



UNIVERSITÀ POLITECNICA DELLE MARCHE

Facoltà di Ingegneria

Corso di Laurea Magistrale in Ingegneria Informatica e dell'Automazione

“Studio e implementazione di algoritmi per la gestione del moto di un manipolatore industriale a 2 gradi di libertà nelle operazioni di pick and place”

“Study and design of algorithms for the motion of a 2 DOF industrial manipulator in pick and place operations”

Tesi di Laurea di:
Christian Centorame

Relatore:

Prof. Alessandro Freddi

Correlatore:

Dott. Ing. Ercole Radocchia

Anno Accademico 2021/2022

A te nonna Giulia,
ovunque tu sia a fare il tifo per me.

Indice

Indice	i
Introduzione	iii
1 I robot e la robotica	1
1.1 Cenni storici	2
1.2 Il robot come sistema	7
1.3 Robot industriali.....	8
1.3.1 Campi di applicazione nei processi manifatturieri	8
1.3.2 Manipolatori robotici.....	9
2 Richiami di Cinematica	12
2.1 Classificazione delle coppie cinematiche.....	13
2.1.1 Coppie cinematiche inferiori	13
2.1.2 Coppie cinematiche superiori	14
2.2 Catene cinematiche	15
2.3 Calcolo dei gradi di libertà dei meccanismi	16
3 Manipolatore parallelo PRRRP 2-DOF	19
3.1 Presentazione della linea di produzione	19
3.2 Analisi cinematica	20
3.2.1 Descrizione del meccanismo	21
3.2.2 Analisi di mobilità	22
3.3 Problema di cinematica inversa di posizione	23
3.4 Problema di cinematica inversa di velocità.....	25
3.5 Problema di cinematica diretta di posizione	26
3.6 Problema di cinematica diretta di velocità	28
3.7 Singolarità cinematiche	29
3.7.1 Condizioni di singolarità nei robot paralleli.....	30
3.7.2 Singolarità del manipolatore PRRRP 2-DOF.....	31
4 Pianificazione della traiettoria dei manipolatori	34
4.1 Algoritmo di pianificazione della traiettoria	34
4.2 Traiettorie nello spazio dei giunti e nello spazio operativo	35
4.3 Moto punto-punto.....	37
4.4 Moto attraverso una sequenza di punti.....	38

5 Hardware e dispositivi	42
5.1 Generazione e controllo del moto	42
5.1.1 Motori lineari.....	42
5.1.2 Azionamenti e PLC	43
5.1.3 Configurazione dei drive	44
5.1.4 Configurazione del PLC	46
5.2 Gestione della sicurezza.....	47
6 Implementazione software	49
6.1 Linguaggio e ambiente di programmazione.....	49
6.2 Preparazione della macchina all'utilizzo dei dispositivi.....	50
6.2.1 Dispositivi interessati	50
6.2.2 Passaggi di stato della macchina	51
6.2.3 Fase implementativa	54
6.3 Movimento del manipolatore PRRRP 2 DOF.....	57
6.3.1 Function Block di libreria.....	58
6.3.2 Algoritmo di gestione del moto del manipolatore	61
6.3.3 Vantaggi ottenibili	62
7 Fase di test	64
7.1 Operazione di pick and place	64
7.2 Confronto prestazionale codice preesistente-codice ottimizzato	66
8 Conclusioni e sviluppi futuri	69
Appendice A	71
Appendice B	74
Bibliografia	76

Introduzione

La trattazione di questa tesi verte sull'impiego dei manipolatori robotici in ambito industriale. In particolare, il focus è sul robot parallelo PRRRP (sigla che indica un meccanismo composto da due giunti prismatici e tre rotoidali) a due gradi di libertà. Tale sistema, nell'azienda in cui è stata condotta l'attività di tirocinio, viene adottato per effettuare operazioni di pick and place in un impianto per l'assemblaggio automatico di componenti in plastica.

La problematica principale affrontata nel presente elaborato è quella di realizzare un software ottimizzato, rispetto ad un codice preesistente, che permetta di gestire il moto del suddetto manipolatore durante l'esecuzione delle azioni di prelievo e di deposito dei pezzi. A tal fine vengono impiegate funzioni di libreria messe a disposizione dalla compagnia Bosch Rexroth, evidenziando i vantaggi che potrebbero derivare dal loro utilizzo.

Un altro problema fronteggiato in questa dissertazione è la scrittura di un programma che permetta di preparare il macchinario all'utilizzo dei dispositivi, in modo che l'operatore possa far partire il ciclo di lavoro. Questi ultimi sono quelli che intervengono nella stazione di lavoro del manipolatore.

Il Capitolo 1 presenta la disciplina della robotica, riportando definizioni a riguardo e i relativi cenni storici. Esso passa successivamente alla descrizione dei robot, indicando i sottosistemi che li caratterizzano. In particolare, sono presi in considerazione i manipolatori industriali e di questi vengono specificati i campi di applicazione nei processi manifatturieri e le tipologie in cui si articolano.

Il Capitolo 2 è incentrato sui richiami di cinematica e, nello specifico, sulla definizione dei concetti di coppie, catene e meccanismi. In seguito viene fornita la nozione di gradi di libertà, indicando formule per il calcolo di essi.

L'argomento centrale del Capitolo 3 è il manipolatore parallelo PRRRP a due gradi di libertà. Inizialmente viene presentato, per sommi capi, l'impianto di assemblaggio in cui esso è adoperato. Dopodiché vengono riportate la descrizione della struttura meccanica del robot in questione e l'analisi di mobilità di tale meccanismo. Successivamente sono determinate le equazioni di cinematica diretta ed inversa di posizione e di velocità e le condizioni di singolarità del sistema robotico.

Nel Capitolo 4 si parla della procedura di pianificazione della traiettoria dei manipolatori sia nello spazio operativo sia in quello dei giunti. In particolare, vengono indicate le principali tecniche che consentono di realizzare tale operazione.

Nel Capitolo 5 sono descritti i dispositivi hardware che entrano in gioco nella generazione e nel controllo del moto del robot e i relativi processi di configurazione.

Il Capitolo 6 presenta e modella le due problematiche, descritte in precedenza, che il lavoro di tesi si propone di risolvere. Inoltre esso fornisce una spiegazione delle soluzioni software adottate per far fronte a tali problemi.

I Capitoli 7 ed 8 si basano sull'esposizione e sulla discussione dei risultati provenienti dalla fase di test del codice scritto per la gestione del movimento del manipolatore.

1 I robot e la robotica

Fornire una definizione univoca del termine robot risulta quasi impossibile, dato che le tipologie di sistemi robotizzati esistite in passato ed esistenti al giorno d'oggi sono molteplici. In aggiunta, diversi sono gli ambiti nei quali questa particolare tecnologia ha dato il proprio contributo funzionale; il tutto sostituendo o assistendo l'uomo nei compiti più pericolosi e faticosi oppure migliorando e agevolando la sua vita quotidiana. Altrettanto multiforme è la robotica, disciplina ingegneristica adibita allo studio e allo sviluppo dei robot, che si articola in svariate categorie, alcune delle quali sono riportate in Tabella 1.1.

Tabella 1.1 - Principali categorie di robotica

Robotica	Descrizione
Industriale	Ha la finalità di automatizzare i processi produttivi. Consiste nell'uso di macchine robotiche nel contesto manifatturiero che prendono il posto dell'uomo nello svolgimento di lavori ripetitivi e logoranti. Fa parte di questa tipologia anche la robotica collaborativa che ha come protagonisti i 'cobot' (collaborative robot), i quali affiancano l'uomo nella realizzazione di compiti specifici, condividendo con lui lo spazio di lavoro in totale sicurezza (grazie all'utilizzo di sensori, sistemi di visione e di intelligenza artificiale)
Medica	Si occupa di progettazione e realizzazione di sistemi robotici per: <ul style="list-style-type: none">• la chirurgia, al fine di diminuire l'invasività delle operazioni garantendo una precisione maggiore e di consentire interventi 'da remoto';• la diagnosi, andando ad evitare o alleviare al paziente il possibile disagio fisico o psicologico che provocherebbe l'utilizzo degli strumenti diagnostici classici;• la riabilitazione;• l'assistenza rivolta a persone diversamente abili e anziani (ad opera dei 'care robot') [2]
Militare	Prevede l'utilizzo di macchine robotiche mobili aeree o terrestri che possono essere autonome o telecomandate (come ad esempio i droni o i robot artificieri). Esse vengono impiegate in operazioni militari che vanno dal trasporto, alla ricerca, al salvataggio, alla difesa ed anche all'attacco. Inoltre tali sistemi consentono di sostituire i soldati in ambienti e situazioni ostili che potrebbero compromettere la loro incolumità [3]
Spaziale	È contraddistinta da robot ai quali sono affidati precisi compiti da svolgere nello spazio come esplorazioni, manutenzione di stazioni spaziali e di satelliti e rimozione di detriti
Domestica	Rientra nell'ambito della domotica ed include tutti quei dispositivi automatici che consentono di migliorare l'efficienza e il comfort di un'abitazione, dando vita alle cosiddette 'Smart Home'
Educativa	È una delle applicazioni più interessanti della robotica. Basata sul metodo del learning by doing, mette a disposizione degli studenti strumenti robotici da impiegare come supporto didattico attraverso cui acquisire nuove competenze specifiche

Nonostante l'eterogeneità e la complessità che caratterizzano questo contesto, sono disponibili alcune definizioni ufficiali che tentano di spiegare in maniera generale e riassuntiva il significato del termine robot:

- *“Un robot è un manipolatore multifunzionale riprogrammabile, progettato per la movimentazione di materiali, pezzi, utensili o altri dispositivi specifici di produzione, attraverso movimenti variabili programmati per l'esecuzione di compiti diversificati”* (Robot Institute of America, 1979) [4];
- *“Un robot è un manipolatore controllato automaticamente, riprogrammabile, multi-funzione, a tre o più assi, che può essere fissato a terra o mobile, ed è utilizzato per applicazioni di automazione industriale”* (Standard ISO 8373) [5].

Non rientrano nelle due precedenti definizioni i robot definiti autonomi; si tratta di quelle macchine che, grazie all'utilizzo di sensori e di sistemi di apprendimento basati su Machine Learning e intelligenza artificiale, sono in grado di percepire l'ambientale circostante e di prendere decisioni in funzione della situazione in cui operano, tutto senza l'intervento da parte dell'uomo [1].

1.1 Cenni storici

Non vi è un'unica data che sancisca la nascita ufficiale della robotica; il suo avvento e il suo sviluppo sono determinati da molteplici episodi significativi in svariati ambiti oltre a quello tecnico-scientifico. Anche se si ritiene che i robot si siano affermati ed evoluti a cavallo tra la seconda e la terza rivoluzione industriale, il concetto di automa ha radici più antiche; se ne trova già traccia nell'ambito della mitologia greca: per esempio nel mito di Talos, il gigante artificiale di bronzo generato da Efesto e messo a guardia dei confini di Creta oppure nell'Iliade i tripodi automatici, portavivande, costruiti da Efesto per le divinità [6].

I primi 'prototipi' di robot semoventi e programmabili sono da attribuire ad Erone di Alessandria, matematico ed inventore della Grecia Antica (I secolo d.C.), celebre per aver ideato gli 'automata'; questi ultimi erano degli automi adibiti all'intrattenimento che si muovevano sul palco seguendo un programma prestabilito e senza l'intervento dell'uomo.

In base a quanto riportato nelle fonti scritte, tra i successori che hanno fornito un contributo rilevante alla robotica spicca, in epoca medievale, il matematico e ingegnere arabo Al-Jazari. Egli nel tredicesimo secolo si dedicò alla progettazione di una nave ricca di automi musicisti, i quali avevano la funzione di intrattenere la nobiltà durante i banchetti. La suddetta figura lavorò anche su meccanismi meno complessi, ma di utilità maggiore come la “fontana del pavone”: tirando la coda del volatile fuoriusciva acqua dal becco dell’animale e comparivano due figure di servitori, uno che offriva ceneri da utilizzare come sapone e l’altro che porgeva un asciugamano (Figura 1.1) [8].



Figura 1.1 – “Fontana del pavone” di Al Jazari

Rientra tra i padri della robotica anche il celebre Leonardo Da Vinci. Lo scienziato, nella sua raccolta di disegni e scritti denominata “Codice Atlantico”, illustra la figura di un cavaliere meccanico, il quale sarebbe in grado di alzarsi in piedi, muovere il capo, le braccia e la mascella (Figura 1.2).



Figura 1.2 - "Automa cavaliere" di Leonardo Da Vinci

Nel 1738 l'inventore ed artista francese Jacques de Vaucanson realizzò i primi automi funzionanti di cui si ha testimonianza; tra questi i più celebri sono il suonatore di flauto automatizzato e un'anatra meccanica in grado di muoversi, mangiare, bere e replicare il processo di digestione (riportata in Figura 1.3).

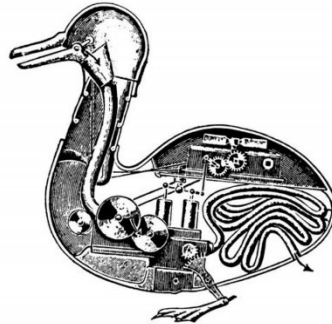


Figura 1.3 - "Anatra digeritrice" di Jacques de Vaucanson (da [44])

Nel corso dell'Ottocento furono poste le fondamenta che hanno consentito lo sviluppo dei robot moderni; infatti si pose l'attenzione su due questioni importanti, senza le quali la robotica si sarebbe fermata alle intuizioni di Erone. La prima era quella di far muovere gli automi più a lungo e in maniera controllata; questo requisito è stato garantito sfruttando la produzione e il controllo dell'energia elettrica. La seconda invece era la necessità di avere a disposizione un sistema che consentisse di programmare i robot, ovvero di tradurre i loro movimenti in una sequenza di simboli che fosse memorizzata in un supporto fisico. La risoluzione a tale problematica fu avviata dal matematico e filosofo britannico Charles Babbage, il quale dimostrò che fosse possibile realizzare una macchina in grado risolvere problemi di grande complessità, a patto di dividerli in passi più semplici.

A questo punto le basi per dare il via alla storia della robotica erano state poste, mancava un nome identificativo per gli automi. Il termine robot fu coniato nel 1920 dal drammaturgo ceco Karel Čapek nella sua opera fantascientifica "*R.U.R. (Rossum's Universal Robots)*" ad indicare il lavoro forzato. Egli utilizzò tale parola come un inglesismo di *robot*, che in lingua ceca significa letteralmente "schiavo, servizio della gleba", per riferirsi agli automi creati da Rossum (protagonista del dramma). Questi ultimi venivano intesi come creature con un comportamento condizionato dai sentimenti e costruite con materiale organico con la finalità di sostituire gli esseri umani in mansioni faticose e pericolose [4].

Lo scrittore sovietico Isaac Asimov, invece, introdusse per la prima volta la parola robotica; la utilizzò nel suo racconto “Runaround” del 1942, derivandola dal termine robot e attribuendole il significato di scienza devota allo studio degli automi. Introdusse inoltre le “tre leggi della robotica” che indicano sostanzialmente come un robot debba comportarsi nei confronti dell’essere umano; queste norme saranno poi adottate in futuro come punto di partenza verso un utilizzo etico della tecnologia [9]. Un’altra figura di rilievo in questo contesto è l’inventore e imprenditore statunitense George Devol, che nel 1954 brevettò il primo robot industriale della storia conosciuto con il nome di “Unimate”. Quest’ultimo è un braccio robotico adibito alla manipolazione di pezzi fabbricati attraverso la pressofusione e saldati sulle carrozzerie delle automobili. Il prototipo è mostrato in Figura 1.4.

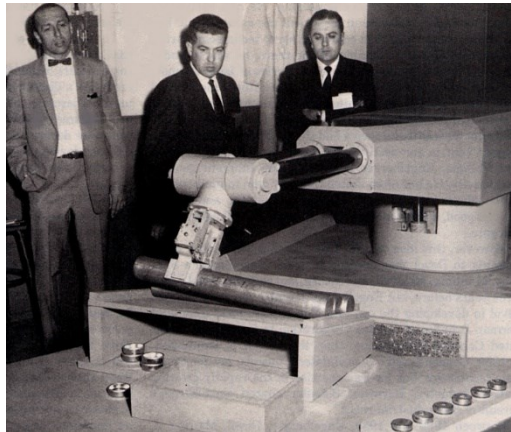


Figura 1.4 - Robot "Unimate" brevettato da George Devol (da [45])

Nel 1962 Devol, insieme all’ingegnere e imprenditore statunitense Joseph Engelberger, costituì la prima azienda di robotica industriale denominata “Unimation”; quest’ultima nel 1978 diede il via alla produzione del manipolatore “PUMA” (“*Programmable Universal Manipulation Arm*”), illustrato in Figura 1.5, la cui cinematica è alla base della realizzazione di molti dei robot industriali moderni.



Figura 1.5 - Robot PUMA (da [46])

Nel corso degli anni '70 la robotica subentrò e fu impiegata su larga scala nel settore secondario al fine di supportare la produzione industriale. Grazie all'avvento, negli anni '80, dei cosiddetti 'robot di servizio' come quelli artificieri, per le missioni nello spazio e sottomarine, per la chirurgia e per l'assistenza, è stata intrapresa l'espansione della robotica al di fuori del contesto manifatturiero.

Nell'ultima decade del XX secolo si sono iniziati a creare i robot-giocattolo; tra questi è da ricordare il cane robotico "Aibo" (riportato in Figura 1.6) presentato dalla Sony per la prima volta nel 1999 e programmato al fine di camminare, dormire ed 'intrattenere'.



Figura 1.6 - Cane robotico "Aibo" della Sony (da [47])

Nei primi anni 2000 si iniziarono a diffondere ampiamente sul mercato i velivoli robotici noti come droni, il cui utilizzo era destinato inizialmente a missioni di tipo militare [7]. Nasce in questa maniera la robotica moderna che al giorno d'oggi è in continua evoluzione e sta conquistando sempre più settori. Grazie alla costante innovazione meccanica e tecnologica, i robot stanno diventando mano a mano maggiormente efficienti ed autonomi; essi al tempo stesso stanno trasformando in realtà quell'immaginario 'fantasioso' di interazione uomo-automa che caratterizzava l'antichità.

1.2 Il robot come sistema

Ogni robot in generale è un sistema costituito da cinque unità funzionali basilari (o sottosistemi).

- *Sottosistema meccanico.* È la struttura fisica del sistema robotico stesso che è sede degli altri organi citati di seguito.
- *Sottosistema sensoriale.* Consiste in un insieme di sensori che si dividono in:
 - propriocettivi: consentono di effettuare misure nel sistema di riferimento solidale al robot e quindi di valutarne lo stato interno;
 - eterocettivi: permettono di eseguire misurazioni nel sistema di riferimento inerziale e quindi di acquisire informazioni in merito all'ambiente esterno al sistema robotico [10].
- *Sottosistema elettrico.* Questo costituente include gli azionamenti (di conseguenza motori ed alimentatori) che consentono di movimentare la struttura meccanica e i cavi elettrici per comandarli e fornire energia ad essi [11].
- *Sottosistema di controllo.* Esso è caratterizzato da attuatori (elettrici, idraulici e pneumatici) e dai relativi algoritmi di controllo che servono a pilotarli. Questi ultimi attuano il robot affinché compia i task che gli vengono assegnati, rispettando i vincoli e le specifiche assegnate.
- *Sottosistema di governo.* È l'organo che, servendosi della descrizione della struttura meccanica e dei dati provenienti dai sensori, permette di programmare e gestire le attività che la macchina robotica deve svolgere. Tale unità è formata dai dispositivi hardware (microprocessori, memorie, etc.) e dai software (programmi applicativi) necessari al funzionamento del sistema [12].

In Figura 1.7 è possibile osservare come si relazionano fra di loro i sottosistemi sopracitati.

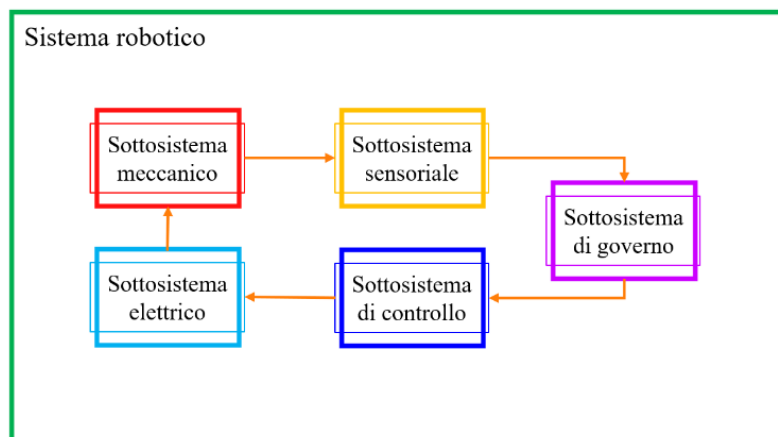


Figura 1.7 - Sistema robotico e relativi sottosistemi

1.3 Robot industriali

I robot hanno fatto il loro ingresso in ambito industriale a partire dagli anni '60; da quel momento in poi si sono rivelati come un componente di fondamentale importanza per la creazione di sistemi automatizzati di produzione. Le macchine robotiche, infatti, hanno portato a numerosi vantaggi tra cui:

- aumento della produttività;
- riduzione dei costi di produzione;
- miglioramento degli standard qualitativi dei prodotti;
- possibilità di eliminare attività alienanti o pericolose.

Un robot, impiegato in questo settore, rientra nella tipologia di automazione industriale definita 'programmabile', la quale è adibita alla produzione di piccoli e medi lotti di manufatti dalle caratteristiche variabili.

1.3.1 Campi di applicazione nei processi manifatturieri

Nel contesto manifatturiero i compiti che vengono solitamente affidati ad un sistema robotico appartengono ai seguenti campi:

- trasporto;
- manipolazione;
- misura.

Della prima categoria fanno parte quei robot che si occupano della movimentazione di pezzi tra le macchine operatrici e macchine utensili. Le tipiche mansioni che essi svolgono in questo particolare ambito sono:

- pallettizzazione e logistica;
- selezione e smistamento di parti;
- carico e scarico dei magazzini;
- confezionamento delle merci.

Il contesto della manipolazione si divide in due sottocategorie, come di seguito riportato.

- *Lavorazione*. Essa comprende tutte quelle macchine robotiche che lavorano sui prodotti mediante un attrezzo. Alcune tipiche operazioni sono:
 - saldatura a punti e ad arco;
 - taglio;
 - verniciatura;
 - la fresatura e la foratura.
- *Assemblaggio*. Quest'ultimo prevede le seguenti funzioni:
 - montaggio di schede elettroniche;
 - avvitatura;
 - cablaggio [13].

Infine le applicazioni robotiche nel campo della misura operano una verifica dei prodotti in attività come: rilevamento di profili, collaudo dimensionale e individuazione di difetti di fabbricazione [4].

1.3.2 Manipolatori robotici

In ambito industriale con il termine manipolatore si indica la struttura meccanica del sistema robotico stesso. Esso è caratterizzato da un insieme di corpi rigidi ('link') connessi fra di loro attraverso determinati snodi detti giunti. Vi sono poi altri due importanti costituenti: il primo è la base, la quale può essere fissata nella postazione di lavoro oppure ancorata ad una piattaforma mobile; il secondo è l'organo terminale (o 'end-effector'), ovvero un utensile, una pinza o un dispositivo specializzato che svolge la mansione per cui il robot è impiegato [16].

Per quanto concerne l'aspetto realizzativo, i robot industriali si classificano in base alla conformazione geometrica che viene scelta per la loro struttura. In questo caso, infatti, si effettua la distinzione tra i seguenti manipolatori:

- cartesiano o a portale;
- cilindrico;
- SCARA;
- polare o sferico;
- articolato o antropomorfo.

Le configurazioni standard sopra citate sono illustrate in Figura 1.8 [14]. La presente immagine indica, per ciascuna categoria di robot, i giunti che la caratterizzano. Questi ultimi vengono rappresentati con le lettere P o R, ad indicare rispettivamente un giunto di tipo prismatico o uno di tipo rotoidale. Queste due tipologie di snodi sono le più impiegate a livello industriale e verranno presentate con maggior dettaglio nel capitolo successivo.

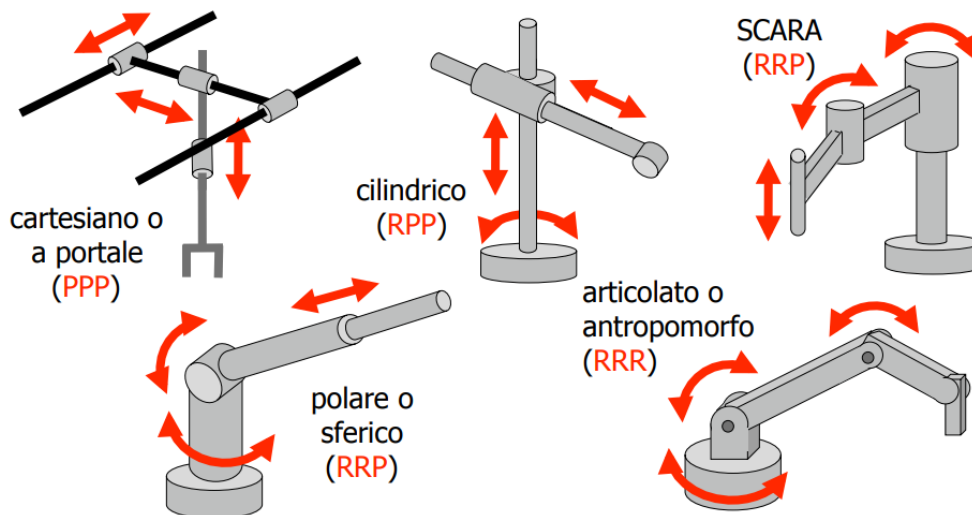


Figura 1.8 - Configurazioni geometriche dei manipolatori

I manipolatori robotici sono catalogati ulteriormente come segue.

- *Seriali*. Essi sono costituiti da un insieme di bracci connessi in serie, attraverso giunti attuati, a formare una catena (definita aperta). La loro struttura ha un estremo fissato a telaio e l'altro estremo libero di muoversi nel piano o nello spazio.
- *Paralleli*. I link formano uno o più percorsi chiusi indipendenti tra loro che hanno un'estremità ancorata alla base e l'altra all'end-effector. In particolare, il telaio e l'organo terminale sono collegati da catene seriali definite 'gambe', ciascuna delle quali è controllata da uno specifico attuatore.
- *Ibridi*. In questo caso i bracci generano sia 'circuiti' chiusi che aperti.

In Tabella 1.2 sono riportati i vantaggi e gli svantaggi dei robot seriali e di quelli paralleli [15].

Tabella 1.2 – Confronto tra manipolatori seriali e paralleli

	Vantaggi	Svantaggi
Manipolatori seriali	<ul style="list-style-type: none"> • spazio di lavoro ampio • destrezza elevata • semplicità nel controllo 	<ul style="list-style-type: none"> • capacità di carico bassa • precisione bassa dovuta alla poca rigidità della struttura
Manipolatori paralleli	<ul style="list-style-type: none"> • accelerazioni elevate dell'end-effector dovute alle basse masse in movimento • capacità di carico elevata • robustezza elevata • accuratezza elevata • costruzione meccanica semplice e modulare 	<ul style="list-style-type: none"> • difficoltà nel controllo • spazio di lavoro limitato e con una geometria complessa • difficoltà nella progettazione • sensibilità alla temperatura • destrezza bassa

2 Richiami di Cinematica

La cinematica è quella branca della fisica che studia il moto dei corpi a prescindere dalle cause che lo generano o lo modificano (ovvero le forze). Tale disciplina permette di fornire un'analisi descrittiva del movimento dei sistemi meccanici; questi ultimi sono costituiti da un certo numero di corpi rigidi (link) interconnessi fra loro attraverso opportuni legami denominati vincoli. In questo modo la mobilità di ciascun link va ad influenzare quella dell'altro e di conseguenza ne risultano limitati i gradi di libertà di ciascun membro e dell'intero sistema.

I gradi di libertà (Degrees of freedom-DOF) si definiscono, in modo generale, come le possibilità di movimento dei corpi rigidi espresse in funzione del contesto in cui essi si trovano. Un membro nello spazio è caratterizzato da sei gradi di libertà (tre traslazioni e tre rotazioni possibili), mentre nel piano ha tre gradi di libertà (due traslazioni e una rotazione) come mostrato in Figura 2.1.

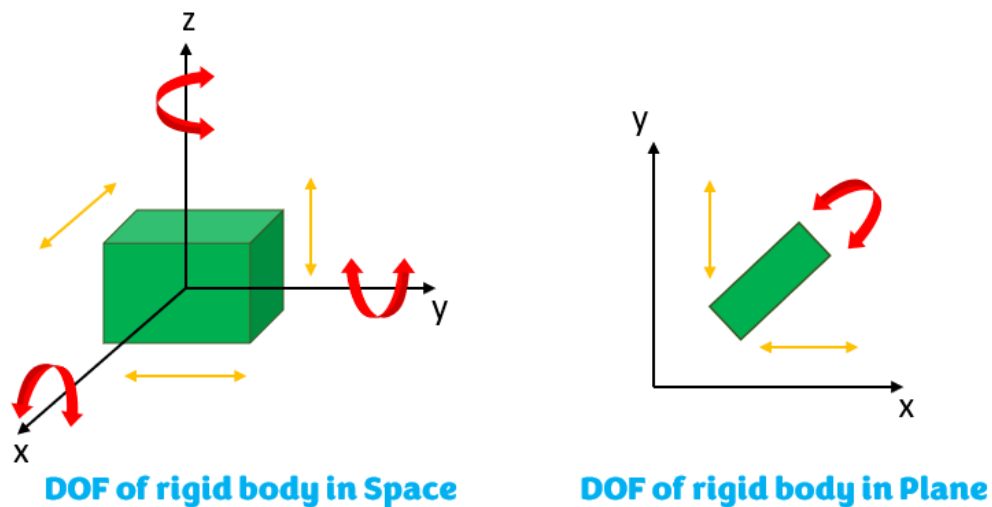


Figura 2.1 - Gradi di libertà di un corpo rigido nello spazio e nel piano

L'insieme di due corpi rigidi adiacenti e connessi viene definito coppia; nel caso in cui vi sia moto relativo tra di essi si parla di coppia cinematica. Lo spostamento relativo fra i membri dipende dalla forma delle superfici a contatto nel movimento che sono chiamate superfici coniugate; in particolare si ha che uno stesso comportamento cinematico può essere realizzato con diverse coppie di superfici coniugate.

Da un punto di vista pratico le coppie possono essere realizzate attraverso:

- *accoppiamenti di forma*, ovvero la forma dei due link permette il collegamento degli stessi;
- *accoppiamenti di forza*, ovvero il contatto tra i due corpi è mantenuto attraverso opportune forze e momenti.

Si definisce infine classe di una coppia cinematica il numero di gradi di libertà residui che essa consente nel moto relativo tra i due membri che la costituiscono.

2.1 Classificazione delle coppie cinematiche

La classificazione principale delle coppie cinematiche è stata ideata dall'ingegnere tedesco Franz Reuleaux che le distingue in:

- inferiori;
- superiori.

2.1.1 Coppie cinematiche inferiori

Le coppie cinematiche inferiori sono gli accoppiamenti che si realizzano tramite contatti di superficie. In questo caso le superfici coniugate sono rigide e devono combaciare in maniera perfetta. Rientrano in questa categoria le coppie che seguono.

- *Prismatica (P)*, che impone un moto relativo che consiste in una traslazione lungo un asse comune.
- *Rotoidale (R)*, che vincola il moto tra i due corpi ad una rotazione attorno ad un asse comune.
- *Elicoidale (E)*, che limita il moto relativo ad una traslazione lungo un asse, accompagnata da una rotazione attorno allo stesso. I due tipi di movimento sono legati fra loro; infatti, se viene impartita una rotazione, si avrà anche una conseguente traslazione e viceversa.
- *Piana (F)*, che consente un movimento traslatorio in un piano, combinato ad una rotazione attorno ad un asse.

- *Cilindrica (C)*, che permette un moto rotatorio intorno ad un asse e una traslazione lungo il medesimo asse; tuttavia, a differenza della coppia elicoidale, i due movimenti non sono legati fra loro, ma sono indipendenti.
- *Sferica (G)*, che è costituita da una sfera piccola alloggiata all'interno di una calotta sferica più grande. La sfera interna può orientarsi liberamente e lo fa attraverso un moto rotatorio attorno a tre assi, i quali si incontrano in un punto materiale fisso (il centro del moto sferico).

Le coppie prismatiche, rotoidali ed elicoidali, che lasciano soltanto un grado di libertà al moto relativo, vengono definite coppie cinematiche elementari. Le coppie piana e sferica invece conservano tre gradi di libertà, mentre quella cilindrica ne consente due. In Figura 2.2 sono riportate graficamente le coppie cinematiche inferiori definite in precedenza.

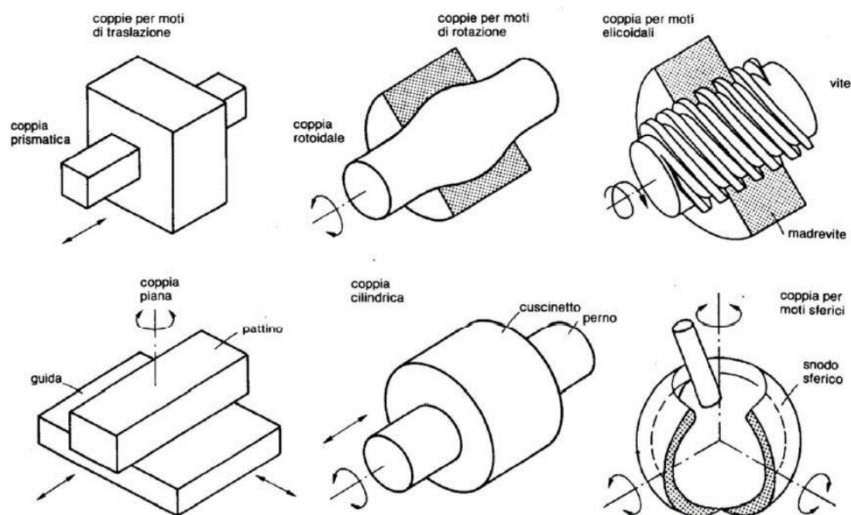


Figura 2.2 - Coppie cinematiche inferiori (da [17])

2.1.2 Coppie cinematiche superiori

Le coppie cinematiche superiori non vengono realizzate mediante contatti superficiali, ma attraverso:

- contatti lineari, nei quali i due corpi rigidi aderiscono lungo un segmento;
- contatti puntiformi, in cui i due link sono tangenti reciprocamente in un punto.

Rientrano in questa categoria le seguenti coppie:

- coppia a *camma piana* (Ca), che limita il movimento tra i link ad una traslazione tangenziale e ad una rotazione attorno ad un asse perpendicolare a quello del moto traslatorio (Figura 2.3);
- coppia *sferica guidata o universale* (U), che permette due rotazioni attorno a due assi che sono perpendicolari tra loro (Figura 2.4).

Per entrambi i tipi di accoppiamento il moto relativo dei membri è a due gradi di libertà.

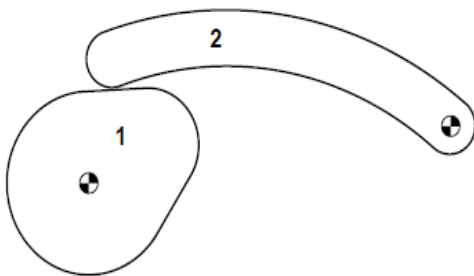


Figura 2.3 - Coppia a camma piana (da [17])

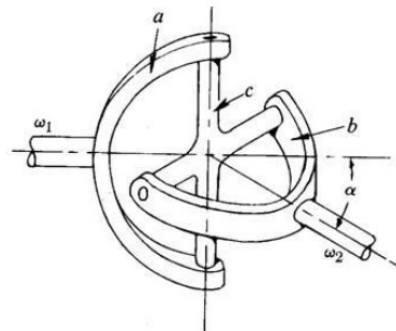


Figura 2.4 - Coppia Universale (da [17])

2.2 Catene cinematiche

Le catene cinematiche sono insiemi di membri connessi tra loro tramite coppie cinematiche. Una catena viene detta:

- chiusa, se si formano percorsi chiusi tra i corpi rigidi che la compongono;
- aperta, se ciascuno dei link è caratterizzato soltanto da due accoppiamenti, uno con il membro che lo precede e l'altro con quello che lo segue;
- semplice, se ciascun membro presenta uno o massimo due accoppiamenti;
- composta, se presenta corpi con più di due accoppiamenti.

Una catena cinematica che rispetto ad un sistema di riferimento assoluto abbia almeno un membro fisso (denominato telaio), viene definita meccanismo.

Nel momento in cui si effettua lo studio delle catene cinematiche si tende ad astrarre rispetto alla loro effettiva realizzazione; a tal fine si utilizza lo schema strutturale, ovvero una rappresentazione topologica che va a descrivere la struttura della catena stessa e che permette di comprendere quanti sono i gradi di libertà che la

caratterizzano. Questa schematizzazione contiene informazioni riguardo al numero di corpi rigidi che compongono il meccanismo (rappresentati mediante una linea), come questi ultimi sono connessi tra di loro e le coppie cinematiche che entrano in gioco (raffigurate con cerchi affiancati dai relativi simboli che le rappresentano). In Figura 2.5 ed in Figura 2.6 sono riportati esempi di schema strutturale di due sistemi meccanici differenti.

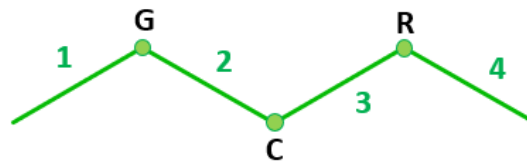


Figura 2.5 - Schema strutturale di una catena cinematica aperta semplice

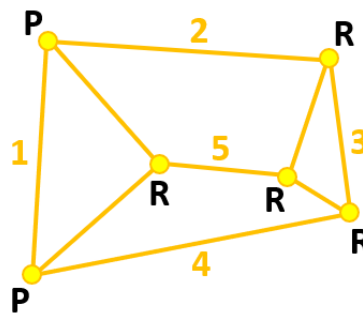


Figura 2.6 - Schema strutturale di una catena cinematica chiusa composta

2.3 Calcolo dei gradi di libertà dei meccanismi

Il numero di gradi di libertà di un meccanismo è il minimo numero intero (n) di parametri indipendenti necessari a identificare univocamente una generica posa (posizione e orientamento) che esso assume nel piano o nello spazio. Per effettuare il calcolo di tale numero si ricorre all'utilizzo delle cosiddette equazioni di struttura. Di queste ultime ne esistono due tipologie:

- *Formula di Kutzbach*, che permette di calcolare il numero di gradi di libertà di una catena cinematica nello spazio:

$$n = 6(m - 1) - 5c_1 - 4c_2 - 3c_3 - 2c_4 - c_5 \quad (1.1)$$

- *Formula di Grübler*, che consente di ricavare il numero di gradi di libertà della catena cinematica nel piano:

$$n = 3(m - 1) - 2c_1 - c_2 \quad (1.2)$$

In entrambe le equazioni (1.1) e (1.2) il parametro m rappresenta il numero di link che compongono il meccanismo, incluso il telaio; mentre c_i è il numero di giunti cinematici di classe i che compaiono nel sistema meccanico. In particolare, c_1 indica il numero di coppie di classe 1 (che permettono un solo grado di libertà nel movimento relativo tra corpi) e c_2 quello delle coppie di classe 2 (che lasciano due DOF nel moto tra i membri).

In base al numero di gradi di libertà si parla di:

- meccanismo (se $n \geq 1$) \rightarrow il movimento è consentito in questo caso;
- struttura isostatica (se $n = 0$) \rightarrow la catena cinematica non si muove dal momento che è costituita da tanti vincoli quanti sono i DOF potenziali;
- struttura iperstatica (se $n < 0$) \rightarrow anche in questo caso non c'è moto perché i vincoli sono in numero maggiore rispetto ai gradi di libertà potenziali [17].

I DOF possono essere anche riferiti al task che il sistema meccanico deve eseguire; in tal caso essi rappresentano i gradi di libertà del compito e si indicano con la lettera r (pertanto è possibile che $r \neq n$). Inoltre, per un meccanismo si definisce la somma δ_m dei gradi di mobilità, uno per ogni giunto della struttura meccanica [18].

Facendo riferimento ai manipolatori robotici descritti nel capitolo 1, si possono indicare i seguenti parametri:

- $M \rightarrow$ dimensione dello spazio operativo, ovvero lo spazio del vettore x che specifica la posa (posizione e orientamento) dell'end-effector del robot;
- $N \rightarrow$ dimensione dello spazio dei giunti (o delle configurazioni), ossia lo spazio del vettore q delle variabili di giunto che indicano le posizioni degli snodi (in termini di angoli e/o elongazioni). Tale quantità coincide con il numero n di gradi di libertà del sistema [19].

In base ai valori assunti delle misure appena definite è possibile che si verifichi uno dei seguenti casi.

- *Ridondanza cinematica intrinseca*, che si ottiene nel momento in cui la dimensione dello spazio operativo è minore di quella dello spazio dei giunti ($M < N$).
- *Ridondanza cinematica funzionale*, che si ha quando la dimensione dello spazio operativo è maggiore del numero di gradi di libertà del compito ($M > r$). La presente casistica, quindi, rappresenta un concetto associato al tipo di task richiesto al sistema robotico.

Se non si verifica una delle due situazioni sopra citate si parla allora di manipolatore non ridondante [20]. In generale la ridondanza viene definita come una proprietà che permette di conferire destrezza e versatilità al robot, in modo che esso possa evitare ostacoli nello spazio operativo o condizioni di singolarità nello spazio dei giunti. Inoltre essa viene impiegata anche per minimizzare il consumo di energia, aumentare l'affidabilità rispetto ai guasti, ottimizzare le coppie richieste ai giunti e distribuire in maniera uniforme/limitare le velocità di questi ultimi. Tuttavia questa caratteristica comporta per i robot industriali una maggiore complessità strutturale (sia meccanica che di attuazione) e algoritmi di cinematica inversa e di controllo maggiormente complessi ([16], [21]).

3 Manipolatore parallelo PRRRP 2-DOF

Il manipolatore PRRRP, oggetto di studio del lavoro di tesi, è un robot industriale parallelo a base fissa, caratterizzato da due gradi di libertà: uno traslazionale verticale ed uno longitudinale. I movimenti possibili avvengono in uno spazio bidimensionale, pertanto il presente sistema robotico viene definito planare, a differenza di quelli detti spaziali che si muovono in tre dimensioni.

Nell'azienda in cui è stata condotta l'attività di tirocinio, tale tipologia di robot è impiegata per operazioni di pick-and-place; in particolare, nel caso in esame, in un macchinario dedicato all'assemblaggio automatico di quattro componenti per la produzione finale di 12 assemblati per ciclo.

3.1 Presentazione della linea di produzione

L'impianto appena citato consiste in una macchina che assembla in maniera automatica. Essa opera con una cadenza di 50 cicli al minuto, producendo 600 pz/min. In particolare, tale linea di produzione è formata da:

1. 5 manipolatori paralleli PRRRP (2 DOF) che, attraverso operazioni di pick & place, assemblano in un processo sequenziale tutti i componenti del prodotto finale riforniti da alimentatori esterni;
2. un manipolatore cartesiano che preleva pezzi a campione dalla linea per inserirli in una stazione che effettua un controllo di tenuta;
3. stazioni intermedie che completano l'assemblaggio con funzioni suppletive e di controllo, come il controllo di altezza, di forma;
4. stazioni con sistemi di visione adibiti al controllo qualità;
5. una stazione che scarta i prodotti classificati come non conformi;
6. una stazione finale per la pallettizzazione degli assemblati ultimati e classificati come idonei.

Non possono essere fornite maggiori informazioni sulla struttura e sul funzionamento del macchinario per questioni di riservatezza.

3.2 Analisi cinematica

Il manipolatore PRRRP, come mostrato in Figura 3.1, è costituito da cinque corpi rigidi:

- il telaio, ovvero la base fissa (link 1);
- due guide verticali (link 2 e 3), collegate al membro 1, che distano tra di loro di una quantità L_1 ;
- due aste (link 4 e 5) di lunghezza L_2 accoppiate ad un'estremità con una delle guide sopra citate e all'altra, quella comune, con l'end-effector.

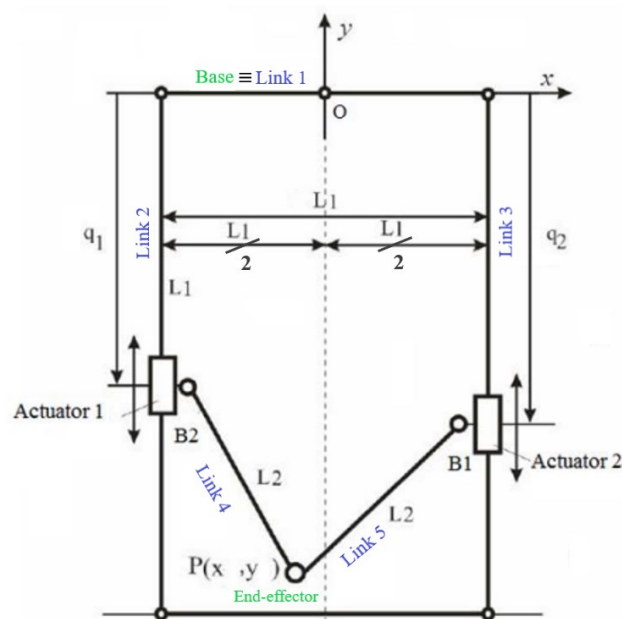


Figura 3.1 – Rappresentazione cinematica del manipolatore PRRRP 2-DOF (da [23])

La struttura meccanica del robot è caratterizzata da cinque coppie cinematiche inferiori elementari: due prismatiche e tre rotoidali. Infatti l'acronimo PRRRP, che fa da nominativo al sistema robotico, è una successione di simboli, ciascuno dei quali rappresenta la prima lettera del nome di un giunto: nello specifico P sta per prismatic joint e R per revolute joint (come indicato nel capitolo 2).

Il manipolatore in questione, essendo parallelo, è costituito da 'gambe' seriali che connettono la base all'organo terminale; in particolare, esso ne ha a disposizione due. Queste ultime sono catene cinematiche aperte PRR con un estremo comune rappresentato dalla seconda coppia rotoidale della gamba [22].

3.2.1 Descrizione del meccanismo

Un'astrazione utile allo studio del presente meccanismo robotico è lo schema strutturale riportato in Figura 3.2. Dall'analisi di quest'ultimo si evince che la struttura del PRRRP è una catena cinematica chiusa e semplice.

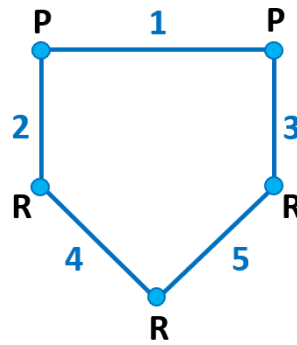


Figura 3.2- Schema strutturale del manipolatore PRRRP

Il manipolatore in esame inoltre è planare, ovvero effettua movimenti che avvengono sul piano (in due dimensioni); pertanto il calcolo dei gradi di libertà di esso, come specificato nel capitolo precedente, si realizza attraverso la formula (1.2) (di Grübler). Essendo tale macchina costituita da cinque membri ($m = 5$) e da cinque coppie cinematiche di classe 1 ($c_1 = 5$), si ha che il numero dei gradi di libertà è dato da:

$$n = 3(5 - 1) - 2 * 5 = 12 - 10 = 2 \quad (3.1)$$

I DOF di un meccanismo coincidono con le variabili indipendenti di giunto del sistema. In questo caso i parametri liberi sono q_1 e q_2 , i quali rappresentano le posizioni dei due giunti prismatici attuati rispetto ad un sistema di riferimento fisso O-xy solidale con il link 1 (come mostrato in Figura 3.1); in particolare queste grandezze descrivono lo spostamento lineare rispetto al telaio degli attuatori associati a tali coppie cinematiche. Di conseguenza il vettore delle variabili di giunto, che permette di identificare in maniera univoca una configurazione dell'intero manipolatore, è il seguente: $q = [q_1 \ q_2]^T$.

Come già specificato nel capitolo precedente, la posa dell'end-effector del robot, invece, viene espressa attraverso un vettore $x = [p \ \varphi]^T$. Nello specifico le componenti p e φ sono i vettori che indicano rispettivamente la posizione e l'orientamento dell'organo terminale del manipolatore rispetto al sistema di riferimento fisso

collocato sulla base della macchina; la loro dimensione dipende dal tipo di moto (spaziale o planare) che il manipolatore è in grado di compiere e dal task che quest'ultimo deve eseguire [22].

3.2.2 Analisi di mobilità

L'analisi di mobilità di un sistema meccanico permette di definire qual è il campo ammissibile per i suoi movimenti e quindi di specificare quali sono i punti raggiungibili dai membri che lo costituiscono. Essa consente di determinare, per il meccanismo in esame, una particolare regione che dipende dalla struttura, dalla geometria e dalla configurazione della catena cinematica: lo spazio di lavoro (workspace) [17]. Quest'ultimo, nel caso di un manipolatore, viene indicato come l'area nel piano o nello spazio tracciata dall'origine di una terna solidale con l'end-effector quando ai giunti vengono fatti eseguire tutti i moti possibili [19].

La struttura del workspace è influenzata dalle limitazioni di tipo meccanico imposte dai giunti. Nel caso particolare del manipolatore PRRRP questi limiti consistono in angoli massimi di rotazione propri delle coppie rotoidali e in distanze massime e minime che possono essere raggiunte dagli attuatori delle coppie prismatiche rispetto alla base fissa.

Lo spazio di lavoro è una delle proprietà cinematiche più rilevanti di un sistema robotico in quanto ha una grande influenza sulla sua progettazione, sul suo posizionamento in ambito industriale e sul suo funzionamento. Pertanto, al fine di ricavare in maniera precisa questa regione, sono state ideate tecniche analitiche e grafiche [22].

In Figura 3.3 è riportato il workspace del manipolatore PRRRP ricavato utilizzando un algoritmo di tipo numerico. In questo caso esso rappresenta l'area nel piano in cui l'organo terminale può essere controllato e spostato in modo continuativo senza incontrare ostacoli ed evitando le possibili condizioni di singolarità [23].

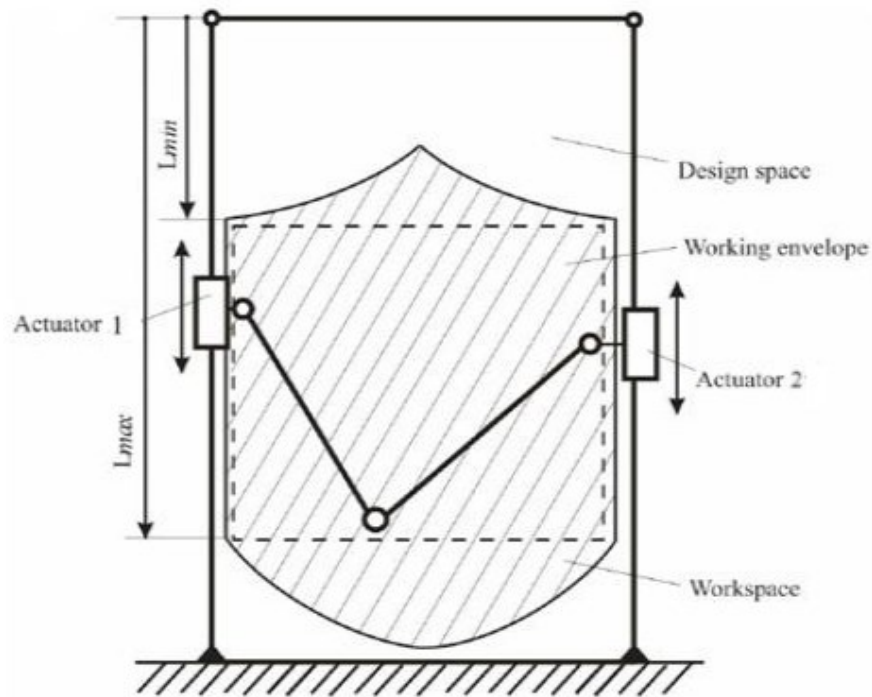


Figura 3.3 - Spazio di lavoro del manipolatore parallelo PRRRP 2-DOF (da [23])

3.3 Problema di cinematica inversa di posizione

Risolvere il problema cinematico inverso di posizione significa determinare i valori delle variabili di giunto che corrispondono ad una determinata posa desiderata per l'end-effector. La soluzione a tale problematica riveste un ruolo di fondamentale importanza; infatti essa permette di trasformare le specifiche di movimento richieste per l'organo terminale nello spazio operativo, nei corrispondenti moti nello spazio dei giunti che permettono di realizzare lo spostamento desiderato del manipolatore [24].

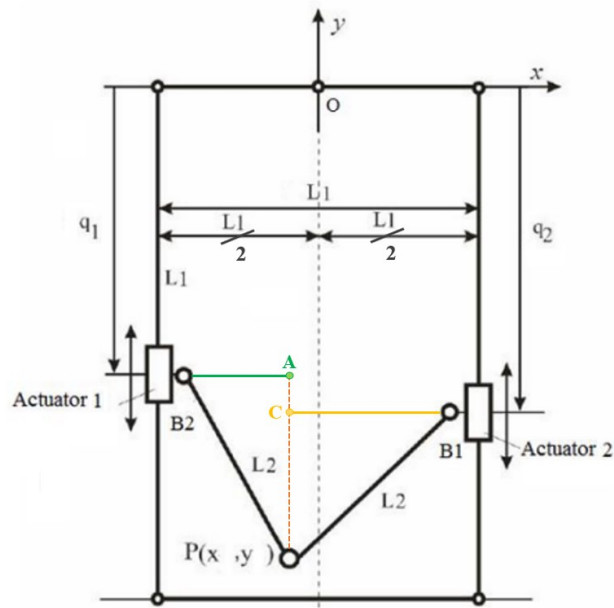


Figura 3.4 – Rappresentazione geometrica del manipolatore PRRRP

Facendo riferimento alla Figura 3.4, se si applica il Teorema di Pitagora ai triangoli rettangoli APB_2 e CPB_1 si riescono a ricavare le equazioni che seguono:

$$\left(x + \frac{L_1}{2}\right)^2 + (y - q_1)^2 = L_2^2 \quad (3.2)$$

$$\left(x - \frac{L_1}{2}\right)^2 + (y - q_2)^2 = L_2^2 \quad (3.3)$$

Se si manipolano le relazioni appena calcolate esprimendole rispetto alle variabili q_1 e q_2 , si ottengono le formule che descrivono la cinematica inversa di posizione del manipolatore parallelo PRRRP:

$$q_1 = y + \sqrt{L_2^2 - \left(x + \frac{L_1}{2}\right)^2} \quad (3.4)$$

$$q_2 = y + \sqrt{L_2^2 - \left(x - \frac{L_1}{2}\right)^2} \quad (3.5)$$

3.4 Problema di cinematica inversa di velocità

Il problema cinematico inverso di velocità (o di cinematica differenziale inversa) afferisce al calcolo delle velocità da imprimere ai giunti al fine di garantire una velocità desiderata di movimento per l'end-effector [24].

Un passaggio fondamentale per determinare le equazioni di cinematica inversa di velocità è quello di derivare le formule (3.2) e (3.3) rispetto al tempo. Effettuando questo calcolo ed opportune rielaborazioni si riesce a ricavare l'espressione in forma matriciale riportata di seguito:

$$\begin{bmatrix} (y - q_1) & 0 \\ 0 & (y - q_2) \end{bmatrix} \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \end{pmatrix} = \begin{bmatrix} \left(x + \frac{L_1}{2}\right) & (y - q_1) \\ \left(x - \frac{L_1}{2}\right) & (y - q_2) \end{bmatrix} \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} \quad (3.6)$$

A questo punto si definiscono due matrici quadrate A e B come segue:

$$A = \begin{bmatrix} (y - q_1) & 0 \\ 0 & (y - q_2) \end{bmatrix} \quad B = \begin{bmatrix} \left(x + \frac{L_1}{2}\right) & (y - q_1) \\ \left(x - \frac{L_1}{2}\right) & (y - q_2) \end{bmatrix} \quad (3.7)$$

L'introduzione di questi due termini consente di riscrivere la formula (3.6) in:

$$A \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \end{pmatrix} = B \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} \quad (3.8)$$

Successivamente bisogna valutare se la matrice quadrata A è non singolare, ovvero invertibile: quindi, per definizione, il suo determinante deve essere diverso da zero ($\det(A) \neq 0$). Se vale questa proprietà allora l'equazione (3.8) può essere scritta come:

$$\begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \end{pmatrix} = A^{-1}B \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} \quad (3.9)$$

Il prodotto $A^{-1}B$ viene comunemente indicato come J^{-1} . Quest'ultima è l'inversa della matrice J detta jacobiana o Jacobiano analitico.

$$J^{-1} = A^{-1}B = \begin{bmatrix} \frac{(x + \frac{L_1}{2})}{(y - q_1)} & 1 \\ \frac{(x - \frac{L_1}{2})}{(y - q_2)} & 1 \end{bmatrix} = \begin{bmatrix} -\frac{(x + \frac{L_1}{2})}{\sqrt{L_2^2 - (x + \frac{L_1}{2})^2}} & 1 \\ -\frac{(x - \frac{L_1}{2})}{\sqrt{L_2^2 - (x - \frac{L_1}{2})^2}} & 1 \end{bmatrix} \quad (3.10)$$

Se si ridefinisce la formula (3.9) prendendo in considerazione il nuovo parametro appena introdotto, si ottiene l'equazione che esprime la cinematica inversa di velocità:

$$\dot{Q} \doteq \begin{pmatrix} \dot{q}_1 \\ \dot{q}_2 \end{pmatrix} = J^{-1} \begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} \doteq J^{-1} \dot{P} \quad (3.11)$$

Essa rappresenta la relazione che sussiste tra le velocità dei giunti (\dot{Q}) e la velocità traslazionale dell'end-effector (\dot{P}) [22]. Dipendendo tale uguaglianza dall'inversa della matrice jacobiana, si ha che il problema di cinematica differenziale inversa ha soluzione se e solo se J è invertibile (ha rango pieno) e quindi se $\det(J) \neq 0$ [24].

3.5 Problema di cinematica diretta di posizione

Il problema cinematico diretto di posizione si propone di individuare la postura dell'organo terminale del manipolatore note le posizioni dei vari giunti che lo compongono [24].

Esaminando la Figura 3.5, se si applica il teorema di Pitagora al triangolo rettangolo in rosso si ottiene:

$$RR = \frac{\sqrt{L_1^2 + (q_2 - q_1)^2}}{2} \quad (3.12)$$

Ricorrendo successivamente al primo teorema trigonometrico sui triangoli rettangoli si ha:

$$AA = \arccos\left(\frac{L_1}{2RR}\right) \quad (3.13)$$

Se si considera il triangolo rettangolo (in azzurro) che ha per ipotenusa l'asta di lunghezza L_2 , si riesce a ricavare:

$$BB = \arccos\left(\frac{RR}{L_2}\right) \quad (3.14)$$

L'angolo CC del triangolo rettangolo in verde, invece, si calcola nel seguente modo:

$$CC = \frac{\pi}{2} - AA - BB \quad (3.15)$$

Nel momento in cui entrambe le variabili di giunto q_1 e q_2 assumono valori nulli, l'end-effector del manipolatore si posiziona in corrispondenza del punto P_0 . In questa configurazione si viene a formare il triangolo rettangolo in giallo; applicando su quest'ultimo il teorema di Pitagora si ottiene:

$$W = \sqrt{L_2^2 - \left(\frac{L_1}{2}\right)^2} \quad (3.16)$$

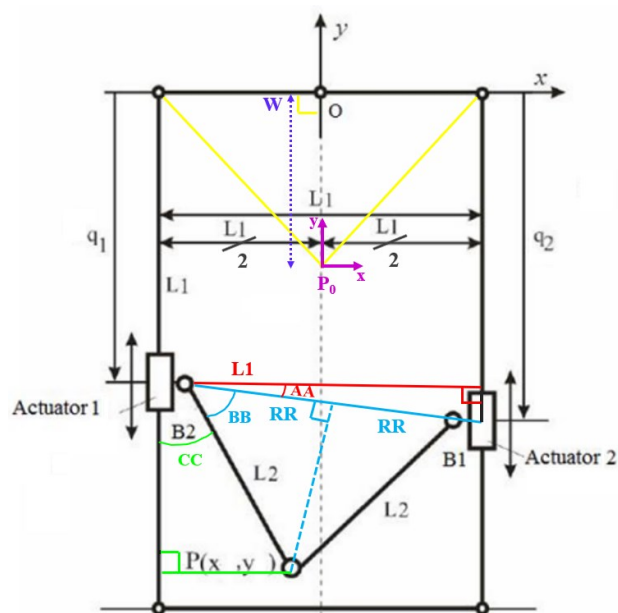


Figura 3.5 -Schema geometrico del manipolatore PRRRP

Facendo riferimento alle grandezze appena calcolate si possono scrivere le equazioni della cinematica diretta di posizione [25]. In questo caso si effettua una distinzione tra due possibilità:

- se $q_1 > q_2$ (casistica di Figura 3.5):

$$x = L_2 \sin(CC) - \frac{L_1}{2} \quad , \quad y = W - L_2 \cos(CC) + q_1 \quad (3.17)$$

- se $q_1 < q_2$:

$$x = \frac{L_1}{2} - L_2 \sin(CC) \quad , \quad y = W - L_2 \cos(CC) + q_2 \quad (3.18)$$

3.6 Problema di cinematica diretta di velocità

Dati i valori delle velocità dei giunti del robot, il problema di cinematica differenziale diretta consiste nel calcolo della velocità corrispondente dell'end-effector [24].

La soluzione alla presente problematica si ottiene risolvendo l'equazione riportata di seguito:

$$\dot{P} = J\dot{Q} \quad (3.19)$$

dove il vettore \dot{P} rappresenta la velocità lineare dell'end-effector, mentre il vettore \dot{Q} le velocità dei giunti. Lo Jacobiano J viene calcolato ricorrendo all'operazione di inversione della matrice J^{-1} della formula (3.10). Di conseguenza si ha che:

$$J = (J^{-1})^{-1} = \begin{bmatrix} \frac{(y-q_1)(y-q_2)}{\left(x+\frac{L_1}{2}\right)(y-q_2)-\left(x-\frac{L_1}{2}\right)(y-q_1)} & -\frac{(y-q_1)(y-q_2)}{\left(x+\frac{L_1}{2}\right)(y-q_2)-\left(x-\frac{L_1}{2}\right)(y-q_1)} \\ -\frac{(y-q_1)\left(x-\frac{L_1}{2}\right)}{\left(x+\frac{L_1}{2}\right)(y-q_2)-\left(x-\frac{L_1}{2}\right)(y-q_1)} & \frac{(y-q_2)\left(x+\frac{L_1}{2}\right)}{\left(x+\frac{L_1}{2}\right)(y-q_2)-\left(x-\frac{L_1}{2}\right)(y-q_1)} \end{bmatrix} \quad (3.20)$$

3.7 Singolarità cinematiche

L'operatore Jacobiano, nell'equazione di cinematica differenziale diretta, consente di definire una trasformazione lineare tra il vettore delle velocità lineare ed angolare dell'end-effector ($v_e = [\dot{p}_e^T \quad \omega_e^T]^T$) e quello delle velocità dei giunti (\dot{Q}).

$$v_e = J(q)\dot{Q} \quad (3.21)$$

La matrice J della formula (3.21) è funzione della configurazione geometrica (Q) assunta dal manipolatore. In particolare, tutte quelle configurazioni in cui il rango di J diminuisce (non è pieno e quindi il determinante di essa è nullo) vengono definite singolarità cinematiche. L'identificazione di queste ultime è di fondamentale importanza per le motivazioni di seguito elencate.

- Le condizioni di singolarità rappresentano situazioni in corrispondenza delle quali si ha perdita di mobilità del robot. Nello specifico vengono impedito determinate direzioni di spostamento dell'organo terminale, per cui non è possibile imporre leggi di moto arbitrarie ad esso.
- Nel momento in cui il manipolatore è in una configurazione singolare, è possibile che esistano infinite soluzioni al problema cinematico inverso.
- Nell'intorno di una singolarità, velocità ridotte nello spazio operativo possono indurre velocità elevate nello spazio dei giunti.

Pertanto, tali configurazioni vanno dapprima individuate e poi evitate. Inoltre, le singolarità cinematiche si classificano nelle due tipologie riportate di seguito.

- *Singolarità ai confini dello spazio di lavoro raggiungibile.* Si verificano nel momento in cui il manipolatore è tutto disteso o tutto ripiegato su sé stesso; esse non rappresentano un grosso problema in quanto possono essere evitate facendo in modo che il sistema robotico lavori all'interno del workspace e non ai confini di quest'ultimo.
- *Singolarità all'interno dello spazio di lavoro raggiungibile.* Sono causate tipicamente dall'allineamento di due o più assi di moto, oppure dall'assunzione da parte dell'end-effector di posture particolari. Questa categoria di condizioni singolari rappresenta un serio problema in quanto esse possono essere raggiunte, senza volerlo, con traiettorie pianificate all'interno dello spazio operativo [20].

3.7.1 Condizioni di singolarità nei robot paralleli

Diverse macchine robotiche a cinematica parallela sono caratterizzate da singolarità interne allo spazio di lavoro che, come specificato in precedenza, sono difficili da identificare e comportano una perdita della controllabilità del manipolatore. L'analisi delle condizioni di singolarità di un robot parallelo viene effettuata applicando il metodo ideato da Clement Gosselin e Jorge Angeles. Tale formulazione permette di ridurre le equazioni cinematiche nella seguente relazione ingresso-uscita tra il vettore q delle variabili di giunto e il vettore p delle coordinate cartesiane dell'end-effector:

$$f(q, p) = 0 \quad (3.22)$$

Derivando la formula (3.22) rispetto al tempo si ottiene una relazione che permette di correlare le velocità dei giunti con quelle dell'end-effector:

$$J_p \dot{p} = J_q \dot{q} \quad (3.23)$$

dove $J_p = \frac{\partial f}{\partial p}$ e $J_q = -\frac{\partial f}{\partial q}$ rappresentano rispettivamente le matrici jacobiane della cinematica differenziale diretta ed inversa. Combinando i due termini si ottiene la seguente espressione:

$$J = J_p^{-1} J_q \quad (3.24)$$

Partendo da considerazioni effettuate sulle due tipologie di Jacobiano appena definite, si possono individuare le tre seguenti categorie di singolarità che caratterizzano i manipolatori paralleli.

- *Singolarità seriali o singolarità della cinematica inversa.* Sono quelle configurazioni in corrispondenza delle quali il determinante della matrice J_q si annulla ($\det(J_q) = 0$).
- *Singolarità parallele o singolarità della cinematica diretta.* Si ottengono in quelle situazioni in cui la matrice J_p risulta singolare ($\det(J_p) = 0$).
- *Singolarità dell'architettura.* Rappresentano quelle condizioni nelle quali il determinante di entrambe le matrici si annulla ($\det(J_q) = \det(J_p) = 0$) ([15], [16]).

3.7.2 Singolarità del manipolatore PRRRP 2-DOF

Per ricavare le condizioni di singolarità del manipolatore parallelo PRRRP a due gradi di libertà è necessario applicare il metodo di Gosselin e Angeles introdotto in precedenza. A tal fine si confrontano le equazioni (3.24) e (3.10) e si ricava che: $J_p = B$ e $J_q = A$. Pertanto, per determinare le singolarità della cinematica inversa, è necessario individuare quelle soluzioni che rendono il determinante della matrice A nullo. Analizzando l'espressione (3.7) si evince che:

$$\det(A) = (y - q_1)(y - q_2) \quad (3.25)$$

Ponendo il determinante di A pari a zero si ottiene:

$$y = q_1 \vee y = q_2 \quad (3.26)$$

La radice $y = q_1$ della (3.26) corrisponde al caso in cui la prima asta (link 4) è parallela all'asse delle ascisse, mentre la seconda (link 5) a quello delle ordinate; mentre la soluzione $y = q_2$ della (3.26) rappresenta il viceversa. Le due situazioni descritte sono riportate graficamente in Figura 3.6 e in Figura 3.7.

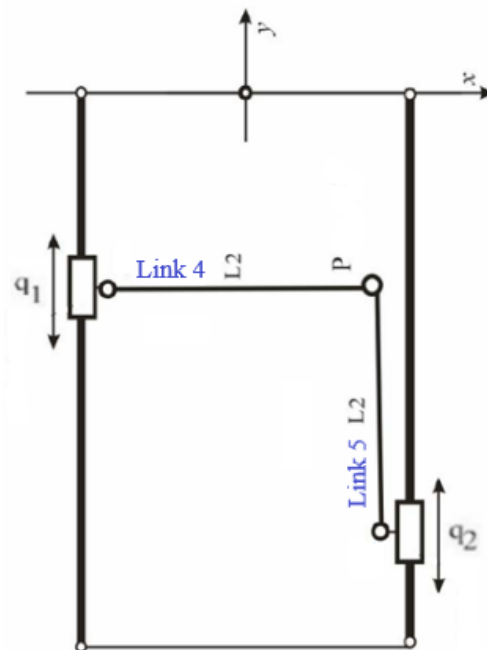


Figura 3.6 – Singolarità della cinematica inversa del manipolatore parallelo PRRRP 2-DOF (Soluzione $y = q_1$)
(da [23])

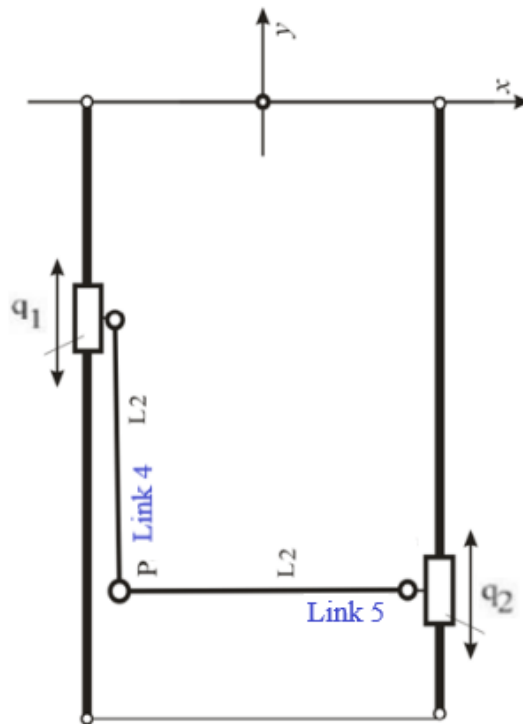


Figura 3.7 - Singolarità della cinematica inversa del manipolatore parallelo PRRRP 2-DOF (Soluzione $y = q_2$) (da [23])

Per determinare le condizioni di singolarità della cinematica diretta del robot oggetto di studio, bisogna porre uguale a zero il determinante della matrice B. Partendo dall'espressione (3.7) si ricava che:

$$\det(B) = \left(x + \frac{L_1}{2}\right)(y - q_2) - \left(x - \frac{L_1}{2}\right)(y - q_1) \quad (3.27)$$

Annullando l'equazione (3.27) si determinano le seguenti soluzioni [25]:

$$se \ x = 0 \Rightarrow y = \frac{(q_1 + q_2)}{2} \quad (3.28)$$

La singolarità parallela appena calcolata è illustrata in Figura 3.8 e corrisponde alla situazione in cui le due aste sono allineate in diagonale.

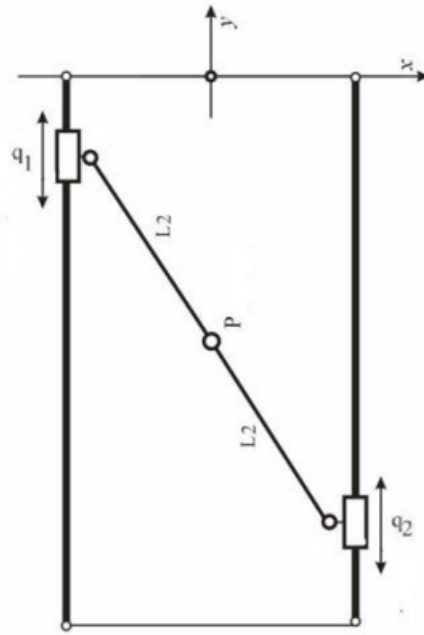


Figura 3.8 - Singolarità della cinematica diretta del manipolatore parallelo PRRRP 2-DOF (da [23])

4 Pianificazione della traiettoria dei manipolatori

La pianificazione della traiettoria consente di stabilire come debba evolvere il movimento del manipolatore (ossia con quali profili di posizione, velocità e accelerazione) partendo da una postura iniziale sino ad arrivare ad una postura finale desiderata. L'obiettivo, in generale, è quello di generare degli ingressi di riferimento per il sistema di controllo del moto. È necessario definire a tal proposito due elementi che insieme costituiscono la traiettoria:

- il percorso geometrico che il robot dovrà seguire nell'adempimento del compito assegnato, inteso come il luogo dei punti nello spazio operativo o nello spazio dei giunti che devono essere raggiunti dal manipolatore;
- una legge oraria da specificare per ogni punto del percorso in termini di velocità e accelerazioni.

L'operazione di pianificazione deve essere adeguata e precisa infatti, in tal modo, il controllore consentirà al manipolatore di inseguire fedelmente la traiettoria desiderata, senza violare i limiti di saturazione degli attuatori e senza indurre sollecitazioni meccaniche dannose alla struttura. Inoltre, come ulteriore specifica, è richiesto che le traiettorie generate siano caratterizzate da una curvatura più "smooth" possibile.

4.1 Algoritmo di pianificazione della traiettoria

Un generico algoritmo di pianificazione della traiettoria riceve come ingressi:

- la descrizione del percorso da compiere;
- i vincoli associati al percorso;
- i vincoli imposti dalla cinematica e dalla dinamica del manipolatore.

In uscita viene prodotta una traiettoria (assegnata all'end-effector o ai giunti) da fornire a sua volta come input al controllore del moto, nella forma di una sequenza di valori assunti da posizione, velocità, accelerazione. In Figura 4.1 viene riportato uno schema a blocchi che descrive, per sommi capi, un tipico algoritmo pianificatore di traiettoria.

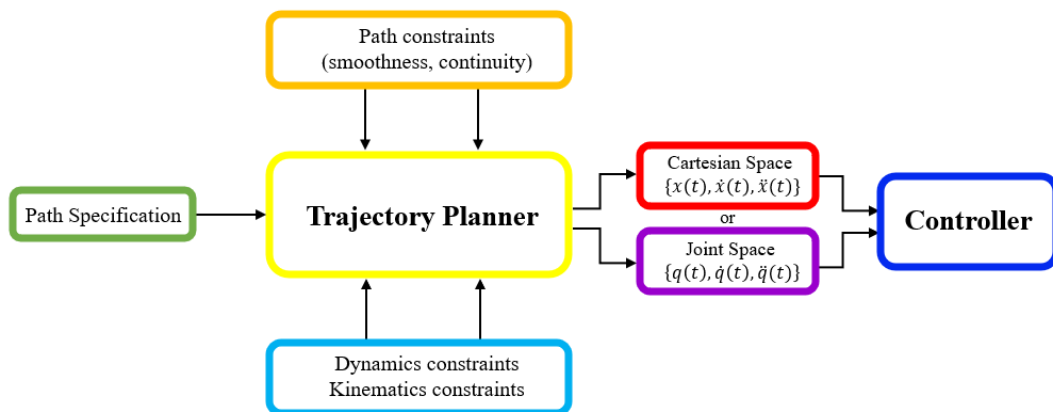


Figura 4.1 – Schema a blocchi del pianificatore di traiettoria

Un utente che intenda applicare una procedura di questo tipo deve stabilire una serie di parametri specifici, i quali sono riportati di seguito.

- *Parametri di percorso.* Si articolano in punti estremi, eventuali punti intermedi e primitive geometriche che interpolano tali punti.
- *Parametri della legge oraria.* Riguardano il tempo totale di esecuzione della traiettoria, le velocità e/o le accelerazioni massime e le velocità e/o le accelerazioni in determinati punti ([20], [26]).

4.2 Traiettorie nello spazio dei giunti e nello spazio operativo

L'attività di pianificazione delle traiettorie può essere effettuata sia nello spazio dei giunti sia nello spazio operativo. Nel primo caso si definisce l'andamento che si desidera per la posizione, la velocità e l'accelerazione di ogni giunto. Nello specifico viene generata una funzione $q(t)$ che va ad interpolare i vettori delle variabili di giunto desiderati rispettando, allo stesso tempo, i vincoli imposti. Nel caso in cui i valori di riferimento delle variabili di giunto non siano noti, essi possono essere ricavati attraverso operazioni di inversione della cinematica a partire dai punti indicati dall'utente nello spazio operativo. Questi ultimi corrispondono a determinate pose da far assumere all'end-effector nel corso del tempo e, se combinati, formano il percorso geometrico che l'organo terminale deve seguire.

Un algoritmo di pianificazione nello spazio dei giunti deve possedere le seguenti caratteristiche:

- una bassa complessità dal punto di vista computazionale, dal momento che le traiettorie possono essere modificate repentinamente o di frequente per reagire ad eventi. Se la procedura fosse onerosa, infatti, si otterrebbe una perdita di reattività da parte del manipolatore;
- le posizioni, le velocità ed eventualmente le accelerazioni specificate devono essere funzioni continue del tempo;
- gli effetti indesiderati (ad esempio curvature irregolari e caratterizzate da oscillazioni eccessive) dovuti ad approssimazioni sia a livello numerico che di calcolo degli algoritmi di controllo, devono essere minimizzati.

Tale tipologia di pianificazione, rispetto a quella effettuata nello spazio operativo, consente di eliminare le problematiche associate sia a movimenti che avvengono nelle vicinanze di configurazioni singolari sia alla presenza di ridondanza nei manipolatori. Se si desidera che il moto del sistema robotico segua un percorso geometrico specificato nello spazio operativo, è necessario pianificare la traiettoria direttamente nel suddetto spazio. A tal proposito si possono distinguere due casistiche principali:

- se il percorso non deve essere seguito con esattezza dal manipolatore, vengono specificati N punti $x_e(t_k)$ i quali indicano le pose assunte dall'organo terminale negli istanti t_k , con $k = 1, \dots, N$. La traiettoria viene poi generata interpolando i punti definiti attraverso una funzione vettoriale;
- se il moto dell'end-effector deve attenersi fedelmente ad un cammino prestabilito, è necessario esprimere la traiettoria in maniera analitica servendosi di primitive di percorso.

In ciascuna delle due casistiche associate allo spazio operativo, per il passaggio alla fase di attuazione è richiesto uno stadio di inversione della cinematica dal momento che, tipicamente, il controllo del manipolatore viene effettuato nello spazio dei giunti. Pertanto la procedura di pianificazione nello spazio operativo richiede una complessità computazionale più elevata. Tuttavia, in questo caso, la descrizione del compito che il robot deve eseguire risulta più intuitiva e si riescono a definire facilmente i vincoli dovuti alla presenza di ostacoli nello spazio di lavoro ([20], [27]).

In entrambi gli spazi le traiettorie possono essere generate adottando una delle seguenti tecniche che verranno descritte nei paragrafi successivi.

- *Moto punto-punto;*
- *Moto attraverso una sequenza di punti (o su percorso assegnato).*

4.3 Moto punto-punto

Nel moto punto-punto vengono specificati le posizioni iniziale e finale che il manipolatore dovrà assumere nell'esecuzione del percorso geometrico e la durata complessiva del movimento risultante. Per realizzare questa tipologia di pianificazione, la funzione matematica interpolante più semplice da impiegare è quella polinomiale. In particolare, se bisogna imporre soltanto il passaggio per n punti di percorso (dove nella presente casistica $n = 2$), si adatterà un polinomio di grado $(N-1)$ per definire la traiettoria; mentre se si desidera associare a ciascun punto anche un determinato valore di velocità, sarà necessario ricorrere all'utilizzo di un polinomio di grado $2(n - 1)$. Infine se si vogliono stabilire anche i valori di accelerazione in corrispondenza di tali punti bisognerà definire un polinomio di grado $3(n - 1)$ per generare traiettoria. Solitamente vengono impiegati polinomi cubici (di terzo grado) che permettono di soddisfare quattro vincoli; nello specifico due di questi ultimi sono sulle posizioni iniziale e finale, mentre i restanti sulle velocità iniziale e finale (generalmente poste uguali a zero). Ne risulta un profilo di velocità parabolico e uno di accelerazione lineare. Nel caso in cui è richiesto di fissare anche le accelerazioni di partenza e di arrivo, le condizioni da garantire diventano sei e ci si avvale di un polinomio di quinto grado (quintico).

In ambito industriale è largamente diffusa, per la semplicità di utilizzo, un'altra modalità di pianificazione nota con il nome di profilo trapezoidale di velocità. La traiettoria risultante è di tipo polinomiale misto ed è costituita da un tratto lineare raccordato con due segmenti parabolici alle posizioni iniziale e finale. Essa si ottiene imponendo un'accelerazione costante nella fase di partenza, un'accelerazione nulla (velocità costante) nella fase intermedia e una decelerazione costante nella fase di arrivo (di ugual durata rispetto a quella di inizio \Rightarrow la traiettoria è simmetrica rispetto all'istante intermedio) ([20], [27]). Dunque, come mostrato in Figura 4.2, si ottiene

una suddivisione in tre intervalli a ciascuno dei quali viene associata una legge oraria differente:

1. accelerazione positiva a gradino, velocità positiva a rampa, posizione a parabola;
2. accelerazione nulla, velocità costante, posizione lineare;
3. accelerazione negativa a gradino, velocità negativa a rampa, posizione a parabola [26].

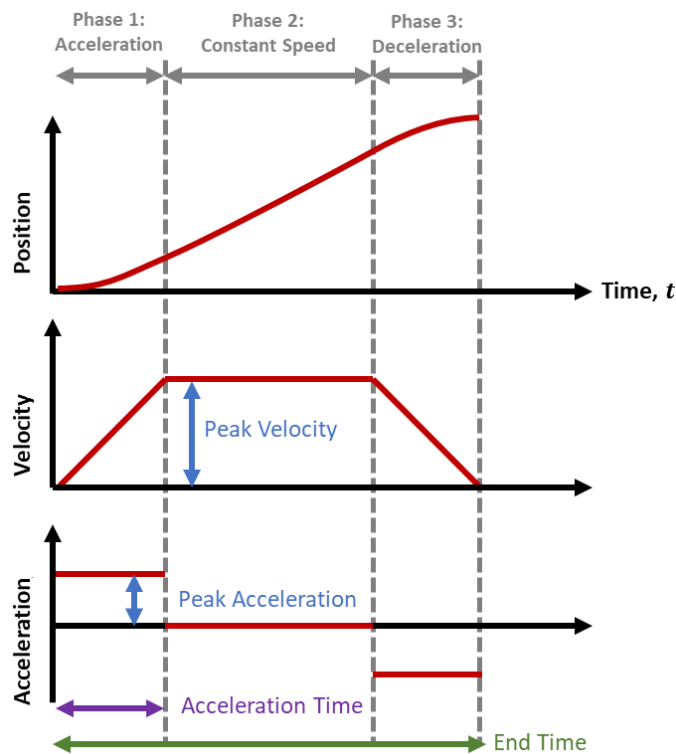


Figura 4.2 – Legge oraria corrispondente al profilo trapezoidale di velocità (da [29])

4.4 Moto attraverso una sequenza di punti

Adottando la tecnica del moto punto-punto si possono ottenere delle condizioni di non continuità delle funzioni di velocità e accelerazione le quali potrebbero comportare problemi al controllo dei giunti. Pertanto in molte applicazioni si adotta una modalità alternativa di pianificazione che è quella del moto su percorso assegnato. Quest'ultima prevede di specificare, oltre ai punti estremi e al tempo di percorrenza del percorso, anche punti intermedi. Dunque il problema è quello di generare una traiettoria che

passi per n punti prestabiliti in determinati istanti di tempo desiderati. La soluzione a tale problematica prevede l'utilizzo di un polinomio di ordine $(n - 1)$. La densità dei punti di passaggio viene stabilita in base a quanto preciso debba essere il movimento del manipolatore e viene aumentata in prossimità di ostacoli nello spazio di lavoro.

Il vantaggio della suddetta modalità, come già accennato, è che la funzione generata tramite interpolazione dei vari punti ha derivate continue. Tuttavia al crescere del numero n indicato in precedenza, aumenta il grado del polinomio che descrive la traiettoria; quest'ultima condizione comporta un'accentuazione del comportamento oscillatorio scaturito dalla funzione polinomiale e una diminuzione dell'accuratezza numerica nel calcolo dei suoi coefficienti. Inoltre, se si va a modificare uno dei punti di percorso assegnati, si renderà necessario ricalcolare tutti i coefficienti. Per risolvere le suddette problematiche, viene sostituito il polinomio di grado $(n - 1)$ con $(n - 1)$ funzioni polinomiali di grado inferiore, ciascuna delle quali descrive un tratto di traiettoria. Generalmente vengono scelti polinomi cubici in modo da garantire condizioni di continuità delle velocità e delle accelerazioni. Tuttavia l'impiego di funzioni di grado pari a tre genera una traiettoria caratterizzata da discontinuità di accelerazione nei punti di passaggio. Per porre rimedio alla questione appena definita, si evita di imporre velocità specifiche nei punti intermedi, stabilendo soltanto la continuità in due tratti adiacenti in termini di posizione, velocità ed accelerazione. La traiettoria risultante ha la caratteristica di essere la funzione interpolante a curvatura minima (a parità di condizioni di continuità sulle derivate) e prende il nome di spline (Smooth Path Line) [26]. Quest'ultima, dunque, si può definire come una traiettoria multi-punto data dall'unione di $(n - 1)$ polinomi (di grado 3 o talvolta anche 5) con vincoli di passaggio per n punti.

Prendendo in considerazione il caso della pianificazione nello spazio dei giunti, la spline cubica si determina in maniera analitica come segue:

$$s(t) = \{q_k(t), t \in [t_k, t_{k+1}], k = 0, \dots, n - 1\} \quad (4.1)$$

dove le funzioni polinomiali in corrispondenza di ciascun tratto risultano:

$$q_k(t) = a_{k0} + a_{k1}(t - t_k) + a_{k2}(t - t_k)^2 + a_{k3}(t - t_k)^3 \quad (4.2)$$

In tal modo ciascuno degli $(n - 1)$ polinomi è caratterizzato da quattro coefficienti da determinare; si ottiene così un numero complessivo di parametri da calcolare pari a $4(n - 1)$.

Inoltre tale procedura di pianificazione richiede di imporre i seguenti vincoli:

- $2(n - 1)$ condizioni di passaggio per i punti, dal momento che ogni cubica deve interpolare gli estremi del segmento che descrive;
- $(n - 2)$ condizioni di continuità delle velocità nei punti intermedi;
- $(n - 2)$ condizioni di continuità delle accelerazioni nei punti intermedi.

Pertanto dal confronto riportato nella formula (4.3) tra il numero di parametri e il numero dei vincoli emerge che si hanno a disposizione 2 gradi di libertà residui; questi ultimi possono essere impiegati ad esempio per imporre determinate condizioni sulla velocità iniziale e finale di $s(t)$.

$$4(n - 1) - 2(n - 1) - 2(n - 2) = 2 \text{ DOF} \quad (4.3)$$

In Figura 4.3 è mostrata una generica traiettoria spline con interpolazione realizzata mediante funzioni cubiche.

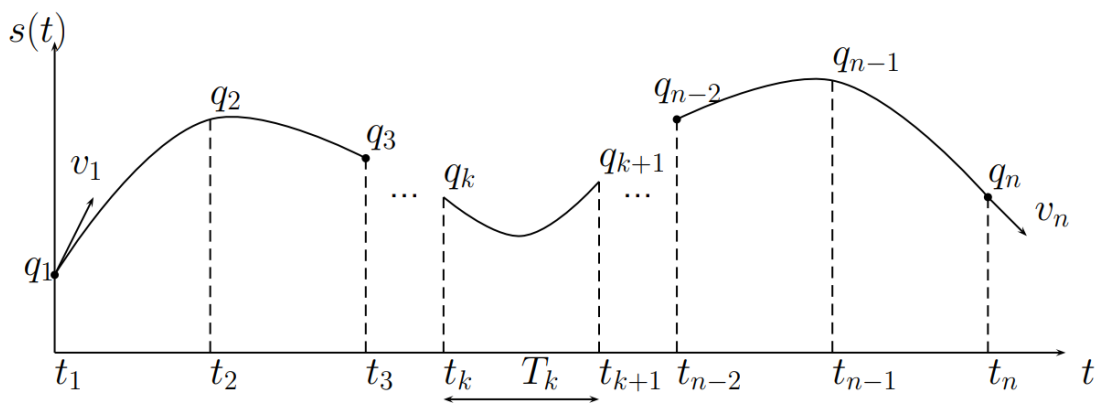


Figura 4.3 – Rappresentazione di una generica spline cubica (da [28])

Il calcolo dei coefficienti del polinomio di grado 3 dell'espressione (4.2) viene realizzato attraverso uno specifico algoritmo che prevede di assumere note le velocità v_k (per $k = 1, \dots, n - 1$) in corrispondenza dei punti intermedi. In particolare, per ciascuna funzione polinomiale è necessario prendere in considerazione le quattro

condizioni al contorno definite in precedenza che danno origine al seguente sistema di quattro equazioni:

$$\begin{cases} q_k(t_k) = a_{k0} = q_k \\ \dot{q}_k(t_k) = a_{k1} = v_k \\ q_k(t_{k+1}) = a_{k0} + a_{k1}T_k + a_{k2}T_k^2 + a_{k3}T_k^3 = q_{k+1} \\ \dot{q}_k(t_{k+1}) = a_{k1} + 2a_{k2}T_k + 3a_{k3}T_k^2 = v_{k+1} \end{cases} \quad (4.4)$$

dove $T_k = t_{k+1} - t_k$.

Infine risolvendo il sistema (4.4), andando a considerare i quattro coefficienti del polinomio cubico come incognite, si ottengono le soluzioni che seguono [28]:

$$\begin{cases} a_{k0} = q_k \\ a_{k1} = v_k \\ a_{k2} = \frac{1}{T_k} \left[\frac{3(q_{k+1} - q_k)}{T_k} - 2v_k - v_{k+1} \right] \\ a_{k3} = \frac{1}{T_k^2} \left[\frac{2(q_k - q_{k+1})}{T_k} + v_k + v_{k+1} \right] \end{cases} \quad (4.5)$$

5 Hardware e dispositivi

Nell'azienda in cui è stata condotta l'attività di tirocinio vengono impiegati diversi componenti hardware e dispositivi necessari per imprimere movimento al manipolatore oggetto di studio e per gestirne il comportamento. Tali strumenti verranno man mano indicati ed illustrati nel corso del presente capitolo.

5.1 Generazione e controllo del moto

In questa sezione viene descritta tutta l'apparecchiatura che entra in gioco nella generazione e nel controllo del moto del sistema robotico.

5.1.1 Motori lineari

I robot PRRRP, nel caso in esame, sono dotati di due attuatori elettromagnetici lineari LinMot. Questi ultimi sono motori elettrici tubolari caratterizzati soltanto da due parti:

- un cursore realizzato con magneti al neodimio immessi in un tubo in acciaio inox;
- uno statore dove sono posizionati gli avvolgimenti, il cuscinetto per il cursore, un sensore per la misura della posizione (encoder) ed uno di temperatura per il monitoraggio termico del motore.

La loro forma semplice li rende facilmente installabili e permette di ridurre l'ingombro. La Figura 5.1 mostra la struttura di un attuatore che rientra in questa tipologia.

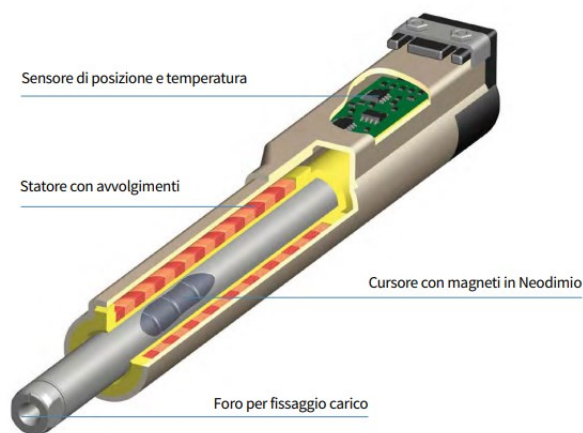


Figura 5.1 – Struttura di un attuatore lineare LinMot (da [30])

Il moto lineare è prodotto in modo puramente elettrico, infatti viene sfruttato il principio elettromagnetico per generarlo. Pertanto non è necessario utilizzare a tal fine alcun organo di trasmissione meccanica come camme, ingranaggi e cinghie. Questo aspetto consente di semplificare il design di tali dispositivi e di minimizzare l'usura dei componenti e la manutenzione di essi. Inoltre un ulteriore vantaggio è rappresentato dal movimento preciso, fluido e silenzioso dei motori sopra citati che li rende ideali in diverse applicazioni ([30], [31]).

5.1.2 Azionamenti e PLC

Gli attuatori lineari del PRRRP sono alimentati da azionamenti elettrici servoassistiti LinMot C1250, mostrati in Figura 5.2, che prelevano energia elettrica da una sorgente ed erogano potenza ai motori ai quali sono associati.



Figura 5.2 - Azionamenti elettrici servoassistiti LinMot C1250 (da [32])

Oltre allo stadio di potenza appena descritto, gli azionamenti sono caratterizzati da un'altra sezione che è quella relativa al controllo. Quest'ultima è costituita da un controllore che, a partire dai segnali di comando inviati dall'esterno da un PLC e dalle misure eseguite dai sensori installati nei motori, genera dei segnali di controllo per gestire la parte di potenza. In particolare, tale sistema di regolazione ha il compito di garantire che l'attuatore corrispondente si comporti nella maniera desiderata eseguendo correttamente i comandi ad esso inviati [33]. In Figura 5.3 viene illustrato lo schema di un generico azionamento elettrico.

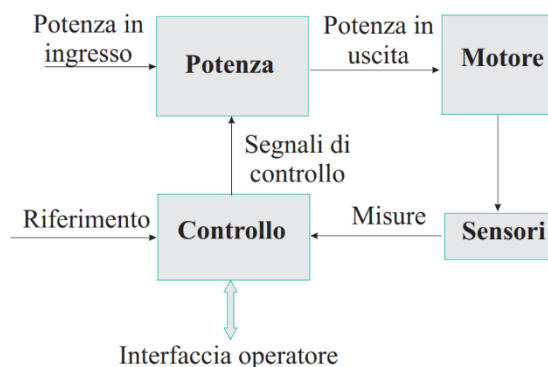


Figura 5.3 - Schema di un generico azionamento elettrico (da [33])

Il Controllore Logico Programmabile (PLC) impiegato per la gestione e il controllo dell'impianto e dei manipolatori è l'XM42 della Bosch Rexroth riportato in Figura 5.4.



Figura 5.4 - PLC XM42 Bosch Rexroth (da [34])

Il PLC e gli azionamenti comunicano attraverso il Sercos III (Serial Realtime Communication System), il quale è un bus di campo digitale per la trasmissione dati in tempo reale, seriale e ad alta velocità. Esso utilizza il protocollo dello standard Ethernet IEEE 802.3 per gestire la comunicazione tra dispositivi [35].

5.1.3 Configurazione dei drive

Al fine di poter utilizzare i drive per comandare gli attuatori del manipolatore è richiesta una fase iniziale di configurazione di essi. L'operazione citata è stata realizzata adottando il software fornito dall'azienda costruttrice LinMot dei motori lineari. Tale procedura prevede di effettuare i due passi indicati di seguito.

1. Collegare il pc al drive attraverso un'interfaccia di configurazione di tipo rs232 per la trasmissione seriale di dati (Figura 5.5).

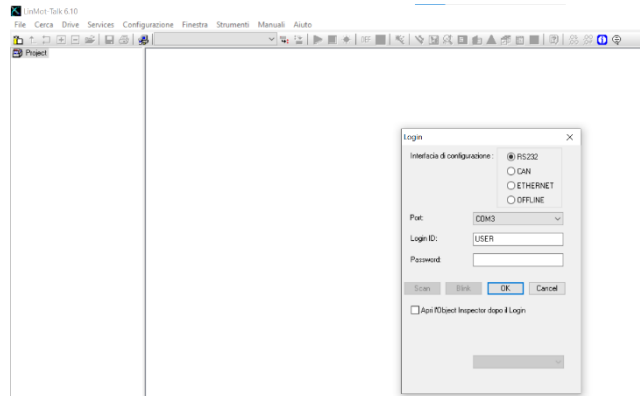


Figura 5.5 - Connessione pc-drive tramite rs232

2. Impostare i dati del motore all'interno del drive effettuando una parametrizzazione del carico che agisce sul singolo attuatore (Figura 5.6), una regolazione dei guadagni del PID dell'azionamento (Figura 5.7) e un settaggio dei parametri relativi al dimensionamento della corsa del motore (Figura 5.8).

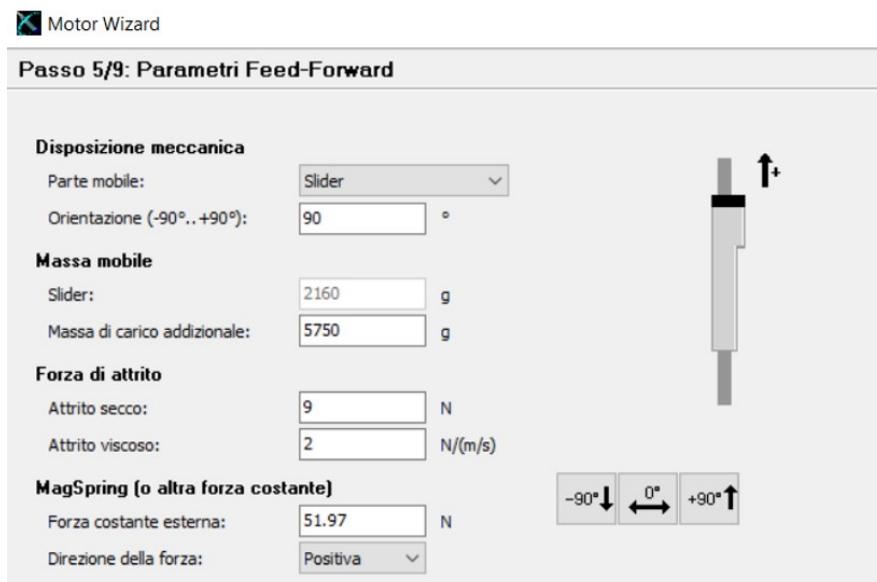


Figura 5.6 – Parametri relativi al carico sul singolo motore

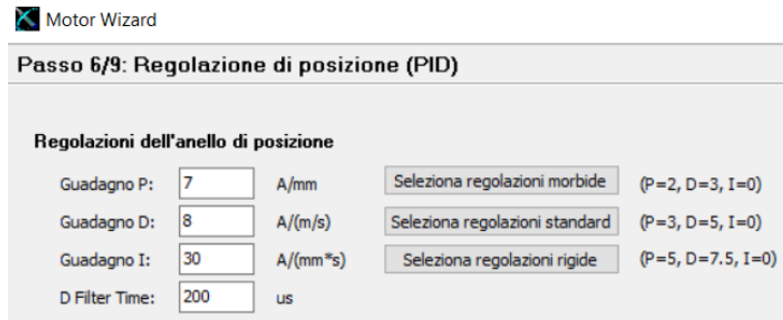


Figura 5.7 -Regolazione dei guadagni del PID del drive

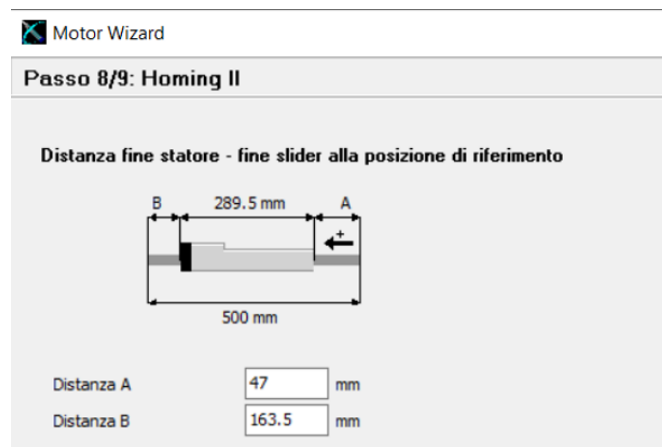


Figura 5.8 – Parametri per il dimensionamento della corsa del motore

Una volta portato a termine questo procedimento è necessario aggiornare il firmware del singolo drive e l'azionamento, di conseguenza, sarà pronto per interagire con il Controllore Logico Programmabile.

5.1.4 Configurazione del PLC

È richiesta anche una fase di configurazione del PLC per garantirne l'utilizzo e per far in modo che esso comunichi con gli azionamenti. L'operazione richiede di effettuare i passaggi che seguono.

1. Assegnare un indirizzo Sercos al drive, ottenibile da una configurazione binaria sugli switch dell'azionamento. In questo modo l'azionamento in questione potrà essere identificato univocamente durante la fase di comunicazione con il Controllore Logico Programmabile.

2. Effettuare, nel software impiegato per la programmazione del controllore, una ricerca dei dispositivi Sercos connessi fisicamente al PLC e rendere essi disponibili alla configurazione del PLC come mostrato in Figura 5.9.

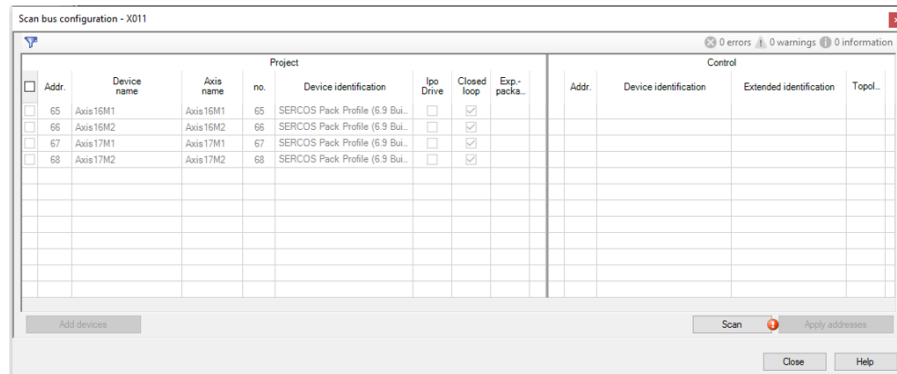


Figura 5.9 – Ricerca dei dispositivi Sercos e configurazione del PLC

Una volta ultimata la procedura di configurazione del Controllore Logico Programmabile si può passare alla stesura del codice che sarà eseguito da esso.

5.2 Gestione della sicurezza

Per assicurare un funzionamento sicuro dell'impianto e per gestire e ridurre eventuali condizioni di pericolo per persone e macchine, vengono impiegati dei moduli di sicurezza prodotti dall'azienda Pilz del tipo mostrato in Figura 5.10 [36].



Figura 5.10 - Modulo di sicurezza Pilz PNOZ m1p-mb0 (da [37])

Tali strumenti non sono altro che sistemi di controllo che si interfacciano con diversi dispositivi di sicurezza quali il tasto di arresto di emergenza e i sensori adibiti alla rilevazione dell'apertura delle porte quando il macchinario è in funzione. I moduli di sicurezza ricevono segnali dai componenti appena citati e se si verifica una condizione di potenziale pericolo, interrompono immediatamente l'alimentazione dei drive in modo sicuro attraverso la funzione "Safe Torque Off" (STO- Disattivazione coppia in sicurezza). Il controllore di safety condivide con il PLC il monitoraggio di tali segnali al fine di gestire gli allarmi e i warning che si generano.

6 Implementazione software

L'attività di sviluppo software condotta per il lavoro di tesi è finalizzata alla realizzazione di due programmi differenti che consentono di risolvere rispettivamente le problematiche che seguono.

- Preparare il macchinario all'utilizzo dei dispositivi che interessano la stazione del robot oggetto di studio.
- Gestire il moto del manipolatore parallelo PRRRP a due gradi di libertà impiegato dall'azienda.

Nei paragrafi successivi saranno descritti nel dettaglio i problemi appena citati e le relative soluzioni adottate.

6.1 Linguaggio e ambiente di programmazione

La stesura dei codici è stata effettuata utilizzando un linguaggio di programmazione ad alto livello, simile al C e al Pascal, che viene impiegato solitamente per programmare i processi industriali automatizzati. Esso è il Testo Strutturato (ST) Codesys definito dalla normativa IEC 61131-3. I software realizzati per il lavoro di tesi sono basati essenzialmente sull'utilizzo di due elementi caratteristici di tale linguaggio. Questi ultimi vengono definiti come Program Organization Unit (POU) e sono riportati di seguito.

- *Program*. È un programma che viene eseguito ciclicamente dal PLC.
- *Function Block (FB)*. Sono moduli di programma riutilizzabili caratterizzati da dati (variabili in ingresso, in uscita e interne) e da un corpo, ovvero l'insieme delle istruzioni che vengono eseguite ogni volta che essi vengono richiamati. A differenza delle funzioni (*Functions*) classiche, i blocchi funzione sono dotati di uno stato interno che permette loro di mantenere, tra due esecuzioni successive, i valori delle variabili interne [38].

L'ambiente di sviluppo software adottato è IndraWorks Engineering della Bosch Rexroth, uno strumento utile per assolvere tutti quei compiti relativi all'automazione basata su Controllori Logici Programmabili e alla messa in servizio degli azionamenti. Inoltre, esso è usato anche per il controllo del moto di macchinari e dei loro

sottocomponenti [39]. All'interno di questo applicativo è possibile testare il codice in simulazione senza avere a disposizione il sistema real time e il PLC.

6.2 Preparazione della macchina all'utilizzo dei dispositivi

Nel presente paragrafo viene descritta e modellata la fase in cui il controllore della macchina chiede ai dispositivi di attivarsi e rendersi pronti, in modo che l'operatore possa far partire il ciclo di lavoro. Nella modellazione vengono considerati gli allarmi che possono derivare da tale procedimento e le azioni da effettuare per reagire ad essi.

6.2.1 Dispositivi interessati

Come già specificato in precedenza, i dispositivi principali considerati nell'operazione di preparazione sono quelli che intervengono nella stazione di lavoro del robot PRRRP, ovvero:

- gli azionamenti dei due motori lineari del sistema robotico;
- la tavola trasporta-pezzi principale dell'impianto;
- l'ordinatore esterno (feeder) che immette nel macchinario determinati componenti. Questi ultimi vengono prelevati dal manipolatore ed assemblati con gli altri pezzi presenti sulla tavola principale.

Tali dispositivi ricevono richieste espresse attraverso specifici comandi emessi dal PLC. Di conseguenza, se non si verificano condizioni di allarme o di guasto, essi forniscono un determinato servizio e, al termine, segnalano al richiedente che l'esecuzione è avvenuta con successo tramite un acknowledge [40]. La Figura 6.1 illustra il diagramma degli stati associato alla situazione appena descritta.

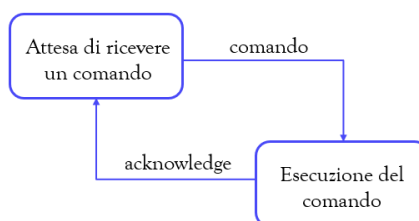


Figura 6.1 – Diagramma degli stati del dispositivo

Oltre ai quattro strumenti principali sopracitati, vengono prese in considerazione anche altre tipologie di dispositivi, le quali sono riportate in Tabella 6.1.

Tabella 6.1 – Tipologie di dispositivi ausiliari

Dispositivi di sicurezza ed allarme	Pulsanti di comando operatore
<ul style="list-style-type: none"> ▪ <i>Modulo di sicurezza</i>: fornisce o toglie il comando di STO (Safe Torque Off) al PLC per comunicare rispettivamente che si è in una condizione di sicurezza oppure no. ▪ <i>Pulsante di arresto di emergenza a fungo</i>: serve per arrestare il funzionamento del macchinario il più rapidamente possibile in condizioni di potenziale pericolo. ▪ <i>Sensori adibiti alla rilevazione dell'apertura delle porte</i>: comunicano al modulo di sicurezza che le porte sono state aperte durante il funzionamento della macchina. 	<ul style="list-style-type: none"> ▪ <i>Tasto di Reset</i>: consente di ripristinare le condizioni di operatività e di sicurezza. ▪ <i>Tasto di Stop</i>: se premuto per un numero stabilito di secondi, mette in attesa il macchinario, disabilitando i dispositivi.

6.2.2 Passaggi di stato della macchina

Nella realizzazione del codice che predispose il macchinario all'utilizzo dei dispositivi sono stati modellati gli stati che attraversa la macchina durante questa fase:

- *Macchina in Power-Off* → L'impianto è alimentato, ma i dispositivi sono spenti;
- *Standstill* → I dispositivi sono alimentati, ma non ricevono ancora l'abilitazione che li rende pronti;
- *Homing* → I dispositivi sono alimentati e si sta eseguendo l'inizializzazione dei driver e della tavola;
- *Ready* → I dispositivi hanno ricevuto l'abilitazione e sono pronti per essere utilizzati. Il macchinario può iniziare, a questo punto, il ciclo di lavoro.

In Figura 6.2 è illustrato il diagramma degli stati della macchina (rappresentati attraverso cerchi) con i relativi passaggi di stato (raffigurati con frecce).

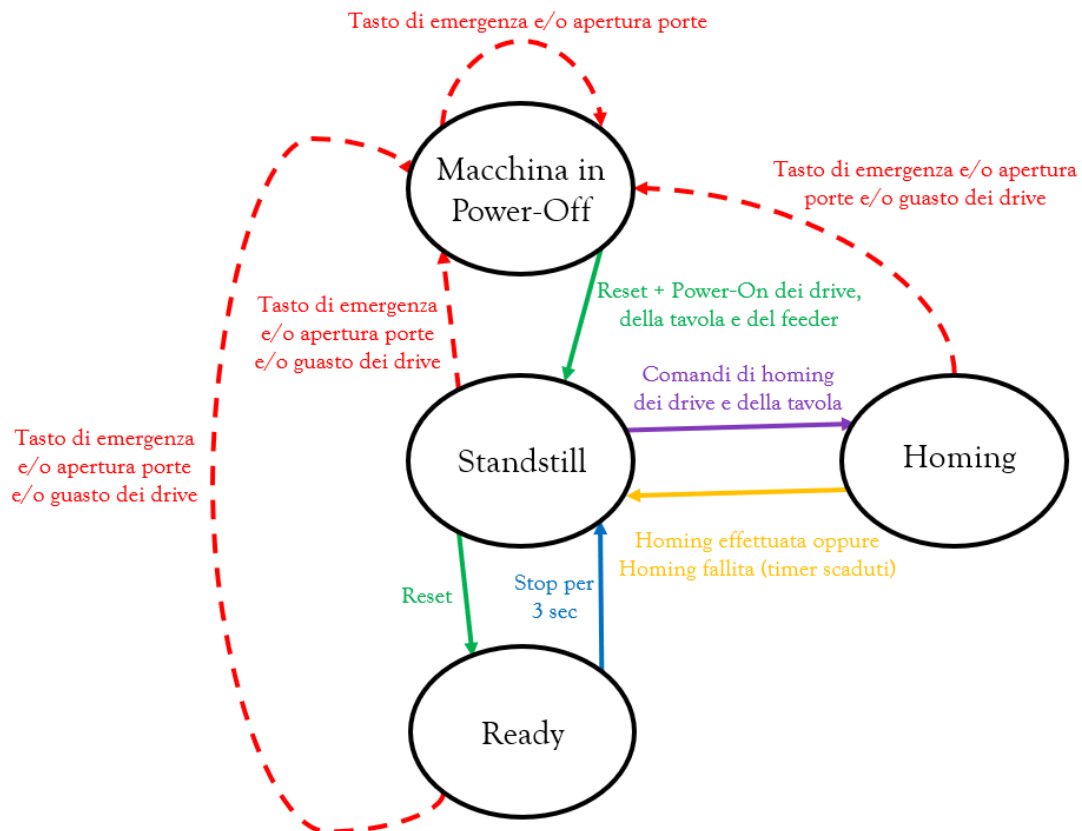


Figura 6.2 Diagramma degli stati della macchina nella fase di preparazione dei dispositivi

Si parte dallo stato di *Macchina in Power-Off* e si preme il tasto Reset; se si ha a disposizione il segnale di STO concesso dal modulo di sicurezza, ad indicare che si è in una condizione di funzionamento sicuro, si può dare un comando di Power-On ai dispositivi per fornirgli l'alimentazione. Una volta ricevuto l'acknowledge da parte di questi ultimi e se non si verificano situazioni di emergenza o guasti agli azionamenti (comunicati dagli stessi), si passa allo stato di *Standstill*. In questa condizione si procede a fornire il Feeder Command all'alimentatore in modo da abilitarlo e renderlo pronto all'utilizzo; è richiesto un tempo pari a 3 secondi per ottenere il segnale di acknowledge da parte di tale dispositivo. A questo punto si può passare allo stato di Homing lanciando comandi di inizializzazione dapprima dei drive e successivamente della tavola principale. Si possono verificare le seguenti casistiche:

- Homing fallito a causa di un allarme (tasto di emergenza premuto e/o apertura delle porte durante il funzionamento e/o guasto negli azionamenti). Si passa allo stato di *Macchina in Power-Off* e si effettuano nuovamente tutti i passi descritti sino ad ora;

- Processo di inizializzazione fallito, perché non portato a termine entro uno dei timer stabiliti per esso. Si ritorna in *Standstill* e si ripete la procedura;
- Homing andato a buon fine. Si ripassa in *Standstill*.

Nel caso di inizializzazione terminata con successo si porta il manipolatore nella configurazione di riposo con l'apposito comando. Una volta eseguita l'azione appena descritta, si procede muovendo la tavola nella posizione desiderata per essa. Se le due operazioni citate hanno esito positivo, l'operatore può premere il tasto di Reset abilitando in questo modo gli azionamenti e la tavola. Di conseguenza dallo stato di *Standstill* si passa a quello di *Ready* in cui tutti i dispositivi sono pronti e si può far partire il macchinario. In questo contesto, tuttavia, è possibile che accadano le due situazioni che seguono:

- si verifica un'emergenza o un guasto. Si passa così in *Macchina in Power-Off*, si ripetono gli step eseguiti sin ora senza però effettuare l'operazione di inizializzazione e senza comandare il movimento del manipolatore e della tavola verso le posizioni di partenza;
- viene premuto il tasto di Stop per 3 secondi e viene tolta, di conseguenza, l'abilitazione ai vari dispositivi. Si torna in *Standstill* e per ripassare a *Ready* è necessario soltanto muovere il robot nella configurazione di riposo e premere nuovamente il tasto Reset.

È necessario precisare che, in generale, il guasto degli azionamenti comporta il passaggio allo stato iniziale in cui i dispositivi sono spenti, ma non provoca la perdita del comando di STO fornito al PLC dal modulo di sicurezza. Il verificarsi di condizioni di allarme, come la pressione del tasto di arresto di emergenza o l'apertura delle porte dell'impianto, invece fa decadere tale segnale, indicando che la macchina si trova in uno stato non sicuro.

6.2.3 Fase implementativa

Il codice che implementa la fase di preparazione dei dispositivi è riportato nell'Appendice. Esso si basa su un costrutto condizionale CASE regolato dalla variabile intera *MachineState* inizializzata ad 1. In base al valore che quest'ultima assume si finisce in uno degli stati descritti in precedenza come mostrato in Figura 6.3.

```
(* 1 -> POWER-OFF
   2 -> STANDSTILL
   3 -> HOMING
   4 -> READY *)
```

Figura 6.3 – Stati corrispondenti ai valori della variabile *Machine State*

Ogni casistica dell'istruzione rappresenta, quindi, un particolare stato ed è caratterizzata da clausole IF che permettono di modellare tutte le condizioni e le situazioni che si verificano in essa.

Nel seguito sono riportati alcuni screenshot provenienti dal codice in questione utili alla comprensione della logica adottata per la scrittura di esso.

È stato utilizzato l'oggetto DUT (Data Unit Types), il quale rappresenta un tipo di dato definito dall'utente, per creare strutture che implementino:

- i comandi inviati ai dispositivi;
- gli acknowledge forniti dai dispositivi;
- gli input della macchina.

I tre elementi appena citati sono rappresentati mediante variabili booleane.

Il tipo di struttura *DUT_Devices_Cmd*, riportato in Figura 6.4, è impiegato per la definizione dei command da dare ai dispositivi. Questi ultimi rappresentano i campi della STRUCT e si articolano in comandi di:

- alimentazione (*PowerOn*);
- inizializzazione (*Homing*);
- movimento del manipolatore nella configurazione di riposo (*MoveManipulatorInRestPosition*);
- movimento della tavola nella posizione desiderata per essa (*MoveTableInPosition*);
- abilitazione dell'alimentatore (*FeederCommand*).

```

TYPE DUT_Devices_Cmd :
STRUCT
    PowerOn : BOOL;
    Homing : BOOL;
    MoveManipulatorInRestPosition : BOOL;
    MoveTableInPosition : BOOL;
    FeederCommand : BOOL;
END_STRUCT
END_TYPE

```

Figura 6.4 – DUT per i comandi inviati ai dispositivi

Il Data Unit Type *DUT_Devices_Ack*, mostrato in Figura 6.5, implementa i seguenti acknowledge provenienti dai dispositivi:

- dispositivo alimentato (*PoweredOn* e *FeederOn*);
- azionamenti e tavola inizializzati (*HomingDrive1Done*, *HomingDrive2Done*, *HomingTableDone* rispettivamente);
- manipolatore mosso nella configurazione di riposo (*ManipulatorMovedInRestPosition*);
- tavola movimentata nella posizione desiderata per essa (*TableMovedInPosition*);
- alimentatore abilitato (*FeederCommandDone*).

```

TYPE DUT_Devices_Ack :
STRUCT
    PoweredOn : BOOL;
    HomingDrive1Done : BOOL;
    HomingDrive2Done : BOOL;
    HomingTableDone : BOOL;
    ManipulatorMovedInRestPosition : BOOL;
    TableMovedInPosition : BOOL;
    FeederOn : BOOL;
    FeederCommandDone : BOOL;
END_STRUCT
END_TYPE

```

Figura 6.5 – DUT per gli acknowledge forniti dai dispositivi

Il tipo di dato *DUT_Input*, riportato in Figura 6.6, permette di rappresentare alcuni degli input che la macchina riceve durante questa fase. Questi ultimi si dividono in segnali di:

- allarme (*StopImmediately* e *DoorOpening*);
- guasto (*Drive1Fault* e *Drive2Fault*);
- ripristino delle condizioni di sicurezza o di operatività (*Reset*);
- disabilitazione dei dispositivi (*Stop*).

Un ulteriore ingresso è rappresentato dal comando di STO che il PLC riceve dal modulo di sicurezza nel caso in cui la macchina si trova in uno stato sicuro.

```

TYPE DUT_Input :
STRUCT
  Stop                : BOOL; //PULSANTE DI STOP PREMUTO
  Reset               : BOOL; //PULSANTE DI RESET PREMUTO
  DoorOpening         : BOOL; //APERTURA PORTE
  StopImmediately     : BOOL; //TASTO DI EMERGENZA
  Drive1Fault         : BOOL; //GUASTO AZIONAMENTO 1
  Drive2Fault         : BOOL; //GUASTO AZIONAMENTO 2
  STO                 : BOOL; //COMANDO DI STO DAL MODULO DI SICUREZZA
END_STRUCT
END_TYPE

```

Figura 6.6 – DUT per gli input della macchina

Il tasto di Reset è implementato attraverso l'istanza *ResetTRIG* del function block standard *R_TRIG* di Codesys. Quest'ultimo rileva un fronte di salita sul proprio ingresso *CLK*, che in questo caso viene uguagliato al campo *Reset* dell'istanza *InputMachine* di *DUT_Input*, e attiva la sua uscita Q come mostrato in Figura 6.7 [41].

```

InputMachine : DUT_Input;
ResetTRIG : R_TRIG;
ResetDone : BOOL;

ResetTRIG(CLK := InputMachine.Reset);
ResetDone := ResetTRIG.Q;

```

Figura 6.7 – Implementazione del tasto Reset

Per gli altri segnali e comandi che richiedono dei timer sono state create istanze del Function Block standard TON, il quale permette di implementare un ritardo di accensione. Quando l'input *IN* diventa TRUE, trascorrerà un determinato tempo, specificato in *PT*, prima che l'output *Q* diventi anch'esso TRUE [42]. In Figura 6.8 è riportata la creazione del timer per il tasto di Stop.

```
TON_Stop : TON;  
StopReady : BOOL;
```

```
TON_Stop(IN:= InputMachine.Stop, PT:=T#3S);  
StopReady := TON_Stop.Q;
```

Figura 6.8 – Implementazione del timer per il tasto di Stop

Il codice è stato testato in simulazione dal momento che, a causa di ritardi nella consegna dei componenti elettronici, l'impianto non è ancora pronto per il funzionamento generale.

6.3 Movimento del manipolatore PRRRP 2 DOF

Il manipolatore PRRRP a due gradi di libertà oggetto di studio, come già specificato in precedenza, è adibito allo svolgimento di operazioni di pick and place. Nell'esecuzione di tali attività esso descrive ciclicamente una traiettoria a C come mostrato in Figura 6.9.

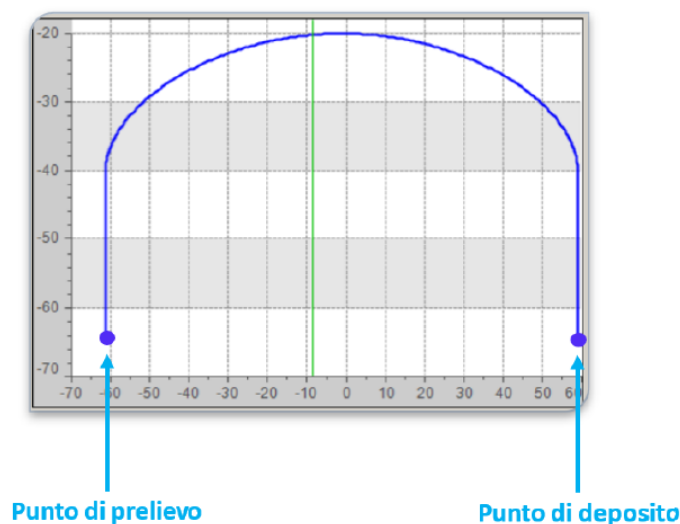


Figura 6.9 – Traiettoria a C del manipolatore PRRRP per il pick and place (da [25])

L'intento del lavoro di tesi è quello di realizzare un software che permetta al robot di svolgere il movimento previsto per esso, ricorrendo però all'utilizzo delle funzioni della libreria *ML_Robot*, creata dalla compagnia Bosch Rexroth. Nel conseguimento di tale obiettivo sono stati esaminati gli eventuali vantaggi che l'uso di questi nuovi strumenti possa garantire rispetto al codice preesistente per la movimentazione del manipolatore.

6.3.1 Function Block di libreria

La libreria di robotica *ML_Robot* della Bosch Rexroth contiene blocchi funzione per la pianificazione della traiettoria e per il controllo del moto dei robot manipolatori. La maggior parte dei suoi FB hanno le seguenti caratteristiche comuni:

- un ingresso per l'attivazione;
- un'uscita per indicare che il Function Block sta svolgendo il proprio task;
- un'uscita per indicare che l'elaborazione è avvenuta correttamente;
- uscite che segnalino condizioni di errore.

Tre blocchi funzione di fondamentale importanza sono *MC_SetKinTransform*, *ML_GroupAllAxes* e *ML_UngroupAllAxes* (riportati in Tabella 6.2). Essi appartengono alla categoria di Function Block che permettono di configurare una cinematica custom.

Tabella 6.2 – Function Block per configurare una cinematica

<i>MC_SetKinTransform</i>	Registra una trasformazione di coordinate di posizione definita dall'utente per una cinematica custom ricevuta in ingresso. Nello specifico, prende come argomento un Function Block creato appositamente per implementare l'interfaccia <i>MC_KIN_REF</i> . Quest'ultima permette di definire la trasformazione di coordinate attraverso i due metodi <i>Forward</i> ed <i>Inverse</i> , i quali realizzano rispettivamente la cinematica diretta ed inversa di posizione.
<i>ML_GroupAllAxes</i>	Aggiunge gli assi dei giunti attuati del robot ad una cinematica custom. Essi vengono raggruppati e possono essere mossi, di conseguenza, in modo coordinato.

<i>ML_UngroupAllAxes</i>	Rimuove gli assi dei giunti attuati del robot da una cinematica custom. In questo modo essi possono essere movimentati singolarmente ed in maniera indipendente l'uno rispetto all'altro.
---------------------------------	---

Un blocco funzione che consente di ottenere informazioni sul movimento del manipolare è *MC_GroupReadActualPosition*, il quale legge la posizione attuale della cinematica che gli viene passata come input. Come ulteriore ingresso, richiede la specifica del sistema di coordinate dell'end-effector (Base coordinate system) oppure degli assi dei giunti (Axis coordinate system). Sulla base del sistema di coordinate scelto restituisce rispettivamente la posizione corrente dell'organo terminale o dei giunti.

Tra i Function Block di libreria che consentono di movimentare il manipolatore è di particolare interesse il *MC_MoveLinearAbsolute*. Esso permette di realizzare un moto a linea retta spostando la cinematica dall'ultima posizione comandata dell'end-effector ad una posizione di destinazione da specificare. In Figura 6.10 è riportato il blocco che elenca gli ingressi (sulla sinistra) e le uscite (sulla destra) di tale blocco funzione, i quali sono descritti e spiegati in Tabella 6.3.

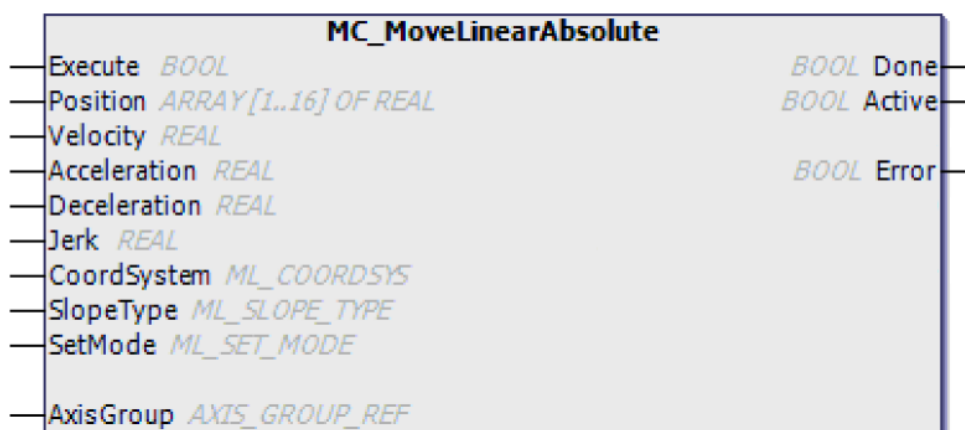


Figura 6.10 – Input e output di *MC_MoveLinearAbsolute* (da [43])

Tabella 6.3 - Ingressi e uscite di MC_MoveLinearAbsolute

Ingressi	
<i>Execute</i>	Abilita l'elaborazione del blocco funzione
<i>Position</i>	È un array i cui elementi rappresentano i valori delle coordinate della posizione di target del moto
<i>Velocity</i>	Rappresenta la velocità massima del percorso
<i>Acceleration</i>	Rappresenta l'accelerazione massima del percorso
<i>Jerk</i>	Rappresenta il jerk massimo del percorso
<i>CoordSystem</i>	Indica il sistema di coordinate della posizione di destinazione. Esso può essere il Base coordinate system o l'Axis coordinate system a seconda che si vogliano definire rispettivamente le coordinate dell'organo terminale oppure quelle dei giunti
<i>SlopeType</i>	Specifica il modo in cui avviene la pianificazione della traiettoria. Se assume il valore <i>ST_BLOCK_SLOPE</i> viene adottato il profilo trapezoidale di velocità, mentre se viene valorizzato a <i>ST_CONTINUOUS_SLOPE</i> si utilizzano le spline cubiche come funzioni interpolanti
<i>SetMode</i>	Indica come debba avvenire l'unione di due movimenti lineari successivi. Se assume il valore <i>SM_TO</i> , il punto di destinazione viene raggiunto diminuendo la velocità fino a zero e soltanto dopo viene avviato il comando di moto seguente. Se viene valorizzato a <i>SM_VIA</i> non viene raggiunta propriamente la posizione di target, ma si genera un moto continuo tra il punto iniziale del primo comando e quello finale del secondo comando
<i>AxisGroup</i>	È un riferimento alla cinematica custom definita
Uscite	
<i>Done</i>	Indica che il moto è stato eseguito con successo
<i>Active</i>	Specifica che il moto è in esecuzione
<i>Error</i>	Segnala una condizione di errore

La pianificazione di una traiettoria del manipolatore si realizza unendo in successione più chiamate di un'istanza del blocco funzione *MC_MoveLinearAbsolute*. Il parametro di *SlopeType*, come già specificato nella Tabella 6.3, permette di stabilire l'algoritmo di pianificazione da utilizzare per interpolare i punti, mentre l'ingresso *CoordSystem* consente di definire se la procedura avviene nello spazio dei giunti o nello spazio operativo [43].

I Function Block descritti in questo paragrafo sono stati impiegati per la realizzazione del software che permette di gestire il moto del manipolatore. Il codice è riportato parzialmente nell'Appendice per questioni di riservatezza.

6.3.2 Algoritmo di gestione del moto del manipolatore

L'algoritmo ideato per gestire il moto del robot PRRRP a due gradi di libertà è caratterizzato dai seguenti passi:

1. Creazione di una nuova cinematica custom;
2. Assegnazione degli assi dei giunti attuati del manipolatore alla cinematica creata;
3. Definizione di una trasformazione di coordinate "user-defined" e relativa implementazione della cinematica diretta e inversa di posizione;
4. Registrazione della trasformazione di coordinate;
5. Raggruppamento degli assi per il moto coordinato di essi;
6. Definizione dell'operazione ciclica di pick and place.

I primi due passaggi sono quelli relativi alla creazione e alla configurazione della cinematica del manipolatore nell'ambiente di programmazione IndraWorks Engineering. Lo step 3 si effettua implementando, con un apposito Function Block, l'interfaccia *MC_KIN_REF* e i suoi metodi *Forward* ed *Inverse* in cui sono riportate le formule per il calcolo della cinematica diretta ed inversa di posizione descritti nel capitolo. I due passi successivi prevedono di richiamare istanze rispettivamente delle funzioni *MC_SetKinTransform* e *ML_GroupAllAxes* descritte in precedenza. L'ultimo passaggio si realizza richiamando più volte un'istanza creata del blocco funzione

MC_MoveLinearAbsolute. Ogni chiamata riceve in ingresso una diversa posizione di target da raggiungere per l'end-effector o per i giunti. Combinando e concatenando fra di loro le diverse chiamate, si consente al manipolatore di descrivere ciclicamente la traiettoria a C durante il suo movimento e quindi di eseguire l'operazione di pick and place richiesta per esso.

6.3.3 Vantaggi ottenibili

L'adozione dei blocchi funzione della libreria *ML_Robot* nella realizzazione del software per la gestione del moto del manipolatore comporta i seguenti vantaggi:

- ✓ Agevolazione nella programmazione. Non è necessario creare algoritmi di pianificazione della traiettoria ad hoc in quanto, come specificato in precedenza, quest'ultima operazione è implementabile attraverso la combinazione di chiamate ad istanze di *MC_MoveLinearAbsolute*;
- ✓ Snellimento del codice. Si tratta per lo più di richiamare istanze delle funzioni di libreria;
- ✓ Possibilità di modificare real time i punti della traiettoria definita, senza interrompere il funzionamento del manipolatore.

L'ultimo punto a favore tra quelli citati è di fondamentale rilevanza per la flessibilità del sistema. Infatti la soluzione precedente, per variare la traiettoria, necessita di eseguire i passi che seguono:

- togliere l'alimentazione agli azionamenti del manipolatore;
- scegliere i nuovi punti del percorso da far eseguire all'end-effector;
- definire le camtable, ovvero quelle tabelle che riportano i valori di posizione e di velocità assunti dall'organo terminale durante una rotazione da 0° a 360° (corrispondente ad un ciclo di pick and place) di un asse master virtuale associato al sistema robotico. Tale asse è sincronizzato ad un asse master principale dell'impianto (che coincide con il motore che fa muovere la tavola portapezzi) e viene impiegato per garantire il sincronismo tra il manipolatore e il resto del macchinario (che avviene grazie all'utilizzo di camme elettroniche) [25];

- determinare le camtable delle posizioni e velocità dei giunti attraverso le formule di cinematica inversa per il controllo del robot;
- fornire l'alimentazione e l'abilitazione ai drive e far ripartire il ciclo di funzionamento.

La nuova soluzione software realizzata, invece, evita l'utilizzo delle camtable per la determinazione della traiettoria e non richiede quindi la necessità di fermare l'impianto per modificare la traiettoria da far eseguire al sistema robotico.

7 Fase di test

L'attività conclusiva del lavoro di tesi è il test sul robot reale del software ideato per la gestione del moto del manipolare PRRRP. Il programma sviluppato è stato caricato nel PLC del macchinario che assembla i componenti in plastica, al fine di verificarne il corretto funzionamento.

Nei paragrafi successivi vengono descritti:

- la fase di sperimentazione dell'azione di prelievo e deposito;
- un confronto prestazionale preliminare tra il codice preesistente e il codice ottimizzato per movimentare il robot.

7.1 Operazione di pick and place

Per implementare l'operazione di pick and place del manipolatore è stato utilizzato il tool grafico, mostrato in Figura 7.1, che consente di:

- richiamare più volte un determinato Function Block adibito alla movimentazione della cinematica (in questo caso il *MC_MoveLinearAbsolute* descritto nel capitolo precedente) semplicemente inserendo righe nella tabella apposita;
- concatenare in successione le diverse chiamate mediante parametro *JumpAfter*;
- leggere la posizione attuale dei due giunti (*JointCoord*) e dell'end-effector (*PathCoordinates*), ottenuta grazie all'utilizzo della funzione di libreria *MC_GroupReadActualPosition*.

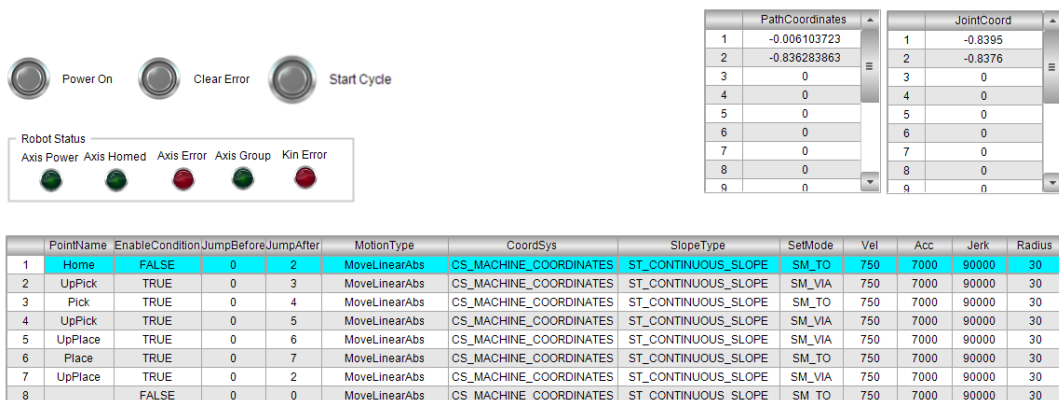


Figura 7.1 – Tool grafico per realizzare l'operazione di pick and place

Inizialmente sono stati scelti i punti di destinazione delle chiamate al *MC_MoveLinearAbsolute*, i quali sono riportati in Tabella 7.1 ed illustrati in Figura 7.2. Assegnando il valore *CS_MACHINE_COORDINATES* al parametro *CoordSys* di Figura 7.1, si specifica che tali punti rappresentano le coordinate di posizione dell'end-effector rispetto al sistema di riferimento cartesiano fisso del manipolatore, collocato sulla base.

Tabella 7.1 – Coordinate dei punti per il pick and place

Nome del punto	Coord. x	Coord. y	Descrizione
<i>Home</i>	0	-10	Posizione di fuori ingombro
<i>UpPick</i>	-50	-25	Posizione di sopra-prelievo
<i>Pick</i>	-50	-84	Posizione di prelievo
<i>UpPlace</i>	48	-25	Posizione di sopra-deposito
<i>Place</i>	48	-81	Posizione di deposito

Come indicato nel Capitolo 6, la procedura di pianificazione della traiettoria si realizza attraverso l'unione dei vari comandi di movimento. In questo caso, dal momento che sono stati definiti i punti che l'organo terminale deve attraversare, essa avviene nello spazio operativo. Ponendo il parametro *SlopeType* pari a *ST_CONTINUOUS_SLOPE* (Figura 7.1) si assume che l'interpolazione delle coordinate di posizione debba essere effettuata mediante spline cubiche. Inoltre, è da precisare che l'argomento *SetMode* per la maggior parte dei punti target viene settato a *SM_TO*, ad indicare che essi debbano essere effettivamente raggiunti. Per *UpPick* e *UpPlace*, invece, tale argomento viene valorizzato con *SM_VIA* per fare in modo che i punti specificati non vengano propriamente toccati, ma che si generi una connessione curvilinea tra tratti lineari successivi, la quale passa in un intorno dei punti. Il parametro *Radius* indica a quale distanza dal punto finale del singolo comando di moto debba avvenire il processo di raccordo appena descritto [43]. Infine *Vel*, *Acc* e *Jerk* indicano rispettivamente i valori massimi di velocità, accelerazione e jerk che il manipolatore può assumere durante la percorrenza del percorso definito. Alle ultime quattro variabili descritte sono

stati assegnati valori tarati attraverso delle prove sperimentali nella fase di messa in servizio del robot.

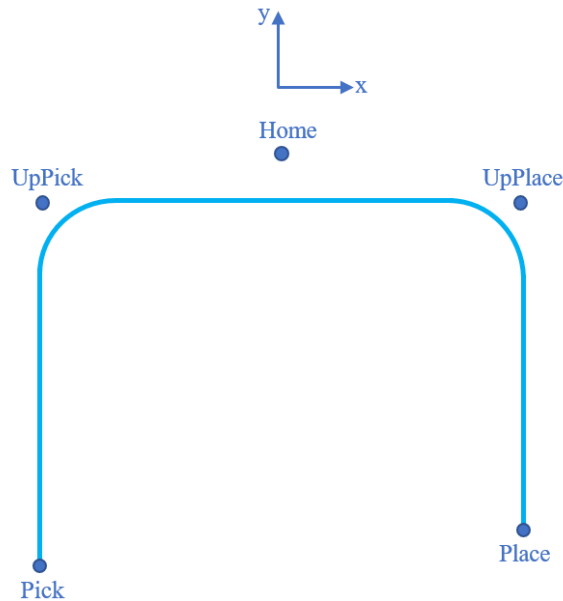


Figura 7.2 – Rappresentazione schematica della traiettoria definita

Per avviare l'esecuzione della sequenza di comandi di moto, si preme inizialmente il pulsante di *Power On* ed in seguito quello di *Start Cycle*. Una volta condotto il manipolatore nella posizione di *Home* attraverso una prima chiamata a *MC_MoveLinearAbsolute*, vengono eseguiti i seguenti movimenti:

UpPick → *Pick* → *UpPick* → *UpPlace* → *Place* → *UpPlace*

Dopodiché il ciclo di lavoro effettivo del sistema robotico è rappresentato dalla seguente successione di spostamenti:

Pick → *UpPick* → *UpPlace* → *Place* → *UpPlace* → *UpPick* → *Pick*

7.2 Confronto prestazionale codice preesistente-codice ottimizzato

È stato realizzato un confronto preliminare tra il software già esistente e quello scritto servendosi dei blocchi funzione di libreria. È stato impiegato un manipolatore PRRRP per l'esecuzione del primo codice ed un altro, della stessa tipologia e appartenente al medesimo impianto, su cui è stato eseguito il secondo programma. Per i due robot, mossi in contemporanea, la traiettoria è stata definita scegliendo gli stessi punti di

percorso. Sono stati tracciati i grafici della posizione dell'organo terminale, delle velocità dei due giunti e delle coppie applicate a questi ultimi, per entrambi i manipolatori. In Figura 7.3 ed in Figura 7.4 sono illustrati gli andamenti di tali grandezze ottenuti rispettivamente con il codice preesistente e con il codice ottimizzato. In entrambe le suddette figure i grafici sono organizzati come segue:

- nella parte superiore sono riportati i valori delle coordinate x e y dell'organo terminale;
- nella parte centrale sono raffigurate le velocità assunte dai due giunti;
- nella parte inferiore sono indicate le coppie applicate ai motori.

Vi sono inoltre due cursori in nero che evidenziano l'inizio e la fine di un singolo ciclo di funzionamento della macchina.

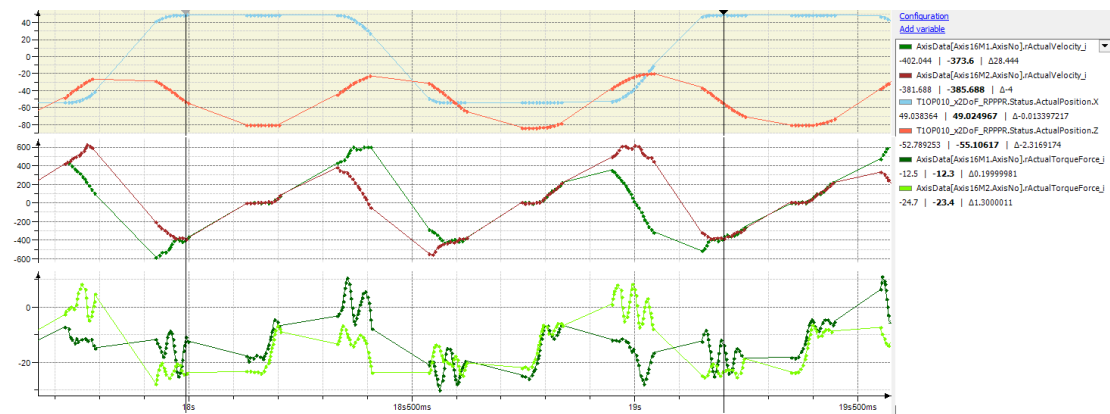


Figura 7.3 – Grafici ottenuti con il codice preesistente

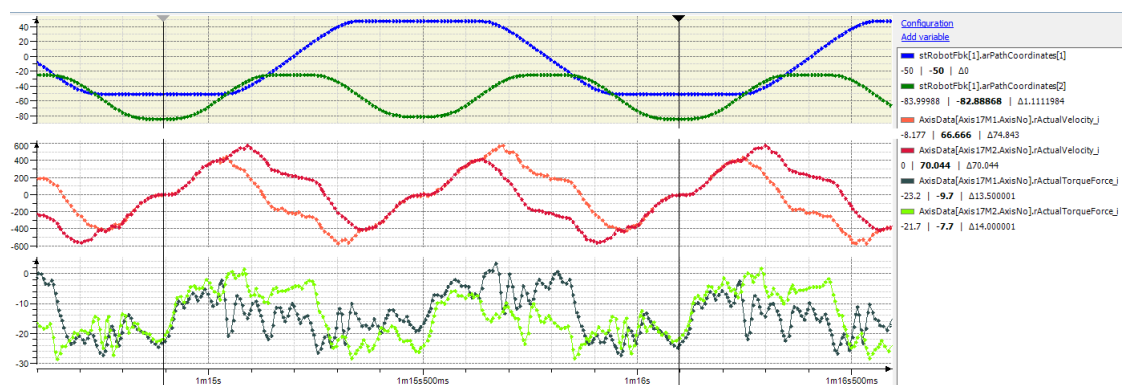


Figura 7.4 - Grafici ottenuti con il codice ottimizzato

Dall'analisi dei grafici si può notare che, nel caso del software scritto adottando le librerie di robotica (Figura 7.4), gli andamenti delle coordinate di posizione dell'end-effector sono caratterizzati da una regolarità puntuale; questa caratteristica si traduce in una regolarità negli andamenti delle velocità e delle coppie ai giunti e in un'assenza di sovraelongazione nelle velocità. Quest'ultimo aspetto è particolarmente evidente se si osservano i tracciati relativi alla velocità e all'accelerazione dell'end-effector riportati in Figura 7.5.

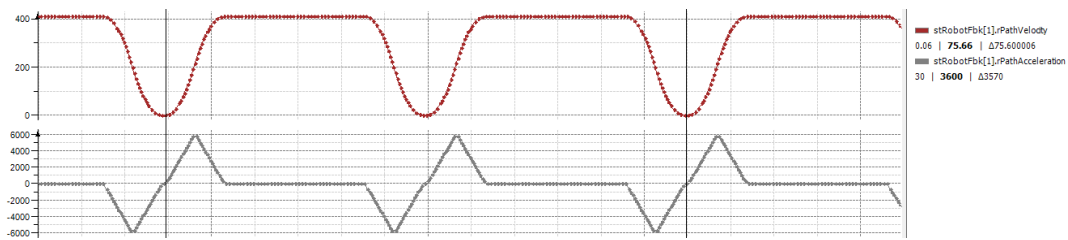


Figura 7.5 – Andamenti della velocità e dell'accelerazione dell'end-effector con codice ottimizzato

Inoltre, tra un ciclo di funzionamento e l'altro, gli andamenti si mantengono pressoché costanti e uniformi. Questo comportamento ottimale non si riscontra nei grafici che derivano dall'utilizzo del codice preesistente (Figura 7.3); in questo caso, infatti, si possono notare delle irregolarità negli andamenti delle grandezze analizzate, un comportamento meno uniforme tra un ciclo e il successivo e la presenza di sovraelongazioni nelle velocità.

La motivazione di questi andamenti diversi risiede nella modalità di generazione della traiettoria che avviene adottando tecniche di calcolo differenti. Il codice di partenza proviene da un attento studio della traiettoria, che viene sviluppata su un editor di camma, ed è basato su un algoritmo di interpolazione dei punti di percorso mediante un polinomio di quinto grado. Il software ottimizzato, invece, adotta funzioni di libreria ideate appositamente per pianificare la traiettoria, le quali permettono di raccordare i punti attraverso la funzione interpolante spline cubica. Un'altra giustificazione è rappresentata dal fatto che le differenti funzioni di libreria per movimentare il sistema robotico, impiegate nei due codici, gestiscono in maniera diversa il moto del manipolatore.

8 Conclusioni e sviluppi futuri

Il presente lavoro di tesi ha avuto come fase iniziale lo studio della geometria e della cinematica del manipolatore PRRRP a due gradi di libertà. Successivamente è stato scritto un codice ottimizzato per gestire il movimento del suddetto sistema robotico nelle operazioni di pick and place. In seguito è stato realizzato un programma che permette di preparare l'impianto all'utilizzo dei dispositivi della stazione di lavoro del robot in questione. Infine sono stati condotti dei test in simulazione del software appena descritto e delle prove sperimentali preliminari su campo del codice descritto in precedenza.

L'obiettivo principale è stato lo sviluppo di un nuovo codice per la gestione del moto del manipolatore, che consentisse di migliorare e ottimizzare quello precedente. L'approccio seguito, basato sull'impiego dei Function Block della libreria *ML_Robot*, ha consentito di ottenere vantaggi evidenti come la stesura di un codice più snello, costituito prevalentemente da chiamate a funzione, e la semplificazione del processo di programmazione, in quanto non è stato necessario implementare algoritmi specifici per pianificare le traiettorie. Inoltre, un altro punto a favore di notevole importanza, è la possibilità di modificare in linea il percorso stabilito da far seguire al manipolatore, senza la necessità di interrompere il funzionamento dell'impianto. Sono stati condotti in seguito test sperimentali che hanno dimostrato il corretto funzionamento del programma sviluppato. È stato realizzato successivamente un confronto preliminare tra il software attuale e quello di partenza per la gestione del movimento del robot: la comparazione dei risultati ottenuti ha mostrato che la nuova soluzione ha un comportamento migliore in termini di regolarità ed uniformità negli andamenti delle grandezze analizzate. Tale analisi ha consentito di confermare che l'utilizzo del codice scritto per il lavoro di tesi comporta diversi benefici.

In futuro si potrebbe pensare di utilizzare, al posto del *MC_MoveLinearAbsolute*, un blocco funzione di considerevole interesse della libreria *ML_Robot*, al fine gestire il movimento del manipolatore: il *ML_RobotMoveJumpAbsolute*. Quest'ultimo permette di realizzare l'operazione di pick and place con un solo comando di movimento, invece di definire diversi spostamenti lineari da raccordare. Ciò nonostante, il moto standard che esso realizza avviene in tre dimensioni e questo aspetto non lo rende attualmente adatto alla casistica in esame, in quanto il manipolatore in questione si muove su un

piano. Tuttavia si potrebbe pensare di modificare dei parametri specifici di default di tale Function Block, in modo da adattare il suo funzionamento al caso considerato nell'elaborato di tesi e valutare, di conseguenza, eventuali vantaggi che si possono ottenere dal suo impiego.

In seguito saranno testati su macchina i codici che implementano la fase di preparazione dei dispositivi e quella di movimentazione del manipolatore, al fine di verificarne il corretto funzionamento in un contesto più generale.

Appendice A

Nella presente appendice sono riportati la sezione di dichiarazione delle variabili e il corpo del programma principale del codice che permette di preparare l'impianto all'utilizzo dei dispositivi.

```
PROGRAM PlcProg
VAR
  MachineState : INT := 1; // STATO DELLA MACCHINA
  InputMachine : DUT_Input;
  ResetTRIG : R_TRIG;
  ResetDone : BOOL;
  Drive1Cmd : DUT_Devices_Cmd;
  Drive2Cmd : DUT_Devices_Cmd;
  ManipulatorCmd : DUT_Devices_Cmd;
  TableCmd : DUT_Devices_Cmd;
  FeederCmd : DUT_Devices_Cmd;
  Drive1Ack : DUT_Devices_Ack;
  Drive2Ack : DUT_Devices_Ack;
  ManipulatorAck : DUT_Devices_Ack;
  TableAck : DUT_Devices_Ack;
  FeederAck : DUT_Devices_Ack;
  TON_HomingDrive1 : TON;
  TON_HomingDrive2 : TON;
  TON_HomingTable : TON;
  TON_Feeder : TON;
  TON_Stop : TON;
  HomingDrive1Failed : BOOL;
  HomingDrive2Failed : BOOL;
  HomingTableFailed : BOOL;
  Homing : BOOL;
  ReadyToPowerOff : BOOL;
  ReadyToStandstill : BOOL;
  StopReady : BOOL;
END_VAR

(* 1 -> POWER-OFF
2 -> STANDSTILL
3 -> HOMING
4 -> READY *)
CASE MachineState OF
  1 : // MACCHINA IN POWER-OFF (ALIMENTATA MA NON ARMAT)
    ResetTRIG(CLK := InputMachine.Reset);
    ResetDone := ResetTRIG.Q;
    IF ResetDone AND NOT InputMachine.StopImmediately AND NOT InputMachine.DoorOpening THEN //SE IL RESET E' STATO EFFETTUATO, NON E' STATO PREMUTO IL TASTO DI EMERGENZA
      InputMachine.STO := TRUE; //E NON SONO STATE APERTE LE PORTE NO L'STO
    END_IF
    //DO LE ALIMENTAZIONI AI VARI DISPOSITIVI
    Drive1Cmd.PowerOn := InputMachine.STO;
    Drive2Cmd.PowerOn := InputMachine.STO;
    TableCmd.PowerOn := InputMachine.STO;
    FeederCmd.PowerOn := InputMachine.STO;
    InputMachine.Reset := FALSE;
    //SE RICEVO L'ACK DEI VARI DISPOSITIVI, SE HO L'STO E SE NON SI SONO VERIFICATI GUASTI AI MOTORI (SERVE PER NON AVERE CONTINUI CAMBIAMENTI DI STATO) PASSO ALLO STATO 2
    IF Drive1Ack.PoweredOn AND Drive2Ack.PoweredOn AND TableAck.PoweredOn AND FeederAck.FeederOn AND InputMachine.STO AND NOT InputMachine.Drive1Fault AND NOT InputMachine.Drive2Fault THEN
      MachineState := 2;
    END_IF
  2 : //MACCHINA IN STATO STANDSTILL
    //SE NON HO UN ARRESTO DI EMERGENZA E SE LE PORTE NON VENGONO APERTE DO IL FEEDER COMMAND
    IF NOT InputMachine.StopImmediately AND NOT InputMachine.DoorOpening THEN
      FeederCmd.FeederCommand := TRUE;
      TON_Feeder(IN:= FeederCmd.FeederCommand, PT:=T#3S); //TIMER PER L'ACK DEL FEEDER
      FeederAck.FeederCommandDone := TON_Feeder.Q; //ACKNOWLEDGE DI FEEDER COMMAND
      ELSE //RITORNO IN POWER-OFF (STATO 1) E PERDO L'STO
        FeederCmd.FeederCommand := FALSE;
        FeederAck.FeederOn := FALSE;
        Drive1Ack.PoweredOn := FALSE;
        Drive2Ack.PoweredOn := FALSE;
        TableAck.PoweredOn := FALSE;
        InputMachine.STO := FALSE;
        MachineState := 1;
      END_IF
    //SE NON SI PREME IL TASTO DI EMERGENZA, SE NON CI SONO GUASTI AI MOTORI, SE NON SI E' PASSATI DA READY A POWEROFF IN PRECEDENZA E SE L'HOMING NON VIENE EFFETTUATO
    IF NOT InputMachine.StopImmediately THEN //PERCHE' LA PRIMA VOLTA O PERCHE NON E' ANDATO A GOOD FINE): STANDSTILL -> HOMING
      IF NOT InputMachine.Drive1Fault AND NOT InputMachine.Drive2Fault THEN
        IF NOT ReadyToPowerOff AND NOT Homing THEN
          IF FeederAck.FeederCommandDone THEN //QUESTO IF SERVE A NON FAR ENTRARE SUBITO NELLO STATO 3 (SERVE PER DAR TEMPO ALL'ACQUISIZIONE DELL'ACK DEL FEEDER COMMAND)
            MachineState := 3; //VADO NELLO STATO DI HOMING
          END_IF
        ELSE
          //IN QUESTO CASO VADO IN POWER-OFF, MA NON PERDO L'STO (PERCHE' E' AVVENUTO UN GUASTO NEI MOTORI)
          MachineState := 1;
          FeederAck.FeederOn := FALSE;
          Drive1Ack.PoweredOn := FALSE;
          Drive2Ack.PoweredOn := FALSE;
          TableAck.PoweredOn := FALSE;
        END_IF
      ELSE
        //RITORNO IN POWER-OFF E PERDO L'STO (E' AVVENUTO UN ARRESTO DI EMERGENZA)
        InputMachine.STO := FALSE;
      END_IF
    END_IF
  3 :
  4 :
END_CASE
```

```

59 MachineState := 1;
60 FeederAck.FeederOn := FALSE;
61 DriveAck.PoweredOn := FALSE;
62 DriveAck.PoweredOn := FALSE;
63 TableAck.PoweredOn := FALSE;
64
65 END_IF
66
67 //SE LA MACCHINA HA EFFETTUATO L'HOMING, HO L'ACK DEL FEEDER COMMAND, SE NON SI E' PASSATI IN PRECEDENZA DA READY A POWER-OFF E SE NON SI E' PASSATI IN PRECEDENZA DA READY A STANDSTILL:
68 IF Homing AND FeederAck.FeederCommandDone AND NOT ReadyToPowerOff AND NOT ReadyToStandstill THEN //DA STANDSTILL VADO IN STATO DI READY
69 ManipulatorCmd.MoveManipulatorInRestPosition := TRUE; //DO IL COMANDO AL MANIPOLATORE DI ANDARE NELLA POSIZIONE DI RIPOSO
70 IF InputMachine.StopImmediately OR InputMachine.DriveFault OR InputMachine.DriveFault OR InputMachine.DriveFault OR InputMachine.DoorOpening THEN
71 //SE SI VERIFICA UNA DELLE CONDIZIONI SPECIFICATE NELLA CLAUSOLA IF CHE SEGUE TORNO IN POWER-OFF
72 IF InputMachine.StopImmediately OR InputMachine.DoorOpening THEN //IN PARTICOLARE SE A VERIFICARSI E' O L'ARRESTO DI EMERGENZA O L'APERTURA DELLE PORTE FERDO L'STO
73 InputMachine.STO := FALSE;
74
75 END_IF
76 ManipulatorCmd.MoveManipulatorInRestPosition := FALSE;
77 MachineState := 1;
78 FeederAck.FeederOn := FALSE;
79 DriveAck.PoweredOn := FALSE;
80 TableAck.PoweredOn := FALSE;
81
82 ELSE //ALTRIMENTI SE RICEVO L'ACK DEL COMANDO, DO IL COMANDO ALLA TAVOLA DI ANDARE IN POSIZIONE
83 IF ManipulatorAck.ManipulatorMovedInRestPosition THEN //SE IL MANIPOLATORE VA NELLA POSIZIONE DI RIPOSO, DO IL COMANDO ALLA TAVOLA DI ANDARE IN POSIZIONE
84 ManipulatorCmd.MoveManipulatorInRestPosition := FALSE;
85 TableCmd.MoveTableInPosition := TRUE;
86 //SE SI VERIFICA UNA DELLE CONDIZIONI SPECIFICATE NELLA CLAUSOLA IF CHE SEGUE TORNO IN POWER-OFF
87 IF InputMachine.StopImmediately OR InputMachine.DriveFault OR InputMachine.DriveFault OR InputMachine.DriveFault OR InputMachine.DoorOpening THEN
88 IF InputMachine.StopImmediately OR InputMachine.DoorOpening THEN //IN PARTICOLARE SE A VERIFICARSI E' O L'ARRESTO DI EMERGENZA O L'APERTURA DELLE PORTE FERDO L'STO
89 InputMachine.STO := FALSE;
90
91 END_IF
92 TableCmd.MoveTableInPosition := FALSE;
93 MachineState := 1;
94 FeederAck.FeederOn := FALSE;
95 DriveAck.PoweredOn := FALSE;
96
97 END_IF
98
99 DriveAck.PoweredOn := FALSE;
100 TableAck.PoweredOn := FALSE;
101
102 ELSE //SE RICEVO L'ACK DALLA TAVOLA, POSSO ABILITARE I DISPOSITIVI PER PASSARE DA STANDSTILL A READY (STATO 4) TRAMITE IL TASTO RESET
103 IF TableAck.TableMovedInPosition THEN
104 ResetTRIG(CLK := InputMachine.Reset);
105 ResetDone := ResetTRIG.Q;
106 IF ResetDone THEN
107 MachineState := 4; //VADO ALLO STATO DI READY
108 InputMachine.Reset := FALSE;
109
110 END_IF
111 END_IF
112
113 END_IF
114
115 // LA CLAUSOLA IF CHE SEGUE SERVE PER I SEGUENTI PASSAGGI DI STATO: READY -> POWER-OFF -> STANDSTILL -> READY (IN QUESTO CASO HO CHE L'HOMING E' STATO EFFETTUATO IN PRECEDENZA)
116 IF ReadyToPowerOff AND FeederAck.FeederCommandDone THEN //SE E' STATO EFFETTUATO IL PASSAGGIO DA READY A POWER-OFF IN PRECEDENZA E SE HO RICEVUTO L'ACK DEL FEEDER COMMAND
117 IF NOT InputMachine.DriveFault AND NOT InputMachine.DriveFault THEN
118 ResetTRIG(CLK := InputMachine.Reset); //SE NON SI VERIFICANO LE CONDIZIONI INDICATE NEI DUE IF PRECEDENTI, POSSO PREMERE RESET E PASSARE ALLO STATO 4 (QUINDI DA STANDSTILL A READY)
119 ResetDone := ResetTRIG.Q;
120 IF ResetDone THEN
121 ReadyToPowerOff := FALSE;
122 MachineState := 1; //TORNO ALLO STATO READY
123 InputMachine.Reset := FALSE;
124
125 END_IF
126 ELSE MachineState := 1; //RITORNO IN POWER-OFF E FERDO L'STO (E' AVVENUTO UN ARRESTO DI EMERGENZA O APERTURA PORTE)
127 InputMachine.STO := FALSE;
128 FeederAck.FeederOn := FALSE;
129 DriveAck.PoweredOn := FALSE;
130 DriveAck.PoweredOn := FALSE;
131 TableAck.PoweredOn := FALSE;
132
133 END_IF
134
135 END_IF
136
137 //SE DA READY SONO PASSATO A STANDSTILL TRAMITE IL TASTO DI STOP E SE IL FEEDER COMMAND E' STATO DATO
138 IF ReadyToStandstill AND FeederAck.FeederCommandDone THEN
139 IF NOT InputMachine.DriveFault AND NOT InputMachine.DriveFault THEN
140 IF NOT InputMachine.DoorOpening AND NOT InputMachine.StopImmediately THEN
141 ReadyToStandstill := FALSE;
142 ManipulatorCmd.MoveManipulatorInRestPosition := TRUE; //SE NON SI VERIFICANO LE CONDIZIONI DEI DUE IF PRECEDENTI DO IL COMANDO PER MUOVERE IL MANIPOLATORE NELLA POSIZIONE DI RIPOSO
143 IF ManipulatorAck.ManipulatorMovedInRestPosition THEN //SE RICEVO L'ACK DEL COMANDO, POSSO PREMERE RESET E RITORNARE A READY (STATO 4)
144 ResetTRIG(CLK := InputMachine.Reset);
145 ResetDone := ResetTRIG.Q;
146 IF ResetDone THEN
147 MachineState := 4;
148 InputMachine.Reset := FALSE;
149
150 END_IF
151 END_IF
152 ELSE MachineState := 1; //RITORNO ALLO STATO 1 E FERDO L'STO
153 InputMachine.STO := FALSE;
154 FeederAck.FeederOn := FALSE;
155 DriveAck.PoweredOn := FALSE;
156 DriveAck.PoweredOn := FALSE;
157 TableAck.PoweredOn := FALSE;
158
159 END_IF
160 ELSE MachineState := 1; //RITORNO ALLO STATO 1 SENZA PERDERE L'STO
161 FeederAck.FeederOn := FALSE;
162 DriveAck.PoweredOn := FALSE;
163 DriveAck.PoweredOn := FALSE;
164 TableAck.PoweredOn := FALSE;
165
166 END_IF
167
168 END_IF

```

```

168
169
170 3 : //FASE DI HOMING
171
172 IF NOT InputMachine.StopImmediately AND NOT InputMachine.DoorOpening THEN //SE NON SI VERIFICA UN ARRESTO DI EMERGENZA O L'APERTURA DELLE PORTE
173 IF NOT InputMachine.Drive1Fault AND NOT InputMachine.Drive2Fault THEN //SE NON SI VERIFICA UN GUASTO AI MOTORI
174 //DO IL COMANDO DI HOMING AI 2 DRIVE
175 Drive1Cmd.Homing := TRUE;
176 Drive2Cmd.Homing := TRUE;
177 TOH_HomingDrive1(IN:= Drive1Cmd.Homing, PT:=T4305); //TIMER PER VALUTARE IL FALLIMENTO DELL'OPERAZIONE DI HOMING DEL DRIVE 1
178 TOH_HomingDrive2(IN:= Drive2Cmd.Homing, PT:=T4305); //TIMER PER VALUTARE IL FALLIMENTO DELL'OPERAZIONE DI HOMING DEL DRIVE 2
179 HomingDriveFailed := TOH_HomingDrive1.Q AND NOT Drive1Ack.HomingDrive1Done; //HOMING DRIVE 1 FALLITO PERCHE' E' SCADUTO IL TIMER
180 HomingDriveFailed := TOH_HomingDrive2.Q AND NOT Drive2Ack.HomingDrive2Done; //HOMING DRIVE 2 FALLITO PERCHE' E' SCADUTO IL TIMER
181 IF HomingDriveFailed OR Drive1Ack.HomingDrive1Done THEN //SERVE PER FALSIFICARE IL COMANDO DI HOMING DEL DRIVE 1
182 Drive1Cmd.Homing := FALSE;
183 END_IF
184 IF HomingDriveFailed OR Drive2Ack.HomingDrive2Done THEN //SERVE PER FALSIFICARE IL COMANDO DI HOMING DEL DRIVE 2
185 Drive2Cmd.Homing := FALSE;
186 END_IF
187 IF Drive1Ack.HomingDrive1Done AND Drive2Ack.HomingDrive2Done THEN //SE HO RICEVUTO L'ACK DA PARTE DI ENTRAMBI I COMANDI DI HOMING 2 DEI DRIVE DO IL COMANDO DI HOMING ALLA TAVOLA
188 TableCmd.Homing := TRUE;
189 TOH_HomingTable(IN:= TableCmd.Homing, PT:=T4305); //TIMER PER VALUTARE IL FALLIMENTO DELL'OPERAZIONE DI HOMING DELLA TAVOLA
190 HomingTableFailed := TOH_HomingTable.Q AND NOT TableAck.HomingTableDone; //HOMING TAVOLA FALLITO PERCHE' E' SCADUTO IL TIMER
191 IF HomingTableFailed OR TableAck.HomingTableDone THEN //SERVE PER FALSIFICARE IL COMANDO DI HOMING DELLA TAVOLA
192 TableCmd.Homing := FALSE;
193 END_IF
194 IF TableAck.HomingTableDone THEN //SE NON FALLISCE L'HOMING DELLA TAVOLA, HO COMPLETATO L'HOMING E TORNO IN STANDSTILL
195 Homing := TRUE;
196 MachineState := 2;
197 ELSEIF HomingTableFailed THEN //ALTRIMENTI TORNO IN STANDSTILL CON L'HOMING FALLITO (A CAUSA DELLA TAVOLA)
198 Homing := FALSE;
199 MachineState := 2;
200 END_IF
201 ELSEIF HomingDrive1Failed OR HomingDrive2Failed THEN //ALTRIMENTI TORNO IN STANDSTILL CON L'HOMING FALLITO (A CAUSA DEI DRIVE)
202 Homing := FALSE;
203 MachineState := 2;
204 END_IF
205 ELSE //ALTRIMENTI SE SI VERIFICA UN GUASTO AI MOTORI TORNO ALLO STATO 1
206 Homing := FALSE;
207 Drive1Cmd.Homing := FALSE;
208 Drive2Cmd.Homing := FALSE;
209 TableCmd.Homing := FALSE;
210 MachineState := 1;
211 FeederAck.FeederOn := FALSE;
212 Drive1Ack.PoweredOn := FALSE;
213 Drive2Ack.PoweredOn := FALSE;
214 TableAck.PoweredOn := FALSE;
215 END_IF
216 ELSE //ALTRIMENTI SE SI VERIFICA UN ARRESTO DI EMERGENZA O L'APERTURA DELLE PORTE VADO IN POWER-OFF E PERDO L'ISTO
217 Homing := FALSE;
218 Drive1Cmd.Homing := FALSE;
219 Drive2Cmd.Homing := FALSE;
220 TableCmd.Homing := FALSE;
221 InputMachine.STO := FALSE;
222 MachineState := 1;
223 FeederAck.FeederOn := FALSE;
224 Drive1Ack.PoweredOn := FALSE;
225 Drive2Ack.PoweredOn := FALSE;
226 TableAck.PoweredOn := FALSE;
227 END_IF
228
229 4 : // MACCHINA IN STATO DI READY
230
231 //SERVE PER MODELLARE IL PASSAGGIO DI STATO READY -> POWEROFF
232 IF InputMachine.StopImmediately OR InputMachine.Drive1Fault OR InputMachine.Drive2Fault OR InputMachine.DoorOpening THEN
233 IF InputMachine.StopImmediately OR InputMachine.DoorOpening THEN //IN PARTICOLARE SE A VERIFICARSI E' UNA DELLE DUE CONDIZIONI DI QUESTO IF PERDO ANCHE L'ISTO (OLTRÈ A TORNARE ALLO STATO 1)
234 InputMachine.STO := FALSE;
235 END_IF
236 ReadyToPowerOff := TRUE;
237 MachineState := 1;
238 FeederAck.FeederOn := FALSE;
239 Drive1Ack.PoweredOn := FALSE;
240 Drive2Ack.PoweredOn := FALSE;
241 TableAck.PoweredOn := FALSE;
242 END_IF
243 //SE SI PREME STOP PER 3 SECONDI (VADO DA READY A STANDSTILL)
244 TOH_Stop(IN:= InputMachine.Stop, PT:=T435);
245 StopReady := TOH_Stop.Q;
246 IF StopReady THEN
247 ReadyToStandstill := TRUE;
248 StopReady := FALSE;
249 MachineState := 2;
250 END_IF
251 END_CASE

```

Appendice B

In questa appendice viene riportato parzialmente, per questioni di riservatezza, il codice ottimizzato per la gestione del moto del manipolatore PRRRP a due gradi di libertà. Inizialmente viene mostrato il blocco funzione *Kinematics*, il quale realizza la trasformazione di coordinate di posizione, con le relative implementazioni dei suoi metodi *Forward* ed *Inverse*.

```
1 FUNCTION_BLOCK Kinematics IMPLEMENTS MC_KIN_REF //FUNCTION BLOCK CHE IMPLEMENTA L'INTERFACCIA DI TRASFORMAZIONE DI COORDINATE (MC_KIN_REF)
2 VAR
3   fbDirKin : FB_x2DoF_RPPPR_DirectKinematics; //ISTANZA DEL FUNCTION BLOCK CHE CONTIENE LE FORMULE DI CINEMATICA DIRETTA DI POSIZIONE
4   fbInvKin : FB_x2DoF_RPPPR_InverseKinematics; //ISTANZA DEL FUNCTION BLOCK CHE CONTIENE LE FORMULE DI CINEMATICA INVERSA DI POSIZIONE
5 END_VAR
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

Successivamente vengono illustrati la sezione di dichiarazione delle variabili e il corpo del programma principale del software per la gestione del movimento del sistema robotico.

```
1 PROGRAM MotionProg
2 VAR
3   fbDefKin : MC_SetKinTransform; //ISTANZA DEL FB MC_SetKinTransform
4   fbKinematics : Kinematics; //ISTANZA DEL FB Kinematics
5   fbGroup : ML_GroupAllAxes; //ISTANZA DEL FB ML_GroupAllAxes
6   fbReadPosition_Axis : MC_GroupReadActualPosition; //ISTANZA DEL FB MC_GroupReadActualPosition
7   fbReadPosition_Joint : MC_GroupReadActualPosition; //ISTANZA DEL FB MC_GroupReadActualPosition
8   bDefineKin : BOOL;
9   bGroup : BOOL;
10  ActPosAxis : ARRAY[1..16] OF REAL := [16(0)]; //VETTORE PER LA POSIZIONE ATTUALE DELL'END-EFFECTOR
11  ActPosJoint : ARRAY[1..16] OF REAL := [16(0)]; //VETTORE PER LA POSIZIONE ATTUALE DEI GIUNTI
12 END_VAR
```



```

1  IF(bDefineKin) THEN
2    fbDefKin.KinTransform := fbKinematics; //ISTANZA DEL BLOCCO FUNZIONE CHE IMPLEMENTA L'INTERFACCIA DI TRASFORMAZIONE MC_KIN_REF
3    fbDefKin.Execute := TRUE;
4    fbDefKin.AxisGroup := Kinematics1;
5    IF(fbDefKin.Done) THEN
6      bDefineKin := FALSE;
7      fbDefKin.Execute := FALSE;
8      fbDefKin.AxisGroup := Kinematics1;
9    END IF
10   IF(fbDefKin.Error) THEN
11     bDefineKin := FALSE;
12   END IF
13 END_IF
14
15 IF fbDefKin.Done THEN
16   fbGroup( //CHIAMATA ALL'ISTANZA DEL BLOCCO FUNZIONE MC_GroupAllAxes
17     Execute:= bGroup,
18     AxisGroup:= Kinematics1,
19     Done=> ,
20     Active=> ,
21     Error=> ,
22     ErrorID=> ,
23     ErrorIdent=> );
24
25 fbReadPosition_Axis( //CHIAMATA ALL'ISTANZA DEL BLOCCO FUNZIONE MC_GroupReadActualPosition PER LA POSIZIONE ATTUALE DELL'END-EFFECTOR
26   Enable:= TRUE,
27   InOperation=> ,
28   Error=> ,
29   ErrorID=> ,
30   ErrorIdent=> ,
31   AxisGroup:= Kinematics1,
32   CoordSystem:= CS_BASE_COORDINATES,
33   Position=> ActPosAxis );
34
35 fbReadPosition_Joint( //CHIAMATA ALL'ISTANZA DEL BLOCCO FUNZIONE MC_GroupReadActualPosition PER LA POSIZIONE ATTUALE DEI GIUNTI
36   Enable:= TRUE,
37   InOperation=> ,
38   Error=> ,
39   ErrorID=> ,
40   ErrorIdent=> ,
41   AxisGroup:= Kinematics1,
42   CoordSystem:= CS_AXES_COORDINATES,
43   Position=> ActPosJoint );
44 END_IF

```

Bibliografia

- [1] Francesco La Trofa, “Cos’è la robotica, come funziona e quali sono gli esempi di applicazione”, 2021, Blog Tech4Future, <https://tech4future.info/robotica-cose-come-funziona-applicazioni/>
- [2] Edoardo Datteri, Guglielmo Tamburrini, “Robotica medica e società”, 2010, Treccani, https://www.treccani.it/enciclopedia/robotica-medica-e-societa_%28XXI-Secolo%29/#:~:text=La%20robotica%20medica%20%C3%A8%20un,sviluppo%20e%20valutazione%20dei%20sistemi
- [3] Alessandro Rubino, “Robot militari, quali sono, cosa sono in grado di fare”, 2022, Blog AI4Business, <https://www.ai4business.it/robotica/robot-militari-quali-sono-cosa-sono-in-grado-di-fare/>
- [4] Paolo Lino, “Introduzione”. Corso di Controllo dei Robot, 2019, Politecnico di Bari
- [5] “La robotica industriale”, 2018, Blog Automazione News, <https://www.automazione.news/la-robotica-industriale/>
- [6] Arianna Colurcio, “Androidi e robot nella letteratura e mitologia greca”, 2019, Blog laCOOLTura, <https://www.lacooltura.com/2019/03/androidi-mitologia-greca-talos-efesto/>
- [7] Matteo Costacurta, “Storia della Robotica: origini e sviluppi”, 2020, Blog Macchina Sociale, https://www.macchinasociale.com/storia-della-robotica/#Storia_della_Robotica
- [8] Jorge Elices Ocón, “Al-Jazari, il genio dell’islam medievale”, 2020, Blog Storica National Geographic, https://www.storicang.it/a/al-jazari-il-genio-dellislam-medievale_14615
- [9] Nicola Nosengo, “I robot ci guardano. Aerei senza pilota, chirurghi a distanza e automi solidali”, 2013, Zanichelli, <https://staticmy.zanichelli.it/catalogo/assets/a04.9788808175489.pdf>

- [10] Alessandro Freddi, “Il problema della localizzazione nei robot mobili”. Corso di Misure e strumentazione per l’automazione, 2020, Università Politecnica delle Marche
- [11] Aquila Mccarthy, Corso di Robotica, 2014, Sito SlideServe, <https://www.slideserve.com/aquila-mccarthy/corso-di-robotica>
- [12] Nicoletta Boldrini, “Robot: cosa sono, come funzionano e modelli disponibili”, 2022, Blog AI4Business, <https://www.ai4business.it/robotica/robot-cosa-sono-come-funzionano/>
- [13] Marco Gabiccini, “Robotica”. Corso di Robotica I, 2009, Università di Pisa
- [14] Domenico Prattichizzo, Gian Luca Mariottini, “Lezioni di ROBOTICA, & VISIONE ARTIFICIALE”, SIRSLab Università di Siena, <http://www.energiazero.org/robotica/Lezioni%20Di%20Robotica.pdf>
- [15] Massimo Callegari, “Architetture robotiche innovative”. Corso introduttivo automazione industriale e robotica, 2003, Università degli studi di Brescia, <http://www.meccanicadellemacchine.it/old/uk/db/articoli/2003%20corso%20SIRI.pdf>
- [16] Bruno Siciliano, Oussama Kathib, “Handbook of Robotics”, Springer, 2008
- [17] Giacomo Palmieri, “Cinematica”. Corso di Meccanica delle Macchine Automatiche, 2020, Università Politecnica delle Marche
- [18] Basilio Bona, “Cinematica 1”. Corso di Meccatronica, 2007, Politecnico di Torino
- [19] Marco Gabiccini, “Cinematica diretta ed inversa di manipolatori seriali”. Corso di Robotica I, 2010, Università di Pisa
- [20] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, Giuseppe Oriolo, “Robotics: modelling, planning and control”, Springer, 2010
- [21] Alessandro De Luca, “Robot con ridondanza cinematica”. Corso di Robotica 2, 2009, Università degli Studi di Roma "La Sapienza"
- [22] Mhretie Molaligh, “Dynamic analysis of a planar 2-DoF manipulator driven by pneumatic actuators”, 2010, Master Universitario, Politecnico di Milano

- [23] Sergiu-Dan Stan, Vistrian Maties and Radu Balan, “Optimal Design of Parallel Kinematics Machines with 2 Degrees of Freedom”, 2008, Technical University of Cluj-Napoca
- [24] Andrea Bonci, “Rigid Body Kinematics”. Corso di Dynamics and Control of Intelligent Robots and Vehicles, 2020, Università Politecnica delle Marche
- [25] Maria Anna Di Natale, “Studio ed implementazione di algoritmi per la pianificazione della traiettoria di un manipolatore planare a 2 gradi di libertà”, 2021, Università Politecnica delle Marche
- [26] Nicoletta Buora, “Come pianificare la traiettoria di un Robot”, 2018, Blog Automazione News, <https://www.automazionenews.it/come-pianificare-la-traiettoria/>
- [27] Andrea Bonci, “Trajectory Planning”. Corso di Dynamics and Control of Intelligent Robots and Vehicles, 2020, Università Politecnica delle Marche
- [28] Luigi Biagiotti, “Generazione del riferimento”. Corso di Controlli Automatici, 2009, Università degli Studi di Modena e Reggio Emilia
- [29] “Design Trajectory with Velocity Limits Using Trapezoidal Velocity Profile”, 2020, Sito MathWorks, <https://it.mathworks.com/help/robotics/ug/design-a-trajectory-with-velocity-limits-using-a-trapezoidal-velocity-profile.html>
- [30] “Motori Lineari Industriali-Smart solution are driven by LinMot®”, Sito LinMot, https://shop.linmot.com/data/import/Dokumente/0185-1100IT_1V1_Product_Overview.pdf
- [31] “Motori lineari”, Sito LinMot, <https://linmot.com/it/prodotti/motori-lineari/>
- [32] “Azionamenti servoassistiti”, Sito LinMot, <https://linmot.com/it/prodotti/azionamenti-servoassistiti/>
- [33] “Azionamenti elettrici-Dispense”, Corso di Azionamenti elettrici, 2009, Dipartimento di Ingegneria Università di Ferrara

- [34] “Unità di controllo integrata XM42”, Sito della Bosch Rexroth, <https://www.boschrexroth.com/it/it/prodotti/prodotti/azionamenti-e-controlli-elettrici/unita-di-controllo-integrata/xm/xm42>
- [35] Sercos.de, “Il protocollo sercos III”, 2012, Sito Yumpu, <https://www.yumpu.com/it/document/view/6903705/il-protocollo-sercos-iii>
- [36] “La sicurezza elettrica per ogni applicazione”, Sito Pilz, <https://www.pilz.com/it-IT/products/relay-modules/monitoring-relays-monitoring-devices>
- [37] “Pilz PNOZ mm0p-T Safety Controller, 20 Safety Inputs, 4 Safety Outputs, 24 V dc”, Sito RS Components, <https://mt.rsdelivers.com/product/pilz/772010/pilz-pnoz-mm0p-t-safety-controller-20-safety-4-24/1919877>
- [38] Matteo Sartini, “Manuale di utilizzo del software Codesys”, Università di Bologna, <http://sapa.campusfc.unibo.it/ManualeCodesys.pdf>
- [39] “Strumento software IndraWorks Engineering”, Sito Bosch Rexroth, <https://www.boschrexroth.com/it/it/prodotti/prodotti/azionamenti-e-controlli-elettrici/engineering/tool-software/indraworks-engineering>
- [40] Francesca Fanfoni, “PLC CodeSys Esercitazione 5”, Università degli Studi di Modena e Reggio Emilia
- [41] “R_TRIG, raising edge trigger”, Sito Elsisit, <https://www.fernhillsoftware.com/help/iec-61131/common-elements/standard-function-blocks/edge.html>
- [42] “TON”, Sito CODESYS Online Help, <https://help.codesys.com/webapp/ton;product=codesys;version=3.5.11.0>
- [43] Manuale “IndraMotion MLC 14VRS Robot Control V2”, Sito Bosch Rexroth, https://www.boschrexroth.com/documents/12605/25199795/R911341588_07.pdf/de3e9db3-6276-f740-aaf6-da9efb74bd9f
- [44] Enrica Merlo, “LA MODERNITA' DI JACQUES DE VAUCANSON: L'ANATRA DIGERITRICE E LA MIGLIORE OFFERTA.”, 2018, Blog

mimancanoifondamentali, <http://www.mimancanoifondamentali.com/2018/03/la-modernita-di-jacques-de-vaucanson.html>

[45] Marco Pinetti, “Joseph Engelberger, un sogno diventato realtà”, 2016, Blog Robotica News, <https://robotica.news/joseph-engelberger-un-sogno-diventato-realta/>

[46] “Computer Integrated Manufacturing Systems(CMS)Unimate Pumo 500 & Pumo 560 robots 1986(1).jpg”, Sito Wikipedia, https://commons.wikimedia.org/wiki/File:Computer_Integrated_Manufacturing_Systems%28CMS%29Unimate_Pumo_500_%26_Pumo_560_robots_1986%281%29.jpg

[47] “Aibo (1999)”, Sito ROBOTS YOUR GUIDE TO THE WORLD OF ROBOTICS, <https://robots.ieee.org/robots/aibo/>