UNIVERSITÀ
DEGLI STUDI
DI PADOVA

UNIVERSITA' DEGLI STUDI DI PADOVA

Dipartimento di Ingegneria Industriale DII

Corso di Laurea Magistrale in Ingegneria Aerospaziale

Progettazione di un braccio robotico per applicazioni spaziali integrato sul rover di Ateneo Morpheus

Relatore: Debei Stefano

Correlatore: Chiodini Sebastiano

Studente: Giacomo Franchini

Mat. 1218959

Anno Accademico 2020/2021

Sommario

Prefazione

Il progetto studentesco Morpheus, acronimo di Mars Operative Rover of Padova Engineering Students, si pone come obiettivo la progettazione e la realizzazione di un rover per effettuare simulazioni di missioni in ambienti lunari e marziani. Il progetto è nato nel 2014 per volere del Professore Stefano Debei, che rimane tutt'ora il docente referente, ed è stato finanziato negli anni grazie alla partecipazione al bando Progetti Innovativi di Studentesse e Studenti Finalizzato al Miglioramento della Didattica, promosso dall'Università di Padova.
Lo scopo finale del progetto Morpheus è la partecipazione alla competizione studentesca European Rover Challenge, una delle più importanti competizioni universitarie europee per quanto riguarda la robotica. Le diverse Università progettano e presentano il proprio rover, che deve essere in grado di simulare il comportamento in ambiente spaziale, affrontando diversi compiti.

Nel presente elaborato viene illustrato il lavoro svolto per la progettazione e la realizzazione di un braccio robotico a quattro gradi di libertà da integrare sul rover di Ateneo. L'obiettivo principale di questo progetto di tesi consiste nello sviluppo di un manipolatore capace di posizionare l'end effector in un qualsiasi punto dello spazio di lavoro con una determinata posa, rispettando i requisiti funzionali scelti e i vincoli imposti. Questo sarà propedeutico alla fase di sviluppo successiva del sistema Morpheus, ossia la possibilità di raccolta di campioni dal terreno una volta individuati dal sistema di visione del rover.
La seguente trattazione rappresenta la conclusione di due anni di partecipazione al progetto Morpheus, durante i quali il manipolatore è stato concepito, costruito e implementato.

1 Introduzione

1.1 La robotica nell'esplorazione spaziale

Lo spazio è uno dei luoghi più difficili e sfidanti a cui l'uomo abbia mai cercato di accedere. La strumentazione esposta all'ambiente spaziale deve confrontarsi ed essere in grado di resistere alle condizioni più estreme, a partire dagli ampissimi salti di temperatura, a pressioni diverse da quella terrestre per esplorazioni planetarie o prossime al vuoto assoluto nello spazio profondo, fino all'influenza del vento solare e di particelle cosmiche ad alta energia.

Sin dagli albori dell'era dell'esplorazione spaziale con il primo satellite artificiale a compiere un'orbita terrestre, il sovietico Sputnik 1 nel 1957, è risultato chiaro che non fosse possibile procedere a missioni con equipaggio umano senza prima aver effettuato decine di voli di test e di prova. Anche la relativamente vicina bassa orbita terrestre non è facile da raggiungere e tutt'oggi molti lanci terminano con un fallimento. Si è dunque reso necessario sviluppare sistemi automatizzati che potessero essere in grado di effettuare test, misure e manovre critiche utili ad ampliare le conoscenze e spianare la strada alle missioni con equipaggio.

Le sonde automatizzate sono attualmente utilizzate nell'ambito dell'esplorazione dei corpi celesti, soprattutto a causa del periodo di tempo necessario per raggiungerli. Per le prime missioni interplanetarie erano previsti solamente dei fly-by dell'obiettivo, ma in seguito sono stati sviluppati orbiter, lander e rover. Nonostante la difficoltà e i costi di missione aumentino estremamente, tra quelli appena citati i rover sono i più interessanti nell'ambito esplorativo. Essi, infatti, sono veicoli su ruote e quindi permettono di aumentare notevolmente lo spazio di lavoro e di sfruttare maggiormente la strumentazione disponibile per la missione trasportandola in prossimità delle zone di possibile interesse scientifico.

La prima nazione al mondo a riuscire ad effettuare la discesa e l'utilizzo di un rover su un altro corpo celeste, la Luna, fu l'Unione Sovietica nel 1970 con il rover Lunochod 1, nell'ambito della missione Luna 17. Da allora in poco più di cinquant'anni di evoluzione tecnologica sono state lanciate decine di missioni per esplorazione tramite rover, sia verso

la Luna che verso Marte, di cui viene ora fatto un breve riepilogo. Per ognuna di esse vengono riportati gli obiettivi, la data di lancio e le caratteristiche principali.

1.1.1 Luna 17 – Luna 21

Agenzia: Programma Spaziale Sovietico
Obiettivi: Esplorazione robotica lunare e supporto all'esplorazione umana
Data di lancio: 10 novembre 1970 – 8 gennaio 1973
Stato: Terminate
Esito: Successo

Durante gli anni della guerra fredda e della corsa alla Luna, l'Agenzia Spaziale Sovietica, in parallelo a quella americana, portava avanti il proprio programma spaziale lunare con l'obiettivo di far atterrare il primo essere umano su un altro corpo celeste. Le missioni Luna 17 e Luna 21 furono inizialmente sviluppate proprio con lo scopo di fornire supporto ai cosmonauti sia durante l'allunaggio, sia una volta sulla superficie lunare. Esse prevedevano l'allunaggio di due rover, denominati rispettivamente Lunochod 1 e 2. I rover erano dotati di otto ruote indipendenti e la strumentazione a bordo consisteva di un set di telecamere per la navigazione, due antenne, uno spettrometro e un telescopio a raggi X ed un rilevatore di raggi cosmici. Su di essi era assemblato anche un braccio robotico che ha permesso di effettuare misurazioni delle proprietà del suolo lunare.

1.1.2 Mars 2 – Mars 3

Agenzia: Programma spaziale Sovietico
Obiettivi: Esplorazione robotica di Marte
Data di lancio: 19 maggio 1971 – 21 maggio 1971
Stato: Terminate
Esito: Fallimento

Il programma Mars consistette in una serie di sette missioni realizzata all'interno del programma spaziale sovietico negli anni '70. Contemporaneamente al lancio della sonda Mars 1, la quale non riuscì ad effettuare l'inserzione in orbita di Marte, vennero lanciate le missioni identiche Mars 2 e 3. Esse consistevano in un orbiter ed un lander per la

discesa sulla superficie, con i principali obiettivi scientifici di produrre immagini della superficie marziana ed effettuare misurazioni meteorologiche e delle caratteristiche del suolo.
Tramite l'utilizzo di un braccio robotico, il lander doveva essere in grado di posizionare sulla superficie Prop-M, un piccolo rover di 4.5 kg collegato al modulo di discesa tramite un cavo ed in grado di muoversi sulla superficie grazie a due pattini fino a una distanza di circa 15 m dal lander.
Entrambi i lander fallirono durante la fase di discesa sulla superficie marziana, e di conseguenza nemmeno i due rover poterono mai funzionare in superficie.

1.1.3 Mars Pathfinder

Agenzia: NASA
Obiettivi: Esplorazione robotica di Marte
Data di lancio: 4 dicembre 1996
Stato: Terminata
Esito: Successo

Sojourner fa parte della missione Mars Pathfinder[1], una delle prime del NASA Discovery Program ed è stato il primo rover ad atterrare e navigare su Marte. Gli scopi principali della missione erano quelli di testare i sistemi e le tecnologie necessarie per effettuare la cosiddetta EDL (Entry, Descent and Landing procedure), atterrando su Marte in modo autonomo, e di dimostrare che fosse effettivamente possibile farvi operare un rover.
Il rover Sojourner aveva una massa complessiva di appena 11 kg. La configurazione scelta prevedeva uno chassis a sei ruote sostenute tramite l'innovativo sistema rocker-bogie, poi divenuta la configurazione di default di ogni rover marziano sviluppato dalla NASA.
Data l'impossibilità di comandarlo in tempo reale a causa del ritardo nelle comunicazioni Terra – Marte di circa 15 min, Sojourner è stato il primo tentativo di navigazione automatica di un rover su un altro pianeta. Nello specifico, per trasmettere i comandi e ricevere la telemetria si doveva aspettare l'apertura della finestra di comunicazione con il rover una volta per ogni Sol, ed esso doveva poi essere in grado di portare a termine la missione in modo autonomo.

Durante il suo funzionamento, Sojourner ha permesso di ottenere dati importanti allo scopo di validare tecnologie per i successivi rover completamente autonomi e ha studiato ed analizzato campioni di rocce e terreno tramite lo spettrometro a raggi X e le telecamere a bordo.

1.1.4 Mars Exploration Rover

Agenzia: NASA
Obiettivi: Esplorazione robotica di Marte
Data di lancio: 10 giugno 2003 – 8 luglio 2003
Stato: Terminata
Esito: Successo

La Mars Exploration Rover[2] ha previsto il lancio di due rover identici, Spirit ed Opportunity, per l'esplorazione dei crateri Gusev ed Eagle, siti equatoriali diametralmente opposti di Marte. I rover, alimentati da un sistema di pannelli solari e batterie, avevano sei ruote motrici di cui due coppie sterzanti di tipo Ackermann, che permettevano rotazioni attorno all'asse di yaw di 360°.
Spirit ed Opportunity erano dotati anche dell'Instrument Deployment Device (IDD), un braccio robotico a cinque gradi di libertà utilizzato per il posizionamento accurato degli strumenti su campioni di rocce e terreno. L'end effector dell'IDD conteneva una camera microscopica, uno spettrometro Mössbauer, uno spettrometro a raggi X e un Rock Abrasion Tool, che permetteva l'esposizione di nuovo materiale al di sotto dello strato superficiale che ricopre le rocce.
Per la missione era stata pianificata una durata di 90 Sol (poco più di tre mesi terrestri), ma dall'atterraggio nel gennaio 2004 Spirit ed Opportunity hanno operato per più di sei e quattordici anni terrestri rispettivamente.

1.1.5 Mars Science Laboratory

Agenzia: NASA

Obiettivi: Esplorazione robotica di Marte

Data di lancio: 26 novembre 2011

Stato: In corso

Esito: Successo

Sulla base della grande esperienza accumulata in seguito alle precedenti missioni marziane, la NASA ideò il rover Curiosity, componente principale della Mars Science Laboratory Mission. Durante l'ultimo decennio, Curiosity è stato il più grande e performante rover ad esplorare Marte, ed attualmente è secondo solamente a Perseverance che è una sua diretta evoluzione. L'obiettivo primario della missione è l'esplorazione del cratere Gale e l'analisi del terreno e dell'atmosfera, con lo scopo di determinare se su Marte sia mai esistito un ambiente favorevole al supporto di forme di vita microscopiche. Cinque dei dodici strumenti totali disponibili sono assemblati su una torretta che costituisce l'end effector del braccio robotico[3] a cinque gradi di libertà di Curiosity. Nello specifico sono presenti un trapano per la raccolta di campioni dalle rocce, la camera MAHLI, il Dust Removal Tool che permette l'eliminazione dei depositi di sabbia sulle superfici di analizzare, uno spettrometro a raggi X e il Collection and Handling for In-Situ Martian Rock Analysis device (CHIMRA), uno strumento che permette di processare e trasferire i campioni acquisiti verso gli altri strumenti presenti sul rover.
Per la missione era stata pianificata una durata di almeno un anno marziano (circa due anni terrestri), ma è tuttora in corso.

1.1.6 Chang'e 5

Agenzia: China National Space Administration

Obiettivi: Esplorazione robotica e ritorno campioni lunari

Data di lancio: 23 novembre 2020

Stato: Terminata

Esito: Successo

Chang'e 5 è stata la prima missione della China National Space Administration a prevedere il recupero di campioni del suolo lunare ed il loro ritorno a Terra. La missione è stata completata nel dicembre 2020 ed ha consentito il recupero di circa 2 kg di materiale. I componenti principali della missione sono quattro: un orbiter, un lander, un modulo di ascesa ed un modulo di rientro a Terra. Dopo l'inserzione in orbita lunare, l'orbiter ha rilasciato il modulo di discesa, che è poi atterrato in una zona inesplorata a nord dell'Oceanus Procellarum[4]. Il lander era dotato di un braccio robotico a quattro gradi di libertà, il cui end effector comprendeva una paletta per la raccolta di campioni di suolo ed un sistema di separazione, attraverso il quale diversi campioni potevano essere acquisiti e stivati senza mai entrare a contatto tra di loro. Terminata la raccolta, l'intero end effector contenente i campioni stivati è stato separato dal braccio robotico e trasferito al modulo di ascesa. In aggiunta, sul lander era presente anche un sistema di trivellazione che ha permesso la raccolta di altro materiale da profondità fino a 2 m.
Terminata la campagna di acquisizione dei campioni, il modulo di salita ha lasciato la superficie lunare ed effettuato una manovra di docking con il modulo orbitante, che ha poi permesso il rientro ha Terra dei campioni ottenuti.

1.1.7 Mars 2020

Agenzia: NASA
Obiettivi: Esplorazione robotica di Marte
Data di lancio: 30 luglio 2020
Stato: In corso

L'ultima finestra di lancio disponibile, tra luglio e settembre 2020, è stata sfruttata dalla NASA per il lancio della missione Mars 2020. Il modulo di navigazione è arrivato a destinazione nel febbraio 2021 ed il modulo di discesa ha effettuato la manovra EDL per permettere l'ammartaggio del rover Perseverance il 18 febbraio. Gli obiettivi della missione sono contigui a quelli della Mars Science Laboratory: la determinazione dell'esistenza di un ambiente che permettesse la vita microbiotica nel passato del pianeta, la raccolta di campioni di rocce e terreno ed il loro immagazzinamento per permettere a successive missioni di riportarli a Terra ed il test di nuove tecnologie, necessarie alla preparazione di missioni con equipaggio verso Marte.

Pur presentando una serie di notevoli evoluzioni, Perseverance è largamente basato sul design del precedente rover marziano Curiosity, fattore che ha permesso di minimizzarne i costi. Anche esso è dotato di un braccio robotico a cinque gradi di libertà sulla cui torretta sono presenti: un trapano che dispone di tre diversi utensili, i quali consentono di collezionare e stivare direttamente i campioni a bordo del rover, gli strumenti SHERLOC e WATSON, che consistono in un sistema di spettrometri, laser e telecamere che permettono l'accurata analisi dei campioni scelti e PIXL, uno spettrometro a raggi X per effettuare analisi della composizione chimica delle rocce.
Sulla struttura del rover è presente anche il Mars Helicopter Ingenuity, un dimostratore tecnologico che verrà utilizzato per testare per la prima volta il volo comandato su un altro pianeta. Una volta identificata una zona sicura per il test, Ingenuity verrà rilasciato dal rover e sarà in grado di alimentarsi grazie a pannelli solari.

1.1.8 Tianwen-1

Agenzia: China National Space Administration
Obiettivi: Esplorazione robotica di Marte
Data di lancio: 23 luglio 2020
Stato: In corso

L'ultima finestra di lancio verso Marte è stata sfruttata anche dalla Cina, per il lancio della sua prima missione interplanetaria Tianwen-1[5], che ha effettuato l'inserzione in orbita attorno al pianeta il 10 febbraio 2021. Tianwen-1 consiste in un orbiter, un lander ed un rover. Dopo aver stabilizzato e circolarizzato l'orbita, il 15 maggio 2021 il modulo di discesa è stato rilasciato ed è atterrato in sicurezza nell'Utopia Planitia. A bordo è presente anche il rover Zhurong: esso è dotato di sei ruote motrici, è alimentato a pannelli solari e dispone di un'antenna ad alto guadagno per le comunicazioni. A bordo è presente un set di sei strumenti scientifici, nello specifico: una telecamera di navigazione e topologica (NaTeCam), una telecamera multispettro (MSCam) un Ground Penetrating Radar (RoPeR) che permette di raggiungere profondità di circa 100 m, un magnetometro (RoMAG) ed una stazione meteorologica (MCS).

Gli obiettivi scientifici della missione riguardano lo studio delle caratteristiche geologiche e topologiche del pianeta, della composizione del terreno e distribuzione del ghiaccio d'acqua, dell'ambiente e del clima marziano, del campo magnetico e di quello gravitazionale.

2 Il progetto Morpheus

2.1 European Rover Challenge

Il progetto studentesco Morpheus[6] si pone come obiettivo la progettazione e la realizzazione di un rover simil-planetario per effettuare simulazioni di missioni in ambienti lunari e marziani. Dal 2014, anno della sua ideazione, fino ad oggi, lo sviluppo e l'evoluzione del progetto hanno permesso di creare un importante laboratorio didattico di robotica spaziale all'Università di Padova, per studenti di diversi corsi di ingegneria. La presenza di un laboratorio didattico permette di applicare attivamente le conoscenze e le nozioni teoriche apprese a lezione, guadagnando esperienza sul campo ed andandosi a scontrare con tutte le problematiche che si possono presentare nella pratica. Inoltre, vi è la possibilità per gli studenti di entrare a far parte di un team di giovani ingegneri, permettendo lo sviluppo di relazioni interpersonali, di problem solving e di capacità di gestione del progetto su diversi livelli.

L'obiettivo finale del progetto Morpheus è quello di testare le capacità del rover realizzato, tramite la partecipazione a competizioni studentesche internazionali come la European Rover Challenge (ERC). Durante la ERC, i rover, costruiti da gruppi di studenti provenienti da diverse università europee, si confrontano in una serie di task simili a quelli che dovrebbero essere in grado di eseguire sulla superficie di Marte o della Luna. Per questo motivo, oltre che ai requisiti che il rover deve soddisfare, durante il design preliminare del sistema sono stati presi in considerazione i compiti che sarà necessario portare a termine durante la competizione, e che verranno ora elencati.

- *Scientific task*: l'obiettivo del task scientifico è quello di elaborare, pianificare ed eseguire un'esplorazione scientifica dell'area di lavoro disponibile. Nello specifico, prima dello svolgimento della competizione occorre analizzare il "landing site" sulla base della documentazione fornita, e quindi generare una serie di obiettivi che si vuole raggiungere, per poi pianificare la missione, che sarà eseguita durante la competizione. In questa fase vengono raccolti i dati necessari alla missione, e al contempo vengono eseguiti i task di navigazione e di raccolta campioni.

- *Navigation task*: per poter raggiungere gli obiettivi di missione, i rover devono essere in grado di navigare in sicurezza all'interno dell'ambiente marziano simulato, in modalità semi-autonoma e completamente autonoma. Si vanno quindi a testare l'efficacia e le performance del sistema di navigazione dei rover, fornite dall'elaborazione dei dati ottenuti dai sensori presenti a bordo.
- *Probing task*: durante la navigazione nell'ambiente di missione, alcuni campioni devono essere raccolti e stivati a bordo del rover, per poi riposizionarli in una seconda posizione target.
- *Maintenance task*: in questa fase viene testata l'abilità e la precisione del sistema durante le operazioni eseguite dall'end effector presente sul braccio robotico. Esso deve interfacciarsi con un pannello elettrico, sul quale sono assemblati degli interruttori che devono essere settati sulle corrette posizioni. Inoltre, occorre effettuare una serie di misure elettriche e verificare gli output del quadro elettrico.
- *Presentation task*: questa fase consente alle varie squadre di presentare il proprio progetto alla commissione di valutazione, andando ad esplicitare le modalità di realizzazione del progetto, le soluzioni tecniche implementate a livello di hardware e software, e l'approccio utilizzato per permettere ai rover di risolvere i diversi task durante la competizione.

Il rover, inoltre, deve essere in grado di funzionare autonomamente e non dipendere da alcuna connessione via cavo, sia per quanto riguarda il sistema di potenza sia per le comunicazioni. La massa del sistema, complessiva di payload, non deve superare i 50 kg, anche se è da notare che questo limite viene considerato per ogni task separatamente, e dunque ad esso non partecipano gli elementi che non sono assemblati durante una particolare fase della competizione. La massima velocità non deve superare 1 m/s, ed il rover deve poter essere controllato tramite un radio link in tempo reale, per un raggio d'azione di circa 50 m.

2.2 Panoramica del rover Morpheus

Morpheus è un rover semi-autonomo, può essere telecomandato da remoto da un utente o navigare in maniera autonoma grazie al set di sensori disponibili a bordo. Il rover è stato progettato per raggiungere una velocità massima pari a 1 m/s, in un tempo massimo di 5 m/s su un terreno che presenta un'inclinazione di 15°. Le dimensioni di ingombro del sistema sono di 820x130x1000 mm, e la massa complessiva è di circa 50 kg.

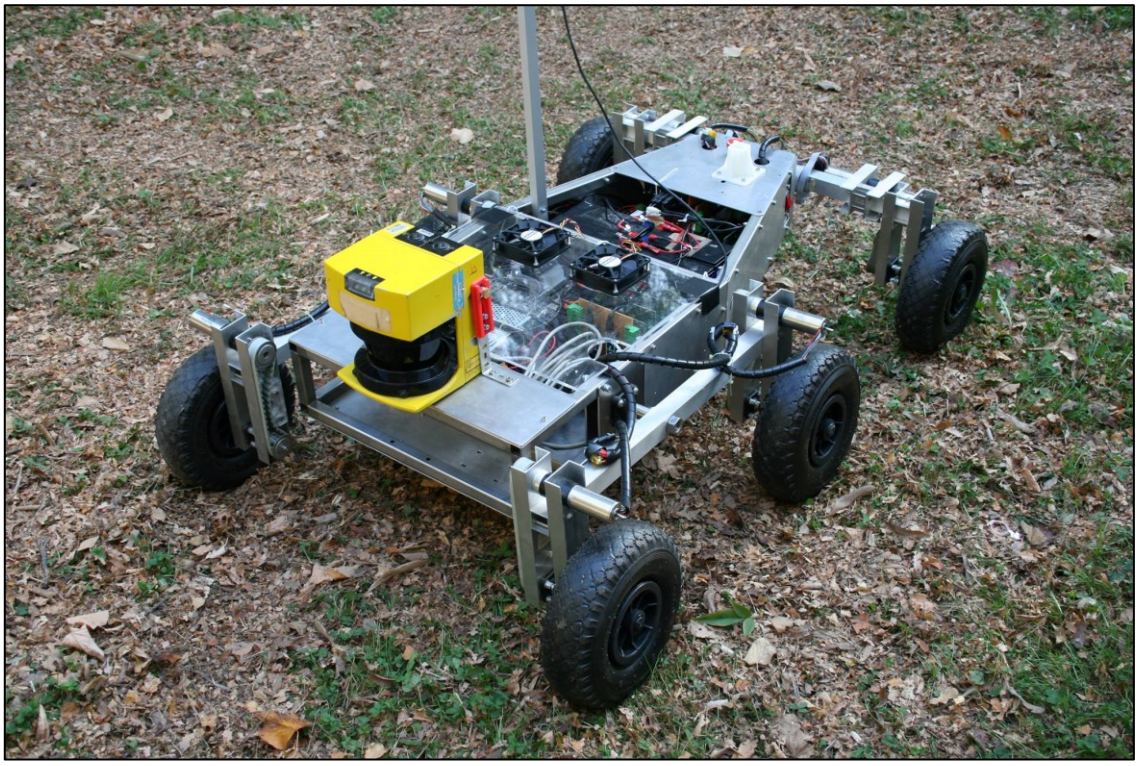

Figura 2.1 –Vista generale del Rover Morpheus.

2.2.1 Meccanica: chassis e sistema di locomozione

La struttura principale del rover Morpheus consiste in uno chassis formato da tubolari di acciaio saldati a sezione quadrata, di dimensioni 20x20x1 mm. Nella zona centro-frontale del rover i profilati sono uniti a formare una sezione rettangolare, mentre nella zona di coda il telaio è rastremato. Il rover è dotato di un set di sei ruote motrici indipendenti, ognuna delle quali presenta un sistema di trasmissione a cinghia ed è messa in rotazione da un motoriduttore brushless. L'elasticità della cinghia di trasmissione permette di assorbire eventuali urti durante la guida e quindi di proteggere l'albero del motoriduttore da carichi radiali anomali. Ogni coppia di ruote viene collegata al telaio tramite un

bilanciere: ne sono presenti due laterali ed uno posteriore. Questa configurazione permette al sistema di adattarsi alle diverse morfologie e inclinazioni del terreno, e di tollerare fino a 50° di angolo di rollio senza ribaltarsi. Il rover non è dotato di un sistema sterzante ma lavora in modalità skid-steering. Nello specifico, si utilizzano diverse velocità angolari delle coppie di ruote per permettere al rover di ruotare attorno al suo asse di yaw.

Attualmente, sia la struttura principale del rover che le ruote sono in fase di revisione, con l'obiettivo di ridurre la massa complessiva del sistema e di avere maggiore capacità di trazione su diverse tipologie di terreno. Il nuovo telaio presenterà una configurazione simile a quella attuale, permettendo però di aumentare il volume disponibile per la componentistica a bordo. I profilati di acciaio verranno sostituiti con profili Bosch in fibra di carbonio, collegati tra loro tramite appositi segmenti angolari nell'ottica di un design modulare semplice da modificare e smontare. Per quanto riguarda il nuovo set di ruote, diverse configurazioni sono in fase di studio. Il team è impegnato nella preparazione di un ambiente di test e raccolta dati con prototipi realizzati con stampa 3D, per scegliere il design definitivo e passare alla fase di realizzazione.

2.2.2 Elettronica: sistema di potenza, di controllo e telecomunicazioni

Il sistema di potenza di Morpheus consiste in due coppie di batterie al piombo ricaricabili. All'interno di una coppia, le singole batterie da 12 V sono collegate in parallelo in modo tale da aumentare la corrente disponibile ai carichi. Le due coppie sono quindi poste in serie, permettendo così di ottenere una tensione in uscita di 24 V. Le diverse tensioni richieste dai sottosistemi del rover (24, 12 e 5 V) sono rese disponibili tramite convertitori DC/DC.

Il componente centrale del rover Morpheus è il computer di bordo NVIDIA JETSON TX2, sulla quale è installato il sistema operativo Ubuntu ed il meta-sistema operativo ROS (Robotic Operating System). Tramite essi, la scheda permette di controllare tutti i sottosistemi e le funzioni del rover, dal sistema di locomozione e braccio robotico, comandando i motori, effettuando la lettura degli encoder e controllandone la temperatura, al sistema di navigazione, ricevendo ed elaborando i segnali dei diversi sensori, fino alle telecomunicazioni. La scheda permette di generare un ponte Wifi tra il rover e la base station, dalla quale l'operatore può controllare il rover da remoto.

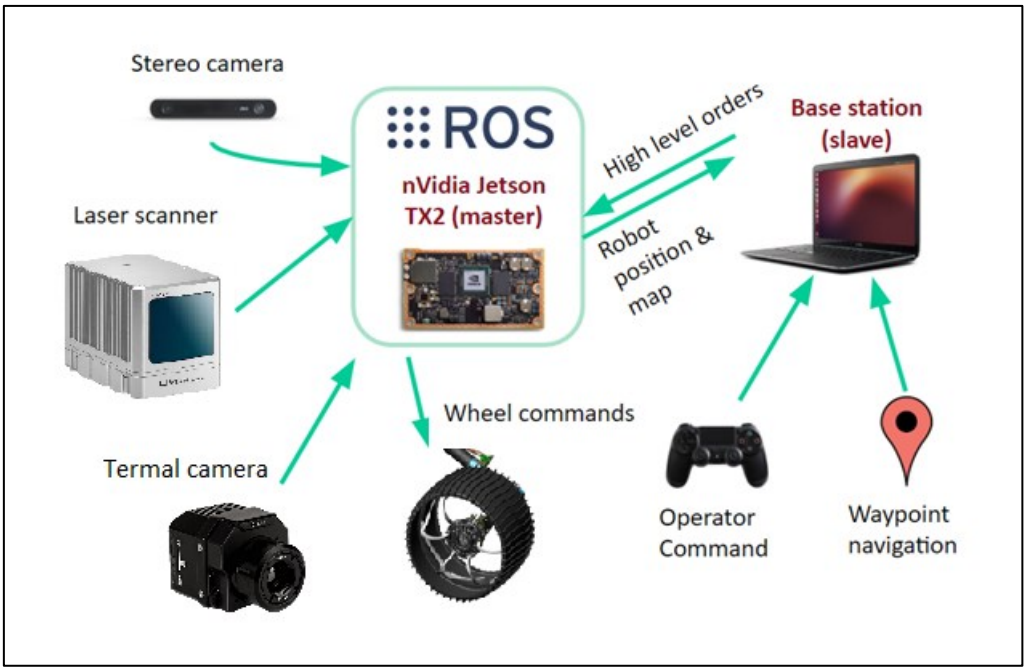

Figura 2.2 – Sistema di controllo del Rover Morpheus.

2.2.3 Sensoristica: sistema di visione

Il rover Morpheus è dotato di un set di sensori che ne definiscono il sistema di visione. Esso permette al rover di navigare in maniera autonoma in un ambiente non strutturato, generandone e memorizzandone una mappa, per poi elaborare il percorso da seguire a seconda dei vincoli identificati. Inoltre, il sistema di visione permette al rover di identificare dei campioni significativi presenti sul terreno esplorato e di individuarne la posa relativa, per poi effettuarne la raccolta tramite il braccio robotico.

I sensori che compongono il sistema di visione sono tre e lavorano in simbiosi tra loro:

- una stereocamera StereoLabs ZED, con risoluzione 1920x1080 pixel @30 fps e campo di vista massimo 90°x60°x100°. La stereocamera è assemblata su un'appendice in posizione elevata rispetto al corpo del rover, per permettere un'ampia visione dell'ambiente circostante;
- un Lidar 3D Livox Horizon, con range di detezione di 260 m in direzione orizzontale e verticale, precisione sul range di 2 cm e di 0.05° angolare, e campo di vista 81.7°x25.1°;

- una telecamera termica FLIR Vue Pro R che lavora nella banda spettrale 7.5 – 13.5 μm LWIR, con risoluzione 336x256 pixel e campo di vista 24°x18° con ottica da 13 mm oppure 34°x26° con ottica da 9 mm.

3 Il braccio robotico di Morpheus

3.1 Obiettivi di missione e requisiti funzionali

Uno dei compiti fondamentali che un rover deve essere in grado di svolgere durante una missione in ambiente lunare o marziano è la raccolta e la stiva di campioni di suolo, tipicamente rocce o altre componenti del terreno, per permetterne l'analisi. Non meno importante, nello scenario futuro di missioni con equipaggio, è la capacità del sistema robotico di interfacciarsi e di fornire supporto agli esseri umani, fattore che si sta rivelando sempre più centrale.

Il rover deve quindi essere dotato di un manipolatore che permetta l'esecuzione dei task sopra citati, nel rispetto dei requisiti funzionali desiderati e sul quale possono essere assemblati differenti tipologie di end effector, come gripper o palette, a seconda del compito da assolvere.

Ponendosi come simulatore di ambiente di missione, sin dalla prima iterazione era previsto che il rover Morpheus fosse dotato di un manipolatore. Dal momento dell'ideazione del progetto Morpheus, sono state concepite diverse configurazioni e design per la realizzazione di un braccio robotico da integrare sul rover, nessuna delle quali è stata però successivamente realizzata.

Il presente elaborato illustra il lavoro svolto durante la concezione, la progettazione e la realizzazione di una nuova configurazione per il braccio robotico, propedeutico alla fase successiva di sviluppo del sistema Morpheus, ossia la possibilità di raccolta di campioni dal terreno una volta individuati dal sistema di visione del rover.

Verranno ora elencati e discussi gli obiettivi da raggiungere, i requisiti funzionali che il sistema deve soddisfare ed i vincoli imposti dal problema.

3.1.1 Obiettivi di missione

L'obiettivo che, tramite il presente progetto di tesi, si desidera raggiungere è la capacità del manipolatore di posizionare il suo end effector in un qualsiasi punto all'interno dello spazio di lavoro scelto, con una determinata posa e rispettando tutti i requisiti funzionali e i vincoli imposti al sistema. Ciò permetterà al rover di risolvere i task di manipolazione per cui è progettato, come la raccolta e la stiva a bordo di campioni di suolo identificati tramite il sistema di visione, ed il loro trasporto alla base station.

3.1.2 Requisiti funzionali e vincoli imposti

Al fine di perseguire gli obiettivi di missione in maniera consona, il sistema robotico che si andrà a sviluppare dovrà soddisfare una serie di requisiti funzionali. La loro definizione è di fondamentale importanza in quanto andrà ad influenzare tutte le scelte progettuali da effettuare durante il processo di design del manipolatore: la scelta della configurazione cinematica, dei motoriduttori ai giunti, della tipologia di end effector, del sistema di potenza e di controllo. I requisiti funzionali che il braccio robotico deve soddisfare sono elencati di seguito:

- *Tipologie di task da eseguire:* il braccio robotico deve poter svolgere il compito per cui è stato concepito, ossia il raggiungimento, la raccolta, il trasporto e la stiva a bordo del rover dei campioni identificati. Il robot dovrà quindi essere dotato di un numero di gradi di libertà tale da permettere l'esecuzione dei diversi task, tenendo però in considerazione che un maggiore numero di gradi di libertà comporta maggiori costi, sia in termini economici, che in termini di difficoltà tecnologiche e di gestione del sistema;
- *Spazio di lavoro raggiungibile:* una volta assemblato sul rover, il braccio robotico deve essere in grado di posizionare l'end effector, con la posa desiderata, all'interno di uno spazio di lavoro di dimensioni 800 mm in lunghezza e 500 mm in larghezza e altezza;
- *Dimensioni e massa del payload:* operando all'interno dello spazio di lavoro appena descritto, il braccio robotico deve essere in grado di afferrare e movimentare un payload di massa fino a 1 kg.
- *Stiva e trasporto del payload:* il payload raccolto deve essere stivato a bordo del rover. Inoltre, durante le manovre di trasporto, si deve avere la possibilità di mantenere fissa l'orientazione del payload rispetto agli assi di roll e pitch del sistema, in modo da evitare perdite di materiale.
- *Assemblaggio sul rover:* le dimensioni di ingombro del braccio robotico devono essere compatibili con quelle del rover. In posizione di riposo, il braccio robotico deve essere contenuto all'interno dell'impronta del telaio del rover e deve esserne impedito qualsiasi movimento;

- *Precisione di posizionamento:* rispetto alla posa target fornita in ingresso, l'end effector deve assumere una posizione ed un assetto che non differiscano per più di 25 mm e 5 gradi su ogni asse. Inoltre, all'interno del sistema deve essere integrato un set di sensori che sia in grado di effettuare e fornire in uscita le misure necessarie per la ricostruzione della posa, in un tempo non superiore a 0.1 s;
- *Rigidezza del sistema:* la rigidezza dell'insieme dei link e dei giunti del braccio robotico deve essere tale da garantire l'esecuzione dei task programmati senza che il sistema esibisca deformazioni elastiche di entità non trascurabile.

In parallelo alla definizione dei requisiti funzionali, occorre andare a considerare i vincoli presenti sul sistema. L'intero processo di progettazione del manipolatore dovrà permettere in ogni istante di soddisfare i requisiti funzionali, garantendo allo stesso tempo il rispetto dei vincoli. Essi sono imposti dalla natura stessa del sistema robotico considerato.

Il primo vincolo deriva dal fatto che un rover sviluppato per effettuare simulazioni di missioni in ambiente spaziale deve poter operare in modo autonomo. Dunque, il sistema di alimentazione, costituito dal set di batterie presenti a bordo, deve essere in grado di soddisfare le richieste di potenza di tutti i sottosistemi, in modo continuo e per l'intera durata della missione. La potenza elettrica disponibile al sottosistema del manipolatore sarà quindi limitata.

Un ulteriore vincolo è dato dal fatto che il braccio robotico dovrà essere integrato con il rover. Questo vincolo si tramuta in un limite circa il range di traiettorie percorribili dall'end effector durante l'esecuzione dei task, in modo da evitare ogni tipo di collisione tra il manipolatore e il rover o la sua strumentazione.

L'ultimo vincolo deriva dalla presenza dei cablaggi necessari al funzionamento di tutti i componenti che andranno a formare il sottosistema del braccio. La configurazione scelta per quest'ultimo dovrà garantire la possibilità di esecuzione dei movimenti desiderati, senza andare a danneggiare o esercitare tensioni eccessive sui cablaggi.

3.2 Cinematica e dinamica dei manipolatori

Un robot manipolatore è un sistema meccanico integrato costituito da elementi strutturali, sensori, attuatori, controllori e comunicazioni, pensato e sviluppato al fine di svolgere compiti e attività al posto dell'uomo. Un manipolatore viene classificato come meccanismo in catena aperta, in quanto gli elementi che lo costituiscono sono collegati fra loro in modo da formare una catena con una ben definita direzionalità: l'end-effector viene collegato alla base del cinematismo, il telaio, tramite dei link rigidi, che costituiscono gli elementi strutturali del sistema. Il moto relativo fra due link consecutivi del robot è consentito dai giunti, le cui tipologie principali sono essenzialmente due: i giunti prismatici, che consentono un moto relativo di tipo traslazionale e i giunti rotoidali, che consentono un moto relativo di tipo rotatorio. La presenza di uno di essi aggiunge un singolo grado di libertà al meccanismo in catena aperta. Se attivi, i giunti sono costituiti da attuatori, sensori, organi meccanici ed elementi di controllo, integrati fra loro.

3.2.1 Modello cinematico di un manipolatore

Ad ogni link di un manipolatore viene associato un sistema di riferimento proprio e solidale. In generale, la scelta di questo sistema di riferimento può essere fatta in modo arbitrario, ma spesso risulta conveniente basarsi sulle regole definite dalla convenzione di Denavit-Hartenberg (D-H). Per la terna di riferimento del generico link *i*-esimo:

- L'asse Z_i giace lungo l'asse del giunto $i+1$;
- L'origine O_i coincide con l'intersezione fra l'asse Z_i e la normale comune agli assi Z_i e Z_{i-1}. Si indica con $O_{i'}$ il punto di intersezione della normale comune con l'asse Z_{i-1};
- L'asse X_i è diretto lungo la normale comune agli assi Z_i e Z_{i-1}, con verso positivo dal giunto i al giunto $i+1$;
- L'asse Y_i è scelto per completare la terna destra.

Scegliendo i sistemi di riferimento dei link sulla base della convenzione di D-H, si ottiene che per la determinazione di una terna rispetto alla precedente sono sufficienti quattro parametri:

- La distanza fra i due punti O_i e $O_{i\prime}$, indicata con a_i;
- La coordinata lungo Z_{i-1} di $O_{i\prime}$, indicata con d_i;
- L'angolo tra gli assi Z_{i-1} e Z_i attorno ad X_i, indicato con α_i e positivo in senso orario;
- L'angolo tra gli assi X_{i-1} e X_i attorno a Z_{i-1}, indicato con θ_i e positivo in senso orario;

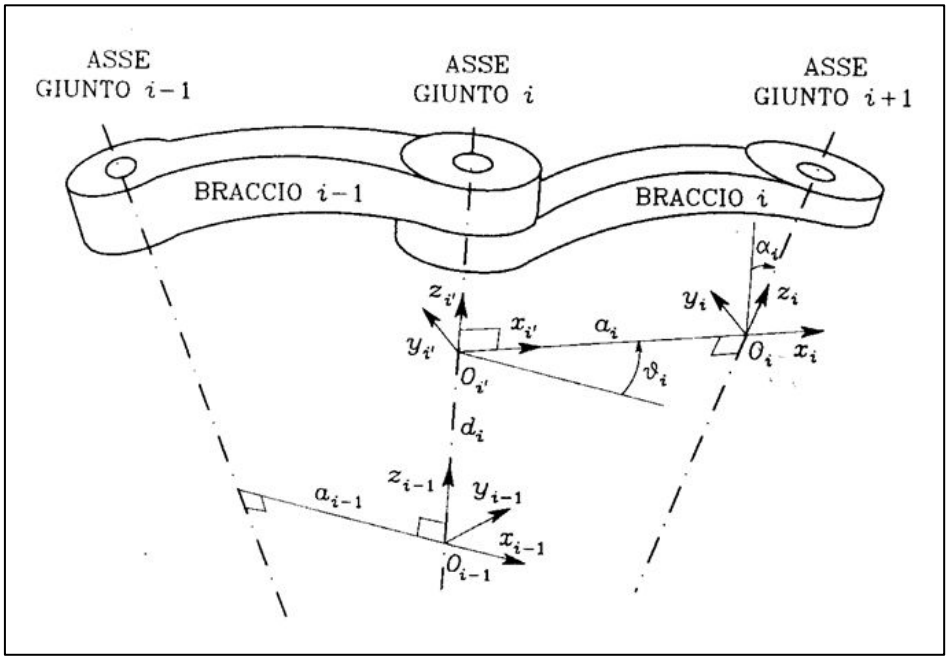

Figura 3.1 - Sistemi di riferimento dei link di un manipolatore secondo la convenzione di D-H.

La convenzione di D-H viene utilizzata al fine di rendere più semplice la costruzione del modello cinematico del manipolatore. Quest'ultimo consiste in una descrizione analitica delle relazioni presenti fra l'insieme delle variabili di giunto e la posizione e l'assetto dell'end effector nello spazio. Per riferire la posizione e l'orientamento del link *i*-esimo rispetto al link precedente (*parent link*) si utilizza la matrice di rototraslazione, o di trasformazione, dal sistema di riferimento i a quello $i-1$, scritta in coordinate omogenee e definita nel modo seguente:

$$A_i^{i-1} = \begin{bmatrix} R_i^{i-1} & o_i^{i-1} \\ 0^T & 1 \end{bmatrix} \quad (3.1)$$

dove R_i^{i-1} è la matrice di rotazione fra la terna solidale al link i e quella solidale al link $i-1$, o_i^{i-1} il vettore che esprime le coordinate dell'origine della terna i nel sistema di riferimento $i-1$ e 0^T il vettore nullo. La matrice di trasformazione totale del manipolatore, la quale definisce la posa dell'end effector rispetto al sistema di riferimento fisso, solidale alla base del robot, si determina una volta ottenute tutte le matrici di trasformazione di una terna rispetto alla precedente. Se nel sistema sono presenti n link, la matrice di trasformazione totale è la seguente:

$$T_n^0(q) = A_1^0(q_1)\, A_2^1(q_2) \cdots A_n^{n-1}(q_n) \quad (3.2)$$

Le variabili q_i sono dette variabili di giunto e possono essere angoli e/o distanze, a seconda dalla tipologia di giunto. L'insieme delle variabili di giunto di un manipolatore definisce lo spazio dei giunti, mentre lo spazio di lavoro (o operativo) è definito dall'insieme delle configurazioni che l'end-effector può assumere, in termini di posizione ed assetto. Si definisce spazio di lavoro raggiungibile la regione descritta dall'origine della terna solidale all'end effector quando i giunti eseguono tutti i moti possibili, mentre lo spazio di lavoro destro è un sottoinsieme di quest'ultimo costituito dai punti raggiungibili con più di un solo orientamento della terna utensile. Lo spazio di lavoro dipende dalla geometria del sistema robotico e dalla corsa disponibile ai giunti.
Per esprimere la velocità di un dato punto del link i rispetto al link $i-1$, si utilizza la matrice di trasformazione di velocità, scritta in coordinate omogenee, e definita come:

$$\dot{A}_i^{i-1} = V_i^{i-1} A_i^{i-1} \text{ dove } V_i^{i-1} = \begin{bmatrix} 0 & -\omega_{zi} & \omega_{yi} & V_{Gxi} \\ \omega_{zi} & 0 & -\omega_{xi} & V_{Gyi} \\ -\omega_{yi} & \omega_{xi} & 0 & V_{Gzi} \\ & 0^T & & 0 \end{bmatrix} \quad (3.3)$$

V_i^{i-1} è la matrice di velocità in riferimento baricentrico, ω_{ji} le V_{Gji} sono rispettivamente le tre componenti della velocità di rotazione e della velocità lineare del baricentro della terna i. Allo stesso modo, per esprimere l'accelerazione di un punto del link i rispetto al

link $i-1$, si utilizza la matrice di trasformazione di accelerazione, scritta in coordinate omogenee e definita come:

$$\ddot{A}_i^{i-1} = W_i^{i-1} A_i^{i-1} \text{ dove } W_i^{i-1} = \begin{bmatrix} \dot{\omega} + \omega^2 & \begin{matrix} a_{Gxi} \\ a_{Gyi} \\ a_{Gzi} \end{matrix} \\ 0^T & 0 \end{bmatrix} \quad (3.4)$$

W_i^{i-1} è la matrice di accelerazione in riferimento baricentrico, $\dot{\omega}$ e ω^2 sono rispettivamente i contributi dell'accelerazione angolare e centripeta, mentre a_{Gji} sono le tre componenti dell'accelerazione del baricentro della terna i. Una volta note tutte le matrici di trasformazione di velocità e di accelerazione, è possibile determinare le velocità e le accelerazioni assolute dell'end-effector, ossia rispetto al sistema di riferimento fisso solidale alla base del robot.

3.2.2 Analisi cinematica diretta e inversa

Definito il modello cinematico del manipolatore, vi sono due problemi da affrontare: la cinematica diretta e quella inversa. La risoluzione del problema cinematico diretto consiste nel determinare la posa dell'end-effector nello spazio di lavoro in funzione del valore delle singole variabili di giunto, contenute nel vettore $\bar{q}$. La posa dell'end effector viene descritta tramite il vettore $\bar{X}$, composto da un vettore $\bar{P}$ che ne descrive la posizione e un vettore $\bar{\phi}$ che ne descrive l'orientamento in termini di tre parametri indipendenti, quali ad esempio gli angoli di Eulero, gli angoli di Roll, Pitch e Yaw o i quaternioni. Le equazioni della cinematica diretta possono dunque essere scritte come:

$$\bar{X} = f(\bar{q}) \quad (3.5)$$

Al contrario, risolvere la cinematica inversa significa determinare il vettore delle variabili di giunto, ossia le configurazioni dei vari giunti del manipolatore, una volta nota la posizione e l'assetto dell'end-effector nello spazio. Il sistema di equazioni da risolvere è il seguente:

$$\bar{q} = g(\bar{X}) \quad (3.6)$$

In questo caso il problema cinematico si complica, poiché in generale le equazioni da risolvere sono non lineari, e dunque non è sempre possibile trovare una soluzione in forma chiusa, ma è necessario ricorrere a metodi numerici. Inoltre, è possibile che vi sia una molteplicità di soluzioni che soddisfano il sistema: per poter scegliere fra le varie possibilità occorre applicare alcuni criteri di scelta e considerare i vincoli del problema, come ad esempio la presenza di ostacoli che non permettono alcuni spostamenti del manipolatore. In generale, si prediligono le soluzioni che prevedono i minori spostamenti possibili nello spazio dei giunti e la minor spesa di potenza durante la loro esecuzione. Il numero delle soluzioni aumenta all'aumentare del numero dei giunti che costituiscono il robot, della corsa disponibile agli stessi, e del numero di parametri di Denavit-Hartenberg del manipolatore non nulli.

Per quanto riguarda il problema cinematico differenziale, ossia l'analisi delle relazioni che intercorrono fra le velocità e le accelerazioni dei giunti e quelle dell'end effector, si fa uso della matrice Jacobiana. Sia $\bar{y} = f(\bar{x})$ una funzione vettoriale di variabili vettoriali, con $f: R^n \rightarrow R^m$. La matrice i cui elementi sono le derivate parziali prime della funzione f è detta matrice Jacobiana:

$$\dot{\bar{y}}_i = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \dot{x}_j \quad \text{con } i = 1, \dots, m \text{ e } j = 1, \dots, n \qquad (3.7)$$

Se la funzione è differenziabile, allora la matrice Jacobiana ne fornisce la migliore approssimazione lineare in un dato punto. Se inoltre la matrice è quadrata, il suo determinante viene detto Jacobiano della funzione. Nel problema cinematico, la matrice Jacobiana viene utilizzata sia per la risoluzione della cinematica diretta di velocità e di accelerazione, sia per quella inversa. In particolare, nel primo caso si determinano i vettori velocità $\bar{v}$ ed accelerazione $\bar{a}$ dell'end effector rispetto al sistema di riferimento alla base del robot, quando sono note le derivate prime e seconde delle variabili di giunto, rispettivamente $\dot{\bar{q}}$ e $\ddot{\bar{q}}$. Il problema si presenta nella forma:

$$\bar{v} = J(\bar{q})\, \dot{\bar{q}} \qquad (3.8)$$

$$\bar{a} = J(\bar{q})\, \ddot{\bar{q}} + \dot{J}(\bar{q})\, \dot{\bar{q}} \qquad (3.9)$$

Nella cinematica inversa, invece, risulta necessario calcolare l'inversa della matrice Jacobiana. Come è noto dall'algebra lineare, condizione necessaria affinché una matrice sia invertibile è che essa sia quadrata. In un manipolatore ciò avviene quando il numero di equazioni è uguale al numero delle derivate delle variabili di giunto da determinare. Se tale condizione è verificata, allora si può scrivere:

$$\dot{\bar{q}} = J(\bar{q})^{-1}\bar{v} \qquad (3.10)$$

$$\ddot{\bar{q}} = J(\bar{q})^{-1}(\dot{\bar{v}} - \dot{J}(\bar{q})\,\dot{\bar{q}}) \qquad (3.11)$$

$$J^{-1} = \frac{J^T}{\det(J)} \qquad (3.12)$$

La relazione 3.12 mette in luce il fatto che, qualora il valore del Jacobiano tendesse a zero, la matrice Jacobiana non avrebbe rango massimo e la sua inversa J^{-1} divergerebbe all'infinito: il manipolatore si troverebbe in una configurazione di singolarità. Le singolarità rappresentano configurazioni di ridotta mobilità per il sistema robotico, non permettendo di imporre un moto arbitrario all'end effector. Nell'intorno delle configurazioni di singolarità, piccoli input da parte dell'end-effector portano ad elevati valori per le velocità e le accelerazioni ai giunti. Le configurazioni di singolarità possono essere suddivise in due gruppi: singolarità presenti ai confini dello spazio di lavoro, le quali possono essere evitate semplicemente impedendo al manipolatore di raggiungere le regioni limite del suo workspace e singolarità presenti all'interno dello spazio di lavoro, più problematiche da gestire rispetto alle precedenti in quanto possono presentarsi per una traiettoria pianificata nello spazio operativo.

3.2.3 Calibrazione cinematica

Si è visto come la risoluzione della cinematica diretta del manipolatore permetta di determinare la posa dell'end-effector nello spazio di lavoro in funzione del valore delle variabili di giunto. Dalla relazione 3.5 è possibile mettere in evidenza la dipendenza della posa dell'end effector, oltre che dai valori delle variabili di giunto, dai parametri di D-H del cinematismo, secondo la relazione 3.13:

$$\bar{X} = f(\bar{a}, \bar{d}, \bar{\alpha}, \bar{\vartheta}) \qquad (3.13)$$

Nella quale i vettori $\bar{a}$, $\bar{d}$, $\bar{\alpha}$ e $\bar{\vartheta}$ contengono i rispettivi parametri di D-H dell'intero sistema. L'accuratezza di posizionamento del manipolatore è quindi influenzata dalla precisione con cui questi ultimi sono noti. Lo scopo del processo di calibrazione cinematica è quello di determinare il valore dei parametri di D-H del cinematismo che rimangono costanti durante il moto. La calibrazione viene eseguita andando a misurare la posizione dell'end effector, ad esempio tramite l'utilizzo di vincoli meccanici che permettano di identificarla in modo univoco o tramite la misura diretta con strumenti esterni (telecamere, laser scanner, ecc.), e ripetendo la misura dopo aver movimentato i giunti del manipolatore di quantità note. Il numero minimo di misurazioni che è necessario effettuare è pari al numero di parametri di D-H che devono essere determinati.

3.2.4 Analisi dinamica diretta e inversa

A differenza della cinematica, la dinamica di un manipolatore si occupa di determinare le cause che determinano il moto dello stesso, ossia le forze e le coppie agenti sul sistema. La descrizione del problema viene fatta attraverso le due equazioni cardinali della dinamica, le quali esprimono la conservazione della quantità di moto e del momento angolare del sistema stesso. Per un robot manipolatore, l'equazione della dinamica assume la seguente forma:

$$\bar{\tau} = H(q)\,\ddot{\bar{q}} + \bar{h}(\bar{q},\dot{\bar{q}}) = H(q)\,\ddot{\bar{q}} + C(\bar{q},\dot{\bar{q}})\,\bar{q} + \bar{G}(\bar{q}) \qquad (3.14)$$

Dove $\bar{\tau}$ è il vettore delle forze generalizzate ai giunti e può contenere sia forze lineari che coppie a seconda della tipologia di giunto, $H(\bar{q})$ è la matrice delle masse generalizzate e contiene sia le masse che le inerzie del sistema mentre $\bar{h}(\bar{q},\dot{\bar{q}})$ è un vettore scomponibile in due componenti $C(\bar{q},\dot{\bar{q}})\,\bar{q}$ e $\bar{G}(\bar{q})$, che sono rispettivamente i contributi dell'accelerazione di Coriolis e dell'accelerazione di gravità.

Effettuare un'analisi dinamica diretta significa determinare la legge di moto dei vari elementi del manipolatore $\bar{q}(t)$, $\dot{\bar{q}}(t)$ e $\ddot{\bar{q}}(t)$ a partire dalle forze generalizzate $\bar{\tau}$ applicate dagli attuatori ai giunti. Al contrario, l'analisi dinamica inversa permette di determinare le forze o le coppie che agiscono ai giunti, quando sono noti gli spostamenti, le velocità e le accelerazioni di tutti gli elementi del manipolatore.

3.3 Progettazione meccanica

La progettazione meccanica di un manipolatore è un processo che si svolge in più fasi. In primo luogo, occorre definire il design cinematico preliminare, con lo scopo di ottenere come risultato il modello cinematico teorico del sistema robotico ed una stima iniziale delle masse e delle inerzie in gioco. Questi verranno utilizzati per effettuare la scelta di un appropriato sistema di attuazione per il manipolatore.

In questa fase iniziale si vanno ad effettuare delle opportune scelte di design sulla base dei requisiti funzionali e dei vincoli presenti sul sistema, discussi al paragrafo 3.1. Da esse si ricava la configurazione cinematica del manipolatore, che viene utilizzata come input di un processo iterativo, utilizzato per la definizione del sistema di attuazione. Durante ogni singola iterazione del processo, vengono simulate delle manovre del modello cinematico, permettendo di ottenere in output i profili di coppia richiesti ai giunti, necessari per effettuare la scelta dei motoriduttori da utilizzare. Una volta che questi ultimi sono stati individuati, occorre andare a verificare che siano rispettati ad ogni livello i requisiti funzionali e i vincoli. In caso affermativo è possibile passare alla fase successiva, altrimenti occorre andare a modificare le scelte effettuate e procedere con un'altra iterazione. Dopo un numero finito di iterazioni si ottiene in uscita la stima delle dimensioni, delle masse e delle inerzie, oltre che il set di attuatori che si utilizzerà per la movimentazione del sistema. La scelta di quest'ultimo è uno degli elementi fondamentali per la progettazione meccanica di un manipolatore, poiché è sulla base di essa che si andrà ad effettuare il design avanzato di tutti gli altri componenti meccanici del braccio robotico.

A questo punto è possibile passare alla seconda fase, la quale riguarda la scelta dei materiali e la progettazione al CAD degli assiemi dei giunti e dei link del manipolatore: ai componenti commerciali scelti si aggiungono quelli customizzati, i quali vengono modellati in 3D per poi procedere alla realizzazione della messa in tavola, necessaria alla successiva produzione. In parallelo viene effettuata la scelta dell'end effector da assemblare sul manipolatore. L'intero processo fornisce come risultato la configurazione definitiva del manipolatore ed il modello cinematico-dinamico reale, contenente le dimensioni, le masse e le inerzie effettive del sistema.

3.3.1 Design cinematico preliminare

Tramite il design cinematico preliminare si vuole andare a determinare il modello cinematico teorico del manipolatore. Una delle decisioni più importanti riguarda il numero di gradi di libertà di cui si vuole dotare il sistema. Questo parametro è fondamentale perché, una volta fissato, andrà ad influenzare tutti gli aspetti del sistema: le dimensioni e la forma dello spazio di lavoro, la destrosità del robot, il sistema di attuazione necessario, la complessità del robot. Risulta chiaro che maggiore è il numero di gradi di libertà che il sistema possiede, maggiore sarà la sua abilità di eseguire i task all'interno dello spazio di lavoro. Un robot dotato di 6 gradi di libertà è in grado di posizionare l'end effector in qualsiasi punto all'interno del proprio spazio di lavoro con la posa desiderata. L'aggiunta di ulteriori gradi di libertà rende il robot ridondante, aumentandone la versatilità dei movimenti. Di contro, ad un maggior numero di gradi di libertà corrisponde un aumento della complessità del sistema, poiché si rende necessaria la presenza di ulteriori attuatori, sensori e componenti, facendone crescere la massa ed il costo complessivo.

Non meno importante del numero è la tipologia dei gradi di libertà di cui si vuole dotare il sistema: differenti tipologie permettono differenti movimenti relativi dei link successivi, necessitano di differenti sistemi di attuazione e trasmissione, modificano le dimensioni e la forma dello spazio di lavoro.

Per il braccio robotico sviluppato, si è scelta una configurazione a 4 gradi di libertà rotativi: il primo giunto è posizionato alla base e permette la rotazione attorno all'asse di yaw, mentre i tre giunti successivi permettono la rotazione attorno all'asse di pitch. In Figura 3.2 è illustrata la schematizzazione della configurazione scelta per il manipolatore. Sono evidenziati gli assi di rotazione dei giunti con le rispettive variabili di giunto q_i ed i sistemi di riferimento dei link per i quali si è adottata la convenzione di D-H. I parametri di D-H corrispondenti sono elencati in Tabella 3.1: i valori scelti permettono di posizionare il sistema di riferimento 1 vicino alla base del robot, in modo da ottenere un design più compatto e diminuire i momenti di inerzia generati dalla massa del giunto 2.

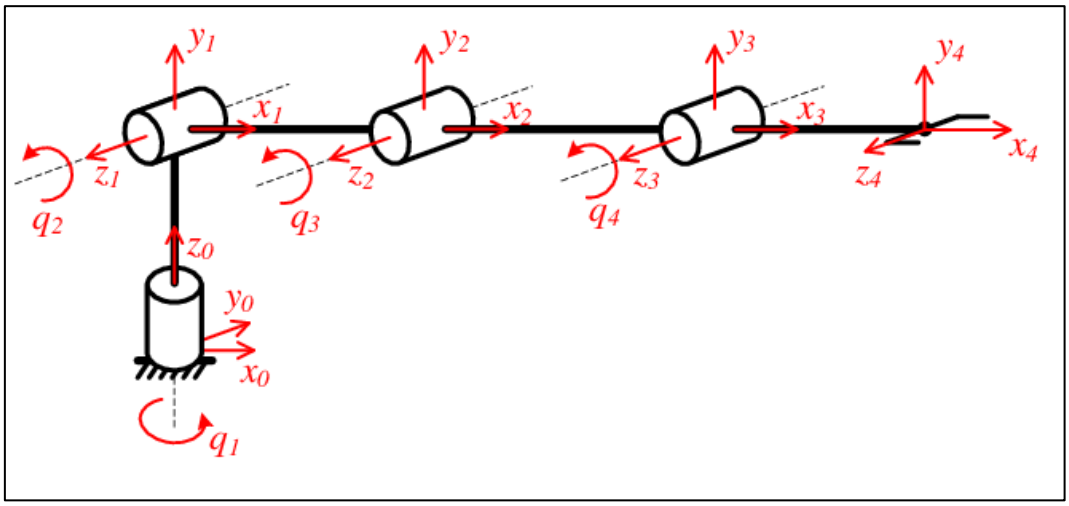

Figura 3.2 – Configurazione cinematica scelta per il braccio robotico.

$\boldsymbol{i}$	$\boldsymbol{\alpha_{i-1}}$ $[rad]$	$\boldsymbol{a_{i-1}}$ $[mm]$	$\boldsymbol{d_i}$ $[mm]$	$\boldsymbol{\vartheta_i}$ $[rad]$
1	$\pi/2$	0	50	q_1
2	0	570	0	q_2
3	0	570	0	q_3
4	0	100	0	q_4

Tabella 3.1 – Parametri di Denavit-Hartenberg del braccio robotico di Morpheus.

I requisiti di raggiungibilità dello spazio di lavoro, di forma ellissoidale con semiassi di 800x500x500 mm, sono soddisfatti grazie alle dimensioni dei due link principali, i numeri 2 e 3, entrambi di lunghezza pari a 570 mm. Il sistema di riferimento dell'end effector è stato posizionato ad ulteriori 100 mm di distanza dal precedente. In Figura 3.3 viene mostrato lo spazio di lavoro raggiungibile dal manipolatore, nel piano perpendicolare agli assi dei giunti 2, 3 e 4. La presenza ulteriore del giunto 1 alla base, il quale permette la rotazione attorno all'asse di yaw (asse Y in Figura 3.3), consente l'estensione dello spazio di lavoro per tutti i 360° di rotazione.

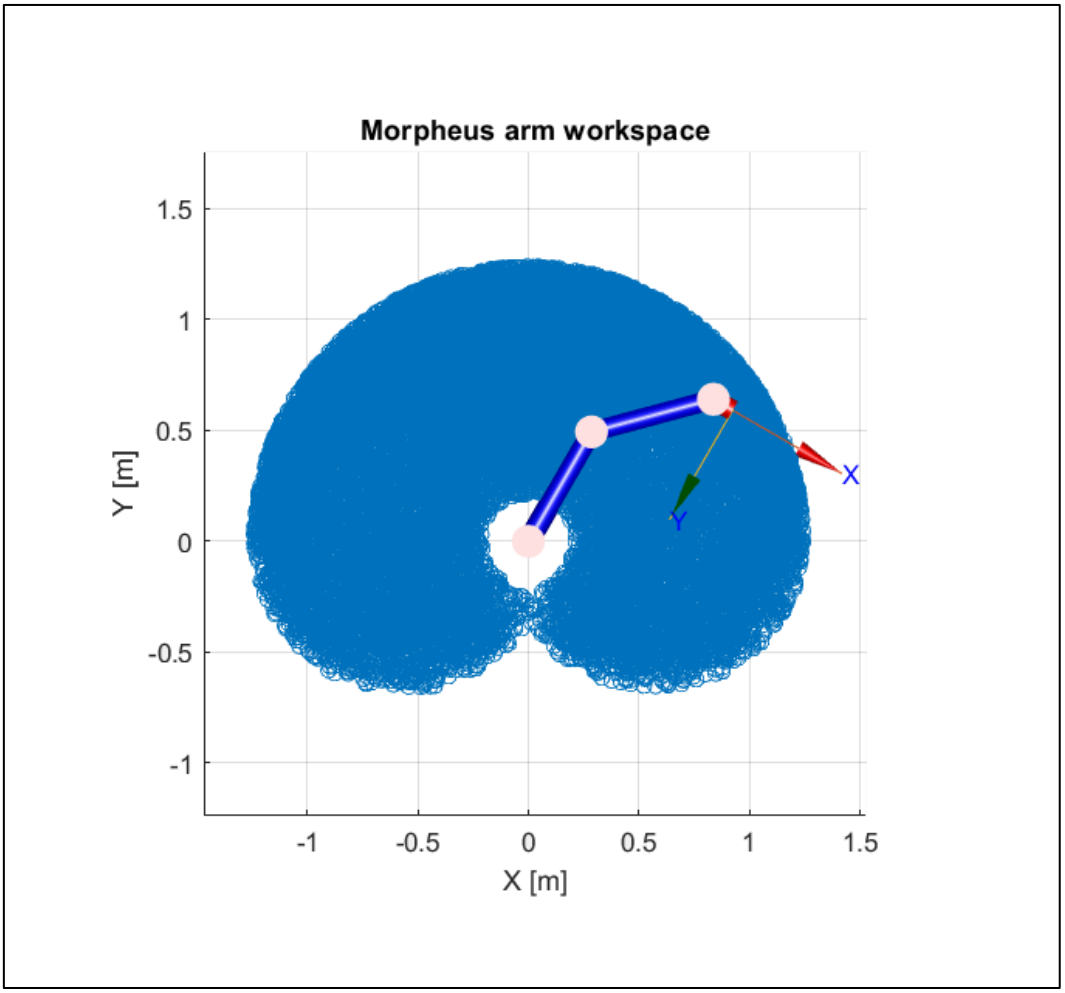

Figura 3.3 – Spazio di lavoro del manipolatore.

Le decisioni che sono state prese per la realizzazione della configurazione cinematica del braccio robotico sono frutto di un compromesso tra la complessità del sistema ed i task effettivamente realizzabili, in funzione dei fondi disponibili da dedicare al progetto. Nello specifico, è stato tenuto in considerazione che il compito principale del manipolatore sarà la raccolta e la stiva a bordo di campioni di terreno. Un sistema a 4 gradi di libertà rotativi con assi orientati secondo la configurazione descritta è in grado di posizionare il proprio end effector in un qualsiasi punto contenuto all'interno dello spazio di lavoro, e consente inoltre di controllare l'angolo di pitch che il sistema di riferimento solidale all'end effector descrive con il riferimento fisso alla base del robot. Ciò risulta sufficiente per l'applicazione desiderata.

3.3.2 Scelta dei motoriduttori

Il modello cinematico del manipolatore appena definito è stato utilizzato per la realizzazione delle simulazioni all'interno del processo iterativo, che hanno consentito la scelta del set di attuatori che andrà a movimentare il sistema. Sono stati considerati solamente motoriduttori elettrici, in quanto unica categoria accettabile per l'applicazione considerata.

Le simulazioni vengono svolte in ambiente virtuale Matlab e Simulink. Il modello fisico del braccio robotico è realizzato attraverso Simscape, un tool di Simulink che ne facilita notevolmente la modellazione, oltre a permetterne l'integrazione diretta con il diagramma a blocchi circostante. Il modello contiene al suo interno la schematizzazione dei link e dei giunti, oltre che le masse e le inerzie del manipolatore. In prima approssimazione, queste vengono considerate concentrate sui giunti e sull'end effector, mentre si considerano i link come privi di massa. Un blocco Simulink si occupa della generazione della traiettoria dei giunti. Nelle simulazioni effettuate si sono ricreate le condizioni più sfavorevoli per il sistema, in modo da rimanere conservativi: il braccio robotico viene mantenuto completamente esteso e fatto ruotare di un angolo di manovra desiderato, in direzione opposta al vettore gravità. Lo schema a blocchi realizzato in Simulink per le simulazioni preliminari è visibile in Figura 3.4, mentre in Figura 3.5 viene illustrato il modello fisico del braccio robotico realizzato in Simscape.

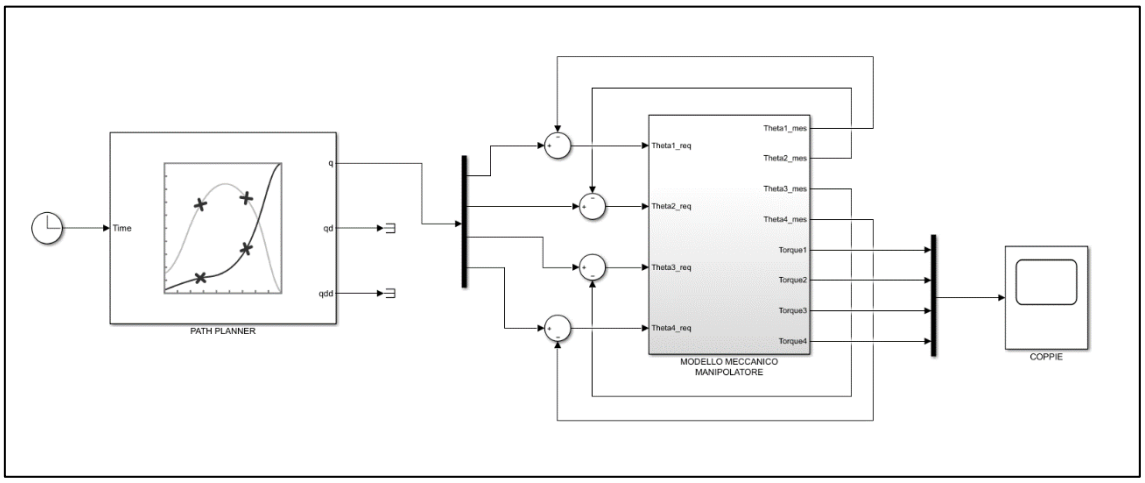

Figura 3.4 – Schema a blocchi Simulink realizzato per le simulazioni preliminari.

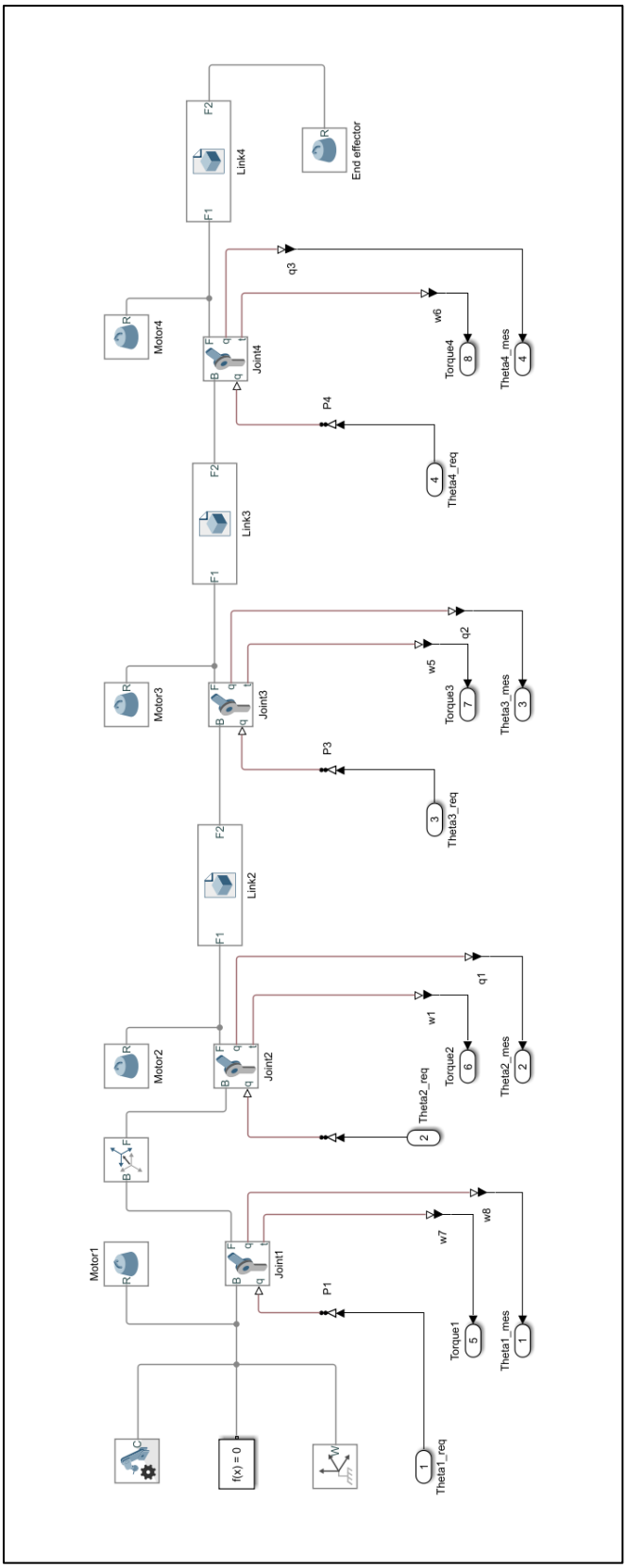

Figura 3.5 – Modello fisico del braccio robotico realizzato in Simscape.

Una volta preparato l'ambiente di simulazione ed il modello fisico del manipolatore, è stato possibile procedere alla scelta dei motoriduttori da installare ai giunti, tramite processo iterativo. Gli step seguiti vengono ora descritti, mentre lo schema a blocchi riassuntivo e gli andamenti delle coppie ai giunti ottenuti per un tempo di manovra di 10 s sono mostrati in Figura 3.6 e in Figura 3.7 rispettivamente.

1. All'interno del modello meccanico viene inserita una prima ipotesi per le masse concentrate ai giunti e all'end effector. Da queste si calcolano i momenti di inerzia corrispondenti;
2. Si procede all'analisi del comportamento del sistema durante diverse manovre. Si imposta il vettore posizione target nello spazio dei giunti $\bar{q}$ ed il tempo di manovra t_{man}. Essi vengono forniti al blocco path planner, che genera una traiettoria polinomiale di terzo grado da fornire ai giunti. In uscita si ottiene l'andamento delle coppie motrici ai giunti durante il moto. Si individua la coppia massima τ_{max} in valore assoluto, e si calcola il valore di coppia root mean square τ_{rms} durante la manovra, secondo la formula:

$$\tau_{rms} = \sqrt{\frac{1}{t_{man}} \int_0^{t_{man}} [\tau(t)]^2 dt} \qquad (3.15)$$

3. Si effettua una prima scelta di motoriduttori, selezionando quelli meno costosi che soddisfino le richieste di coppia massima e rms. τ_{motor_peak} e $\tau_{motor\_continuous}$ sono rispettivamente le coppie massime all'albero del motore in funzionamento intermittente e continuo, mentre N è il rapporto di riduzione del riduttore associato all'attuatore scelto:

$$\tau_{motor_{peak}} > \frac{|\tau_{max}|}{N} \qquad (3.16)$$

$$\tau_{motor_{continuous}} > \frac{\tau_{rms}}{N} \qquad (3.17)$$

4. Le masse dei motoriduttori individuati vengono quindi utilizzate per aggiornare il modello meccanico del manipolatore, considerando che la massa complessiva di ogni giunto sia pari al doppio di quella del motoriduttore associato. Con il modello meccanico aggiornato vengono rieseguite le simulazioni, andando a verificare la compatibilità fra le caratteristiche di coppia massima dei motoriduttori scelti con i risultati ottenuti. In caso negativo, occorre ritornare al punto 3 ed effettuare una diversa scelta, altrimenti si procede.

   Dai datasheet dei motoriduttori scelti, si determinano le costanti di coppia K_T dei motori e si calcolano le correnti massime e rms assorbite durante la manovra. Queste devono essere confrontate con i valori di corrente accettabili dagli avvolgimenti dei motori, in funzionamento intermittente e continuo, presenti all'interno dei datasheet:

$$I_{motor_{peak}} > I_{max} = \frac{|\tau_{max}|}{NK_T} \quad (3.18)$$

$$I_{motor_{continuous}} > I_{rms} = \frac{\tau_{rms}}{NK_T} \quad (3.19)$$

   L'ultima verifica tiene in considerazione che, durante il funzionamento del sistema, i motori scelti saranno alimentati tramite sistema di potenza del rover Morpheus, il quale consiste in due coppie di batterie al piombo. È necessario dunque controllare che i valori di tensione e corrente di alimentazione, in funzionamento nominale ed intermittente, non superino mai quelli massimi erogabili dalle batterie.

   Se tutte le verifiche sono soddisfatte, il set di motoriduttori scelto è soddisfacente per l'applicazione sul braccio robotico da integrare sul rover Morpheus, altrimenti occorre ritornare al punto 3 e procedere con un'altra iterazione.

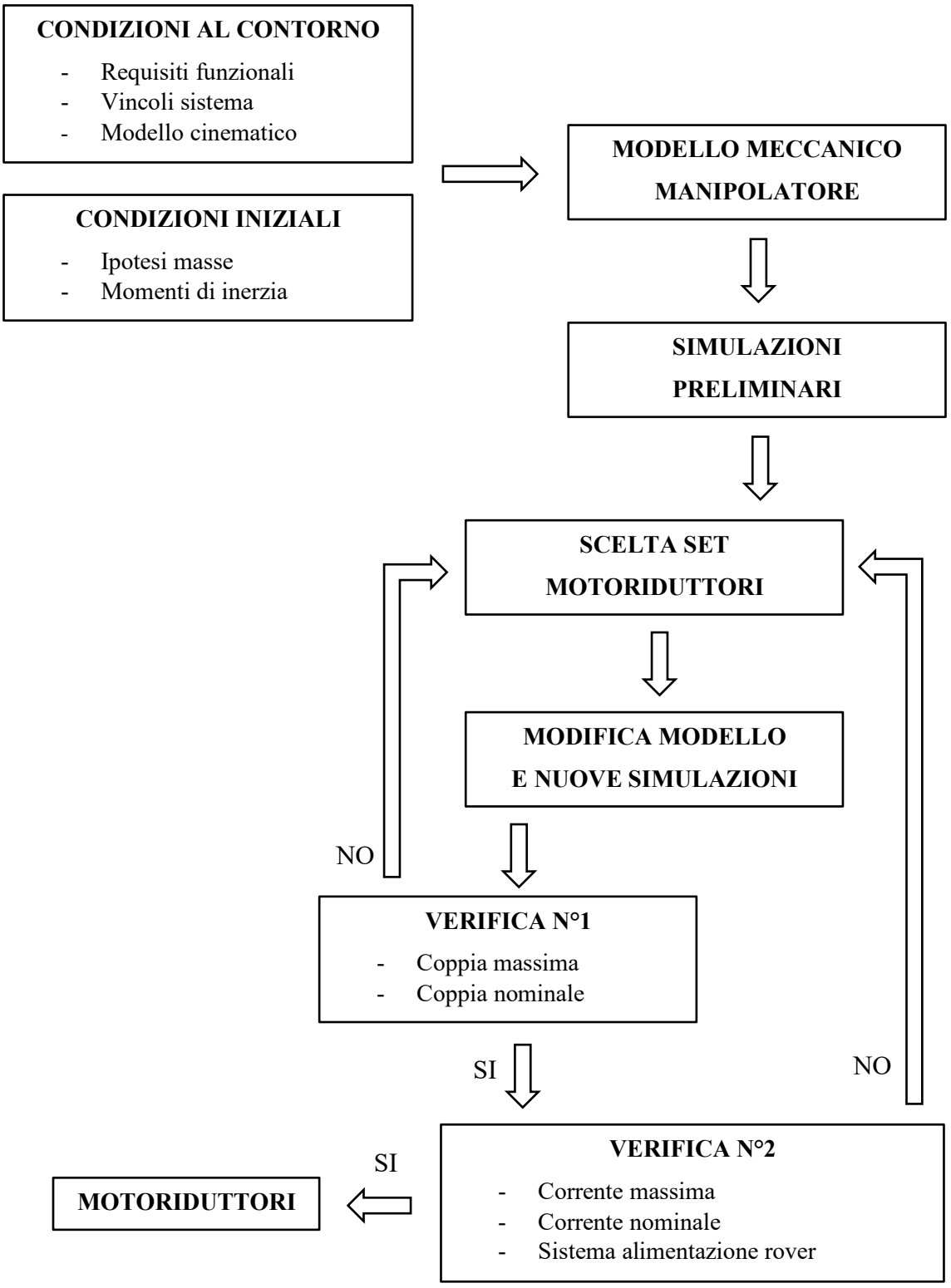

Figura 3.6 – Schema a blocchi del processo di scelta dei motoriduttori per il braccio robotico di Morpheus.

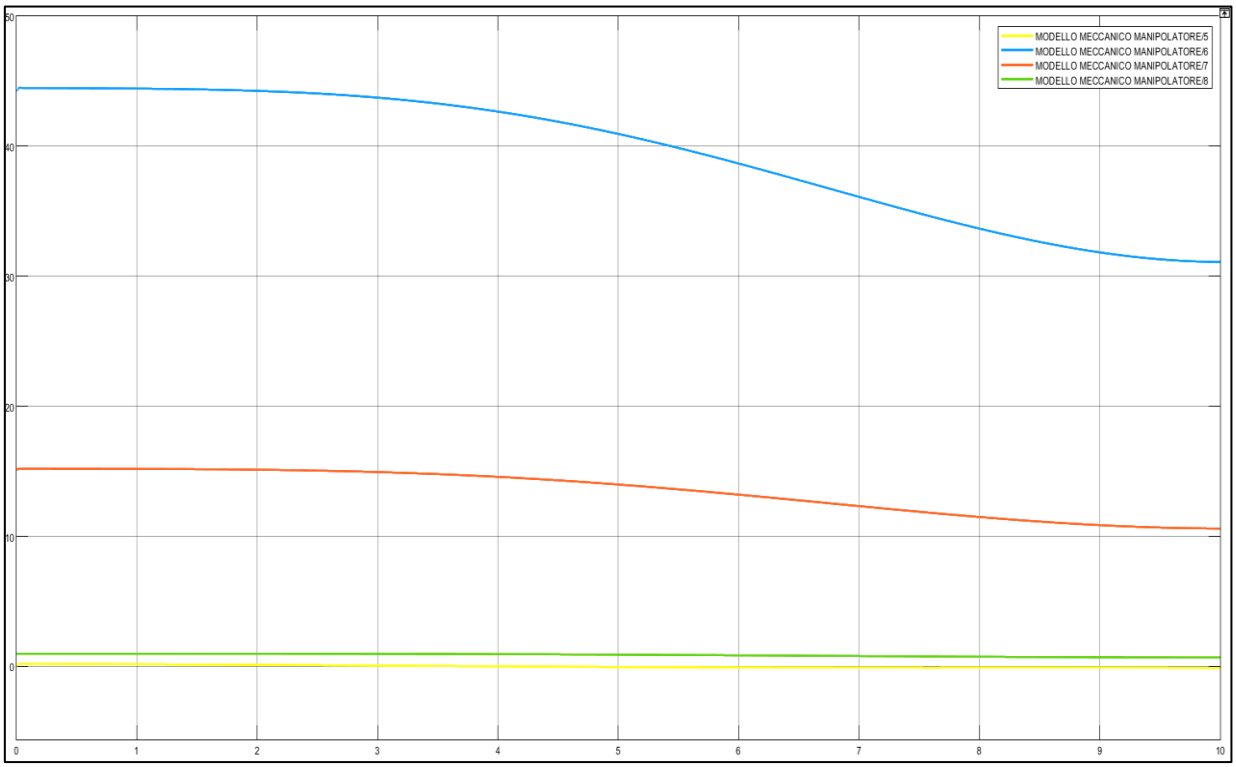

Figura 3.7 – Andamenti delle coppie ai giunti, tempo di manovra di 10 s.

Durante l'esecuzione delle leggi di moto desiderate ai giunti si desiderano basse velocità di rotazione e coppie motrici elevate. Inoltre, in aggiunta ai requisiti descritti ai punti 3 e 4 del paragrafo 3.3.2, il sistema di attuazione deve garantire:

- Basse inerzie rotoriche ed elevato rapporto potenza/peso;
- Possibilità di sovraccarico per brevi intervalli di tempo;
- Capacità di sviluppare elevate accelerazioni e decelerazioni;
- Ampio intervallo di velocità angolari;
- Elevata precisione di posizionamento.

Dunque, per il manipolatore sono stati scelti motori elettrici brushless EC (electronic commutation) a magneti permanenti, con riduttore epicicloidale a più stadi integrato. I motori elettrici presentano uno statore fisso alla carcassa, sul quale sono alloggiati gli avvolgimenti, o le fasi, del motore. Lungo ogni avvolgimento viene fatta circolare una corrente continua, che permette di generare un campo magnetico con l'orientamento desiderato.
Sul rotore, la parte rotante del motore, sono posizionati i magneti permanenti: essi tendono a trascinare il rotore in modo tale da allinearsi al campo magnetico generato dagli avvolgimenti, generando coppia motrice. Tipicamente questi sono motori trifase, e dunque gli avvolgimenti statorici sono tre e vengono ugualmente ripartiti attorno all'asse centrale del motore. Al fine di ottenere una coppia motrice costante ad ogni istante risulta necessario effettuare continue commutazioni sulla circolazione della corrente tramite un inverter, in ogni avvolgimento e nel senso corretto, in funzione della posizione angolare del rotore. Per questo, all'attuatore deve essere sempre accoppiato almeno un sensore di posizione relativa fra rotore e statore, tipicamente ad effetto Hall o encoder, che fornisce l'informazione sulla posizione del rotore all'inverter, per effettuare la commutazione. La commutazione elettronica dei motori brushless permette di eliminare i problemi di usura e le perdite per attrito dovute alla presenza delle spazzole nei classici motori brushed, nei quali la commutazione è di tipo meccanico. Una rappresentazione schematica dei componenti che costituiscono un motore brushless viene fornita in Figura 3.8.

I riduttori epicicloidali permettono di ottenere elevati rapporti di riduzione, trasmettendo elevate potenze meccaniche mantenendo contenuti gli ingombri e le masse. Queste peculiarità li rendono particolarmente adatti ad applicazioni in campo robotico.

Un rotismo epicicloidale è costituito da quattro componenti principali: una ruota dentata esterna principale detta solare, che mantiene l'asse di rotazione fisso durante il moto, una ruota dentata interna detta corona, posizionata esternamente e mantenuta fissa, una o più ruote dentate esterne denominate satelliti, i quali ingranano con la corona e con il solare, e i cui assi ruotano durante il moto, perché collegati ad un elemento rigido chiamato porta satelliti, che ruota attorno allo stesso asse fisso della ruota solare. Più riduttori epicicloidali possono essere collegati in serie, connettendo l'albero condotto del primo con l'albero motore del successivo, andando a creare un meccanismo a più stadi e permettendo di ottenere rapporti di riduzione elevati, fondamentali se ai giunti del manipolatore devono essere forniti elevati livelli di coppia motrice. Una rappresentazione schematica dei componenti che costituiscono un riduttore epicicloidale viene fornita in Figura 3.10.

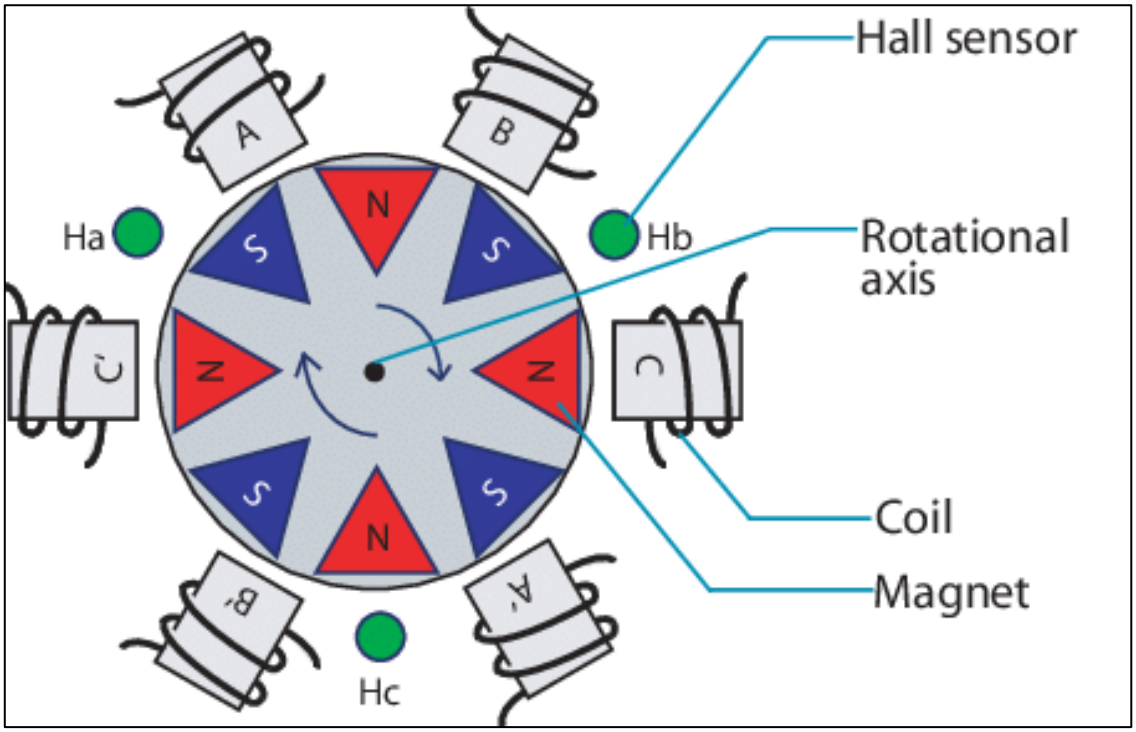

Figura 3.8 – Rappresentazione schematica di un motore elettrico brushless EC.

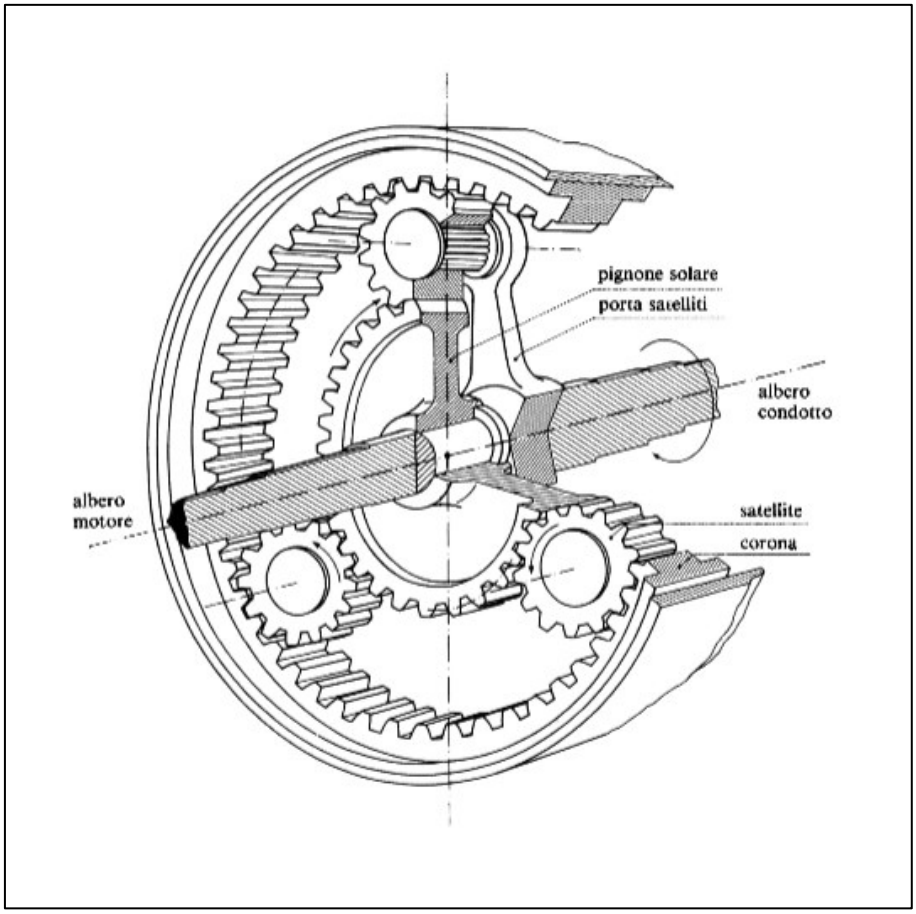

Figura 3.9 – Rappresentazione schematica di un riduttore epicicloidale.

3.3.3 Motoriduttori selezionati

I motoriduttori che sono stati scelti per azionare i giunti del braccio robotico da integrare sul rover Morpheus sono prodotti dall'azienda Maxon Motor. Motore elettrico, riduttore epicicloidale e sensori vengono venduti già integrati. Per effettuare la commutazione delle correnti negli avvolgimenti sono presenti tre sensori di Hall e un encoder ottico relativo. I motoriduttori non possono essere collegati direttamente al sistema di potenza, ma devono essere alimentati e controllati tramite delle schede di controllo dedicate, sempre prodotte dalla Maxon Motor.

Il giunto numero 1, posizionato alla base del rover, si occupa di modificare il piano verticale in cui il manipolatore andrà ad operare e necessita di piccole coppie motrici per il suo azionamento anche durante le manovre più gravose. Per questo è stato scelto un motoriduttore con autorità contenuta. Ciò è sinonimo di dimensioni contenute, fattore favorevole perché occorre considerare che una volta che il manipolatore sarà installato sul rover, il motoriduttore del giunto 1 avrà asse verticale e dovrà essere contenuto all'interno dello spessore del telaio. Il motore ed il riduttore sono mostrati in Figura 3.10 mentre le caratteristiche principali sono elencate in Tabella 3.2.

I giunti numero 2 e 3 sono quelli che richiedono le coppie di azionamento maggiori, e dunque anche la taglia degli attuatori dovrà aumentare di conseguenza. Per entrambi i giunti si è scelta la stessa tipologia di motoriduttore, mostrata in Figura 3.11 e con le principali caratteristiche elencate in Tabella 3.3.

I motoriduttori finora descritti sono della serie EC-i. Per il giunto 4 è stata scelta una diversa tipologia di motore, della serie EC-flat, in modo tale da favorire la compattezza assiale dell'assieme del giunto rispettando i requisiti funzionali. A differenza dei precedenti, i motori della serie flat presentano un rotore esterno e dunque posizionato più lontano dell'asse di rotazione. Questo permette di realizzare coppie più elevate, comportando però un aumento di inerzia rotorica. Dunque, risulta preferibile utilizzare questi motori con piccole velocità di rotazione, casistica adatta all'applicazione. Il motore ed il riduttore sono mostrati in Figura 3.12 mentre le caratteristiche principali sono elencate in Tabella 3.4. Occorre notare che tutti i motori scelti possiedono costanti termiche degli avvolgimenti sufficientemente elevate. Questo fattore permette ai motori di lavorare attorno al livello di coppia massima intermittente per un tempo dell'ordine di decine di secondi, in modo tale da soddisfare il requisito sulla capacità di sovraccarico.

Figura 3.10 - Motore e riduttore selezionati per il giunto 1.

Motore giunto 1	
Modello	Maxon motor EC-i 30
Tensione alimentazione nominale [V]	24
Corrente alimentazione nominale [A]	3.64
Corrente alimentazione massima [A]	10
Costante termica avvolgimenti [s]	32.7
Costante di coppia [mNm/A]	28.6
Efficienza massima	88%
Massa [kg]	0.24
Dimensioni [mm]	Ø30x64
Sensori	effetto Hall / Encoder relativo
Riduttore giunto 1	
Modello	Planetary gearhead GP32C
Rapporto di riduzione	103:1
Numero di stadi	3
Coppia massima nominale [N/m]	6
Coppia massima intermittente [N/m]	7.5
Potenza massima nominale [W]	49
Potenza massima intermittente [W]	61
Massa [kg]	0.19
Dimensioni [mm]	Ø32x43.1

Tabella 3.2 – Caratteristiche principali motoriduttore giunto 1.

Figura 3.11 - Motore e riduttore selezionati per i giunti 2 e 3.

Motore giunti 2 e 3	
Modello	Maxon motor EC-i 52
Tensione alimentazione nominale [V]	24
Corrente alimentazione nominale [A]	8.96
Corrente alimentazione massima [A]	24
Costante termica avvolgimenti [s]	19.9
Costante di coppia [mNm/A]	48.1
Efficienza massima	90%
Massa [kg]	0.82
Dimensioni [mm]	Ø52x80
Sensori	effetto Hall / Encoder relativo
Riduttore giunti 2 e 3	
Modello	Planetary gearhead GP52C
Rapporto di riduzione	81:1
Numero di stadi	3
Coppia massima nominale [N/m]	35
Coppia massima intermittente [N/m]	50
Potenza massima nominale [W]	230
Potenza massima intermittente [W]	350
Massa [kg]	0.77
Dimensioni [mm]	Ø52x78.5

Tabella 3.3 – Caratteristiche principali motoriduttore giunti 2 e 3.

Figura 3.12 - Motore e riduttore selezionati per il giunto 4.

Motore giunto 4	
Modello	Maxon motor EC45 flat
Tensione alimentazione nominale [V]	24
Corrente alimentazione nominale [A]	3.21
Corrente alimentazione massima [A]	10
Costante termica avvolgimenti [s]	29.6
Costante di coppia [mNm/A]	36.9
Efficienza massima	85%
Massa [kg]	0.14
Dimensioni [mm]	Ø42.8x26.7
Sensori	effetto Hall / Encoder relativo
Riduttore giunto 4	
Modello	Planetary gearhead GP42C
Rapporto di riduzione	156:1
Numero di stadi	3
Coppia massima nominale [N/m]	15
Coppia massima intermittente [N/m]	22
Potenza massima nominale [W]	81
Potenza massima intermittente [W]	120
Massa [kg]	0.46
Dimensioni [mm]	Ø42x70

Tabella 3.4 – Caratteristiche principali motoriduttore giunto 4.

3.3.4 Design dei giunti e dei link

Per il design meccanico dei giunti del manipolatore del rover Morpheus, si è partiti dalle dimensioni di ingombro degli assiemi motore-riduttore individuati al paragrafo 3.3.3. Ciò poiché si è cercato di realizzare un design dei giunti che permetta di integrare i motoriduttori al proprio interno, in modo tale da ridurre le dimensioni assiali e quindi i carichi a sbalzo sul sistema. Tutti i componenti customizzati sono stati realizzati in lega di alluminio-silicio-magnesio-manganese AA6082. Le leghe 6000 presentano un buon compromesso fra resistenza meccanica e densità, offrendo resistenza alla corrosione e una ottima lavorabilità alle macchine utensili.

Il giunto numero 1 del manipolatore è quello che ne permette l'interfaccia ed il collegamento con il telaio del rover. Il vincolo principale che è stato tenuto in considerazione durante la sua progettazione riguarda la posizione di installazione del braccio robotico sul rover. Il manipolatore, infatti, non può essere installato in un qualsiasi punto a bordo del rover ma devono considerarsi solo le zone dove è presente un supporto strutturale fornito dal telaio, e dove non si vada ad interferire con gli altri sottosistemi di Morpheus. Dopo diverse analisi si è scelto di collegare il braccio al telaio nella zona di testa del rover, in modo tale che, quando completamente retratto, si mantenga all'interno di quest'ultimo. Inoltre, sugli elementi del telaio erano già presenti dei fori per il collegamento, e si è deciso di sfruttarli evitando di praticarne altri per non indebolire ulteriormente la struttura.

Dunque, per il giunto numero 1 è stata prevista una piastra di base per il supporto, di dimensioni 120x110 mm e di spessore 10 mm. Su di essa sono presenti 4 fori per il collegamento a telaio, con lo stesso interasse di quelli sul rover, ed un foro centrale che permette il passaggio dell'albero del motore. Quest'ultimo viene collegato con delle viti nella zona inferiore della piastra.

Nella parte superiore è collegata una flangia, che permette l'assemblaggio di una ralla. La ralla acquistata è il modello IPT090 e prodotta dalla Otar S.p.A., con diametro esterno di 129 mm e diametro interno di 51mm. Sulla ralla sono presenti due anelli, uno fisso ed uno mobile, ognuno con una serie di fori che ne permettono il collegamento alla flangia fissa ed al link successivo rotante. La ralla consente di sostenere i carichi assiali derivanti dal peso del braccio robotico ed i momenti ribaltanti che si generano durante l'esecuzione dei task.

L’anello mobile della ralla è collegato ad una seconda flangia che viene posta in rotazione dal motoriduttore. Quest’ultimo non è direttamente connesso alla flangia ma tra di essi è posizionato un giunto elastico: il componente utilizzato è il Ruland FCMR 16-6-6-A. Due viti permettono di assicurare per attrito il giunto all’albero del motore e all’albero di uscita, che trascina la flangia superiore tramite un collegamento con linguetta. Si è ritenuto necessario inserire un giunto elastico fra i due alberi in modo tale da consentire la presenza di disallineamenti radiali e per proteggere il motore da eventuali urti o elevati carichi radiali.

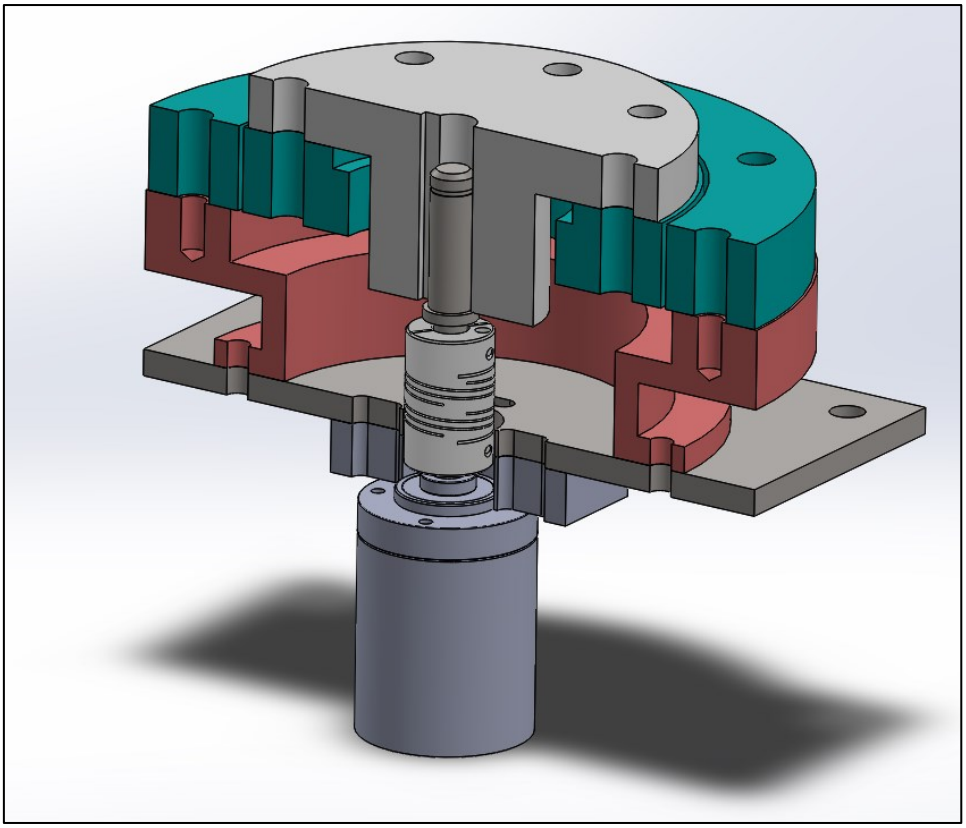

Figura 3.13 - Vista in sezione del giunto N°1 del manipolatore.

Per i giunti 2, 3 e 4, rispettivamente della spalla, del gomito e del polso del manipolatore, si è utilizzato lo stesso design meccanico. Il motoriduttore viene parzialmente inserito all’interno di un coperchio, e fissato con un collegamento filettato in testa ad esso, dove è presente anche un foro per il passaggio dell’albero motore. L’inserimento del motore all’interno della struttura del giunto è stato pensato per permettere di diminuirne notevolmente l’ingombro assiale. Il moto viene trasmesso dall’albero motore tramite una

flangia, collegata ad esso con una linguetta. La flangia viene avvitata ad un ulteriore coperchio che si collega a sua volta ad un manicotto esterno, sul quale viene assemblato il link successivo.

La guida ed il supporto di un albero rotante richiedono la presenza di almeno due cuscinetti, assemblati ad una certa distanza assiale tra loro: per permettere il moto relativo fra il manicotto ed il coperchio del motore, e dunque fra il link precedente e quello successivo, sono stati inseriti due cuscinetti a sfere a sezione sottile. Per i giunti 2 e 3, i cui motori hanno diametro maggiore, sono stati utilizzati i cuscinetti Schaeffler CSCA025, mentre per il giunto 4, il quale ha la stessa configurazione ma dimensioni più contenute, sono stati utilizzati i cuscinetti Schaeffler CSCA020. A differenza dei classici cuscinetti a sfere, i cuscinetti a sezione sottile permettono di avere diametri interni elevati mantenendo una sezione limitata. Questa peculiarità consente di realizzare design con dimensioni radiali più contenute, e dunque di minimizzare la massa e gli ingombri dei giunti. Al fine di compensare le dilatazioni termiche che possono presentarsi durante il funzionamento e di permettere il corretto alloggiamento dell'albero, occorre progettare un corretto bloccaggio dei cuscinetti. Nello specifico, un cuscinetto dovrà essere completamente bloccato in direzione assiale, prevedendo degli spallamenti sia sull'anello interno che su quello esterno. Esso farà da guida all'albero e sopporterà i carichi assiali esterni. Il secondo cuscinetto, invece, deve essere bloccato in modo tale da lasciare libero lo spostamento assiale di uno dei due anelli del secondo cuscinetto. Affinché l'anello possa effettivamente spostarsi per compensare le dilatazioni, è necessario che l'accoppiamento scelto permetta spostamenti, e che quindi sia presente gioco. Il bloccaggio ideato per i cuscinetti scelti è illustrato in Figura 3.15.

I giunti vengono collegati ai link precedenti e a quelli successivi tramite degli elementi "ad L", realizzati dall'unione di due piastre, le quali presentano una serie di fori per il collegamento, e due nervature di rinforzo. Il giunto numero 2 viene collegato direttamente alla flangia condotta del giunto numero 1, realizzando un design compatto alla base del robot. Tra il giunto 2, 3 e 4 sono presenti i due link principali del manipolatore, realizzati con tubolari in fibra di carbonio Carbosix, di diametro 40 mm, spessore 1.5 mm e lunghezza 570 mm ognuno. In testa e in coda ai tubolari sono state incollate due flange di alluminio forate, per il collegamento ai giunti tramite gli elementi ad L.

Per l'incollaggio è stata utilizzata la resina epossidica Scotch-Weld 3M 490, particolarmente adatta per incollaggi tra materiali diversi. Le flange di alluminio sono state precedentemente lavorate con carta vetrata, in modo da aumentarne la rugosità nella zona di interfaccia con i tubolari ed aumentare ulteriormente la presa della resina.

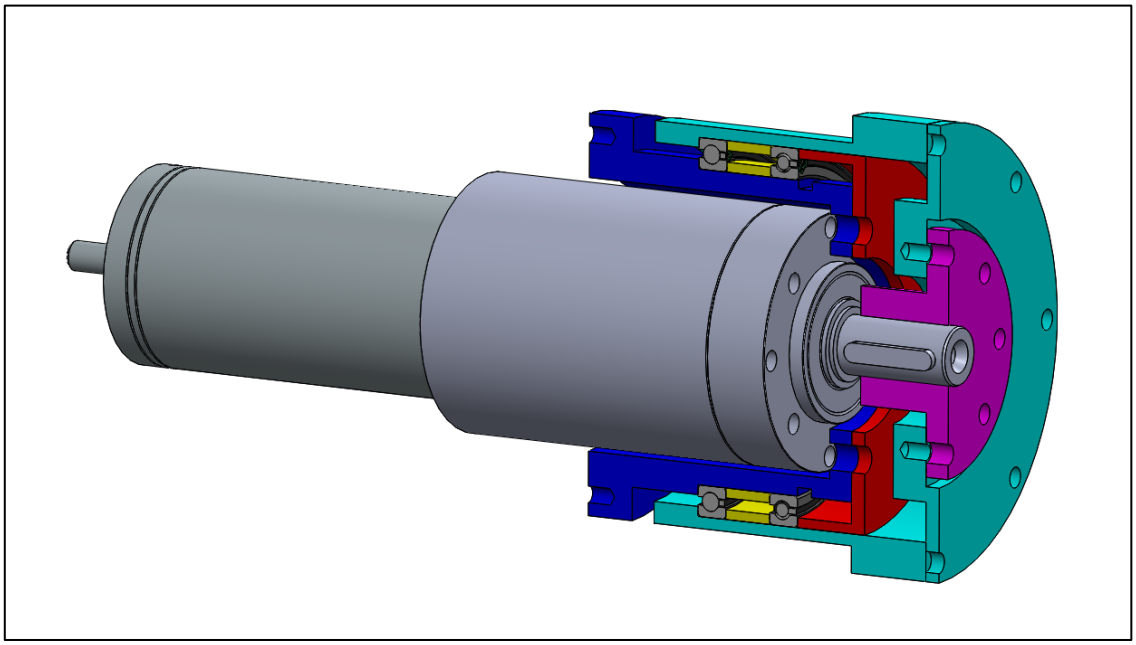

Figura 3.14 - Vista in sezione dei giunti N°2, 3 e 4 del manipolatore.

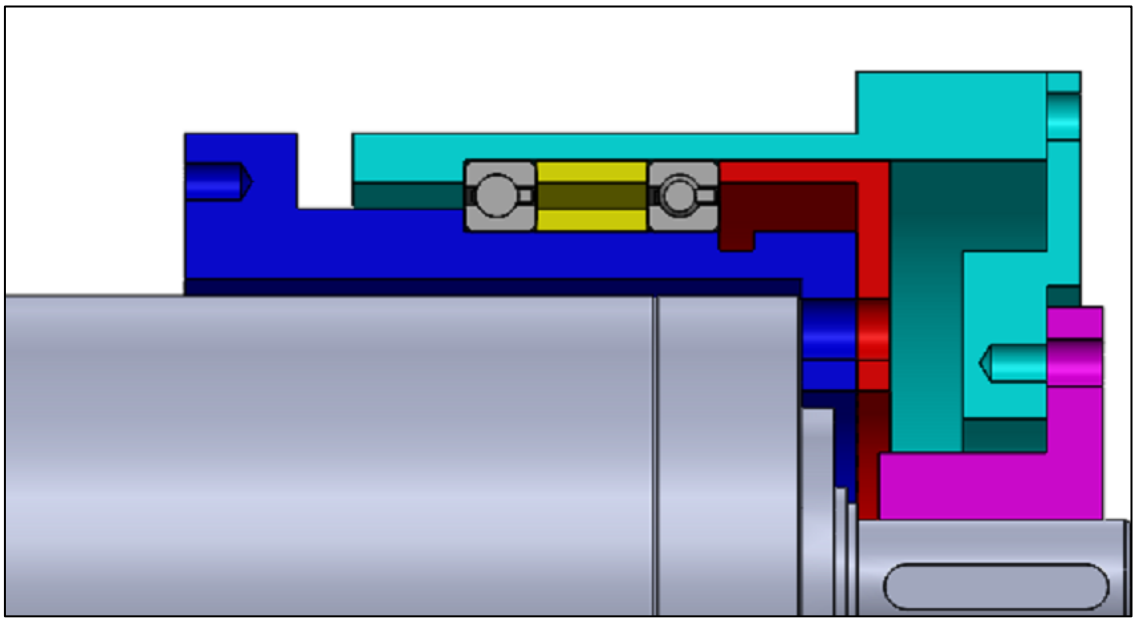

Figura 3.15 – Particolare del sistema di bloccaggio per i cuscinetti dei giunti N° 2, 3 e 4.

3.3.5 Scelta dell'end effector

L'end effector selezionato per la raccolta dei campioni è il gripper Robotiq 2F-85 Adaptive, che ha un'apertura massima delle dita di 85 mm. Il gripper è dotato di due dita realizzate tramite quadrilateri articolati, ognuna formata da due falangi, la cui movimentazione è realizzata tramite un solo attuatore.

La configurazione descritta permette al gripper di adattarsi alla forma dell'oggetto da afferrare, garantendo fino a cinque punti di contatto, di cui quattro con le falangi ed uno con il palmo. Quando viene fornito il comando di chiusura delle dita, il gripper è in grado di realizzare in autonomia il corretto posizionamento delle falangi in modo da effettuare un grasping robusto dell'oggetto, sia in funzione della forma che della posizione relativa fra il gripper e quest'ultimo.

Il gripper è dotato di una base separata, che ne consente l'alimentazione a 24 V ed il trasferimento dei dati, oltre che l'installazione sul braccio robotico tramite 4 viti M6. Essa viene collegata alla piastra di uscita azionata dal giunto numero 4 del manipolatore di Morpheus.

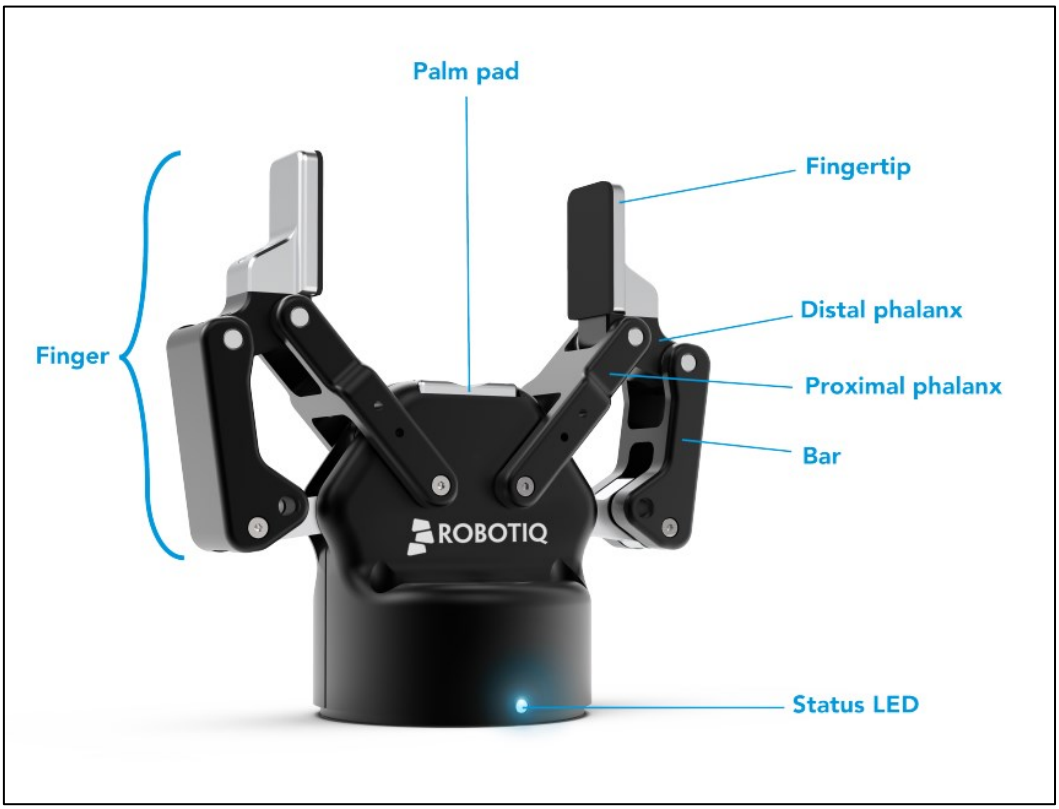

Figura 3.16 – Robotiq 2F-85 Adaptive Gripper.

3.3.6 Il manipolatore assemblato

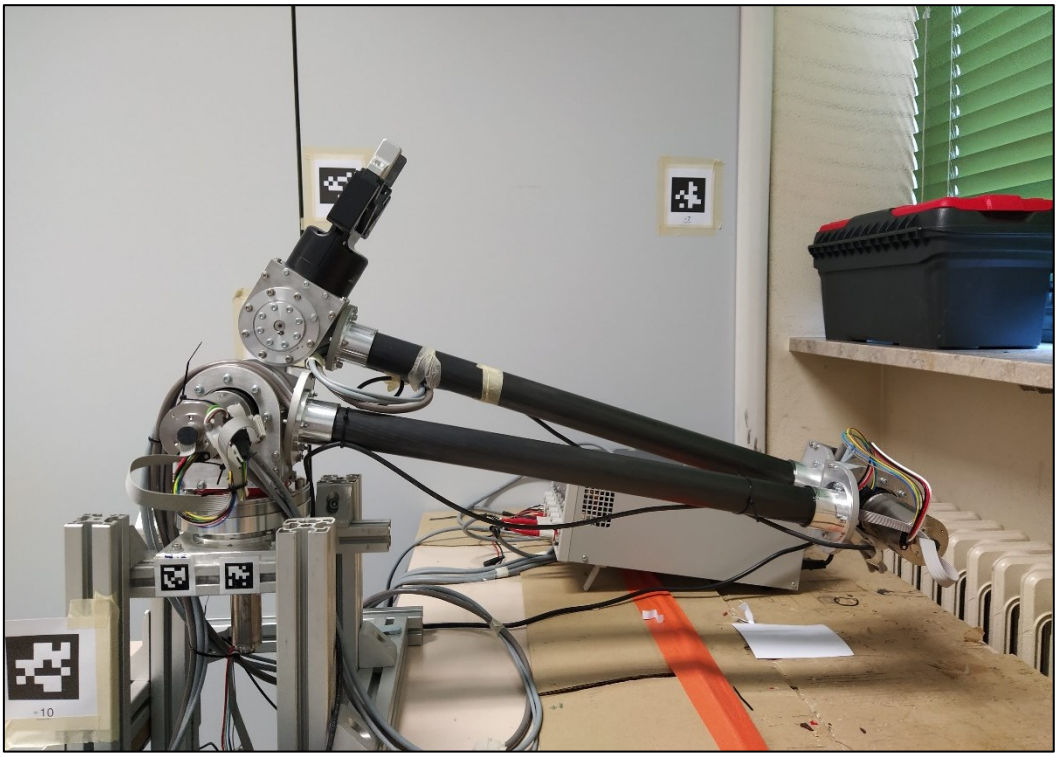

Figura 3.17 – Il manipolatore assemblato al banco di prova.

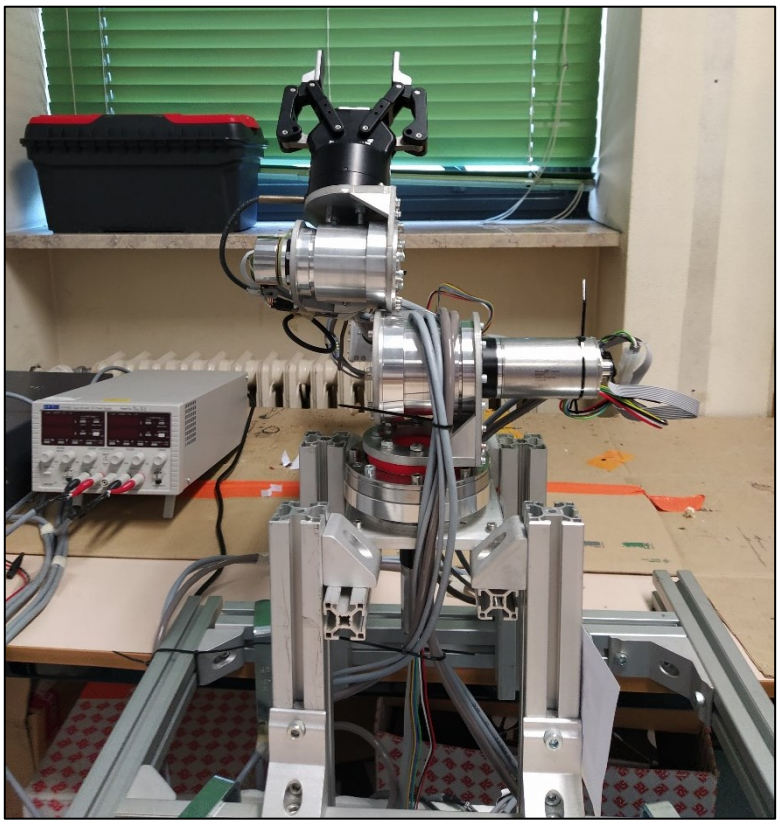

Figura 3.18 – Il manipolatore assemblato al banco di prova.

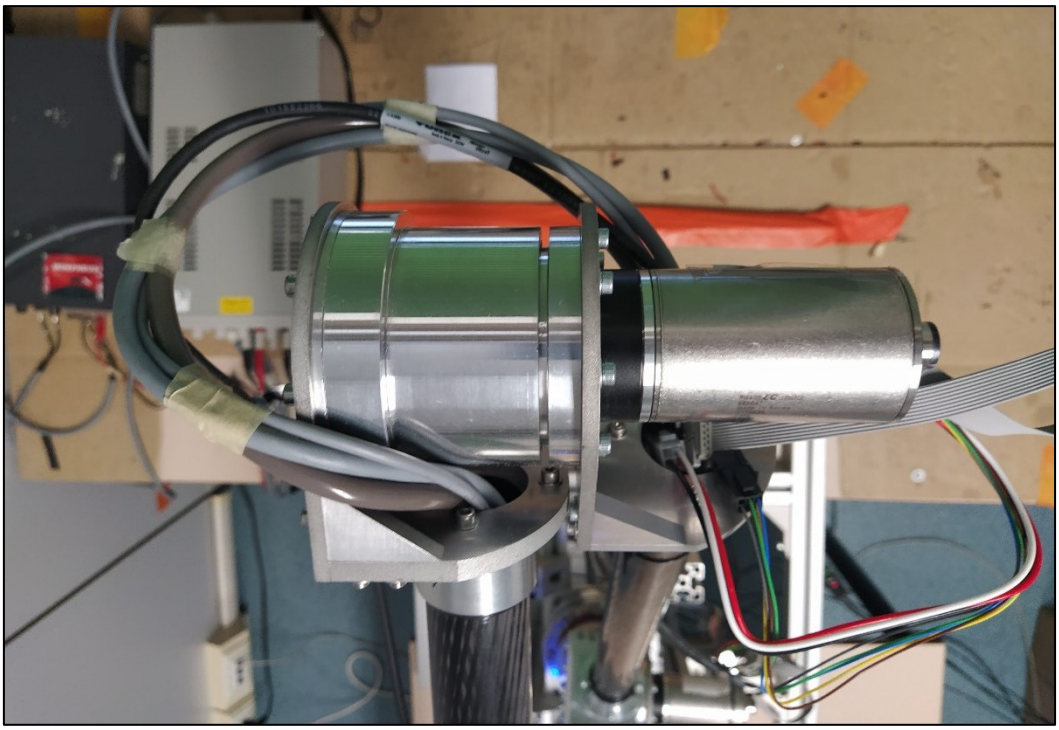

Figura 3.19 – Particolare del passaggio dei cavi all'interno dei link del manipolatore.

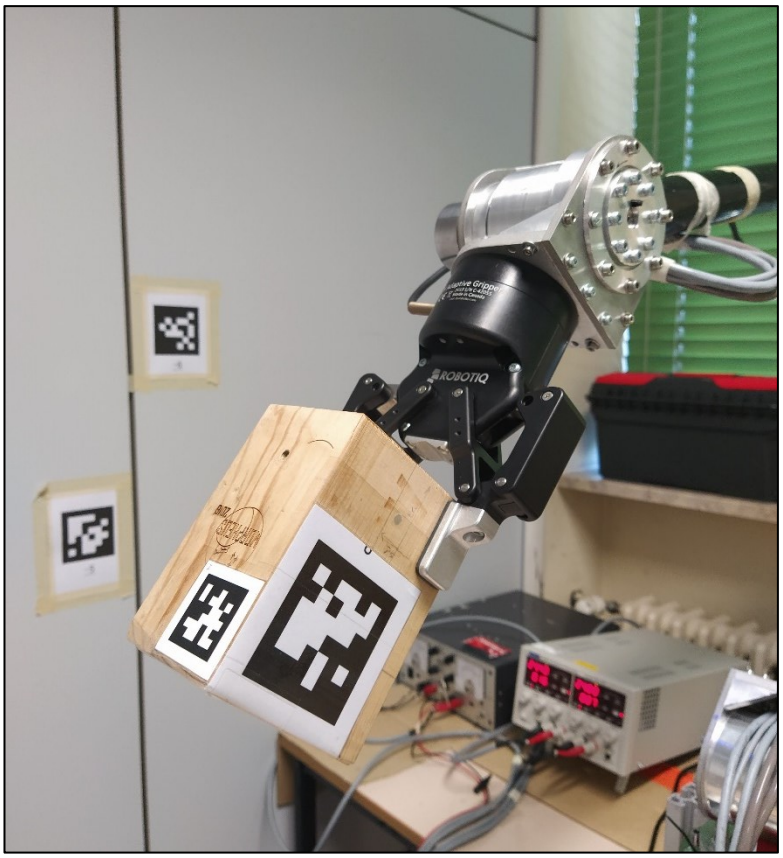

Figura 3.20 – Particolare dell'end effector del manipolatore.

3.4 Controllo dei motoriduttori

3.4.1 Setup dei motoriduttori

Una volta che tutti i componenti del manipolatore sono stati assemblati, si è passati al settaggio dei motoriduttori dei giunti. Questi non possono essere collegati direttamente al sistema di potenza, ma devono essere alimentati e controllati tramite una scheda di controllo dedicata, sempre prodotta dalla Maxon Motor. I prodotti selezionati sono le schede Maxon EPOS4 CAN Compact 50/8 per i motoriduttori dei giunti 2, 3, e 4, e la scheda Maxon EPOS4 CAN Compact 50/5 per il motoriduttore del giunto 1. I moduli EPOS4 Compact sono unità completamente digitali che permettono il controllo del posizionamento dei motori. Le schede presentano un'elevata densità di potenza, fattore che permette di contenerne le dimensioni e di aumentarne la flessibilità, in quanto possono essere utilizzate sia per il controllo di motori DC a spazzole che per motori brushless EC, con potenze fino a 400 W. Esse, inoltre, rendono possibile la lettura dei feedback forniti dai sensori di Hall e dagli encoder, sia incrementali che assoluti. Le unità possono essere comandate e controllate tramite un network CANopen o EtherCAT, oppure tramite una workstation connettendole tramite una porta USB o RS232. Per il braccio robotico di Morpheus si utilizzano connessioni via USB, perché permettono di avere una banda dati sufficientemente larga per l'applicazione e si sono rivelate semplici da implementare.

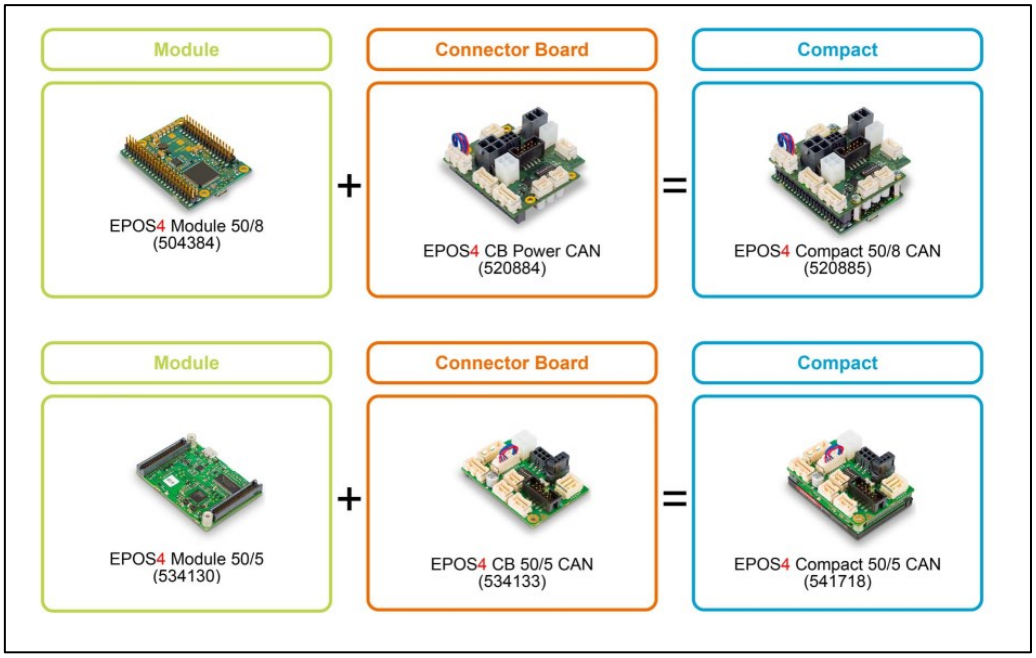

Figura 3.21 – Maxon Epos4 CAN Compact configuration overview.

Sulle schede di controllo sono presenti una serie di ingressi che permettono l'alimentazione, il collegamento dei motoriduttori e della workstation via USB in modalità "Plug&Play", ossia semplice, veloce e pronto all'uso, tramite connettori Molex. In Figura 3.22 vengono illustrate le connessioni per le diverse porte delle schede ed in Tabella 3.5 si elencano quelle utilizzate per i motoriduttori del braccio robotico.

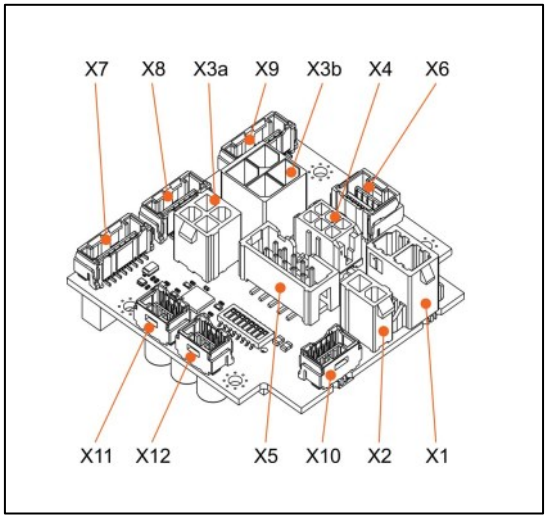

Figura 3.22 – Ingressi presenti sulle schede.

Ingresso	Designazione
X1	Alimentazione scheda 24V
X3a	Alimentazione motore
X4	Sensori di Hall
X5	Encoder
USB	(sul modulo)

Tabella 3.5 – Ingressi utilizzati.

Il setup iniziale è stato realizzato tramite il software dedicato EPOS Studio. Esso è un'interfaccia grafica user-friendly che permette di effettuare la configurazione dei parametri delle schede EPOS, la taratura dei loop dei controllori interni, il controllo dei motoriduttori e l'analisi dei feedback. Una volta creato un nuovo progetto per il singolo motoriduttore, selezionando l'opzione Startup all'interno della scheda Wizards del menu principale si procede con il processo di inizializzazione della scheda di controllo. In questa fase, su di essa vengono caricati i principali dati operativi del motore e del riduttore, quali la corrente nominale e massima, il numero di coppie polari, la costante termica degli avvolgimenti, la costante di coppia, la velocità angolare massima, il rapporto di riduzione, i limiti operativi e di sicurezza. Inoltre, vengono selezionati i sensori presenti e le loro caratteristiche. I motoriduttori scelti presentano 3 sensori di Hall digitali, per i quali occorre definire la polarità e la posizione relativa al rotore, ed un encoder incrementale, per il quale occorre definire il numero di canali, la modalità di decodifica dei dati, il passo e la direzione della rotazione relativa dei due canali principali.

Gli encoder assemblati sui motoriduttori presentano 3 canali, due dei quali, A e B, determinano la rotazione del motore ed il senso del moto, orario o antiorario, mentre un terzo determina la posizione di 0. Il passo degli encoder è di 1024 step al giro e per la decodifica del segnale viene utilizzato il codice Gray.
In questa fase viene anche definita la modalità di commutazione delle correnti sugli avvolgimenti, necessaria perché si utilizzano motoriduttori brushless. La scelta ricade su due tipologie disponibili:

- Commutazione a blocchi: è la modalità di commutazione più semplice in quanto può essere effettuata tramite i soli sensori di Hall. Consiste nel fornire e togliere l'alimentazione agli avvolgimenti con una forma della corrente a gradino. In contrasto, l'utilizzo di quest'ultima genera un ripple di coppia di circa il 14% della coppia nominale per ogni ciclo di commutazione. Si ricorda che il ripple è un'ondulazione della coppia motrice generata proprio dal contenuto armonico presente nella corrente di alimentazione, e che può andare ad eccitare le risonanze del robot;
- Commutazione sinusoidale: in questa modalità la commutazione viene realizzata applicando agli avvolgimenti una corrente con andamento sinusoidale, in funzione dell'informazione sulla posizione angolare del rotore. Questo metodo permette di ottenere bassi ripple di coppia minimizzando il rumore generato dal motore, ma necessita della presenza di un sensore aggiuntivo per la misurazione accurata della rotazione del motore.

Essendo disponibile un encoder incrementale in aggiunta ai sensori di Hall, la commutazione nei motoriduttori del braccio robotico di Morpheus viene realizzata in modalità sinusoidale. Per i motori brushless, un giro "elettrico" dell'albero motore corrisponde ad un giro "meccanico" rapportato al numero di coppie polari del motore. All'interno del primo giro elettrico, quando ancora non è disponibile l'informazione accurata della posizione del rotore da parte dell'encoder, si utilizzano i sensori di Hall per effettuare la commutazione a blocchi, la quale viene poi modificata con quella sinusoidale non appena l'informazione è resa disponibile.

3.4.2 Controllo dei motoriduttori

La sezione successiva all'interno del processo di Startup riguarda la definizione della struttura dei loop di regolazione dei motoriduttori. Per il controllo di questi ultimi, infatti, è disponibile un'ampia varietà di modalità operative, che si basano sul controllo in posizione, in velocità e in corrente. In generale, l'architettura dei regolatori delle schede EPOS4 vede integrati al suo interno tre loop differenti: un regolatore di corrente, utilizzato in qualsiasi modalità operativa, un regolatore di posizione ed uno di velocità, utilizzati rispettivamente nelle modalità operative basate su posizione e velocità. A seconda della modalità scelta, una richiesta di posizione o velocità target viene fornita in ingresso al regolatore associato. L'output del regolatore entra nel loop di controllo in corrente, il quale pilota il sistema di potenza che va a comandare il motore. Se la richiesta in ingresso è in corrente, il primo step viene bypassato. Il controllo dei motoriduttori del manipolatore viene realizzato in modalità Profile Position Mode (PPM), la quale verrà descritta nel dettaglio in seguito, che si basa sulla regolazione della posizione. Dunque, ci si concentra sulla descrizione dei loop di regolazione di posizione e di corrente, che sono quelli effettivamente utilizzati in questa applicazione. In Figura 3.23 viene illustrata l'architettura di controllo generale delle schede EPOS4.

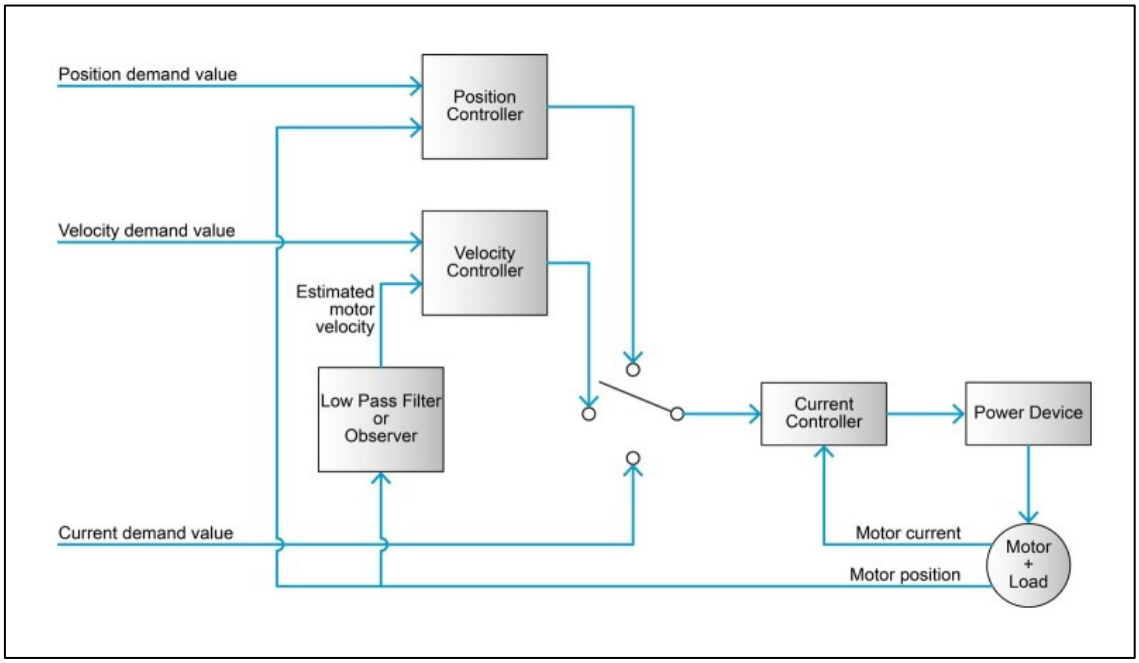

Figura 3.23 – Schema di controllo generale dei motoriduttori.

Il loop principale delle schede EPOS4 è quello di regolazione della corrente, il quale permette di controllare la coppia erogata dal motore. In Figura 3.24 viene mostrato lo schema a blocchi corrispondente: un controllore proporzionale-integrale PI riceve in ingresso la richiesta di corrente e fornisce in uscita la tensione di comando, che viene utilizzata per pilotare il sistema di potenza del motore. Al suo interno è implementato un algoritmo anti-windup che permette di prevenire la degradazione delle performance del controllore, nel caso in cui l'ingresso rimanga per lungo tempo attorno al suo valore limite, intervenendo sulla parte integrale e mantenendola al di sotto del suo valore massimo. La funzione di trasferimento del regolatore di corrente è descritta dall'equazione 3.20. K_{P_c} e K_{I_c} sono rispettivamente il guadagno proporzionale ed integrale del regolatore.

$$T_{current}(s) = K_{P_c} + \frac{K_{I_c}}{s} \qquad (3.20)$$

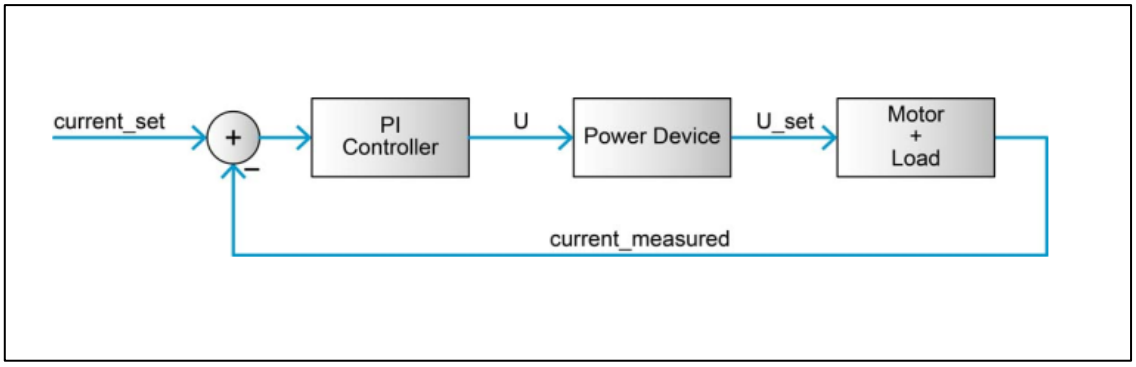

Figura 3.24 – Loop di regolazione di corrente delle schede di controllo EPOS4.

Il loop di regolazione della corrente è contenuto all'interno del loop di regolazione della posizione. Il controllore di posizione è del tipo proporzionale-integrale-derivativo (PID), supportato da una parte in feedforward in velocità ed accelerazione: la prima fornisce una corrente addizionale che viene utilizzata come compensazione di carichi che dipendono dalla velocità, come ad esempio l'attrito, mentre la seconda fornisce una corrente addizionale in caso siano presenti elevate accelerazioni o carichi con inerzie elevate. Per evitare picchi in uscita dalla componente derivativa, è presente un filtro passa-basso a valle della stessa, il quale previene l'influenza negativa del rumore ad alta frequenza sulla misura della posizione del rotore.

In Figura 3.25 viene mostrato lo schema a blocchi corrispondente. La funzione di trasferimento del regolatore di posizione è descritta dall'equazione 3.21. K_{P_P}, K_{I_P} e K_{D_P} sono rispettivamente il guadagno proporzionale, integrale e derivativo del regolatore. FF_w e FF_a sono rispettivamente i guadagni in feedforward di velocità ed accelerazione.

$$T_{position}(s) = K_{P_P} + \frac{K_{I_P}}{s} + \frac{sK_{D_P}}{1 + s\frac{K_{D_P}}{10K_{P_P}}} \qquad (3.21)$$

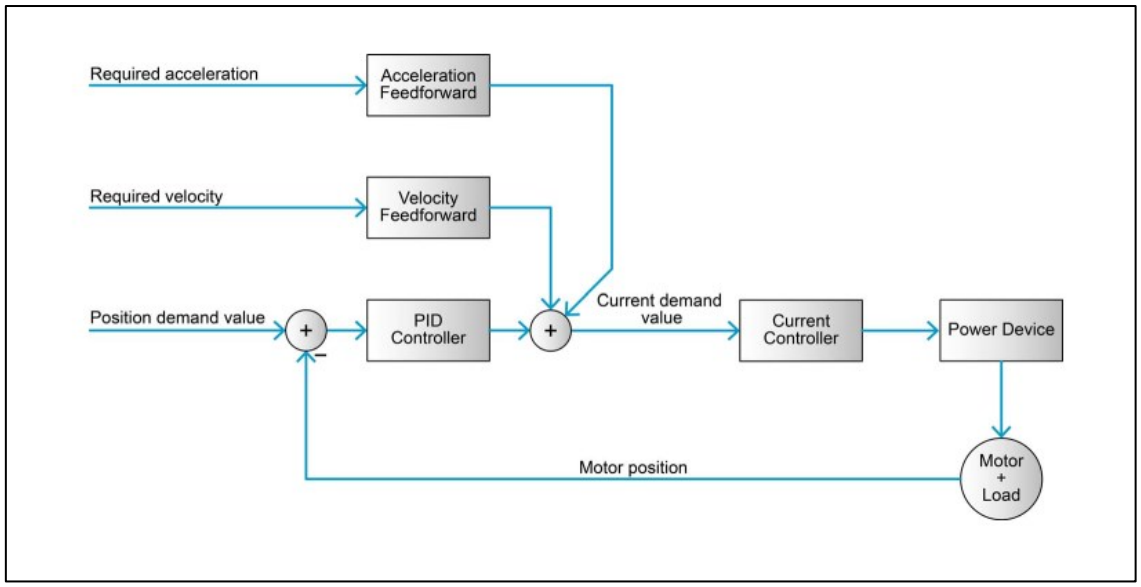

Figura 3.25 – Loop di regolazione di posizione delle schede di controllo EPOS4.

Una volta definita la struttura dei loop di regolazione dei motoriduttori, si procede con il processo di tuning dei controllori. Esso viene effettuato in modo automatico grazie al tool presente all'interno di EPOS Studio. Il tuning dei loop di corrente e di posizione viene realizzato secondo un processo ben definito: durante la prima fase viene applicato un ingresso a gradino sul sistema, la cui ampiezza può essere modificata dall'utente, per poi andarne a monitorare la risposta. In secondo luogo, vengono misurati i parametri identificativi del loop di regolazione: è possibile modificare manualmente i parametri calcolati, in modo tale da andare a soddisfare determinati requisiti. Inoltre, è possibile modificare la rigidezza del controllore. In ultimo, si vanno a monitorare le performance del sistema che utilizza i parametri calcolati, tramite un segnale di verifica. Durante il processo di autotuning, è importante che sull'albero del motoriduttore sia presente il carico che esso dovrà movimentare durante il funzionamento. Nel caso di un manipolatore, il carico presente sul motore è rappresentato dall'inerzia degli elementi posti a valle dello stesso. Ovviamente, essa può variare a seconda della configurazione assunta dal braccio robotico. Dunque, il tuning dei regolatori è stato effettuato ponendosi nella configurazione più gravosa, ossia con il sistema completamente esteso.

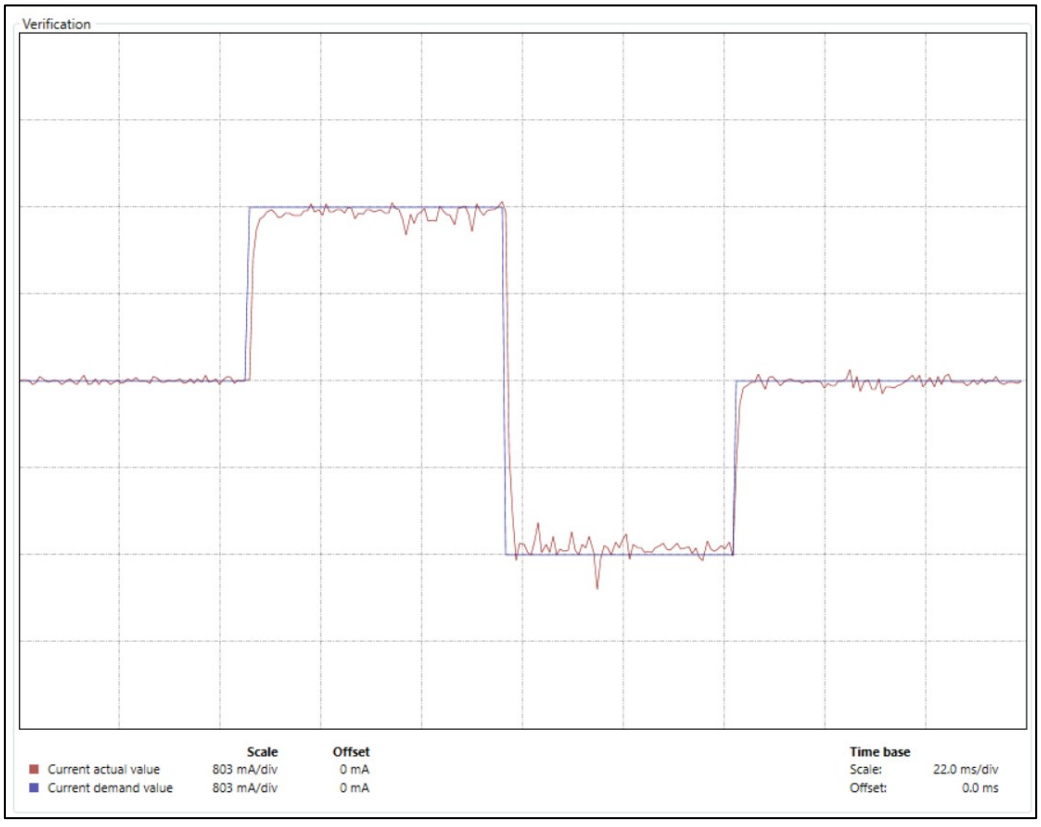

Figura 3.26 – Risposta del sistema ad una richiesta in corrente a gradino.

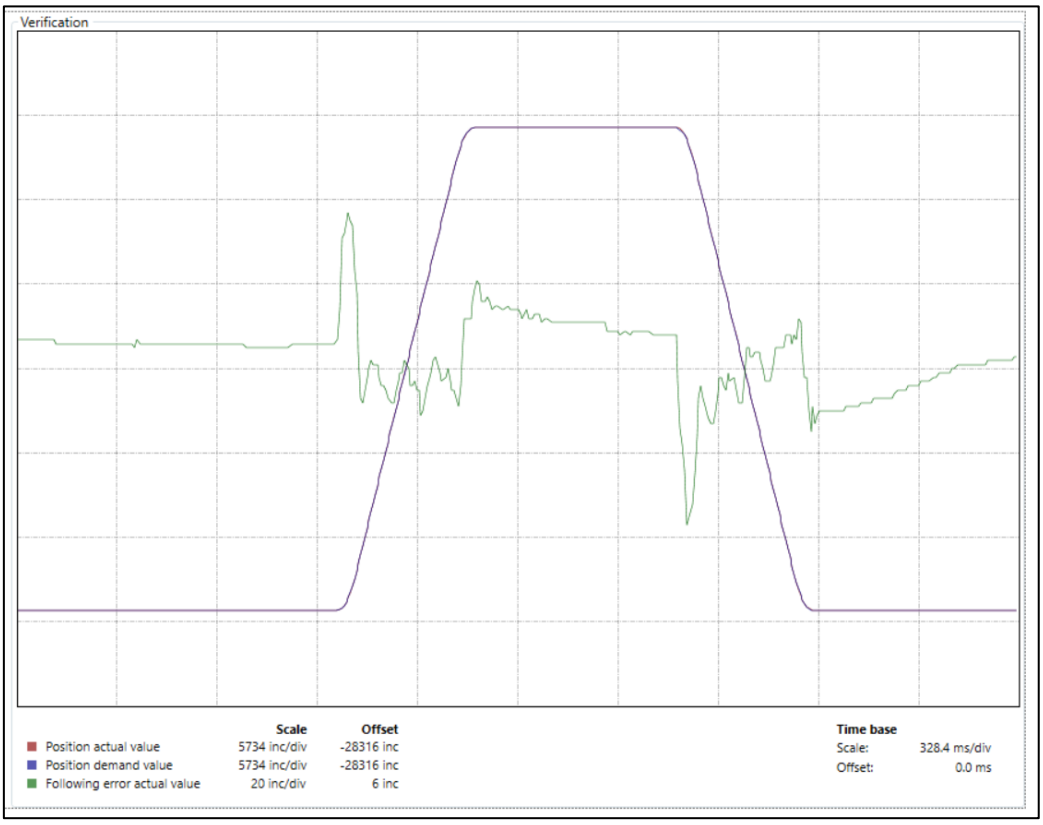

Figura 3.27 – Risposta del sistema ad una richiesta in posizione ed errore di posizionamento.

I motoriduttori lavorano in modalità Profile Position Mode (PPM). In questa modalità operativa, al sistema viene fornita in ingresso la posizione target che il motore deve raggiungere. Essa viene utilizzata da un blocco che si occupa della generazione della traiettoria: nello specifico, viene generato un profilo trapezoidale per la velocità angolare del motore. Ad esso corrisponde un profilo di posizione di secondo grado nel tratto iniziale e finale dove la velocità varia linearmente, e lineare nel tratto centrale a velocità costante. L'accelerazione avrà un andamento a gradino e sarà nulla nella fase di velocità costante. In aggiunta alla posizione target, al blocco di path planning deve essere fornito in ingresso il valore di velocità da raggiungere a regime, i valori di accelerazione e decelerazione per realizzare il profilo di velocità desiderato ed il valore di decelerazione "quick-stop", utilizzata in caso di blocco di emergenza del motore. Questa modalità operativa risulta conveniente in quanto la generazione di traiettoria viene eseguita in modo automatico dalle schede di controllo dei motori, senza la necessità di realizzarla

mediante l'uso software esterni, riducendo i tempi di commissionamento dei motori. D'altra parte, il profilo di velocità trapezoidale è l'unico realizzabile e, data la presenza di un profilo di accelerazione a gradino, possono riscontrarsi problematiche dovute alle discontinuità che causano valori elevati del jerk. Va tenuto in considerazione che questa è solamente una prima modalità di funzionamento del braccio robotico, il cui controllo potrà essere migliorato con sviluppi futuri.

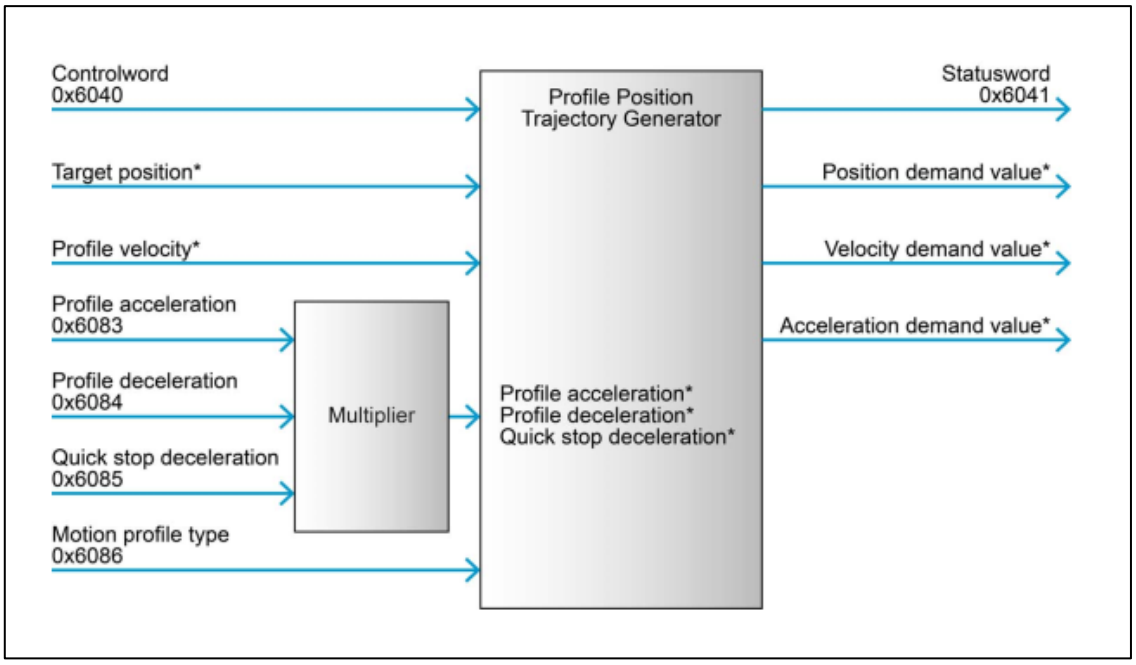

Figura 3.28 – Blocco di generazione di traiettoria per la modalità PPM.

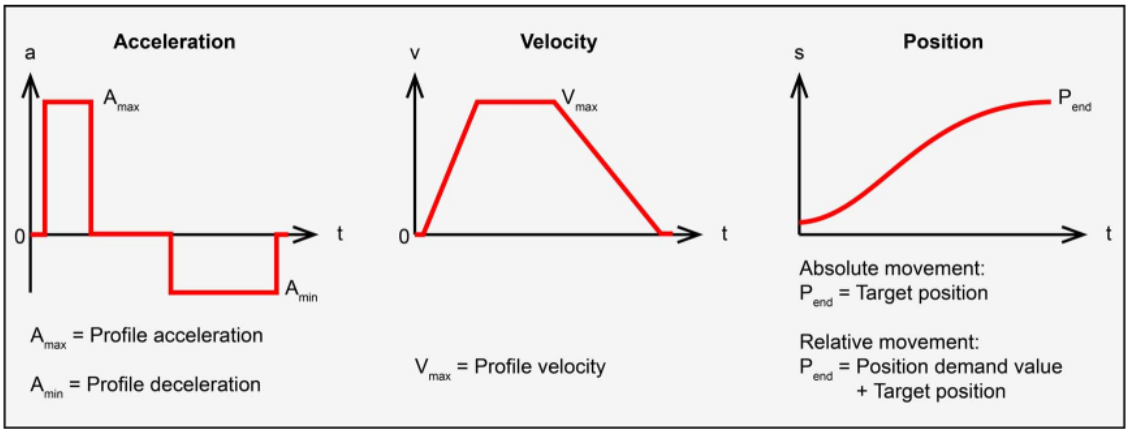

Figura 3.29 – Profilo di velocità trapezoidale per la modalità PPM.

4 Integrazione del manipolatore con ROS

4.1 Introduzione a ROS

Il sistema ROS, acronimo di Robot Operating System, è uno tra i più flessibili framework dedicati alla scrittura e all'implementazione di software per la robotica. Esso consiste in una collezione di strumenti, librerie e convenzioni che hanno lo scopo di semplificare la generazione di comportamenti di robot differenti, in maniera robusta e su una varietà di piattaforme. La nascita del sistema ROS è conseguente all'idea degli sviluppatori di incoraggiare un'implementazione di software con logica collaborativa, favorendo gli scambi di idee e di skill fra diverse realtà, ognuna delle quali può essere più o meno esperta nell'affrontare un certo tipo di compito. In questo modo, un certo gruppo può lavorare sulla base del lavoro svolto da un altro, e viceversa, collaborando alla crescita della community.

ROS è stato progettato per essere il più modulare possibile: all'interno dell'offerta disponibile, l'utente può scegliere quali parti e componenti ritiene utili, ed utilizzarle per la propria applicazione in parallelo a quelle sviluppate esternamente. Il framework è organizzato su tre livelli concettuali: il Filesystem level, il Computation Graph level ed il Community level.

In informatica, con filesystem si indica il meccanismo con il quale i file vengono organizzati e storati su dispositivi per l'archiviazione dei dati, in questo caso l'hard disk del computer. Il Filesystem level ricopre quindi le diverse risorse che si possono trovare sul disco. Le risorse utilizzate da ROS sono le seguenti:

- I pacchetti, ossia la principale unità di organizzazione di software. Essi possono contenere processi (nodi), librerie di dipendenze, dataset, file di configurazione e software di terze parti, ed hanno lo scopo di rendere queste funzionalità semplici da utilizzare;
- Le repositories, ossia una collezione di pacchetti che condividono uno stesso sistema di controllo di versione (VCS). Queste sono facilmente scaricabili, per esempio dal sito GitHub, e possono essere installate ed utilizzate localmente;

- Le tipologie di messaggi scambiati fra i nodi in ROS, la loro descrizione e struttura;
- Le tipologie di servizi utilizzati dai nodi in ROS, la loro descrizione e struttura. I servizi vengono utilizzati per effettuare operazioni di richiesta e risposta fra nodi, con conseguente scambio di messaggi.

Il Computation Graph level è una rete di processi peer-to-peer in ROS che elaborano dati contemporaneamente. La rete è costituita da un insieme di elementi che forniscono dati al Graph in diverse modalità, e che verranno ora descritti:

- I nodi, fondamentali per il funzionamento del sistema in quanto sono gli elementi che eseguono i calcoli. Essi sono scritti in linguaggio Python o C++. Tipicamente un sistema robotico può contenere una vasta gamma di nodi che svolgono compiti diversi e che interagiscono tra loro scambiandosi messaggi, ossia semplici strutture di dati, che possono essere di diverse tipologie. Questi messaggi sono indirizzati tramite un sistema di trasporto con logica publish / subscribe: un nodo fornisce in uscita un messaggio pubblicandolo sul relativo topic, il quale è utilizzato per identificare il contenuto del messaggio. Un secondo nodo, interessato ad una certa tipologia di dati, andrà a sottoscriversi al topic appropriato e sarà in grado di ricevere il messaggio desiderato. Le richieste di pubblicazione e sottoscrittura possono essere realizzate tramite i servizi;
- Il ROS Master, il quale gestisce ed organizza l'intero Computation Graph. Grazie ad esso i nodi sono in grado di interagire, scambiare messaggi e invocare servizi. Esso stocca i topic e i servizi disponibili ai nodi, i quali comunicano con esso riportando le informazioni di registrazione e pubblicazione, in modo da realizzare le corrette comunicazioni;
- Il Parameter Server, ossia un server atto allo stoccaggio di dati. Un generico dato che viene caricato sul Parameter Server può essere ritrovato ed utilizzato da un qualsiasi processo in corso;
- Le Bag, ossia un formato che permette la registrazione, il salvataggio e la riproduzione di messaggi di dati. Le bag rendono molto semplici i processi di raccolta dati, per esempio delle misure fornite in uscita dai sensori.

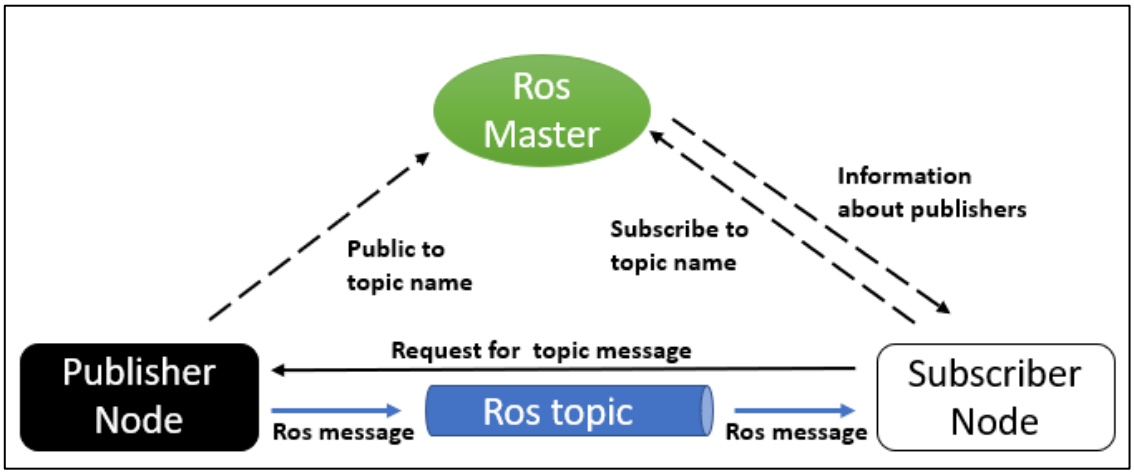

Figura 4.1 – Computation Graph del framework ROS.

In ultimo, il ROS Community Level permette lo scambio di risorse, di conoscenze e di software fra community. Queste risorse comprendono distribuzioni, repository, e la ROS Wiki, ossia un forum online che racchiude documentazione, informazioni e tutorial su tutto il sistema ROS.

4.1.1 Unified Robot Description Format

Il vantaggio principale nell'utilizzo di ROS per la gestione ed il controllo di un sistema robotico è il fatto che, assieme ad esso, vengono fornite molteplici librerie e strumenti specifici, che permettono di agevolare e velocizzare il controllo di un robot.

In primo luogo, una grande semplificazione è data dalla presenza di un set di messaggi standardizzati che coprono la maggior parte dei casi di utilizzo in robotica: esistono messaggi per la descrizione della posa di un elemento del sistema o delle trasformazioni fra elementi diversi, messaggi per l'utilizzo di sensori quali telecamere, piattaforme inerziali, LIDAR e messaggi per la gestione di dati di navigazione come odometria, percorsi e mappe.

In secondo luogo, ROS fornisce un formato per la descrizione del modello cinematico e dinamico del robot in una forma leggibile dal computer. Esso è denominato Unified Robot Description Format (URDF) e consiste in un documento XML all'interno del quale vengono descritte le proprietà fisiche del robot e la renderizzazione di ogni sua parte. Esso è organizzato a livelli, ognuno dei quali descritto da un tag che contiene le informazioni.

L'elemento a livello più alto è il tag <*robot*> all'interno del quale sono incapsulati tutti gli altri. Al livello inferiore si trovano i due elementi principali del modello cinematico, ossia i <*link*> ed i <*joint*>. I primi descrivono i link del robot come corpi rigidi, e ne denotano la massa, le componenti della matrice di inerzia, le caratteristiche di visualizzazione e di collisione e i sistemi di riferimento ad esse associati. Si fa notare che il sistema di riferimento generale del link, ossia quello utilizzato nella cinematica del sistema robotico, viene fatto coincidere per convenzione con la terna di riferimento scelta per il giunto precedente al link stesso. I secondi, invece, descrivono la cinematica e la dinamica dei giunti. In particolare, è presente la tipologia del giunto, i nomi del parent link e del child link, gli effetti dinamici quali smorzamento e attrito, i limiti sulla posizione superiore e inferiore, sulla velocità e sullo sforzo massimo. Questi dipendono dalla tipologia del giunto stesso: per un giunto rotoidale si hanno limitazioni sulla rotazione, sulla velocità angolare e sulla coppia, espresse in rad, rad/s e Nm rispettivamente.

Per poter descrivere le relazioni che intercorrono fra i giunti del robot e gli attuatori che li movimentano sono presenti gli elementi <*transmission*>. Essi trasformano le variabili di sforzo o di flusso, come coppie e velocità angolari, in modo da mantenerne costante il prodotto, e dunque la potenza meccanica. All'interno del blocco è definita la tipologia di trasmissione, l'attuatore a cui il giunto è collegato, il rapporto di trasmissione, e l'interfaccia hardware, quest'ultima necessaria per definire la tipologia di controllo del giunto.

Oltre ai principali elementi descritti, all'interno di un modello URDF possono essere presenti anche tag <*gazebo*>, che contengono informazioni necessarie durante l'esecuzione di simulazioni in ambiente Gazebo, e tag <*sensor*> i quali descrivono le proprietà di sensori di visione presenti sul robot, quali telecamere e laser scanner.

In Figura 4.2 è schematizzato l'insieme del giunto e dei link, con i rispettivi sistemi ed assi di riferimento, mentre in Figura 4.3 vengono schematizzati i componenti ed i relativi sistemi di riferimento del generico link di un robot. Il Codice 4.1 contiene un esempio di descrizione di un robot in formato URDF.

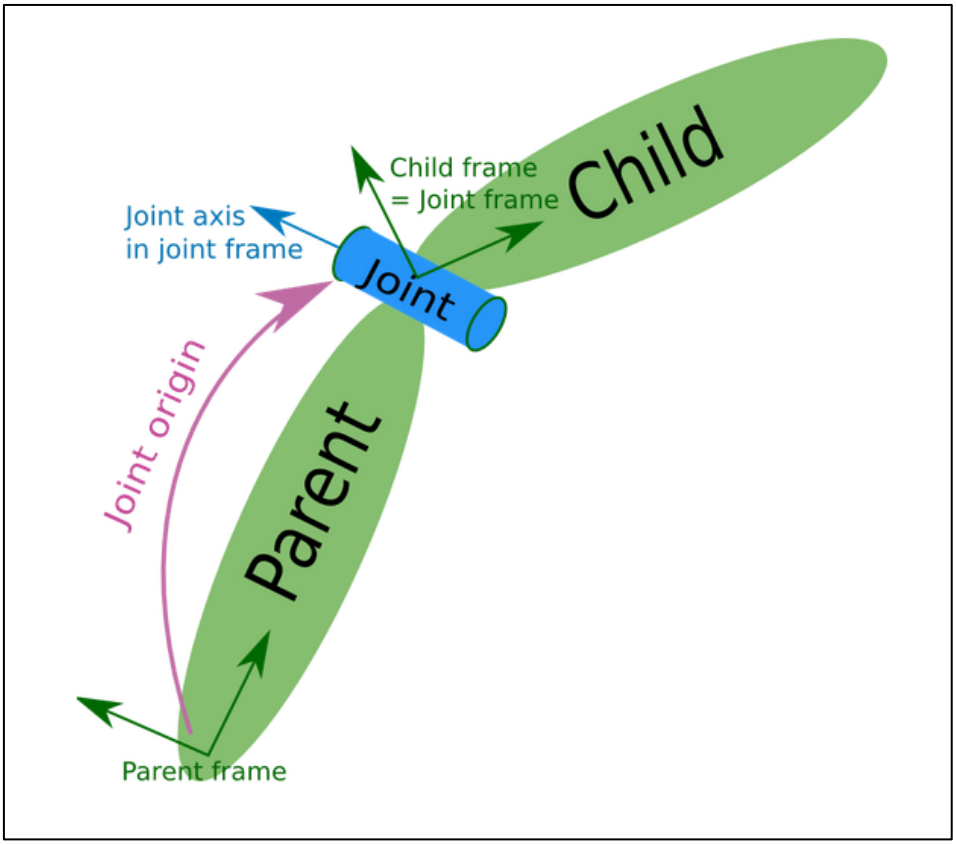

Figura 4.2 – Sistemi di riferimento di giunto e link nel formato URDF.

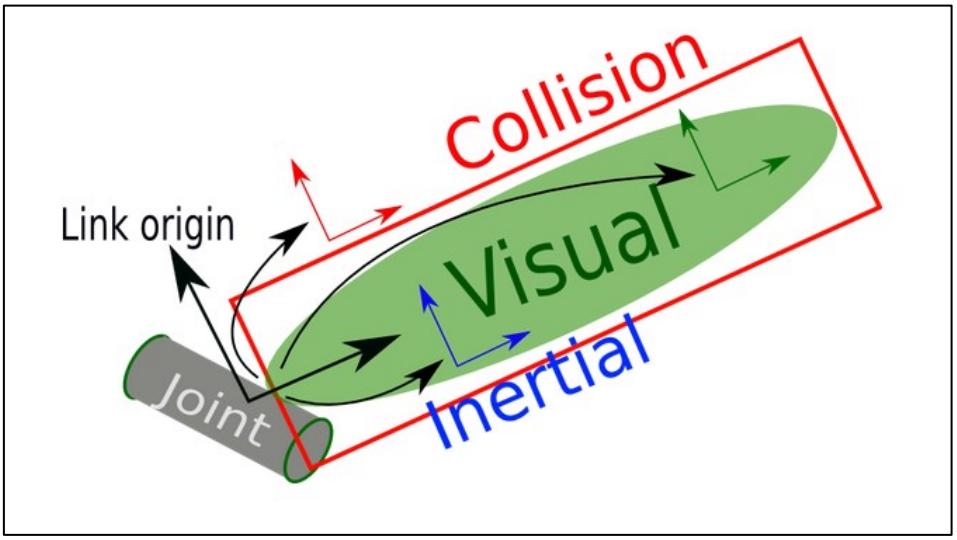

Figura 4.3 – Componenti di un link nel formato URDF.

```
<robot name="my_robot">

 <link name="my_link1">
   <inertial>
     <origin xyz="0 0 0.5" rpy="0 0 0"/>
     <mass value="1"/>
     <inertia ixx="100"  ixy="0"  ixz="0" iyy="100" iyz="0" izz="100" />
   </inertial>

   <visual>
     <origin xyz="0 0 0" rpy="0 0 0" />
     <geometry>
       <box size="1 1 1" />
     </geometry>
     <material name="Cyan">
       <color rgba="0 1.0 1.0 1.0"/>
     </material>
   </visual>

   <collision>
     <origin xyz="0 0 0" rpy="0 0 0"/>
     <geometry>
       <cylinder radius="1" length="0.5"/>
     </geometry>
   </collision>
 </link>

 <joint name="my_joint" type="revolute">
   <origin xyz="0 0 0" rpy="0 0 0" />
   <parent link="my_link1" />
   <child  link="my_link2" />
   <axis   xyz="0 0 1" />
   <limit
     lower="-3.4907"  upper="3.4907"
     effort="7.5"
     velocity="7" />
   <dynamics damping="1" friction="0.1" />
 </joint>

 <link name="my_link2">
   ...
 </link>

 <transmission name="my_joint_transmission">
   <type>transmission_interface/SimpleTransmission</type>
   <joint name="my_joint">

<hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface>
   </joint>
   <actuator name="my_joint_motor">
      <mechanicalReduction>100</mechanicalReduction>
   </actuator>
 </transmission>

</robot>
```

Codice 4.1 – Esempio di descrizione di un robot in formato URDF.

4.1.2 La libreria tf

Per un sistema robotico, la conoscenza della posizione e dell'assetto relativi, e quindi delle matrici di trasformazione, dei diversi link del robot è fondamentale. All'interno di ROS, il problema viene affrontato tramite l'utilizzo della libreria *tf*. La libreria *tf* è stata progettata con lo scopo di standardizzare il processo di determinazione e tracking nel tempo delle matrici di trasformazione fra gli elementi che compongono il sistema[7]. Essa può essere suddivisa in due moduli principali: un Broadcaster ed un Listener. Il primo si occupa di diffondere le informazioni riguardanti le trasformazioni fra sistemi di coordinate all'interno sistema, mentre il secondo riceve le informazioni e le mantiene in memoria, in modo da poter rispondere a successive richieste da parte di un qualsiasi processo. Nello specifico, il Broadcaster diffonde i messaggi contenenti le trasformazioni ogni qual volta si ha un aggiornamento della configurazione del sistema, o comunque con una frequenza minima fissata. Il Listener raccoglie le informazioni e, quando richiesto, può fornire in uscita la trasformazione desiderata. Se nell'istante specifico, nessuna informazione è disponibile, esso è in grado di interpolare fra i due valori più vicini. Ovviamente, l'approssimazione ottenuta tramite l'interpolazione sarà tanto più accurata tanto la più la frequenza di pubblicazione è elevata. La capacità di interpolare correttamente è estremamente importante in questa applicazione, in quanto permette alle diverse fonti di informazioni di pubblicare con frequenze diverse e in modo asincrono, peculiarità tipica dei sistemi robotici dove i dati sono spesso ricevuti da diversi componenti dell'hardware, come i sensori o i feedback degli attuatori, in modo indipendente.

Le trasformazioni fra sistemi di riferimento vengono descritte attraverso un vettore posizione ed un vettore di quaternioni normalizzato e sono organizzate in una struttura "ad albero". All'interno della struttura le terne di riferimento sono schematizzate come dei nodi e le trasformazioni relative come delle frecce che collegano due nodi consecutivi. Ciò permette al sistema di ridurre i tempi di ricerca della particolare trasformazione richiesta.

Per il calcolo di una trasformazione fra due sistemi generici, il Listener va a cercare all'interno dell'albero il collegamento fra la terna sorgente e quella target, passando per i nodi intermedi. Una volta che questo è stato identificato, si può determinare la trasformazione netta andando a moltiplicare in successione le trasformazioni consecutive.

4.1.3 La libreria ROS Control

La libreria ROS Control contiene un insieme di pacchetti dedicati al controllo di sistemi robotici generici. Il design modulare, la capacità di gestione dei controllori in tempo reale e la possibilità di implementarla con ogni tipologia di robot la rendono estremamente performante[8]. Inoltre, ROS Control è stata pensata per interfacciarsi con sistemi di gestione dei robot di terze parti, come ad esempio MoveIt e il ROS navigation stack. In questo modo, per un sistema che deve navigare in modo autonomo e dotato di un manipolatore di cui deve essere pianificata la traiettoria, come è il caso del rover Morpheus, non è necessaria la scrittura di alcun codice aggiuntivo se non per il settaggio dei file di configurazione dei controllori utilizzati.

Il componente principale del framework è l'Hardware Abstraction Level, il quale funziona da ponte fra i controllori e il robot controllato, sia in ambito di simulazione che reale. Al suo interno viene definita la classe *hardware_interface::RobotHW*, utilizzata per l'interfaccia con il robot, e dunque responsabile della lettura e scrittura delle informazioni da e verso il sistema da controllore. Inoltre, essa tiene in considerazione la presenza dei limiti su posizionamento, velocità e coppia ai giunti e delle trasmissioni meccaniche come i riduttori. ROS Control rende permette la composizione di differenti interfacce hardware già implementate, caratteristica molto utile durante la realizzazione di sistemi robotici con componenti da fornitori diversi, i cui driver sono disponibili per l'utilizzo.

La seconda parte del framework è il controller manager, responsabile della gestione dei controllori veri e propri. A seconda dell'interfaccia hardware da controllare, sono disponibili diverse tipologie di questi ultimi: controllori di posizione, di velocità, di coppia, di traiettoria e per la lettura dello stato dei giunti, che possono essere caricati, lanciati o arrestati in qualsiasi momento tramite l'utilizzo di specifici servizi ROS. Il controller manager gestisce l'intero ciclo di vita dei controllori, dall'invocazione, al funzionamento e allo spegnimento. In sostanza, esso fornisce un'interfaccia fra i controllori e ROS, ricevendo in ingresso i messaggi che contengono le richieste da eseguire e fornendole ai controllori dedicati.

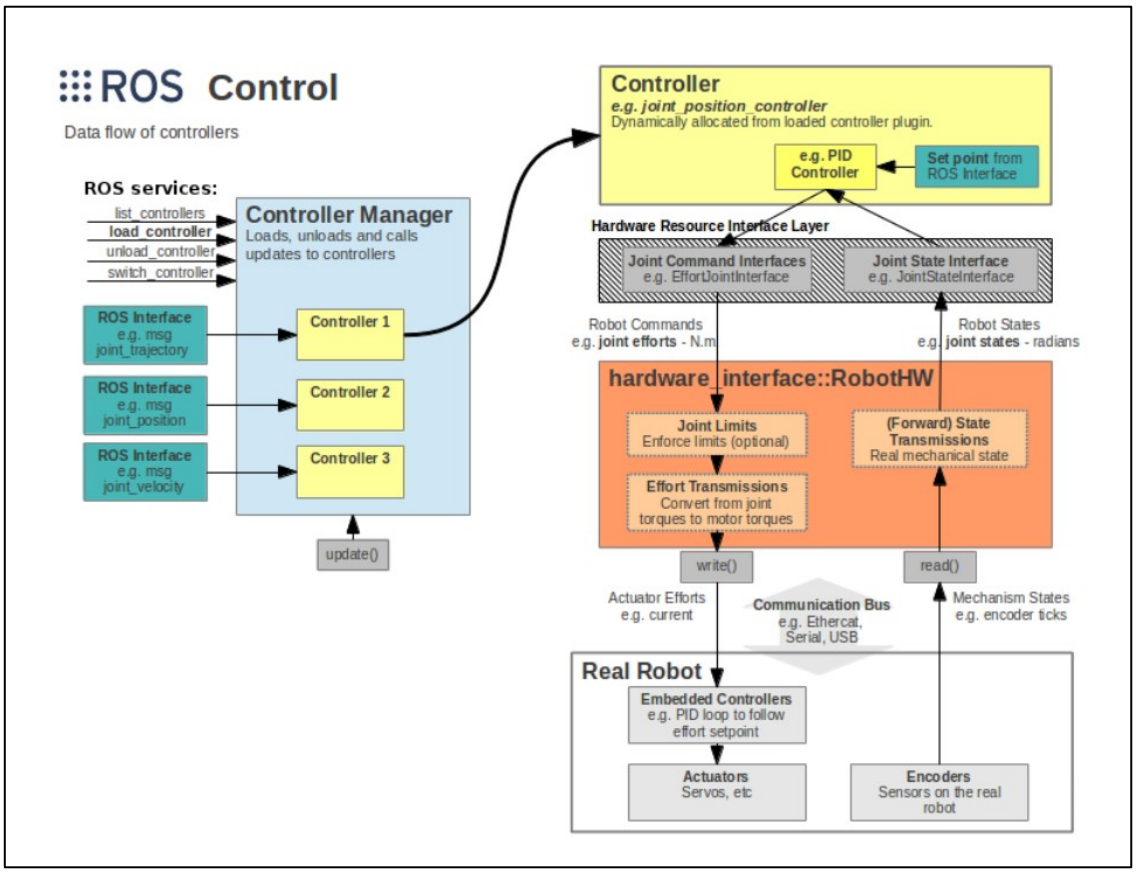

Figura 4.4 – Schema di funzionamento di ROS Control (Fonte: http://gazebosim.org/tutorials/?tut=ros_control).

4.1.4 La libreria MoveIt!

Principalmente, la libreria MoveIt si occupa della pianificazione della traiettoria del sistema robotico, sia esso una piattaforma mobile o un manipolatore. MoveIt viene lanciato in parallelo al software di visualizzazione Rviz, in modo tale da permettere all'utente di interfacciarsi con il robot, scegliendo la configurazione dei giunti o posizionando l'end effector all'interno dello spazio di lavoro, con la possibilità di visualizzazione dei sistemi di riferimento, di ostacoli e di oggetti presenti nello spazio.

Il cuore della libreria è il nodo *move_group*, il quale integra i diversi componenti e le varie funzionalità: esso riceve in ingresso i comandi forniti dall'utente direttamente da Rviz, oppure tramite dei nodi esterni che utilizzano le interfacce dedicate scritte in C++ o in Python. Inoltre, esso riceve dal parameter server la descrizione del robot in formato URDF e SRDF e la sua configurazione per Moveit, interpretandone la cinematica, la dinamica, i limiti dei giunti e le collisioni, oltre ad interfacciarsi con i controllori del robot, per la scrittura dei comandi agli attuatori e la lettura dei feedback dai sensori.

4.2 Il Morpheus workspace in ambiente ROS

Come ambiente di lavoro in ROS per lo sviluppo, la gestione ed il controllo del braccio robotico da integrare sul rover Morpheus, è stato utilizzato un catkin workspace. Esso è essenzialmente una directory all'interno della quale è possibile modificare, compilare ed installare i pacchetti ROS. Il workspace è diviso in quattro sottocartelle: lo spazio Source, il quale contiene i pacchetti che si desiderano compilare ed utilizzare, lo spazio Build, dove il processo di compilazione dei pacchetti viene invocato, lo spazio Development, che contiene i nodi che sono stati compilati, prima che siano installati nello spazio Install.

4.2.1 I pacchetti installati

All'interno dello spazio Source del catkin workspace del manipolatore sono presenti una serie di pacchetti ROS, dedicati alla gestione di quest'ultimo.

Il pacchetto *braccio_morpheus* è stato generato a partire dall'assieme CAD del sistema, una volta definiti i sistemi di riferimento dei link e gli assi di rotazione dei giunti, tramite il tool Solidworks to URDF Exporter. Esso permette di esportare parti o assiemi Solidworks in formato URDF, generando il pacchetto ROS contenente le mesh e le texture delle varie parti del robot, oltre che alla descrizione in URDF.

Una volta che quest'ultimo è disponibile, tramite il MoveIt Setup Assistant si è generato il pacchetto *braccio_morpheus_moveit_config*. Il Setup Assistant è un'interfaccia grafica che permette di configurare un generico robot per l'utilizzo all'interno di MoveIt. Il pacchetto contiene al suo interno il file SRDF (Semantic Robot Description Format) del manipolatore, ossia una descrizione complementare all'URDF, nella quale sono definiti i gruppi di giunti da comandare in modo sincrono, la terna solidale all'end effector, le configurazioni di collisione fra elementi del robot e le posizioni di default se presenti. Inoltre, sono presenti una serie di file YAML di configurazione del robot, come la descrizione del solutore della cinematica inversa, dei limiti di velocità e accelerazione ai giunti e dei controllori utilizzati.

La cinematica inversa viene risolta tramite il pacchetto *braccio_morpheus_ik_fast*. IKFast è un solutore cinematico reso disponibile all'interno della libreria OpenRAVE. Esso permette la risoluzione delle equazioni della cinematica inversa in modo analitico e non numerico, come spesso prevedono i solutori disponibili, minimizzando i tempi di elaborazione delle soluzioni.

Inoltre, IKFast può essere utilizzato per robot che possiedono configurazioni cinematiche arbitrarie, anche complesse e con assi dei giunti non intersecanti. Esso fornisce in uscita un file C++ per la soluzione della cinematica inversa, indipendente dalla libreria e che può essere facilmente integrato in qualsiasi progetto.
Il pacchetto è stato generato a partire dalla descrizione cinematica del robot, contenuta all'interno del file URDF, che viene convertito in Collada (COLLAborative Design Activity), un formato di interscambio fra applicazioni 3D. Esso viene fornito in ingresso ad uno script disponibile nella libreria OpenRave[9] dedicato alla generazione del solutore cinematico. Oltre alla descrizione del robot, lo script richiede in input:

- La tipologia di cinematica inversa che si vuole realizzare, a seconda dei gradi di libertà dell'end effector che si desiderano controllare. Per il manipolatore a 4 gradi di libertà del rover Morpheus è stata scelta la tipologia TranslationZAxisAngle4D, la quale permette di posizionare il gripper in una qualsiasi posizione 3D dello spazio di lavoro, specificando l'angolo che il sistema di riferimento associato deve formare con l'asse Z della terna fissa alla base del manipolatore, ossia l'angolo di pitch dell'end effector;
- I sistemi di riferimento solidali alla base e all'end effector del robot;
- Il planning group, ossia l'insieme dei giunti utilizzati per la movimentazione dell'end effector.

Lo script fornisce in uscita il pacchetto *braccio_morpheus_ik_fast*, che contiene al suo interno lo script *braccio_morpheus_ikfast_solver.cpp*. Esso è il processo responsabile per la soluzione delle equazioni della cinematica inversa del manipolatore e contiene al suo interno le funzioni *ComputeFk* e *Compute IK*, che si occupano di risolvere le equazioni della cinematica diretta ed inversa rispettivamente.
L'interfaccia hardware con i motoriduttori dei giunti e con i gripper è realizzata dai pacchetti *epos_hardware* e *robotiq_2f_gripper_control* rispettivamente. *Epos_hardware* è un pacchetto di driver costruito attorno alla libreria EPOS Command Library, fornita dalla Maxon Motor e utilizzata per il controllo delle schede EPOS4. Il file *epos_arm.launch* si occupa del lancio e della gestione dei driver dei motoriduttori.

Nello specifico, esso carica la descrizione URDF del manipolatore, il file di configurazione dei controllori, nel quale sono elencate tipologia e caratteristiche ed i file di configurazione dei motoriduttori. Questi contengono tutte le informazioni utili per identificare ed inizializzare i motoriduttori connessi alla workstation: il nominativo, la tipologia di connessione ed il numero di serie della scheda di controllo EPOS, le caratteristiche operative del motore e del sensore associato, la mappa che associa le modalità operative delle schede con i controllori dedicati, le condizioni al contorno della Profile Position Mode, quali la velocità a regime ed i valori dell'accelerazione e decelerazione.

In seguito, viene lanciato il nodo *epos_hardware_node*, al quale sono forniti come argomenti i nomi degli attuatori associati ai giunti, caricati in precedenza. Il nodo inizializza la classe *epos_hardware::EposHardware*, all'interno della quale saranno gestiti i driver e avvia il controller manager di ROS control, che si occupa di associare i controllori alla modalità operativa scelta per i motoriduttori. A questo punto viene avviato un ciclo *while* con frequenza 50 Hz, all'interno del quale vengono effettuate la lettura dello stato dei motori, e quindi dei giunti, e la scrittura delle informazioni per il controllo dell'hardware. In ultimo, viene lanciato il nodo *robot_state_publisher*, il quale si sottoscrive al topic */joint_states*, pubblicato dai driver dei motori e contenente lo stato dei giunti, in termini di posizione, velocità e coppia, per poi pubblicare le trasformazioni relative fra i diversi sistemi di riferimento dei link del manipolatore, sul topic */tf*. Gli attuatori dei giunti sono quindi inizializzati e pronti a ricevere i comandi per la movimentazione del braccio robotico.

Anche il gripper Robotiq 2F-85 viene connesso alla workstation tramite collegamento USB. Per l'integrazione con ROS viene utilizzato il pacchetto di driver *robotiq_2f_gripper_control*. Esso carica la descrizione URDF del gripper e le informazioni utili al riconoscimento del device, come la porta USB di connessione ed il baudrate, per poi lanciare il nodo python *Robotiq2FGripperRtuNode*, che si occupa dell'interfaccia con il gripper. Quest'ultimo viene comandato direttamente dal terminale tramite un secondo nodo, denominato *Robotiq2FGripperSimpleController*. È possibile selezionare la posizione di apertura del gripper che è compresa all'interno di un intervallo da 0 a 255, oltre che la velocità di apertura/chiusura e la forza esercitata dalle dita.

4.2.2 Planning pipeline nello spazio dei giunti

Una volta inizializzati i motoriduttori ed il gripper, la pipeline di posizionamento di quest'ultimo è realizzata dal nodo *pose_node* contenuto all'interno del pacchetto *pose*: esso è uno script in C++ realizzato utilizzando la *move_group_interface* di MoveIt, in modo tale da integrarlo con la libreria per poterla utilizzare per la pianificazione della traiettoria dei giunti. Si fa notare che, alla fase attuale del progetto, la pianificazione della traiettoria dei giunti viene eseguita internamente dalle schede di controllo assi, come descritto al paragrafo 3.4.2. Dunque, la libreria MoveIt viene utilizzata solamente come interfaccia grafica e per implementare il controllo delle collisioni fra gli elementi del braccio robotico, ma non per l'effettivo planning. Ciononostante, la presenza di un elemento di integrazione fra il controllo del robot e MoveIt sarà utile per i futuri sviluppi del progetto, poiché renderà possibile lo sviluppo di diverse tipologie di pianificazione, utili per la realizzazione di task più complessi del manipolatore, per l'obstalce avoidance e per l'utilizzo di nuovi controllori.

Il nodo *pose*,dunque, inizializza la planning interface con il nodo *move_group*, basandosi sul gruppo di giunti da comandare, in questo caso i numeri da 1 a 4 del manipolatore. Viene quindi effettuata la lettura dal parameter server della posa target che l'end effector deve raggiungere, in termini delle 3 coordinate x, y, e z e dell'angolo di pitch. In questa fase la posa viene fornita manualmente dall'utente in fase di lancio del processo ma, successivamente, essa verrà ricavata in modo automatico dalle informazioni ottenute dal sistema di visione del rover, che individuerà la posizione dei campioni da raccogliere. A questo punto, le coordinate e l'angolo forniti in ingresso devono essere convertiti in un formato che permetta la soluzione della cinematica inversa e della pianificazione della traiettoria: si inizializza un oggetto *PoseStamped*, che contiene al suo interno un header e una posa. L'header contiene l'informazione riguardante il sistema di riferimento in cui è definita la posa, in questo caso quello solidale alla base del robot, mentre la posa contiene 3 valori per la traslazione e 4 valori per l'orientazione, poiché quest'ultima è definita tramite un vettore di quaternioni normalizzato. L'oggetto *PoseStamped* viene dato in input alla funzione *setJointValueTarget*. Essa va a richiamare il solutore della cinematica inversa, il quale determina il vettore delle variabili di giunto che permette all'end effector di raggiungere la posa target desiderata.

Nell'applicazione corrente, il vettore delle variabili di giunto viene fornito in ingresso alle schede di controllo dei motoriduttori, che eseguono la pianificazione della traiettoria, altrimenti viene richiamata la funzione *plan* dell'interfaccia con il *move_group*, e la traiettoria dei giunti viene pianificata da MoveIt. Il workspace è stato progettato per poter realizzare pipeline appena descritta anche nell'ambiente simulato Gazebo. Questo risulta molto utile in fase di test, e per verificare la traiettoria seguita dall'end effector a seguito di un comando, prima di realizzarla effettivamente con l'hardware. In Figura 4.7 sono riportati i ROS graph contenenti tutti i nodi attivi durante il funzionamento del sistema, sia per il controllo del braccio robotico sia per le simulazioni. Lo script completo del nodo *pose_node* in C++ è allegato in appendice.

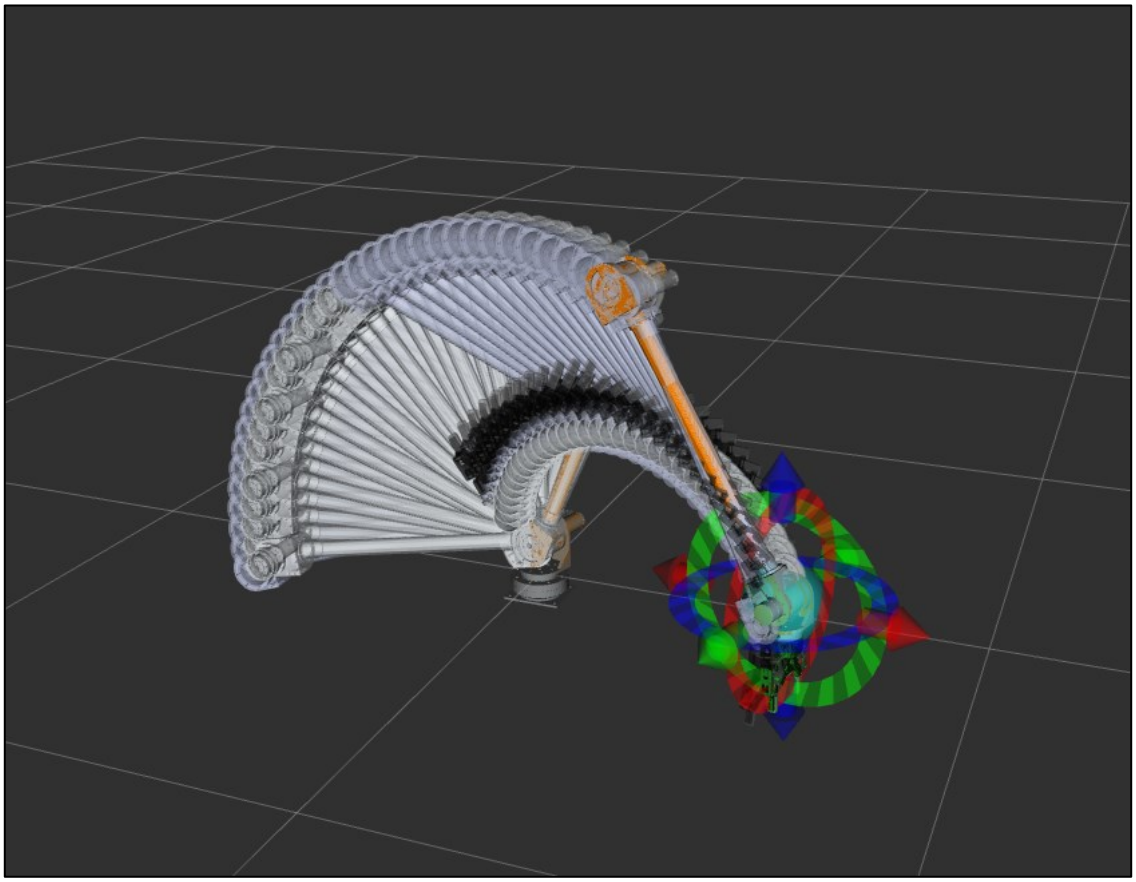

Figura 4.5 – Visualizzazione in Rviz della traiettoria pianificata con MoveIt.

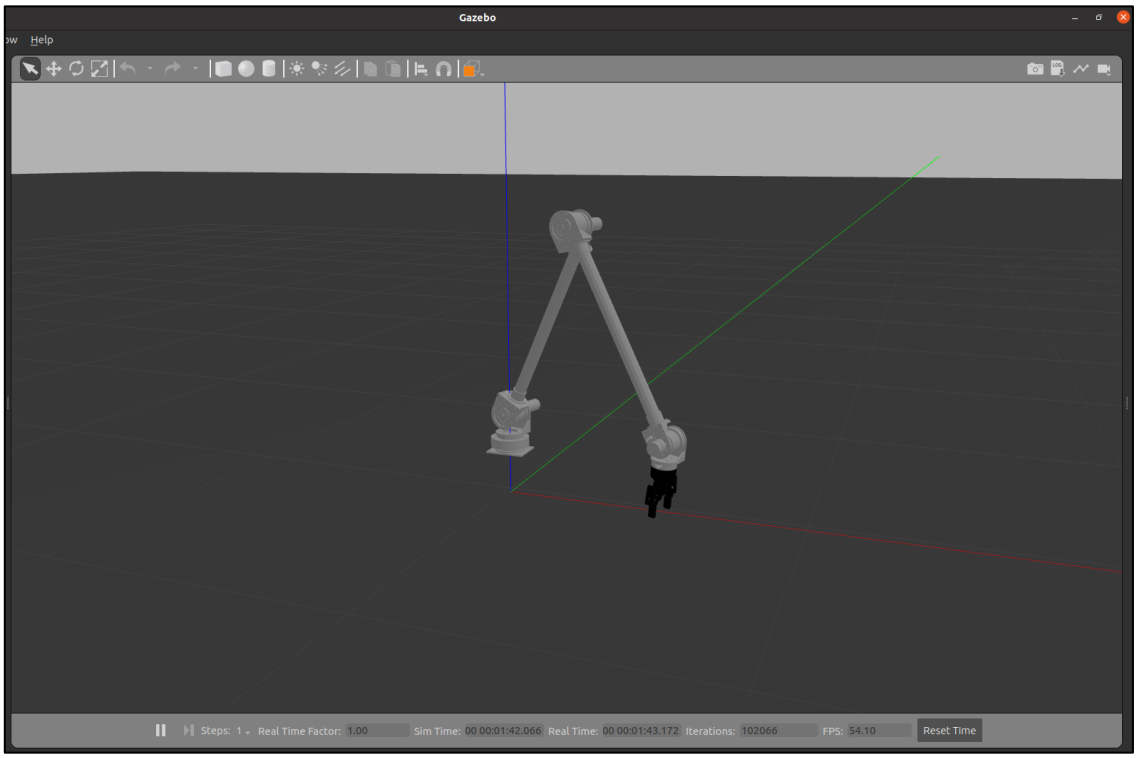

Figura 4.6 – Simulazione del comportamento del manipolatore in Gazebo.

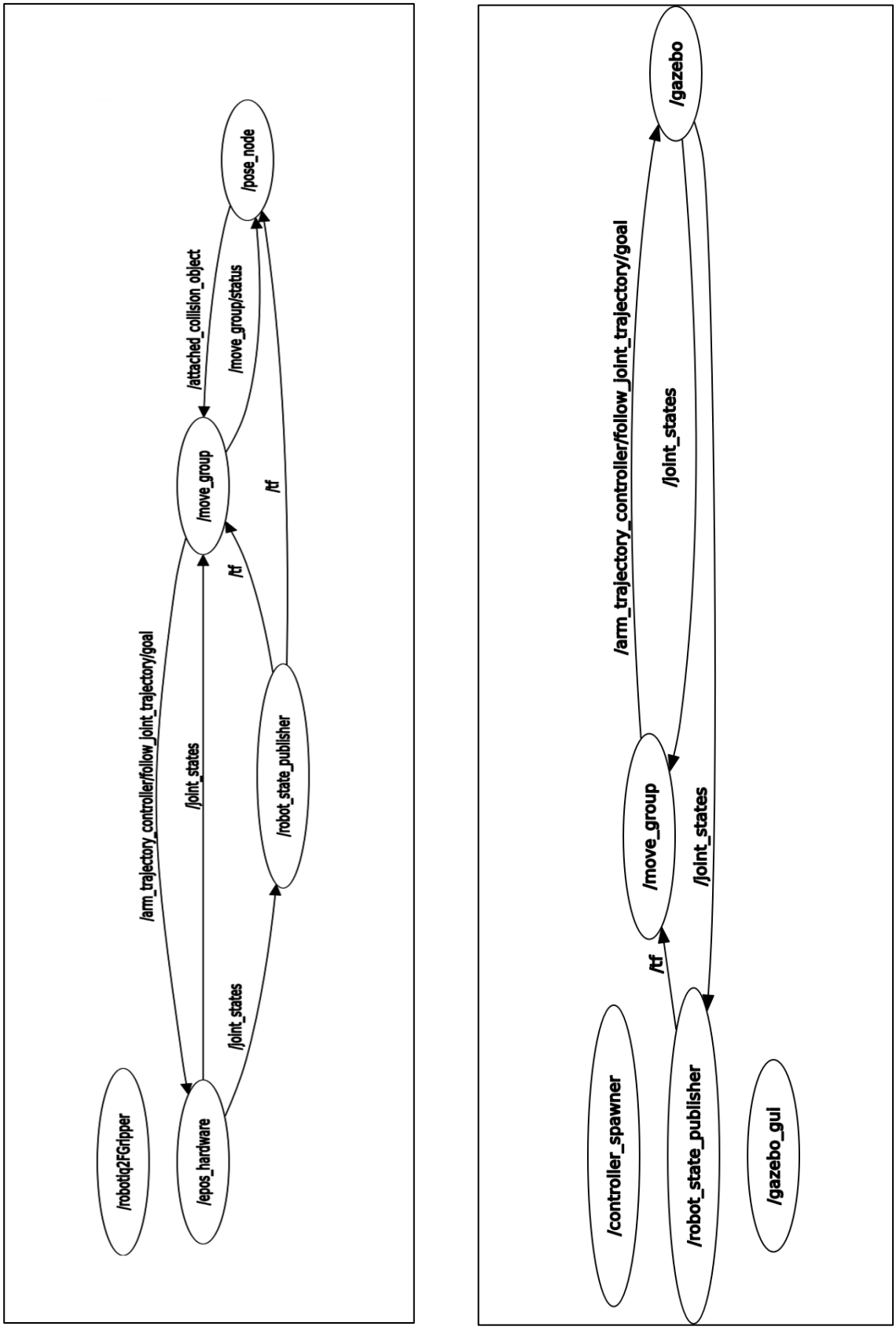

Figura 4.7 – ROS graph del Morpheus workspace: funzionamento con hardware reale (a sinistra) e simulato (a destra).

5 Test e risultati

5.1 Test di accuratezza e ripetibilità del sistema

In seguito all'integrazione del manipolatore con ROS, è stata svolta una campagna di test per la determinazione dell'accuratezza e della ripetibilità del sistema, in termini di posizionamento ed orientazione dell'end effector nello spazio di lavoro. L'accuratezza è definita come la capacità del robot di far raggiungere all'end effector la posa comandata, rispetto al sistema di riferimento fisso solidale alla base. La ripetibilità è invece definita come la capacità del robot di far raggiungere all'end effector posizioni memorizzate in precedenza. Entrambe le caratteristiche concorrono alla quantificazione della differenza presente fra la posa comandata e la posa effettivamente raggiunta. Le differenze presenti possono essere causate da diversi fattori, come le caratteristiche del sistema di controllo del braccio robotico, errori nelle trasformazioni fra sistemi di riferimento, differenze fra il modello cinematico ideale definito nel file URDF e quello reale, cause di natura meccanica come la presenza di giochi e attriti.

Al fine di analizzare allo stesso tempo l'accuratezza e la ripetibilità del sistema, viene fissata una serie di punti che l'end effector deve raggiungere in successione ed il percorso viene ripetuto per un certo numero di volte. Per la scelta dei punti ci si è basati sulla normativa ISO 9283:1998, la quale disciplina i criteri di classificazione delle performance dei manipolatori e i metodi di test. Viene quindi scelta una porzione parallelepipeda dello spazio di lavoro, con le facce parallele agli assi del sistema di riferimento fisso alla base del robot. La porzione dello spazio di lavoro scelta deve coincidere con la zona di maggiore operatività del sistema. Viene quindi definito un piano diagonale all'interno del volume del parallelepipedo, denominato piano di misura. Cinque punti di misura, P1, P2, P3, P4 e P5 vengono scelti sulle due diagonali del piano di misura: P1 corrisponde all'intersezione delle due diagonali mentre i restanti quattro possono essere scelti arbitrariamente purché sufficientemente distanti tra loro e da P1. Per valutare la ripetibilità, al robot vengono fatti raggiungere i 5 punti in successione per 10 volte, utilizzando il pacchetto *pose* descritto al paragrafo 4.2.2. I punti di misura sono illustrati in Figura 5.1, mentre le rispettive pose del gripper sono elencate in Tabella 5.1.

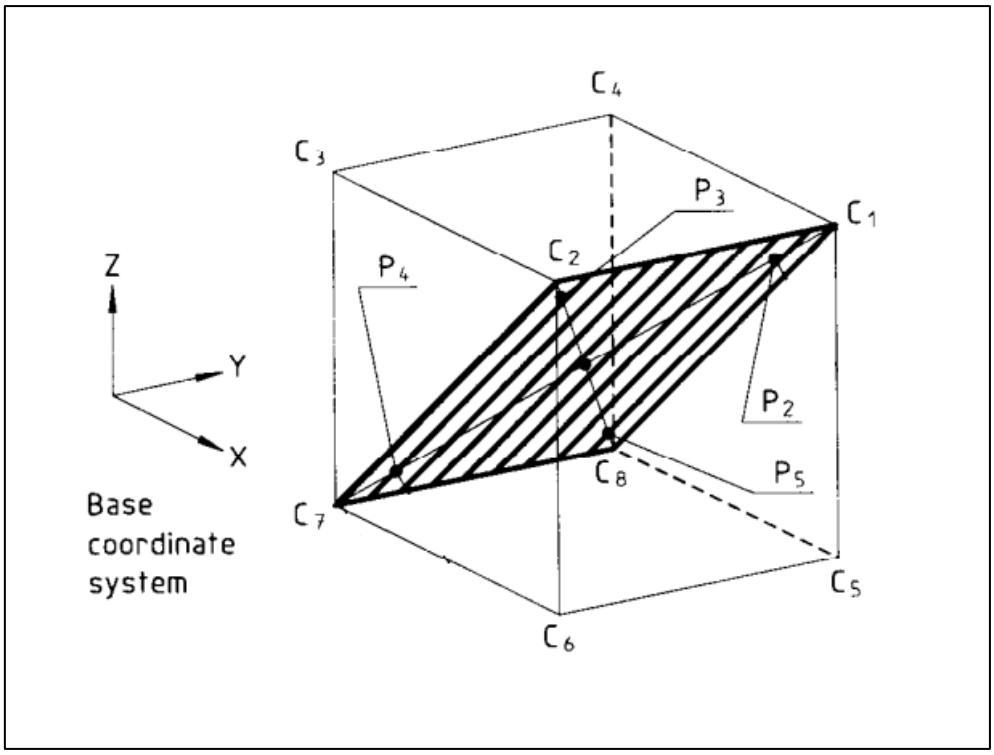

Figura 5.1 – Punti di misura utilizzati per i test, definiti secondo normativa ISO 9283:1998.

Punto	X [m]	Y [m]	Z [m]	Pitch [rad]
P1	0.3	- 0.076	0.4	0.5
P2	0.5	0.3	0.6	0.5
P3	0.5	- 0.3	0.6	0.5
P4	0.2	-0.3	0.2	1
P5	0.2	0.3	0.2	1

Tabella 5.1 – Pose comandate all'end effector per i vari punti di misura.

5.1.1 Setup sperimentale

Per effettuare la stima della posa dell'end effector del manipolatore sono state effettuate due serie di misure differenti, una realizzata tramite un sensore esterno (una telecamera) ed una realizzata tramite sensori interni (gli encoder dei motoriduttori).

Per la ricostruzione della posa dell'end effector a partire dalle misure effettuate tramite telecamera, si è utilizzato il pacchetto *Tagslam*[10]. Quest'ultimo è un pacchetto OpenSource basato su ROS, che permette di realizzare la Simultaneous Localization and Mapping (SLAM), in modo flessibile e robusto, utilizzando i marker fiduciari AprilTag. Gli AprilTag[11] sono marker sviluppati dall'APRIL Robotics Laboratory dell'Università del Michigan e sono utilizzati per un'ampia varietà di task, come la realtà aumentata, la robotica e la calibrazione delle telecamere. I tag possono essere scaricati e stampati tramite una normale stampante, applicati su un oggetto ed utilizzati per determinarne la posizione e l'orientazione, per mezzo di una telecamera, tramite un detection software presente all'interno della libreria dedicata. Gli AprilTag sono concettualmente simili ai codici QR, in quanto anch'essi appartengono alla famiglia dei codici a barre bidimensionali, ma sono pensati per codificare un numero minore di dati, in modo tale da facilitarne il riconoscimento. Durante i test eseguiti con il manipolatore, sono stati utilizzati tag della famiglia 36h11, di cui un esempio è visibile in Figura 5.2, di diverse dimensioni.

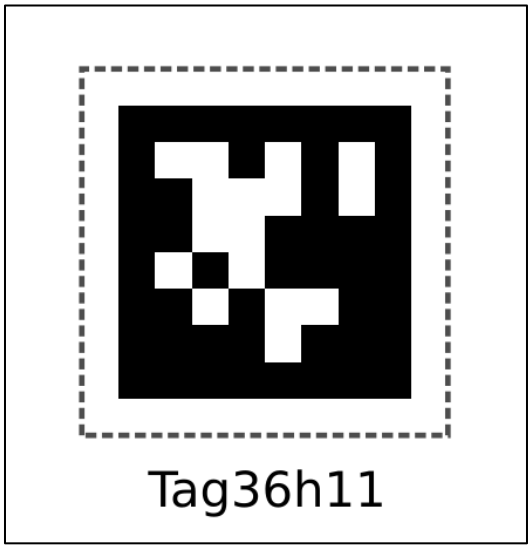

Figura 5.2 – AprilTag della famiglia 36h11.

In generale, le tecniche di SLAM permettono di costruire una mappa dell'ambiente circostante utilizzando un insieme di landmark: i sensori presenti nel sistema, come ad esempio le telecamere, effettuano una serie di misurazioni ed all'interno di esse l'algoritmo ricerca delle feature, individuando i landmark che saranno inseriti all'interno della mappa. Quando, nelle misure successive, viene riconosciuto lo stesso landmark, sarà possibile determinare la posa dell'osservatore tramite triangolazione. Le difficoltà nell'utilizzo di queste tecniche sono essenzialmente due: il riconoscimento dei landmark precedentemente noti, quando vengono osservati da punti di vista o sotto condizioni di illuminazione differenti ed il mantenimento in memoria della mappa generata. L'idea dietro al pacchetto *Tagslam* è quella di utilizzare gli AprilTag come landmark fiduciari noti e di eseguire la SLAM sulla base di essi. In questo modo i problemi di memoria e velocità di esecuzione vengono meno. Inoltre, *Tagslam* è in grado di effettuare il tracking di oggetti multipli, ai quali sono stati apposti uno o più tag. Per la campagna di test che si desidera svolgere, sarà sufficiente apporre una serie di tag sull'end effector per poterne determinare la posa durante il movimento nello spazio di lavoro.
Il software si basa su poche e semplici astrazioni, le quali permettono di simulare scenari complessi senza la necessità di scrivere alcun codice aggiuntivo:

- *Bodies*: all'interno di *Tagslam*, ogni tag ed ogni telecamera presenti devono essere associati ad un "corpo" del sistema. In questo modo, le pose di ciascuno di essi saranno espresse rispetto al corpo associato. Per ogni corpo, almeno una posa deve essere fornita in ingresso all'algoritmo;
- *Poses*: le pose vengono classificate in funzione della loro dipendenza dal tempo, come pose statiche, ossia che rimangono fisse nel tempo, o pose dinamiche, le quali possono variare tra frame successivi. Il funzionamento prevede che le pose dei tag e delle camere siano statiche, mentre sarà il corpo a loro associato che potrà o meno muoversi. Inoltre, le pose possono essere fornite in ingresso all'algoritmo se sono conosciute, altrimenti verranno determinate dallo stesso. Per convenzione, il sistema di riferimento associato ai tag presenta il piano xy complanare al piano del tag, mentre l'asse z è ortogonale uscente dalla faccia del tag.

Nel setup sperimentale realizzato per la stima della posa dell'end effector, sono presenti 3 elementi *bodies*, denominati *lab*, *base* ed *end_effector* ed un singolo elemento camera, tutti descritti nel file di configurazione tagslam.yaml. Il corpo *lab* è considerato statico, e descrive l'ambiente di lavoro all'interno del quale vengono realizzate le misure. Ad esso sono associati 4 tag, fissati sul piano di lavoro e sulle pareti del laboratorio. Il corpo *base*, anch'esso statico, rappresenta la base fissa del manipolatore, rispetto alla quale si vuole determinare la posa dell'end effector: il suo origine viene perciò considerato coincidente con l'origine del link 0 del manipolatore e ad esso sono associati 3 tag. La posa di uno di essi è stata calibrata ed inserita nel file di configurazione. In ultimo, il corpo dinamico *end_effector*, il cui origine coincide con l'origine della terna solidale al gripper. Anche ad esso sono associati 3 tag, che sono stati fissati ad un elemento calibrato di forma parallelepipeda, che può essere facilmente afferrato e trasportato dal gripper. Anche in questo caso, la posa di uno dei tag è stata calibrata ed inserita nel file di configurazione. Il setup sperimentale descritto è illustrato in Figura 5.3 e Figura 5.4. Una volta definito il setup sperimentale e posizionata la telecamera, si procede con la raccolta dati: viene lanciato il nodo *sync_and_detect* che si occupa della detezione degli Apriltag presenti all'interno del video registrato dalla telecamera. I numeri identificativi dei tag visualizzati vengono pubblicati su un apposito topic. Al sistema viene quindi fornita in ingresso una posa target per l'end effector: i tag presenti vengono registrati all'interno di una bag durante l'esecuzione della traiettoria del braccio robotico. La bag viene utilizzata come input all'algoritmo di *Tagslam*, che si occupa di determinare le pose dei tag, e quindi dei corpi associati, rispetto ad ogni sistema di riferimento presente. Le trasformazioni di interesse sono quelle fra la terna solidale all'end effector e la terna solidale alla base del robot: la posa ottenuta in uscita viene dunque confrontata con quella target fornita in ingresso, per determinare gli errori di posizionamento e di orientazione. Al fine di ottenere risultati attendibili, la camera con la quale si effettuano le misurazioni deve essere opportunamente calibrata. La calibrazione è stata eseguita tramite il pacchetto ROS *camera_calibration*[12], mostrando alla telecamera una scacchiera contenente 8x8 caselle, con pose differenti. Il pacchetto fornisce in uscita un file .yaml al cui interno sono presenti i parametri intrinseci della camera calibrata, che viene fornito in ingresso all'algoritmo di *Tagslam*. La ricostruzione del setup sperimentale, con la visualizzazione dei tag, dei corpi e dei sistemi di riferimento associati è illustrata in Figura 5.5 e Figura 5.6.

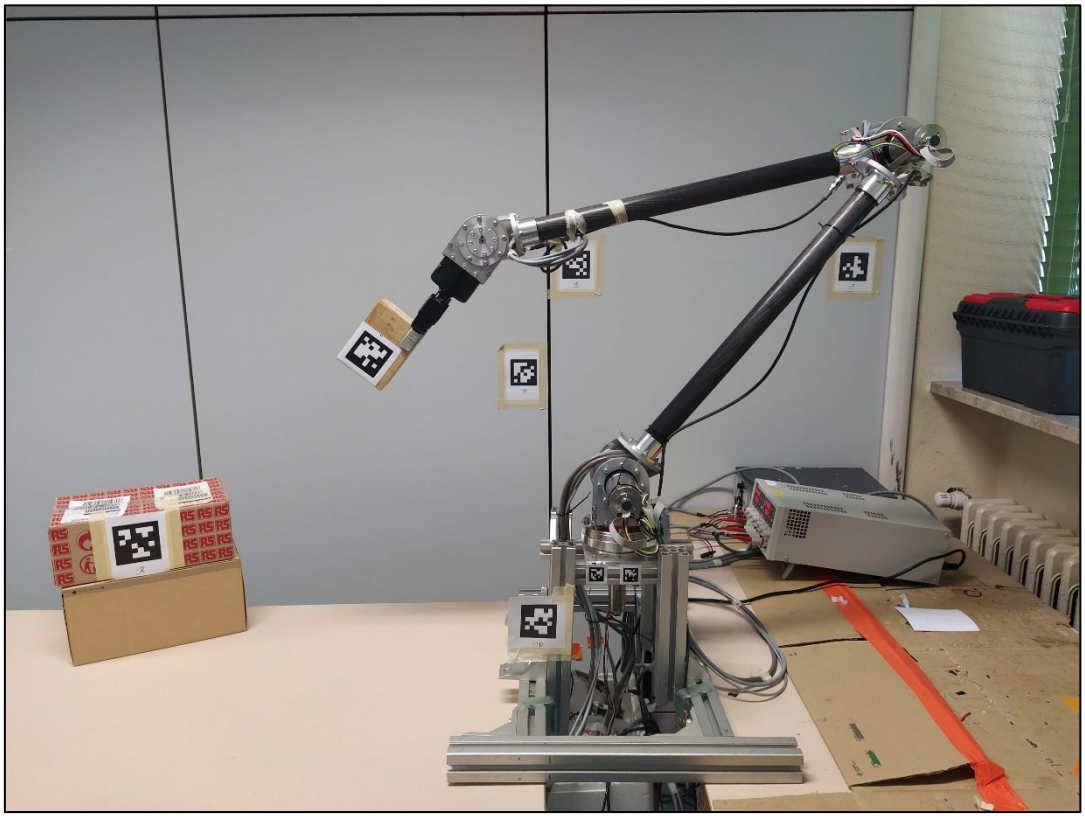

Figura 5.3 – Setup sperimentale per i test di accuratezza e ripetibilità del manipolatore.

Figura 5.4 – Setup sperimentale per i test di accuratezza e ripetibilità del manipolatore.

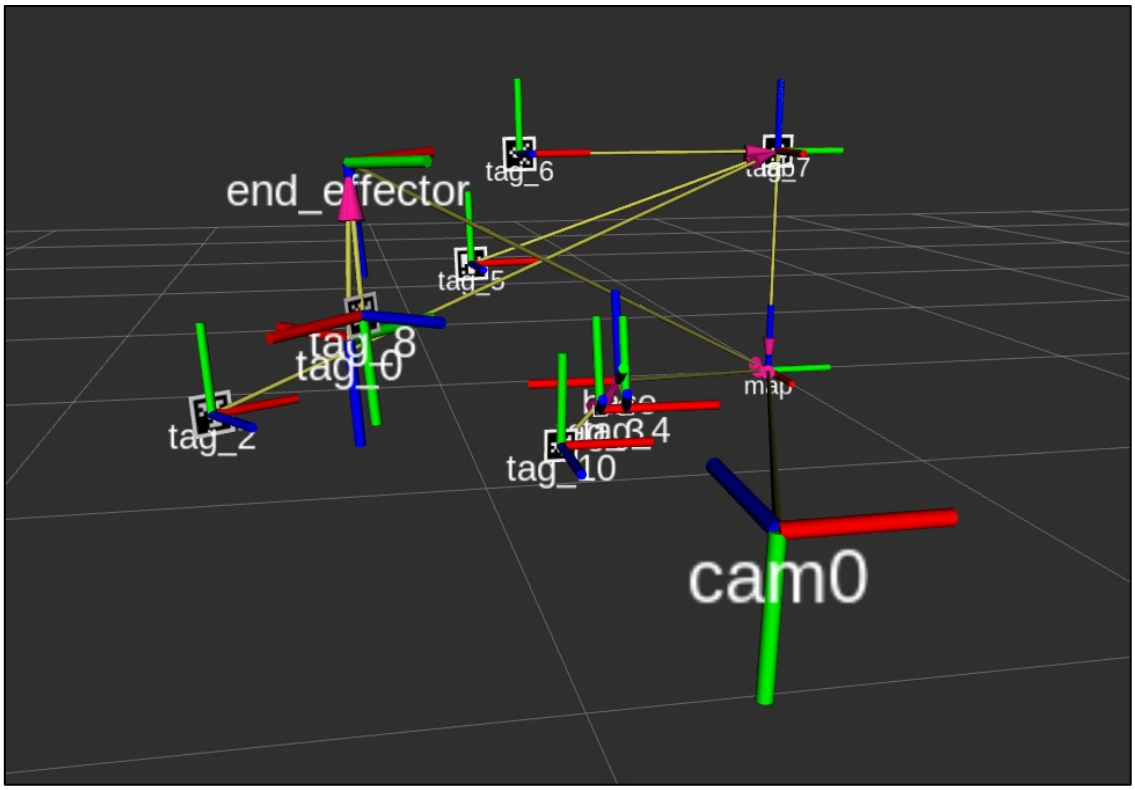

Figura 5.5 – Ricostruzione del setup sperimentale in Rviz.

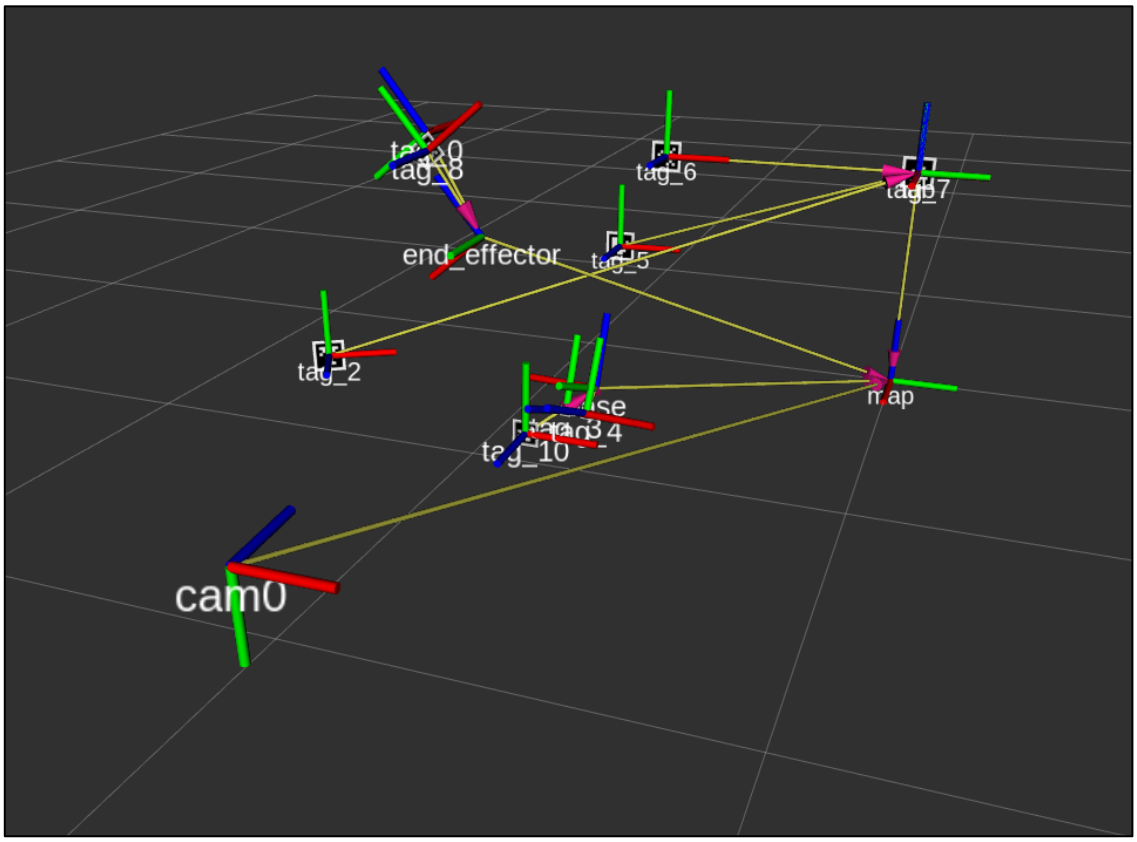

Figura 5.6 – Ricostruzione del setup sperimentale in Rviz.

5.1.2 Risultati ottenuti

I dati raccolti dalle serie di test eseguiti sono stati utilizzati per la determinazione dell'accuratezza e della ripetibilità del sistema, in termini di posizionamento ed orientazione dell'end effector nello spazio di lavoro.

L'accuratezza sulla posa raggiunta dall'end effector può essere suddivisa in due componenti: l'accuratezza sulla posizione dell'end effector, definita come la differenza fra la posizione della posa comandata ed il baricentro delle posizioni effettivamente ottenute. Le equazioni 5.1-5.4, mostrano il calcolo dell'accuratezza sulla posizione totale e lungo ogni asse:

$$AP_P = \sqrt{(\bar{X} - X_c)^2 + (\bar{Y} - Y_c)^2 + (\bar{Z} - Z_c)^2} \quad (5.1)$$

$$AP_X = (\bar{X} - X_c)^2 \quad (5.2)$$

$$AP_Y = (\bar{Y} - Y_c)^2 \quad (5.3)$$

$$AP_Z = (\bar{Z} - Z_c)^2 \quad (5.4)$$

Con:

$$\bar{X} = \frac{1}{n}\sum_j X_j \quad (5.5)$$

$$\bar{Y} = \frac{1}{n}\sum_j Y_j \quad (5.6)$$

$$\bar{Z} = \frac{1}{n}\sum_j Z_j \quad (5.7)$$

Dove $\bar{X}$, $\bar{Y}$ e $\bar{Z}$ sono le coordinate del baricentro delle misure ottenute dalle n ripetizioni delle varie pose, X_c, Y_c e Z_c sono le coordinate delle pose comandate e X_j, Y_j e Z_j sono le coordinate della generica posa j. L'accuratezza sull'orientazione dell'end effector, definita come la differenza fra l'orientazione della posa comandata e la media delle orientazioni effettivamente ottenute. L'equazione 5.8 mostra il calcolo dell'accuratezza sull'orientazione attorno all'asse di pitch:

$$AP_{pitch} = (\bar{\vartheta} - \vartheta_c) \quad (5.8)$$

Con:

$$\bar{\vartheta} = \frac{1}{n}\sum_j \vartheta_j \quad (5.9)$$

Dove $\bar{\vartheta}$ è il valore medio degli angoli di pitch misurati ϑ_j per ogni singola posa e ϑ_c è l'angolo di pitch della posa comandata.
La ripetibilità del posizionamento del sistema, per una singola posa, è espressa dal valore del raggio della sfera centrata nel baricentro delle misure RP_l, calcolata secondo l'equazione 5.10:

$$RP_l = \bar{l} + 3S_l \quad (5.10)$$

Con:

$$\bar{l} = \frac{1}{n}\sum_j l_j \quad (5.11)$$

$$l_j = \sqrt{\left(X_j - \bar{X}\right)^2 + \left(Y_j - \bar{Y}\right)^2 + \left(Z_j - \bar{Z}\right)^2} \quad (5.12)$$

$$S_l = \sqrt{\frac{\sum_j \left(l_j - \bar{l}\right)^2}{n-1}} \quad (5.13)$$

Dove l_j indica la distanza Euclidea del j simo punto misurato dal baricentro ed $\bar{l}$ il valore medio delle distanze misurate. S_l indica la deviazione standard delle misure rispetto al baricentro.
La ripetibilità dell'orientazione del sistema attorno all'angolo di pitch, per una singola posa, è espressa dalla deviazione standard S_{pitch} attorno al valore medio delle orientazioni misurate, come espresso dall'equazione 5.14:

$$RP_{pitch} = \pm 3S_{pitch} = \pm 3\sqrt{\frac{\sum_j \left(\vartheta_j - \bar{\vartheta}\right)^2}{n-1}} \quad (5.14)$$

In Figura 5.7, Figura 5.8, Figura 5.9, Figura 5.10 e Figura 5.11 sono illustrati i risultati ottenuti in termini di accuratezza e ripetibilità del posizionamento. La posizione target da raggiungere è indicata dalla croce blu, mentre i valori ottenuti dalle misure tramite telecamera sono indicati dai cerchi rossi. La linea retta congiunge la posizione target al baricentro delle misure effettuate e fornisce un indice visivo dell'accuratezza del sistema. In azzurro viene riportata la sfera che indica la ripetibilità sul posizionamento del sistema, centrata nel baricentro delle misure effettuate. Dunque, l'accuratezza del sistema è determinata dal contributo degli errori sistematici presenti durante l'esecuzione delle misure, che comportano la presenza di un offset fra la posa misurata e quella comandata, e dal contributo della ripetibilità del sistema, ossia dalla sua precisione, la quale è influenzata dagli errori di misura di tipo random.

In Tabella 5.2 solo elencati i risultati ottenuti dalle misure con telecamera, per ogni punto di misura. Il caso peggiore è rappresentato dal punto numero 3. Qui l'accuratezza sul posizionamento si attesta attorno ai 150 mm nello spazio, a causa dell'elevata componente lungo l'asse z, pari a circa 130 mm. Per quanto riguarda l'accuratezza sull'orientazione, essa vale quasi 1 rad (circa 50°). Dal confronto con i valori ottenuti negli altri 4 punti di misura, si nota un divario piuttosto elevato. Ciò è attribuibile ad errori nel riconoscimento della posa degli Apriltag solidali all'end effector, spiegabili con il fatto che il punto di misura in questione è quello più lontano e meno visibile dalla telecamera, che vanno a penalizzare l'accuratezza. Infatti, i punti più vicini alla telecamera, numero 1 e 3, sono quelli che forniscono accuratezze e ripetibilità di un ordine di grandezza più piccole rispetto al caso 4. I valori medi di accuratezza di posizionamento e di orientazione si attestano a circa 70 mm e 0.35 rad (circa 20°), ricordando che sono fortemente penalizzati dalle misure nel punto 4. Dunque, si è ritenuto plausibile non tenere in considerazione il punto di misura numero 3 nella determinazione delle prestazioni del manipolatore, in quanto i risultati ottenuti in questa posizione sono fortemente influenzati dall'algoritmo di determinazione della posa.

I risultati ottenuti non soddisfano i requisiti funzionali sull'accuratezza del posizionamento e dell'orientazione dell'end effector. La fonte di errore di maggiore influenza, riscontrabile anche visivamente durante lo svolgimento delle manovre da parte del sistema, è dovuta alla presenza di giochi importanti tra gli elementi in moto relativo all'interno dei giunti. I giochi sono particolarmente importanti nella movimentazione del giunto numero 1 alla base del robot. La causa è presenza di una flangia realizzata tramite stampa 3D, la quale non rispetta le tolleranze dimensionali sulla cava per la linguetta che trasmette il moto, ed è inoltre soggetta ad una maggiore usura. Il componente in questione è stato stampato in 3D poiché sono state apportate modifiche alla versione originale successivamente alla realizzazione alle macchine utensili di tutti gli altri componenti in lega di alluminio. Dunque, per poter completare l'assemblaggio del manipolatore e avviare la fase di integrazione software, di controllo e di test è stato necessario realizzarne un modello in tempi brevi. Lo stesso verrà comunque realizzato nuovamente in lega di alluminio, con particolare attenzione alle tolleranze dimensionali e geometriche necessarie sulla cava per la linguetta, al fine di recuperare i giochi ora presenti al giunto 1. Ulteriore fonte di errore può essere ricondotta alla discordanza fra il modello cinematico ideale e quello effettivo del manipolatore, come citato in precedenza. Questa potrà essere calmierata procedendo con la calibrazione cinematica del sistema, in modo tale da determinarne i parametri di D-H reali. In questo modo sarà possibile aggiornare il modello cinematico ed aumentare notevolmente le prestazione della cinematica diretta ed inversa.

	P1	**P2**	**P3**	**P4**	**P5**
$\boldsymbol{AP_P}$ $[\boldsymbol{m}]$	0,0184	0,0524	0,1516	0,0814	0,0349
$\boldsymbol{AP_X}$ $[\boldsymbol{m}]$	0,0117	0,0151	0,0028	0,0653	0,0168
$\boldsymbol{AP_Y}$ $[\boldsymbol{m}]$	0,0005	0,0187	0,0702	0,0366	0,0256
$\boldsymbol{AP_Z}$ $[\boldsymbol{m}]$	0,0142	0,0465	0,1343	0,0321	0,0168
$\boldsymbol{AP_{pitch}}$ $[\boldsymbol{rad}]$	0,07	0,1792	0,9404	0,346	0,1774
$\boldsymbol{RP_l}$ $[\boldsymbol{m}]$	0,01	0,0307	0,1547	0,0151	0,0206
$\boldsymbol{RP_{pitch}}$ $[\boldsymbol{rad}]$	0	0	0	0	0

Tabella 5.2 – Risultati ottenuti per accuratezza e ripetibilità del manipolatore.

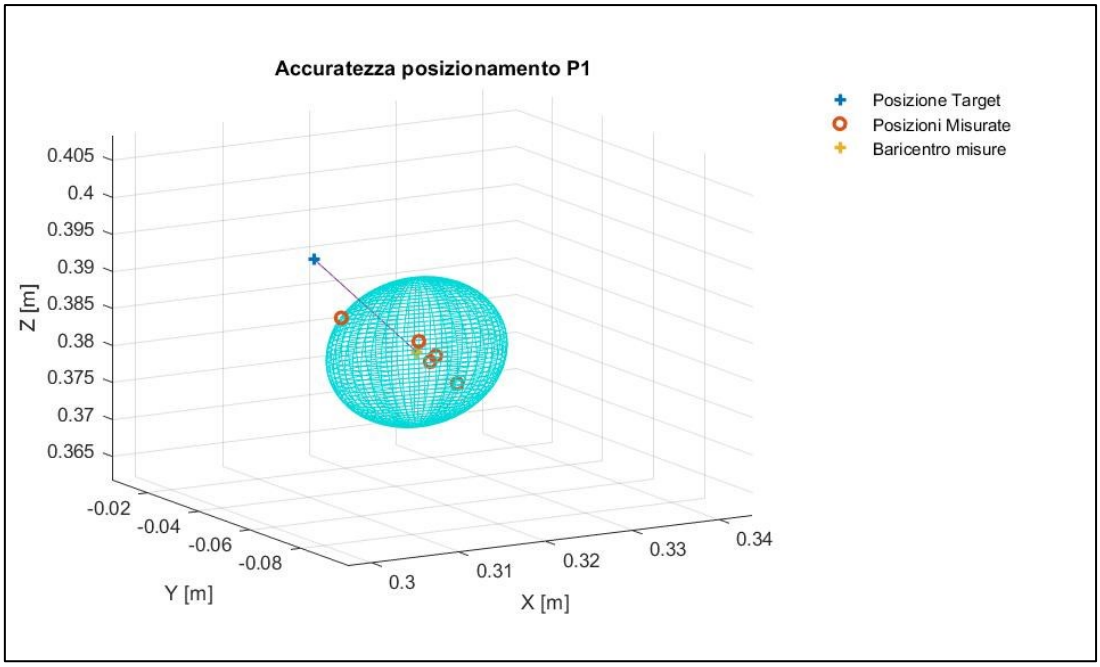

Figura 5.7 – Accuratezza sul posizionamento nel punto di misura 1.

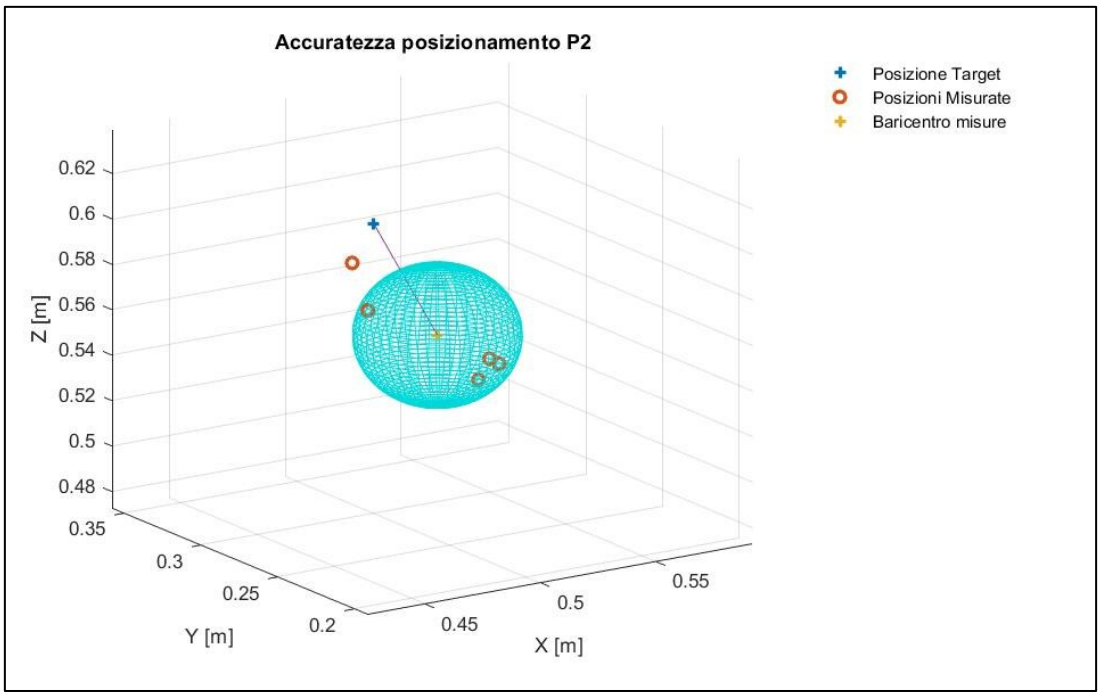

Figura 5.8 – Accuratezza sul posizionamento nel punto di misura 2.

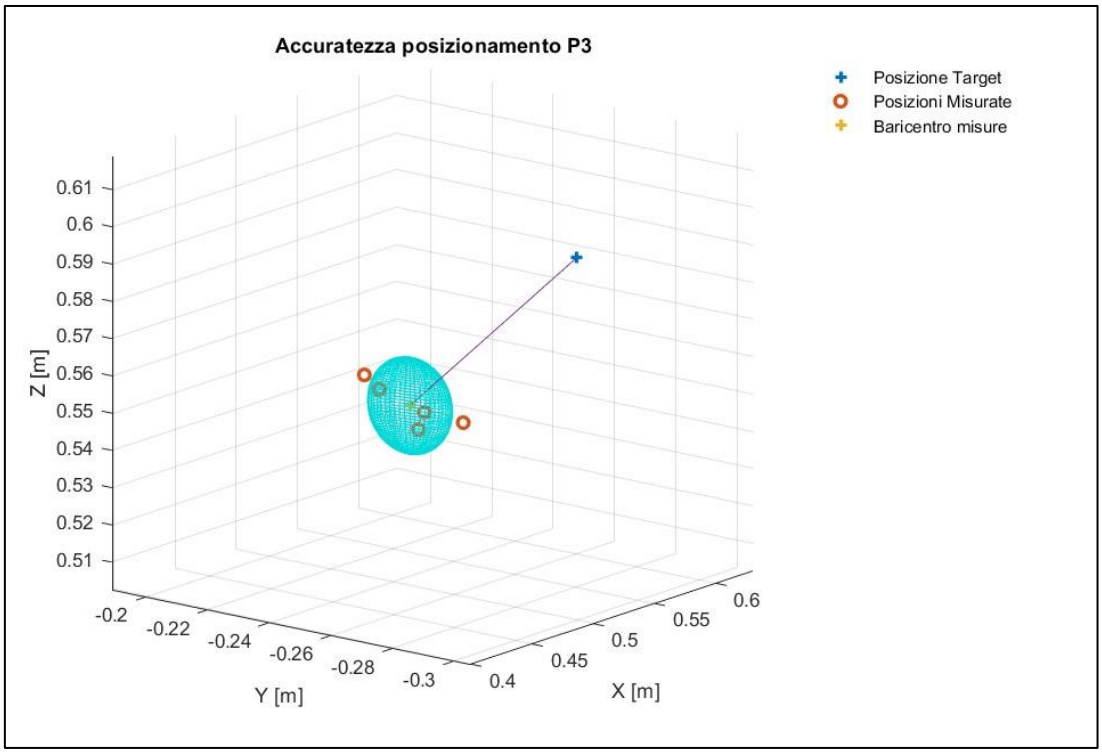

Figura 5.9 – Accuratezza sul posizionamento nel punto di misura 3.

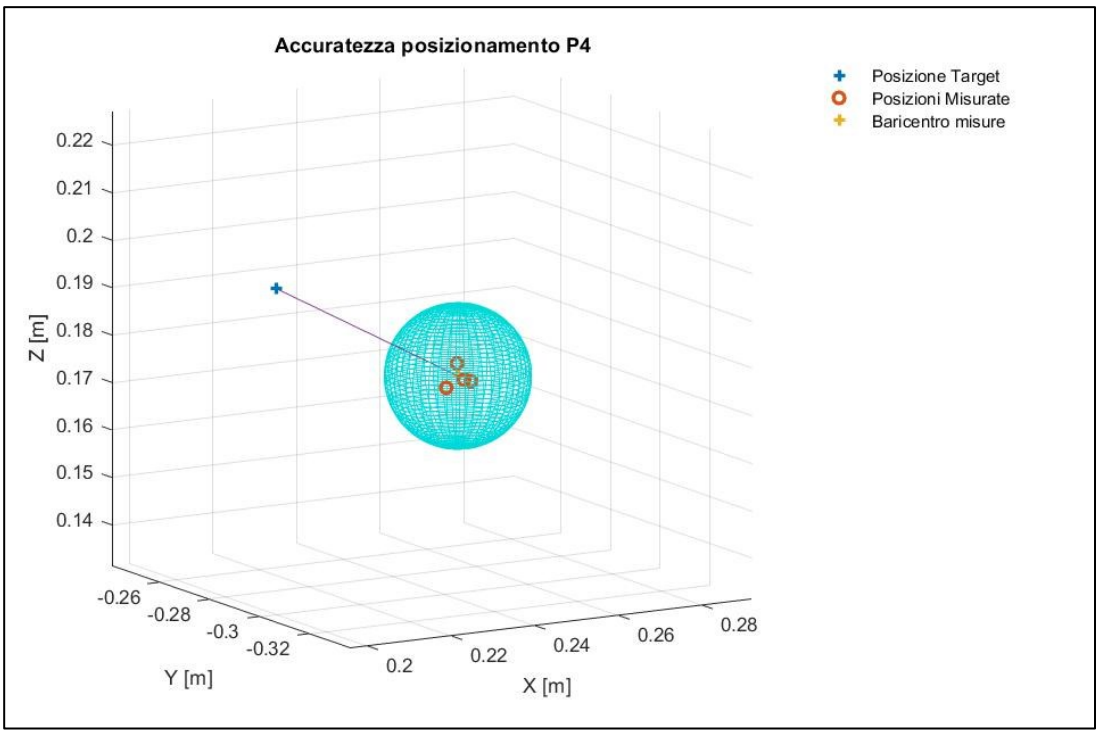

Figura 5.10 – Accuratezza sul posizionamento nel punto di misura 4.

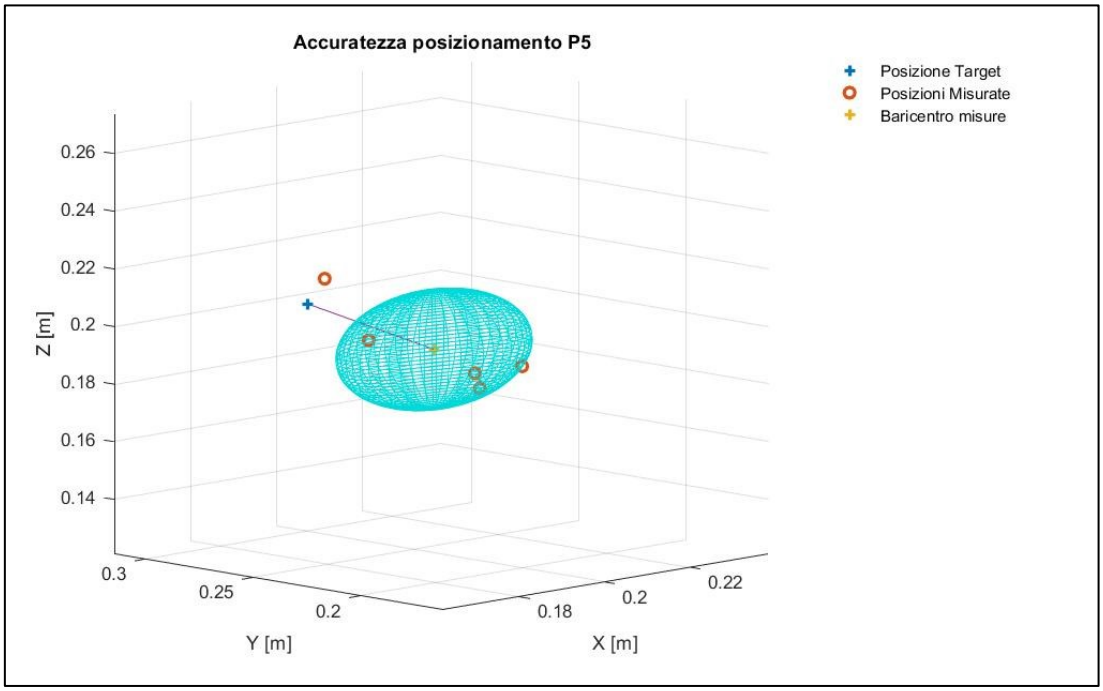

Figura 5.11 – Accuratezza sul posizionamento nel punto di misura 5.

Di seguito vengono illustrate le differenze tra le traiettorie percorse dall'end effector durante i test, ricostruite dalle misure della telecamera tramite il pacchetto *Tagslam* e dalle misure degli encoder rispettivamente. Quest'ultima è stata ricostruita tramite le equazioni della cinematica diretta del manipolatore. Il processo viene svolto in maniera automatica dal nodo *robot_state_publisher*, una volta fornito in ingresso il vettore delle variabili di giunto determinato dalle misure degli encoder. La traiettoria ricostruita tramite le misure degli encoder differisce dalla prima in quanto la cinematica diretta viene computata considerando che il modello cinematico del manipolatore sia quello ideale definito nella descrizione del file URDF. In realtà ciò non è vero, ma bensì nel sistema reale sono presenti altri effetti non tenuti in considerazione come i giochi che possono essere presenti tra gli organi dei giunti in moto relativo, la non perfetta lavorazione ed assemblaggio dei componenti del braccio robotico. La Figura 5.12 mostra la traiettoria completa percorsa nello spazio dall'end effector durante le prove sperimentali, mentre le Figura 5.13, Figura 5.14 e Figura 5.15 mostrano le singole coordinate dell'end effector.

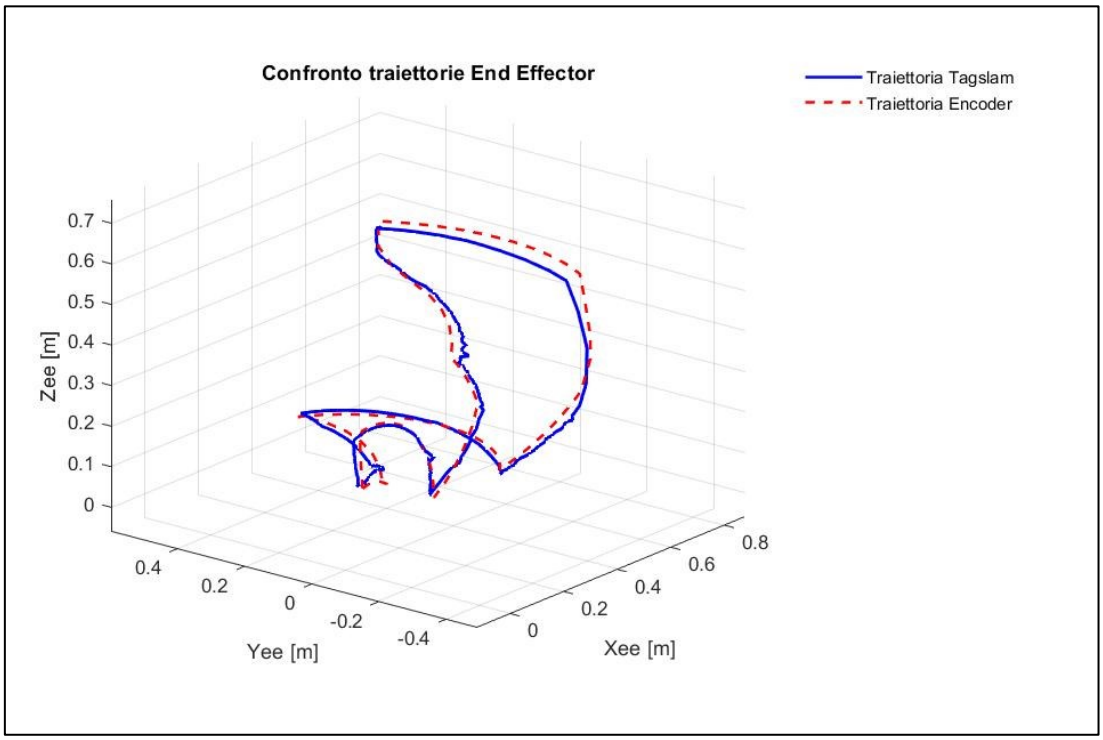

Figura 5.12 – Traiettoria completa percorsa dall'end effector durante i test.

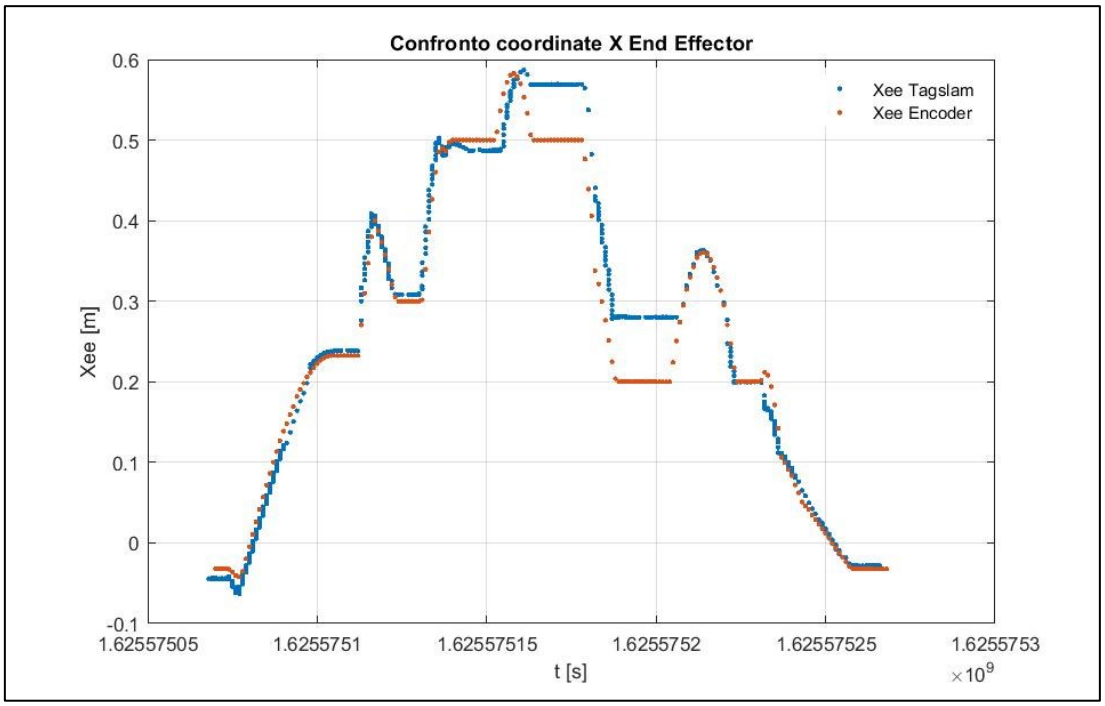

Figura 5.13 – Andamento della coordinata X dell'end effector nel tempo durante i test.

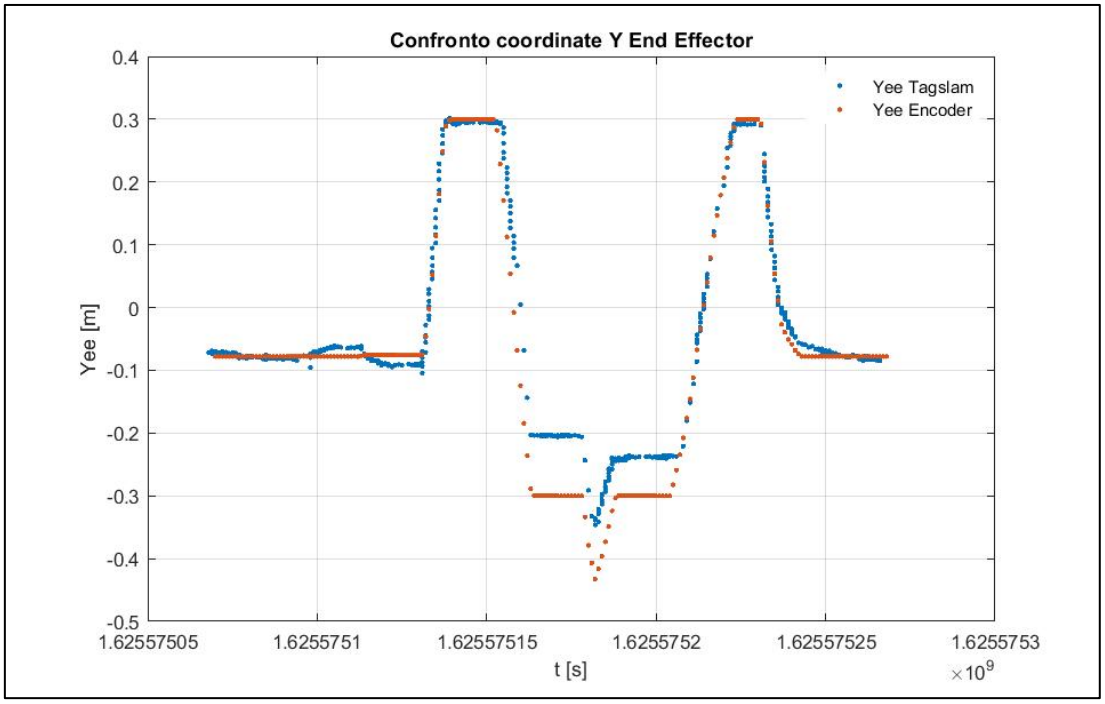

Figura 5.14 – Andamento della coordinata Y dell'end effector nel tempo durante i test.

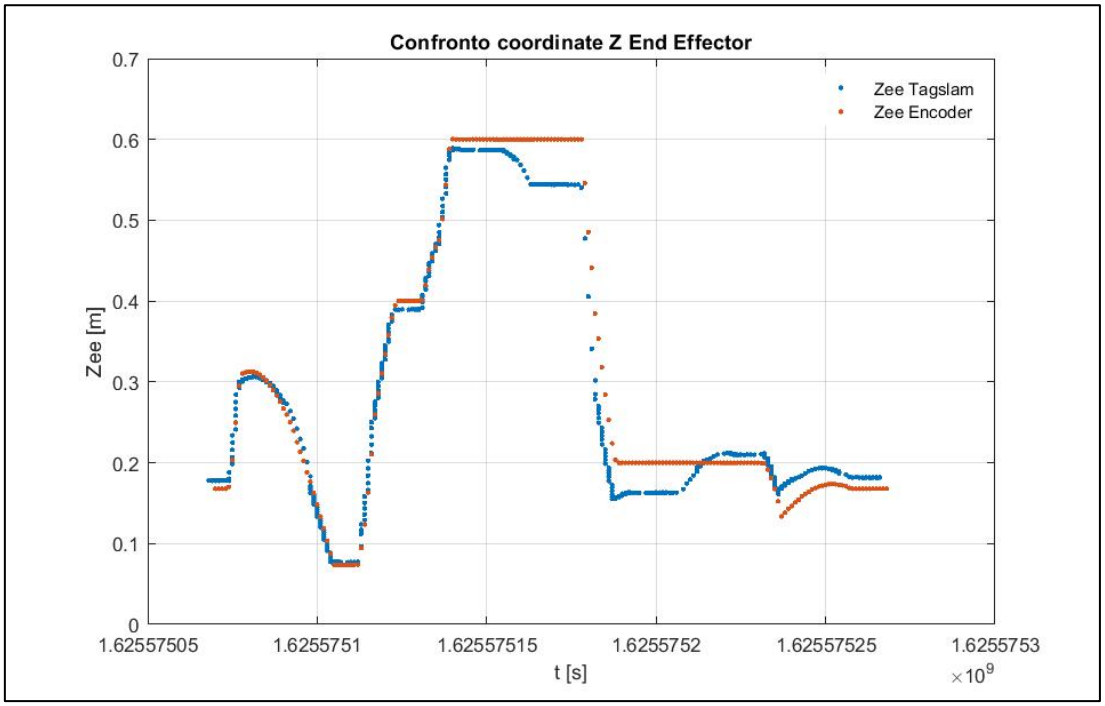

Figura 5.15 – Andamento della coordinata Z dell'end effector nel tempo durante i test.

6 Conclusioni

Nel presente elaborato è stato illustrato il lavoro svolto per la progettazione e la realizzazione di un braccio robotico a quattro gradi di libertà da integrare sul rover di Ateneo. L'obiettivo principale che questo progetto di tesi si è posto è stato lo sviluppo della capacità del manipolatore di posizionare il suo end effector in un qualsiasi punto dello spazio di lavoro con una determinata posa, rispettando i requisiti funzionali scelti e i vincoli imposti.

La scelta del modello cinematico è ricaduta su un manipolatore a quattro gradi di libertà rotativi ed è stata frutto di un compromesso tra la complessità del sistema ed i task effettivamente realizzabili. Ciò risulta sufficiente per l'applicazione desiderata.

Un processo iterativo per la scelta dei motoriduttori da installare ai giunti è stato implementato, sulla base dei dati ottenuti da simulazioni svolte in ambiente Matlab e Simulink. Su questa scelta si è basato il design meccanico dei 4 giunti, che sono stati concepiti per integrare i motoriduttori al loro interno. Così facendo è stato possibile minimizzare gli ingombri assiali e radiali dei giunti stessi. Per consentire la raccolta di campioni dal terreno ed il loro trasporto, come end effector del manipolatore è stato scelto un gripper a due dita dotato di sensore di forza.

Una volta che il braccio robotico è stato assemblato e le schede di controllo motori sono state settate correttamente, si è proceduto all'integrazione del sistema software con il framework ROS, in funzione del successivo funzionamento in simbiosi con il rover Morpheus: un workspace dedicato alla gestione e al controllo del manipolatore è stato sviluppato, esso contiene l'interfaccia hardware con i motoriduttori dei giunti, la descrizione cinematica e dinamica del braccio robotico, la pipeline di pianificazione della traiettoria dei giunti per il raggiungimento di una posa target per l'end effector.

In ultimo, le prestazioni del sistema sono state determinate tramite una campagna di test. È stato realizzato un setup sperimentale al fine di eseguire misure della posa di vari Apriltag, solidali all'end effector e alla base del manipolatore, per determinare i valori di accuratezza e ripetibilità, sul posizionamento e sull'orientazione dell'end effector. I risultati ottenuti sono stati discussi, le problematiche messe in luce assieme agli sviluppi futuri del progetto.

Bibliografia

1. Golombek, M. P. *et al.* Overview of the Mars Pathfinder Mission and assessment of landing site predictions. *Science (80-.).* **278**, 1743–1748 (1997).
2. Tunstel, E. *et al.* Mars Exploration Rover mobility and robotic arm operational performance. *Conf. Proc. - IEEE Int. Conf. Syst. Man Cybern.* **2**, 1807–1814 (2005).
3. Billing, P. & Fleischner, C. Mars science laboratory robotic arm. *14th Eur. Sp. Mech. Tribol. Symp. – ESMATS 2011* 363–370 (2011).
4. Qian, Y. *et al.* Young lunar mare basalts in the Chang'e-5 sample return region, northern Oceanus Procellarum. *Earth Planet. Sci. Lett.* **555**, 116702 (2021).
5. Zou, Y. *et al.* Scientific objectives and payloads of Tianwen-1, China's first Mars exploration mission. *Adv. Sp. Res.* **67**, 812–823 (2021).
6. Chiodini, S. *et al.* Morpheus: a Field Robotics Testbed for Soil Sampling and Autonoumus Navigation. (2015).
7. Foote, T. Tf: The transform library. *IEEE Conf. Technol. Pract. Robot Appl. TePRA* (2013) doi:10.1109/TePRA.2013.6556373.
8. Tsouroukdissian, A. ros_control: An overview. (2020) doi:10.36288/roscon2014-900186.
9. Diankov, R. & Kuffner, J. OpenRAVE : A Planning Architecture for Autonomous Robotics. *Robotics* 34 (2008).
10. Pfrommer, B. TagSLAM : Robust SLAM with Fiducial Markers.
11. Olson, E. AprilTag: A robust and flexible visual fiducial system. *Proc. - IEEE Int. Conf. Robot. Autom.* 3400–3407 (2011) doi:10.1109/ICRA.2011.5979561.
12. Zhang, Z. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.* **22**, 1330–1334 (2000).

Appendice

```
#include <moveit/move_group_interface/move_group_interface.h>
#include <moveit/planning_scene_interface/planning_scene_interface.h>
#include <moveit_msgs/DisplayRobotState.h>
#include <moveit_msgs/DisplayTrajectory.h>
#include <moveit_msgs/AttachedCollisionObject.h>
#include <moveit_msgs/CollisionObject.h>
#include <moveit/robot_model_loader/robot_model_loader.h>
#include <moveit/robot_model/robot_model.h>
#include <moveit/robot_state/robot_state.h>
#include <moveit_visual_tools/moveit_visual_tools.h>
#include <tf2/LinearMath/Quaternion.h>
#include <tf2/convert.h>
#include <geometry_msgs/PoseStamped.h>


int main(int argc, char** argv)
{
  ros::init(argc, argv, "pose");
  ros::NodeHandle node_handle;
  ros::AsyncSpinner spinner(1);
  spinner.start();

  namespace rvt = rviz_visual_tools;
  moveit_visual_tools::MoveItVisualTools visual_tools("braccio");

  // start move group interface

  static const std::string PLANNING_GROUP = "braccio";
  moveit::planning_interface::MoveGroupInterface
move_group_interface(PLANNING_GROUP);
  moveit::planning_interface::PlanningSceneInterface
planning_scene_interface;

  ros::Publisher                    display_publisher                    =
node_handle.advertise<moveit_msgs::DisplayTrajectory>("/move_group/dis
play_planned_path", 1, true);
  moveit_msgs::DisplayTrajectory display_trajectory;

  ROS_INFO("Reference                    frame:                    %s",
move_group_interface.getPlanningFrame().c_str());
  ROS_INFO("Reference                    frame:                    %s",
move_group_interface.getEndEffectorLink().c_str());

  move_group_interface.setMaxVelocityScalingFactor(1);
  move_group_interface.setMaxAccelerationScalingFactor(1);

  // construct a robot model

  robot_model_loader::RobotModelLoader
robot_model_loader("robot_description");
  const       moveit::core::RobotModelPtr&       kinematic_model       =
robot_model_loader.getModel();
  ROS_INFO("Model          frame:          %s",          kinematic_model-
>getModelFrame().c_str());
```

```
  moveit::core::RobotStatePtr                    kinematic_state(new
moveit::core::RobotState(kinematic_model));
  kinematic_state->setToDefaultValues();
  const     moveit::core::JointModelGroup*     joint_model_group     =
kinematic_model->getJointModelGroup("braccio");
  const  std::vector<std::string>&  joint_names  =  joint_model_group-
>getVariableNames();

  // Planning to a Pose goal in cartesian space

  move_group_interface.setGoalJointTolerance(0.001);

  double x1, y1, z1, pitch1, x2, y2, z2, pitch2, x3, y3, z3, pitch3, x4,
y4, z4, pitch4, x5, y5, z5, pitch5;

  node_handle.getParam("/pose_node/x1", x1);
  node_handle.getParam("/pose_node/y1", y1);
  node_handle.getParam("/pose_node/z1", z1);
  node_handle.getParam("/pose_node/pitch1", pitch1);

  node_handle.getParam("/pose_node/x2", x2);
  node_handle.getParam("/pose_node/y2", y2);
  node_handle.getParam("/pose_node/z2", z2);
  node_handle.getParam("/pose_node/pitch2", pitch2);

  node_handle.getParam("/pose_node/x3", x3);
  node_handle.getParam("/pose_node/y3", y3);
  node_handle.getParam("/pose_node/z3", z3);
  node_handle.getParam("/pose_node/pitch3", pitch3);

  node_handle.getParam("/pose_node/x4", x4);
  node_handle.getParam("/pose_node/y4", y4);
  node_handle.getParam("/pose_node/z4", z4);
  node_handle.getParam("/pose_node/pitch4", pitch4);

  node_handle.getParam("/pose_node/x5", x5);
  node_handle.getParam("/pose_node/y5", y5);
  node_handle.getParam("/pose_node/z5", z5);
  node_handle.getParam("/pose_node/pitch5", pitch5);

  visual_tools.prompt("Press 'next' in the RvizVisualToolsGui window to
continue");

  std::vector<double>  joint_values_initial  =  {0,  -2,  0.2,  0};    //
movimento iniziale
  move_group_interface.setJointValueTarget(joint_values_initial);
  move_group_interface.move();

  visual_tools.prompt("Press 'next' in the RvizVisualToolsGui window to
continue");

  geometry_msgs::PoseStamped target_pose1; // posizione 1
  tf2::Quaternion quaternion1;
  quaternion1.setRPY(0.0, pitch1, 0);
  quaternion1.normalize();

  target_pose1.header.frame_id = "world";
  target_pose1.pose.position.x = x1;
  target_pose1.pose.position.y = y1;
```

```
  target_pose1.pose.position.z = z1;
  target_pose1.pose.orientation.x = quaternion1.x();
  target_pose1.pose.orientation.y = quaternion1.y();
  target_pose1.pose.orientation.z = quaternion1.z();
  target_pose1.pose.orientation.w = quaternion1.w();

  move_group_interface.setJointValueTarget(target_pose1.pose, "link3");
  moveit::planning_interface::MoveGroupInterface::Plan joint_plan1;

  bool  joint_success1  =  (move_group_interface.plan(joint_plan1)  ==
moveit::planning_interface::MoveItErrorCode::SUCCESS);
  ROS_INFO("Visualizing plan 1 (pose goal) %s", joint_success1 ? "" :
"FAILED");
  move_group_interface.move();

  visual_tools.prompt("Press 'next' in the RvizVisualToolsGui window to
continue");

  geometry_msgs::PoseStamped target_pose2; // posizione 2
  tf2::Quaternion quaternion2;
  quaternion2.setRPY(0.0, pitch2, 0);
  quaternion2.normalize();

  target_pose2.header.frame_id = "world";
  target_pose2.pose.position.x = x2;
  target_pose2.pose.position.y = y2;
  target_pose2.pose.position.z = z2;
  target_pose2.pose.orientation.x = quaternion2.x();
  target_pose2.pose.orientation.y = quaternion2.y();
  target_pose2.pose.orientation.z = quaternion2.z();
  target_pose2.pose.orientation.w = quaternion2.w();

  move_group_interface.setJointValueTarget(target_pose2.pose, "link3");
  moveit::planning_interface::MoveGroupInterface::Plan joint_plan2;

  bool  joint_success2  =  (move_group_interface.plan(joint_plan2)  ==
moveit::planning_interface::MoveItErrorCode::SUCCESS);
  ROS_INFO("Visualizing plan 1 (pose goal) %s", joint_success2 ? "" :
"FAILED");
  move_group_interface.move();

  visual_tools.prompt("Press 'next' in the RvizVisualToolsGui window to
continue");

  geometry_msgs::PoseStamped target_pose3; // posizione 3
  tf2::Quaternion quaternion3;
  quaternion3.setRPY(0.0, pitch3, 0);
  quaternion3.normalize();

  target_pose3.header.frame_id = "world";
  target_pose3.pose.position.x = x3;
  target_pose3.pose.position.y = y3;
  target_pose3.pose.position.z = z3;
  target_pose3.pose.orientation.x = quaternion3.x();
  target_pose3.pose.orientation.y = quaternion3.y();
  target_pose3.pose.orientation.z = quaternion3.z();
  target_pose3.pose.orientation.w = quaternion3.w();

  move_group_interface.setJointValueTarget(target_pose3.pose, "link3");
```

```
  moveit::planning_interface::MoveGroupInterface::Plan joint_plan3;

  bool joint_success3 = (move_group_interface.plan(joint_plan3) ==
moveit::planning_interface::MoveItErrorCode::SUCCESS);
  ROS_INFO("Visualizing plan 1 (pose goal) %s", joint_success3 ? "" :
"FAILED");
  move_group_interface.move();

  visual_tools.prompt("Press 'next' in the RvizVisualToolsGui window to
continue");

  geometry_msgs::PoseStamped target_pose4; // posizione 4
  tf2::Quaternion quaternion4;
  quaternion4.setRPY(0.0, pitch4, 0);
  quaternion4.normalize();

  target_pose4.header.frame_id = "world";
  target_pose4.pose.position.x = x4;
  target_pose4.pose.position.y = y4;
  target_pose4.pose.position.z = z4;
  target_pose4.pose.orientation.x = quaternion4.x();
  target_pose4.pose.orientation.y = quaternion4.y();
  target_pose4.pose.orientation.z = quaternion4.z();
  target_pose4.pose.orientation.w = quaternion4.w();

  move_group_interface.setJointValueTarget(target_pose4.pose, "link3");
  moveit::planning_interface::MoveGroupInterface::Plan joint_plan4;

  bool joint_success4 = (move_group_interface.plan(joint_plan4) ==
moveit::planning_interface::MoveItErrorCode::SUCCESS);
  ROS_INFO("Visualizing plan 1 (pose goal) %s", joint_success4 ? "" :
"FAILED");
  move_group_interface.move();

  visual_tools.prompt("Press 'next' in the RvizVisualToolsGui window to
continue");

  geometry_msgs::PoseStamped target_pose5; // posizione 5
  tf2::Quaternion quaternion5;
  quaternion5.setRPY(0.0, pitch5, 0);
  quaternion5.normalize();

  target_pose5.header.frame_id = "world";
  target_pose5.pose.position.x = x5;
  target_pose5.pose.position.y = y5;
  target_pose5.pose.position.z = z5;
  target_pose5.pose.orientation.x = quaternion5.x();
  target_pose5.pose.orientation.y = quaternion5.y();
  target_pose5.pose.orientation.z = quaternion5.z();
  target_pose5.pose.orientation.w = quaternion5.w();

  move_group_interface.setJointValueTarget(target_pose5.pose, "link3");
  moveit::planning_interface::MoveGroupInterface::Plan joint_plan5;

  bool joint_success5 = (move_group_interface.plan(joint_plan5) ==
moveit::planning_interface::MoveItErrorCode::SUCCESS);
  ROS_INFO("Visualizing plan 1 (pose goal) %s", joint_success5 ? "" :
"FAILED");
  move_group_interface.move();
```

```
  visual_tools.prompt("Press 'next' in the RvizVisualToolsGui window to
continue");

  std::vector<double> joint_values_retracted = {0, 0, 0, 0};  // ritorno
configurazione 0
  move_group_interface.setJointValueTarget(joint_values_retracted);
  move_group_interface.move();

  ros::shutdown();
  return 0;
}
```