentazione CX1100-0004

4.2 Moduli EtherCAT

L'architettura hardware offerta da Beckhoff permette di realizzare sistemi di controllo modulari, assemblando il sistema con componenti specifici a seconda delle esigenze del controllo da effettuare. Nella realizzazione del prototipo Sophia3 sono stati utilizzati i moduli che in seguito verranno indicati ed analizzati, divisi per tipologie.

4.2.1 EtherCAT Coupler

EK1110

Come gli E-bus e i terminali, l'estensione EtherCAT EK1110 è connessa alla fine del blocco EtherCAT terminale. Il terminale offre la possibilità di connettere un cavo ethernet RJ 45, con ciò è possibile estendere la rete EtherCAT con un cavo elettricamente isolato fino a 100 m. All'interno del dispositivo i segnali E-bus sono convertiti al volo nella rappresentazione di segnale 100BASE-TX. L'alimentazione è fornita direttamente attraverso E-bus. Nessuna parametrizzazione o configurazione del dispositivo è richiesta.



Figura 4.2: Modulo EtherCAT Coupler EK1110

EK1100

Il terminale di capolinea EK1100 permette di connettere la rete EtherCAT agli altri terminali EtherCAT (ELxxxx). Una stazione consiste in un terminale di capolinea EK1100, un numero di terminali EtherCAT ed un terminatore di BUS. Il dispositivo converte i telegrammi provenienti dalla trasmissione Ethernet 100BASE-TX alla rappresentazione di segnale E-bus. Il capolinea è connesso alla rete attraverso l'interfaccia Ethernet superiore. Il connettore RJ 45 inferiore può essere utilizzato per collegare ulteriori dispositivi EtherCAT nella stessa linea.



Figura 4.3: Modulo EtherCAT Coupler EK1100

4.2.2 Input digitali

EL1018

Il modulo EL1018 acquisisce i segnali binari dal processo di livello e li trasmette, in forma elettricamente isolata, all'unità di automazione ad alto livello. Gli 8 input di ingresso vengono preventivamente filtrati con un filtro da $10\mu s$. L'assorbimento di corrente tipico è pari a $90mA$.

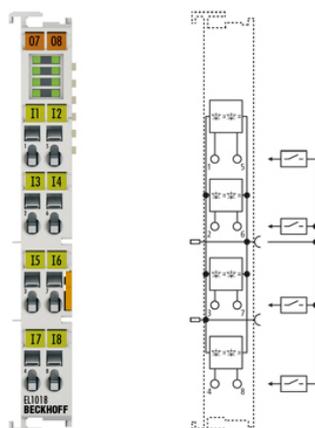


Figura 4.4: Modulo di input digitale EL1018

4.2.3 Input analogici

EL3064

Il terminale di input EL3064 permette di processare segnali elettrici nel range compreso tra 0 e 10 V. La tensione è digitalizzata con una risoluzione di 12 bit. Il dispositivo presenta 4 input analogici composti ciascuno da un collegamento a due fili, con un comune potenziale di terra comune. Il contatto a 0 V è utilizzato come riferimento di connessione a massa per gli input. Inoltre, lo stato del dispositivo è indicato mediante lampeggio di quattro LED. Altre importanti caratteristiche sono illustrate in tabella 4.2.

Dati tecnici	EL3064
Frequenza filtro di input	1 kHz
Tempo di conversione	0.625 ms (di default, configurabile)
Risoluzione	12 bit
Errore di misura	$< \pm 0.3\%$ (relativo al valore massimo)
Isolamento elettrico	500 V (E-bus/segnale)
Corrente assorbita	130 mA
Lunghezza processo immagine	16 byte

Tabella 4.2: Caratteristiche tecniche del modulo EL3064

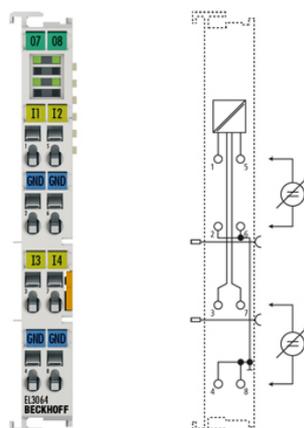


Figura 4.5: Modulo di input analogico EL3064

4.2.4 Output analogici

Il terminale di output analogico EL4134 genera segnali nel range compreso tra -10 e +10 V. Il livello di tensione è fornito dall'immagine di processo con una risoluzione pari a 16 bit ed è elettronicamente isolato. I quattro canali del dispositivo hanno il potenziale a massa comune. Gli stadi di uscita sono alimentati da una alimentazione a 24 V. Lo stato del dispositivo è indicato mediante lampeggio di quattro LED. Tale modulo permette di essere utilizzato anche in modalità clock distribuito. In tabella 4.3 vengono presentate ulteriori caratteristiche del dispositivo.

Dati tecnici	EL4134
Precisione clock distribuito	$< 1 \mu s$
Carico	$> 5 K\Omega$
Tempo conversione	$< 80 \mu s$
Errore di misura	$< \pm 0.1\%$ (relativo al valore massimo)
Isolamento elettrico	500 V (E-bus/segnale)
Corrente assorbita	265 mA

Tabella 4.3: Caratteristiche tecniche del modulo EL4134

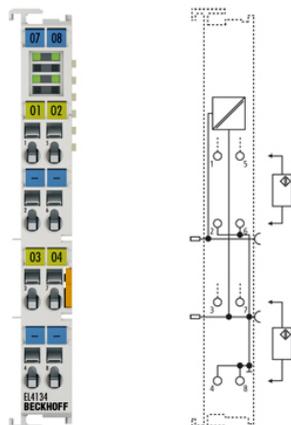


Figura 4.6: Modulo di output analogico EL4134

4.2.5 Output digitali

EL2808

Il terminale di output digitale EL2808 permette di connettere i segnali di controllo binari dal dispositivo d'automazione al processo di alto livello mediante isolamento elettrico. Il dispositivo fornisce protezione contro inversioni di polarità, corto-circuiti e sovraccarichi. La tensione corrispondente al livello logico alto è pari a 24 V. Ulteriori caratteristiche tecniche sono riportate nella tabella 4.4.

Dati tecnici	EL2808
Tipologia di carico	resistivo, induttivo
Massima corrente di corto-circuito	2A
Energia di rottura	$< 150mJ$
Tempo di commutazione (ON)	$60\mu s$
Tempo di commutazione (OFF)	$300\mu s$
Isolamento elettrico	500 V (E-bus/segnale)
Corrente assorbita	110 mA

Tabella 4.4: Caratteristiche tecniche del modulo EL2808

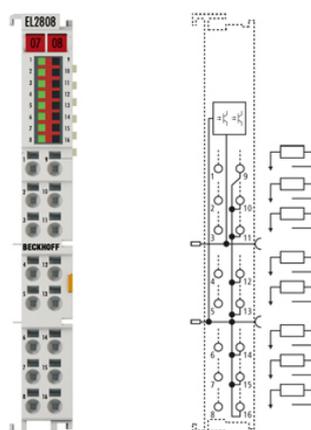


Figura 4.7: Modulo di output digitale EL2808

EL2624

Il terminale di output EL2624 consiste in 4 relè ciascuno con un singolo contatto. Il contatto del relè è adatto per tensioni fino a 125 V AC e 30 V DC. La massima corrente supportata è pari a 0,5 A. I cicli di apertura/chiusura dei contatti ammontano a 10^8 cicli operativi.

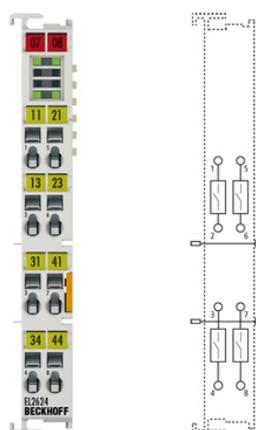


Figura 4.8: Modulo di output (relè) digitale EL2624

4.2.6 Encoder

Il terminale EtherCAT EL5101 è un'interfaccia per la diretta connessione di encoder incrementali con input differenziale (RS485). Un contatore a 32-16 bit con un decoder in quadratura ed un latch a a 32-16 bit per l'impulso di zero

possono essere letti, settati ed abilitati. Il collegamento con encoder con un output d'allarme è possibile. L'intervallo temporale di lettura può avere una risoluzione fino a $100ns$. Con la funzione opzionale di interpolazione dei microincrementi, il dispositivo può fornire una misura più accurata. La lettura sincrona è supportata, mediante l'utilizzo del clock-distribuito.

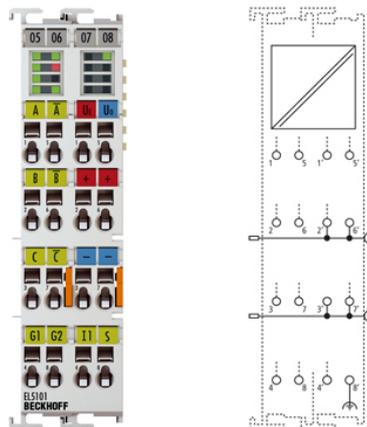


Figura 4.9: Modulo encoder incrementale EL5101

4.2.7 Collegamenti ingressi-uscite ai dispositivi EtherCAT

In tabella 4.5 è presente lo schema riassuntivo dei collegamenti di input ed output dei moduli EtherCAT verso le schede di condizionamento del segnale, al circuito di emergenza ed ai driver.

In figura 4.10 è presente l'immagine dei collegamenti effettuati dalla stazione moduli agli input ed output esterni.

4.3 Controllo del motore in corrente continua: tecnica PWM

Il controllo del quadrilatero era inizialmente demandato all'applicazione di una tensione costante $U_{in} = 12V$ per mezzo di un selettore manuale. Il selettore permetteva di controllare la direzione del moto alternando l'alimentazione di due relè (K2 e K4 in figura 4.12) disposti in modo da formare un ponte ad H.

4.3. CONTROLLO DEL MOTORE IN CORRENTE CONTINUA: TECNICA PWM65

EL3064	Ch.1	Potenziometro tavolo
EL4134	Ch.1	Riferimento coppia motore 1
	Ch.2	Riferimento coppia motore 2
	Ch.3	Riferimento coppia motore 3
	Ch.4	Riferimento velocità motore 4
EL1018	Ch.1	Stato EMERGENZA
	Ch.2	Sensore induttivo 1
	Ch.3	Sensore induttivo 2
	Ch.4	Input MARCIA
EL2808	Ch.1	EMERGENZA Controllore (Watchdog)
	Ch.2	Enable Driver
EL2426	Ch.1	Freno motore 1
	Ch.2	Freno motore 2
	Ch.3	Freno motore 3
	Ch.4	Freno motore vite 4
EL5101	Ch.1-8	Emulazione Encoder Driver 1
EL5101	Ch.1-8	Emulazione Encoder Driver 2
EL5101	Ch.1-8	Emulazione Encoder Driver 3
EL5101	Ch.1-8	Emulazione Encoder Driver 4

Tabella 4.5: Collegamenti di input ed output dai moduli EtherCAT all'hardware esterno

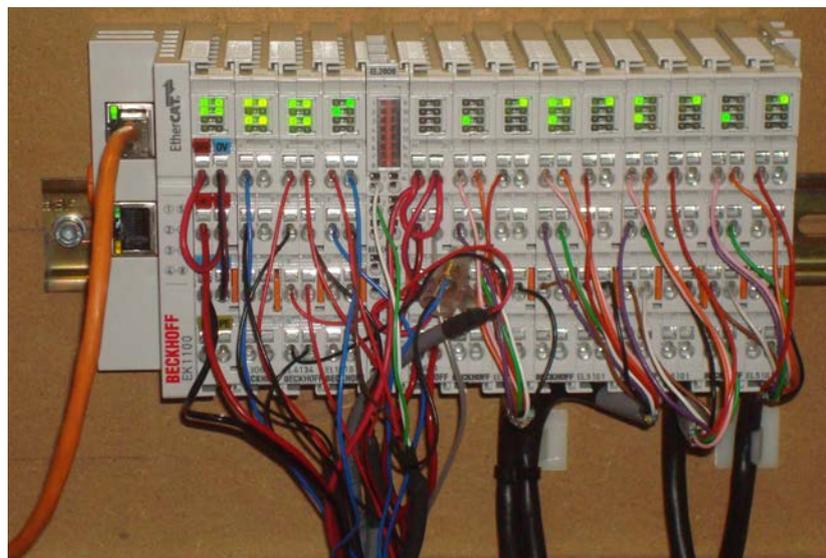


Figura 4.10: Collegamento dei vari moduli EtherCAT all'hardware esterno

Inoltre, due micro interruttori prevengono il funzionamento del motore oltre i limiti consentiti (per esempio 0 e 60 gradi). I primi test sperimentali, hanno evidenziato come tale tipologia di controllo fosse non appropriata, causando forti vibrazioni all'inizio e alla fine di ciascun movimento. Al fine di ridurre questo effetto indesiderato è stato sviluppato un custom-made driver per poter fornire un profilo di tensione trapezoidale per ciascun movimento (il tempo di accelerazione/decelerazione è stato fissato pari a $T_a = T_d = 1024ms$). Per quanto concerne la parte hardware, il ponte ad H consiste di 4 MOSFET di potenza controllati da un microcontrollore (dsPIC30F3011) attraverso quattro transistor BJT. Il dsPIC gestisce gli input digitali provenienti dal selettore manuale e dai finecorsa. Tutti i componenti sono montati in un PCB (figura 4.11). La tensione di alimentazione è fornita alla scheda mediante il connettore di potenza (L186200, label U3 in figura 4.12), mentre la tensione che alimenta il motore è fornita da terminali simili localizzati nel lato opposto della scheda (label U4). Un fusibile ($I_n = 25A$, label U38) protegge il dispositivo da potenziali sovraccarichi ed una serie di 5 condensatori in parallelo ($C_{tot} = 13,5\mu F$, label U19-U23 e U37) sono interposti tra la fonte di alimentazione ed il ponte ad H. Essi operano come *over-voltage snubber*; in presenza di un carico induttivo, l'interruzione improvvisa della corrente porta ad un brusco aumento della tensione attraverso il dispositivo

4.3. CONTROLLO DEL MOTORE IN CORRENTE CONTINUA: TECNICA PWM⁶⁷

che genera l'interruzione. Tale picco potrebbe comportare un guasto transitorio o permanente del dispositivo.

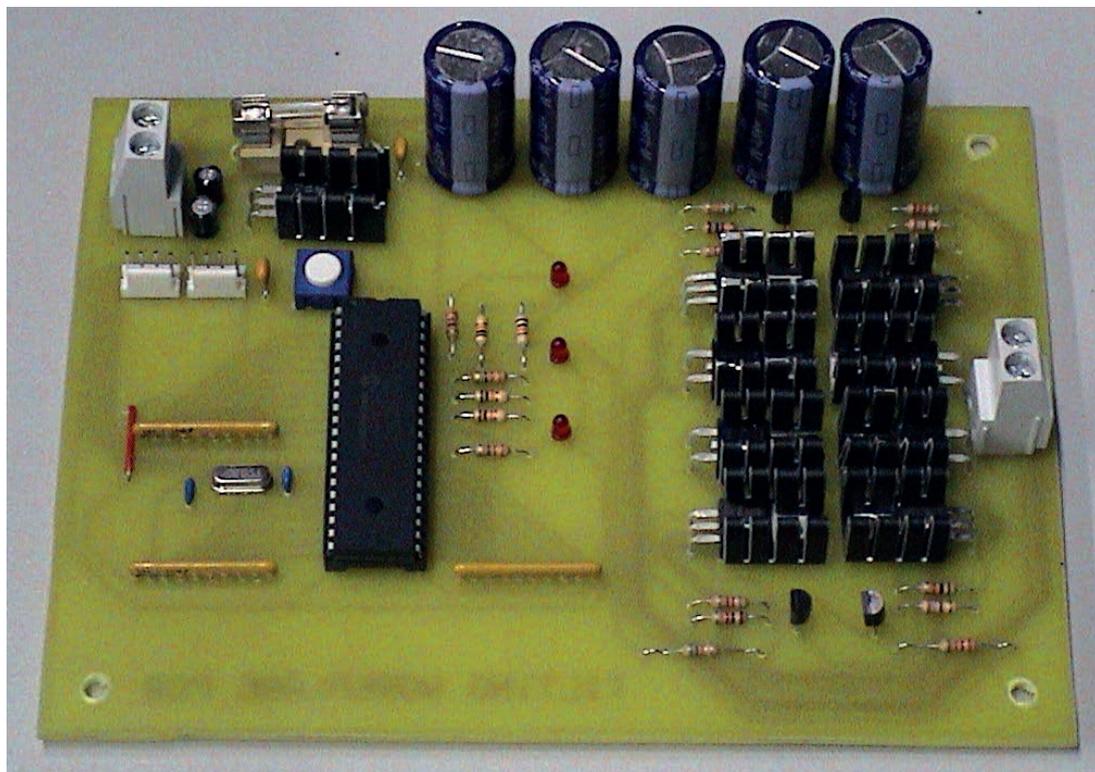


Figura 4.11: PCB del driver utilizzato per il controllo in PWM del motore DC

La corrente nominale per il motore a corrente continua EC100.120 è $I_n = 16,8 A$ e la tensione nominale di alimentazione è $U_n = 12 V$. Naturalmente, il microcontrollore non può gestire tali potenze direttamente. Infatti, gli input digitali (usati come periferica PWM) sono in grado di supportare al massimo $i_{\max} = 25 mA$ a $5 V$. Ciascun interruttore elettronico del ponte ad H è costituito da un MOSFET in parallelo ad un diodo. Gli interruttori sono disposti in modo simmetrico e la corrente circola da un ramo superiore al ramo inferiore opposto (CTRL_A CTRL_B). La funzione dei diodi (FFPF30U60STU, labels U24-U27) è quella di proteggere il corrispondente MOSFET dalle tensioni di picco potenzialmente generate dal motore.

Ciascun interruttore del ramo superiore è formato da un MOSFET-p (STP80PF55, label U16 e U17), che hanno la caratteristica di supportare correnti massime pari a $I_{\max} = 80 A$. Quando una differenza di potenziale negativa $V_{GS} \leq -3V$ è appli-

cata tra il gate ed il source, il MOSFET permette alla corrente di circolare tra il source ed il drain. Quando la tensione V_{GS} è prossima alla tensione nominale di -10 V , il dispositivo si comporta come un interruttore chiuso ideale. Al contrario, quando V_{GS} è circa 0 V , il MOSFET si comporta come un interruttore aperto. Nel ramo inferiore i MOSFET sono di tipo n (STP75NF75, label U14 e U18), che presentano caratteristiche opposte rispetto ai MOSFET di tipologia p. In conclusione, i MOSFET possono essere comandati dalla differenza di potenziale V_{GS} .

La caduta di tensione richiesta dai MOSFET è fornita dai BJT, i quali, a turno, sono controllati dagli output digitali del dsPIC. Nello specifico, gli interruttori superiori sono costituiti da transistori di tipo npn (U30 e U31), mentre quelli inferiori da transistor pnp (U28 e U29). I transistor possono funzionare come degli interruttori, controllando la corrente di base i_b . Nel ramo superiore, i BJT di tipo npn, sono dei BC33725TA. Quando il corrispondente segnale digitale è a livello logico basso, la base e l'emettitore sono allo stesso potenziale; il dispositivo si comporta come un interruttore aperto. Al contrario, quando una specifica tensione positiva è applicata tra la base e l'emettitore, la corrente di collettore i_C che fluisce tra il collettore e l'emettitore è proporzionale alla corrente di base i_B , la correlazione è espressa dal coefficiente $f_{FE} \simeq 100$ ($i_C = h_{FE} i_B$). Lo stesso concetto è applicabile al BJT di tipo pnp (BC33725TA) presente nel ramo inferiore, ad eccezione per la differente tensione tra base ed emettitore che deve essere negativa (oltretutto la direzione di i_B è uscente dalla base). Quando la corrispondente uscita digitale (CTRL_B) è attiva, la corrente scorre attraverso la resistenza R3 e R6, perciò viene fornita la caduta di tensione richiesta per attivare i MOSFET U16 e U18. I resistori sono stati dimensionati per garantire adatte correnti e cadute di tensione.

La logica implementata all'interno del microcontrollore è discussa nella sezione 5.5. Il dsPIC gestisce 4 ingressi digitali. La porta U1 è utilizzata per ricevere i segnali dai micro interruttori posti a segnalare le condizioni limite di moto del piano, la porta U2 per i segnali del selettore. Tutti i segnali lavorano in logica negata. Sono presenti, inoltre, un oscillatore al quarzo (HC49SLF) per il segnale di clock del microcontrollore, un regolatore di tensione (LM1085IT5) per convertire

la tensione dai 12V ai 5V necessari. In caso di raggiungimento di una condizione fault (descritte nella sezione 5.5) , la presenza di un microinterruttore (3CTL9) permette di resettare il dispositivo.

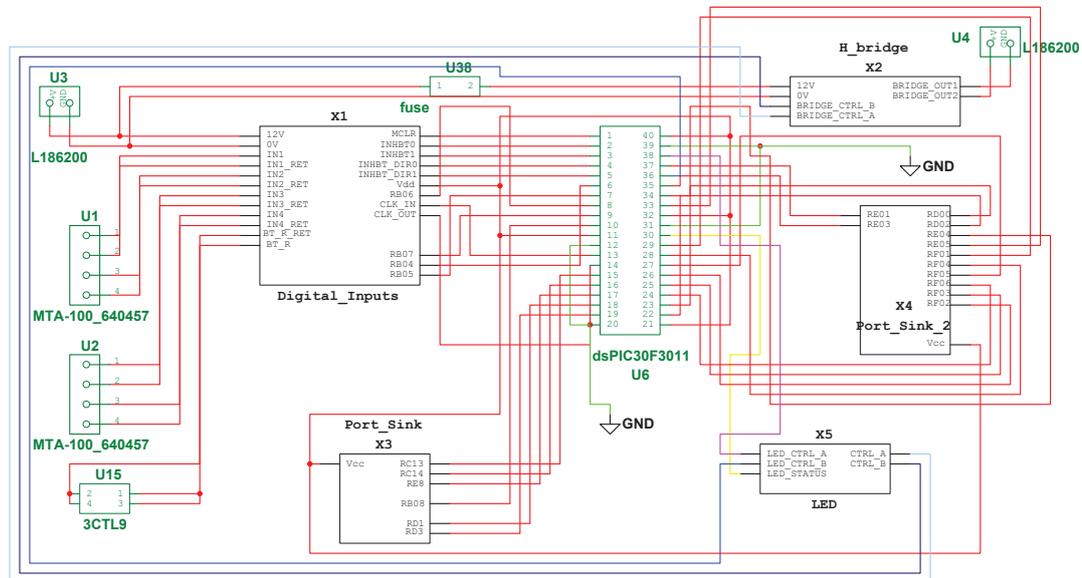


Figura 4.12: Schema elettrico del driver per il controllo in PWM del motore a spazzole

4.4 Gestione input/output controllore

Per poter interfacciare gli input e gli output dei moduli EtherCAT al prototipo sono state realizzate due schede elettroniche, una per la gestione del circuito di emergenza, dell'enable driver e per l'input di marcia; la seconda per la gestione degli induttivi e del potenziometro relativo alla lettura dell'inclinazione del tavolo. Gli schemi elettrici sono raffigurati in figura 4.14 e 4.15.

4.4.1 Layout d'insieme

Il controllore Beckhoff CX 1020 si trova all'interno del quadro elettrico generale. Attraverso un cavo Ethernet il controllore è collegato alla stazione dei dispositivi EtherCAT posta sotto al tavolo. I driver Elmo sono interposti tra i moduli

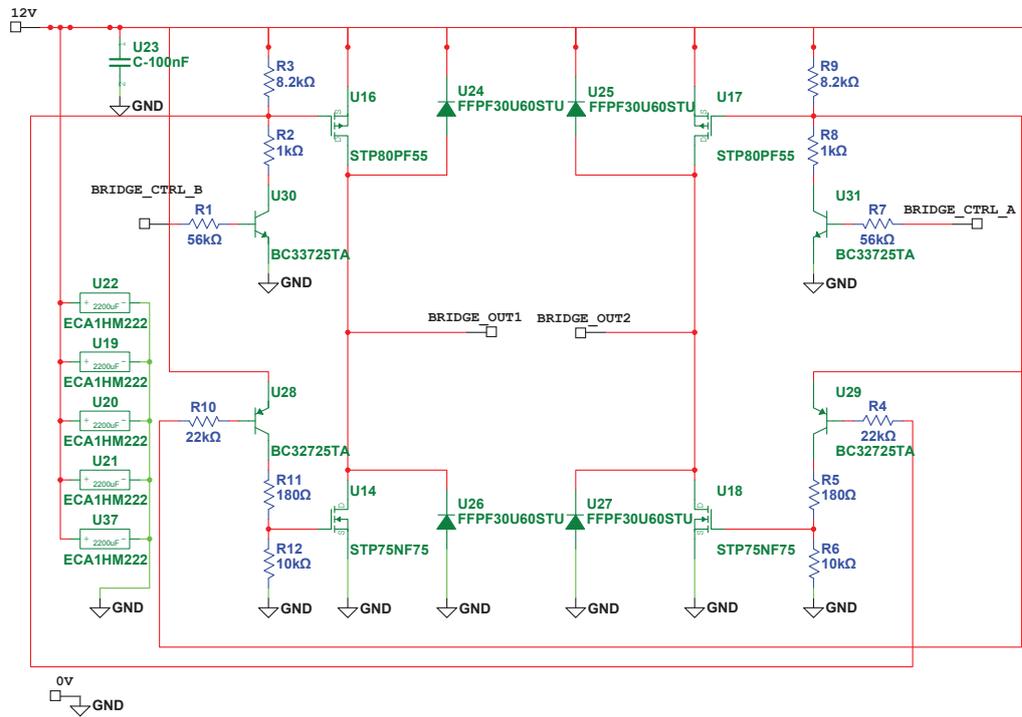


Figura 4.13: Schema elettrico del bridge utilizzato per pilotare il motore

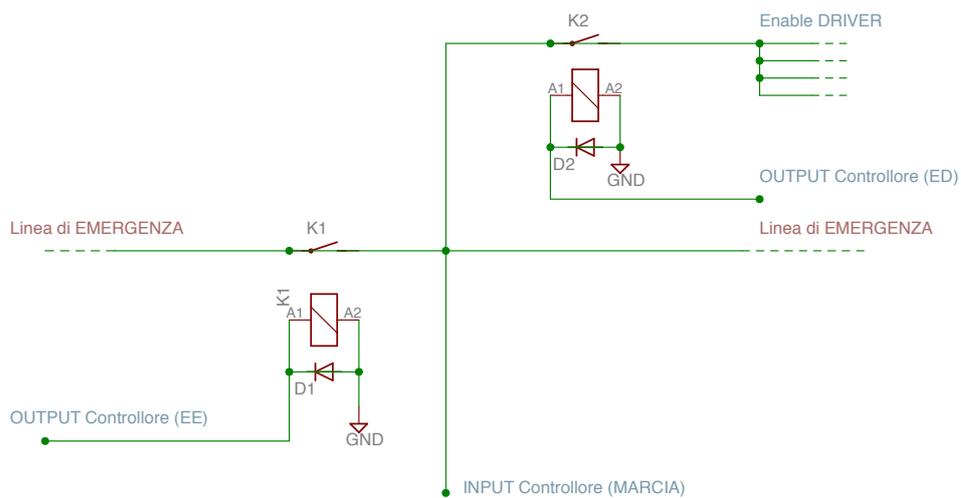


Figura 4.14: Schema elettrico della scheda di gestione del circuito di emergenza, watchdog ed enable driver

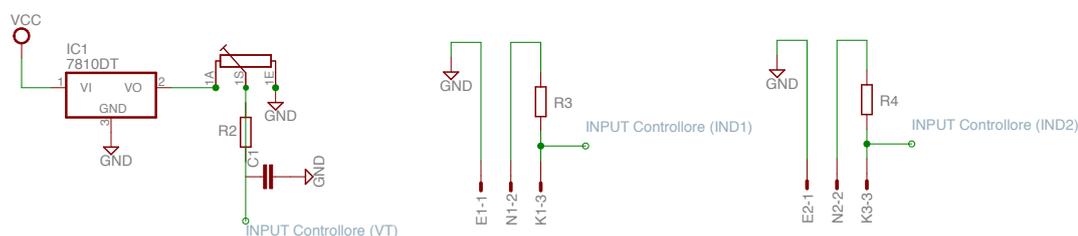


Figura 4.15: Schema elettrico della scheda per la lettura del potenziometro e dei sensori induttivi

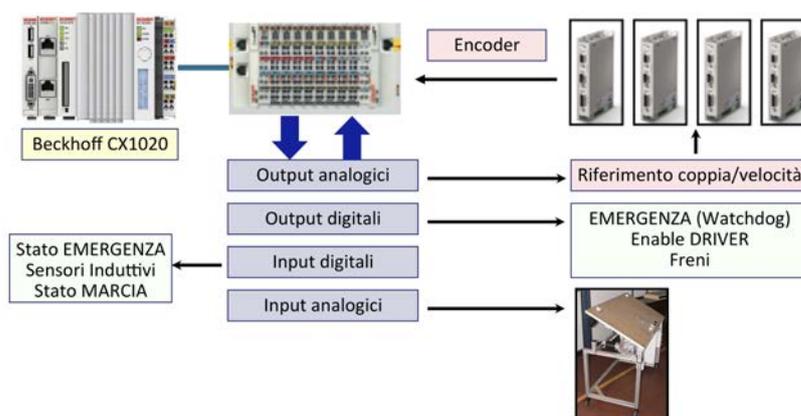


Figura 4.16: Layout d'insieme del prototipo

EtherCAT ed i motori (figura 4.16). In figura 4.17 è possibile osservare la componentistica installata all'interno del quadro elettrico, mentre in figura 4.18 sono presenti i dispositivi montati nel retro del tavolo.

4.4.2 Sensori induttivi

Sono sensori di tipo NPN normalmente aperti (NO). I tre fili che fuoriescono dal sensore hanno tre diversi colori: blu, nero, marrone. Il filo blu si dovrà connettere all'alimentazione, il filo nero di segnale indicherà la presenza o meno di materiale ferromagnetico nello spazio utile di intervento, il filo marrone, invece, si connetterà a massa. Il carico si dovrà connettere tra il filo blu ed il filo nero, ovvero tra alimentazione e segnale. Con sensore non attivo (materiale ferromagnetico non rilevato) l'interruttore è aperto, il filo nero risulta essere alla stessa tensione dell'alimentazione e non c'è caduta di tensione sul carico. Con sensore

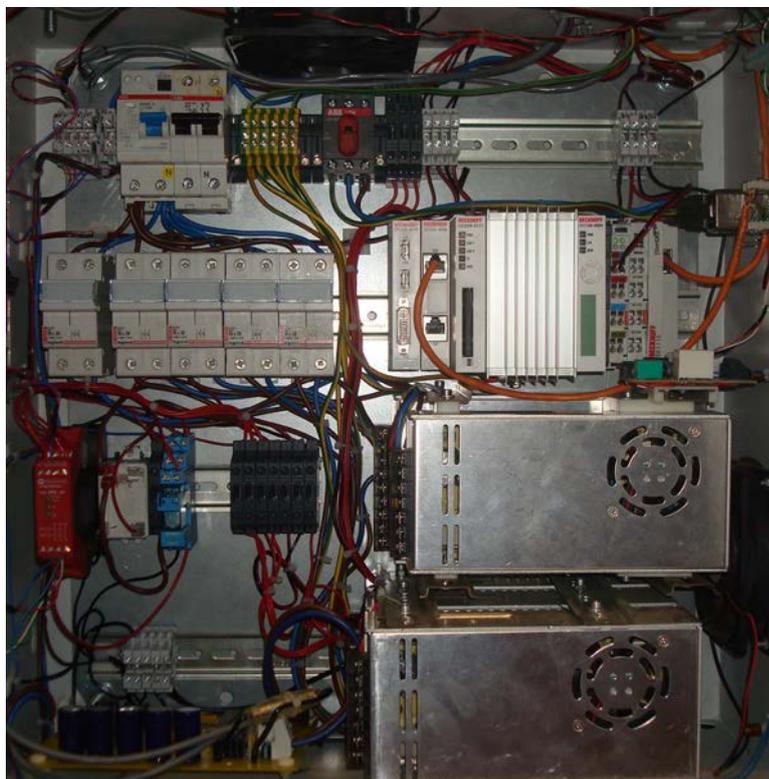


Figura 4.17: Quadro elettrico generale

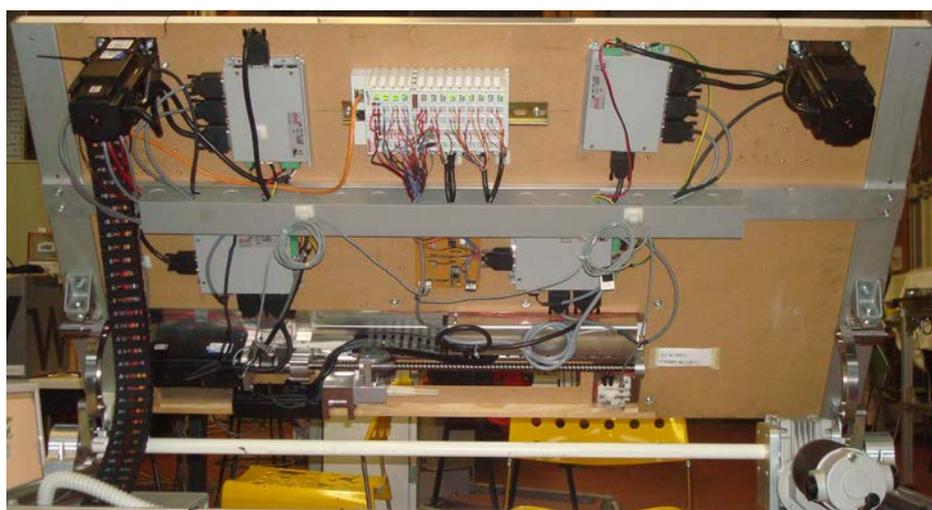
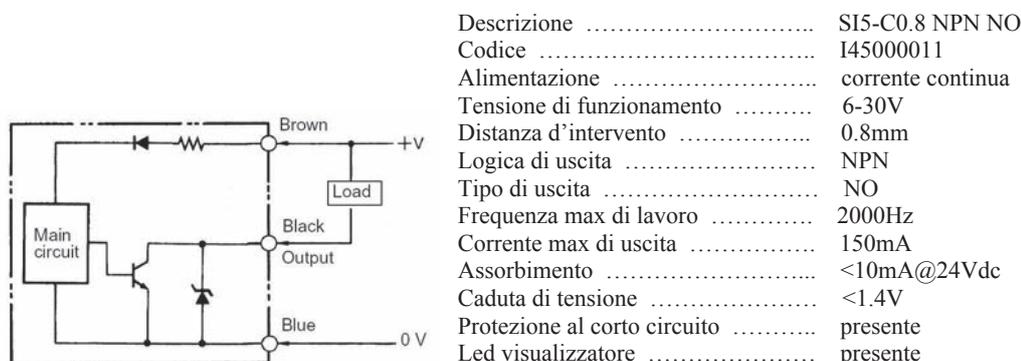


Figura 4.18: Vista del retro del tavolo

attivo (materiale ferromagnetico rilevato), l'interruttore è chiuso, il filo nero è cortocircuitato a massa e sul carico è presente una caduta di tensione.



Come anticipato in precedenza, il segnale degli induttivi deve essere sfruttato per realizzare l'homing, ossia l'azzeramento della lettura encoder del motore della vite in corrispondenza della corsa minima dell'asse lineare.

Per il calcolo della resistenza da utilizzare (R3 e R4 in figura 4.15), si è utilizzata la seguente formula:

$$R = \frac{V_{cc} - V_{cdt}}{I_{input}} \simeq 753\Omega \quad (4.1)$$

in cui la tensione di alimentazione degli induttivi è indicata con $V_{cc} = 24 V$, la corrente richiesta dal modulo EL1018 è $I_{input} = 30 mA$ e la caduta di tensione del sensore induttivo è pari a $V_{cdt} = 1,4 V$. Il valore della resistenza commerciale scelto è pari a 860Ω .

Capitolo 5

Descrizione del Software

5.1 Windows CE

Windows Embedded CE è un sistema operativo hard-real-time (il sistema garantisce la fattibilità di schedulazione entro una scadenza temporale prefissata), modulare, a 32 bit, con un kernel unificato e potenti strumenti di sviluppo embedded, progettato per dispositivi intelligenti, connessi e orientati ai servizi: è utilizzabile da sistemi GPS portatili a controller industriali real-time [22]. Per la sua modularità e flessibilità, è stato sviluppato in specifiche versioni per dispositivi differenti, oltre che per processori differenti (x86, MIPS, ARM, Hitachi SuperH, Intel XScale). Supporta vari sottoinsiemi delle *Microsoft Win32 API*, oltre che a diverse interfacce di programmazione. Il sistema API (Application Program Interface) è disponibile per tutte le applicazioni attraverso la libreria *coredll.dll*, che è collegata a tutti i moduli eseguibili del sistema operativo.

Il kernel di Windows Embedded CE 6.0 interagisce con l'hardware mediante *OAL* (OEM Adaption Layer), uno strato di astrazione dall'hardware, che nasconde allo sviluppatore l'implementazione e l'inizializzazione del hardware stesso; tale tecnica permette di migliorare la portabilità del codice su dispositivi diversi. Può utilizzare fino a 2 GB di memoria virtuale. Il nucleo del sistema operativo è eseguito nel processo *Nk.exe*, nel quale sono presenti componenti core, come il file-system, ed anche tutte le librerie dinamiche, responsabili di numerose funzionalità del sistema; tutti i device driver sono caricati all'interno del kernel. I vantaggi di

questa scelta architetturale sono il minor overhead per le chiamate di sistema e la maggior condivisione di codice tra i vari server. Per contro vi possono essere potenziali problemi di sicurezza e robustezza, in quanto i driver in kernel-mode hanno accesso a tutte le risorse del sistema.

La versione 6.0 di Windows CE presenta le seguenti caratteristiche, rispetto alle versioni precedenti:

- spazio degli indirizzi per i processi è stato incrementato da 32 MB a 2GB;
- il numero dei processi eseguibili è stato incrementato da 32 a 32000;
- i device-driver possono essere eseguiti sia in user-mode che in kernel-mode;
- i processi Device.exe, filesys.exe, GWES.exe sono stati spostati a livello di kernel;
- prestazioni delle system-call notevolmente migliorate.

5.1.1 Tecnologie e supporti

Windows Embedded CE 6.0 offre allo sviluppatore una vasta gamma di opportunità, supporti e di tecnologie. In particolare garantisce sistemi rapidi per lo sviluppo di applicazioni tipo [22] :

- AYGShell API, che assicura la compatibilità con le applicazioni Windows Mobile.
- .NET Compact Framework 2.0 e 3.5, tra cui Active Template Library (ATL), Microsoft Foundation Classes (MFC), Windows Template Library (WTL), Standard Template Library (STL), ActiveSync, Exchange Server Client , Global Positioning System (GPS) driver, Speech API 5.0, Windows Messenger, Pocket Outlook Object Model (POOM), Extensible Markup Language (XML) e Microsoft SQL Server Compact 3.5.
- Simple Network Management Protocol (SNMP).
- Production Quality OAL (PQOAL), un set di librerie e il codice sorgente per la creazione dell'OAL.

Inoltre, dispone delle tecnologie di comunicazione:

- Transmission Control Protocol/Internet Protocol (TCP/IP), IPv4, IPv6, Network Driver Interface Specification (NDIS) 5.1, Winsock 2.2, Internet Protocol security (IPsec) v4.
- Personal area network (PAN), local area network (LAN), wide area network (WAN), Bluetooth, 802.11.
- SOAP, OBject EXchange (OBEX), Lightweight Directory Access Protocol (LDAP) client, Remote Desktop Protocol (RDP).
- VoIP, real-time communications (RTC), Session Initiation Protocol (SIP).
- Radio Interface Layer (RIL), support for Short Message Service (SMS), Wireless Application Protocol (WAP), support for Subscriber Identity Module (SIM) cards.
- Remote API (RAPI) and RAPI2, Point-to-Point Protocol over Ethernet (PPPoE), Telephony Application Programming Interface (TAPI), virtual private network (VPN).

Fornisce le tecnologie Server-side [\[22\]](#) :

- Telnet, File Transfer Protocol (FTP), server message block (SMB), Common Internet File System (CIFS), Microsoft Message Queuing (MSMQ), Remote Access Service (RAS), Point-to-Point Tunneling Protocol (PPTP), Universal Plug and Play (UPnP).
- Server Web con supporto per Active Server Pages (ASP).

Offre diverse applicazioni multimediali come [\[22\]](#) :

- DirectDraw, DirectShow, Direct3D.
- Windows Media Player, Windows Media Audio (WMA), MP3.
- Internet Explorer.
- interfaccia DVD.

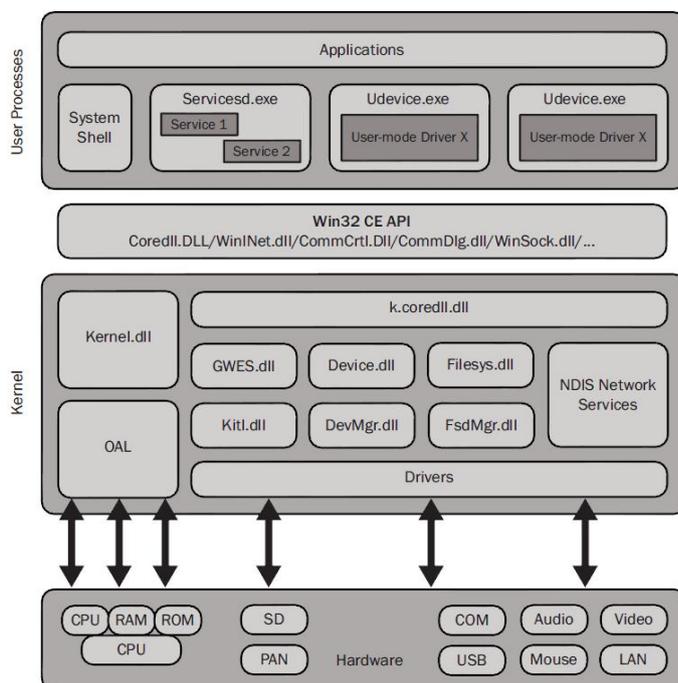


Figura 5.1: Architettura del sistema operativo Windows CE

5.1.2 Caratteristiche di sicurezza e robustezza

Al fine di aumentare la sicurezza e la robustezza del sistema, caratteristiche molto importanti per un sistema mission-critical, il kernel presenta le seguenti caratteristiche [23]:

- convalida dei parametri nelle chiamate di sistema;
- page table e handle table privati in ogni processo;
- kernel stack per le chiamate di sistema;
- heap maggiormente sicuri, con la separazione delle strutture di controllo dei dati;
- strutture dati read-only per i client;
- adozione di un modello one-tier: presenza di moduli trusted e moduli non trusted, i quali non possono essere eseguiti;

- restrizione nell'accesso alla memoria fisica ai soli componenti kernel; quelli user richiedono l'utilizzo di un proxy;
- possibilità di utilizzo di una firma digitale e di tecniche crittografiche per file eseguibili e dll;
- impossibilità di eseguire codice per UI all'interno del processo kernel.

5.2 TwinCAT System Manager

TwinCAT System Manager è un'applicazione fondamentale di TwinCAT per la configurazione del programma. Le informazioni derivanti dai dispositivi collegati al controllore vengono organizzate e collegate ai segnali di ingresso e di uscita dei task del programma di controllo, mappando gli indirizzi delle variabili in un'immagine di processo. Questa applicazione consente la comunicazione "task to task" in modo che "in and outputs" di un task possono essere modificati ciclicamente con "out and inputs" di un altro task conservando in modo ottimale i dati scambiati.

Le variabili possono essere individuali (bits, bytes, 16 bit data words, 32 bit data words etc.) oppure strutturate in arrays di dati. La più piccola unità che può essere indirizzata e collegata è una variabile booleana singola. TwinCAT System Manager produce assegnazioni individuali che contengono i collegamenti creati, in particolare tra task del programma di controllo e dispositivi sul fieldbus e tra vari task se hanno variabili collegate tra loro in comune. L'utilizzo di questa applicazione ci consente di gestire facilmente le variabili del programma di controllo senza conoscere il tipo di struttura e le proprietà di ogni fieldbus usato.

TwinCAT System Manager supporta tutti gli standard commerciali di fieldbus e anche altri standard di interfacce PC : Beckhoff Lightbus; Profibus DP, MC (DP-V2); Interbus; CANopen; SERCOS; DeviceNet; Ethernet; EtherCAT; Beckhoff Real-Time; Porta di stampa Ethernet PC (8 inputs e 8 outputs basati su TTL); Serial Bus Coupler BK8100 to COM; USB Bus Coupler BK9500 e USB Control Panel interface (CPx8xx); Dual-ported memory interface (DPRAM) per cards PC NOVRAM (Non-Volatile RAM) System Management Bus (SMB) per il monitoraggio del hardware PC come ventole, temperatura, e altro [18].

Per accedere a TwinCAT System Manager basta entrare nel TwinCAT menù e selezionare “System Manager”. La finestra TwinCAT System Manager viene mostrata in figura 5.2 . Nella barra principale viene scritto il nome del progetto e il nome del dispositivo collegato, mentre nelle barre sottostanti è presente il menù con le varie toolbar. Nello spazio centrale è presente una struttura ad albero che contiene i principali componenti di configurazione. Di particolare interesse sono :

- “Configurazione SYSTEM”
- “Configurazione I/O”.

Nella barra in fondo alla finestra viene indicato il nome del dispositivo collegato e lo stato attuale di TwinCAT:

- **TwinCAT stopped** con barra di colore rosso.
- **TwinCAT started** con barra di colore giallo. Il sotto programma TwinCAT I/O viene parametrizzato secondo la configurazione scelta.
- **TwinCAT running** con barra di colore verde ed indicando la percentuale di CPU utilizzata. In questa modalità è possibile verificare lo stato in linea delle variabili forzando il valore delle stesse.
- **TwinCAT is in Config mode** con barra di colore blu. In questa modalità è possibile configurare le variabili e collegarle ai vari dispositivi.

System Configuration Rappresenta l'applicazione di configurazione principale di TwinCAT System Manager la quale consente in primo luogo di creare un collegamento tra il pc remoto e l'embedded pc utilizzato per il controllo. Nel Pannello principale, nel quale compare la versione del programma, premendo il tasto “choose Target...” si accede ad una finestra che permette di ricercare la presenza di reti ethernet o di eventuali fildbus. Cliccando il tasto “Search (Ethernet)” compare la finestra “Add Route Dialog” che permette di ricercare i dispositivi collegati a reti ethernet. Premendo su “Broadcast Search” comparirà una lista nella finestra centrale e per aggiungere il proprio dispositivo alla rete TwinCAT, basterà selezionarlo nell'elenco e premere “Add Route”. In questo modo

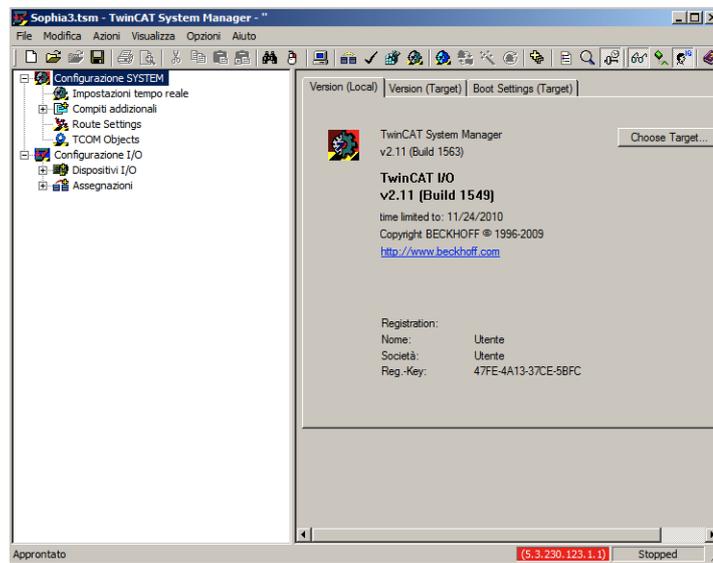


Figura 5.2: Finestra TwinCAT System Manager

il programma entra in comunicazione con il dispositivo e consente di impostare le configurazioni di sistema che sono suddivise in:

- **Impostazioni tempo reale** : consente di configurare il “Base Time” e la percentuale limite di utilizzo della CPU da parte del “real-time” di TwinCAT.
- **Compiti Addizionali** : in questa sezione si possono aggiungere i tasks addizionali che permettono, quando non viene utilizzata la funzione PLC, di creare variabili di ingresso e di uscita per creare delle applicazioni con diversi tipi di programmi (ad esempio Visual Studio 2008). Le variabili create vengono collegate agli ingressi e alle uscite dei terminali presenti nel dispositivo, cliccando con il tasto destro del mouse e selezionando il comando “modifica collegamento...”. Nella finestra “Task” si possono configurare il numero della porta AMS, il nome del compito addizionale, la priorità e il tempo di ciclo. In questo modo si va a configurare l’immagine di processo del compito addizionale creato ed è possibile esportare tale configurazione, cliccando con il tasto destro del mouse sul task creato e selezionando il comando “Export Header File”. Il File così ottenuto contiene il numero di porta ADS del task, la dimensione delle immagini di processo degli input e

degli output (in bytes) e le variabili dell'immagine di processo che vengono rappresentate mediante una struttura.

- **Route Setting** : consente le configurazioni dei routers e delle comunicazioni con i dispositivi trovati.
- **TCOM Objects** : descrive gli oggetti e le operazione utilizzate.

I/O configuration Rappresenta il livello base di TwinCAT System Manager ed è sempre presente nello schema ad albero perchè non necessita di una licenza specifica (come per altro accade per le configurazioni di sistema). Essa è suddivisa in due diverse sezioni:

- **Dispositivi I/O** : vengono configurate gli ingressi e le uscite dei dispositivi collegati al controllore. Per aggiungere un dispositivo si può procedere con una scansione automatica , cliccando con il tasto destro del mouse e selezionando “cerca dispositivo”. Una volta terminata la scansione, vengono visualizzati dal programma i dispositivi trovati e gli eventuali terminali collegati. Le variabili di ingresso e di uscita dei vari terminali possono essere collegate alle variabili dei compiti addizionali ma necessitano dell'assegnazione ad una “Sync Units” (modulo che definisce un insieme di informazioni di processo che possono essere scambiati in modo sincrono tra il master e uno o più dispositivi slave EtherCAT) che va poi a collegarsi con il tempo di ciclo del compito addizionale.
- **Assegnazioni** : una volta creata e configurata l'immagine di processo vengono generate le assegnazioni che mappa gli indirizzi delle variabili dei vari dispositivi con le variabili dei tasks addizionali. Per creare le assegnazioni cliccare con il tasto destro su “Assegnazioni” e selezionare il comando “genera assegnazioni”.

5.2.1 TwinCat I/O

TwinCat I/O è un applicazione di TwinCat che include un sistema “real-time” per gestire il funzionamento del fieldbus e un interfaccia DLL per gestire i programmi

e le applicazioni. TwinCat I/O si comporta da “real-time” driver per programmi Windows (nel nostro caso che sono in esecuzione con Windows CE), creando le relazioni corrette tra le variabili del programma, i dispositivi presenti nel sistema e i vari canali di connessione [18]. L’interfaccia TwinCat I/O permette, invece, di creare applicazioni Windows CE e di accedere al sottosistema TwinCat I/O senza utilizzare la comunicazione tramite ADS. Per fare questo viene messa a disposizione una DLL con funzioni specifiche che consentono di comunicare con i dispositivi attraverso il fieldbus e di modificare le informazioni di processo.

Per i dispositivi Beckhoff basati su Windows CE viene messa a disposizione la funzionalità “TcTimer” che fornisce un timer deterministico che consente di eseguire programmi scritti in con codici di programmazione comuni (ad esempio C++). Tale funzione “TcTimer” di base può essere scalata fino $100\mu s$ ed è derivata dal livello maggiore di priorità di Windows CE. Essa permette all’applicazione creata di accedere, per mezzo di un puntatore, al “IO-buffer”, di accedere in modo sincrono al fieldbus e di cambiare le informazioni dei vari task mappando le modifiche sul livello fisico degli I/O.

5.2.2 Configurazione dei dispositivi EtherCAT

L’applicazione di controllo viene sviluppata e compilata nella Workstation mediante l’utilizzo di un ambiente di sviluppo, ad esempio Microsoft Visual Studio. L’utilizzo dei dispositivi EtherCAT necessitano di alcune configurazioni ed impostazioni preliminari.

Configurazione dispositivo

Dopo aver collegato i vari dispositivi utilizzati alla rete EtherCAT, è necessario farli riconoscere al software. Per fare questo viene quindi utilizzato il software di Beckhoff TwinCAT System Manager. Dopo la scansione automatica, i dispositivi riconosciuti vengono elencati in una sottofinestra del programma. In figura 5.3 viene illustrata la procedura necessaria alla scansione dei dispositivi ed il risultato finale ottenuto.

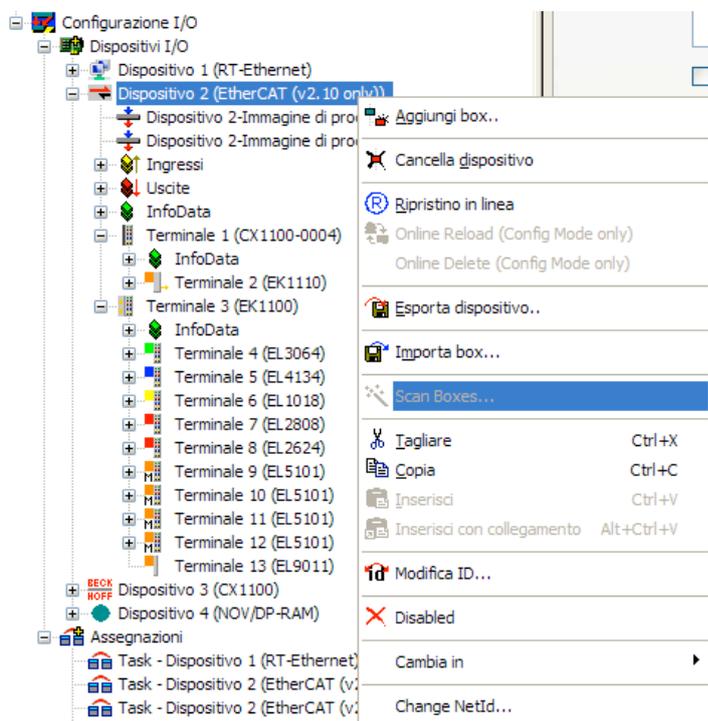


Figura 5.3: Scansione rete EtherCAT al fine di trovare i dispositivi connessi

Selezionando un dispositivo di interesse è possibile visualizzare le informazioni generali della periferica e le configurazioni della rete EtherCAT, come ad esempio il numero del nodo e l'indirizzo del dispositivo all'interno della rete. Di particolare importanza è la finestra relativa ai "Process Data" che contiene informazioni ed indirizzi relativi ai parametri e ai valori scambiati tra il dispositivo ed il controllore. Inoltre, è possibile accedere alla configurazione di impostazioni e parametri supplementari. Per esempio, è possibile impostare se il contatore degli encoder sia un registro a 16 bit oppure a 32 bit, oppure configurare le impostazioni relative al watchdog hardware integrato nei dispositivi Beckhoff.

Task aggiuntivi

Per garantire all'applicazione di controllo l'accesso diretto alle variabili è necessario costituire un task aggiuntivo, utilizzando il software di Beckhoff TwinCAT System Manager. Una volta creato un *Additional Task* si possono aggiungere le variabili di input ed output che servono per l'applicazione di controllo. Tali variabili vengono successivamente collegate agli ingressi e alle uscite dei dispositivi, presenti nella

rete EtherCAT. Per non incorrere in errori indesiderati, è opportuno includere al processo anche le variabili `EtherCAT_DevState` e `CX1100_State` al processo.

In figura 5.4 è possibile verificare il procedimento descritto in precedenza; sono presenti una parte degli input utilizzati nel prototipo Sophia-3. Si nota come le variabili presentino il loro indirizzo e la loro grandezza; il software di gestione controlla che non sia possibile assegnare una variabile di ingresso o uscita ad una variabile di tipo differente.

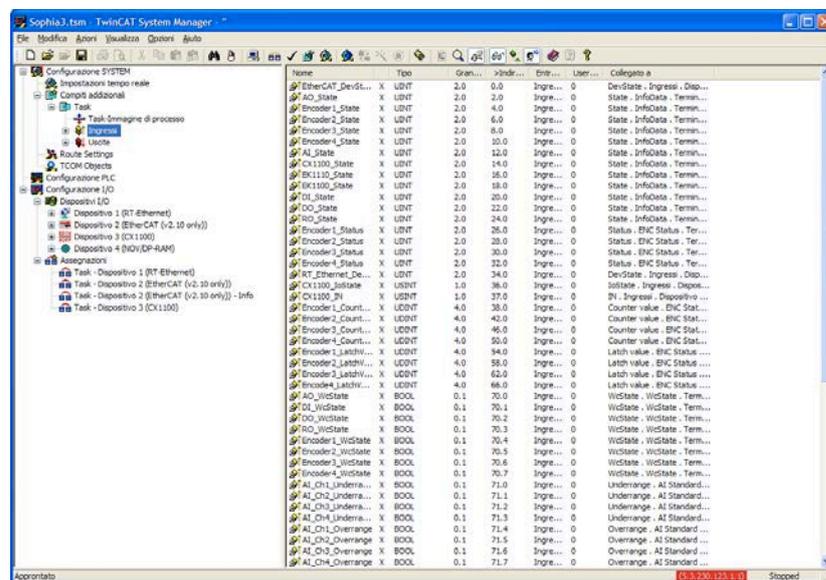


Figura 5.4: Ingressi digitali dei vari dispositivi mappati all'interno del Task creato

Ogni dispositivo della rete EtherCAT necessita di una “*Sync unit*”, che definisce un insieme di dati di processo, scambiati in modo sincrono tra il controllore e il driver. Nella nostra applicazione ogni oggetto PDO è assegnato alla stessa area di processo in modo tale che tutte le variabili siano scambiate e modificate con la stessa sincronizzazione. Al fine di verificare quanto descritto in precedenza, è necessario verificare le impostazioni avanzata della rete EtherCAT. In figura 5.5 è presente la schermata di interesse.

Cliccando sul pulsante “*Sync Unit Assignment*” si può accedere alla finestra corrispondente. In questa finestra, è necessario assegnare il particolare modulo al particolare task di interesse. Nello specifico, tutti i dispositivi EtherCAT sono stati assegnati allo stesso task. In figura 5.6 è presente il risultato di quanto descritto.

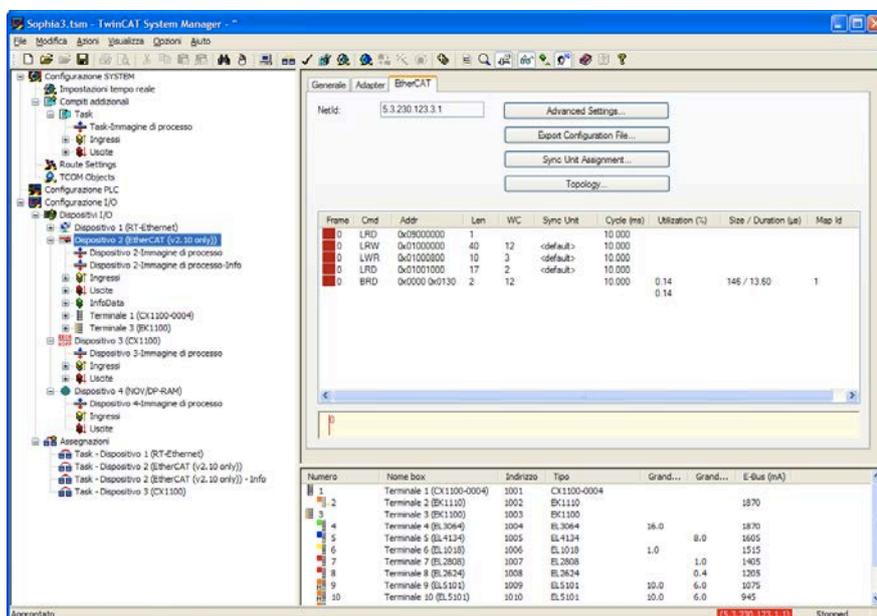


Figura 5.5: Impostazioni avanzate per la gestione dei dispositivi EtherCAT

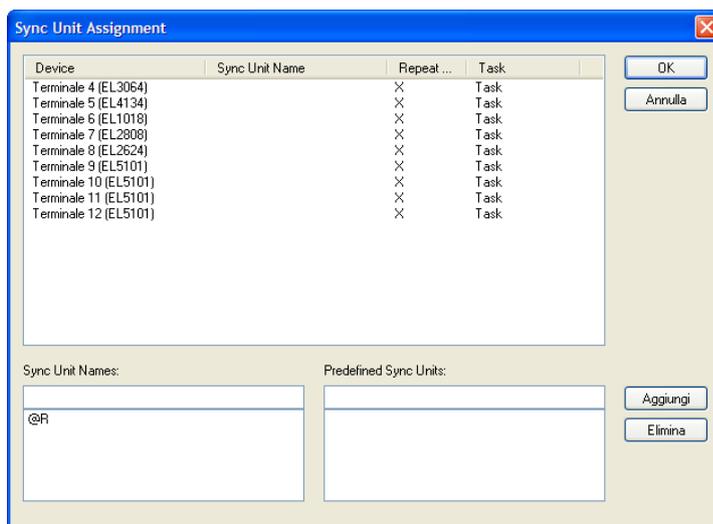


Figura 5.6: Assegnazione SyncUnit ai dispositivi EtherCAT per un determinato Task

Ulteriori impostazioni da settare sono relative al *Task*. Le impostazioni che si possono visualizzare in figura 5.7 permettono il corretto funzionamento del software di controllo.

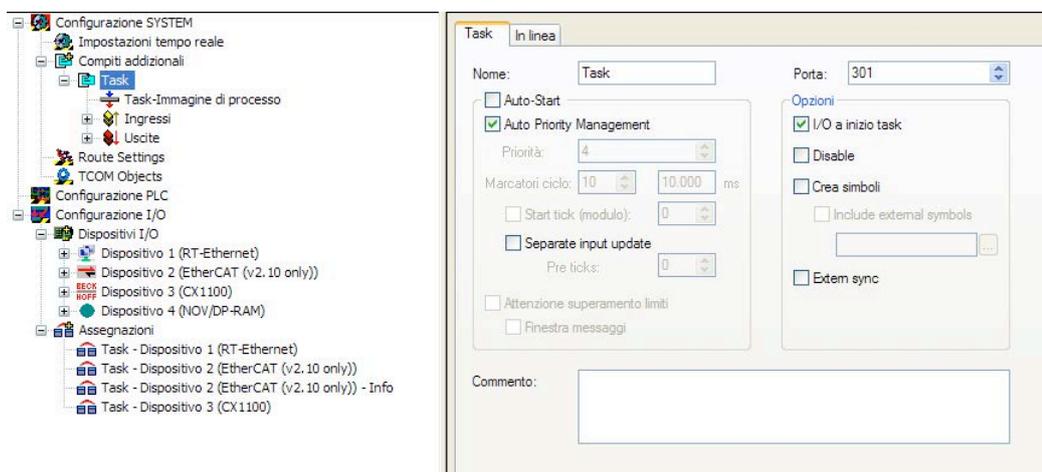


Figura 5.7: Configurazione del Task

A seconda della specifica applicazione in fase di realizzazione, è necessario impostare il tempo di ciclo a cui il task dovrà funzionare. Cliccando sulla voce *Impostazioni tempo reale*, apparirà una schermata simile a quella raffigurata in figura 5.8. Basterà selezionare il quanto temporale di interesse per l'applicazione.

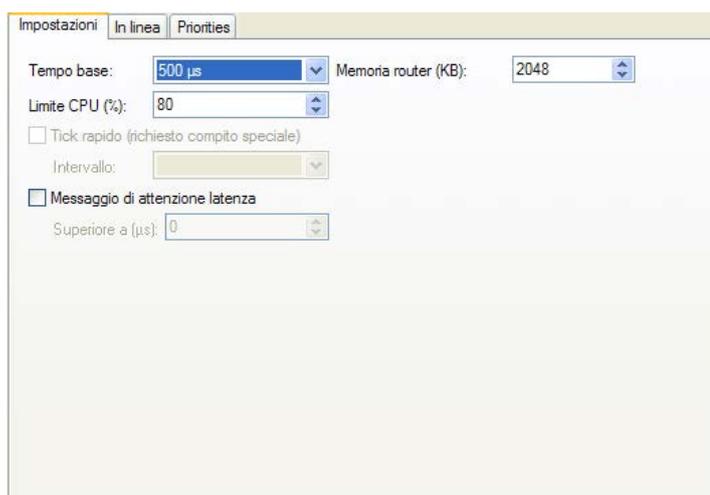


Figura 5.8: Configurazione delle impostazioni real-time del Task

A questo punto, per poter salvare le impostazioni nel controllore, è necessario:

-  generare l' assegnazione;
-  verificare la configurazione;
-  attivare la configurazione.

La configurazione tra variabili del task addizionale e ingressi ed uscite del dispositivo va quindi attivata cliccando sul tasto “apply configuration” presente nel tool bar di TwinCAT System Manager. Tale comando crea la mappatura delle variabili con i corrispondenti fieldbus I/O e genera le assegnazioni del compito addizionale. Come ultima cosa basterà riavviare il programma nella modalità “Run Mode”, premendo il tasto “start/restart” dopo aver salvato.

L'ultima operazione da compiere è quella relativa all'esportazione dell'header del task, che successivamente andrà incluso nel progetto software dell'IDE utilizzato per sviluppare l'applicazione di controllo. Il file così ottenuto contiene il numero di porta ADS del task, la dimensione delle immagini di processo degli input e degli output (in bytes) e le variabili dell'immagine di processo che vengono rappresentate mediante strutture, una relativa agli input ed un'altra relativa agli output. In figura 5.9 è rappresentato il processo di esportazione.

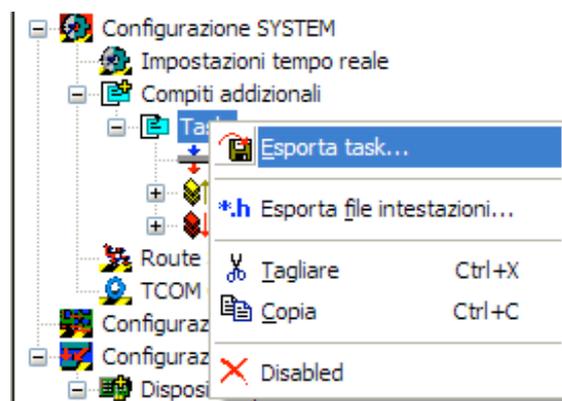


Figura 5.9: Esportazione del file header al fine di essere incluso nel progetto software

5.3 Programmazione driver Cello

Il drive Cello è un potente servo drive che può operare in modalità stand-alone o come parte di una rete “multi-axis”. Set up e tuning dell'azionamento vengono

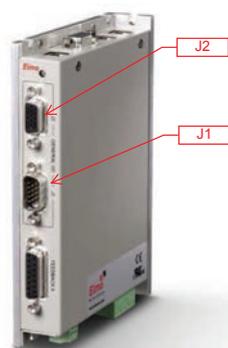


Figura 5.10: Cello digital servo drive

eseguiti con il software “Elmo’s Composer”, una applicazione che consente agli utenti di configurare rapidamente e semplicemente il servo drive per un utilizzo ottimale con i relativi motori. Cello incorpora un motion controller digitale con caratteristiche di connessione di rete CANopen, un loop di posizione, un loop di velocità, di corrente ed un ampio range di opzioni di feedback e di I/O.

Il Cello è dotato di due porte di I/O, J1 può essere utilizzato per collegare 6 input digitali e 5 output digitali. J2 è una porta di input che permette il collegamento di 4 distinti ingressi digitali e 2 ingressi analogici. La descrizione dettagliata di J1 e J2 (tipologia di connettori, funzionalità di ciascun pin, schemi di riferimento) è riportata nel manuale MAN-CELIG (Cello Servo Drive - Installation Guide) ¹ L’esigenza dell’applicazione che coinvolgono I/O di driver (e controllore) è data dalla funzionalità di **Inhibit/enable motor**. I driver non devono essere in grado di fornire comandi di potenza ai motori qualora la macchina non sia in stato di marcia. Impostando un ingresso come “inhibit” si assicura che, in stato di arresto, i driver siano disabilitati e che quindi non ci sia assolutamente nessun coinvolgimento dei motori.

5.3.1 Controllo in coppia

I motori 1,2 e 3 vengono controllati in coppia mediante un input analogico del modulo EL3064. I driver sono stati tarati per poter generare la coppia massima

¹reperibile nelle sezione *Downloads > Manuals* del sito <http://www.elmomc.com>.

in presenza del massimo valore di tensione fornito (10 V) dal modulo EtherCAT. In seguito è proposto il codice utilizzato per la programmazione del driver.

```
#@AUTOEXEC
//Comandi da impartire prima della programmazione del driver
//TS=70; SV;          //Aumento della frequenza PWM
MO=0;                //Motor disable
YA[4]=4;            //Emulazione Encoder
AS[1]=0.041561571;  //Offset riferimento analogico
KP[1]=16;
KI[1]=2000;
XP[2]=4;            //Fast PWM
RM=1;                //Enable analog ref input
UM=1;                //Torque mode
AG[1]=-0.539746;    //Fattore di scala
CL[1]=4.5;          //Massima corrente continuativa
PL[1]=5.5;          //Massima corrente picco
MO=1;                //Motor enable
TC=0;                //Riferimento coppia interno
UI[1]=AN[1];        //User Float Array ad input analogico
DV[1];              //Comando Torque Control
```

5.3.2 Controllo in velocità

Il moto traslatorio del carrello porta-puleggia, ottenuto tramite vite a ricircolo è movimentato dal motore 4. Sul driver che controlla il motore, è stato programmato un controllo in velocità. La tensione fornita dal modulo EL3064 sarà quindi proporzionale alla velocità di rotazione del motore. A livello software del controllore si effettua in controllo ad anello chiuso della variabili dei posizione.

```
#@AUTOEXEC
//Comandi da impartire prima della programmazione del driver
//TS=70; SV;          //Aumento della frequenza PWM
MO=0;                //Motor disable
YA[4]=4;            //Emulazione Encoder
AS[1]=4.88138282;    //Offset riferimento analogico (5V - offset)
KP[1]=8;
KI[1]=2000;
XP[2]=4;            //Fast PWM
```

```
RM=1;           //Enable analog ref input
UM=2;           //Speed mode
AG[2]=50000;    //Fattore di scala (V/tacche encoder)
CL[1]=2;        //Massima corrente continuativa
PL[1]=2.5;      //Massima corrente picco
MO=1;           //Motor enable
TC=0;           //Riferimento coppia interno
UI[1]=AN[1];    //User Float Array ad input analogico
DV[2];          //Comando Speed Control
```

Nella programmazione dei driver, non sono stati utilizzati i comandi specifici `CA[16]` e `CA[25]`, i quali permettono di settare il verso positivo di rotazione e di erogazione della coppia. Con l'utilizzo di tali comandi, i driver presentavano dei malfunzionamenti; si sono quindi impostate correttamente le variabili di alcuni parametri del driver. Ad esempio è stato impostato il fattore di scala `AG[1]` negativo, ottenendo lo stesso risultato dell'istruzione `CA[25]=1`.

5.4 Software di Controllo

Il software di controllo del sistema Sophia-4 si basa su una rete di processi connessi tra loro sviluppata ex novo. Questa struttura risulta essere una buona soluzione di partenza per il controllo in tempo reale di una macchina a stati finiti e, quindi, si presta ad un vastissimo numero di applicazioni per le quali, ovviamente, deve essere modificato ed adattato. Nei paragrafi successivi verranno analizzati singolarmente i vari task, dando una breve descrizione della loro forma e della loro funzione all'interno del software principale.

Il software, scritto in C++, è stato sviluppato secondo il paradigma della programmazione ad oggetti. Ogni oggetto ha specifiche competenze e funzionalità; nel file header (`.h`) dell'oggetto sono presenti le definizioni dei metodi della classe mentre nel file (`.cpp`) le implementazioni.

Nel file `commonDefine.h` sono specificati tutte le costanti utilizzate nel software, oltre che alla definizione delle varie strutture utilizzate. Ad esempio, ad

ogni stato della macchina è associata una particolare struttura dati contenente le variabili utilizzate. Inoltre, sono presenti le enumerazioni degli stati della macchina e dei comandi utilizzati nell'interscambio dei dati tra i vari thread ed il software di gestione esterno.

All'interno del file `stdafx.h` sono presenti i file di inclusione per il sistema standard o file di inclusione specifici del progetto utilizzati di frequente, ma modificati raramente.

5.4.1 Metodi di sincronizzazione tra thread

All'avvio dell'applicazione vengono creati i thread costituenti l'architettura di controllo del prototipo. Al fine di sincronizzare la partenza tra i vari thread viene utilizzato un semaforo binario, denominato `startSemaphore`, mediante il suo rilascio. Per terminare l'esecuzione dei thread è stato utilizzato un event, chiamato `stopEvent`. Gli event sono una forma elementare di sincronizzazione diretta: la funzione `pulseEvent(HANDLE hEvent)` viene utilizzata per liberare tutti i thread in attesa e per cancellare automaticamente l'evento. Dopo tale operazione vengono compiute tutte le operazioni necessarie a liberare le risorse in precedenza allocate.

5.4.2 Comunicazione tra thread

Le `MessageQueue` (Code di Messaggi) costituiscono un metodo di comunicazione tra processi, di livello superiore ai semafori. Esse permettono l'invio e la ricezione di un numero variabile di messaggi, mediante una coda FIFO, tra un task e l'altro, tenendo presente che una comunicazione full-duplex tra due processi richiede due code distinte. Una volta create ed inizializzate, esse instaurano un vero e proprio canale di comunicazione tra due task. Il processo che trasmette non deve fare altro che inviare un messaggio alla coda che il processo ricevente gestirà appena avrà disponibilità delle risorse necessarie. Mediante l'utilizzo di una union, vengono inglobate tutte le strutture dati necessarie alla comunicazione tra thread. Mediante la lettura della variabile `command` è possibile risalire al comando ricevuto

ed utilizzare l'opportuna struttura dati. Si riporta, come esempio, le strutture dati utilizzate a tal scopo.

```
union typeDataCommunication
{
    structDataExercize dataExercize;
    structDataLog dataLog;
    structDataNull dataNull;
    structRtData rtData;
    structDebugMode debugMode;
    structStrData strData;
    structDataWallForce wallForce;
    structDataCircleWallForce circleWallForce;
};

struct DataCommunication
{
    int command;
    typeDataCommunication cmdStruct;
};
```

Nella sottosezione 5.4.3 verrà spiegato il funzionamento della classe sviluppata a tal scopo.

5.4.3 Oggetti implementati

In questa sottosezione verranno illustrate le classi implementate, descrivendone il loro funzionamento.

Conversion

All'interno di tale classe sono contenuti i metodi utilizzati per poter gestire, in maniera agevole, le conversioni tra unità di misura o entità differenti.

- `voltageToShort` permette di convertire un livello di tensione elettrica (V) nella corrispondente variabile `short`, la quale sarà utilizzata come input software per gestire il modulo IO. Un suo overload permette la gestione di offset, in quanto la variabile di ingresso dei moduli è di tipo `unsigned`;

- `velocityToVoltage` effettua la conversione da un valore di velocità espresso in m/s ad un riferimento di tensione. Tale metodo è utilizzato ad esempio per il controllo in velocità del motore mobile;
- `millimeterToEncoder` converte da mm a tacche encoder, considerando il CPR (Count Per Revolution) dell'encoder considerato;
- `alphaToEncoder` restituisce in valore in tacche encoder dell'angolo α espresso in radianti. A tal scopo sono necessari la conoscenza del parametro CPR e del rapporto di riduzione k . Quest'ultimo è diverso da 1 nel caso del motore 1, in quanto il cavo viene riavvolto in una vite calettata dal diametro differente della puleggia.
- `encoderToAlpha` presenta la funzione inversa del metodo precedente;
- `tensionToVoltage` tale metodo permette di convertire una tensione espressa in Newton (N) nel riferimento di tensione (V) corretto.

Dynamics

Tale classe è adibita ai calcoli dinamici durante varie fasi di terapia. I metodi pubblici sono:

- `tensionDistribution` calcola la tensione da attribuire ai cavi dati in ingresso il vettore β , ovvero gli angoli della matrice di struttura 3.46, ed il vettore \mathbf{F} . Un suo overload permette di ottenere i valori di tensioni computati senza effettuare la scalatura delle stesse;
- `isTensionScaled` permette di conoscere se le tensioni dei cavi sono state scalate o meno.

Encoder

Questo oggetto è stato implementato al fine di rendere più agevole e flessibile la gestione degli encoder. Ad esempio, se si volessero settare gli encoder mediante una serie di comandi EtherCAT, sarebbe necessario attendere un determinato tempo, oltre che eseguire una determinata serie di operazioni, affinché essi vengano svolti.

Con tale gestione, invece, le operazioni di set e reset degli encoder saranno immediatamente effettuate. Compito del costruttore dell'oggetto è quello di mappare il corretto indirizzo dell'encoder nella struttura dati utilizzata da TwinCAT I/O e di associare il verso di rotazione ad un incremento positivo delle tacche/encoder. Il metodo `readEncoder` permette la lettura del valore tacche encoder, mentre `setEncoder` ha il compito di settare in modo opportuno l'encoder.

Kinematics

Tale classe è stata sviluppata per svolgere i calcoli riguardanti la cinematica del Sophia-3.

- `directKinematics` tale funzione permette di svolgere la cinematica diretta, utilizzando gli algoritmi descritti in dettaglio nella sezione 3.1;
- `reverseKinematics` tale metodo permette di eseguire i calcoli necessari al fine di risolvere il problema di cinematica inversa; la variabile booleana `fixedMotor` permette di stabilire se per tale calcolo si considera il motore mobile in una posizione fissa o meno (particolare utilizzato, ad esempio, nell'algoritmo di autocalibrazione);
- `calculatesMotorPosition` calcola la posizione del motore mobile, date le coordinate dell'EE;
- `calculatesVectorBeta` permette di calcolare i valori del vettore β in maniera computazionalmente meno esosa in termini di operazioni effettuate rispetto alla cinematica inversa;
- `calculatesMeanP` data la matrice \mathbf{P} contenente le coordinate dell'EE calcolate con le tre coppie di cavi, restituisce il punto medio;
- `getMotorPosition` restituisce la posizione in cui il motore mobile, teoricamente, dovrebbe possedere.

ManageIO

Questo oggetto permette di gestire i dispositivi I/O. Il costruttore di tale classe ha il compito di far puntare le variabili interne alle strutture di processo TwinCAT (PTask_Outputs pTOut e PTask_Inputs pTIn).

- `initializeIO` esegue le operazioni di inizializzazione dell'hardware. Inoltre setta le variabili di output ai valori iniziali;
- `getMotorPosition` permette di ottenere la reale posizione del motore mobile;
- `prepareEncoderToSetValue` tale funzione permette di preparare gli encoder, nel caso in cui si utilizzassero direttamente le variabili del processo EtherCAT, al settaggio del proprio counter.
- `unlockBrakes` e `lockBrakes` rispettivamente, sbloccano o bloccano i freni della macchina.

MessageQueue

Tale classe permette la gestione della comunicazione tra thread, in maniera semplice, attraverso l'astrazione di particolari funzioni o costrutti del sistema operativo Windows CE. Oltre al costruttore, in cui viene specificato il nome univoco della coda di messaggi e stabilita la direzione della coda attraverso l'appropriato settaggio della variabile bool `bReadAccess`, le funzioni principali sono `writeMsgQ` e `readMsgQ` che permettono di inviare e ricevere i dati. La funzione `getNmessage` permette di conoscere il numero di messaggi presenti in coda; utile nel caso in cui si voglia processare più comandi all'interno dello stesso ciclo di controllo.

Tokenizer

La gestione della comunicazione attraverso la GUI di controllo, o qualsiasi altra tipologia di applicazione, avviene attraverso il protocollo TCP. Quando il data-rate di trasmissione aumenta, a causa del fenomeno di auto-adattamento della finestra di trasmissione, è possibile che la funzione `rec`, facente parte della libreria `WinSock`, restituisca una concatenazione dei messaggi spediti dal software remoto. La classe

Tokenizer ha lo scopo di separare ed eventualmente ricomporre la suddivisione in più parti di una sequenza dati. Viene utilizzata anche per la gestione dei vari parametri, che compongono un comando, ricevuti dalla GUI.

5.4.4 I thread attivi

In questa sezione vengono presentati i thread che compongono il sistema di controllo. In figura 5.11 si può vedere l'organigramma che descrive le interazioni esistenti tra i vari processi del programma.

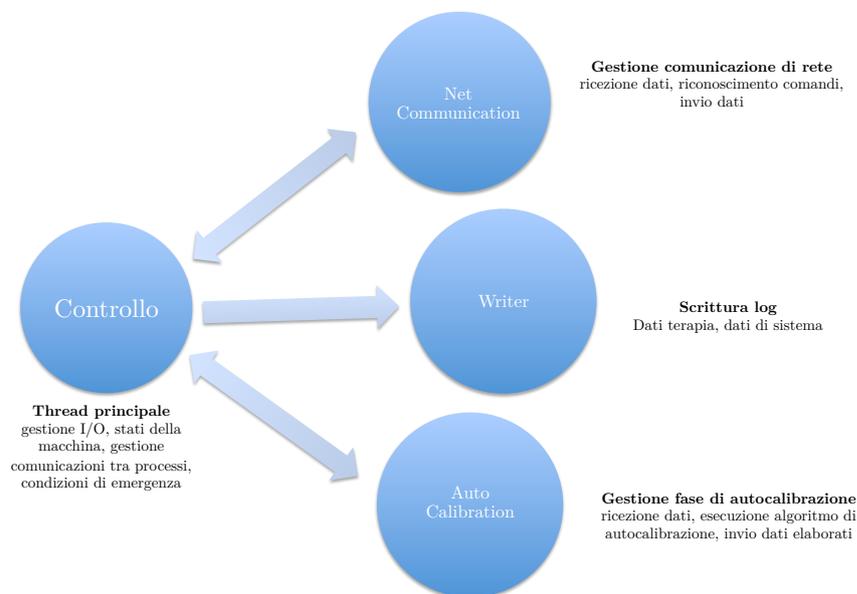


Figura 5.11: Organigramma dei thread in esecuzione

Controllo

Questo thread è il cuore principale del sistema di controllo, gira a priorità real-time e si occupa di inizializzare e gestire l'hardware. Ad ogni ciclo, ha il compito di aggiornare gli input e gli output, eseguire il codice relativo al preciso stato in cui la macchina si trova, spedire eventualmente dei dati agli altri processi, controllarle l'arrivo ed eventualmente elaborarli. Ad ogni ciclo vengono verificate le condizioni di emergenza: nel caso in cui almeno una delle condizioni si verificasse la macchina va immediatamente in EMERGENZA. La comunicazione tra processi è gestita

tramite MessageQueue, come spiegato nella sezione 5.4.3.

L'inizializzazione dell'hardware avviene attraverso la seguente porzione di codice:

```
// Get the process image pointer
if ( TCatIoGetOutputPtr(TASK_PORT,
    (void**)&pTOut, sizeof(Task_Outputs) ) == 0 &&
    TCatIoGetInputPtr( TASK_PORTNUMBER, (void**)&pTIn,
    sizeof(Task_Inputs) ) == 0) {
// Create Events for the Tasks
g_hEvent=CreateEvent(NULL, FALSE, FALSE, TEXT("evt_TASK_TICK"));
// Set Events in TcTimer
idTimer=TcTimerSetEvent(TASK_DELAY, TEXT("evt_TASK_TICK"),
    TIME_CALLBACK_EVENT_PULSE|TIME_PERIODIC);
}
else {
printf("Failed to get the process image pointer.\n");
return FALSE;
}
```

in cui prende il puntatore all'immagine di processo TwinCAT, si crea l'evento per il task di IO, e successivamente si setta tale evento in un timer deterministico.

La porzione di codice per poter leggere e scrivere gli input e gli output è possibile riassumerla in poche righe, di seguito esposte:

```
while(WaitForSingleObject(stopEvent, 0) == WAIT_OBJECT_0) {
    WaitForSingleObject(g_hEvent, INFINITE);
    if ((errorCodeTwincatIO = TCatIoInputUpdate(TASK_PORT)) == 0) {
        // CODE //
        TCatIoOutputUpdate(TASK_PORT);
    }
}
```

`g_hEvent` rappresenta l'evento del timer deterministico offerto dalle librerie Beckhoff, `TCatIoInputUpdate` e `TCatIoOutputUpdate` le funzioni utilizzare per aggiornare l'immagine di processo. In caso di errori, la valutazione della variabile `errorCodeTwincatIO` permette di ottenere il codice di errore relativo al fault avvenuto.

NetCommunication

Tale thread permette di gestire la comunicazione di rete creando un server TCP in ascolto sulla porta 5000. Gestisce possibili disconnessioni accidentali e altre problematiche relative alla trasmissione di rete. Mediante l'istruzione `iReturn=recv(remoteSocket, buffer, NBUFFER, 0)` riceve i dati dalla scheda di rete, provvedendo a raccordarli mediante la funzione `receiveMessage` dell'oggetto `stringTokenizer`. Ha inoltre il compito di gestire i comandi di sistema o di terapia, estrapolando, eventualmente, i parametri ricevuti.

Writer

Il compito di questo thread è quello di scrivere in un file di testo tutte le variabili che potrebbero servire ad una elaborazione off line. Alla ricezione del comando `CMD_DATA_TO_LOG` da parte del thread Controllo, appende al file di testo il contenuto della struttura `cmdStruct.dataLog`.

Autocalibration

Tale thread permette di svolgere i calcoli necessari all'autocalibrazione. Tale algoritmo non è stato implementato nel thread di controllo al fine di poterlo eseguire a bassa priorità.

In figura 5.12 è rappresentato il diagramma UML degli stati implementati della macchina.

5.4.5 Implementazione della macchina a stati finiti

La macchina a stati finiti è stata implementata con l'ausilio di puntatori a funzioni. Tale metodo, permette di non utilizzare il costrutto `switch` al fine di rendere il codice maggiormente leggibile e gestibile. Attraverso la definizione di un nuovo tipo di dato, nello specifico `typedef void (*StateFunc)(void)`, è possibile creare un array di puntatori a funzioni. A titolo di esempio, la creazione di tale costrutto può avvenire con le seguenti istruzioni semplificate:

```
const StateFunc state_machine [NSTATE] =  
{
```

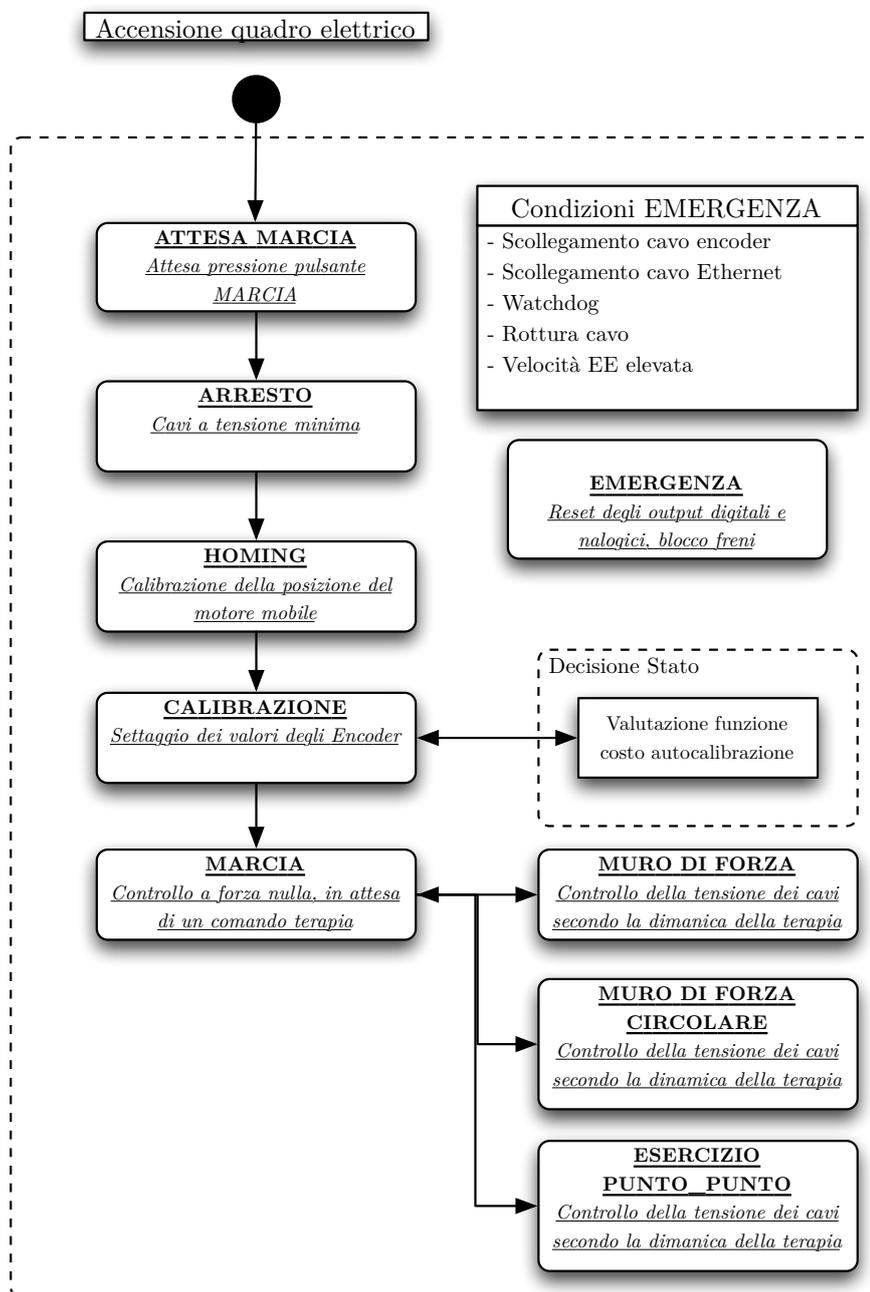


Figura 5.12: Diagramma UML della macchina a stati implementata nel software del controllore

```
        state1 ,
        state2 ,
};
```

All'interno del ciclo di controllo, l'esecuzione di un particolare stato `state` può essere richiamata con l'istruzione `state_machine[state]()`;

Nel software di controllo ad ogni stato dovranno essere assegnate precise istruzioni da compiere all'ingresso, all'uscita e durante l'esecuzione di ogni ciclo di controllo. Per soddisfare tali specifiche implementative, la funzione di cambio stato `goState` è implementata nel seguente modo:

```
void goState(int _state)
{
    exit_state_machine[state]();
    state=_state;
    enter_state_machine[state]();
}
```

5.4.6 Protocollo di comunicazione

Il riconoscimento del comando ricevuto, come descritto nella sottosezione 5.4.3 , si basa su un algoritmo di string tokenizer, associato ad un semplice riconoscimento diretto di una serie di caratteri. Allo stadio implementativo attuale il software di controllo gestisce i seguenti comandi impartiti:

- **OK_CONNECT**: Con questo comando il controllore è in grado di inizializzare una comunicazione con una applicazione di comunicazione;
- **CALIBRATION**: Permette di stabilire che il punto in cui si trova l'End-Effector sia la posizione esatta in cui effettuare l'operazione di calibrazione;
- **TROVA_PUNTO**: Stringa per poter portare l'End-Effector in un determinato punto del piano di lavoro;
- **ARRESTO**: Il software di controllo va nello stato *ARRESTO*. In tale situazione i driver dei motori vengono disabilitati;

- **HOMING**: Con questo comando il software di controllo è in grado di avviare il processo di homing per il motore mobile;
- **WALL_FORCE**: Tale comando permette di eseguire l'esercizio del muro virtuale.

Il protocollo di comunicazione per poter impartire al sistema di controllo la creazione di un muro di forza è il seguente:

```
WALL_FORCE@XM@YM@THETA@K@
```

Dove, rispettivamente, **XM**, **YM** sono le coordinate dell'origine del sistema di riferimento del muro di forza. **THETA** rappresenta l'inclinazione di tale muro e **K** la rigidità di quest'ultimo. L'algoritmo in grado di estrarre dalla stringa di testo ricevuta i parametri relativi all'esecuzione dell'esercizio è stato implementato in modo tale da rendere semplice la possibilità di modificare il numero e la tipologia di valori ricevuti come ingresso. È anche possibile effettuare un overload del comando di generazione dell'esercizio in base al numero di parametri forniti come input.

Durante l'esercizio il controllore invierà alla GUI i seguenti dati:

- **p[0], p[1]**: le coordinate dell'end-effector;
- **t[0], t[1], t[2]**: le tensioni dei cavi non scalate;
- **tns[0], tns[1], tns[2]**: le tensioni dei cavi scalate;
- **mMotor[0][0]**: la posizione x del motore mobile;
- **beta[0], beta[1], beta[2]**: il vettore β per poter costruire off-line la matrice di struttura.

5.5 Controllo del motore DC: macchina a stati

In questa sezione viene presentato il software per poter controllare il motore a corrente continua. L'obiettivo principale del driver è quello di fornire un profilo

di tensione trapezoidale al motore. Rispetto ad una semplice legge on-off, questi profili permettono di ottenere un movimento esente da forti vibrazioni.

Il microcontrollore gestisce quattro input digitale e tre uscite digital. Gli input digitali denominati INHBT0 e INHBT1 (pin 2 e 3 del dsPIC) corrispondono ai micro interruttori posti ad individuare i limiti di movimentazione del piano (0 e 60 gradi). Un livello logico alto corrisponde al fatto che il tavolo ha raggiunto il limite, con ciò tutte le sotto sequenze di input dirette oltre tale posizione, saranno naturalmente ignorate.

Gli input digitali chiamati DIR0 e DIR1 (pin 4 e 5) corrispondono ai comandi di moto *down* ed *up*, rispettivamente. Essendo in logica negata, un livello basso produrrà l'inizio del moto nella direzione prestabilita.

Gli output digitali CTRL_A e CTRL_B (pin 38 e 36) sono connessi alle due porte PWM del microcontrollore (PWM1L e PWM2L). Queste porte controllano i MOSFET di potenza del ponte ad H attraverso i transistor BJT. Il duty-cycle della PWM è visualizzato da due LED. Il terzo output digitale, denominato LED_STATUS (pin 30), è connesso al LED di status. Questo LED permette di fornire informazioni sullo stato operativo del controllo, lampeggiando a differenze frequenze.

Il software, scritto in C++, simula un sistema operativo real-time con un unico thread. La frequenza di controllo è pari a $f = 1953 \text{ Hz}$, gestita da un timer hardware del microcontrollore. A cadenze temporali prestabilite, il timer permette di modificare la variabile *go*, variabile che il thread di controllo monitora al fine di far partire la sua esecuzione.

```
//Initialization Timer
// Priority level is 1
IPC0 = IPC0 | 0x1000;
// Timer1 interrupt enabled
IEC0 = IEC0 | 0x0008;
// T = PR1*25.6us
PR1 = 20;
// Timer1 enabled (internal clock divided by 256)
T1CON = 0x8030;

void Timer1Int() iv IVT_ADDR_T1INTERRUPT {
```

```

    T1IF_bit = 0;    // Clear T1IF
    go=~go;
}

```

Durante l'attesa per l'esecuzione del prossimo ciclo di controllo, il dsPIC viene messo in uno stato particolare, definito *power safe* al fine di limitare il consumo e la dissipazione termica del componente.

```

while(!go) asm PWRSAV #1;
go=0;

```

In figure 5.13 è rappresentata la macchina a stati finiti implementata nel dsPIC. All'accensione del sistema, lo stato di default è quello di *stand-by* e la variabile che indica la direzione del moto è settata al valore *WAIT*.

Ad ogni iterazione del ciclo di controllo, vengono effettuati i controlli riguardanti le *condizioni di emergenza*, l'eventuale *raggiungimento dei limiti* di lavoro, ed infine sul *cambiamento dello stato degli ingressi*.

Le verifiche riguardanti le *condizioni di emergenza*, vanno a verificare le potenziali discrepanze sugli input digitali (per esempio INHBT0 e INHBT1 sono simultaneamente settati a livello logico alto, oppure DIR_0 e DIR_1 sono entrambi a livello logico basso). Nel caso in cui una discrepanza venga rilevata, il sistema passa allo stato di *Emergenza*: i segnali PWM vengono ridotti a zero, i driver PWM vengono disattivati e tutti i futuri comandi verranno ignorati fino a che non sarà premuto il pulsante di reset del dispositivo. Il controllo sulle *raggiungimento dei limiti* legge i valori degli input INHBT0 e INHBT1: se un limite viene raggiunto, l'uscita PWM attualmente attiva viene azzerata e disabilitata, lo stato passa a *WAIT* e solo il movimento in direzione opposta sarà possibile. Il *cambiamento dello stato degli ingressi* monitora i valori DIR0 e DIR1: se nessun cambiamento è rilevato, lo stato rimarrà lo stesso di quello del ciclo di controllo precedente. Oltre allo stato di *WAIT*, vi sono altri tre stati corrispondenti alle tre fasi del profilo di tensione trapezoidale (*tensione costante, tensione crescente, tensione decrescente*). Se il moto è permesso lungo una data direzione, la tensione incrementa linearmente da zero al valore massimo consentito ($\delta = 100\%$) in un prefissato intervallo temporale ($\Delta t = 1s$). Quando il selettore viene riportato nella zona centrale, la tensione progressivamente diminuisce a zero con lo stesso coefficiente angolare. I cambi di

direzione durante una rampa di salita o discesa sono supportati. Addizionalmente, un controllo di posizione/tempo viene effettuato agli estremi di funzionamento: se il corrispondente segnale INHBT non cambia in una sufficientemente piccola quantità di tempo, si andrà nello stato di *emergenza*.

5.6 Revisione automatica del codice: cppcheck

Il codice sorgente dell'intero progetto è stato sottoposto a delle verifiche automatizzate da un apposito tool al fine di aumentare la qualità del codice, la sicurezza e di rilevare pratiche scorrette di programmazione. Cppcheck[27] è un tool open-source che supporta un'ampia varietà di controlli statici sul codice sorgente, che non vengono effettuati dal compilatore in modo autonomo. I controlli riguardano analisi statiche rigorose, piuttosto che euristiche, indirizzate a trovare le problematiche qui di seguito elencate:

- controllo automatico delle variabili;
- verifica del superamento dei limiti degli array;
- controllo delle Classi al fine di verificare funzioni non utilizzate, la corretta inizializzazione delle variabili e relative problematiche di duplicazioni in memoria;
- utilizzo di funzioni deprecate o superate in accordo con il consorzio Open-Group;
- controllo di sicurezza sulle eccezioni, sull'allocazione della memoria e sulla verifica dell'utilizzo dei distruttori;
- memory leaks, per prevenire la non deallocazioni delle variabili;
- resource leaks, per prevenire la non corretta chiusura degli handle di sistema;
- utilizzo non corretto della Standard Library;
- controllo su un determinato set di errori stilistici e di performance.

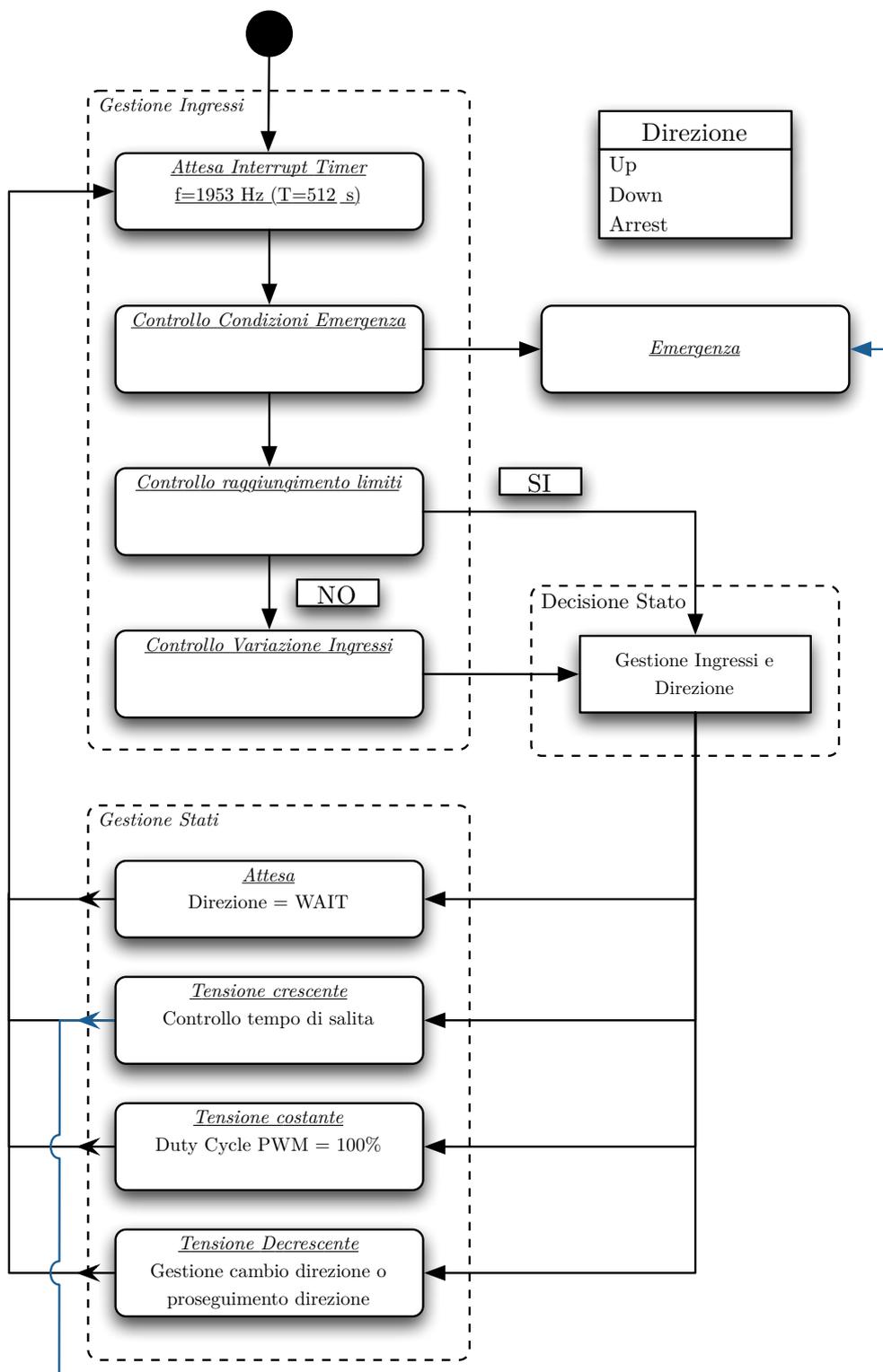


Figura 5.13: Diagramma UML della macchina a stati finiti dedicata al controllo del piano inclinabile

Capitolo 6

Risultati sperimentali

In questo capitolo verranno illustrati i primi test sperimentali ottenuti con il Beckhoff CX1020 e l'architettura EtherCAT. Una linea dritta è stata implementata come muro di forza all'interno dello spazio di lavoro, all'utente è stato chiesto di eseguire il compito di muovere l'end-effector lungo un segmento rettilineo, il quale parzialmente era all'interno del muro di forza virtuale.

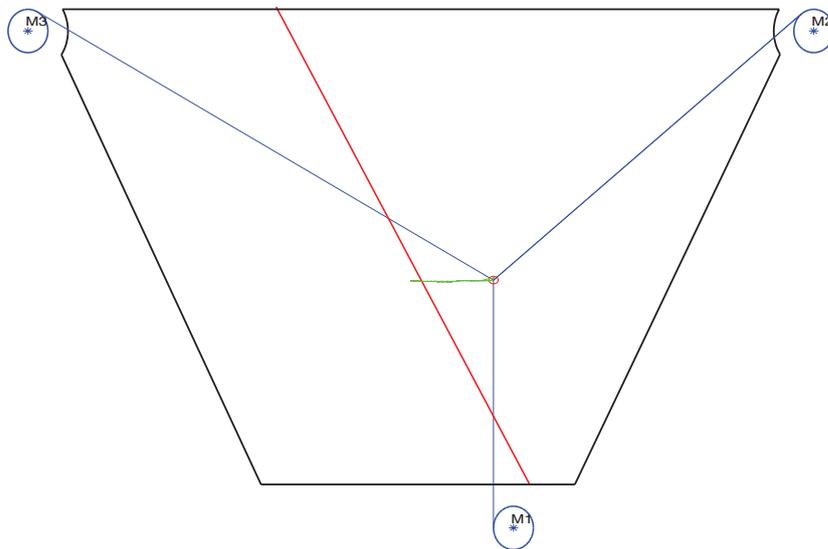


Figura 6.1: Percorso seguito dal paziente durante il test (linea verde)

La figura 6.1 mostra il percorso seguito dall'utente (curva verde), il quale è stato stimato dall'algorithmo di cinematica diretta. Le estremità del segmento lineare

rappresentano il percorso nominale sulla superficie del tavolo di legno (lunghezza totale 100 mm). Nel frame di riferimento (la quale origine coincide con il centroide del trapezio e l'asse x è orizzontale e orientato alla destra dell'utente) le coordinate sono: $x_1 = [-15, 0]^T$ e $x_2 = [85, 0]^T$. Il confine del muro di forza è indicato dalla linea continua rossa mostrata in figura 6.1, la quale divide lo spazio di lavoro in due regioni. Il muro virtuale è localizzato nella regione di sinistra. La linea forma un angolo $a = 2\pi/3$ con l'asse x e passa attraverso l'origine del frame di riferimento. Perciò, si suppone che l'utente entri all'interno del muro per approssimativamente per 15 mm . Il muro è perfettamente elastico, con $k_{\text{wall}} = 10\text{ N/mm}$. I seguenti valori sono stati scelti per garantire la stabilità delle interazioni.

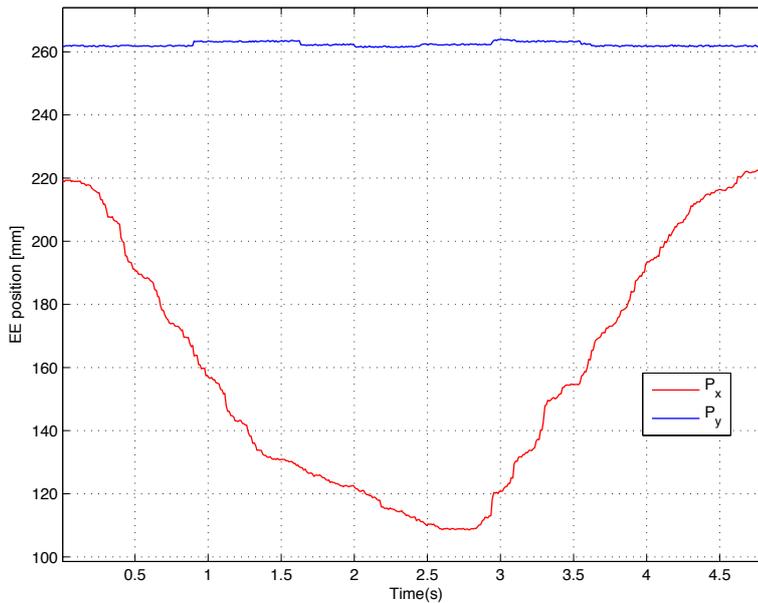


Figura 6.2: Proiezione della traiettoria compiuta dall'utente lungo gli assi x e y

La figura 6.2 mostra le proiezioni dell'attuale traiettoria lungo gli assi del frame di riferimento. L'utente per completare il percorso ha impiegato $\Delta t \approx 4,8\text{ s}$, con una velocità media $V_{\text{avg}} \approx 41,3\text{ mms}^{-1}$.

La figura 6.3 mostra le tensioni cui i cavi sono sottoposti (linee continue). Le tensioni sono normalizzate al valore massimo possibile ($f_{\text{max}} = 100\text{ N}$). Una tensione minima $f_{\text{min}} = 5\text{ N}$ è imposta. Le corrispondenti forze operative sono visualizzate in figura 6.4 (linee continue rosse) in coordinate polari. Ancora, le

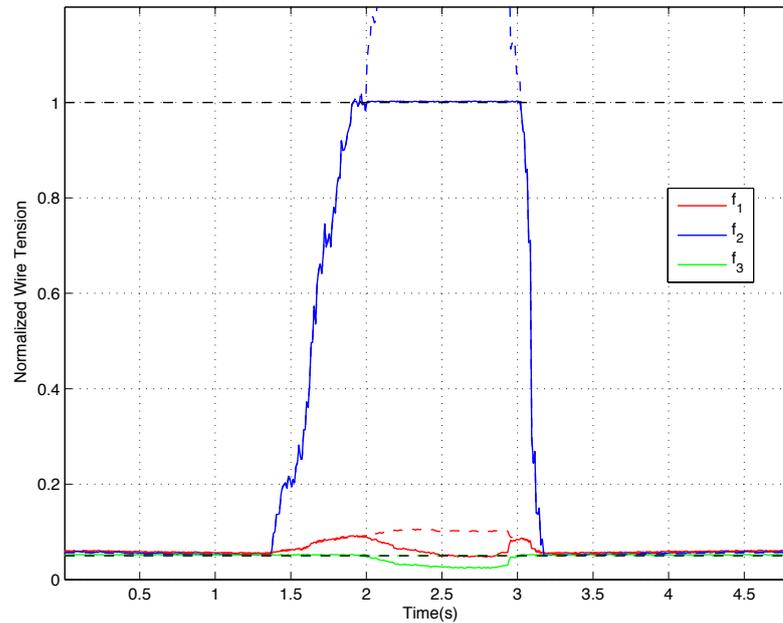


Figura 6.3: Tensioni dei cavi ottenute come stima dai monitor di corrente. Le linee tratteggiate corrispondono ad \mathbf{F}_{id}

tensioni sono normalizzate a f_{\max} .

Come l'end-effector si avvicina al muro virtuale muovendosi nel percorso orizzontale, le tensioni positive sono computate per fornire una forza operativa $\mathbf{F} = 0$ (diagramma superiore della figura 6.4). Appena l'end-effector entra nel muro, una reazione elastica è generata, la quale è ortogonale agli estremi del muro (in questo caso $\phi = \pi/6$). Oltre un certo valore X_{EE} , la forza richiesta causa un maggior carico su f_2 che raggiunge il limite superiore f_{\max} (figura 6.3). Un ulteriore incremento nel modulo di \mathbf{F} conduce alla saturazione dello stesso cavo, e l'attuale vettore delle tensioni dei cavi differisce in una componente da quella calcolata. Questo, in generale provoca che la forza esercitata differisce da quella richiesta sia per modulo che per direzione.

La figura 6.4 mostra che dopo la saturazione del cavo 2, non solo il modulo $\|\mathbf{F}_{\text{sat}}\|$ è minore di quella richiesta $\|\mathbf{F}_{id}\|$, ma anche, ma anche la forza non è diretta lungo $\phi_{id} = \pi/6$. In effetti, la forza esercitata mostra un incremento non desiderato nella componente x a sfavore della componente y . La situazione sarà peggiore se un secondo cavo sarà saturato. Per superare questo inconveniente,

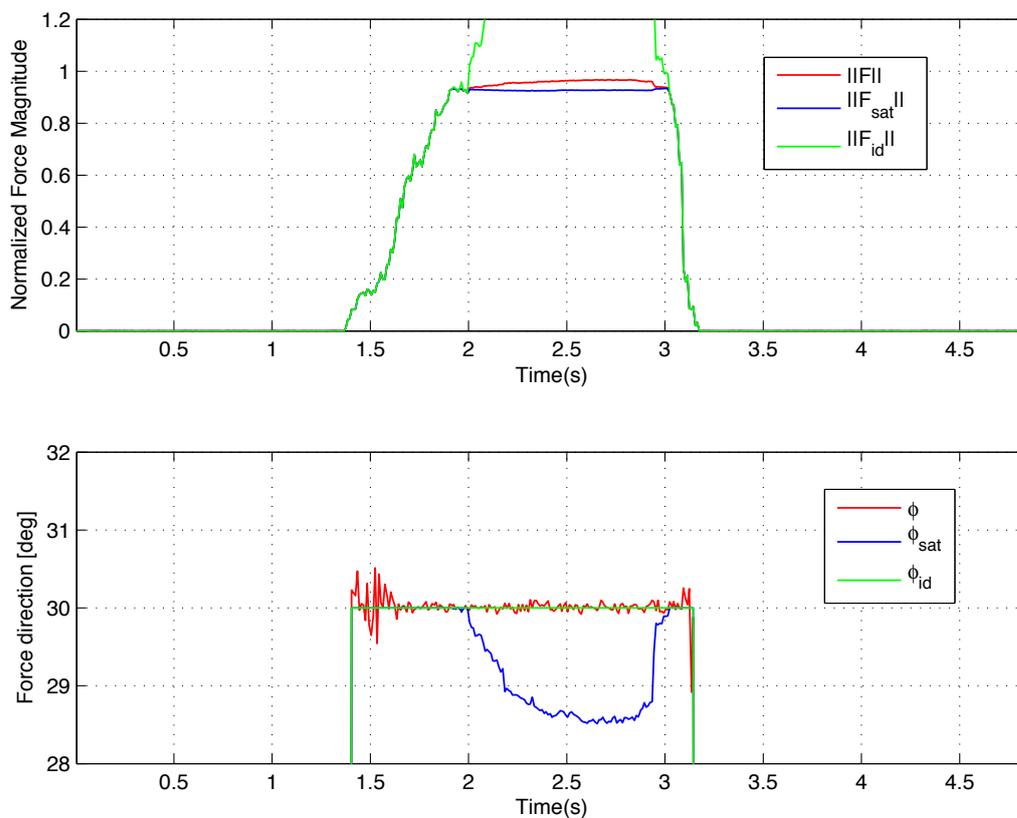


Figura 6.4: Forze operazionali in coordinate polari: forza attuale esercitata (\mathbf{F}), muro di forza virtuale (\mathbf{F}_{id}) e forza ottenuta mediante la non controllata saturazione di f_2 (\mathbf{F}_{sat})

tutte le forze vengono scalate al raggiungimento della tensione massima f_{\max} da parte di un cavo. Data la relazione lineare dell'equazione 3.43, il vettore delle forze esercitate avrà la stessa orientazione delle forze nominali ed un modulo ridotto da un fattore di scala.

La figura 6.3 mostra le tensioni corrispondenti alla forza ideale \mathbf{F}_{id} con linee tratteggiate. Il valore di picco di f_2 ideale è circa $2,2 f_{\max}$, con ciò si oltrepassa il limite superiore dell'asse y . Queste tensioni sono state calcolate offline basandosi sulla registrazione dei dati di posizione e sulle caratteristiche del muro virtuale. Per questo esse non superano il limite superiore f_{\max} . Le linee continue rappresentano l'attuale tensione dei cavi, e corrispondono alla forza \mathbf{F} raffigurata in figura 6.4 (linea rossa). E' possibile notare che, se f_2 raggiunge f_{\max} , tutte le tensioni vengono scalate, in questo modo f_2 rientra nei limiti imposti. Questo riduce gli effetti dannosi sulla forza esercitata. Infatti, la figura 6.4 (plot inferiore) mostra che gli angoli ϕ_{id} e ϕ sono idealmente coincidenti (salvo disturbi). Inoltre, il modulo della forza esercitata è minore rispetto a quella esercitata nel caso precedente (plot superiore).

Chiaramente tale tecnica è maggiormente adatta per un display aptico, fornendo all'utilizzatore un feedback di forza più naturale. La forza esercitata è ancora in grado di correggere la traiettoria del paziente, anche se il modulo è minore alla forza computata.

Conclusioni

Il lavoro di Tesi si inserisce nel contesto del progetto **Sophia3** (**String Operated Planar Haptic Interface for upper-Arm rehabilitation**), che ha come obiettivo finale lo sviluppo di una macchina ad interfaccia aptica a cavi per la neuroriabilitazione degli arti superiori.

In questo lavoro di Tesi è stato sviluppato il sistema del controllo del robot utilizzando l'ambiente di sviluppo Microsoft Visual Studio, il sistema operativo Windows CE ed il linguaggio di programmazione C++. E' stato sostituito il fieldbus CAN con quello EtheCAT al fine di avere maggiori prestazioni. Sono stati condotti i primi test sperimentali che hanno dimostrato la validità e la robustezza del sistema di controllo sia dal lato macchina che dal lato utente, riuscendo a fornire una sensazione reale di forza sull'end-effector.

Bibliografia

- [1] G. Rosati, R. Secoli, D. Zanotto, A. Rossi, and G. Boschetti. Planar robotic systems for upper-limb post-stroke rehabilitation. In Proceedings of the ASME International Mechanical Engineering Congress & Exposition IMECE 2008, Boston, MA, USA, Oct 31 - Nov 6 2008.
- [2] G. Rosati and D. Zanotto. A novel perspective in the design of cable-driven systems. In Proceedings of the ASME International Mechanical Engineering Congress & Exposition IMECE 2008, Boston, MA, USA, Oct 31 - Nov 6 2008.
- [3] G. Rosati, D. Zanotto, and S. K. Agrawal. On the design of adaptive cable-driven systems. *Journal of Mechanisms and Robotics*, 3(2):021004, 2011.
- [4] Y. Shen, H. Osumi, and T. Arai. Set of manipulating forces in wire driven systems. In Proc. of the IEEE International Conference on Intelligent Robots and Systems, pages 1626–1631, 1994.
- [5] P. Lafortune, M. Llibrec, and C. Reboulet. Design of a parallel wire-driven manipulator for wind tunnels. In Proceedings of the Workshop on Fundamental Issues and Future Research Directions for Parallel Mechanisms and Manipulators, October 3-4 2002.
- [6] M. Hiller, S. Fang, S. Mielczarek, R. Verhoeven, and D. Franitza. Design, analysis and realization of tendon-based parallel manipulators. *Mechanism and Machine Theory*, 40(4):429–445, 2005.
- [7] S. Krut, O. Company, and F. Pierrot. Velocity performance indices for parallel mechanisms with actuation redundancy. *Robotica*, 22(2):129 – 139, 2004.

-
- [8] M. Gouttefarde, S. Krut, O. Company, and F. Pierrot. Advances in Robot Kinematics: Analysis and Design, chapter On the Design of Fully Constrained Parallel Cable-Driven Robots, pages 71–78. 2008.
- [9] P. Bosscher and I. Ebert-Uphoff. Wrench-based analysis of cable-driven robots. In Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on, volume 5, pages 4950–4955 Vol.5, April-1 May 2004.
- [10] P. Bosscher and I. Ebert-Uphoff. Disturbance robustness measures for under-constrained cable-driven robots. In Proc. of the IEEE International Conference on Robotics and Automation (ICRA), pages 4205–4212, May 2006.
- [11] D. Surdilovic, R. Bernhardt, and T. Schmidt. STRING-MAN: A new wire robotic system for gait rehabilitation. In Proceedings of the IEEE 8th International Conference on Rehabilitation Robotics ICORR2003, Daejeon, Republic of Korea, Apr 2003.
- [12] X. Diao and O. Ma. Workspace determination of general 6-d.o.f. cable manipulators. Advanced Robotics, 22, March 2008.
- [13] D.Zanotto. Analisi del workspace e parziale redesign di una macchina planare a cavi per riabilitazione, Tesi di laurea, Padova, 2006-2007.
- [14] U.Pescante. Progettazione di macchina planare a cavi orientabile nello spazio di lavoro, Tesi di laurea, Padova, 2008-2009.
- [15] E.Mazzon. Realizzazione e controllo di robot planare a cavi per riabilitazione, Tesi di laurea, Padova, 2008-2009.
- [16] A. Preti. Controllo di servomotore elettrico tramite Ethercat, Tesi di laurea, Padova, 2009-2010.
- [17] EtherCAT Brochure, EtherCAT Technology Group, ETG Headquarters Ostendstraße 196 90482 Nuremberg Germany, Ottobre 2009.
- [18] TwinCAT I/O, Beckhoff Automation GmbH, Eiserstr. 5 33415 Verl Germany.

-
- [19] D. Nicola, “Bus di campo per l’automazione industriale,” Ph.D. dissertation, Università degli studi di Padova - Facoltà di ingegneria, 2007-2008.
- [20] E. T. Group, “Ethercat the ethernet fieldbus.” ETG Headquarters Ostendstraße 196 90482 Nuremberg Germany: EtherCAT Technology Group, Maggio 2005.
- [21] M. Rostan and J. E. Stubbs, “Ethercat - enabled advanced control architecture.” ETG Headquarters Ostendstraße 196 90482 Nuremberg Germany: EtherCAT Technology Group, Luglio 2010.
- [22] S. Pavlov and P. Belevsky, Windows Embedded CE 6.0 Fundamentals, Microsoft.
- [23] Dispense del corso di “Sistemi Operativi 2”, Prof. M. Moro, Università degli Studi di Padova. 2007.
- [24] P. Boltzern, R. Scattolini, N. Schiavoni Fondamenti di Controlli Automatici, II edizione, Mc Graw-Hill.
- [25] G. Marro, Controlli Automatici, V edizione, Zanichelli Editore, Bologna
- [26] K.J. Astrom, B. Wittenmark Computer Controlled systems: Theory and Design, II edition, Prentice Hall.
- [27] <http://sourceforge.net/apps/mediawiki/cppcheck>