Università degli Studi di Padova

Dipartimento di tecnica e gestione dei sistemi industriali

Corso di laurea magistrale in Ingegneria Meccatronica

Sviluppo di una linea di confezionamento con robot collaborativi

Relatore:

Prof. Giovanni Boschetti

Co-Relatore

Dott. Damiano Pasetto

Laureando:

Niccolò Conterno

Anno Accademico 2018-2019

INDICE

INDICE 3

INTRODUZIONE 7

CAPITOLO 1 9

1.1 La Robotica industriale e collaborativa 9

1.1.1 Origine del termine robot e della robotica 9

1.1.2 Storia della Robotica 9

1.1.3 La robotica industriale 10

1.1.4 I CoBots, collaborative robot 11

1.1.5 Ambiti di utilizzo dei robot collaborativi 13

1.1.6 Stato dell'arte dell'industria 4.0 e l'inserimento di cobot nelle linee di produzione e assemblaggio 14

1.2 Stato dell'arte dei cobot in commercio 15

1.2.1 FANUC CR-4iA, CR-7iA, CR-7iA/L, CR-35iA 16

1.2.2 Universal Robots UR3, UR5, UR10 17

1.2.3 YASKAWA Motoman HC 10 18

1.2.4 KUKA LBR iiwa 19

1.2.5 RETHINK ROBOTICS Baxter e Sawyer 19

1.2.6 Comau AURA Project 20

1.2.7 ABB IRB 14050, detto Single Arm YuMi 22

1.2.8 Sintesi delle caratteristiche dei cobots 23

CAPITOLO 2 27

IRB 14000 YUMI IL ROBOT COLLABORATIVO DI ABB 27

2.1 Descrizione ABB IRB 14000: Yumi 27

2.2 Caratteristiche Tecniche 29

2.3 Sicurezza ed ergonomia per gli operatori 30

2.4 Sensori e calibrazione del cobot 31

2.4.1 Sensori ad Effetto Hall e calibrazione del cobot 31

2.5 Polso del braccio e grippers 34

2.6 Il controller di IRB14000 37
2.7 Il Controller IRC5 38
2.7.1 RobotWare 39
2.7.2 RobotStudio 41
2.7.3 Schede aggiuntive del controllore 43
2.8 MultiMove: strumento per coordinare più robot 45
2.9 Cinematica dello Yumi 46
2.9.1 Convenzione di Denavit-Hartenberg 47
2.9.2 Analisi dello Yumi con Denavit-Hartenberg 47
CAPITOLO 3 49
CASO APPLICATIVO: SVILUPPO DELLA LINEA DI CONFEZIONE E ANALISI DEL SOFTWARE DEL COBOT 49
3.1 Layout della stazione 50
3.2 Simulazione del ciclo 52
3.3 Programmazione del robot 53
3.3.1 Segnali di Input Output 54
3.3.2 Segnale di Supervisione del Movimento 55
3.4 L'istruzione Move 57
3.4.1 RobTarget, la struttura dati che indentifica la posa di un robot 58
3.4.2 Altre funzioni dell'istruzione Move, concorrenza di istruzioni e sincronismo 60
3.4.3 Tooldata, identificazione del tool e del carico da movimentare 61
3.4.4 Logica di calcolo del controller e utilizzo delle informazioni definite nell'istruzione 63
3.5 DESCRIZIONE DEL PROGRAMMA SVILUPPATO 64
3.5.1 La Procedura Main 65
3.6 Ciclo di carico 68
3.6.1 Descrizione della procedura di prelievo e deposito 68
3.7 Le routine Trap 69
CAPITOLO 4 71

LA CINEMATICA DIRETTA E INVERSA: SINGOLARITÀ E CONFIGURAZIONI 71
4.1 La cinematica diretta e inversa dei robot correlata alle problematiche di singolarità e alla ridondanza 71
4.1.1 Le singolarità 73
4.1.2 Ridondanza e self-motion 74
4.1.3 Cinematica inversa 76
4.2 Evitare le singolarità cinematiche tramite algoritmi 77
4.3 Evitare le singolarità cinematiche tramite funzioni presenti nel Rapid 79
CAPITOLO 5 81
SISTEMA DI SUPERVISIONE DEI MOVIMENTI E GESTIONE DELLE COLLISIONI 81
5.1 Supervisione dei movimenti e il rilevamento delle collisioni 81
5.1.1 Sistema di Supervisione dei Movimenti 81
5.1.2 Rilevamento delle collisioni: Collision Detection 82
5.1.3 Limitazioni della routine Collision Detection 84
5.1.4 Force Control e SafeMove2 85
5.2 Gestione personalizzata degli errori tramite *Error Handler* 86
5.2.1 Error Handling: gestire gli errori con routine personalizzate 86
5.2.2 Gestione separata della collisione 88
5.3 Evitare collisione tramite algoritmi 90
CONCLUSIONI 93
Appendice A 95
INDICE DELLE FIGURE 105
INDICE DELLE TABELLE 106
Bibliografia 107
Sitografia 109

INTRODUZIONE

I primi robot utilizzati in ambito industriale sono arrivati negli anni ’70 del secolo scorso con lo scopo di sostituire l’uomo nei lavori più gravosi e pericolosi. L’ottica delle case costruttrici di robot nei successivi quarant’anni è stata quella di separare l’uomo dal robot per rendere sicuro lo spazio di lavoro attraverso l’utilizzo di barriere e protezioni. Con lo sviluppo dell’automazione industriale i robot sono comparsi in molteplici settori, dall’assemblaggio di componenti elettronici all’alimentare, dalla gestione al packging andando sempre più a sostituire l’uomo in lavori semplici e ripetitivi.

L’idea dei robot collaborativi nasce dall’esigenza di poter insegnare ai robot delle azioni semplici che possano replicare attraverso la guida, da parte di un operatore, del braccio robotico. La ricerca ha portato alla produzione di robot antropomorfi a 6 e 7 assi in grado di interagire con l’uomo in sicurezza e in maniera flessibile. Il braccio dei cobot è stato arrotondato e i materiali impiegati sono stati sostituiti: attraverso l’impiego di plastiche, leghe di alluminio e imbottiture, i costruttori hanno cercato di renderlo più human friendly.

Il lavoro fatto in questa tesi nasce dalla curiosità di capire e analizzare più approfonditamente il mondo dei robot capendone i limiti e le possibilità. Si è voluto analizzare, in particolare, i robot collaborativi detti *co-robot o cobot* che negli ultimi anni stanno popolando le aziende in tutto il mondo e che facilitano l’operatore nello svolgere il proprio lavoro.

Qualche decennio fa era impensabile poter lavorare a lato di un robot senza avere barriere o protezioni tali da porre in sicurezza l’operatore. La tesi, dunque, nasce dalla curiosità di come i cobot si possono interfacciare fisicamente con l’operatore e come è possibile ripristinare il lavoro del robot nel caso in cui l’operatore entri in quell’area di lavoro o nel caso in cui avvenga una collisione: tutto ciò risulta possibile in quanto non è necessario che l’area venga identificata e protetta da barriere, per la norma ISO 15066 del 2016, ed è possibile che l’operatore venga colpito dal braccio robotico.

Il cuore della tesi è la gestione della collisione e il ripristino delle funzionalità del robot: questo risulta essere il problema principale per le stazioni robotiche di ultima generazione, dove sono presenti cobot, poiché vengono installate senza barriere tali da dividere fisicamente l’area del robot dall’esterno.

La presente tesi, come si può capire dal titolo, tratta lo sviluppo di una linea di confezionamento con robot collaborativi, in particolare con lo YuMi, il primo robot collaborativo presentato dalla casa ABB.

La tesi è stata sviluppata in azienda, presso Tech.Pa. Spa: come si può evincere dal titolo della tesi il lavoro svolto presso l’azienda è stato uno studio di fattibilità che ha portato allo sviluppo di una linea di confezionamento rispettando alcune richieste del cliente.

Il cliente aveva richiesto l’utilizzo di robot collaborativi per poter avere la possibilità di eseguire il controllo della linea senza dover fermare la linea di confezionamento e monitorare l’inserimento delle confezioni all’interno della scatola di imballaggio. La produttività dello stabilimento negli anni è aumentata, ed essendo passato a lavorare in continuo su tre turni, è stato necessario sostituire alcuni lavori ripetitivi, svolti dal personale, con dei robot. Le caratteristiche della linea saranno illustrate all’interno del terzo capitolo.

Il percorso offerto in questo elaborato parte da una ricerca sullo stato dell'arte dei robot collaborativi che le case costruttrici propongono al mercato; si sono inoltre analizzate anche le diverse caratteristiche offerte che verranno poi riassunte in una tabella.
Si prosegue quindi andando ad analizzare in particolare l'IRB14000 chiamato YuMi, cobot utilizzato per lo sviluppo della linea. Nel secondo capitolo si andrà ad analizzare alcune carattestiche che rendono lo YuMi unico e come la casa costruttrice abbia risolto alcuni problemi legati alla sicurezza per l'operatore che lavoro fianco a fianco con questo cobot.
Successivamente, nel terzo capitolo, verrà presentata la linea e il suo funzionamento presentando lo script sviluppato e andando ad illustrare alcune chiamate di sistema utili per capire il problema delle singolarità di movimento.
Nel quarto capitolo si proseguirà andando ad analizzare più approfonditamente la tematica delle singolarità, andando poi ad illustrare alcune tecniche per risolverle.
Infine, nel quinto capitolo verrà illustrato il problema delle collisioni tra cobot e operatore e come è stato implementato il programma per poter riavviare il ciclo del cobot e riprendere il normale ciclo di lavoro.

CAPITOLO 1

1.1 La Robotica industriale e collaborativa

1.1.1 Origine del termine robot e della robotica

Il termine robot deriva dal termine ceco *robota*, che significa lavoro pesante o lavoro forzato. L'introduzione di questo termine si deve allo scrittore ceco Karel Čapek, il quale usò per la prima volta il termine nel 1920 nel suo dramma teatrale *"I robot universali di Rossum"*. Il vero inventore della parola, però, fu fratello Josef, scrittore e pittore cubista, il quale aveva già affrontato il tema in un suo racconto del 1917, Opilec (L'ubriacone), in cui però aveva usato il termine *automat*, da automa.

Il termine *robotica* venne usato per la prima volta nel racconto di Isaac Asimov intitolato *"Bugiardo!"* del 1941, presente nella sua famosa raccolta "*Io, Robot*". In esso, lo scrittore citava le famose *"Tre leggi fondamentali della robotica"*

Fin dall'antichità l'uomo ha pensato di creare macchine che potessero muoversi o oggetti in grado di "prendere vita" e riprodurre i movimenti dell'uomo: basti pensare al mito di Pigmaglione, dove la statua Galatea prese vita, oppure alla Mitologia greca e latina in cui Efesto, o Vulcano, creò dei servi meccanici.

Proseguendo nella linea temporale si arriva ad avere il primo progetto di robot umanoide con *Leonardo Da Vinci*, attorno al 1495: nei suoi appunti, basati sull'uomo vitruviano, troviamo progetti dettagliati di un cavaliere meccanico in grado di alzarsi in piedi e agitare le braccia.

Il primo robot funzionante fu realizzato nel 1738 da *Jacques de Vaucanson* che fabbricò un androide che suonava il flauto e un'anatra meccanica in grado di mangiare.

Con l'aumentare della tecnologia meccanica, a partire dalla rivoluzione industriale dell'800, gli scrittori iniziarono a creare romanzi e storie creando così il filone narrativo della fantascienza e della robotica, sfociato con l'avvento del cinema in film e serie tv. Nell'immaginario comune, oggigiorno, i robot hanno la forma degli automi: ma in realtà il robot nasce come braccio meccanico in grado di compiere movimenti: nel paragrafo seguente verrà illustrata la storia dei manipolatori robotici.

1.1.2 Storia della Robotica

La storia della robotica ebbe inizio negli anni 1950-1960 quando si iniziarono a vedere manipolatori che sostituivano l'uomo nei lavori più pericolosi e insalubri. Il primo robot industriale fu creato per General Motors dalla società statunitense Unimate: questo robot assisteva l'impianto di pressofusione estraendo i getti di metallo ad altissima temperatura ed immergendoli in un bagno d'acqua per farli raffreddare.

L'altra industria statunitense leader nell'adozione dei robot all'interno del processo produttivo fu quella elettromeccanica. A metà degli anni '70 entrarono nell'industria robotica grandi aziende, come le statunitensi Cincinnati Milacron, Adept, IBM, AMF, la francese ACMARenault, l'italiana COMAU-Fiat, le tedesche Kuka

e Volkswagen, le nipponiche Fanuc, Yaskawa, Seiko. I robot furono apprezzati anche fuori dell'industria automobilistica ed elettromeccanica a partire dagli anni '80 e vennero introdotti in modo massiccio in tutti gli altri processi industriali.

I robot industriali possono essere classificati in tre stadi temporali evolutivi, ossia in tre fasi storiche, a seconda del grado di tecnologia che utilizzano per operare. Si possono distinguere così tre generazioni di robot:

- I robot della prima generazione sono quelli introdotti all'inizio degli anni '60. Essi erano in grado di compiere operazioni di carico/scarico di macchine utensili o semplici operazioni di manipolazione di pezzi e materiali. Tali robot erano progettati per ripetere una successione di operazioni predeterminate indipendentemente dai cambiamenti dell'ambiente circostante.
- I robot della seconda generazione sono stati introdotti negli anni '70. Essi erano in grado di svolgere compiti più complessi quali saldature a punto, verniciatura, taglio, foratura. Avevano elementari capacità di comunicare con l'ambiente circostante: erano infatti dotati di capacità sensoriali grazie a sensori che trasmettevano informazioni relative alla presenza, alla posizione e all'orientazione di oggetti circostanti. Queste macchine rispetto alle precedenti avevano in tal senso una maggiore conoscenza dell'ambiente.
- I robot della terza generazione sono stati introdotti negli anni '80. Sono in grado di svolgere operazioni altamente sofisticate come le operazioni di assemblaggio, la saldatura ad arco adattiva, le ispezioni tattili, la prova di componenti e prodotti, lavorazioni complesse di trasformazione di pezzi, ecc. Inoltre sono in grado di regolare attivamente i propri movimenti e di compensare i cambiamenti di posizione ed orientamento dei pezzi, cioè sono robot capaci di accorgersi di eventuali cambiamenti esterni potendo così modificare le loro azioni in modo corretto.
- È possibile individuare anche una quarta generazione di robot che sono quelli attualmente in fase di sviluppo. Questi robot saranno in grado di prendere decisioni in modo totalmente indipendente sulla base di eventi e regole -la cosiddetta intelligenza artificiale- al fine di eseguire compiti più complessi. In questi robot si cerca di realizzare anche una raffinata facoltà sensoriale in modo tale da raggiungere la vera coordinazione oculo-manuale propria dell'uomo. È inoltre probabile che i robot della quarta generazione siano delle apparecchiature non metalliche in quanto la condivisione del lavoro tra robot ed operatore richiederà maggiore sicurezza. Si tratta dei cosiddetti robot collaborativi approfonditi in questa trattazione.

1.1.3 La robotica industriale

La robotica industriale nasce dall'esigenza di agevolare operazioni ripetitive e che richiedono grandi sforzi fisici all'uomo.

Se volessimo rispondere alla domanda "Che cos'è la robotica industriale?" potremmo dare due risposte comunemente accettate e una terza più generica:

- Secondo la Robotic Institute of America (RIA), un robot è un manipolatore multifunzionale e riprogrammabile progettato per muovere materiali, parti, attrezzi o dispositivi specialistici attraverso movimenti programmati variabili, per l'esecuzione di una varietà di compiti. Un robot acquisisce inoltre informazioni dall'ambiente e si muove in modo intelligente di conseguenza.
- Nella normativa ISO 8373 troviamo questa definizione: un robot è un manipolatore controllato automaticamente, riprogrammabile, multifunzione, a tre o più assi, che può essere fissato a terra o mobile, ed è utilizzato per applicazioni di automazione industriale.
- Un robot è la connessione intelligente tra percezione e azione.

Queste tre definizioni presentano sfumature diverse ma da ciascuna di esse emerge come un robot sia molto più che un semplice dispositivo meccanico. Un robot, o meglio un sistema robotico, è infatti un sistema complesso che può essere rappresentato come una serie di sottosistemi interconnessi:

1) Una delle componenti essenziali di un sistema robotico è sicuramente il sistema meccanico. Nell'ambito della robotica per applicazioni industriali il sistema meccanico coincide con l'apparato di manipolazione. Nella robotica di servizio e nella robotica per l'esplorazione invece il sistema meccanico può essere ulteriormente dotato di un apparato di locomozione.

2) La capacità di compiere un'azione di manipolazione è fornita dal sistema di attuazione costituito dai servomeccanismi, i motori e le trasmissioni. Facendo un paragone anatomico, mentre il sistema meccanico coincide con lo scheletro, il sistema di attuazione rappresenta il sistema muscolare.

3) In analogia all'essere umano, la capacità di percezione è garantita dal sistema sensoriale in grado di acquisire dati sullo stato interno del sistema meccanico -si parla in questo caso di sensori propriocettivi, come i trasduttori di posizione- ma anche sullo stato esterno dell'ambiente -sensori esterocettivi, come i sensori di forza e i sensori di visione-.

4) Infine, la capacità di connettere percezione e azione in modo intelligente è fornita dal sistema di controllo, in grado di comandare l'esecuzione di azioni al sistema di attuazione sulla base del compito (task) che il robot deve eseguire e sulla base dei vincoli imposti dal robot stesso e dall'ambiente esterno. Il sistema di controllo è a tutti gli effetti il sistema nervoso centrale di un robot. Nella robotica industriale il sistema di controllo risiede fisicamente nell'unità di controllo, generalmente dotata di un dispositivo di interfaccia uomo-macchina e di programmazione chiamato teach pendant.

1.1.4 I CoBots, collaborative robot

I *cobots*, abbreviazione per *collaborative robots,* conosciuti anche come co-robot, sono pensati per lavorare insieme, o per meglio dire, a contatto con l'uomo condividendo le stesse aree di lavoro. Anche in questo settore la robotica industriale sta facendo passi da gigante, negli ultimi 10 anni. I cobots possono lavorare gomito a gomito con i "colleghi" umani perché sono automuniti di dispositivi di sicurezza che non mettono a rischio la vita degli operatori, anzi svolgono lavori ripetitivi, rischiosi, faticosi e usuranti che vengono affidati a un robot per aumentare l'efficienza della filiera produttiva.

Sono considerati i robot industriali del futuro e si mettono in netta contrapposizione rispetto ai macchinari usati a partire dagli anni Quaranta nelle industrie di tutto il mondo. Un tempo, infatti, i robot venivano strutturati per svolgere in maniera autonoma il proprio lavoro, spesso senza la necessità dell'intervento umano. Nell'Industria 4.0 invece si punta alla creazione di macchinari che possano interagire con la componente umana di un'impresa in maniera tale da rendere più efficiente, per entrambe le parti, il lavoro. I cobots sono stati inventati nel 1996 da J. Edward Colgate e Michael Peshkin, due professori presso la Northwestern University dell'Illinois.

L'idea sui robot collaborativi, però, era già nata qualche anno prima, nel 1994, da un progetto finanziato dalla General Motors. L'obiettivo era quello di creare dei macchinari autosufficienti che fossero simili alle persone e che con gli uomini sapessero interagire in maniera complessa. I primi cobot sono stati realizzati per la sicurezza, all'interno di aziende o spazi condivisi, delle persone. Non avevano potere motorio ed era l'uomo, attraverso l'utilizzo di periferiche o appositi pannelli di controllo, a gestirne i movimenti. Con il tempo poi anche i cobot sono stati sviluppati con la capacità di spostarsi all'interno di determinati spazi. General Motors, che per anni ha utilizzato questi androidi nella gestione dei materiali industriali e nell'assemblaggio di componenti per automobili, li definisce come Intelligent Assist Device (IAD). Questi sistemi hanno ricevuto uno standard per la sicurezza per la prima volta nel 2002 che è stato poi aggiornato nel 2016.

Non ci sono barriere o gabbie che dividono l'area di lavoro dei cobot da quella degli operatori: l'installazione è facilitata in quanto non si devono prevedere aree di protezione impattando in modo positivo sull'investimento economico. Questa convivenza è resa possibile dal fatto che i robot collaborativi sono dotati di meccanismi di sicurezza, basati sul controllo della forza e sul costante monitoraggio di quanto avviene attorno a loro.

I *collaborative robots* sono specializzati nello svolgimento di compiti specifici che "imparano" direttamente sul campo. Possono essere più o meno autonomi e stanno rivoluzionando i settori della logistica e dell'automazione di fabbrica.

I cobot sono per lo più robot collaborativi antropomorfi. Per antropomorfo non si intendono robot umanoidi, ma si tratta di automi industriali dotati di telecamere e bracci flessibili in grado di eseguire contemporaneamente anche due mansioni diverse.

A differenza dei robot industriali tradizionali, che per funzionare devono essere programmati, il cobot in genere apprende work in progress[1], mentre lavora sul campo, memorizzando e replicando le manovre che gli sono state mostrate pochi minuti prima dal "collega" umano.

I cobot sono robot industriali dal prezzo relativamente contenuto e per questo sta crescendo il numero di aziende che all'interno della linea produttiva, nel settore agroalimentare e elettronico, inseriscono questi robot: è possibile acquistarli per meno di 25 mila euro, quanto un'automobile di fascia media.

Grazie a queste caratteristiche e ai costi contenuti, molte aziende italiane nell'ultimo anno hanno investito sugli impianti per aggiornarli; questo impulso è stato dato grazie al piano industriale per l'industria 4.0.

[1] In realtà si tratta di una programmazione intuitiva, basata sulla movimentazione manuale del braccio e l'apprendimento dei punti target

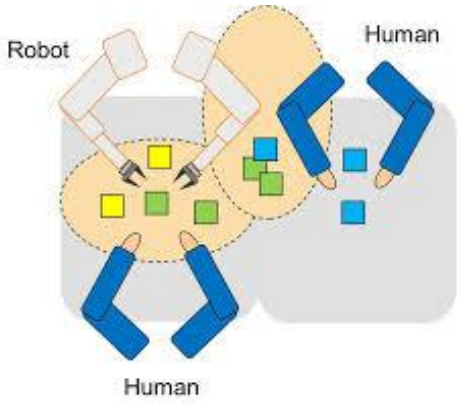


Figura 1 - Robot collaborativo che condivide lo spazio di lavoro con operatori umani

1.1.5 Ambiti di utilizzo dei robot collaborativi

I robot collaborativi non sono adatti a movimentazione di carichi pesanti o ad applicazioni molto veloci ma possono trovare spazio in tutti i settori industriali. In particolare, nelle applicazioni di:

- Lucidatura: il robot lucida anche le superfici curve e irregolari con forza impostabile per un risultato costante.
- Stampaggio ad iniezione: il robot può essere utilizzato in tutte le fasi della produzione di plastiche, con precisione e CAD enza richiesta.
- Analisi di laboratorio: sollevare gli operatori dai lavori più ripetitivi con un robot per migliorare i processi di testing.
- Avvitatura: lasciare che un robot automatizzi i processi di avvitatura ripetendo lo stesso movimento più volte con la stessa precisione e velocità, per migliorare la produttività e la qualità dei prodotti.
- Confezionamento e palletizzazione: potrebbero essere faticose e ripetitive per l'operatore e quindi si lascia che il robot svolga questi compiti.
- Incollaggio e dosatura: il robot può aggiungere efficienza ai processi continui, come incollaggio e dosatura, erogando ogni volta la corretta quantità di materiale con la massima precisione.
- Asservimento macchine: il robot può essere utilizzato per la maggior parte delle applicazioni di asservimento macchine utensili, ed è facilmente riadattabile al variare del mix produttivo.
- Assemblaggio: il robot gestisce senza sforzo il montaggio di plastiche, legni, metalli e un'ampia gamma di altri materiali, migliorando allo stesso tempo la rapidità e la qualità del processo.
- Prelievo e posizionamento: un robot può eseguire la maggior parte dei compiti di prelievo e posizionamento in modo autonomo riducendo tempi di ciclo e sprechi di materiale.
- Controllo qualità: un robot, correttamente equipaggiato, può rilevare scarti e componenti difettosi prima che siano confezionati, in modo da mantenere alta la qualità del prodotto spedito ai clienti.

1.1.6 Stato dell'arte dell'industria 4.0 e l'inserimento di cobot nelle linee di produzione e assemblaggio

Il termine "Industria 4.0" è stato utilizzato per la prima volta in Germania nel 2011, precisamente durante la Fiera di Hannover. In questa occasione un gruppo di lavoro (Henning Kagermann, Wolf-Dieter Lukas, Wolfgang Wahlster) rappresentanti dell'economia, della politica e della scienza utilizzarono questo termine per la prima volta pubblicamente in una relazione intitolata: "Industria 4.0: L'Internet delle cose sulla strada della quarta rivoluzione industriale".

L'industria 4.0 passa per il concetto di smart factory che si compone di 3 parti:

- Smart production: nuove tecnologie produttive che creano collaborazione tra tutti gli elementi presenti nella produzione ovvero collaborazione tra operatore, macchine e strumenti.
- Smart services: tutte le "infrastrutture informatiche" e tecniche che permettono di integrare i sistemi ma anche tutte le strutture che permettono, in modo collaborativo, di integrare le aziende (fornitore – cliente) tra loro e con le strutture esterne (strade, hub, gestione dei rifiuti, ecc.)
- Smart energy: tutto questo sempre con un occhio attento ai consumi energetici creando sistemi più performanti e riducendo gli sprechi di energia secondo i paradigmi tipici dell'Energia sostenibile.

La chiave di volta dell'industry 4.0 sono i sistemi ciberfisici (CPS) ovvero sistemi fisici che sono strettamente connessi con i sistemi informatici e che possono interagire e collaborare con altri sistemi CPS. Questo sta alla base della decentralizzazione e della collaborazione tra i sistemi che è strettamente connessa con il concetto di industria 4.0.

La quarta rivoluzione industriale si centra sull'adozione di alcune tecnologie definite abilitanti; alcune di queste sono "vecchie" conoscenze, concetti già presenti ma che non hanno mai sfondato il muro della divisione tra ricerca applicata e sistemi di produzione veri e propri. Oggi, invece, grazie all'interconnessione e alla collaborazione tra sistemi, il panorama del mercato globale sta cambiando portando alla customizzazione di massa e diventando di interesse per l'intero settore manifatturiero.

Le nove tecnologie abilitanti definite da Boston Consulting sono (Boston Consulting, 2015):

- Advanced manufacturing solution: sistemi avanzati di produzione, ovvero sistemi interconnessi e modulari che permettono flessibilità e performance. In queste tecnologie rientrano i sistemi di movimentazione dei materiali automatici e la robotica avanzata, che oggi entra sul mercato con i robot collaborativi o cobot.
- Additive manufacturing: sistemi di produzione additiva che aumentano l'efficienza dell'uso dei materiali.
- Augmented reality: sistemi di visione con realtà aumentata per guidare meglio gli operatori nello svolgimento delle attività quotidiane.
- Simulation: simulazione tra macchine interconnesse per ottimizzare i processi.
- Horizontal e vertical integration: integrazione e scambio di informazioni in orizzontale e in verticale tra tutti gli attori del processo produttivo.

- Industrial internet: comunicazione tra elementi della produzione, non solo all'interno dell'azienda, ma anche all'esterno grazie all'utilizzo di internet.
- Cloud: implementazione di tutte le tecnologie cloud come lo storage online delle informazioni, l'uso del cloud computing e di servizi esterni di analisi dati, ecc. Nel Cloud sono contemplate anche le tecniche di gestione di grandissime quantità di dati attraverso sistemi aperti.
- Cyber-security: l'aumento delle interconnessioni interne ed esterne aprono la porta a tutta la tematica della sicurezza delle informazioni e dei sistemi che non devono essere alterati dall'esterno.
- Big Data Analytics: tecniche di gestione di grandissime quantità di dati attraverso sistemi aperti che permettono previsioni o predizioni.

I modelli di controller robotici, di PLC, telecamere, ecc, usciti negli ultimi 5 anni e utilizzati negli impianti automatici, sono dotati di porte ethernet per creare delle reti locali legate all'impianto e quindi poter scambiare dati e segnali tra di loro, ma anche per poter interfacciarsi con altri dispositivi. In particolare, il grosso salto che molte aziende hanno fatto nello scorso anno solare è legato alla messa in rete degli impianti con il server dell'azienda. Questo è uno dei requisiti chiesti per poter entrare a far parte del piano industriale 4.0.
Lo scopo principale, per aziende dotate di molti impianti e macchinari distribuiti su una vasta area, è quello di raccogliere tutte le informazioni riguardanti la manutenzione e gli allarmi, in un database aziendale all'interno del server, creando uno storico.
Il secondo motivo è legato alla gestione degli impianti. Infatti è possibile, da remoto, modificare i dati di funzionamento dell'impianto così da poter controllare anche a distanza, da una sorta di cabina di controllo, tutti i dosaggi necessari per una impastatrice, nel caso dell'industria alimentare.
Molte aziende, grazie al Piano Nazionale Industriale 4.0, hanno deciso di installare cobot per due motivi: il primo riguarda lo spazio di impiego ridotto, in quanto non necessitano di barriere, grazie alla loro sicurezza intrinseca. Il secondo motivo è legato al primo poiché eliminando le barriere si riducono i costi.

1.2 Stato dell'arte dei cobot in commercio

In tutti i contesti aziendali sviluppati è possibile avere dei robot collaborativi fissi legati ad una postazione di lavoro in base al tipo di attività che viene richiesto, all'ambiente circostante ed al tipo di business. Inoltre, è possibile fissare i cobot su carrelli mobili avendo così dei layout di fabbrica flessibili. In particolare, è possibile trovarli nelle applicazioni di handling, assemblaggio, avvitatura, verniciatura, asservimento ad una linea di produzione, logistica e così via. Ma anche in processi di produzione più hard in cui operano da anni robot non collaborativi, come nell'automotive o nelle lavorazioni meccaniche e plastiche. Essendo, infine, prodotti leggeri e molto flessibili sia a livello operativo sia applicativo, si adattano a lavori in spazi ridotti, ad ambienti sensibili, alla manipolazione in ambito food&beverage, biomedicale o farmaceutico.[2]

[2] Automazione-plus robot collaborativi articolo del 2017

1.2.1 FANUC CR-4iA, CR-7iA, CR-7iA/L, CR-35iA

Forte di un'esperienza consolidata che registra oltre 400.000 robot industriali installati nel mondo, FANUC ha recentemente introdotto una speciale linea di robot collaborativi caratterizzata dalla copertura in morbida gomma verde. I modelli attualmente disponibili sono: CR-4iA, CR-7iA, CR-7iA/L e CR-35iA.

CR-4iA è il robot collaborativo più piccolo della gamma, con sei assi, un braccio di 550 mm ed una portata massima di 4 kg. È in grado di gestire attività leggere, ripetitive e altamente manuali. La sua natura compatta consente di eseguire i lavori più piccoli in aree con spazi limitati. Il montaggio a muro o capovolto consente di eseguire una vasta gamma di movimenti senza interferire con lo spazio di lavoro dell'operatore. Può collaborare anche su attività più complesse che richiedono un lavoro più interattivo tra robot e operatore, come ad esempio: il robot gestisce i componenti e l'operatore il controllo qualità. Oltre al design compatto e snello, la funzione di arresto di sicurezza consente di collaborare con gli operatori senza la presenza di protezioni aggiuntive. Questa funzione speciale aumenta lo spazio disponibile e riduce in modo considerevole i costi.

CR-7iA -con braccio di 717 mm- e CR-7iA/L -con braccio di 911- sono due formati abbastanza simili: uno con braccio standard ed uno con braccio lungo ma hanno la stessa capacità di carico (7kg). I vantaggi di ciascuna versione dipendono dalle esigenze del cliente: il braccio standard è ideale in caso di problemi di spazio, mentre il braccio lungo presenta una portata maggiore per ambienti di lavoro più ampi. In base alle esigenze del cliente è possibile programmarli per l'esecuzione di interi flussi di produzione che richiedono livelli di qualità affidabili e costanti. Le attività in questione spaziano dall'assemblaggio di piccoli componenti ad attività molto ripetitive come lo spostamento di oggetti da una parte all'altra. La versione con il braccio di 911 mm è la soluzione ideale per le applicazioni di pallettizzazione e asservimento delle macchine. Non occorrono barriere: la comprovata tecnologia del sensore integrato si arresta in automatico in caso di collisione con oggetti fissi o umani.

CR-35iA ha una capacità di carico di 35 kg, l'estensione del braccio fino a 1.813 mm e 6 assi di movimento che rendono la scelta ideale per svolgere tutti quei lavori ripetitivi che prevedono la movimentazione di carichi pesanti tipici dei settori automotive, packaging e distribuzione e lavorazione dei metalli. Grazie a carichi da 4 kg, 7 kg e 35 kg la serie CR è adatta per processi di movimentazione che espongono l'operatore umano al rischio di sforzi ripetitivi. Questa serie di robot collaborativi presenta benefici in termini di qualità e ripetibilità in quanto il robot, a causa della sua forza, risulta essere più preciso dell'operatore. La serie CR è fornita da un sistema di protezione chiamato *collision stop*. Lo stop avviene delicatamente dopo la collisione con persone o cose. Il sistema di sicurezza garantisce che il robot si fermi ad un massimo di 150 N, limite che può essere modificato via software.

Figura 2 - Modelli CR-7iA e CR-7iA/L

Applicazioni possibili sono:

- Assemblaggio: i modelli CR sollevano e posizionano componenti pesanti sulla linea di lavorazione e assemblaggio per prevenire il rischio di danni dovuti agli sforzi tra linea di assemblaggio e lavoratori.
- Ispezione di parti: la serie CR è adatta per questo tipo di applicazione in quanto è necessario frequentemente la presenza dell'operatore nelle fasi di carico e scarico delle parti ispezionate. Il robot gestisce i componenti e l'operatore il controllo qualità.
- Prelievo, confezionamento e pallettizzazione: sulle linee di confezionamento e pallettizzazione, la serie CR può essere usata per portare a termine attività di handling che possono essere noiose, ripetitive e sporche dando spazio agli operatori per attività più complesse e diversificate tra loro.
- Dispensing: se i cobot si usano per applicare sigilli, adesivi, vernice o altri fluidi.

1.2.2 Universal Robots UR3, UR5, UR10

I robot UR possono essere implementati praticamente in qualsiasi settore industriale, in qualsiasi processo e facilmente gestibili da parte di qualsiasi dipendente. I robot UR possono automatizzare dall'assemblaggio alla verniciatura, dall'avvitatura all'etichettatura, dal confezionamento alla lucidatura, dallo stampaggio a iniezione a qualsiasi altra operazione. Grazie alla loro flessibilità i robot UR sono anche economicamente adatti ai processi produttivi caratterizzati da piccoli lotti e mix di prodotto.
La gamma UR è composta da UR3, UR5 e UR10, così denominati in base alla rispettiva capacità di carico in kg, e tutti dotati di avanzate capacità collaborative. Per questo sono i cobot più utilizzati nelle linee di produzione e si possono così riassumere:

- UR3: Il più piccolo della gamma, UR3 è la scelta perfetta per compiti leggeri di assemblaggio e lavori che richiedono un'assoluta precisione. Con una rotazione di 360 gradi su tutti i giunti ed una rotazione infinita sull'articolazione finale, UR3 è il robot più piccolo, versatile e collaborativo attualmente sul mercato. Inoltre, automatizza operazioni fino a 3kg ed ha uno sbraccio fino a 500 mm.

- UR5: leggermente più grande è ideale per automatizzare operazioni con media portata, come manipolazione e collaudo. Questo robot è facile da programmare, rapido da installare e, proprio come gli altri robot collaborativi della gamma UR, offre uno dei periodi di recupero dell'investimento più rapidi del settore. Inoltre, automatizza operazioni fino a 5 kg ed ha uno sbraccio fino 850 mm.
- UR10: Il robot più grande nella gamma; garantisce elevata precisione e maggiore capacità di carico automatizzando operazioni con payload fino a 10 kg. Grazie ad un raggio orizzontale di 1300 mm, il robot UR10 è adatto in particolare al confezionamento, alla pallettizzazione, all'assemblaggio e manipolazione, laddove sia richiesta una maggiore area operativa.

1.2.3 YASKAWA Motoman HC 10

Lo Yaskawa Motoman HC10 (HC acronimo di Human Collaborative) è il primo robot collaborativo ad essere introdotto da Yaskawa al di fuori del Giappone. Il prototipo, con uno sbraccio di 1,2 m e con 10 kg di portata, è progettato per essere certificato secondo la specifica tecnica ISO TS15066. Il robot HC10 garantisce la sicurezza necessaria a diretto contatto con l'operatore mediante un sensore di forza/coppia sofisticato su ogni asse consentendo, così, l'interazione flessibile tra il braccio del robot e il suo ambiente. L'HC10 non richiede misure di protezione supplementari, per esempio una recinzione protettiva, risparmiando spazio e costi. L'installazione è estremamente flessibile e adatta ad una vasta tipologia di stazioni di lavoro. Oltre agli aspetti di sicurezza l'obiettivo principale nella progettazione del nuovo HC10 riguarda il funzionamento particolarmente user-friendly. La programmazione può essere infatti eseguita come *Easy teaching*, con la funzione manuale *Smart HUB*, ossia istruzione e programmazione precisa guidando il braccio robot con le mani. Il braccio robot è stato progettato per evitare urti pericolosi e in caso di arresto per un contatto, il robot può essere riattivato direttamente dal manipolatore stesso. Il passaggio dei cavi interni e il cablaggio, per esempio, garantiscono un elevato grado di affidabilità e design smussato.

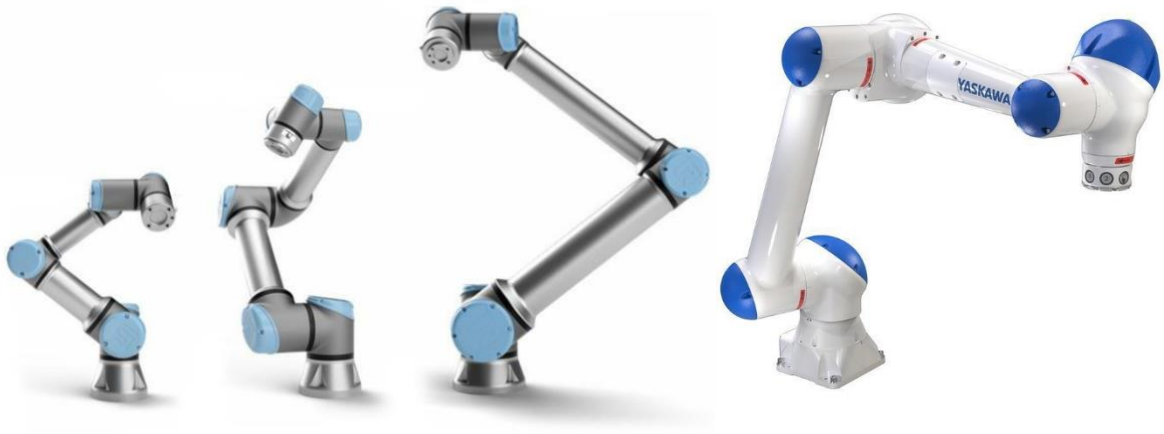


Figura 3 - A sinistra modelli UR3, UR5, UR10 di Universal Robot, a destra modello HC10 della Yaskawa

1.2.4 KUKA LBR iiwa

Quaranta anni dopo il primo utilizzo del robot industriale, KUKA si sta aprendo verso nuovi scenari nella storia industriale della robotica con *LBR iiwa*. *LBR* sta per "Leichtbauroboter", in tedesco robot leggero, e "iiwa" per *intelligent industrial work assistant*. *LBR iiwa* ha sensori di coppia ai giunti integrati in tutti e sette gli assi. Inoltre, è possibile trovare due modelli di robot collaborativi: LBR iiwa 7 R800 e *LBR iiwa 14 R820*. Il primo ha un peso di appena 23.9 kg ed una capacità di carico di 7 kg, il secondo invece ha un peso 29.9 kg ed una capacità di carico di 14 kg. Inoltre, i giunti e i gradi di libertà possono essere programmati individualmente. Grazie all'implementazione di tecnologie di sicurezza, questi robot rispondono alla minima forza esterna permettendo così una collisione sicura. In caso di contatto improvviso *LBR iiwa* riduce la sua velocità in un istante limitando la sua energia cinetica ad un valore tale da non provocare danni a persone o cose. Questo permette all'operatore di condividere spazi di lavoro e attività con il robot collaborativo. *LBR iiwa* è fatto interamente in alluminio essendo così più leggero e sicuro e grazie al suo design semplificato, senza spigoli, ha eliminato il pericolo di tagli. *LBR iiwa* gioca un ruolo fondamentale nello scenario della produzione futura in quanto si sta preparando per la massima personalizzazione del cliente. Allo stesso tempo esso è modulare e facile da programmare grazie ad una tecnologia Java poiché è il linguaggio di programmazione più usato nel mondo. Il basso peso del robot collaborativo permette di essere montato su di un carrello lavorando così in diverse aree, in base alle differenti specifiche agendo in maniera collaborativa o da solo.

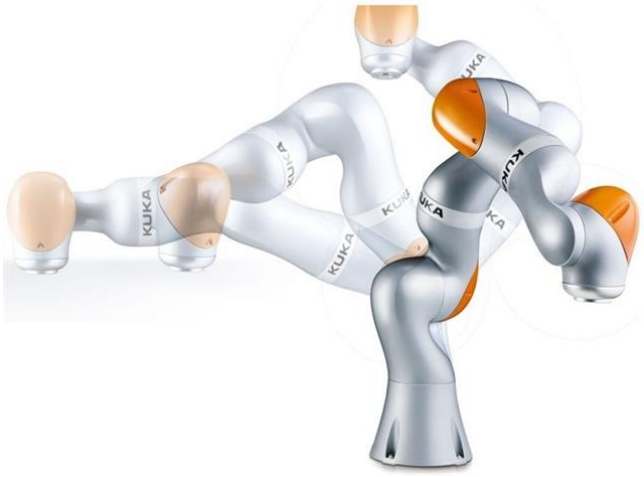


Figura 4 - Kuka LBRiiwa

1.2.5 RETHINK ROBOTICS Baxter e Sawyer

Baxter è stato il primo robot collaborativo con 2 bracci introdotto nel 2012 rivoluzionando così il mondo manifatturiero. Oggi molte industrie si affidano a Baxter per attività ripetitive di produzione ottenendo un significativo vantaggio competitivo nel loro business. Baxter è una soluzione per attività come il packing, kitting, loading e movimentazione dei materiali. Inoltre, lavora in maniera sicura e interattiva con l'operatore senza alcuna gabbia e può essere manualmente impostato in pochi minuti. Baxter è caratterizzato da 7 gradi di

libertà per braccio, ognuno con capacità massima di allungarsi di 1210 mm, e può essere sviluppato per lavorare su due attività indipendenti in maniera tale da poter operare simultaneamente sulla stessa attività massimizzandone il risultato. Le competenze e le performance di Baxter sono in continuo sviluppo grazie ad una piattaforma chiamata "Intera", la quale permette di scaricare software aggiornabili permettendo così di accedere all'ultima funzionalità disponibile.
Sawyer invece è una versione riservata per attività che richiedono più precisione, come ad esempio nel testare circuiti di bordo stampati con precisione di 0,1 mm. Inoltre, possiede una capacità di allungarsi di 1260 mm e 7 gradi libertà favorendo la sua operatività in stazioni di lavoro strette. L'abilità di Sawyer è proprio quella di lavorare in qualsiasi spazio che presenta dei limiti fisici per il braccio. Infatti, ogni giunto del cobot, eccetto il polso, può ruotare massimo di 350 gradi. Inoltre, la testa di Sawyer è un display LCD dove si può guardare menù, espressione degli occhi, modificare schermate, etc. Contiene anche una videocamera e luci che comunicano le condizioni del cobot e può essere spostata lungo lo stesso giunto/asse dove si sta muovendo con una rotazione massima di 350 gradi.

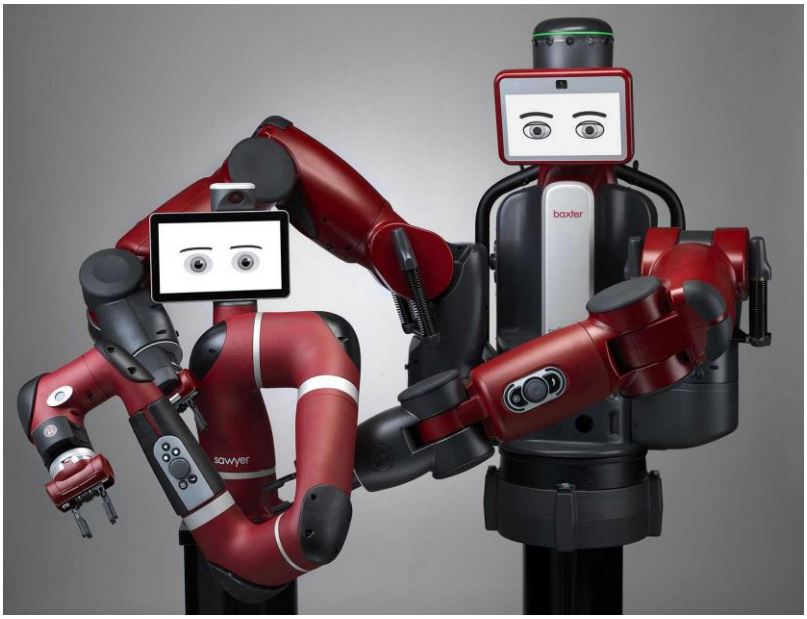


Figura 5 - Modelli Sawyer eBaxter della Rethink

1.2.6 Comau AURA Project

Il progetto presentato da Comau, Advanced Use Robot Arm (AURA), è l'integrazione di diverse tecnologie in quanto diventa possibile garantire la necessaria sicurezza per una reale e costante cooperazione tra macchina e uomo. AURA, infatti, è una tipologia di robot ricoperto da una speciale pelle protettiva sotto la quale coesistono sensori di forza, percettivi e di torsione. Il sistema cooperativo sfrutta inoltre l'integrazione di un sistema di visione e di laser scanner. È proprio la combinazione di tutta questa tecnologia il fattore abilitante che consente agli operatori di muoversi in totale sicurezza nello spazio di lavoro condiviso con il robot. Questo fa sì che si possa guidare l'azione in modo diretto, con la guida manuale, o facendo mutare al robot la traiettoria

semplicemente toccandolo oppure in modo indiretto grazie all'interazione con la sensoristica applicata al robot e all'intera area di lavoro. Esattamente come accade tra persone, i robot collaborativi di oggi recepiscono cosa accade nell'area accanto e si comportano di conseguenza. Se fino ad oggi le soluzioni di robotica collaborativa hanno riguardato macchine di piccola taglia e con basso payload, ora la soluzione AURA consente la trasformazione di un robot "tradizionale" in robot collaborativo anche per macchine ad elevato payload, in grado quindi di applicarsi ad attività pesanti. Un esempio su tutti sono le manipolazioni che avvengono nell'assemblaggio finale di un'automobile, come il posizionamento del cruscotto, dei sedili o delle batterie. Nulla vieta, tuttavia, l'applicazione in altri contesti come piegatura, altre lavorazioni meccaniche o altri asservimenti. Dipende dalla configurazione richiesta e dal livello di integrazione/cooperazione necessario da attivare. Ogni operazione svolta avviene inoltre senza l'ausilio di alcuna barriera protettiva e questo significa un forte risparmio di costi in termini di hardware ridondante e di layout in fabbrica (*Logisticanews*, 2016).

L'idea di COMAU è iniziata considerando che i loro clienti non potevano rinunciare ai vantaggi dei robot industriali, come ad esempio alti payload e flessibilità. Il principale vantaggio della produzione di massa sarà quello di avere la possibilità di comprare un robot che potrebbe essere considerato collaborativo e non collaborativo allo stesso tempo, secondo le attività richieste durante il ciclo di produzione. Tutto questo è garantito da un sistema di Vision/Laser che rileva la presenza umana in prossimità del cobot mandando un segnale al robot con lo scopo di cambiare lo stato della macchina da standard a collaborativo. "The SKIN" è stata sviluppata con la collaborazione dell'Università Sant'Anna di Pisa e le sue peculiarità sono:

- Capacitive Sensor Layer: rileva la prossimità dell'operatore
- Piezo-electric Sensor Layer: rileva il contatto con l'operatore
- Maggiore ergonomia
- Grande modularità
- Tecnologia brevettata
- Facile da riparare

Il robot standard di COMAU ha la possibilità di lavorare in tre stati differenti:

- Program Mode
- Automatic Mode
- Remote Mode

Sul robot AURA, progettato iniziando da un robot tradizionale, sono ancora presenti gli stati standard ma Automatic e Remote Modes sono suddivsi in altri tre stati non selezionabili. Quando il robot è nello stato Automatic o Remote, il laser scanner manda un input al robot che dipende dalla presenza umana a causa di uno degli stati selezionati:

- Standard Speed: il Laser scanner non rileva nessuna presenza umana.
- Collaborative Speed: il laser scanner rileva la presenza umana e automaticamente riduce la velocità del robot. Il sistema calcola il tempo necessario per raggiungere la velocità collaborativa e se questo tempo non è sufficiente, ferma il robot attraverso la modalità Drive-off. Il riavvio sarà possibile solo premendo l'interruttore dell'armadietto del robot.

- Emergency stop:
    1. Automatic Restart: il sensore AURA SKIN Proximity rileva la presenza umana e ferma i movimenti del robot. Successivamente riprende in modo automatico il movimento dopo un certo tempo.
    2. Manual Restart: i sensori AURA SKIN Proximity and Contact rilevano il contatto umano soft e fermano il robot. Il robot continua il movimento precedente solo quando l'operatore preme il bottone reset sul robot.
    3. Drive Off: i sensori AURA SKIN Proximity and Contact rilevano un forte contatto con l'operatore e ferma immediatamente il robot. Il restart è possibile solo con il Drive-On dato sul teach pendant o su altri dispositivi fuori dalla zona di lavoro del robot

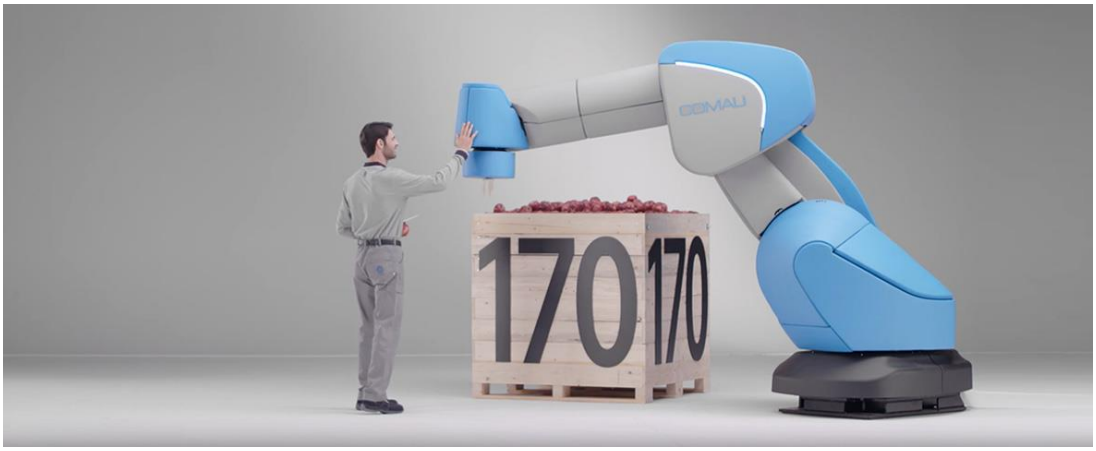


Figura 6 - Aura, il robot collaborativo di Comau

1.2.7 ABB IRB 14050, detto Single Arm YuMi

A seguito del successo del braccio a due bracci YuMi di ABB, introdotto nel 2015 e considerato il primo robot veramente collaborativo al mondo, ABB ha sviluppato lo YuMi a braccio singolo per espandere il proprio portafoglio collaborativo. È stato presentato nel giugno del 2018 e disponibile da novembre.
YuMi a braccio singolo è compatto e leggero (9,5 kg) e supporta il montaggio in qualsiasi direzione, inclusi il soffitto, il tavolo e il montaggio a parete per un'installazione rapida e flessibile capace di adattarsi alle linee di produzione esistenti. Il braccio in magnesio ultraleggero ruota su sette assi per imitare i movimenti umani con maggiore agilità rispetto ai robot a 6 assi. Il robot è stato specificamente progettato per soddisfare le esigenze di produzione flessibili richieste dai processi di assemblaggio di componenti di piccole dimensioni, tra cui elettronica di consumo, beni di consumo e piccole e medie imprese.
YuMi a braccio singolo offre anche la stessa programmazione intuitiva e di facile utilizzo attraverso la programmazione YuMi a doppio braccio, in modo che i lavoratori possano insegnare i movimenti e le posizioni del robot in modo semplice e rapido semplificando notevolmente la programmazione del robot. Combinando questa semplicità con la flessibilità di implementazione del robot, aiuterà i produttori in molti settori a

compensare la carenza di lavoratori qualificati e contribuire a ridurre le barriere all'ingresso per i nuovi potenziali utenti di robot, in particolare le piccole e medie imprese.

Estremamente versatile, la famiglia di robot YuMi può essere combinata in numerose configurazioni. Ad esempio, lo YuMi a braccio singolo può essere utilizzato per alimentare parti su YuMi a doppio braccio in modo da aumentare la flessibilità oppure venire aggiunto come braccio supplementare per operazioni di assemblaggio più complesse che richiedono più di 2 bracci robotici.

- Il controller esterno YuMi a braccio singolo consente al braccio del robot di avere un ingombro molto ridotto per adattarsi a spazi ristretti;
- Il peso ridotto e la possibilità di montare lo YuMi a braccio singolo con qualsiasi angolazione lo rendono ancora più flessibile e agile.

Differenze con lo ABB 14000 Yumi Dual-Arm

- Il controller esterno YuMi a braccio singolo consente al braccio del robot di avere un ingombro molto ridotto per adattarsi a spazi ristretti;
- Il peso ridotto e la possibilità di montare lo YuMi a braccio singolo con qualsiasi angolazione lo rendono ancora più flessibile e agile.

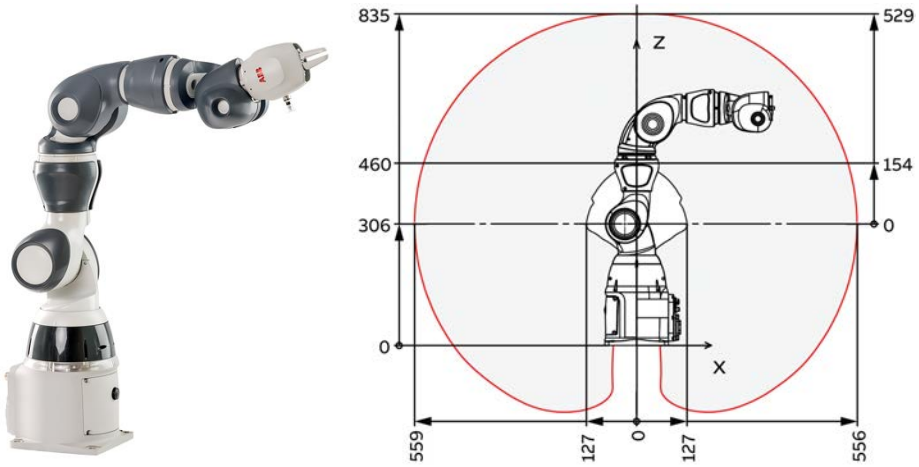


Figura 7 - ABB IRB14050 Single Arm Yumi e area di lavoro

1.2.8 Sintesi delle caratteristiche dei cobots

Nei paragrafi precedenti si è andati ad analizzare i principali cobot presenti sul mercato riportando casi reali di applicazioni collaborative. In questo paragrafo invece si proporrà una suddivisione dei diversi modelli affrontati precedentemente attraverso una sintesi in base alle diverse caratteristiche tecniche:

- Numeri di assi
- Capacità di carico
- Estensione del braccio
- Peso
- Velocità
- Visione e sensori e altre caratteristiche

Tabella 1 -Confronto caratteristiche dei cobot attualmente in commercio

	Numero assi	**Carico (kg)**	**Braccio (mm)**	**Peso (kg)**	**Velocità**	**Applicazioni**	**Visione e sensori**
ABB Yumi	2 bracci, 7 assi per braccio	0.5 per braccio	559	38	1500 mm/s	Assemblaggio di piccole parti	Sistema di visione su end effector e sensori di coppia
ABB Yumi Single Arm	1 bracci, 7 assi	0.5	559	9,5	1500 mm/s	Assemblaggio di piccole parti	
Fanuc CR-4iA	1 braccio, 6 assi	4	550	48	TCP: 500/1000 mm/s	Controllo qualità	imbottitura esterna per monitorare l'impatto e bottone di riavvio rapido
FANUC CR-7iA		7	717	53		calzature sportive, pallettizzazione e asservimento macchine	
FANUC CR-7iA/L		7	911	55		pallettizzazione e asservimento macchine	
FANUC CR-35iA		35	1813	990		automotive, confezionamento e lavorazione metalli	
UR3	1 braccio, 6 assi	3	500	11	TCP: 1m/s	saldatura, incollaggio, avvitatura, prelievo e posizionamento	stop del braccio in caso di collisione
UR5		5	850	18,4			
UR10		10	1300	28,9			
YASKAWA Motoman HC 10	1 braccio, 6 assi	10	1200	47	giunti: da 130°/s a 250°/s	assemblaggio, prelievo e posizionamento, controllo qualità	doppio sensore di coppia per giunto

KUKA LBR iiwa	1 braccio, 7 assi	7	911	22,3	giunti: 90°/s a 180°/s;	assemblaggio, confezionamento, prelievo e posizionamento	posizione integrata e coppia di sensori in ogni giunto
		14	931	29,5	70°/s a 180°/		
BAXTER	2 bracci, 7 assi	2,3	1041,4	75	TCP: 1m/s	prelievo e posizionamento, kitting, confezionamento	1 camera integrata per braccio, sensori di forza integrati, 1 camera frontale per rilevamento umano
SAWYER	1 braccio, 7 assi	4	1026	19	1m/s senza capacità di carico; 0,6 m/s con	test di circuiti stampati, movimentazione materiali,	
COMAU RACER3	1 braccio, 6 assi	3	630	30	giunti: da 430°/s a 900°/s	assemblaggio, sigillatura, confezionamento, lucidatura	tecnologia SKIN, sensori di forza e di coppia
COMAU NJ 60 **NJ 110**	1 braccio, 6 assi	60	2258	645	giunti: da 170°/s a 340°/s;	saldatura, assemblaggio, sigillatura, confezionamento, lucidatura, testing	
		110	2980	1070	da 110°/s a 230°/s		
COMAU AURA	1 braccio, 6 assi	170	2800	1615	Massima velocità 2000 mm/s Max Velocità Collaborativa 500 mm/s		

CAPITOLO 2

IRB 14000 YUMI IL ROBOT COLLABORATIVO DI ABB

2.1 Descrizione ABB IRB 14000: Yumi

Al giorno d'oggi, in molti contest aziendali sviluppati, è possibile trovare robot collaborativi in grado di aiutare l'operatore nelle mansioni sostituendolo anche in movimenti ripetitivi e ad alte velocità.
In questo capitolo andremo ad analizzare, in particola modo, un cobot prodotto da ABB: lo IRB14000, detto Yumi, presentato alla fiera delle tecnologie industriali Hannover Messe nell'aprile del 2015.
Il suo nome deriva dalla filosofia di pensiero con cui l'azienda svedese ha voluto realizzare il robot. L'idea è la collaborazione tra uomo e robot, concepito anch'esso come parte fondamentale del lavoro, in grado di lavorare gomito a gomito con l'uomo: il nome non è altro che l'abbreviazione di You and Me.
Lo Yumi non è un robot di grandi dimensioni; la sua particolarità è che fa parte della prima generazione di robot industriali a due braccia, con sette gradi di libertà di braccio.
È stato concepito con l'obiettivo di rivoluzionare il modo dell'assemblaggio automatico poiché è in grado di svolgere le stesse mansioni di un operatore; con la precisione tale da poter inserire un filo nella cruna di un ago e con la massima sicurezza per tutte le persone che operano nell'area.
YuMi è un robot pensato per l'assemblaggio di piccoli pezzi dotato di mani flessibili, sistemi per l'alimentazione dei pezzi, telecamera per il riconoscimento delle parti e controllo avanzato. Può lavorare al fianco di operatori in carne e ossa, in un normale ambiente produttivo consentendo così alle aziende di trarre il meglio dall'interazione fra uomo e robot. Una delle caratteristiche esclusive di YuMi è la sua classificazione di "sicurezza intrinseca" che indica come il robot possa lavorare "gomito a gomito" con gli operatori, senza alcun rischio per la loro sicurezza.
YuMi è stato progettato specificamente per le esigenze di assemblaggio di piccole parti, in particolare nel settore dell'elettronica di consumo. Una delle sue peculiarità è che il robot è parte integrante di una soluzione completa per l'assemblaggio di piccoli pezzi, che comprende mani adattabili, alimentatori flessibili, sensori per il controllo della forza, guida con sistemi di visione oltre che a sistemi di controllo e software avanzati.
YuMi è veloce ma rispetta le norme di sicurezza previste per i robot "intrinsecamente sicuri". Può svolgere i movimenti richiesti per l'assemblaggio di piccoli pezzi in spazi molto ristretti mantenendo, al tempo stesso, un raggio d'azione simile a un addetto umano. Questo aspetto è fondamentale per contenere l'ingombro sull'impianto e consentire l'installazione di YuMi anche su postazioni di lavoro attualmente utilizzate solo da addetti umani.
La casa costruttrice ABB ha cercato di creare un cobot che rispondesse a questi punti:

- Sicurezza intrinseca
- Parti di alimentazione flessibili gestione
- Assemblaggio guidato dalla visione

- La migliore precisione della categoria
- Assemblaggio rapido ed efficace

YuMi è stato progettato specificamente per le esigenze di flessibilità e agilità in produzione dell'industria elettronica di largo consumo ma può essere impiegato in qualsiasi processo di assemblaggio di piccoli componenti, grazie al doppio braccio di magnesio che flette su 7 assi, alle "mani" flessibili, al sistema universale per l'alimentazione dei componenti, alla telecamera per l'individuazione dei pezzi e al controllo di movimento avanzato ad alta precisione.

Il robot ha uno scheletro di magnesio leggero ma estremamente rigido, rivestito da un involucro di plastica con morbide imbottiture per attutire eventuali colpi. YuMi è compatto, come si può vedere dalle immagini indicanti le dimensioni, in grado di compiere movimenti simili a quelli dell'uomo, per trasmettere un senso di sicurezza e tranquillità ai suoi colleghi in carne e ossa.

L'area di lavoro dello Yumi è pressoché simile a quella di un operatore umano, come si può vedere dalle immagini in cui sono riportate le dimensioni in millimetri.

Le dimensioni dello Yumi non sono grandi come i robot manipolatori o pallettizzatori, inerzia e massa sono molto piccole rispetto ad altri robot antropomorfi per cui in caso di scontro con operatori umani le forze in gioco sono minori.

Quindi un primo vantaggio è la possibilità di sostituire, nella postazione, il robot senza dover modificare la disposizione ed essendo un robot di tipo collaborativo non ha bisogno di protezioni o barriere di sicurezza.

Infatti, i due bracci contengono sensori ad effetto hall: lo scontro produce una forza che i sensori rilevano inviando un segnale al sistema di supervisione che andrà a bloccare i movimenti di entrambi i bracci del cobot.

Molto importante è anche la scelta da parte di ABB di creare i link molto arrotondati e rivestiti da imbottiture morbide, rendendo così difficile rimanere impigliati e non sussiste alcun rischio di lesioni dovute all'apertura e chiusura degli assi. Le imbottiture inoltre riducono anche le forze scambiate con gli ostacoli esterni e risultano utile poiché fungono anche da protezione per il cobot.

Quando YuMi si imbatte in un ostacolo imprevisto, ad esempio il contatto con un addetto, si arresta nel giro di pochi millisecondi, dopodiché il suo funzionamento può essere ripristinato facilmente premendo il pulsante play sulla flex pendant.

Figura 8 - ABB YuMi IRB 14000 YuMi

2.2 Caratteristiche Tecniche

Yumi è un robot di dimensioni ridotte, come si può osservare dalle immagini. L'apertura dei bracci è limitata in quanto è stato pensato per l'assemblaggio di componentistica elettronica. Risulta più facile per gli operatori lavorare vicino a questo robot senza aver bisogno di protezioni perché dispone di molte funzioni di sicurezza oltre che sistemi di protezioni per gli operatori.

Il cobot è composto da:

- Base, dove è presente alimentazione con un banco di condensatori e sistema di ventilazione;
- Controller, contenuto all'interno della scocca;
- Base dei bracci;
- Due Bracci ognuno formato da 7 moduli molto simili tra loro che danno la possibilità di arrivare a 7 gradi di mobilità.

Robot Specifications		Robot Motion Speed		Robot Motion Range	
Axes:	7	J1	180 °/s (3.14 rad/s)	J1	±168.5°
Payload:	1.00kg	J2	180 °/s (3.14 rad/s)	J2	+43.5° - 143.5°
H-Reach:	500.00mm	J3	180 °/s (3.14 rad/s)	J3	+80° - 123.5°
Repeatability:	±0.0200mm	J4	400 °/s (6.98 rad/s)	J4	±290°
Robot Mass:	38.00kg	J5	400 °/s (6.98 rad/s)	J5	+138° - 88°
Structure:	Articulated	J6	400 °/s (6.98 rad/s)	J6	±229°
Mounting:	Floor				

Tabella 1 - Tabella riassuntiva delle caratteristiche tecniche, velocità e range di movimento dell'IRB 14000

Il braccio contiene:

- Motori
- Blocchi meccanici per limitare le rotazioni degli assi
- Sensori ad effetto hall su assi 1,2,7,3,4 che permettono la calibrazione
- Imbottiture e protezioni

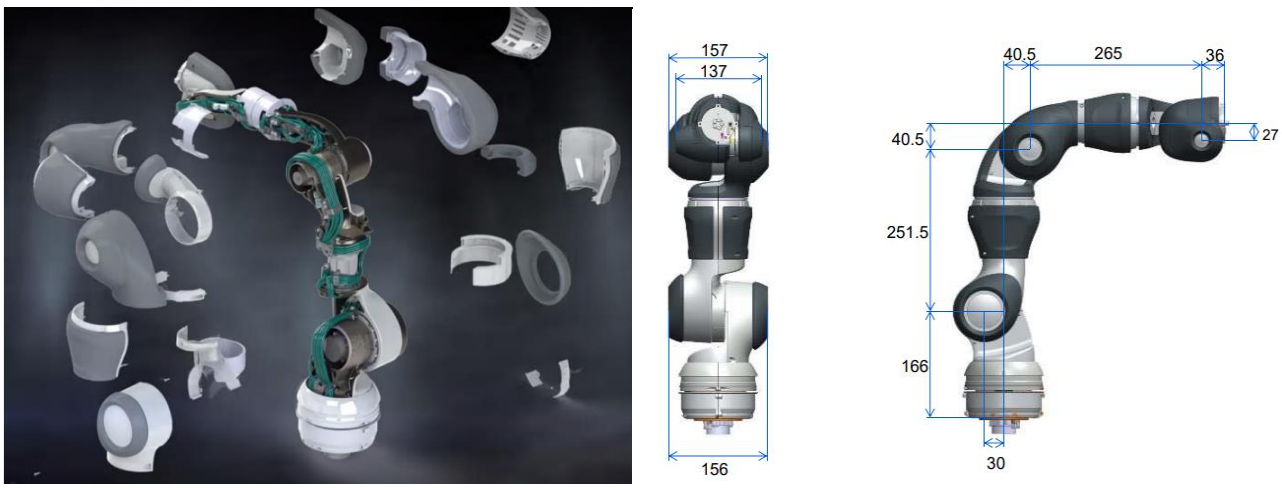


Figura 9 -Esploso del braccio IRB 14000 e dimensioni dei link

Figura 10 - Alcuni componenti dei bracci, in ordine: motore, imbottitura e blocco meccanico

2.3 Sicurezza ed ergonomia per gli operatori

La sicurezza è uno dei requisiti fondamentali che le macchine automatiche devono avere oggigiorno, per questo motivo molto è il lavoro e la ricerca in questo ambito durante la progettazione di qualsiasi macchinario. Un robot in movimento costituisce una macchina potenzialmente letale. Durante la corsa, il robot può eseguire dei movimenti imprevisti e a volte irrazionali. Inoltre, tutti i movimenti vengono effettuati con notevole forza e possono ferire gravemente il personale e/o danneggiare le attrezzature che si trovano entro la portata operativa del robot. L'ambiente di lavoro deve essere sicuro sia per gli operatori umani che lavorano sia per i macchinari e i robot. Per soddisfare questo requisito ABB ha sviluppato i bracci dello YuMi con forme molto arrotondate evitando di avere spigoli vivi e impedendo che l'operatore possa rimanere impigliato o viceversa.

A differenza degli altri robot, anche di quelli collaborativi, la struttura dei bracci dello YuMi è ricoperta da componenti in plastica e da imbottiture per limitare le forze scambiate con oggetti esterni e in particolare con operatori che entrano nel campo di lavoro del cobot. Il cobot è stato realizzato in modo che il contatto con l'operatore risulti innocuo[3], come previsto dalla norma ISO/TS 12100[4] del 2016, per cui la limitazione di forza e velocità è intrinseca: infatti la massima velocità è di 1500mm/s in confronto ad altri robot che arrivano a 5000 mm/s.

Per migliorare la sicurezza vengono controllati i parametri di forza e velocità tramite i sensori, descritti nel prossimo paragrafo, che sono installati all'interno di ogni braccio.

Un altro requisito è l'ergonomia e fa parte sempre della sicurezza per l'operatore. Il design è essenzialmente un compromesso tra esigenze biologiche e fisiologiche umane, come determinato dalle linee guida dell'ergonomia, e dei requisiti fisici di sicurezza della workstation in termini di dimensioni, funzioni, frequenza e metodi di utilizzo delle singole attrezzature o delle interazioni con i robot.

[3] Per contatto innocuo si intende che la pressione che viene esercita dal robot su un corpo esterno deve essere inferiore a dei valori limite. In figura vengono indicate alcune forze e pressioni massime che si possono scambiare.

[4] Questa norma va ad integrare le norme relative alla sicurezza per macchinari e robot esistenti quali la ISO 12100 contenente la normativa generale per la valutazione del rischio con dei macchinari e la ISO 10218-2 con la specifica per le applicazioni robotizzate. Nel 2016 è stata rilasciata una norma ISO relativa ai robot collaborativi la 15066, dove vengono definiti alcune criteri che i cobot devono rispettare per essere tali: tra questi uno dei più importanti è quella relativa al contatto e il suo comportamento quando avviene tale evento.

I moduli che compongono il braccio del cobot sono adatti alla presa dell'operatore e mantengono lo spazio per la mano che guiderà il braccio dello YuMi, senza il rischio di subire schiacciamenti. La presa risulta confortevole perché la dimensione del braccio è stata pensata per la mano dell'uomo per cui non risulta difficile movimentare il cobot. La modalità manuale permette di disinserire i freni che bloccano i giunti e poiché i giunti hanno bassissime forze di attrito, dovute al peso ridotto dei componenti, l'operatore può movimentare il braccio.

Tabella 2 - Tabella dei limiti biomeccanici

Body Region	Specific Body Area		Quasi-Static Contact		Transient Contact	
			Maximum Allowable Pressure (p_{max}) [N/cm²] (see NOTE 1)	Maximum Allowable Force (F_{max}) [N] (see NOTE 2)	Maximum Allowable Pressure Multiplier P_T (see NOTE 3)	Maximum Allowable Force Multiplier F_T (see NOTE 3)
Skull and forehead	1	Middle of forehead	130	130	N/A	N/A
	2	Temple	110		N/A	
Face	3	Masticatory muscle	110	65	N/A	N/A
Neck	4	Neck muscle	140	150	2	2
	5	Seventh neck vertebra	210		2	
Back and shoulders	6	Shoulder joint	160	210	2	2
	7	Fifth lumbar vertebra	210		2	2
Chest	8	Sternum	120	140	2	2
	9	Pectoral muscle	170		2	
Abdomen	10	Abdominal muscle	140	110	2	2
Pelvis	11	Pelvic bone	210	180	2	2
Upper arms & elbow joints	12	Deltoid muscle	190	150	2	2
	13	Humerus	220		2	
Lower arms and wrist joints	14	Radial bone	190	160	2	2
	15	Forearm muscle	180		2	
	16	Arm nerve	190		2	
Hands and fingers	17	Forefinger pad D	300	140	2	2
	18	Forefinger pad ND	270		2	
	19	Forefinger end joint D	280		2	
	20	Forefinger end joint ND	220		2	
	21	Thenar eminence	200		2	
	22	Palm D	260		2	
	23	Palm ND	260		2	
	24	Back of the hand D	200		2	
	25	Back of the hand ND	190		2	
Thighs and knees	26	Thigh muscle	250	220	2	2
	27	Kneecap	220		2	
Lower legs	28	Middle of shin	220	130	2	2
	29	Calf muscle	210		2	

Note 1: Limits from pain onset level studies, and represent the 75th percentile of the range of recorded values for a specific body area
Note 2: From literature study of 188 sources and based on body regions, not on specific areas. Could lead to minor injuries 1 on AIS.
Note 3: From literature sources, transient limits can be at least twice as quasi-static values for force and pressure

2.4 Sensori e calibrazione del cobot

All'interno del braccio sono presenti dei sensori di coppia e dei sensori ad effetto hall su ogni giunto del braccio: i sensori di coppia servono a controllare le coppie scambiate in ogni giunto, mentre i sensori ad effetto hall servono per la calibrazione e l'aggiornamento dei contagiri.

2.4.1 Sensori ad Effetto Hall e calibrazione del cobot

I sensori ad *effetto hall* sono posizionati sui giunti: vengono utilizzati per poter aggiornare la posizione dei contagiri e poter calibrare il cobot. Come per ogni robot c'è una posizione di calibrazione: i giunti hanno delle tacche che indicano la posizione corretta poiché indicano dove si trovano i sensori. Le tacche o contrassegni indicano la posizione che è stata identificata come zero per la calibratura. Nella tabella riportata qui sotto sono indicati i valori in gradi dei vari assi.

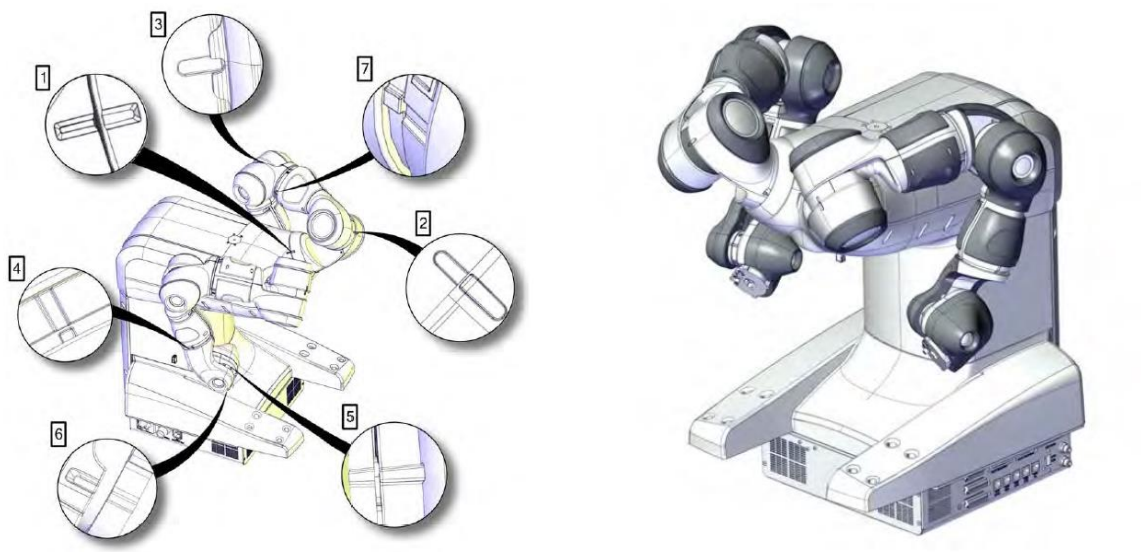


Figura 11 - La figura a sinistra mostra i contrassegni utili per la calibrazione del cobot. Il cobot si trova in posizione di calibratura, come è visibile, in modo migliore, nella figura a destra

L'effetto Hall[5] mette in relazione la tensione tra due punti di un materiale conduttore o semiconduttore con un campo magnetico attraverso il materiale. Se usati da soli, i sensori a effetto Hall possono rilevare soltanto oggetti magnetizzati. Invece, se usati in associazione a un magnete permanente come quello mostrato nella Figura, essi sono in grado di rilevare tutti i materiali ferromagnetici.

Tabella 3 - Tabella indicante i gradi per ogni asse nella corretta posizione di calibrazione per i bracci dello YuMi

Asse	IRB 14000 ROB_R	IRB 14000 ROB_L
1	0°	0°
2	-130°	-130°
3	30°	30°
4	0°	0°
5	40°	40°
6	0°	0°
7	-135°	135°

In questo modo un dispositivo a effetto Hall capta un forte campo magnetico in assenza di un metallo ferromagnetico nelle immediate vicinanze. Quando questo materiale è portato in stretta prossimità del dispositivo, il campo magnetico si affievolisce in corrispondenza del sensore a causa della curvatura delle linee di campo attraverso il materiale. Questa CAD uta di tensione è la chiave per la percezione della prossimità con i sensori a effetto Hall. Le decisioni binarie riguardanti la presenza di un oggetto vengono prese

[5] L'effetto Hall è un fenomeno elettromagnetico: su un conduttore si può formare una differenza di potenziale sulle facce opposte del medesimo, detto potenziale di Hall, dovuto a un campo magnetico perpendicolare alla corrente elettrica che scorre in esso. Gli elettroni che si muovono nel conduttore sono soggetti ad un Forza di Lorentz, definita come $\vec{F} = q\vec{v} * \vec{B}$, da cui è possibile ricavare la tensione di Hall calcolabile con la seguente $\Delta V = \frac{iB}{nqd}$, dove n indica il numero di cariche, q la carica elettronica, d spessore del conduttore, i la corrente e B il campo magnetico e v la velocità.

confrontando la tensione del sensore con una soglia. È importante notare che utilizzando un semiconduttore, per esempio silicio, si hanno diversi vantaggi in termini di dimensione, robustezza e immunità all'interferenza elettrica. Inoltre, l'uso dei materiali semiconduttori permette la costruzione di circuiti elettronici per l'amplificazione e il rilevamento direttamente sul sensore stesso, riducendone così le dimensioni e il costo.

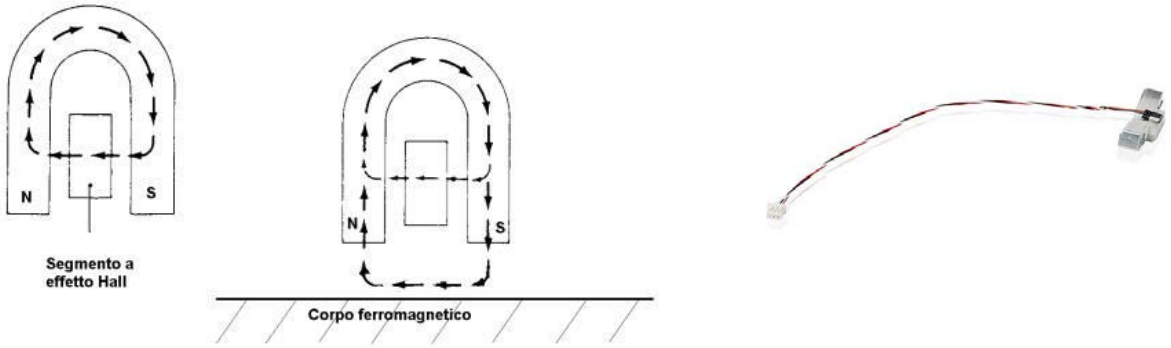


Figura 12 - A sinistra funzionamento di massima del sensore ad effetto Hall.

All'interno del controller c'è una memoria dove vengono salvati i dati di calibrazione e dove vengono memorizzate le posizioni di arresto del robot. Questa memoria rimane alimentata tramite delle batterie, in questo modo è possibile spegnere o riavviare il controller senza perdere informazioni preziose quali la calibrazione e le posizioni di arresto del cobot.

Una volta posizionati i link dei bracci nella corretta posizione, utilizzando i contrassegni, si possono avviare, tramite il tastierino del cobot, delle funzioni preimpostate tra cui la *calibrazione fine* e l'*aggiornamento dei contagiri.* Per eseguire le routine di calibrazione segue l'ordine numerico crescente degli assi partendo, quindi, dal primo e arrivando al settimo.

È importante che i giri siano sempre aggiornati: infatti all'interno del controller sono presenti due batterie che vanno ad alimentare il circuito di memoria dove sono salvati i dati di aggiornamento dei giri. L'importanza di questo fattore è legata alla sicurezza del robot; infatti nel caso in cui si voglia muovere il robot senza aver l'aggiornamento dei giri aggiornato si rischia di danneggiare i giunti e in particolare i blocchi meccanici che limitano la rotazione dei giunti. È possibile anche danneggiare oggetti esterni perché se i contagiri non vengono aggiornati, o mancano i dati, il cobot cercherà di seguire le traiettorie definite nelle routine sviluppate dall'utente; quindi c'è la possibilità di danneggiare oggetti uscendo dall'area di lavoro pensata per l'applicazione.

Per *calibrazione fine o absolute accuracy* si intende la misura che viene eseguita sulla posizione dei link del cobot: con questa funzione si può migliorare la precisione della misura dell'angolo di ogni asse del braccio così da poter eseguire i movimenti ripetitivi con una ripetitività elevata. Questa funzione esegue dei cicli, per ogni giunto, sfruttando il sensore ad effetto hall: il cobot inizia ad oscillare l'asse intorno a delle posizioni. Il controller andrà a verificare i valori misurati dal sensore con dei valori soglia definiti e finché i valori misurati non sono inferiori ai valori di soglia il controller fa oscillare il giunto. La sequenza che viene eseguita è:

1. oscilla nell'intorno positivo dell'asse

2. oscilla nell'intorno negativo dell'asse
3. oscilla nell'intero intorno del contrassegno
4. riduce l'intorno del contrassegno e riesegue alcune oscillazioni

In questo modo il sensore ad effetto hall ottiene i dati e li confronta con quelli salvati con la prima calibrazione eseguita dalla casa costruttrice. Con questa routine si può arrivare ad una precisione di 0,01° gradi. Oltre a collocare il manipolatore nella posizione iniziale, la calibratura *absolute accuracy* compensa anche:

• le tolleranze meccaniche nella struttura del robot

• la flessione causata dal carico

Quindi si concentra sulla precisione di posizionamento nel sistema di coordinate cartesiane del robot. Per riottenere nuovamente prestazioni *absolute accuracy* al 100%, il manipolatore deve essere ricalibrato per *absolute accuracy*: risulta utile eseguire questa routine quando si installano dei tool nel polso. Questo è possibile se è installato nel controller la scheda *absolute accuracy* che verrà illustrata successivamente.

L'*aggiornamento dei giri o calibrazione standard* è una routine che esegue delle semplici oscillazioni nell'intorno del contrassegno dell'asse. A differenza della calibrazione fine, l'aggiornamento dei giri controlla la posizione attuale del robot tendando di trovare la posizione definita del sensore, andando ad eseguire solo i punti 3 e 4 della lista precedente.

2.5 Polso del braccio e grippers

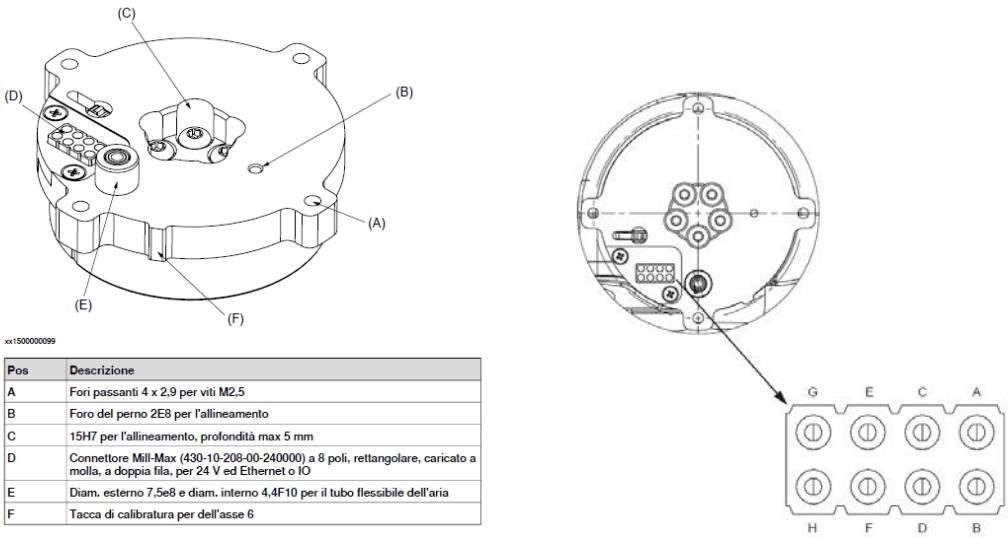
Il TCP, abbreviazione per Tool Center Point, è il centro del polso di ogni braccio dello Yumi nel quale viene applicato il tool. Il TCP è il punto centrale del polso del robot ed è uno dei sistemi fondamentali per l'orientazione e la definizione di offset e per la corretta orientazione del sistema. Il polso prevede un sistema di riferimento definito come *tool0* cioè un sistema di riferimento definito con il punto d'origine e una terna di assi cartesiani destrorsi con asse z uscente dal polso. Il TCP è il punto che il robot muove per posizionarsi nel target: quindi risulta importante definire, per ogni utensile o pinza applicata, un ulteriore TCP[6] oppure modificare il TCP aggiungendo un determinato offset[7].

È possibile utilizzare le pinze standard che ABB ha realizzato per il cobot oppure si possono realizzare delle pinze apposite per le operazioni che deve eseguire.

Esistono diverse tipologie di gripper perché il polso è stato pensato per collegare svariati utensili. Come si può vedere nella figura la flangia ha la possibilità di facilitare la comunicazione di segnali tramite il connettore Mill-Max e grazie all'attacco per l'aria è possibile utilizzare valvole senza dover portare una tubazione esterna al braccio. All'interno del braccio, come visto nei paragrafi precedenti, è predisposto il collegamento per l'aria. Questo fa in modo che il cobot possa utilizzare tutte le configurazioni di braccio senza correre il rischio di attorcigliare elementi esterni.

[6] È possibile definire vari TCP per ogni robot o per ogni tool che il robot utilizza.

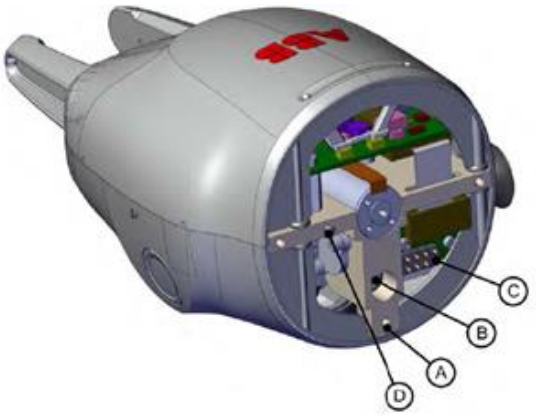
[7] Gli offset possono essere definiti lungo le direzioni del sistema cartesiano oppure come rotazioni.



Pos	Descrizione
A	Fori passanti 4 x 2,9 per viti M2,5
B	Foro del perno 2E8 per l'allineamento
C	15H7 per l'allineamento, profondità max 5 mm
D	Connettore Mill-Max (430-10-208-00-240000) a 8 poli, rettangolare, caricato a molla, a doppia fila, per 24 V ed Ethernet o IO
E	Diam. esterno 7,5e8 e diam. interno 4,4F10 per il tubo flessibile dell'aria
F	Tacca di calibratura per dell'asse 6

Figura 13 - Vista dettagliata della flangia e dei pin

La pinza progettata da ABB è quindi un utensile intelligente e multifunzionale per la gestione e l'assemblaggio di parti. La pinza ha un modulo servoassistito di base e due moduli funzionali, aspirazione e visione. I tre moduli, sistema servoassistito, modulo di aspirazione e di visione possono essere combinati insieme per offrire cinque diverse combinazioni per le diverse esigenze degli utenti.



Pos	Descrizione
A	Viti consigliate, tre M2,5 x 8
B	Tubo dell'aria
C	Connettore a 8 pin (caricato a molla)
D	Perno guida

Figura 14 - Dettaglio della pinza ABB e varie combinazioni di grippers realizzate da ABB per l'IRB14000

In più si possono combinare due tipologie diverse di pinze per i due bracci dell'IRB14000 questo è un vantaggio molto importante: è possibile, ad esempio, utilizzare il sistema di visione per analizzare gli elementi da prendere con la pinza o la ventosa. Con un robot di un altro tipo bisognerebbe cambiare il tool oppure installare una strumentazione esterna andando così ad aumentare il tempo ciclo. Mentre un braccio vede l'altro intanto andare a prelevare e successivamente, memorizzata la posizione dell'elemento, il braccio che ha il sensore ottico può eseguire un'altra azione andando così a mascherare il tempo per compiere le diverse azioni. La pinza offerta da ABB è modificabile: sia le dita che le ventose possono essere sostituite dall'integratore di sistema in base alle esigenze.

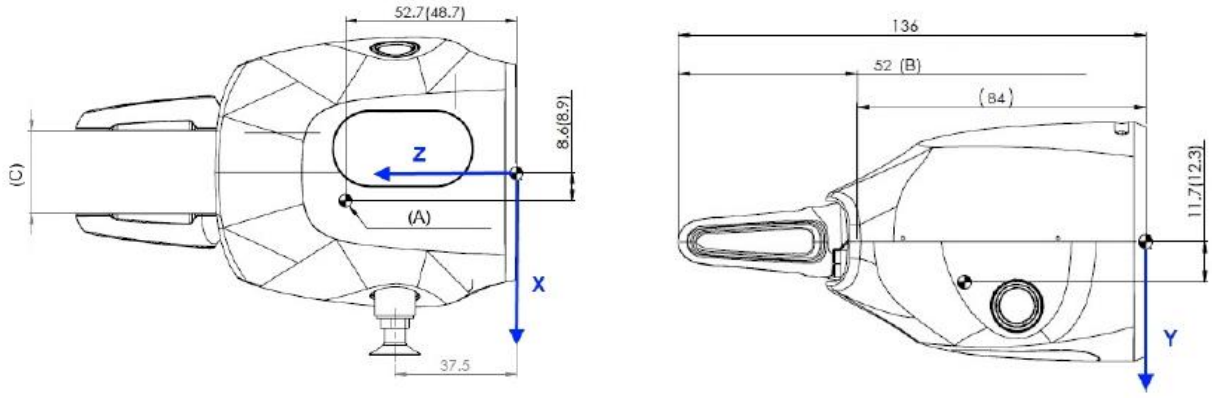


Figura 15 - Dimensioni della pinza e assi di riferimento del TC

Rispetto al polso è definito un altro parametro importante ossia il carico che può sostenere ogni braccio del cobot. Si ricorda che l'IRB14000 è stato progettato per l'assemblaggio di piccoli componenti per cui le masse che può movimentare sono molto limitati.

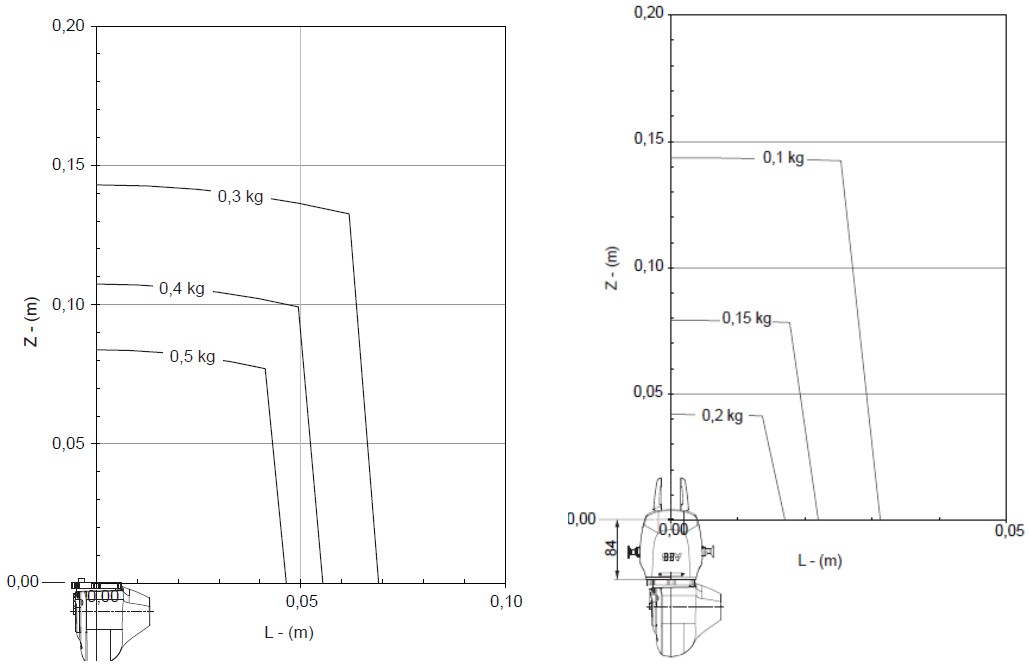


Figura 16 - Diagramma di payload senza pinza e con pinza

Come si può vedere dai diagrammi il carico risulta essere molto limitato e più ci allontana dal *tool0*[8] più il carico che il cobot può movimentare è minore. Al posto di utilizzare le pinze offerte da ABB, che dimezzano il carico che si può movimentare a vuoto, vengono realizzare dall'integratore di sistema pinze adatte per realizzare il compito pensato: questo significa usare materiali leggeri come alluminio o leghe leggere così da ridurre il più possibile il peso dell'utensile.

2.6 Il controller di IRB14000

Il controller dello Yumi è incorporato nella struttura portante dei bracci. Come si può vedere il cablaggio del sistema e le varie schede sono state inserite negli spazi della struttura rendendo così compatto il cobot.
Il controller è basato sulla famiglia IRC5 con le funzionalità standard. Il controller richiede una potenza RMS di 100-240 Watt e una tensione di 230V: questo permette una semplice installazione in tutti gli ambienti.
Il controller prevede schede dedicate per ogni braccio: la coordinazione dei movimenti è possibile perché è montata la scheda *"MultiMove"* che permette il sincronismo dei movimenti.

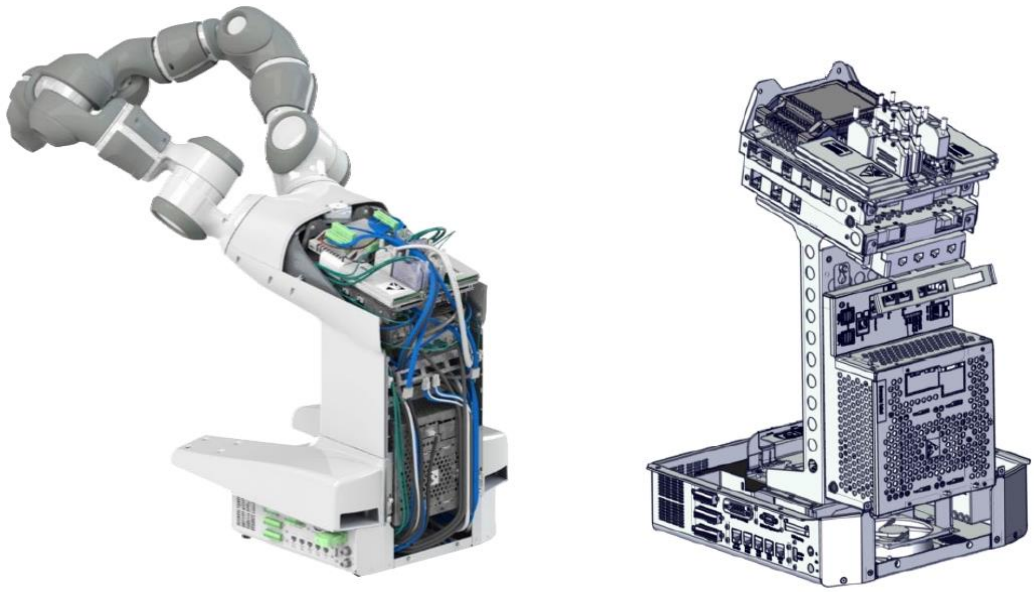


Figura 17 - Controller integrato nello YuMi. A destra visione dei componenti principali del controller

Si può vedere, partendo dalla base, che il controller è dotato di varie possibilità di collegamento con dispositivi esterni, sia digitali che analogici. In particolare, nell'immagine in dettaglio si può vedere come è possibile collegare tramite cavi ethernet il cobot con altri dispositivi tramite reti private, WAN o LAN. In questo modo è possibile interfacciare il sistema con un PLC che gestisce la linea su cui opera il cobot, oppure si può mettere il cobot in rete con l'azienda o ancora, collegare un PC per poter programmare i tasks che deve compiere tramite l'applicazione RobotStudio.

[8] Il TCP centrale del polso definito come sistema di riferimento zero per il polso

All’interno del controller è possibile collegare le tubature d’aria al sistema con gli ingressi a destra. Questi attacchi arrivano al TCP dei bracci dello YuMi grazie ai tubicini azzurri. È possibile collegare poi ingressi e uscite analogiche con i due rack predisposti a sinistra, tramite la DeviceNet di I/O. Questi ingressi sono collegati nella parte superiore del controller: è possibile utilizzare device aggiuntivi al polso e inviare i segnali registrati tramite i collegamenti mostrati nel paragrafo precedente.

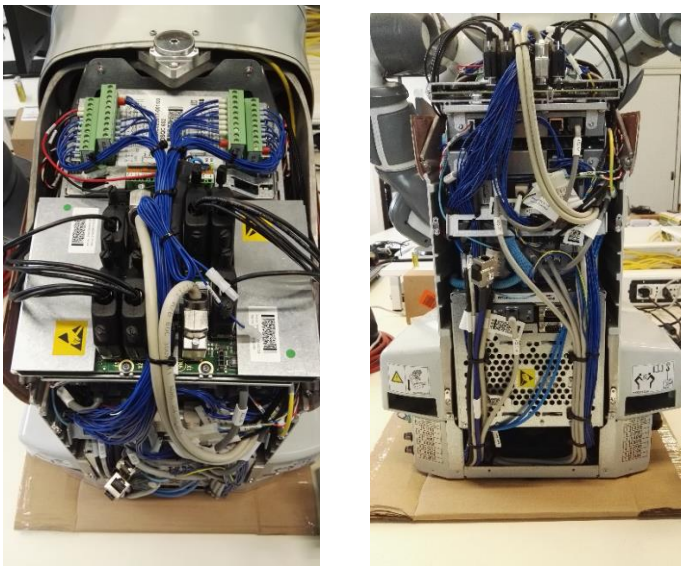


Figura 18 - Controller IRB 14000 a sinistra visto dall'alto, a destra visto dal retro

2.7 Il Controller IRC5

In questo paragrafo si vuole focalizzare l’attenzione non tanto sul controller dello YuMi, ma sul controller IRC5 su cui è basato per poter introdurre tematiche legate all’ottimizzazione e alla sicurezza delle isole robotiche.

Il controller è il cervello vero e proprio del robot e può essere collegato a sistemi di gestione e movimentazioni esterni. La Flex Pendant, o Teach Pendant, e *RobotStudio* [9] sono i due sistemi che si utilizzano per comandare il robot. Si interfacciano al controller in maniera differente.

La Teach Pendant ha un collegamento diretto con il controller del robot e ha una priorità di accesso al controller maggiore: infatti se si vuole accedere al controller del robot da pc, utilizzando l’app *RobotStudio*, si deve richiedere l’accesso che viene autorizzato nel caso in cui il tastierino è impostato in automatico mentre deve essere concesso nel caso in cui si trovi in manuale. La Flex Pendant utilizza il sistema operativo *RobotWare* che gestisce in tutto e per tutto il controller.

[9] Applicazione su pc dove è possibile scrivere il programma e simulare la stazione. Anche nella Flex Pendant, detta anche tastierino, può essere utilizzata per scrivere il programma ma risulta più laborioso. Unendo questi due strumenti è possibile velocizzare l’elaborazione del programma.

2.7.1 RobotWare

RobotWare è il sistema operativo del controller del robot: si basa sul linguaggio Rapid. A prima vista il linguaggio di programmazione risulta semplice e funzionale, flessibile e di alto livello. Andando ad analizzare la struttura delle varie funzioni si scopre che in realtà il linguaggio risulta estremamente articolato e permette la creazione di soluzioni sofisticate.

Grazie all'ampio toolbox, si permette l'accesso a caratteristiche e funzionalità kernel profonde consentendo, così, di creare caratteristiche ancor più ricche e applicazioni e soluzioni robotiche realizzate su misura. Le moderne interfacce di programmazione e dei sensori, insieme agli specifici pacchetti di applicazioni, sono tutti esempi degli innumerevoli e potenti strumenti ora disponibili in *RobotWare*.

Il controllo del movimento è alla base delle prestazioni dei robot in termini di precisione del percorso, velocità, tempi ciclo, programmabilità e sincronizzazione con dispositivi esterni quali nastri trasportatori e telecamere di visione. Ottimizzando il controllo di questi parametri, *RobotWare* è in grado di aumentare la qualità, la produttività e l'affidabilità. Inoltre, grazie alla prevedibilità e alle alte prestazioni del suo comportamento, garantisce l'indipendenza del percorso dalla velocità, senza alcuna necessità di aggiustamenti.

Le release rilasciate a partire dal lancio dello YuMi prevedono funzioni integrate legate al mondo dei robot collaborativi ma vanno a potenziare le precedenti funzionalità dei robot.

In particolare, si vogliono segnalare, in quanto argomenti trattati successivamente, queste funzionalità:

- Modalità del processo di movimento: ottimizzazione del comportamento del robot in base a esigenze specifiche, cioè ottimizzazione delle prestazioni per la specifica applicazione.
- Gestione degli errori di movimento: preserva l'esecuzione di RAPID in caso di errori di movimento quali collisioni o singolarità

Figura 19 - Flex pendant con sistema RobotWare

2.7.1.1 La Flex Pendant Interfaccia uomo-macchina (HMI) e le possibili movimentazioni

Il tastierino permette di muovere il robot, apprendere target[10], creare percorsi e, per i collaborative robot, insegnare manualmente le traiettorie.

Per poter muovere lo YuMi, ma questo vale per qualsiasi altro robot, con il tastierino ci si deve anzitutto porre con l'idea che in realtà si ha davanti due robot, cioè i bracci dello YuMi: questi vengono identificati come *Task_L* e *Task_R*.

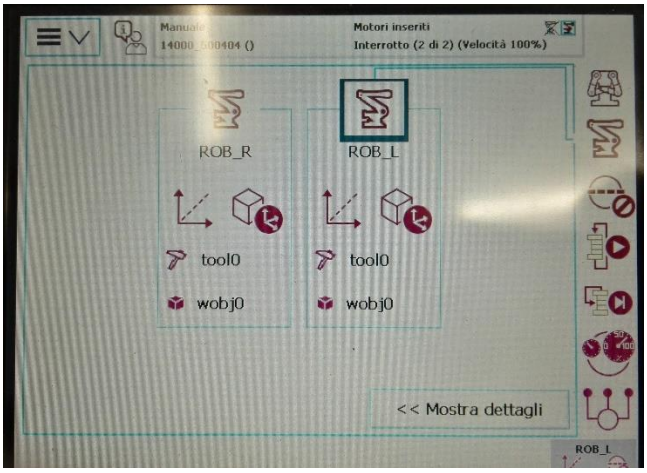


Figura 20 - Tasks selezionabili da FlexPendant, ogni task è riferito ad una unità meccanica

Prendendo in mano il tastierino si ha la possibilità, andando a selezionare nel menu principale la voce Movimento Manuale, di avere le varie tipologie di movimentazione dello YuMi che sono:

- *Movimento Giunto*: con il joystick è possibile muovere ogni giunto singolarmente. Il joystick ha 3 possibilità di movimenti, le due direzioni piane e la rotazione del pomello. Lo YuMi ha sette assi per braccio per cui questo tipo di movimentazione è possibile andando a selezionare 3 assi per volta.
- *Movimento Lineare*: attraverso il joystick è possibile muovere tutto il braccio nelle direzioni spaziali. Questo perché prende come punto di riferimento il TCP[11]. Come si vede nell'immagine è selezionabile il workobject, ovvero il sistema di riferimento in cui ci si vuole muovere, e l'utensile. Nel caso in cui sia presente un utensile viene mossa la posizione del sistema di riferimento dell'utensile, cioè la punta dell'utensile.
- *Movimento di braccio*: con questa modalità è possibile modificare la configurazione e l'orientazione del braccio. Infatti, il sistema di riferimento dell'utensile non viene mosso dal punto e il braccio ruota mantenendo il punto finale fisso.
- *Movimento Lead-Thorugh*: si può muovere il braccio del robot e posizionarlo manualmente, cioè condurlo con le mani nello spazio. Questo è possibile attivando dalla Flex Pendandt l'abilitazione di

[10] I target sono punti nello spazio definiti tramite coordinate: in particolare per rendere univoco il punto e come deve essere posizionato il braccio del robot si utilizzano le coordinate spaziali e i quaternioni per indicare il punto e la sua orientazione.

[11] TCP cioè Tool Center Point, il centro della flangia del robot. Questo è vero fintantoché si considera nelle impostazioni presenti della pagina di movimento manuale il tool0, sistema di riferimento definito per il punto

movimentazione manuale. Questa modalità è possibile con l'IRB 14000 Yumi e l'IRB 14050 Single Arm.

L'utilizzo di queste modalità permette di gestire al meglio il posizionamento del braccio rendendo più agevole la scrittura del programma. In particolare, è possibile insegnare i movimenti ad un collaborative robot: cercando su internet si trovano vari filmati demo che mostrano come si può insegnare il movimento al braccio del cobot utilizzando la modalità Lead-Through.

Attraverso la Flex Pendant è possibile salvare i target, definire sistemi di rifermento i workobject e attivare varie funzioni che possono aiutarci a identificare, ad esempio, il carico dell'utensile e i momenti di inerzia.

Tutto ciò serve per validare o correggere ciò che viene definito attraverso l'applicazione *RobotStudio*, potente strumento dove è possibile scrivere il software ma anche simularlo.

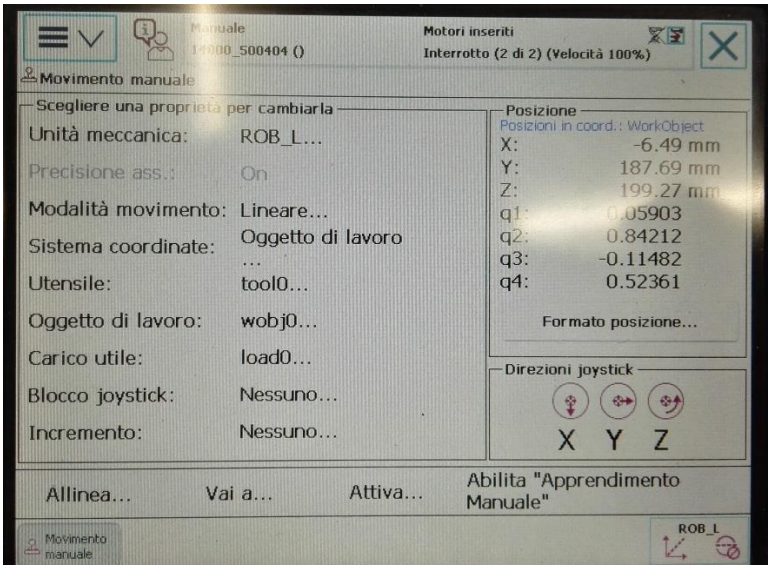


Figura 21 - Schermata della flex pendant, definita "Movimento Manuale" in cui è possibile definite come muoversi. In basso a destra si può vedere l'opzione Abilità Apprendimento Manuale cioè la modalità Lead-Through

2.7.2 RobotStudio

RobotStudio è il software che la casa ABB ha sviluppato per la programmazione dei robot. Mette a disposizione strumenti per la modellazione, la programmazione offline e la simulazione di celle del robot. *RobotStudio* consente di lavorare con un controller offline, ovvero un controller IRC5 virtuale in esecuzione locale sul PC. Tale controller offline viene anche indicato come controller virtuale (VC). *RobotStudio* consente inoltre di lavorare con il controller IRC5 fisico reale che viene semplicemente indicato come controller reale.

Quando *RobotStudio* viene utilizzato con controller reale, viene indicato come in modalità online. Quando funziona senza essere connesso ad un controller reale, oppure se connesso ad un controller virtuale, *RobotStudio* è in modalità offline.

L'applicazione è suddivisa in questi sottomenu:

- *Home*: in questa scheda è possibile definire target, percorsi e importare oggetti disegnati con software CAD , utensili che il robot deve usare e sistemi di riferimento.

- *Modellazione*: la scheda mette a disposizione strumenti per definire geometrie e oggetti complessi. La scheda Modellazione contiene i comandi utili alla creazione e al raggruppamento di componenti, alla creazione di corpi, alle misurazioni e alle operazioni CAD .
- *Simulazione*: la scheda contiene i comandi necessari a impostare, configurare, controllare, monitorare e registrare le simulazioni.
- *Controllore*: contiene i comandi necessari per la sincronizzazione, la configurazione e i task assegnati al controller virtuale (VC). Contiene inoltre i comandi per la gestione del controller reale.
- *Rapid*: contiene l'editore RAPID integrato, usato per la modifica di tutti i task del robot tranne il movimento del robot.
-

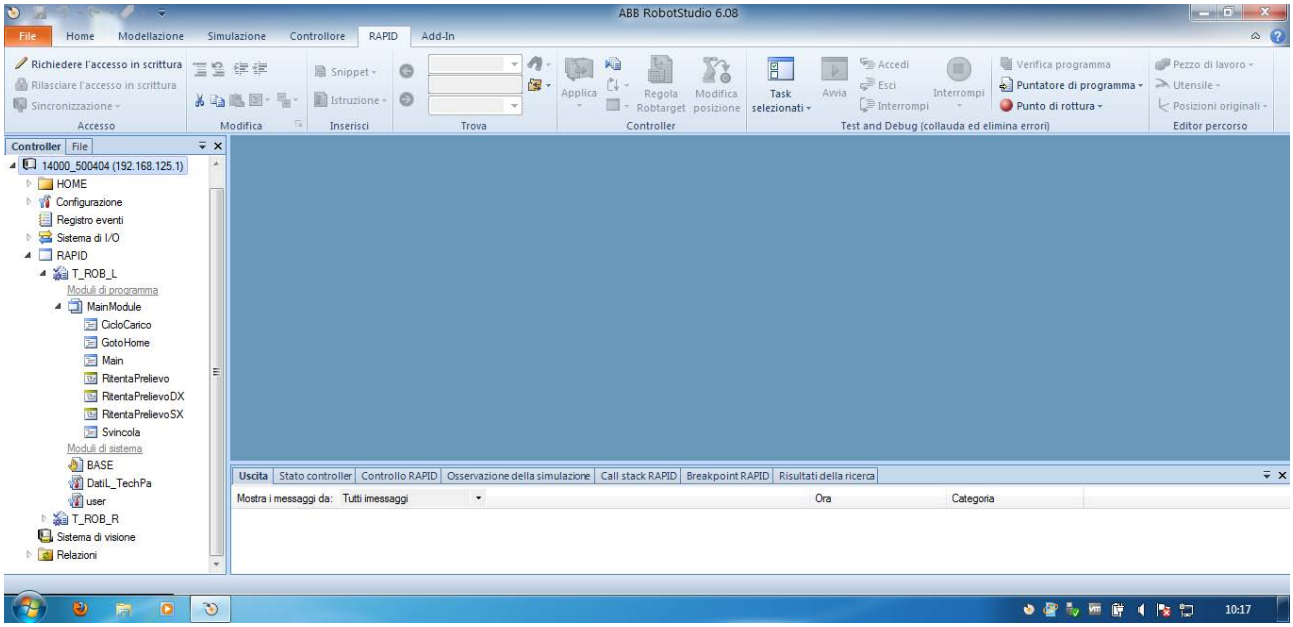


Figura 22 - Schermata del software RobotStudio con vista delle varie schede.
Nella finestra a sinistra è possibile vedere il controller suddiviso, sotto la dicitura Rapid, in Task_L e Task_R

Il software è pensato per lavorare con due approcci: online e offline, come già accennato precedentemente. Per semplificare la comprensione del software si può pensarlo diviso in due ambienti: la parte di programmazione e la parte di simulazione.

La parte di programmazione è legata alle schede Controllore e Rapid, dove nella prima si vanno a definire le caratteristiche del controllore e le tipologie di schede che supporta, mentre nella seconda scheda si scrive il task. Come visto nel paragrafo precedente il controllore che ABB offre è molto semplice; una peculiarità molto interessante è la possibilità di aggiungere diverse schede sia di sicurezza sia con funzionalità aggiuntive. Ad esempio, la scheda WolrdZone aggiunge la possibilità di conoscere in real-time la posizione dei giunti e quindi del TCP nello spazio, oppure le schede di comunicazione con altri sistemi come per esempio la scheda profinet che permette la comunicazione con PLC o sistemi che utilizzano questo protocollo di comunicazione.

Le altre schede fanno parte dell'area di simulazione: con l'ausilio delle prime due si possono definire geometrie, sistemi di riferimento, target, le configurazioni con cui il robot deve muoversi, importare geometrie

e sistemi definiti con l'ausilio di software CAD sia in 2D che in 3D, definire e progettare utensili, programmare le traiettorie.

Tramite l'opzione di sincronizzazione[12], strumento che consento di salvare le modifiche fatte nell'area di programmazione verso l'area di simulazione e viceversa, è possibile testare le traiettorie e vedere eventuali problemi come singolarità o interferenze con oggetti presenti nella stazione.

RobotStudio è un potente strumento che aiuta e velocizza la programmazione ma, soprattutto, permette di testare, all'interno di una stazione virtuale, tutti i movimenti che il robot deve fare. Risulta più semplice e intuitivo lavorare con l'area di simulazione per il semplice fatto che la parte di simulazione ha moltissime funzioni appartenenti ai CAD . Una peculiarità degna di nota è la possibilità di importare moltissimi formati provenienti dai diversi software di disegno; durante l'importazione suddivide l'assieme di elementi nei diversi elementi che definisco l'oggetto importato. Grazie a questa importazione è possibile modificare gli oggetti senza dover utilizzare un altro software.

Questa applicazione permette, in una prima fase di elaborazione del progetto, di analizzare come disporre la stazione, visualizzare gli ingombri e vedere i limiti che il robot può avere, in particolare punti di singolarità non evitabili facilmente e limiti fisici.

2.7.3 Schede aggiuntive del controllore

Il controllore IRC5, su cui si basa anche il controller dello YuMi, ha la possibilità di inserire schede aggiuntive per implementare in particolare protocolli di comunicazione, di sicurezza o di accuratezza e ripetibilità. Le schede implementano capacità che possono essere utili in determinate situazione o, nel caso della comunicazione, per poter scambiare dati e segnali con un PLC o con altri dispositivi che hanno protocolli di comunicazione particolari.

1) Il controller, grazie al sistema operativo RobotWare, supporta i più recenti bus di campo per I/O industriali e collabora correttamente con qualsiasi rete di stabilimento, andando così incontro alle esigenze della normativa europea riguardante l'industria 4.0. Il software è progettato per lavorare con una vasta gamma di dispositivi HMI, tra i quali anche il FlexPendant di ABB, i display industriali e i tablet e gli smartphone normalmente disponibili in commercio. Come impostazione di default il controller utilizza il protocollo *Ethernet\IP* ma è possibile utilizzare protocolli[13] quali *Profinet*, *Modbus* e altri protocolli integrati con funzioni safety.
2) Un'altra scheda molto interessante è quella dell'*absolute accuracy*: questa scheda permette di svolgere con estrema precisione e ripetibilità i task che deve svolgere il robot andando a ridurre l'errore di posizionamento. Infatti, le tolleranze meccaniche intrinseche e le deflessioni dovute al carico e alla

[12] Opzione selezionabile in tutte e due le aree. L'opzione è presente sia nella scheda Controllore che in quella Rapid e nella scheda Home.

[13] I protocolli di comunicazione sono il linguaggio di comunicazione che viene utilizzato per poter scambiarsi segnali e dati e far in modo che questi siano significativi. I più famosi sono Profibus-Dp/Dpv1, Profinet, Ethernet, EtherCat, DeviceNet, Modbus, CanOpen.

struttura riduce la precisione assoluta del robot: la compensazione di tali errori è un problema complesso e non lineare. La soluzione offerta da ABB è quella di compensare le posizioni internamente nel controller offrendo così, una precisione misurabile sul posizionamento dell'utensile. Risulta possibile mantenere l'accuratezza delle celle del robot per tutta la durata della vita del robot. Questo è possibile attraverso un algoritmo che utilizza dei dati che si possono introdurre nel robot, in particolare ci si riferisce ai dati del baricentro e dei momenti di inerzia dell'utensile.

3) Per migliore la sicurezza del robot è possibile aggiungere schede come la Safety e la SafeMove. Le due schede appena citate compiono compiti differenti: la prima, quella di Safety, ha il compito di gestire la catena della sicurezza, mentre la seconda introduce una prima forma di collaborazione tra robot e uomo. Se integrate nello stesso controller le due schede collaborano per garantire un livello di sicurezza elevato.

   L'utilizzo della SafeMove permette al controller un alto livello di sicurezza: infatti il robot, tramite l'utilizzo di funzioni di supervisione, è in grado di ridurre la velocità o di fermarsi. In genere viene integrato con altri dispositivi, quali barriere fotoelettriche per rilevare gli spostamenti dell'operatore. La SafeMove è una scheda che introduce:

   i) la possibilità di supervisionare la posizione e l'orientamento del target;
   ii) la posizione e le velocità degli assi;
   iii) permette l'arresto del robot senza dover spegnere i motori e si possono gestire le classi di fermata;
   iv) si può definire un'area di contatto in cui il robot può esercitare una forza maggiore, senza andare in collisione, perché magari sta cambiando un tool oppure sta eseguendo una rettifica o una saldatura.

Altre schede possono essere integrate nel controller per poter svolgere al meglio il lavoro del robot. Qui sono stati riportati alcuni esempi significativi perché introducono temi di discussione quotidiana all'interno di aziende che progettano e sviluppano isole robotiche.

Dopo aver introdotto nel 2008 SafeMove, soluzione a sicurezza certificata per il monitoraggio dei movimenti dei robot, la supervisione di utensili e di macchine in stato di inattività/fermo e la limitazione della velocità, ABB ha reso disponibile SafeMove2, che contribuisce alla creazione di scenari di produzione più flessibili ed efficienti e integra la comunicazione sicura degli IO tramite bus di campo ProfiSafe nella famiglia di controller robot IRC5 di ABB, senza bisogno di PLC di sicurezza esterno.

Il sistema SafeMove2 si è evoluto ed è stato pensato per favorire lo sviluppo a cura della redazione di applicazioni robotizzate innovative, integrando le caratteristiche di sicurezza direttamente nel controllore robot e facilitando la collaborazione uomo/robot. Questa evoluzione comprende la possibilità di programmare funzioni specifiche dedicate a più aree, gamme e strumenti; ulteriori funzionalità possono essere aggiunte nel tempo. Con questo prodotto possono essere definite fino a 16 aree, attivabili e disattivabili in modo dinamico e sicuro. Con SafeMove 2, invece, è possibile abilitare l'attività del robot esclusivamente sulla cella in cui non sono necessari interventi umani. In questo modo le attività produttive, in quella cella, possono continuare anche se non ci sono vere e proprie barriere di separazione tra le due celle. Non solo, con l'ausilio di sensori di

sicurezza, in caso di presenza umana nell'area di lavoro del robot, è possibile la fermata sicura di questo senza taglio dell'alimentazione permettendo la ripartenza immediata non appena l'area è libera. SafeMove 2 consente il controllo di applicazioni pericolose come ispezioni a raggi X e taglio laser, comprende hardware dedicato miniaturizzato per garantire le prestazioni del sistema di sicurezza includendo un affidabile Safety IO.

2.8 MultiMove: strumento per coordinare più robot

MultiMove è progettato per consentire a un unico controller di gestire più robot. Questo permette di risparmiare sui costi hardware, oltre che a offrire un coordinamento avanzato tra robot diversi e altre unità meccaniche. In particolare, lo YuMi ha questa funzionalità presente all'interno del controller permettendo così tutte quelle operazioni di movimentazioni sincronizzate e coordinate che sono possibili, per gli altri robot, con l'ausilio di un controller aggiuntivo che coordina i controller di ogni robot.

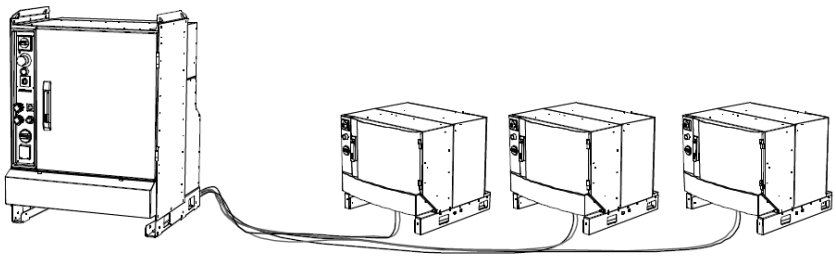


Figura 23 - Sistema MultiMove con controller collegati

Il sistema MultiMove si può pensare come un sincronizzatore: può movimentare fino a 6 task, dove per task si intende una sequenza di istruzioni di movimento, ma non più di quattro azionamenti, dove per azionamento va inteso il controller IRC5. Come si può capire dall'immagine il MultiMove è un controller di supervisione che può essere cablato con quattro controller: ogni controller movimenta un braccio meccanico.
Il MultiMove consente di realizzare:

- *Movimenti indipendenti*: si intende che i robot compiono delle azioni indipendenti tra di loro, cioè ogni sistema robotico avrà un task da eseguire indipendentemente da altri sistemi robotici. Può risultare utile per gestire il caso in cui due o più robot devono scambiarsi oggetti e non sia presente un PLC: in questo caso è possibile sfruttare alcune opzioni di sincronizzazione per farsi che avvenga lo scambio di oggetti.
- *Movimenti semicoordinati*: si intende la possibilità di eseguire azioni su uno stesso oggetto di lavoro a condizione che l'oggetto non sia in movimento. I movimenti semicoordinati richiedono una forma di sincronizzazione tra i task anche se i movimenti risultano indipendenti tra di loro. Per spiegare meglio cosa si intende si può pensare a questo caso: un posizionatore può spostare l'oggetto di lavoro quando i robot non sono coordinati con esso e i robot possono essere coordinati all'oggetto di lavoro quando non è in movimento. L'alternanza tra spostamento dell'oggetto e coordinazione dei robot viene definita "movimenti semicoordinati". Il posizionatore deve sapere quando è possibile spostare l'oggetto di lavoro e

i robot devono sapere quando possono agire su di esso. Tuttavia, non è necessario che tutte le istruzioni di movimento siano sincronizzate.

Il vantaggio consiste nel fatto che ciascun robot può lavorare in modo indipendente con l'oggetto di lavoro. Se i vari robot eseguono compiti molto diversi tra loro, questo può consentire di ridurre il tempo necessario al ciclo rispetto a quanto avverrebbe scegliendo di sincronizzare tutti i movimenti dei robot.

- *Movimenti coordinati sincronizzati*: possibili con l'opzione MultiMove Coordinated. Questa opzione è la più interessante per lo sviluppo software che sarà illustrato successivamente. Con questa opzione è possibile sincronizzare non solo una sequenza di azioni che segue una logica di causa-effetto ma significa che i programmi task RAPID, che gestiscono ciascuno un'unità meccanica, eseguono le rispettive istruzioni di movimento simultaneamente. I movimenti coordinati sincronizzati consentono generalmente di ridurre il tempo necessario per il ciclo, dal momento che i robot non devono attendere mentre l'oggetto di lavoro viene spostato.

Per quanto riguarda lo YuMi, come detto precedentemente, si deve considerare l'opzione MultiMove già presente. Infatti, si tratta di due manipolatori robotici che lavorano insieme e grazie a questa opzione è possibile muovere il cobot in modo sincronizzato o coordinato.

Il sistema MultiMove non modifica l'interfaccia tipica sulla teach pendant e risulta quindi intuitivo muoversi all'interno di essa. Naturalmente, essendo due bracci, si hanno due task di movimento e tutto verrà quindi duplicato.

Verrà illustrato successivamente, nei paragrafi ***3.4.2 Altre funzioni dell'istruzione Move, concorrenza di istruzioni e sincronismo***e ***3.6.1 Descrizione della procedura di prelievo e deposito***, come sincronizzare i movimenti e come utilizzare questa opzione particolarmente interessate.

2.9 Cinematica dello Yumi

In questo paragrafo andremo ad analizzare un braccio dello YuMi poiché ha delle particolarità che rendono più complessa la soluzione della cinematica inversa, la quale sarà affrontata nel CAPITOLO 4.

Per cinematica diretta di posizione si intende il problema legato al ricavare la posa del robot per raggiungere un determinato punto. Il problema ha sempre soluzione in quanto si conoscono i valori delle variabili di giunto e le dimensioni dei bracci per cui la soluzione è certa.

Per cinematica inversa, invece, si intende il problema legato al ricavare le posizioni dei giunti conoscendo la posizione dell'end effector. Questo problema può avere soluzioni all'interno dello spazio di lavoro. All'interno dello spazio di lavoro posso raggiungere lo stesso target con pose differenti del robot e, in questo caso, si possono avere infinite soluzioni. Un altro problema è legato alle singolarità, come verrà illustrato nel CAPITOLO 4.

2.9.1 Convenzione di Denavit-Hartenberg

La convenzione di Denavit – Hartenberg, abbreviata DH, è usata per scegliere i sistemi di riferimento utilizzati in applicazioni robotiche. Permette di rappresentare una trasformazione geometrica in un numero minimo di parametri nello spazio euclideo. Il numero minimo è quattro e di seguito verranno illustrati.
Per poter descrivere la trasformazione per prima cosa è necessario definire, come segue i sistemi di riferimento. Considerando due giunti consecutivi:

- l'asse Z_{n-1} si sceglie coincidente con l'asse del giunto $i-1$, mentre Z_n coincidente con l'asse del giunto i
- l'asse X_{n-1} può essere scelto liberamente, ma è conveniente porlo in direzione del giunto successivo, e si interseca con Z_{n-1} in corrispondenza del giunto $i-1$; l'asse X_n corre lungo la normale comune fra gli assi Z_{n-1} e Z_n
- gli assi Y_{n-1} e Y_n sono scelti in modo da completare le terne seguendo la regola della mano destra.

La trasformazione è allora descritta da quattro parametri:

- $\boldsymbol{d}$, distanza dell'asse Z_{n-1} dalla normale comune; nel csao vi siano infinite normali comuni, cioè Z_{n-1} e Z_n paralleli, si scegierà il valore più conveniente di d
- $\boldsymbol{\theta}$ angolo di rotazione intorno all'asse Z_{n-1} necessario per allineare X_{n-1} e X_n
- $\boldsymbol{\alpha}$ indica l'angolo di rotazione intrno alla normale comune per allineare l'asse Z_{n-1} a Z_n
- $\boldsymbol{a}$ è la distanza minima fra gli assi Z_{n-1} e Z_n

Si fa notare che l'asse X_n è perpendicolare sia all'asse Z_{n-1} che all'asse Z_n e interseca entrambi. Ad ogni parametro è associata una matrice da cui si ricava che:

$$T_{i,i-1} = R_x(\alpha_{i-1})\ T_x(a_{i-1})\ R_z(\theta_i)\ T_z(d_i)$$

Dove R indica una matrice di rotazione e T una matrice di traslazione.

2.9.2 Analisi dello Yumi con Denavit-Hartenberg

Come si può vedere nella Figura 24 - Rappresentazione minima del braccio dell'IRB14000, gli assi sono disallineati, gli assi, evidenziati in rosso, del braccio sono disallineati, come si può confrontare con la Figura 9 -Esploso del braccio IRB 14000 e dimensioni dei link. Questo complica la risoluzione della cinematica inversa. Di seguito vengono riportati i parametri di Denavit-Hartenberg per l'IRB14000.

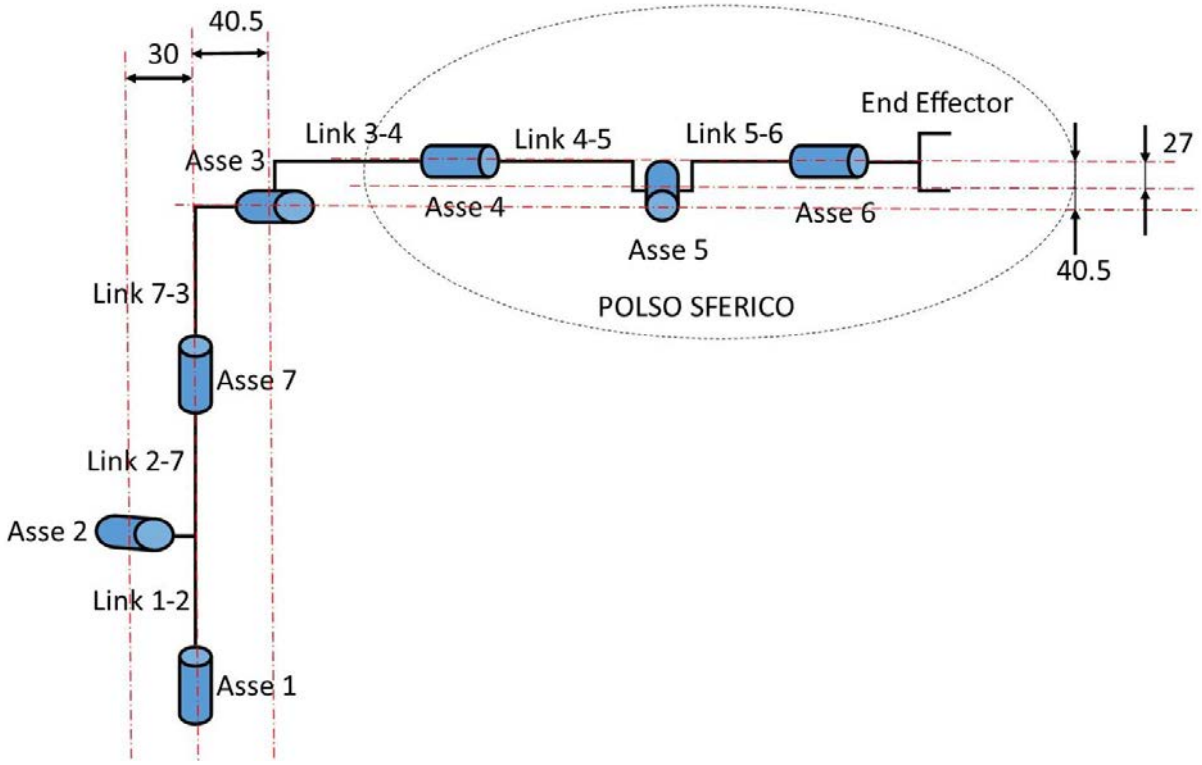


Figura 24 - Rappresentazione minima del braccio dell'IRB14000, gli assi sono disallineati. I valori sono espressi in millimetri.

Tabella 4 - Parametri di Denavit - Hartenberg per il braccio destro dello YuMi

Link	$T_{i,i-1}$	α_{i-1}	a_i	θ_i	d_i
Base - 1	$T_{0,1}$	0	0	0	0
1 - 2	$T_{1,2}$	90	0.03	180	0
2 - 7	$T_{2,7}$	-90	0.0269	0	0
7 - 3	$T_{7,3}$	90	0.029	0	0.25
3 - 4	$T_{3,4}$	-90	0.0405	-90	0
4 - 5	$T_{4,5}$	-90	0.0405	-180	0.2649
5 - 6	$T_{5,6}$	-90	0.02699	180	0

I parametri in tabella sono stati ricavati dalla libreria di RobotStudio, software di ABB. Esplorando all'interno del programma ci si è imbattutati in una cartella che contiene tutti i parametri di tutti i modelli di robot della casa costruttrice. I parametri contenuti nel file sono necessari per eseguire le simulazioni e poterne caricare la geometria. All'interno di tale file vengono espresse tutte le condizioni di vincolo e come sono legati tra loro i diversi assi e componenti del braccio. La struttura disallineata del braccio del cobot è stata realizzata per ridurre i problemi di singolarità di spalla, gomito e polso. Infatti, rispetto ad un antromorfo a sei assi della stessa casa, lo YuMi presenta molti meno problemi di singolarità. Si lascia la trattazione al paragrafo 4.1.1 ***Le singolarità***.

CAPITOLO 3

CASO APPLICATIVO: SVILUPPO DELLA LINEA DI CONFEZIONE E ANALISI DEL SOFTWARE DEL COBOT

L'applicazione sviluppata è legata ad una linea di confezionamento: per confezionamento non si intende il packging vero e proprio, ma l'inserimento di confezioni all'interno di uno scatolone. La stazione in studio della linea ha il seguente scopo: il cobot deve recuperare le confezioni da un nastro trasportatore, per posizionarle successivamente all'interno di uno scatolone. Questo a sua volta sarà inviato, tramite nastro verso la stazione di pallettizzazione. Come si può vedere dal layout, al paragrafo 5.1, sono presenti due cobot che lavorano contemporaneamente sulle due linee parallele. La scelta di utilizzare questo robot è legata all'esigenza del cliente di non dover posizionare barriere e avere la possibilità di lavorare insieme ad operatori.
Lo studio inizialmente si è basato sull'analisi delle dimensioni spaziali per poter ottimizzare gli spazi e poter raggiungere tutti i punti con il braccio dello YuMi. Per poter verificare tutto ciò è stato necessario tener presente che:

- il range di lavoro del cobot
- le dimensioni degli scatoloni e delle confezioni

a queste fondamentali informazioni dimensionali si deve aggiungere poi:

- le dimensioni dei nastri trasportatori per poter ottimizzare gli spazi
- il posizionamento dello scatolone dove riporre le confezioni tenendo presente che lo spazio occupato da esso va sottratto allo spazio in cui può muoversi lo YuMi
- le forme dello scatolone infatti ve ne sono due tipi, uno che presenta delle lingue laterali e quindi el confezioni si devono far entrare ruotate, e uno che non le presenta
- il robot lavora con due tipi di confezioni differenti per ingombri e pesi

In questa applicazione, lo YuMi non ha le smartgripper di ABB: sono state realizzati dei tool specifici cioè un sistema di aspirazione con quattro ventose. La scelta è legata a due motivi: il primo legato al limite di peso che può movimentare lo YuMi, che come si è detto precedentemente, ha un carico[14] massimo di 500 grammi, comprendendo il tool applicato al polso. Il tool sarà realizzato in lega di alluminio e avrà un peso di 130 grammi. In questo modo si avrà la possibilità di movimentare fino a 370 gr di confezione. In realtà il peso delle confezioni è variabile e va dai 240 ai 280 grammi.
Il secondo motivo è legato al voler garantire una maggiore affidabilità di presa in modo da poter prelevare e movimentare la confezione con un baricentro stabile, in asse con il baricentro del polso del braccio (TCP).
Il layout definito per la successiva realizzazione della linea è stato definito dopo vari tentativi utilizzando l'ambiente di simulazione offerto da RobotStudio, di cui si parlerà nel paragrafo successivo.

[14] Come si può vedere nel diagramma di carico, essendo prossimi al limite massimo il carico movimentabile decresce all'aumentare della distanza dalla spalla del cobot.

3.1 Layout della stazione

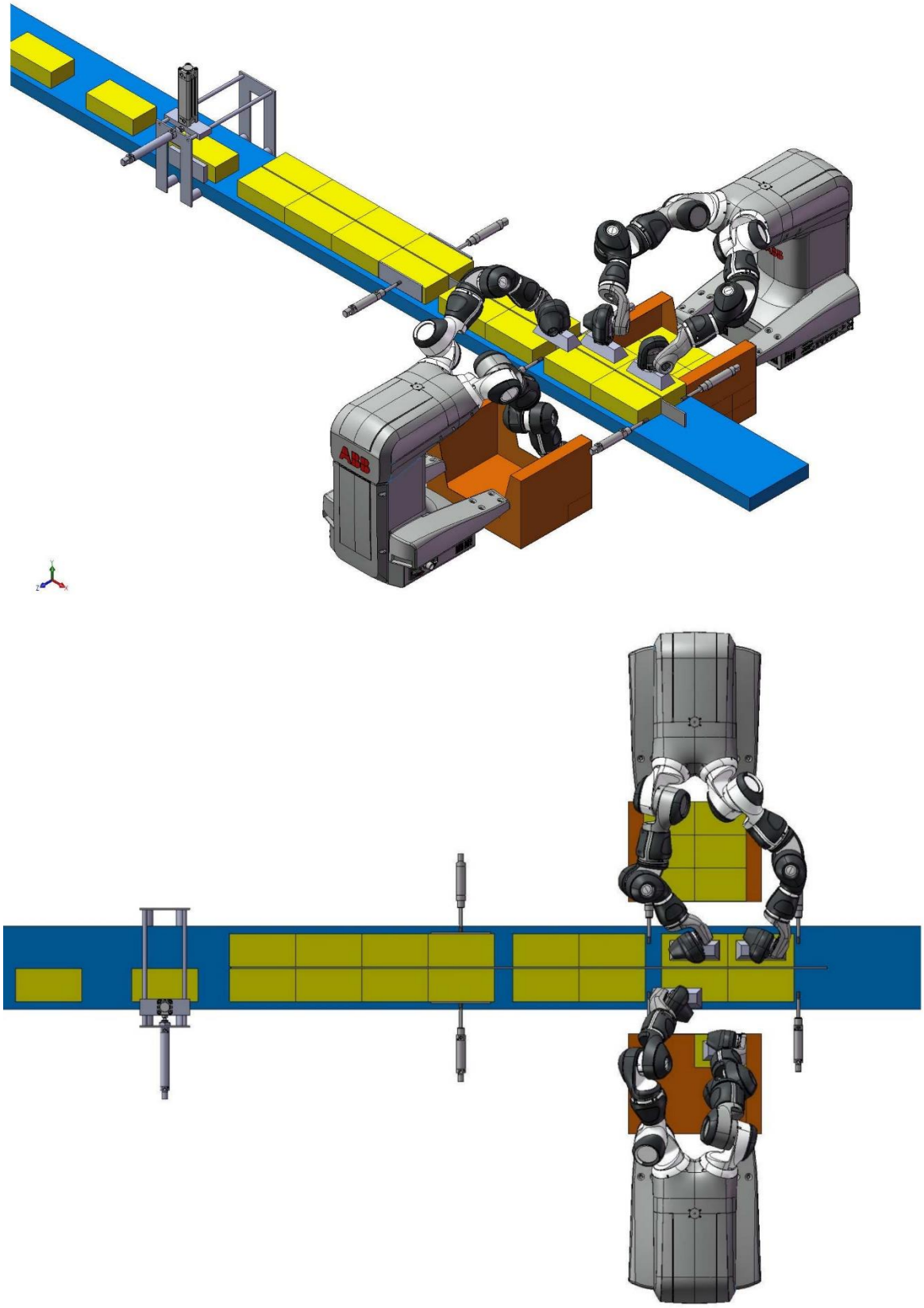


Figura 25 - Layout della linea di confezionamento

Il layout presentato in figura è la stazione robotica dove sono presenti i due robot collaborativi. Lo studio delle dimensioni e delle posizioni è stato fatto per tentativi in quanto, inizialmente, è stato eseguito uno studio esplorativo per verificarne la fattibilità. Essendo lo spazio occupato dello scatolone piuttosto elevato poichè risulta occupata gran parte dell'area di lavoro al di sotto dei bracci del robot, circa un terzo dell'area totale, è stato possibile determinare la disposizione del layout della stazione grazie all'utilizzo del simulatore presente nel software Robotstudio.

Considerati i limiti massimi e minimi del robot si è analizzata la posizione di prelievo e la posizione dello scatolone e quindi la posizione dei nastri e degli oggetti presenti considerandone gli ingombri. Le posizioni, o per meglio dire la posa del manipolatore e definita nel linguaggio Rapid di ABB *robtarget*[15], vengono definite attraverso le coordinate cartesiane e le rotazioni attorno agli assi in un sistema di riferimento definito. Un altro parametro fondamentale per definire la posa di un manipolatore è la configurazione del braccio dello YuMi: la configurazione del braccio viene indicata con quattro numeri, che corrispondo ai quadranti in cui il giunto deve andare.

All'interno del simulatore, che è un potentissimo strumento basato su CAD e a cui si rimanda a testi specifici per informazioni più approfondite, è possibile impostare tutti i target necessari, impostare la configurazione di ogni posa e definire le traiettorie. Grazie a questo tool è stato possibile rendersi conto di alcuni limiti che verranno spiegati teoricamente nei prossimi capitoli.

Un primo problema che è stato necessario affrontare è stato la risoluzione delle traiettorie, per certi aspetti obbligati a causa degli ingombri, in cui erano presenti punti di singolarità, si rimanda la trattazione al capitolo 4. Un secondo problema più complesso è stato gestire le collisioni, lo scopo centrale della tesi e su cui ci si è soffermati più a lungo e dove si è ricercata una soluzione pratica ed efficace da poter riproporre in altre situazioni.

A causa delle traiettorie obbligate si è dovuto verificare la possibilità di poter raggiungere tutti i punti interni allo scatolone: per poter entrare all'interno degli scatoloni è necessario inclinare le confezioni in quanto lo scatolone presenta delle ali. Questo ha portato il sistema in alcune situazioni limiti dalle quali si è potuti uscirne.

Dopo aver eseguito vari test con il simulatore e aver capito che è la realizzazione della stazione è fattibile si è passati al cobot reale per poter analizzare alcuni limiti che il simulatore faticava a calcolare. Vicino al limite dell'area di lavoro e nell'intorno precedente ai blocchi meccanici il simulatore fatica a calcolare il comportamento del cobot e mostra l'impossibilità di raggiungere tali punti: i limiti sono dovuti in parte alla potenza limitata del calcolatore su cui veniva fatta girare l'applicativo di ABB e l'altro dovuto ad un calcolo dei movimenti e delle traiettorie iterativo.

Si è passati quindi al controller reale e attraverso la movimentazione Lead-Through, apprendimento manuale, attivabile dal tastierino del robot, in cui è possibile muovere il braccio del robot accompagnando il

[15] È una struttura dati presente nel linguaggio Rapid di ABB. Verrà illustrata nello specifico nei prossimi paragrafi

manipolatore robotico in un determinato punto ed in una determinata configurazione, è stato possibile verificare il comportamento del braccio nei casi in cui il simulatore dava errore di calcolo. Si è visto che il problema è legato al fatto che molte delle configurazioni obbligate per il passaggio del sistema erano prossime ai limiti meccanici dei giunti, in particolare del giunto 1 a 7. Tramite questo tipo di movimentazione si è sono impostate manualmente le posizioni dei giunti andando ad aumentare il numero di punti dove necessario per poter eliminare eventuali movimenti che intercettavano limiti fisici dei giunti.

3.2 Simulazione del ciclo

Per poter verificare l'effettiva realizzabilità della linea pensata si è ricorsi in un primo momento al simulatore presente in RobotStudio dove sono state realizzate diverse simulazioni e dove si è studiata la disposizione della stazione.

Come detto precedentemente il simulatore offerto dal software RobotStudio è uno potente strumento per poter visualizzare la stazione di lavoro, il robot e il controller. Grazie ad esso si può testare il sistema e le possibili configurazioni che si possono impostare per poter raggiungere i vari punti.

All'interno dell'ambiente di simulazione si può creare un sistema virtuale dove si possono creare o importare geometrie definite tramite ambienti CAD. Una volta definiti o importati gli oggetti che compongono e definiscono la stazione è possibile introdurre il controller virtuale.

È possibile costruirsi delle geometrie, attraverso le funzionalità simili ad un software di sviluppo CAD 3D, ed importare geometrie di tipo CAD. Una volta importato l'oggetto o la stazione, è possibile effettuare delle modifiche: in particolare è possibile collocare un controller virtuale per movimentare il robot e poter vedere, il ciclo programmato.

Il controller virtuale permette di eseguire simulazioni del comportamento del robot all'interno della stazione. Nello spazio creato è possibile definire dei target, da assegnare al controller, grazie ai quali è possibile definire traiettorie. All'interno del simulatore è possibile definire il controller come si desidera andando a scegliere le opzioni necessarie: ad esempio è possibile aggiungere la scheda per gestire le WorldZone oppure le schede di comunicazione di rete o le schede safety.

Definito quindi il robot e la CPU si può iniziare a scrivere il programma nella scheda Rapid di RobotStudio e assegnarlo al controller tramite la possibilità di sincronizzazione tra controller virtuale e script. All'interno della Scheda Home è possibile definire tutti i target necessari direttamente nello spazio, definendone le configurazioni, e, con essi, definire le diverse traiettorie.

Il simulatore è risultato molto utile nella prima fase di analisi poiché grazie ai diversi tentativi si è potuto analizzare una prima fattibilità della stazione robotica.

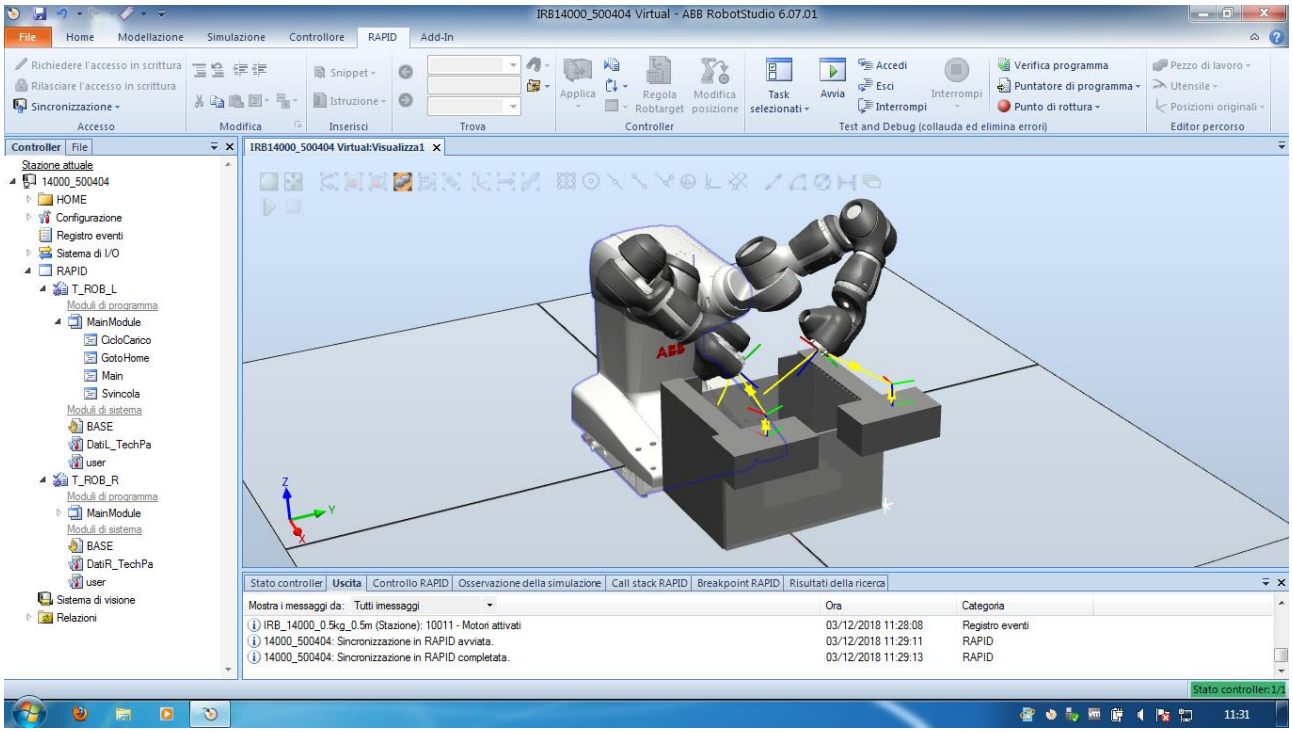


Figura 26 - Visione del simulatore, rappresentazione semplificata della stazione con analisi di movimentazione del cobot. Nell'immagine sono definite le traiettorie, in giallo, che passo per i target rappresentati da dei sistemi di riferimento

3.3 Programmazione del robot

Lo Yumi può essere visto come un robot collaborativo composto da due unità robotiche separate e coordinate da un sistema definito MultiMove. Il controllore del robot, che è anche la base dei bracci del cobot è composto da una scheda un MultiMove, che supervisione e permette di sincronizzare tutti o alcuni movimenti, due controller dedicati ad ogni braccio, alla risoluzione delle traiettorie e alla esecuzione del programma, e due bracci a 7 gradi di libertà.

Ogni manipolatore ha un suo controller nel quale è possibile programmare un task, ovvero il programma che ripeterà ciclicamente. Il task a sua volta è diviso in moduli: a sua volta si possono definire moduli di sistema utili per definire le variabili necessarie e le funzioni che devono essere eseguite all'avvio del controller e i moduli base dove è presente la procedura Main, la prima cercata dal controller e dalla quale vengono chiamate a catena tutte le altre definite nei vari moduli. Nel nostro caso si sono utilizzati tre moduli, di cui uno di sistema dove sono state definite tutte le variabili e due base, uno per le trap function, di cui parleremo successivamente e uno dove è stato definito il ciclo.

La suddivisione del programma tra dati e procedure è stata fatta per una maggior leggibilità per chi dovrà accedervi ed effettuare la manutenzione.

Successivamente verranno spiegate tutte le procedure e le istruzioni che sono state utilizzate all'interno del programma, riportato nell'**Appendice A**.

Prima di passare all'analisi di alcune parti del programma scritto per la stazione robotica, si vuole porre l'attenzione su alcune chiamate di sistema che ci aiuteranno a comprendere come è stato possibile risolvere

problemi di singolarità e di ripristino dell'esecuzione del programma del cobot. Queste attenzioni risultano essere di carattere generale e valgono per qualsiasi robot della casa ABB. Si vuol far notare che alcuni robot possiedono alcune caratteristiche speciali per il tipo di lavoro che devono eseguire. Ad esempio l'IRB990, ovvero il robot di tipo SCARA di ABB, che può eseguire il ciclo di prelievo e deposito attraverso un'unica istruzione nella quale si possono definire fino a cinque target. Altri robot, dedicati quasi esclusivamente alla saldatura e alla verniciatura, possono elaborare determinate funzioni tali da eseguire la saldatura o la verniciatura in modo più semplice ed efficace.

Prima di procedere con l'analisi, si vuole mostrare come creare la comunicazione e come viene gestita dal controller. Nel paragrafo ***3.3.2 Segnale di Supervisione del Movimento*** verrà illustrato un particolare segnale gestito all'interno del controller: questo è risultato necessario per la gestione della supervisione del movimento e per risolvere il problema della collisione.

3.3.1 Segnali di Input Output

Per poter gestire in automatico il prelievo e il deposito il cobot deve poter comunicare con il sistema che gestisce la linea rappresentata in Figura 25 - Layout della linea di confezionamento I due cobot comunicano con il PLC che gestisce la movimentazione dei nastri, controlla la presenza di confezioni e da prelevare e dello scatolone in cui depositare. È stato, quindi, necessario definire alcuni segnali digitali scambiati tramite ethernet illustrati nell'immagine.

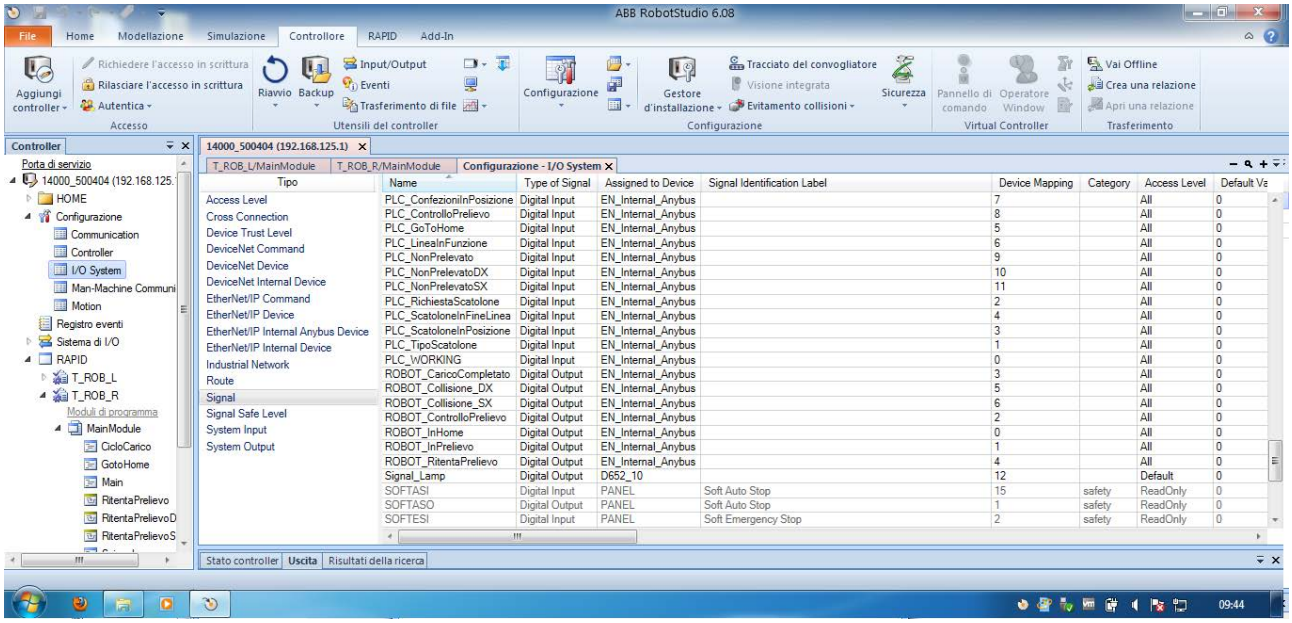


Figura 27 - Segnali scambiati tra controller robot e PLC

Il PLC ha lo scopo di gestire la linea logicamente e in questo caso corrisponde agli occhi del sistema. Questo è possibile perché, nelle posizioni dei prelievo e nella posizione in cui è presente lo scatolone, verranno cablate delle fotocellule di presenza. Il PLC corrisponde, per voler fare un paragone, al cervello di un operatore e permette la coordinazione "occhio-mano" tra linea e bracci meccanici del cobot.

Nell'immagine sopra si può notare che i segnali sono stati identificati con un prefisso per indicare chi comunica. I segnali, ora assegnati internamente alla scheda EN_Internal_Anybus, in quanto si stava lavorando con un controller virtuale, saranno assegnati alla scheda di comunicazione Ethernet\IP con la quale è possibile stabilire una comunicazione con il mondo esterno al controller andando a definire un IP con la medesima sottoclasse.

I segnali che si possono scambiare tramite un patch cable sono raggruppabili in tre categorie:

1. Digital: si intende un segnale legato ad un bit, come quelli mostrati in figura
2. Group digital: si possono definire dei blocchi di bit, di solito si ragione con due byte, ovvero 16 bit, in modo da poter passare un valore intero. Nulla toglie di poter definire la dimensione del gruppo di segnali come si vuole. In questo caso si possono inviare solo valori assoluti
3. Analog: gli analog sono segnali che possono essere singolo bit oppure a gruppi di bit. In questo caso è possibile definire anche valori negativi. Risultano comodi per poter comunicare al cobot dati e altri segnali provenienti dal campo. In particolare, verranno utilizzati, anche se non si vede nella figura precedente, per definire gli offset che possono essere impostati da HMI.

3.3.2 Segnale di Supervisione del Movimento

Il segnale di supervisione di movimento viene generato dal controller quando uno dei sensori di coppia presenti sul braccio robotico rileva una coppia superiore alla massima. Nell'immagine viene riportato il controllo eseguito sulla forza vista dai sensori in funzione del tempo. Nel caso in cui vengano superati i valori il controller genera uno stop della movimentazione e visualizza sul tastierino l'errore di supervisione del movimento.

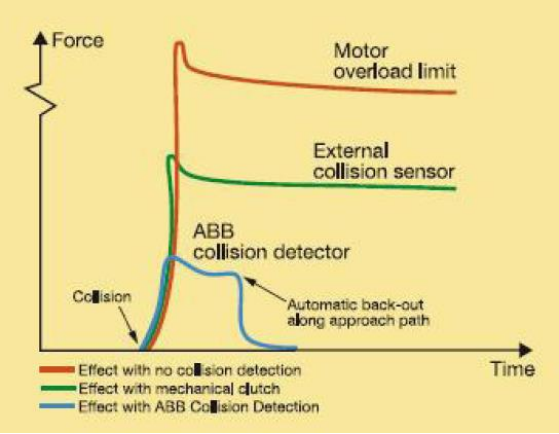


Figura 28 - Limiti di forza in funzione del tempo

Il sistema di *collision detection* di ABB limita l'area del grafico: nel caso in cui venga vista una forza superiore o prolungata nel tempo il controller genera uno stop di sistema. Per poter gestire le collisioni, obiettivo principale della ricerca svolta in azienda, è stato necessario intercettare tale segnale.

È stato necessario creare una sorta di cross-connection. All'interno della parte di configurazione del sistema, nella parte relativa alla gestione dell'IO si è generato un segnale in uscita dal sistema collegato allo status di

Motion Supervision Triggered, per entrambi i bracci del cobot. In questo modo di può impostare sorvegliare lo stato di tale evento e collegarlo, come illustrato nel **CAPITOLO 5**, ad una gestione dell'errore.

Nella schermata in alto si può vedere che sono stati creati due segnali di sistema chiamati Robot_Collision_DX e SX collegati all'evento di supervisione di movimento. Ogni segnale viene definito, come visualizzato nella schermata sotto, con un argomento grazie al quale è possibile identificare l'unità meccanica da cui proviene l'evento interessato. Per unità meccanica si intende il braccio del cobot.

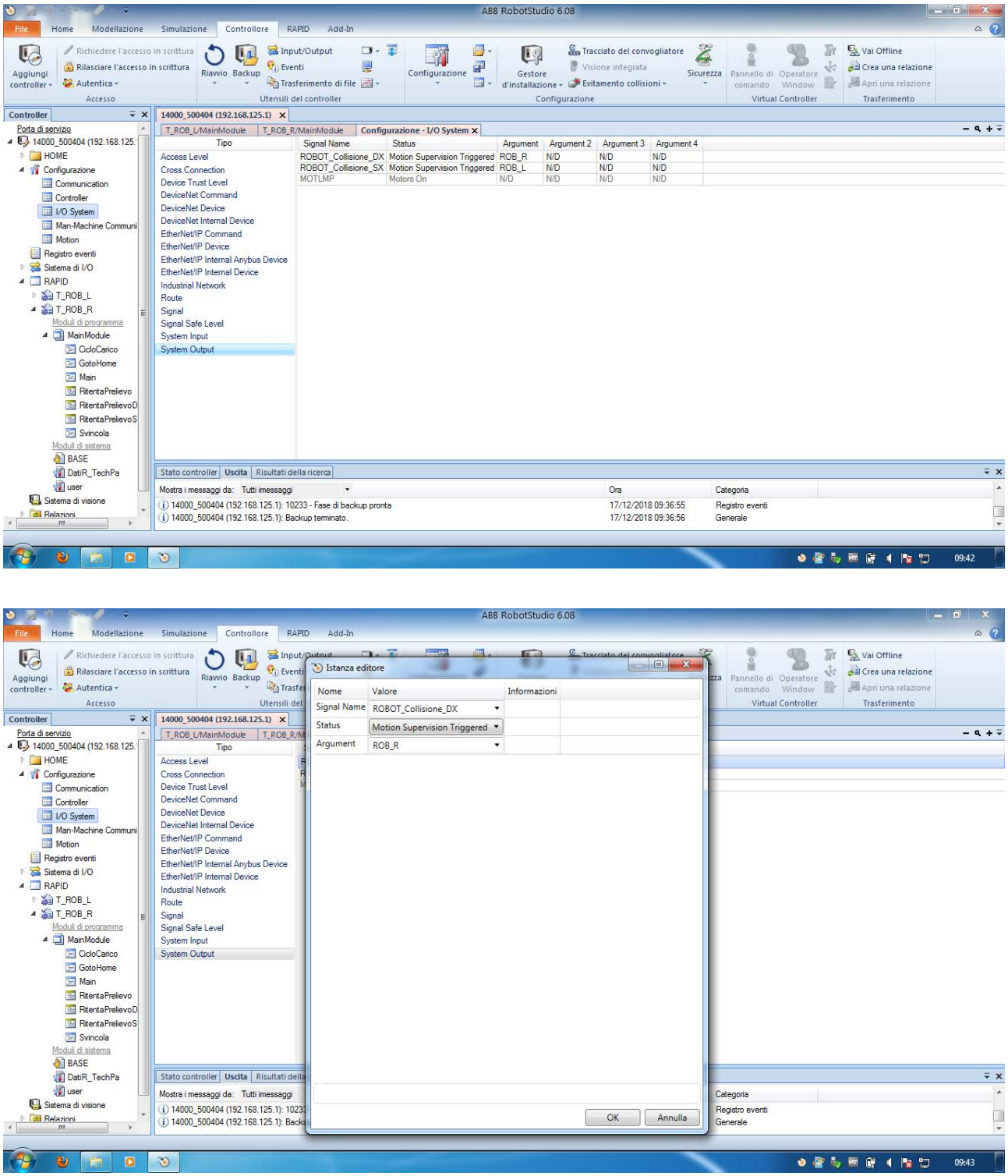


Figura 29 – Schermate del software RobotStudio dove si può vedere i due segnali definiti per ogni task e come vengono creati

3.4 L'istruzione Move

Per capire come ragiona il controller e capire il problema delle singolarità, che sarà affrontato nel capitolo quarto, è necessario analizzare la chiamata di sistema Move. Questa è la funzione base del robot, poiché è la chiamata che permette di impostare il movimento partendo dalla posizione attuale, conosciuta in quanto calcolabile dall'attuale posizione dei giunti, verso il target finale. L'importanza di questa istruzione è data dal fatto che molte altre chiamate di sistema si basano su di essa, per esempio le funzioni TriggL, TriggJ, MoveC, MoveLDO, ecc.

Definizione:

MoveL ToPoint , speed , zone , tool\WObj:=workobject;

Esempio:

MoveL Target, v1000, fine, Pinza\Wobj:=SistemaRiferimento1;

I parametri presenti nella chiamata di funzione Move sono:

- *ToPoint*, di tipo robtarget. Il punto di destinazione del robot e degli assi esterni. Viene definito come una posizione denominata o memorizzata direttamente nell'istruzione.
- *Speed*, di tipo speeddata. I dati di velocità applicati ai movimenti. Questi dati definiscono la velocità del TCP, il riorientamento dell'utensile e gli assi esterni. È possibile specificare la velocità del TCP in mm/s tramite un valore reale di tipo num direttamente nell'istruzione aggiungendo in questo campo [\V]. Attraverso la specifica [\T] si indica il tempo totale, in secondi, durante il quale il robot si muove. Successivamente, queste specifiche vengono sostituite con i dati di velocità corrispondenti.
- *Zone*, di tipo zonedata. È il dato che indica la zona di movimento attorno al target. I dati della zona descrivono la dimensione del percorso d'angolo generato. Nel caso in cui come nell'esempio riportato ci sia scritto *fine* il robot arriverà nel punto preciso con il TCP. Se invece viene scritto z50 il robot può anche non arrivare nel punto finale. Attraverso la specifica [\Z] si indica la precisione della posizione del TCP del robot direttamente nell'istruzione. La lunghezza del percorso ad angolo è specificata in mm. Con [\Inpos], in position, si può specificare i criteri di convergenza per la posizione del TCP del robot nel punto di arresto. Questo parametro ci aiuterà a capire anche il concetto di singolarità del robot. Per spiegarlo in termini più pratici si può pensare il target come il centro di una sfera e il dato zone indica il raggio di tale sfera: più il raggio è grande più il robot può muoversi su un percorso d'angolo all'interno della zona definita.
- *Tool*, di tipo tooldata. L'utensile in uso durante il movimento del robot. Il TCP (Tool Centre Point) è il punto spostato sul punto di destinazione specificato. A questo parametro viene aggiunto un altro elemento fondamentale cioè il work object, che tradotto letteralmente significa l'oggetto di lavoro, cioè il sistema di riferimento rispetto al quale il target è definito. Questo argomento può essere omesso

e, in tal caso, la posizione sarà correlata al sistema di coordinate universali. Se, al contrario, vengono utilizzati assi esterni coordinati o il TCP fisso, questo argomento deve essere specificato.

Si vuole ora analizzare con maggiore interesse alcune particolarità di questa istruzione grazie alle quali si può comprendere meglio come lavoro il controller e come risolvere eventuali problemi di movimentazione.

3.4.1 RobTarget, la struttura dati che indentifica la posa di un robot

Il *robtarget* è una struttura presente nel linguaggio rapid di ABB: viene utilizzata per definire la posa del robot e come devono essere configurati eventuali assi aggiuntivi esterni. La struttura è definita nel kernel in questo modo:

```
dataobject of robtarget {[ trans of pos(x,y,z),
                           rot of orient(q,q2,q3,q4),
                           robconf of confdata (cf1,cf4,cf6,cfx),
                           extax of extjoint (eax_a, eax_b, eax_c, eax_d, eax_e, eax_f)]}
```

La struttura è composta a sua volta da strutture dati di tipo num, cioè numeri reali. Per definire la posa del robot sono necessarie le prime tre sottostrutture; l'ultima serve per poter definire eventuali assi esterni

1. Translation: La posizione (x, y e z) del TCP (tool centre point) espressa in mm. La posizione viene specificata in relazione al sistema di coordinate oggetto corrente, compreso lo spostamento del programma. Se non viene specificato alcun oggetto di lavoro, viene applicato il sistema di coordinate universali.

2. Rot: rotazione; questo è un tipo di dato definito come orient. Indica l'orientamento dell'utensile espresso in formato di quaternione (q1, q2, q3 e q4). L'orientamento viene specificato in relazione al sistema di coordinate oggetto corrente, compreso lo spostamento del programma. Se non viene specificato alcun oggetto di lavoro, viene applicato il sistema di coordinate universali.

3. Robconf, ovvero robot configuration basato sul dato confdata.
   La configurazione degli assi del robot: (cf1, cf4, cf6, and cfx). Questa viene definita come il quarto di giro corrente dell'asse 1, dell'asse 4 e dell'asse 6. Il primo quarto di giro positivo, da 0 a 90°, viene indicato con 0. Il significato del componente cfx dipende dal tipo di robot. Grazie all'utilizzo del quaternione è possibile scrivere in modo più conciso la matrice legata alla rotazione. Gli assi dei sistemi di coordinate ruotati (x, y, z) sono vettori che possono essere espressi nel sistema di coordinate di riferimento nel modo seguente:

   $x = (x_1, x_2, x_3)$ $y = (y_1, y_2, y_3)$ $z = (z_1, z_2, z_3)$

Questo significa che il componente x del vettore x nel sistema di coordinate di riferimento sarà x1, il componente y sarà x2 e così via. È possibile inserire questi tre vettori in una matrice di rotazione, dove ciascuno di essi formi una delle colonne:

$$R = \begin{pmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{pmatrix}$$

Un quaternione consente semplicemente di descrivere in modo più conciso questa matrice di rotazione; i quaternioni vengono calcolati in base agli elementi della matrice di rotazione. Di seguito viene riportato il calcolo del valore assoluto e del segno dei quaternioni

$$q_1 = \frac{\sqrt{x_1 + y_2 + z_3 + 1}}{2}$$

$$q_2 = \frac{\sqrt{x_1 - y_2 - z_3 + 1}}{2} \qquad sign\ q_2 = sign(y_3 - z_2)$$

$$q_3 = \frac{\sqrt{y_2 - x_1 - z_3 + 1}}{2} \qquad sign\ q_3 = sign(z_1 - x_3)$$

$$q_4 = \frac{\sqrt{z_3 - x_1 - y_2 + 1}}{2} \qquad sign\ q_4 = sign(x_2 - y_1)$$

4. Extax, cioè external axes. Queste informazioni riguardano gli assi aggiutivi che possono esserci come no e indicano le posizioni di posizionatori o di manipolatori di pezzi. La posizione è definita come segue per ciascun singolo asse (eax_a, eax_b...eax_f): per gli assi rotanti, la posizione viene definita in base alla rotazione in gradi a partire dalla posizione di calibratura. Per gli assi lineari, la posizione viene definita in base alla distanza in mm dalla posizione di calibratura. Gli assi aggiuntivi eax_a ... sono logici. La relazione tra il numero dell'asse logico e il numero dell'asse fisico viene definita nei parametri di sistema. Il valore 9E9 viene definito per gli assi che non sono collegati. Il robot può controllare fino a sei assi aggiuntivi oltre ai suoi sei assi interni, per un totale di dodici assi. I sei assi aggiuntivi vengono indicati logicamente: a, b, c, d, e, f. Ciascun asse logico può essere collegato a un asse fisico e, in tal caso, il collegamento viene definito nei parametri di sistema. I dati di tipo extjoint vengono utilizzati per assegnare valori di posizione a ciascun asse logico da a ad f. Per ciascun asse logico collegato a un asse fisico, la posizione viene definita come indicato di seguito:
   - per gli assi rotanti, la posizione viene definita in base alla rotazione in gradi a partire dalla posizione di calibratura.
   - per gli assi lineari, la posizione viene definita in base alla distanza in mm dalla posizione di calibratura.

   Se un asse logico non è collegato a un asse fisico, il valore 9E9 viene utilizzato come un valore di posizione per indicare il mancato collegamento. Al momento dell'esecuzione, vengono controllati i dati di posizione di ciascun asse e verificato l'eventuale collegamento dell'asse corrispondente. Se i

valori di posizione memorizzati non sono conformi alla connessione effettiva degli assi, si applica quanto segue:

- se la posizione non è definita nei relativi dati (il valore è 9E9), il valore viene ignorato se l'asse è collegato, ma non attivato. Se l'asse è attivato si verificherà un errore.
- se la posizione è definita nei dati relativi, anche se l'asse non fosse collegato, il valore viene ignorato.

Non viene eseguito alcun movimento ma non viene generato alcun errore per un asse con dati di posizione validi, se l'asse non è attivato. Se si utilizza l'offset di un asse aggiuntivo (istruzione EOffsOn o EOffsSet), le posizioni vengono specificate nel sistema di coordinate ExtOffs. Se un asse aggiuntivo sta funzionando in modo indipendente e un nuovo movimento deve essere eseguito dal robot e dai suoi assi aggiuntivi, i dati di posizionamento per l'asse aggiuntivo in modo indipendente non devono essere a 9E9. I dati devono essere un valore arbitrario non utilizzato dal sistema.

3.4.2 Altre funzioni dell'istruzione Move, concorrenza di istruzioni e sincronismo

Come si è intuito la funzione Move risulta essere la chiamata di sistema più importante in quanto su di essa si basa il funzionamento del robot. La CPU esegue l'istruzione fino a che il TCP non si trova all'interno della sfera definita attorno al punto. Tale funzione però può provocare sovraccarichi della CPU nel caso in cui i target risultino essere troppo ravvicinati o nel caso in cui le aree definite intorno al target, tramite il parametro zone, si intersecano. Per risolvere questo problema di calcolo si possono eseguire le istruzioni in modo concorrente. Per poter fare ciò si deve inserire nell'istruzione il flag [\Conc] che indica la controller di prendere le successive istruzioni di tipo Move e calcolarle insieme in modo da definire una traiettoria ottimizzata e rispettando punti, zone e velocità.

Si deve porre attenzione al fatto che utilizzando l'argomento \Conc, il numero delle istruzioni di movimento in successione è limitato a cinque. In un paragrafo del programma che comprende *StorePath-RestoPath*, le istruzioni di movimento non sono consentite con l'argomento \Conc. Questa particolarità ha portato qualche difficoltà nella risoluzione delle traiettorie necessarie per la stazione robotica studiata: infatti tramite le istruzioni sopraccennate si è riusciti a ripristinare le funzionalità del robot e ripartire dal punto in cui il braccio robotico ha rilevato la collisione. Nella maggior parte dei casi il flag \Conc viene omesso e se ToPoint non è un punto di arresto, l'istruzione successiva viene eseguita prima che il robot raggiunga la zona programmata. Per questo motivo il parametro zone è fondamentale per definire il modo con cui il robot deve eseguire la sequenza di punti che definisco la traiettoria. Di questo se ne parlerà nel prossimo paragrafo.

Un ulteriore motivo per cui si è potuto utilizzare questo flag è che nei sistemi MultiMove, come lo è lo YuMi, non è possibile utilizzarlo utilizzato nel movimento coordinato sincronizzato.

Per movimento coordinato sincronizzato si intende che i due bracci robotici eseguono l'istruzione nel medesimo tempo e al medesimo momento: risulterà quindi che uno attende l'altro. Si vuole sottolineare la differenza tra movimento concorrente e sincronizzato: il concorrente, descritto sopra, è una sequenza di

istruzioni che vengono attivate quasi contemporaneamente e che a livello di controller vengono calcolate insieme. Questo vale per un singolo manipolatore.
Il movimento sincrono è quel movimento tale per cui due o più manipolatori, coordinati dal controller MultiMove, eseguono più l'istruzione nel medesimo tempo. Per poter eseguire movimenti sincroni è necessario definire il punto di partenza del sincronismo e come verrà illustrato successivamente è necessario richiamare altre funzioni. In questo paragrafo si vuole puntare l'attenzione sul significato del valore [\Id] che è possibile inserire all'interno dell'istruzione Move, come si può notare dall'esempio:

MoveL Target\ID:=10,Speed,z10,tool0\WObj:=wobj0;

Il synchronization id, di tipo identno, è l'argomento che identifica l'istruzione di movimento che deve essere specificato in un sistema MultiMove, se si utilizza un movimento coordinato sincronizzato con altri manipolatori: non è consentito negli altri casi. Viene considerato solo se viene attivata la funzione di sincronismo tramite la chiamata SyncMove\On. L'Id viene utilizzato dal controller MultiMove, che supervisiona il sistema sincrono, per poter mantenere la coordinazione tra i movimenti e per far in modo che i movimenti non vengano confusi in fase di runtime. Il tempo massimo di movimentazione viene definito dal più lento dei bracci meccanici e le velocità vengono regolate dal controller per poter eseguire l'istruzione nel medesimo tempo. Il numero di ID specificato deve essere uguale in tutti i task di programma cooperanti.

3.4.3 Tooldata, identificazione del tool e del carico da movimentare

L'istruzione di movimento viene definita anche tramite il parametro tool. Grazie alla struttura dati tooldata il controller può calcolare la posizione del TCP. All'interno della struttura sono definite anche altre informazioni che adesso verranno illustrate:
dataobject of tooldata {robhold,

[trans of pos (x , y , z), rot of orient (q_1 , q_2 , q_3 , q_4)],
tload (mass of num), cog of pos (x , y , z), aom of orient(q_1 , q_2 , q_3 , q_4), I_x , I_y, I_z]}

La struttura è composta da tre informazioni necessarie per poter gestire al meglio traiettorie e movimentazioni, ed è composta da un booleano e due strutture, quella di posizione e la struttura loaddata. La prima, *robhold*, è un booleano che indentifica se il tool è sorretto dal robot oppure no. La seconda riga identifica la posizione del TCP, tramite informazioni già note poiché compaiono anche nella definizione di robtarget. Successivamente si passa alla dichiarazione della massa e delle grandezze collegate, cioè la posizione del baricentro e le inerzie del tool.
Queste informazioni possono essere definite utilizzando dati ricavabili dai modelli CAD oppure si possono definire utilizzando delle funzioni definite all'interno del controller, e richiamabili tramite la flex pendant, che permettono l'identificazione del carico e la posizione del TCP.

Per quanto riguarda la prima funzione, LoadIdentify, il robot esegue varie movimentazioni ruotando gli assi del polso sferico, cioè il 4, 5 e il 6, per poter misurare tramite i sensori di coppia presenti sugli assi la forza che il carico libero esercita sul motore. Il processo viene eseguito a varie velocità in modo da eccitare con range di frequenze diverse il sensore di coppia. Una volta eseguita la procedura, che dura circa 10 minuti, il sistema salva il data come loaddata all'interno del file di sistema.

Per poter identificare il TCP invece si deve procedere in maniera differente. Si deve prendere un riferimento esterno, detta contropunta[16], e muoversi attorno alla punta con una punta montata sul manipolatore. Per definire il TCP si possono utilizzare 4 o 9 punti in base al grado di precisione che si vuole ottenere. Già con l'utilizzo di quattro punti si arriva ad avere una precisione attorno a qualche centesimo di millimetro.

Grazie a queste funzioni è possibile ottenere dati con un'accuratezza eccezionale. Questi tool integrati nel sistema vengono utilizzate sempre perché, come si sa, la realtà ha sempre un certo grado di incertezza rispetto ai modelli CAD.

Il robot può lavorare con delle strumentazioni per lavorare un oggetto oppure deve eseguire dei pick and place: In questo caso può movimentare pesi che possono compromettere la precisione, la ripetibilità e l'accuratezza del robot. Come si è visto nel capitolo due, dove si analizzato lo YuMi, ogni robot ha un carico massimo che può trasportare e nella Figura 16 - Diagramma di payload senza pinza e con pinza del capitolo due, il payload che il polso può sopportare si riduce all'aumentare dalla distanza dalla flangia sia nella direzione perpendicolare che nelle altre due direzioni.

La presenza della pinza attaccata al polso riduce notevolmente il carico del sistema, per cui un ulteriore carico può far superare al robot il limite. Se viene superato di poco, e con poco si intende ordini di grandezza inferiori a quello definito come massimo, riesce comunque ad eseguire tutte le movimentazioni. Se il carico risulta essere superiore di molto allora i sensori di coppia rilevano forze eccessive e il robot non si muove e rimane fermo in emergenza.

Nell'istruzione Move è possibile introdurre il carico aggiuntivo grazie all'attributo [\TLoad] cioè il carico totale utilizzato nel movimento. Il carico totale rappresenta il carico dell'utensile assieme al carico utile trasportato dall'utensile stesso.

Un'alternativa per poter aggiungere un carico durante l'esecuzione del task è l'istruzione GripLaod grazie alla quale si va ad identificare la massa dell'oggetto sollevato che viene sommata al carico della pinza.

[16] Per contropunta si intende un semplice oggetto che possiede una punta rivolta verso l'alto. Si dice contropunta poiché per punta, in gergo robotico, si intende un oggetto appuntito che è possibile fissare sulla flangia del robot, oppure sulla pinza, che permette di definire sistemi di riferimento reali. Come detto precedentemente il simulatore è un potente strumento per poter analizzare la fattibilità di una stazione. Come si sa la realtà è sempre diversa e anche se vengono posizionati tutti gli oggetti nel modo più accurato possibile risulta esserci sempre un errore: per cui risulta necessario prendere tutti i sistemi di riferimento sul sistema reale attraverso l'uso di un puntale lavorato.

3.4.4 Logica di calcolo del controller e utilizzo delle informazioni definite nell'istruzione

Nei precedenti paragrafi ci si focalizzati nell'illustrazione dell'istruzione Move per poter ora spiegare come il controller esegue l'istruzione e come calcola il movimento dei giunti. La trattazione sarà di tipo discorsiva in quanto non è lo scopo di questa tesi, ma risulta essere necessaria per poter poi capire evitare diverse problematiche.
Per semplificare la gestione dei movimenti e la definizione dei target di interesse per il programmatore si ragiona attraverso l'utilizzo di sistemi di riferimento, i work object. Il controller identifica questi sistemi come delle matrici di posizione e rotazione, cioè delle matrici 6x6.
Ogni target è definito rispetto ad un sistema di riferimento: l'ausilio di sistemi di riferimento sono utili per poter gestire il posizionamento degli oggetti tramite degli offset impostabili da pannello o per poter muovere il robot lunga una traiettoria rettilinea ad esempio una saldatura.
Per poter eseguire la movimentazione il controller deve elaborare i dati per poter anzitutto recuperare il punto finale. Per poter fare ciò il controller deve eseguire una prima elaborazione tale da poter riferire il target rispetto al work object wobj0, tenendo presente la matrice legata al TCP.
Per semplificare il calcolo, poiché si dovrebbe anche risolvere la conversione dei quaternioni, si consideri il vettore $P_w := (x_w, y_w, z_w, q_{1w}, q_{2w}, q_{3w}, q_{4w})$ che indica il punto finale definito rispetto al suo work object, la matrice T che contiene i dati del tool, e la matrice W che indica il sistema di riferimento.
Per poter arrivare al vettore $P_{w0} := (x_{w0}, y_{w0}, z_{w0}, q_{1w0}, q_{2w0}, q_{3w0}, q_{4w0})$ dovrà eseguire questo calcolo:

$$P_{w0} = [W]^{-1} * [T]^{-1} * P_w$$

In questo modo il controller arriva ad avere la posizione della flangia rispetto al suo sistema di riferimento. A questo punto però si ha sempre le coordinate dello spazio cartesiano mentre la movimentazione, mascherata dalle funzioni del kernel, si sviluppa nello spazio dei giunti. Per poter arrivare a questo punto è necessario l'ausilio della cinematica inversa. Questo ultimo passaggio è il punto critico che verrà trattato più approfonditamente nel capitolo quarto dove si illustrerà il problema delle singolarità.
Una volta risolto e trovato la configurazione finale rispetto all'attuale posa del robot, il controller deve calcolare la traiettoria che può essere lineare oppure di giunto: questa informazione condiziona notevolmente il movimento in quanto non è detto che possa arrivare al punto finale poiché potrebbe insorgere limiti meccanici. Con questa informazione la CPU cerca di imporre la traiettoria che minimizza un determinato indice: nel caso del movimento di giunto il controller esegue l'istruzione utilizzando meno giunti possibili, mentre nel caso lineare è obbligato ad utilizzare tutti gli assi per poter seguire una linea. Questo è possibile utilizzando le informazioni di velocità e configurazione. Per poter eseguire la traiettoria il controller deve imporre delle configurazioni che il robot deve eseguire e che devono essere scelte in modo tale da massimizzare l'indice lungo il percorso, tenendo però in conto due fattori. Il primo è il fatto che tra due configurazioni consecutive il robot non deve transitare in una zona in cui uno o più giunti superino i limiti di giunto o in zone

nelle quali la manipolabilità è nulla[17], cioè in zone in cui l'indice totale assume valore pari a zero; il secondo è il fatto che il robot deve avere tempo sufficiente per ruotare i suoi giunti per assumere quella configurazione. In particolare, si ha che la rotazione permessa al giunto vale:

$$q_{permesso} = \frac{velocità\ massima\ del\ giunto}{tempo}$$

Il tempo viene definito come:

$$tempo = \frac{spazio\ da\ percorrere}{velocità\ del\ movimento}$$

Quindi la rotazione permessa risulterà

$$q_{permesso} = \frac{velocità\ massima\ del\ giunto * velocità\ del\ movimento}{spazio\ da\ percorrre}$$

L'ultima informazione importante per eseguire il movimento è quella di zone: questa informazione, come detto precedentemente, serve per definire un volume sferico intorno al punto definito nell'istruzione. Nel caso in cui il punto sia di arresto l'istruzione termina non appena il controller vede che la posa definisce un target appartenente alla sfera, mentre nel caso in cui il punto serva per definire un passaggio, in modo ad esempio di evitare ostacoli, il parametro zone viene sfruttato per definire il percorso d'angolo: la traiettoria viene calcolata in modo da gestire, in funzione di accelerazioni e velocità, una curva tale da raccordare nel miglior modo possibile il movimento eseguito dal robot. Più il parametro ha valore elevato più la legge di moto nell'intorno del punto è dolce: la movimentazione tiene conto anche delle inerzie presenti dell'end effector, dati che sono definiti all'interno del tooldata e che possono essere ricavati grazie al servizio LoadIdentify.
L'istruzione Move risulta essere quindi l'istruzione base per la movimentazione del robot: la complessità del calcolo risulta essere particolarmente onerosa nel caso in cui il robot si muova vicino ai limiti dell'area di lavoro, se i target risultano essere vicini[18] tra loro o nel caso in cui il robot passi per un punti di singolarità. In questi casi il controller, che deve garantire una soluzione entro un tempo massimo, in quanto lavora in realtime, non riesce a converge ad una soluzione.

3.5 DESCRIZIONE DEL PROGRAMMA SVILUPPATO

Il programma è stato allegato all'*Appendice A* allegata al testo. Nei prossimi paragrafi si analizzeranno alcune parti funzionali del programma necessarie per poter gestire al meglio la stazione e rendere completamente automatiche le varie procedure e i vari errori di sistema.

[17] Per manipolabilità nulla
[18] Per target vicini si intende che i volumi sferici si sovrappongono oppure che, a causa delle velocità e delle accelerazioni impostate, il controller non riesca a trovare una soluzione per la pianificazione delle legge di moto entro i limiti di movimentazione del motore.

3.5.1 La Procedura Main

La procedura Main è la prima procedura che il controller richiama: all'interno di questa è stato definito il ciclo che il controller deve gestire e far eseguire al cobot.
Il ciclo di lavoro è stato suddiviso in più procedure:

- le funzioni prelievo e deposito cioè la normale esecuzione del job del robot, definita all'interno della procedura chiamata CicloLavoro
- *l'error handler*, descritto più approfonditamente nel quinto capitolo, grazie al quale è possibile gestire tutta una serie di errori e dove si è sviluppato, in maniera semplice dato il caso di studio, la gestione degli errori
- le funzioni *trap* che intervengono nel momento in cui viene segnalato al robot un segnale interrupt

Nei successivi paragrafi e capitoli verranno illustrate più approfonditamente le funzioni più interessanti che hanno permesso la realizzazione del programma del cobot: l'intero script è stato riportato nell'appendice A. L'errore handler verrà illustrato nel capitolo quinto mentre mentre si illustrerà la parte legata alla procedura definita CicloLavoro nel prossimo paragrafo e le routine trap con le quali è possibile ritentare il prelievo della confezione non presa. In questi sotto paragrafi vengono spiegate alcune istruzioni inserite per poter gestire al meglio il task associato ad ogni braccio del cobot.

3.5.1.1 Reset dei segnali e procedura di Home

Per garantire l'esatta funzionalità del tool e del cobot, nonché la logica implementata nel PLC per gestire i nastri trasportatori, si deve anzitutto inizializzare il programma con delle funzioni di reset dei segnali utili e una procedura che riporti il cobot in una posizione di Home, servono, quindi, per poter resettare il ciclo e poter ripartire in fase il cobot con tutto il sistema.
Una volta eseguito il reset dei segnali utili, raggruppati in una procedura a parte, che viene richiamata unicamente in questo punto, si passa alla movimentazione del robot. Non conoscendo la posizione effettiva del braccio del robot è bene recuperare la posizione attuale grazie alla chiamata CRobT, istruzione richiamata nella procedura di Svincola. Questa chiamata di sistema va a leggere, grazie agli encoder presenti sui giunti del robot la posizione che verrà in seguito convertita una posizione robtarget.
La procedura *Gotohome* ha lo scopo di riportare il cobot nella posizione di Home per ogni braccio che corrisponde alla posizione di attesa delle confezioni. Questa posizione è una posizione che possiamo definire *"safe"* in quanto non è di ingombro né per lo scatolone che può arrivare sotto ai bracci del cobot né per le confezioni.

3.5.1.2 Inizializzazione degli interrupt e trap routine

```
VAR intnum sig1int;                          definita nel modulo di sistema
IDelete sig1int;                             all'interno del Main
CONNECT sig1int WITH RitentaPrelievo;
```

```
ISignalDI PLC_NonPrelevato, 1, sig1int;
```

Figura 30 - Inizializzazione di un segnale di interrupt e collegamento con trap routine

Con l'istruzione IDelete è possibile cancellare la memoria associata di una variabile definita all'interno del modulo di sistema Dati. La variabile di tipo *Intnum, interrupt numeric,* viene utilizzata per identificare un interrupt. Attraverso questo identificativo, tramite il comando CONNECT … WITH … è possibile associare la variabile alla funzione ad una trap routine.

Questi comandi servono per poter attivare la funzione, ma deve essere collegata ad un segnale fisico. È possibile collegarlo sia a segnali di input che di output: nel nostro caso il segnale sarà di tipo input. Infatti il PLC, dopo che il robot ha tentato il primo prelievo, controlla che effettivamente l'operazione abbia avuto successo. Nel caso in cui non abbia avuto successo il robot vede in ingresso il segnale. Attraverso la definizione ISignalDI si può collegare un segnale digitale d'ingresso ad un intnum e viene collegato nel caso in cui il valore del segnale sia alto, indicato con 1 nell'istruzione.

Nel Rapid l'interrupt può essere connesso una sola volta alla trap, mentre vari interrupt possono essere connessi alla stessa trap.

Sono state create tre routine trap, una nel caso in cui il robot non riesca a prelevare nessuna confezione, una che indica il prelievo della confezione che è rimasta in posizione, mentre l'altra rimane in attesa con la confezione in "mano".

Per evitare di creare un'infinità di variabili da collegare alla trap, così da non appesantire il programma[19] e mantenere spazio libero, si è costruita una sequenza di variabili, riportata di seguito.

In questo modo si risolve il limite imposto per definizione del kernel Rapid di ABB e si ottimizza il programma.

Per poter resettare la variabile e riutilizzarla, per prima cosa controlliamo che il segnale proveniente dal PLC sia tornato a 0: limitando nel tempo il controllo. Se così non fosse il robot invia il segnale di svuotamento della linea di prelievo, andando a segnalare un errore all'operatore.

```
!RESET INTERRUPT PER POTER RIUTILIZZARLO
WaitDI PLC_NonPrelevatoDX,0\MaxTime:=1\TimeFlag:=bmaxtime{10};
IDelete sig2int;
CONNECT sig2int WITH RitentaPrelievoDX;
ISignalDI PLC_NonPrelevatoDX, 1, sig2int;
```

Figura 31 - Reset segnale di inteerupt alla fine della trap routine associata per poterlo riutilizzare

Dopo l'IF viene reimpostato il segnale andando a ridefinire la connessione tra variabile e trap e tra input e variabile, come descritto precedentemente. Questo è possibile solo nel caso in cui il segnale è ritornato basso: Se non si aspetta che il segnale ritorni a livello 0, la variabile associata non può essere rilasciata e quindi non è possibile utilizzare l'istruzione IDelete per cancellarne il contenuto e così riassegnare alla variabile intnum il segnale di interrupt.

[19] A livello di programmazione è sempre bene ripulire le zone di memoria in modo da alleggerire il software ed ottimizzarlo. Creare nuove variabili significa allocare spazio in memoria, appesantendo il programma.

3.5.1.3 Gestione informazioni e casi possibili

La main procedure prosegue con il passaggio di informazioni del tipo di scatolone e confezioni. La linea ha la possibilità di caricare su due tipi di scatoloni e ogni scatolone ha un tipo di confezione.

In particolare, la differenza principale è il numero di confezioni da depositare in altezza, che possono essere tre o quattro.

Il robot riceve dal PLC la tipologia di scatolone e, in base a quello, recupera le informazioni definite nel modulo di sistema *Dati*. Successivamente, richiede che venga posizionato uno scatolone e attende una risposta dal PLC che indica se effettivamente è presente. Se lo scatolone è effettivamente in posizione di carico, allora il robot può procedere con le istruzioni successive.

Iniziano poi due cicli for annidati per poter gestire le posizioni di deposito: in questi due cicli vengono fatti dei controlli continui, in particolare sul funzionamento della linea di confezionamento, sulle posizioni di prelievo delle confezioni e sulla presenza dello scatolone. Se queste informazioni sono vere il robot esegue il ciclo di carico.

Nel caso in cui le confezioni non siano presenti sulla linea lo YuMi rimane in attesa per un determinato tempo e se supera un tempo massimo, impostato con \MaxTime nella riga d'istruzione seguente, viene rimandato nella posizione del programma P1.

```
WaitDI PLC_ConfezioniInPosizione,high\MaxTime:=1\TimeFlag:=bmaxtime{2};
IF bmaxtime{2}=TRUE THEN
bmaxtime{2}:=FALSE;
GOTO P1;
ENDIF
```

Figura 32 - Controllo del tempo di attesa di un segnale proveniente da PLC

In questo modo si evita che il robot possa rimanere in attesa di un segnale che non può arrivare e ricontrolla i vari casi.

Un altro caso che è stato scelto di inserire tra i possibili è quello riguardante l'assenza dello scatolone: in questo caso il cobot ritorna in home con una velocità ridotta e poi invia un segnale al PLC chiedendo una svuota linea scatolone. Quindi rimanda alla posizione P0 andando poi a richiedere uno scatolone sotto la posizione dello YuMi.

Ritornando alla situazione ottimale, una volta che il robot ha finito di caricare le confezioni invia il segnale di carico completato: dopo questo segnale il robot rimane in attesa, in posizione di home, dello scarico dello scatolone.

Essendoci un ciclo esterno a tutte le casistiche che è impostato sempre a true, cioè viene eseguito continuamente, il controller continua ad eseguire le istruzioni scritte all'interno, per cui ritornerà a richiedere uno scatolone per poter caricare le confezioni.

3.5.1.4 Error Handler

La gestione degli errori viene posizionata alla fine della procedura di main e viene impostata inserendo l'istruzione ERROR. In questo modo il controller sa che entrerà in questa parte della procedura solo nel caso in cui si verifichi un errore.

Per l'applicazione è stato pensato di gestire l'errore di collisione, che come descritto nei capitoli precedenti, può avvenire nel caso in cui il robot vada a scontrarsi con qualche operatore, così da non causare danni.

Poiché i movimenti del robot sono eseguiti in modo sincrono, si è voluto separare la gestione dell'errore. Questo è stato possibile, come descritto nel capitolo 5, grazie alla definizione di un segnale di System Output, che il controller intercetta, legato ad entrambi i tasks.

Per poter discriminare il movimento, come verrà spiegato anche successivamente, è stato necessario creare un registro, legato all'ID di movimento. Attraverso queste due casistiche vengono eseguiti movimenti diversi a seconda del braccio.

Nel caso in cui il braccio non sia andato in collisione, rimarrà fermo in attesa che la condizione, definita tramite una variabile booleana ROBOT_Collisione_DX_ok venga posta a vera.

3.6 Ciclo di carico

Il ciclo di carico è effettivamente il cuore del programma in quanto sono presenti le istruzioni di movimento vero e proprio del cobot. La posizione di Home è la prima che compare nella procedura perché, come verrà illustrato nel paragrafo dedicato alla procedura definita *Svincola*, si vuole che ognuno dei bracci robotici si posizionino in attesa in una zona fuori ingombro. È necessario fare in modo che il cobot non sia in una condizione che può creare interferenza o intralciare il funzionamento della linea, per cui la posizione di Home risulta essere la posizione di partenza per tutti i task che deve compiere e rimarrà in attesa delle condizioni favorevoli per iniziare il ciclo.

3.6.1 Descrizione della procedura di prelievo e deposito

La procedura contiene le istruzioni di movimento, cioè le vere azioni che deve compiere ogni braccio. Come accennato precedentemente, lo Yumi esegue due task, ognuna dedicata ad un braccio. Ogni braccio quindi disegnerà nello spazio una traiettoria tale da evitare ostacoli ed eseguire le azioni desiderate. Nel paragrafo 3.4 **L'istruzione Move**si è illustrata la chiamata di funzione Move: grazie a questa è possibile eseguire tutte le movimentazioni necessarie.

Il punto di prelievo risulta essere un punto fisso: le confezioni, grazie all'utilizzo di fermi meccanici sulla linea di trasporto, si fermeranno sempre nel medesimo punto. Infatti, come si può leggere nell'Appendice A, il prelievo viene eseguito tramite un precedente punto di approccio, dove viene generato il vuoto tramite dei

Venturi. La funzione TriggL genererà il segnale definito nell'attributo trigg, chiamato in questo caso *tg_EVVuotoOn,* quando arriverà in una posizione nell'intorno del punto finale.
Una volta effettuato il prelievo il braccio ritorna per la medesima traiettoria in modo da evitare interferenze con i nastri o lo scatolone. Passando per il punto Home si prosegue con il deposito.
Il deposito deve avvenire inserendo le confezioni inclinate in modo da evitare le alette superiori dello scatolone. si deve porre attenzione che, a differenze del prelievo, il punto di deposito varia: come si può vedere dall'appendice A il punto di deposito è stato fatto variare utilizzando delle funzioni, *Offs*[20] e *Reltool*[21], che permettono di modificare il punto.
Gli offset del punto vengono definiti con i valori Xindex e Zindex questi vengono calcolati sulla base del doppio ciclo for che gestisce il carico dello scatolone.
Nelle istruzioni di movimento si può vedere che risulta essere inserito l'attributo ID: questo, come descritto nel paragrafo 3.4 **L'istruzione Move**, è necessario per poter sincronizzare la movimentazione dei due bracci meccanici. In questo modo, finché i robot depositano le confezioni è possibile caricare altre due confezioni da prelevare per poter ridurre il tempo ciclo.
La possibilità di utilizzare l'attributo ID è legata al fatto che il controller è un sistema MultiMove e viene attivato e disattivato dal comando SyncMove. Questa chiamata attiva il supervisore di controllo e verifica le tempistiche delle movimentazioni. La velocità viene gestita in modo tale da completare l'azione nel medesimo istante: in questo modo il supervisore controlla e garantisce che entrambi i bracci si trovino nel medesimo punto.
Al di sotto di ogni movimentazione è presente un aggiornamento del dato IdMove. Questo dato è un parametro che ci salviamo e che ritornerà utile nel CAPITOLO 5 quando verrà illustrata la logica per riprendere il ciclo dei bracci e recuperare anche il sincronismo della movimentazione.

3.7 Le routine Trap

Per trap routine si intende una funzione che può essere avviata ed eseguire un nuovo movimento temporaneo, ed infine riavviare quello originale. All'interno del modulo di sistema ce ne sono tre.
La trap ruotine può essere interpellata una tantum e questo è un limite della funzione. Nel nostro caso viene attivata ogni volta che si presente un segnale esterno proveniente dal PLC.

[20] La funzione Offs è una funzione che permette di modificare le coordinate del punto finale andando ad aggiungere i valori di offset, con segno, alle coordinate del punto. È definita in questo modo Offs(Target, OffsetX, OffsetY, OffsetZ) dove Target è l'obiettivo e le altre tre componenti sono valori reali da sommare alle coordinate X,Y,Z del punto.
[21] La funzione Reltool, che può contenere a sua volta la funzione Offs, modifica l'orientazione del tool. è possibile modificare la rotazione attorno agli assi dell'utensile. Se si desidera modificare l'orientazione del punto, che si ricorda essere definita con l'utilizzo di un quaternione, è necessario convertire la rotazione, tramite la funzione EulerXYZ, in un quaternione. RelTool viene utilizzato in questo modo RelTool(Target, Off_RX, Off_RY, Off_RZ\Rz:=degrees). Il target può essere variabile e si possono impostare delle rotazioni direttamente negli offset, andando a modificare la posa del polso, oppure impostare sul singolo asse, durante al movimentazione, una rotazione del polso. Nel secondo caso il robot cercherà di muoversi direttamente con la configurazione desiderata, ad esempio ruotato di 5 gradi attorno all'asse x.

Nel sotto paragrafo *3.5.1.2* **Inizializzazione degli interrupt e trap**sono state illustrate delle righe inserite all'inizio dello script. Queste righe permettono la creazione di una cross connection[22] tra segnale del PLC e un segnale di tipo interrupt al quale è legata l'attivazione di una delle routine.

Per risolvere il problema della chiamata una tantum è stato scelto di cancellare la creazione della cross connection e di rigenerarla prima di uscire dalla routine.

Il PLC, che gestisce le linee, controlla se, dopo che il cobot gli invia il segnale di prelievo effettuato, siano presenti o no confezioni sulla linea. Nel caso in cui rimane una confezione o entrambe le confezioni, il PLC invia un segnale dicendo che sono ancora presenti confezioni. Il controller riceve questo bit e esegue la routine trap in modo da andare a prelevare nuovamente la confezione. Quando il segnale viene ricevuto dal controller il robot blocca la movimentazione, analizza la posa dei bracci grazie alla funzione CrobT(). In questo modo, e associando l'ID al quale ci si è fermati, è possibile ritentare il prelievo delle confezioni in modo da andare a depositare entrambe le confezioni, mantenendo così il sincronismo del sistema e non dovendo gestire cicli for differenziati. La routine, una volta completata, può ritornare al normale ciclo di lavoro del robot poiché, appena si entra nella routine viene richiamata la funzione StorePath grazie alla quale è possibile salvare la posizione del puntatore di sistema. Al termine della routine viene interpellata la funzione RestoPath che permette di ripartire, andando a leggere la posizione del cursore salvata precedentemente, dalla posizione del programma in cui si era bloccato il sistema.

[22] Per cross connection di intende un ponte vero e proprio tra due segnali. È possibile eseguire tale connessione impostando dalle pagine controller di RobotStudio e andando su Configurazione\IOSystem\ CrossConnection. In questo caso la cross connection è una sequenza che fa insorgere una routine: l'errore visto dal supervisore viene collegato ad un segnale sigint, cioè un segnale di interrupt di sistema, che inizializza, a sua volta, la routine Trap.

CAPITOLO 4

LA CINEMATICA DIRETTA E INVERSA: SINGOLARITÀ E CONFIGURAZIONI

4.1 La cinematica diretta e inversa dei robot correlata alle problematiche di singolarità e alla ridondanza

Un punto cruciale nell'analisi di un manipolatore robotico è la capacità di trasformare le coordinate nello spazio operativo in coordinate nello spazio dei giunti. Con il termine spazio operativo si intende lo spazio nel quale viene tipicamente specificata la posizione e l'orientazione del manipolatore e può essere pensato come uno spazio tridimensionale caratterizzabile con delle coordinate cartesiane e degli angoli che specificano l'orientazione dell'end effector. Per spazio dei giunti, invece, si intende lo spazio vettoriale nel quale sono specificate le variabili di giunto le quali vengono fissate in fase di pianificazione della traiettoria permettendo all'end effector del manipolatore di raggiungere una certa coordinata spaziale con eventualmente una specifica orientazione.

La cinematica diretta è la chiave di volta che permette il passaggio dalle coordinate dallo spazio dei giunti alle coordinate nello spazio operativo attraverso la nota relazione:

$$x(e) = f(q)$$

Dove: $x(e)$ è il vettore $(m * 1)$ delle coordinate nello spazio operativo;

q è il vettore $(n * 1)$ delle coordinate nello spazio dei giunti, dove *n* è il numero dei giunti;

f è la funzione vettoriale continua non lineare la cui forma è ricavabile conoscendo la struttura del manipolatore.

La relazione alla base della cinematica differenziale che permette il calcolo della velocità di traslazione e rotazione dell'end effector, conoscendo le velocità dei giunti costituenti la catena cinematica, è:

$$v_e = \boldsymbol{J}(q)\dot{}$$

Dove: v_e è il vettore $(m * 1)$ delle velocità nello spazio operativo;

$\dot{q}$ è il vettore $(n * 1)$ delle velocità nello spazio dei giunti;

$\boldsymbol{J}$ è la matrice Jacobina di dimensioni $(m * n)$.

La funzione vettoriale f è ricavabile dalla geometria del manipolatore e dalla tipologia dei giunti che compongo la catena cinematica; la matrice Jacobiana, analogamente è ricavabile noto il vettore delle variabili di giunto q.

Prima di procedere è opportuno fare un'osservazione sulle dimensioni dei vettori e delle matrici che sono state utilizzate per definire la cinematica. Si osservi che è indicato con n il numero di giunti di cui è composto il

manipolatore e, facendo riferimento a giunti rotoidali o giunti prismatici, n corrisponde anche al numero di gradi di libertà di cui è dotato il manipolare, dove ogni giunto conferisce un grado di libertà al robot.

Con m si indica, invece, la dimensione dello spazio operativo. Pertanto, risulta m = 2 con riferimento a traslazione su un piano bidimensionale, m = 3 nel caso di traslazione nello spazio e m = 6 se si indicano tutti i gradi di libertà di rotazione e traslazione nello spazio per garantire non solo un prefissato posizionamento ma anche una specifica orientazione dell'end effector nello spazio operativo.

Infine, si indica r con il numero di componenti dello spazio operativo che sono richieste per la realizzazione di uno specifico task.

Fissata questa convenzione si riporta ora un'efficace rappresentazione grafica che permette la comprensione della relazione tra lo spazio operativo e lo spazio giunti.

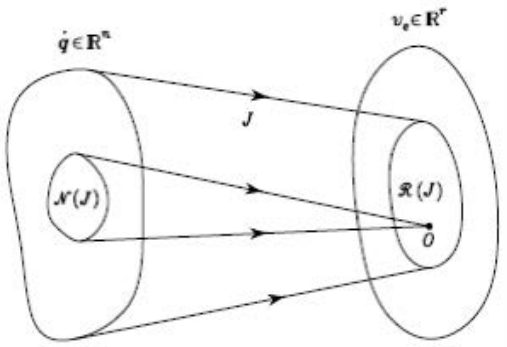


Figura 33 - Relazione spazio operativo e spazio giunti

L'equazione della cinematica differenziale può essere espressa mediante l'Immagine e il Kernel della trasformazione ricordando che: l'immagine di $\boldsymbol{J}$, che viene indicata con $Imm(\boldsymbol{J})$, è il sottospazio in $\mathbb{R}^r$ che individua le velocità dell'end effector che possono essere generate dalle velocità di giunto nella configurazione assegnata al manipolatore. Il kernel $\boldsymbol{J}$, che viene indicato con $Ker(\boldsymbol{J})$, è il sottospazio in $\mathbb{R}^n$ a cui appartengono le velocità di giunto che non producono alcuna velocità all'end effector nella configurazione assegnata al manipolatore.

Prima di proseguire si vuol ricordare la differenza tra lo Jacobiano analitico e lo Jacobiano geometrico che sono due strumenti molto importanti. La matrice $\boldsymbol{J}$, di dimensioni$(6 * n)$ riportata nell'equazione della cinematica differenziale e che pertanto lega la velocità nello spazio operativo alle velocità di giunto, è la Jacobiana geometrica definita in questo modo:

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{J_P} \\ \boldsymbol{J}_0 \end{bmatrix}$$

Dove: $\boldsymbol{J_P}$ è la matrice $(3 * n)$ che lega il vettore velocità di giunto alla velocità lineare $\dot{\boldsymbol{p}}_{\boldsymbol{e}}$ mediante la relazione $\dot{\boldsymbol{p}}_{\boldsymbol{e}} = \boldsymbol{J_P}(\boldsymbol{q})\dot{\boldsymbol{q}}$

$\boldsymbol{J}_0$ è la matrice $(3 * n)$ che lega il vettore velocità di giunto alla velocità angolare $\boldsymbol{\omega_e}$ mediante la relazione $\boldsymbol{\omega_e} = \boldsymbol{J}_0(\boldsymbol{q})\dot{\boldsymbol{q}}$

Tale matrice è detta Jacobiana geometrica perché per calcolarla è necessario adottare un procedimento geometrico.
La matrice Jacobiana Analitico si può ricavare mediante derivazione delle relazioni cinematiche rispetto alle derivate delle variabili di giunto. Pertanto, tale matrice si ricava derivando l'equazione della cinematica diretta:

$$\dot{x}_e = \begin{bmatrix} \dot{\boldsymbol{p}}_e \\ \dot{\boldsymbol{\Phi}}_e \end{bmatrix} = \begin{bmatrix} \boldsymbol{J}_P(q) \\ \boldsymbol{J}_0(q) \end{bmatrix} \dot{\boldsymbol{q}} = \boldsymbol{J}_A(\boldsymbol{q})\dot{\boldsymbol{q}}$$

E pertanto risulta:

$$\boldsymbol{J}_A(\boldsymbol{q}) = \frac{\partial \boldsymbol{k}(\boldsymbol{q})}{\partial \boldsymbol{q}}$$

In cui si ricorda che $\boldsymbol{k}(\boldsymbol{q})$ è la funzione vettoriale continua non lineare la cui forma è ricavabile conoscendo la struttura del manipolatore.

4.1.1 Le singolarità

Si definisce *configurazione singolare* di un manipolatore una configurazione dei giunti in cui lo Jacobiano geometrico non ha rango massimo. In tale condizione la dimensione dell'immagine dello Jacobiano diminuisce e allo stesso tempo aumenta quella del kernel in quanto vale sempre la relazione:

$$dim[Imm(\boldsymbol{J})] + dim[Ker(\boldsymbol{J})] = n$$

In una configurazione singolare è impossibile generare velocità in determinate direzioni e si ha una ridotta mobilità del manipolatore.
Esistono tre tipologie di singolarità:

- *Singolarità cinematiche:* sono dovute a particolari pose assunte dal manipolatore. In tali condizioni si ha la perdita di mobilità e le soluzioni del problema cinematico inverso sono infinite. In questa condizione ridotte velocità richieste nello spazio operativo possono richiedere elevate velocità nello spazio giunti.
- *Singolarità di rappresentazione:* sono quelle configurazioni in cui diventa singolare la matrice usata per la rappresentazione dell'orientamento, ovvero quella matrice che lega la velocità angolare e la derivata del vettore che esprime l'orientamento.
- *Singolarità algoritmiche*: si possono verificare quando si cerca di risolvere l'inversione cinematica con un sub-task.

Le singolarità di maggiore interesse per un manipolatore sono le singolarità cinematiche, in quanto, in tali configurazioni si riduce il rango della matrice Jacobiana rispetto al massimo rango. Nelle configurazioni di singolarità si riduce la mobilità della struttura, le soluzioni della cinematica inversa diventano infinite anche per manipolatori non ridondanti e nelle vicinanze di una condizione di singolarità piccole velocità nello spazio operativo possono causare elevate velocità nello spazio dei giunti.
Inoltre, le singolarità solitamente si distinguono ulteriormente in: *singolarità di bordo* e le *singolarità interne*.
Le *singolarità di bordo* si verificano in corrispondenza delle estremità dell'area di mobilità del manipolatore

e possono essere evitate molto più agevolmente rispetto alle *singolarità interne* che si verificano nello spazio operativo raggiungibile e sono solitamente causate dall'allineamento di uno o più assi del manipolatore.
Le più comuni singolarità interne di un manipolatore antropomorfo con polso monocentrico sono:

• Singolarità del polso, avviene quando l'asse 4 e 6 sono coincidenti, ovvero i due assi dei giunti in considerazione sono sovrapposti;

• Singolarità della spalla, avviene quando il centro del polso sferico incrocia l'asse del giunto 1;

• Singolarità del gomito, avviene quando si arriva alla frontiera dello spazio di lavoro.

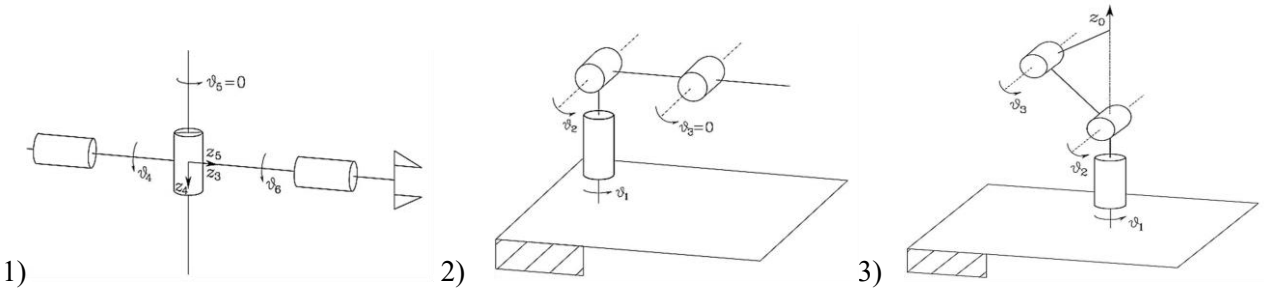


Figura 34 - Singolarità di Polso (1), di gomito (2) e di spalla (3)

Mentre le singolarità di spalla e gomito sono più semplici da gestire in quanto si possono gestire con la corretta pianificazione delle traiettorie, la singolarità di polso è più complessa. Buona norma per evitare il più possibile questa singolarità è utilizzare una pinza che ponga l'asse 4 perpendicolare all'asse 6. In questo modo i due assi sono nel punto più distante e danno un buon margine di movimento al robot. La singolarità di polso può avvenire in ogni punto dello spazio di lavoro e per questo motivo è più difficile da evitare. Alcuni manipolatori vengono strutturati con assi disallineati, ad esempio il manipolatore PUMA. In questo modo si possono evitare problemi di singolarità di gomito e spalla. Rimane comunque da affrontare il problema relativo alle singolarità di polso: alcune tecniche verranno spiegate nei prossimi paragrafi.

4.1.2 Ridondanza e self-motion

Un manipolatore robotico si definisce cinematicamente ridondante quando è dotato di un numero di gradi di libertà superiore a quello necessario per eseguire il compito assegnato ($n > r$). Pertanto, è bene sottolineare che la condizione di ridondanza è definita in funzione al task che deve essere svolto. Un manipolatore cinematicamente ridondante è solitamente più complesso da controllare però ha il considerevole vantaggio di poter eseguire un determinato task in modi diversi e questo ne concede maggiore flessibilità nei movimenti.
Si noti inoltre che un manipolatore ridondante è dotato di self-motion ovvero dalla possibilità di effettuare dei movimenti della struttura senza variare la posizione e l'orientazione dell'EE nello spazio sfruttando il cosiddetto spazio nullo.
Matematicamente, in base alle definizioni di immagine e spazio nullo data in precedenza, ogni qualvolta che il kernel $Ker(J)$risulta diverso dallo spazio nullo il manipolatore è dotato di gradi di ridondanza.

Si definisce pertanto grado di ridondanza la dimensione dello spazio nullo, ovvero, la differenza tra il numero di gradi di libertà del manipolatore e il numero di gradi di libertà necessari per l'esecuzione di un determinato task:

$$l = dim[ker(\boldsymbol{J})] = n - r$$

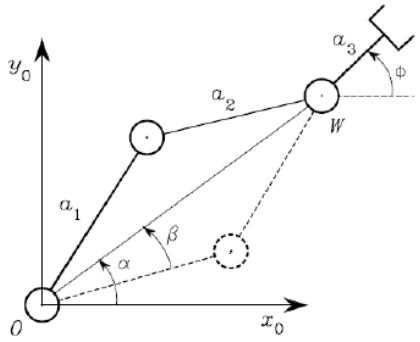


Figura 35 - Esempio di manipolatore planare ridondante che acquisisce la stessa posizione dell'end effector

Indicando con $\boldsymbol{q}^*$ la soluzione dell'equazione della cinematica differenziale e con **P** una matrice $(n * n)$ tale che:

$$\mathcal{R}(\boldsymbol{P}) \equiv Ker(\boldsymbol{J})$$

Anche il vettore velocità di giunto scritto:

$$\dot{\boldsymbol{q}} = \dot{\boldsymbol{q}}^* + \boldsymbol{P}\dot{\boldsymbol{q}}_0$$

Con $\dot{\boldsymbol{q}}_0$, vettore arbitrario di velocità nello spazio giunti e delle dimensioni opportune, si ottiene la soluzione dell'equazione della cinematica differenziale prima citata. Ed è proprio questo il vettore indicativo del fenomeno di self-motion.

4.1.2.1 Esempi di ridondanza

Un manipolatore viene detto ridondante da un punto di vista cinematico quando possiede un numero di gradi di libertà maggiore del numero di variabili necessarie alla caratterizzazione di un determinato compito.
In termini degli spazi dei giunti ed operativo, un manipolatore è intrinsecamente ridondante se $m < n$.
La ridondanza è peraltro un concetto relativo al tipo di compito da svolgere. Anche se $m = n$ un manipolatore può essere funzionalmente ridondante se interessano solo r componenti dello spazio operativo. Per esempio, un manipolatore planare a 3 bracci, dove $m = 3\, e\, n = 3$, se interessa anche l'orientazione dell'organo terminale $r = m = n = 3$, il manipolatore è non ridondante. Lo diventa se non interessa l'orientamento in quanto $r = 2\,, m = n = 3$.
Sicuramente al massimo $m = 6$ in quanto nello spazio si hanno 6 gradi libertà, tre traslazioni e tre rotazioni. Per cui un robot con n= 6 è non ridondante se $r = 6$. Lo diventa se ad esempio consideriamo una operazione di taglio laser con asse del fascio uguale all'asse di approccio[23]. In tal caso la rotazione attorno all'asse

[23] Con asse di approccio si intende uno dei 3 assi definiti, secondo la convenzione di Denavit-Hartenberg, in particolare all'asse perpendicolare alla flangia del polso.

d'approccio è inessenziale e dunque il manipolatore a sei assi risulta ridondante per tale compito, poiché $r = 5$.

La ridondanza serve per conferire caratteristiche di destrezza e versatilità, ad esempio per evitare ostacoli nello spazio operativo, continuando ad esercitare il compito assegnato. Il braccio umano ha 7 gradi di libertà per cui risulta essere una volta ridonante.

Considerando il manipolatore utilizzato per il caso applicativo studiato in questa tesi, lo YuMi possiede 7 gradi di libertà per braccio per cui risulta essere una volta ridondate a braccio.

4.1.3 Cinematica inversa

L'operazione di inversione cinematica permette di risalire al valore che bisogna imporre alle variabili di giunto per ottenere una specifica posa del manipolatore. Il problema principale di tale operazione, che risulta molto più complessa della cinematica diretta, è che le relazioni che esprime la posa del manipolatore nello spazio operativo e il vettore delle variabili di giunto sono non lineari. Un modo concettualmente adottabile per risolvere il problema è avvalersi dell'equazione della cinematica differenziale (1.2). Se si ipotizza $n = r$ è possibile ricavare il vettore delle velocità di giunto invertendo la relazione della cinematica differenziale e pertanto calcolare l'inversa della matrice Jacobiana:

$$\dot{\boldsymbol{q}} = \boldsymbol{J}^{-1}(\boldsymbol{q})\boldsymbol{v_e}$$

Da tale relazione, se è noto il vettore delle variabili di giunto all'istante iniziale mediante integrazione, è possibile calcolare lo stesso ad un qualsiasi istante temporale$\boldsymbol{t_f}$:

$$\boldsymbol{q}(t) = \int_0^{t_f} \dot{\boldsymbol{q}}(t)dt + \boldsymbol{q}(0)$$

Tale integrazione nel calcolatore è fatta per via numerica e può pertanto essere espressa mediante l'equazione:

$$\boldsymbol{q}(\boldsymbol{t}) = \boldsymbol{q}(t - \Delta t) + \dot{\boldsymbol{q}}(t - \Delta t)$$

indicando sempre con Δt l'intervallo di discretizzazione temporale delle variabili considerate.

Il problema di questo metodo risolutivo è che, per essere matematicamente valido, la matrice Jacobiana deve essere quadrata e non singolare ovvero con determinante diverso da zero. Tali condizioni non vengono soddisfatte nelle due condizioni particolari descritte nel paragrafo precedente:

- In condizioni singolari, perché in questo caso la matrice Jacobiana anche se quadrata ha determinante nullo;
- Nel caso in cui il manipolare sia ridondante, quindi la matrice Jacobiana non è quadrata.

Inoltre, ulteriore complicazione nell'applicare questo metodo di inversione dell'equazione della cinematica differenziale è che, se matematicamente questo approccio non presenta problemi, quando lo si applica a variabili discrete, dovendo ricorrere all'operazione di integrazione numerica, si introduce un fenomeno di deriva.

Esistono algoritmi per risolvere questi problemi di calcolo. Questi algoritmi tendono ad appesantire il calcolatore e quindi possono produrre degli stati di incertezza nell'esecuzione del task venendo utilizzati in condizioni particolari come per esempio quando il robot deve seguire una traiettoria obbligata all'interno dell'isola robotica per non andare in collisione con oggetti esterni.
Nello sviluppo del software ci si trova spesso a risolvere problematiche legate alle singolarità che spesso si riescono a risolvere gestendo le configurazioni dei giunti cercando di evitare le varie "bolle"[24] attorno i punti di singolarità.

4.2 Evitare le singolarità cinematiche tramite algoritmi

I due modi più comuni per evitare il problema delle singolarità del manipolatore sono:

- evitare la situazione in cui il rango della matrice Jacobiana si riduce
- pianificare la traiettoria in modo che il manipolatore non si disponga mai in una condizione di singolarità.

Un primo metodo per fornire una soluzione approssimata della cinematica inversa di robot non ridondanti intorno a delle configurazioni singolari è denominato *Singularity-robust inverse* (SRI). Il metodo appena citato è un metodo che si rifà alla teoria del controllo ottimo: infatti si basa nell'introduzione di un termine di regolazione nella matrice Jacobina per evitare il mal condizionamento della matrice in prossimità delle singolarità.
Questo metodo utilizza il Damped Least Squares DLS, *minimi quadrati smorzati*, per permettere un controllo del moto approssimato vicino alla traiettoria nello spazio cartesiano desiderata. Il metodo riduce la coppia applicata ma non permette l'effettivo raggiungimento dei punti singolari. Per compensare questi problemi è possibile adottare un approccio per il controllo del percorso ibrido che sia stabile nell'intero spazio operativo anche in prossimità delle regioni che contengono singolarità. La stabilità del metodo è verificabile utilizzano il metodo delle funzioni di Lyapunov multiple.
Un passo avanti è quello fatto con l'*on-line task modification method* (OTMM). Tale metodo ha l'obiettivo di evitare le singolarità di manipolatori ridondanti e non a livello della velocità. Con questo metodo non è necessario verificare, in fase di pianificazione della traiettoria nel caso di manipolatori ridondanti, se la singolarità è evitabile ma si effettua direttamente online una modifica della traiettoria impostata. Il problema nasce dal fatto che, in prossimità di una configurazione ridondante: gli algoritmi di cinematica inversa basati sulla matrice Jacobiana possono non funzionare: questo perché la matrice ha rango inferiore e quindi il vettore di velocità non può essere ottenuto mediante combinazione lineare dei vettori colonna della matrice Jacobiana.
Un'altra soluzione è quella classica del SRI, valida sia per robot ridondanti che non ridondanti, che sfrutta la matrice Jacobiana. Tale approccio consiste nel fissare una soglia, quando il determinante dello Jacobiano

[24] Per "bolle" si indica il volume sferico attorno al punto di singolarità. Anche se si è nell'intorno del punto il calcolatore non riesce a trovare una soluzione e quindi viene segnalato sulla flex pendant l'errore di traiettoria per singolarità.

scende sotto questa soglia si sostituisce la riga dello Jacobiano mal condizionata con il differenziale di uno tra i non tendenti a zero determinanti. Questo metodo dipende fortemente dalla forma della cinematica diretta.
Un differente approccio si basa sullo stabilire una condizione locale sufficiente per assicurare che il rango dello Jacobiano sia preservato. Il metodo definito forma normale presentato utilizza il classico algoritmo di Newton lontano dalle singolarità per generare la traiettoria mentre, in prossimità delle configurazioni singolari la traiettoria non è generata nello spazio operativo ma nello spazio dei giunti. Infine, le due o più parti della traiettoria vengono unite. Questo metodo però ha il problema dell'onerosità computazionale. Per un robot ridondante si possono evitare solo alcune configurazioni singolari, quelle definite evitabili, utilizzando un approccio off-line, tramite l'utilizzo di un simulatore, in fase di pianificazione della traiettoria.
Per evitare anche le singolarità definite inevitabili occorre sviluppare un approccio on-line. L'idea di base è circoscrivere le singolarità modificando direttamente la traiettoria nello spazio operativo.
Prima di arrivare ad utilizzare metodi di calcolo onerosi a livello computazionale si utilizzano metodi off-line e on-line: nello sviluppare le traiettorie si analizza la possibilità di muoversi nello spazio cartesiano grazie al supporto software di un simulatore dove è possibile caricare file CAD delle isole robotiche. Grazie a questo supporto, come detto precedente, si può valutare in prima istanza la capacità di eseguire la movimentazione e vedere se le traiettorie attraversano punti di singolarità.
Nel caso in cui non si riesca off-line a risolvere tale problematica si utilizza la possibilità di apprendimento manuale per modificare la traiettoria, inserendo i punti necessari per evitare la singolarità critica.
Oltre al metodo *Singular Value Decomposition* (SVD) altre strade percorribili per evitare le singolarità sono:

- l'algoritmo di Greville,
- il metodo dello spazio nullo
- il metodo della perturbazione.

Questi metodi hanno lo svantaggio di essere computazionalmente onerosi per risolvere la cinematica inversa. Le più recenti ricerche si concentrano sull'utilizzo delle reti neurali artificiali in modo da distribuire il calcolo e ottenere la soluzione della cinematica entro un tempo massimo.
Utilizzare una strategia di switch è importante se si vogliono migliorare le performance quando si risolve il problema cinematico inverso. L'idea di fondo è fare uno switch dal metodo della pseudo-inversa al metodo della perturbazione ottimale basandosi su una misura di manipolabilità[25]: nel caso in cui si è in prossimità della configurazione singolare tale valore tenderà a zero.
Altra soluzione alternativa al problema delle singolarità si basa sulla *minima norma infinito* della velocità di giunto per permettere al manipolatore di passare attraverso le singolarità senza eccessive velocità di giunto. L'idea di fondo di questo metodo è di pesare gli errori di tracciamento della traiettoria dell'EE e la norma delle velocità di giunto. Esso permette di creare continuità nella transizione tra configurazioni singolari e non. Alcuni autori propongono di esprimere il fattore di smorzamento come una funzione della misura di manipolabilità,

[25] La manipolabilità viene espressa mediante il modulo del determinante del prodotto dello Jacobiano per la sua trasposta, dove J indica la matrice Jacobiana $|\det J * J^T|$

in quanto, la manipolabilità è definita in base allo Jacobiano. Pertanto, la manipolabilità può essere usata come indice per indicare la vicinanza ad una configurazione singolare del manipolatore.
In ogni caso la maggior parte dei ricercatori che si concentrano sulla norma della velocità di giunto si basano sulla norma due. In realtà, utilizzare la norma infinito può essere più significativo, perché in questo modo si minimizza la massima tra le velocità di giunto.
Altri studi osservano che non esistono solo le singolarità cinematiche ma anche le singolarità algoritmiche. Il problema delle singolarità algoritmiche permane sia se si utilizza la ridondanza del manipolatore mediante il metodo della task-space augmentation sia se si considera il task dell'EE come primario e si aggiungano altri task con priorità inferiore. Come nel caso delle singolarità cinematiche anche quelle geometriche comportano alte velocità di giunto e soluzioni mal condizionate. La soluzione damped least-square (DLS) con filtri numerici rappresenta un buon compromesso tra l'accuratezza nel tracciare la traiettoria tipica della soluzione derivante dall'uso della pseudo-inversa e del trovare soluzioni con velocità di giunto raggiungibili tipico del DLS.
Infine, alcuni studi congiungono l'ottimizzazione della coppia con l'evitare le singolarità utilizzando un approccio basato sul DLS e considerando come indice di misura della distanza dalla condizione di singolarità la manipolabilità dinamica del manipolatore.

4.3 Evitare le singolarità cinematiche tramite funzioni presenti nel Rapid

Il linguaggio Rapid possiede delle funzionalità che permettono di evitare il problema delle singolarità.
Il modo più semplice per superare il problema di singolarità è modificare la traiettoria nell'intorno della zona interessata. Questo è possibile nel caso in cui non ci siano problemi di ingombri o interferenze o collisioni meccaniche con altri oggetti o nel caso in cui i movimenti non interessano lavorazioni sul prodotto finito. Nel caso di saldature o assemblaggi non è possibile modificare la traiettoria in questo modo. Il linguaggio Rapid offre una ulteriore possibilità, esposta precedentemente a livello teorico: quello di ridurre il numero di assi rispetto al numero di gradi di libertà nello spazio.
Tramite la funzione *SingArea\LockAxis4* è possibile bloccare l'asse quattro, in questo modo, attivandola nei tratti in cui sono presenti punti di singolarità, si può eseguire il movimento senza problemi. La funzione risulta utile nel caso di assemblaggi o traiettorie programmate dove il tool center point deve passare esattamente in un determinato di punto di singolarità. L'utilizzo di questa istruzione però vale per robot antropomorfi con 6 assi: la funzione serve per ridurre, in determinati percorsi, il numero di assi pari a 5.
Per quanto riguarda lo YuMi questa soluzione non è applicabile: questo è dovuto al fatto che il braccio è ridondante con $n > m$, dove n è il numero di assi del braccio del manipolatore e m è il numero di gradi di libertà nello spazio. In questo caso al CPU, per risolvere il problema delle singolarità, utilizza un metodo numerico cercando una soluzione al problema partendo dalle condizioni al contorno, ad esempio un metodo come Newton-Rhapson. Tale metodo non permette una soluzione certa poiché dipende dalle condizioni iniziali: per poter agevolare la soluzione, in modo da evitare blocchi della movimentazione dovuti a

sovraccarichi della CPU, vengono posti più target intorno al punto di singolarità. Quindi, sfruttando i parametri dell'istruzione Move, in particolare l'attributo *\Conc* e il parametro *Zone*, si può risolvere il problema e garantire che, entro il tempo massimo di calcolo, il controller abbia trovato una soluzione. Il problema delle singolarità si trova maggiormente nelle movimentazioni di tipo lineare, poiché può capitare che due o più assi si pongo in tale condizione. Al posto di eseguire la movimentazione lineare dal punto iniziale a quello finale si suddivide la traiettoria in diversi tratti: nell'intorno del punto di singolarità si pongono più punti. Si ricordi che nella definizione di *robtarget* è definita la configurazione con la quale il braccio robotico deve porsi. Si sostituisce a questo punto si sostituisce il movimento lineare con quello di giunto, cioè al posto di eseguire un MoveL si porrà il MoveJ. Attivando quindi l'attributo *\Conc*[26] la CPU cercherà una possibile interpolazione dei punti passando nell'intorno di tali target grazie all'impostazione del parametro zone. Questa è l'alternativa più veloce per risolvere il problema delle singolarità all'interno di una traiettoria.

Nel paragrafo 4.2 sono stati illustrati alcuni metodi applicabili per risolvere il problema di calcolo ma, come si è visto, risultano essere onerosi a livello computazionale. I metodi con cui è possibile risolvere le singolarità risultano essere numerici e sono necessari se si ha a che fare con robot ridondanti ($n > m$) o nel caso in cui si abbiano singolarità. Risultano essere più lente ma di facile derivazione in quanto richiedono il calcolo dello Jacobiano analitico della cinematica diretta e perché utilizzano metodi iterativi come il metodo di Newton-Rhapson o del gradiente.

[26] Si rimanda alla spiegazione nel paragrafo 3.4

CAPITOLO 5

SISTEMA DI SUPERVISIONE DEI MOVIMENTI E GESTIONE DELLE COLLISIONI

In questo capitolo si analizzeranno le opzioni introdotte da ABB per supervisionare i movimenti effettuati da un robot e le opzioni proposte per gestire le collisioni, opzioni che vengono introdotte nel controller attraverso schede aggiuntive. In particolare, si analizzerà come vengono supervisionati i movimenti e le collisioni.
Le collisioni sono scontri che un braccio robotico può avere con operatori o con oggetti nel suo spazio di lavoro. Risulta pertanto scomodo avere collisioni poiché vanno ad interrompere il ciclo dei movimenti. Oltre ad un problema di rallentamenti dei cicli, le collisioni possono provocare danni al robot e ai sistemi esterni al robot.

5.1 Supervisione dei movimenti e il rilevamento delle collisioni

5.1.1 Sistema di Supervisione dei Movimenti

Il sistema operativo RobotWare è dotato di una funzionalità definita *supervisione dei movimenti* e, durante l'esecuzione del programma, è di default sempre attiva per garantire l'integrità del robot. Per quanto riguarda l'IRB14000 il sistema di supervisione dei movimenti è legato al sistema di *collision detection*: questi due sistemi insieme consento di rendere collaborativo il robot e garantire la sicurezza per l'operatore e per il robot stesso. Quando viene rilevata una collisione, viene eseguita di default una sequenza di azioni:

- il robot si ferma nel giro di pochi millisecondi;
- diminuisce le forze residue spostandosi in direzione inversa per una breve distanza lungo il suo percorso;
- blocca l'esecuzione del programma un messaggio di errore;
- rimane fermo con i motori accesi in modo che in seguito all'intervento dell'operatore possa riprendere l'esecuzione, dopo che l'errore di collisione è stato riconosciuto.

La supervisione dei percorsi è sempre attiva sia in modalità automatica che in modalità manuale ed è utilizzata per prevenire danni meccanici dovuti al fatto che il robot si imbatte in un ostacolo durante l'esecuzione del programma.
Per poter mantenere attivo questo sistema di controllo di sicurezza bisogna anzitutto aver eseguito una calibrazione dei giunti. Il sistema di supervisione dei movimenti esegue un'analisi in tempo reale della posizione e della velocità di ogni singolo giunto e li confronta con il range impostato. I motori presenti sui giunti hanno a bordo degli encoder che riescono a calcolare la posizione attuale rispetto alla posizione zero definita nella calibrazione.

Durante l'installazione di qualsiasi robot la prima cosa che viene richiesta all'operatore da RobotWare è quella di portare in configurazione di calibrazione il robot ed eseguire la calibrazione, descritta nel paragrafo 2.4.1.
Al termine della calibrazione il controller salva, in un registro interno, i dati che gli servono per eseguire l'analisi di movimento. In questo modo, anche nelle successive accensioni, può sempre controllare la posizione riferendosi ai dati di calibrazione.
L'algoritmo di supervisione dei movimenti risulta uno strumento di protezione dell'integrità di ogni robot infatti, evita di generare forze eccessive sull'asse che arriva a fine corsa[27]. Nel caso in cui le forze fossero eccessive potrebbero causare problemi. Infatti, tutti i motori hanno un limite massimo di coppia erogabile controllato dal sistema. Il controllo della coppia verrà illustrato nel paragrafo successivo.

5.1.2 Rilevamento delle collisioni: Collision Detection

La *Collision Detection*, o rilevamento delle collisioni, è un'opzione di controllo avanzata che rileva automaticamente le collisioni. Questo rilevamento va a fermare, in pochi istanti, il robot riducendone la forza e indietreggiando lungo il percorso, per evitare di lascare il robot o l'utensile soggetto a pressione.
Questa opzione prevede un rilevamento da tutte le direzioni proteggendo, non solo l'utensile, ma anche i pezzi da lavorare o i componenti da assemblare e il robot stesso.
Essendo un'opzione integrata nello YuMi non richiede parti meccaniche aggiuntive o cablaggi, assicurando una maggiore affidabilità e riducendo i costi. In più, non avendo ulteriori accessori attaccati all'utensile, non si va a ridurre il carico massimo che può movimentare il robot e consente un migliore riorientamento dell'utensile.
Il rilevamento delle collisioni si basa sulla supervisione dei sensori di coppia presenti sui giunti: la collisione appare come un picco di forza visto su di un asse, come si può vedere dall'immagine.
C'è la possibilità di impostare il sistema modificandone i parametri. Normalmente non è richiesto ma la sensibilità può essere modificata tramite istruzione Rapid. La supervisione può anche essere disattivata completamente, permanentemente o temporaneamente. Questo può essere necessario quando le forze in gioco che agiscono sul robot sono elevate.
Questa funzionalità, oltre ad avere il vantaggio di non dover aggiungere componenti, cablaggi o parti meccaniche ha un altro punto di forza: viene eseguita insieme alla routine di servizio denominata *Load Identification*[28] andando così ad ottimizzare l'esecuzione del software. La *collision detection* prevede:

[27] Per *fine corsa di un asse* si intende il blocco meccanico che è presente su ogni asse. In realtà è possibile aggiungere questi blocchi per ridurre l'intervallo di rotazione degli assi nel caso in cui l'area di lavoro in cui deve stare il robot sia minore rispetto all'area che potrebbe coprire.
[28] La routine di servizio *Load Identification*, di cui si è accennato nel capitolo 3, rende il robot in grado di eseguire un'accurata identificazione dei dati di carico semplicemente scuotendo l'utensile. I dati che la routine può identificare sono massa, centro di gravità e momento d'inerzia. Per eseguire un'accurata identificazione del carico si deve aver inserito correttamente i dati dell'utensile. In caso è possibile eseguire la routine una prima volta con l'utensile a vuoto, per identificare il carico dell'utensile e successivamente rieseguire la routine con il carico. Per eseguire l'identificazione, il

- *Path Supervision* o supervisione dei percorsi, presente anche nel sistema base di *RobotWare*, in modalità automatica e manuale a piena velocità utilizzata per prevenire danni meccanici dovuti al fatto che il robot si imbatte in un ostacolo durante l'esecuzione del programma.
- *Jog Supervision* o supervisione dei movimenti a scatti, utilizzato per prevenire danni meccanici al robot durante il jogging.

Risulta estremamente utile in molte applicazioni e in svariate situazioni:

- quando si eseguono programmi per la prima volta e ci sono percorsi incerti;
- in caso di collisione causata dal movimento del robot nella direzione errata;
- quando una pinza viene chiusa per errore, per esempio a causa di problemi elettrici o errori di programmazione;
- nella situazione comune in cui vengono utilizzati i morsetti, il pezzo da lavorare in posizione nell'attrezzatura. Se i morsetti non sono chiusi, ad es. a causa di un errore umano, il rilevamento delle collisioni impedirà urti violenti con i morsetti;
- nelle applicazioni in cui sono presenti altri oggetti in movimento durante il lavoro (pallet impilati, trasportatori in movimento, eccetera);
- quando si fanno movimenti su pezzi di grandi dimensioni e l'operatore non può vedere gli utensili sul lato più lontano. Ad esempio, se un operatore sta cercando di insegnare un percorso per posizionare un lato della carrozzeria o un telaio in una stazione di serraggio, potrebbe non vedere se sta esercitando

---

robot sposta il carico dopo uno schema specifico e calcola i dati. Gli assi che si muovono sono 3, 5 e 6. Nella posizione di identificazione, il movimento per l'asse 3 è di circa 3 gradi in alto e 3 gradi in basso e per l'asse 5 è di circa 30 gradi in alto e 30 gradi in basso. Per l'asse 6 il movimento viene eseguito attorno a due punti di configurazione. Il valore ottimale per l'angolo di configurazione è 90 gradi. La funzione non può essere utilizzata quando è impostato il tool0 e/o il load0, cioè assenza d'utensile e carico.

Il carico dell'utensile deve avere questi prerequisiti per poter eseguire la routine:

- Lo strumento è selezionato nel menu di jogging.
- Lo strumento è montato correttamente.
- L'asse 6 è vicino all'orizzontale.
- Il carico dell'arto superiore è noto, se la massa deve essere identificata.
- Gli assi 3, 5 e 6 non sono vicini ai loro limiti di campo di lavoro corrispondenti.
- La velocità è impostata al 100%.
- Il sistema è in modalità manuale.

Il carico utile invece deve avere questi prerequisiti:

- Lo strumento e il carico utile sono montati correttamente.
- L'asse 6 è vicino all'orizzontale.
- Il carico dell'utensile è noto (eseguire *LoadIdentify* per lo strumento prima).
- Il carico dell'arto superiore è noto, se la massa deve essere identificata.
- Quando si utilizza un TCP in movimento, lo strumento deve essere calibrato (TCP).
- Quando si utilizza un TCP stazionario, l'oggetto di lavoro corrispondente deve essere calibrato (frame utente e frame dell'oggetto).
- Gli assi 3, 5 e 6 non sono vicini ai loro limiti di campo di lavoro corrispondenti.
- La velocità è impostata al 100%.

Il sistema è in modalità manuale.

pressioni elevate contro una zona della cella. Con il rilevamento delle collisioni, il robot percepirà la forza anormale e con la fermata preventiva impedirà danni alla cella;

- processi di saldatura. Quando termina il processo di saldatura ad arco, capita spesso che il filo si attacchi al pezzo da lavorare. Il sistema fermerà il robot prima di continuare il movimento assicurando che l'attrezzatura non venga danneggiata. Legato al problema della saldatura c'è anche la possibilità che la pistola si possa incollare. Questo problema può comportare la piegatura degli elettrodi della pistola o danni all'utensile mentre il robot si sposta al punto successivo. Con il rilevamento il robot si ferma dove è bloccato permettendo all'operatore di spostare manualmente il robot e sostituire la punta danneggiata, che impedisce ore di fermo per pistola o costi di sostituzione dell'utensile;
- asservimento della macchina. Problemi comuni nell'asservimento della macchina sono: parti bloccate, porte che non si aprono completamente, il disallineamento parziale, ecc. Ciascuna di queste situazioni può essere catastrofica per la pinza del robot. Con c*ollision detection*, il danno può essere evitato o almeno minimizzato, risparmiando così tempi di fermo e costi di sostituzione della pinza.

Come si può vedere dall'immagine, attraverso l'utilizzo delle ruotine di servizio, le forze scambiate nelle collisioni sono minori proteggendo così robot, utensile e ambiente circostante.
L'opzione di rilevamento delle collisioni è stata definita come una routine infatti, non presenta alcun sensore fisico aggiuntivo ma si basa sui sensori presenti negli assi e nei motori. Tramite i sensori la routine può leggere i dati andando così a controllare la coppia erogata dai motori. La routine crea uno storico dei dati letti che vengono poi analizzati e verifica che:

- la coppia attualmente erogata sia inferiore alla coppia massima;
- la variabilità della coppia, in particolare l'incremento, sia inferiore al valore massimo impostato.

La routine risulta sempre attiva così si ottiene un controllo in tempo reale delle coppie scambiate. Grazie a questo controllo software, in conclusione, si elimina la necessità di ulteriori accessori e pesi senza richiedere il collegamento al circuito di arresto di emergenza e non presenta problemi di usura. Inoltre, consente l'indietreggiamento automatico dopo la collisione e la regolazione dei valori.

5.1.3 Limitazioni della routine Collision Detection

Questa funzione ammette delle limitazioni poiché è una funzione collegata al controllo del sistema e alle coppie in gioco dei motori:

- il carico del sistema deve essere conosciuto esattamente poiché influenza il comportamento della funzione e anche per questo motivo la funzione coopera con la routine *Load Identification*;
- è attiva solo per gli assi del robot e non è possibile attivarla per assi esterni controllati del robot[29];

[29] Il controller IRC5 permette di collegare il robot con un asse esterno per attivare la sincronizzazione dei movimenti. Naturalmente questo collegamento prevede un invio di segnali legati solamente alla sincronizzazione, in particolare dei dati encoder, e non vengono trasmesse altre informazioni.

- il sistema deve essere disattivato quando il robot è montato su una traccia mobile. Quando si attiva il movimento della traccia mobile il sistema può rilevare collisioni a causa delle forze in gioco anche se non ci sono collisioni;
- quando viene attivata il sistema softmove[30].

5.1.4 Force Control e SafeMove2

Prima di passare alla gestione degli errori di collisione si vuole dare una breve panoramica su alcune opzioni, che possono essere aggiunte tramite ausilio di altre schede all'interno del controller, per gestire le forze in gioco.

Il *Force Control*, un controllo di tipo adattivo, si basa sui dati letti dal sensore di forza applicabile al TCP.

I robot tradizionali prevedono il controllo della posizione con percorso e velocità predefiniti. Grazie all'opzione Force Control, il robot reagisce all'ambiente circostante e devia dal percorso programmato o varia la propria velocità in base al feedback del sensore di forza. Il movimento così adattato da parte del robot è determinato dalla strategia scelta per adattarsi perfettamente all'applicazione specifica.

Risulta quindi una differente implementazione di controllo delle forze in gioco. Infatti, mentre la collision detection monitora i dati provenienti da ogni asse, il force control esegue un adattamento delle forze in gioco. In questo modo è possibile gestire al meglio istruzioni o movimenti che possono introdurre forze variabili con tolleranze ridotte.

Un'altra opzione è il SafeMove, introdotto precedentemente nel paragrafo ***2.7.3 Schede aggiuntive del controllore***. La casa costruttrice dello YuMi ha prodotto la prima versione nel 2008 introducendo un buon livello di gestione della sicurezza per l'operatore. Nel 2017 ha lanciato la versione SafeMove2 introducendo molte funzioni utili per la gestione della sicurezza del robot e degli operatori. In particolare, per quanto riguarda le forze, l'opzione SafeMove2 introduce la possibilità di gestire la tolleranza sulle forze. Per spiegare meglio il caso si pensi ad effettuare un cambio utensile con tool che presentano un attacco a baionetta per avere una sicurezza meccanica dell'incastro. In questo caso di deve esercitare una forza maggiore in quanto si deve vincere la forza elastica della baionetta. Questa opzione mette a disposizione molti tool gestibili online con RobotStudio: è possibile definire dei volumi[31] da controllare su cui si possono impostare delle tolleranze di forza. In questo modo è possibile superare momentaneamente i limiti massimi per poter consentire, nel caso illustrato nell'esempio, il cambio del tool.

[30]SoftMove è una routine di sistema che permette di diminuire la rigidità del robot in una direzione predefinita mantenendo principalmente l'originale comportamento nelle altre direzioni.
Questo è possibile modificando i parametri di smorzamento e elasticità lungo una direzione. La rigidità può essere ridotta anche lungo l'asse verticale compensando l'effetto della gravità. Questa opzione è utile per macchine di stampaggio a iniezione o una macchina di pressofusione ma anche quando ci sono variazioni nella posizione del pezzo o nella direzione specificata: la ruotine fa in modo che il robot segua queste variazioni ed evita una forte collisione.

[31] I volumi vengono definiti attraverso sfere o capsule, definibili delle dimensioni desiderate, applicabili agli assi o ai giunti del robot

5.2 Gestione personalizzata degli errori tramite *Error Handler*

La collisione può essere gestita tramite la *collision detection* come visto precedentemente. Un obiettivo posto per lo sviluppo della tesi è proprio quello di comprendere se è possibile gestire autonomamente gli errori del sistema e in particolare la collisione con gli operatori.
Quando avviene una collisione di default il robot si arresta e sulla teach pendant viene visualizzato il tipo di errore. Partendo da questo punto si è pensato di ricavare il valore numerico associato all'errore mostrato e poter creare un handler che intervenga in caso di errore, proprio come è possibile in altri linguaggi di programmazione.
Gestire gli errori è una parte molto importante a livello di programmazione perché è possibile eliminare dei fermi macchina che possono andare a rallentare il sistema.

5.2.1 Error Handling: gestire gli errori con routine personalizzate

All'interno del main process, è possibile creare un gestore di errori come si può vedere nell'esempio riportato qui sotto. Ogni routine[32] può includere una gestione degli errori dichiarando ERROR alla fine delle operazioni della routine definendo così opzioni personalizzate per gestire i diversi codici di errore.

```
FUNC num safediv( num x, num y)
        RETURN x / y;
   ERROR
        IF ERRNO = ERR_DIVZERO THEN
        TPWrite "The number cannot be equal to 0";
        RETURN x;
        ENDIF
ENDFUNC
```

Figura 36 - Esempio della gestione degli errori all'interno di una funzione

Nel caso in cui si manifesti un errore il robot, di default, si ferma generando, sul tastierino, un codice di errore. Se l'errore non viene gestito è necessario l'intervento di un operatore che farà ripartire l'esecuzione del ciclo del robot. Se all'interno della procedura o della funzione viene definito l'handler è possibile gestire alcuni codici di errore. Quando si verifica un errore, è possibile risalire al valore numerico dell'ultimo errore avvenuto tramite il parametro ERRNO, dato persistente che viene salvato nella memoria della CPU, confrontandolo con gli errori definiti nel sistema, come illustrato nell'esempio sopra riportato.

[32] Per routine si intendono sia processi (PROC set di istruzioni che non restituisce valori), che funzioni (FUNC set di istruzioni che restituisce un valore) che trap function (set di istruzioni che interviene in caso di interrupt)

In particolare, nello spezzone di programma riportato, si può vedere come è stato gestito l'errore di collisione, creando un semplice algoritmo di gestione per l'applicazione illustrata nel capitolo seguente.

```
!GESTIONE ERRORE DI COLLISIONE
ERROR
StorePath;
StopMove;
IF ROBOT_Collisione_DX=high AND ROBOT_Collisione_SX=high THEN
    !Diminuisco la velocità e recupero la posizione corrente
    StartMove;
    AccSet 50,50;
    pTempR:=CRobT(\Tool:=tool0 \WObj:=wobj0);
    !Muovo indietro e ritento il passaggio
    TEST IDmovimento
      CASE 0,10,50,90:
      MoveL RelTool (Offs(pTempR,0,0,-10),0,0,0),v500,fine, tool0\WObj:=wobj0;
      CASE 20:   !SopraPrel
      MoveL RelTool (Offs(pTempR,-30,0,20),0,0,0),v500,fine, tool0\WObj:=wobj0;
      CASE 30,31:   !Prel
      MoveL RelTool (Offs(pTempR,0,0,30),0,0,0),v500,fine, tool0\WObj:=wobj0;
      CASE 40:   !Prel
      MoveL RelTool (Offs(pTempR,0,0,10),0,0,0),v500,fine, tool0\WObj:=wobj0;
      CASE 60,80:   !SopraDep
      MoveL RelTool (Offs(pTempR,0,0,25),0,0,0),v500,fine, tool0\WObj:=wobj0;
      CASE 70,71:   !Dep
      MoveL RelTool (Offs(pTempR,0,0,30),0,0,0),v500,fine, tool0\WObj:=wobj0;
      DEFAULT:
      Stop;
    ENDTEST
```

Figura 37 - Parte dell'error handler sviluppato per il caso applicativo

Con l'istruzione CrobT[33] il controller del robot va a leggere le posizioni dei giunti recuperando la posizione di stop del robot. Dopo aver controllato, tramite un IF se l'errore corrisponde a quello di collisione, si passa a testare quale movimento si stava compiendo.

[33] In questo caso, tra i parametri della funzione si possono impostare il sistema di riferimento e il tool da considerare, non essendo presenti utensili il tool è definito come tool0 per cui calcolerà il target in funzione del sistema di riferimento del cobot andando a considerare il punto centrale della flangia del braccio.

In particolare, come verrà spiegato nel capitolo successivo, ogni movimento, essendo sincronizzato tra i due bracci del robot, ha un identificativo. Viene sfruttato questo sistema per indentificare il movimento corrente.
Dopo aver recuperato le informazioni legate a movimento e posizioni si tenta di far raggiungere nuovamente la posizione finale dello spostamento andando a vedere se è presente ancora l'ostacolo che ha provocato la collisione.
La gestione in questo caso viene gestita retrocedendo di poco per ridurre le forze in gioco e tentare di liberarsi dall'ostacolo, tramite la funzione *Reltool.*
Per poter riprendere il movimento programmato è importante, prima di eseguire qualsiasi altra operazione, all'interno del gestore errori, chiamare la funzione *StorePath.* Questa funzione permette di salvare le informazioni riguardanti il puntatore del programma sapendo così dove si è stato interrotto il sistema.
Una volta eseguite le operazioni per gestire l'errore, si vuole riprendere la sequenza standard di istruzioni: tramite la chiamata *RestoPath* si recuperano le informazioni salvate precedentemente e con la chiamata *Retry* si fa ripartire il robot. Questa chiamata, che è possibile chiamare solo nel gestore di errori, oltre a riprendere il processo del sistema va ad eliminare il codice di errore che ha chiamato l'handler degli errori.
La chiamata Retry tiene conto anche delle volte che si tenta di riprendere l'esecuzione del programma in seguito ad un messaggio di errore: se il numero di tentativi supera il quattro, l'esecuzione del programma viene interrotta, obbligando l'intervento dell'operatore.
Tramite il gestore di errori è quindi possibile gestire qualsiasi tipo di errore creando una sequenza di istruzioni per risolvere l'errore.
Per quanto riguarda i robot collaborativi, questa possibilità risulta utile, in quanto nel caso in cui acCAD a una collisione con l'operatore, si può gestire con velocità ridotte e forze ridotte la ripresa dei percorsi.

5.2.2 Gestione separata della collisione

La collisione, per definizione avviene quando un braccio si scontra con un altro braccio o con un oggetto. Il sistema di accorge della collisione perché rileva una forza maggiore applicata ad un giunto e quindi una coppia maggiore[34] di un certo valore per un determinato motore.
Tramite il tastierino, collegato ad un sistema MultiMove, che in cascata è collegato ai controller dei singoli bracci robotici, il sistema sa quale giunto di quale braccio è andato in collisione, mostrandolo nell'errore.
Quello che si è cercato di fare, anche per la gestione della collisione dello Yumi, è capire quale braccio è andato in errore e muovere solo quello per poter vedere se la collisione è temporanea, causato da uno scontro con oggetti o operatori di passaggio.
Per poter fare ciò è necessario creare un segnale d'uscita di sistema. Per prima cosa, nel file EIO[35], si va a definire un segnale per task. Come si può vedere dall'immagine, all'interno della zona configurazione, tra le

[34] Nell'immagine in Figura 28 - Limiti di forza in funzione del tempoè possibile vedere come il sensore valuta la forza in funzione del tempo trascorso
[35] EIO: file di configurazione del sistema in cui vengono definiti i vari segnali, andando ad indicare la tipologia, tramite che scheda vengono scambiati, l'identificativo detto mappatura di sistema. In questo modo i segnali risultano univoci.

definizioni di I/O nell'interno della sezione System Output e si crea un segnale legato al Motion Supervision Triggered, assegnando il trigg del sistema di supervisione per ogni task al segnale d'uscita definito nell'EIO.

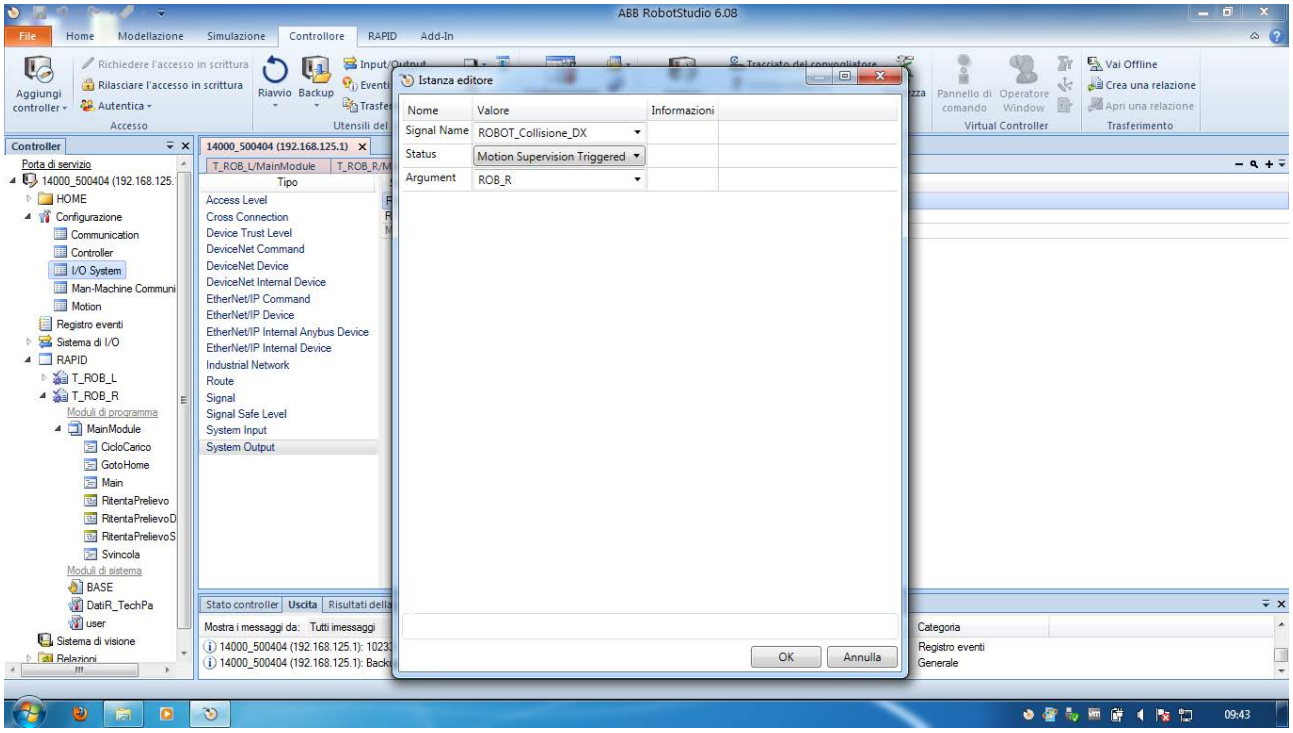


Figura 38 - Definizione del segnale di sistema di collisione

In questo modo è stata stabilita una connessione tra il sistema robot e "l'esterno" poiché è possibile andare a controllare se le unità meccaniche sono andate in collisione oppure no. Lo Status definito nella figura è un segnale che viene generato quando un sensore rileva una forza superiore per un determinato tempo.

Ogni volta che nel sistema viene generato un errore viene attivato l'handler, nel caso sia stato definito, altrimenti, di default, viene fermata l'esecuzione.

Una volta che viene segnalato l'errore di collisione (ERR_COLL_STOP) e quindi attivato l'handler di errore, il sistema MultiMove recupera l'informazione dal segnale di output, generato dal controller del singolo braccio, e in base ad esso esegue i movimenti descritti. Nella parte mostrata in Figura 37 - `Parte dell'error handler sviluppato per il caso applicativo` si fa riferimento al caso in ambedue le unità meccaniche siano in collisione.

Grazie all'*IDMovimento*, variabile creata[36] e che viene aggiornata al termine di ogni movimentazione, è possibile differenziare il comportamento del braccio.

Si vuole porre l'attenzione ora sul sincronismo delle movimentazioni. Il cobot esegue le movimentazioni in modo sincronizzato per cui è necessario che tutte le movimentazioni vengano eseguite nel medesimo tempo. Quando viene generato un errore, il sistema MultiMove blocca l'esecuzione del task ed entra nel gestore di

[36] È stata creata la variabile per poter capire in che momento dell'esecuzione il cobot si bloccava. L'id presente nell'istruzione Move è interna al sistema e viene generata solo per il tempo dell'esecuzione della movimentazione, per cui non è possibile utilizzarla come flag di posizione.

errori. Grazie alla chiamata di sistema *StorePath()* il controller esegue una istantanea delle informazioni. In questo modo riesce a memorizzare l'id di sincronizzazione presente nell'istruzione Move permettendo, quindi, di poter riprendere la movimentazione dei bracci meccanici senza perdere il sincronismo. Il vantaggio principale è quello di poter agire solo sull'unità meccanica che è andata in errore.

5.3 Evitare collisioni tramite algoritmi

In questo paragrafo si è svolta una ricerca tra articoli scientifici per analizzare come affrontare il problema.
È bene distinguere due diverse applicazioni:

1) generazione di una traiettoria per evitare la collisione del manipolatore con oggetti fermi;
2) generazione di una traiettoria e controllo real-time per evitare la collisione con oggetti mobili.

Occorre distinguere queste due applicazioni in quanto, in base alla metodologia utilizzata per perseguire questo scopo variano i tempi di esecuzione e conseguentemente variano anche le possibili applicazioni.
Il parametro più comunemente adottato in questo campo risulta essere la distanza di uno o più punti del manipolatore da uno o più oggetti presenti nello spazio operativo ma ci sono ulteriori approcci che valutano anche altri parametri fondamentali come: inerzia, velocità relativa, parte del corpo urtata ecc. Un approccio molto analizzato in letteratura è quello che fa uso degli algoritmi genetici (GA).
Tale approccio prevede prima di tutto la determinazione di uno spazio operativo limitato all'operazione richiesta che generi un parallelepipedo avente diagonale il punto iniziale ed il punto finale della traiettoria che si vuole determinare. Determinato tale volume si traccia una griglia di possibili configurazioni in cui si può disporre il manipolatore. La griglia deve essere sufficientemente fitta da eliminare la possibilità che un oggetto di possa disporre tra una configurazione e l'altra. L'approccio utilizzato consiste nel minimizzare la somma della distanza tra due configurazioni adiacenti e la distanza tra la configurazione intermedia da raggiungere allo step successivo rispetto alla configurazione finale del manipolatore.
L'algoritmo genetico spesso utilizza l'Hermen cubic interpolation method che permette il passaggio da una configurazione intermedia alla successiva con un polinomio del terzo ordine.
Il tempo di esecuzione dell'algoritmo è di alcune decine di secondi pertanto presumibilmente non è adatto per operazioni real-time ma studia una traiettoria che ottimizza il tempo di esecuzione del task evitando gli ostacoli presenti nello spazio operativo. Tale algoritmo può essere adottato per qualsiasi configurazione cinematica del robot, non importa che esso sia ridondante (31).
Altra soluzione è quella del QPSO (Quantum Behaved Particicle Swarm Optimization) un algoritmo più rapido del GA ed ha potenziale applicazione online. Un metodo per procedere con tale tecnica consiste nel considerare una funzione di costo costituita da 4 indici, ognuno con il proprio peso, che deve essere minimizzata.
Il metodo SQP (Sequential Quadratic Programming) è un metodo iterativo per l'ottimizzazione di vincoli non lineari. Tale metodo può essere adottato al fine di ottimizzare diversi obiettivi come ad esempio: massimizzare la distanza dai limiti di giunto, mantenere in un range prefissato la velocità dei giunti e massimizzare la distanza di un punto del manipolatore da un ostacolo esterno. Tale metodo, come i precedenti, evita l'utilizzo del

metodo della pseudoinversa ma presenta alcuni svantaggi non trascurabili: difficoltà di bilanciare i costi e evitare la collisione, in alcuni casi la collisione non può essere evitata, in certe situazioni il robot deve avvicinarsi troppo all'ostacolo.

Il metodo Deapth Space Approach è quello adottato all'Università Sapienza di Roma, al contrario dei precedenti i quali presuppongono nota la posizione (fissa o mobile) degli ostacoli da evitare e si concentrano sulla generazione di una opportuna traiettoria integra le due cose.

Infatti, tale metodo utilizza un Kinect che da una informazione 2-3 dimensionale. Lo studio è stato avanzato su un KUKA a 7 gradi di libertà controllato in posizione e pertanto 4 volte ridondante utilizzando un algoritmo di potenziale che genera un campo attrattivo verso il target e un campo repulsivo verso gli ostacoli. L'algoritmo di generazione della traiettoria utilizzato è tra i più rapidi di quelli elencati perché genera un controllo ogni pochi millisecondi permettendo una reale implementazione online.

Altro metodo che si distingue dai precedenti perché non valuta esclusivamente la distanza del manipolatore dall'ostacolo ma ne considera anche la velocità relativa è quella del CDF (Cumulative Danger Field) tale approccio è stato ottimizzato nel corso degli anni da studi condotti da ricercatori del Politecnico di Milano.

CONCLUSIONI

I robot sono stati realizzati per poter aiutare l'uomo in lavori degradanti e pericolosi: lo sviluppo tecnologico e industriale ci ha portati a costruire robot sempre più prestanti e capaci di aiutare l'uomo nelle diverse mansioni. Durante gli anni e con l'aumento della presenza dei robot all'interno delle aziende è sorta una paura tra i lavoratori di essere sostituiti in tutti i lavori dai robot, una paura che nasce da quel disagio che l'uomo prova nei confronti dell'ignoto. Ci si deve ricordare che i robot sono macchine complesse ma a servizio dell'uomo e che non possiedono una propria intelligenza. L'uomo avrà sempre posto in questo mondo che spinge verso la più totale automazione dei lavori. Le case costruttrici stanno spingendo verso la robotica collaborativa anche per aiutare l'uomo a disinnescare questo timore: infatti spinge ancora più in là l'asticella di come un robot possa aiutare l'uomo nel compiere lavori ripetitivi, degradanti o pericolosi.

All'interno di questa tesi non si è affrontata la tematica relativa all'etica dell'uso dei robot e l'impatto psicologico che producono si lascia questa analisi ad altri testi e ricerche. Quello che è possibile affermare è che, dalle ricerche eseguite nel campo, per le generazioni più giovani al di sotto dei 50 anni è molto più semplice accettare la presenza di robot nel ciclo produttivo e per le generazioni con età inferiore ai 40 risulta normale l'inserimento di robot anche in altri campi, come la medicina ma in generale nel primo e nel terzo settore, in particolare di quelli collaborativi.

L'obiettivo della tesi, come espresso precedentemente, è quello di analizzare la possibilità di gestire gli errori di collisione tra cobot e operatori. Come si è visto nel quinto capitolo, è possibile andare a gestire gli errori creando delle funzioni ad hoc per ogni tipologia di errore.

Analizzando la posizione reale del cobot e memorizzando l'id di movimentazione è possibile recuperare informazioni rispetto a quale punto del ciclo ci si ferma. La possibilità di gestire errori, grazie all'*error handler* definibile all'interno di ogni procedura, è una funzionalità che potenzia la capacità del robot poiché, in automatico, può riprendere il ciclo di lavoro normale dopo aver svolto determinati movimenti e controlli. All'interno dell'handler è possibile gestire, con le informazioni immagazzinate, la movimentazione per svincolarsi da un eventuale ostacolo. Nel caso in cui si superi un valore massimo di tentativi per svincolarsi, valore che può essere impostato e che è definito pari a quattro da ABB, il robot si ferma e generale un errore per avvisare che è impossibile ripristinare il ciclo. Il numero di tentativi dipende da quante volte il cobot cerca di ritornare nel ciclo uscendo dal gestore degli errori.

La gestione degli errori, in particolare quelli legati alla collisione, è stata ripresa anche nell'ultimo capitolo della tesi. La supervisione dei movimenti è una funzione presente nel controller che monitora le movimentazioni, i limiti dei giunti, i limiti di coppia e, nel caso in cui sia presente la scheda safe, le zone di sicurezza e i componenti del braccio robotico. Per lo scopo della tesi ci si è soffermarti sull gestione degli errori di collisione che vengono visiti dal controller come forze superiori ad un determinato limite. Per poter gestirlo tramite l'error handler si è creata una cross connection su un segnale digitale in modo da controllare lo stato del sistema.

Prevedere tutti gli errori risulterebbe un lavoro arduo, ma la possibilità di gestirli da un enorme vantaggio per automatizzare alcune problematiche, come eventuali collisioni, senza che l'operatore intervenga.
Un altro obiettivo che ci si era posti all'inizio della ricerca è quello di poter richiamare determinate funzioni a seconda delle condizioni in cui si trova il robot in funzione della situazione sulla linea. Questo è stato possibile grazie all'implementazioni di *trap routine.* Attraverso la generazione di un interrupt di sistema, collegato ad un segnale esterno proveniente dal PLC, è possibile gestire una determinata casistica con l'ausilio delle trap routine. All'interno del caso applicativo sviluppato presso l'azienda ve ne sono tre che corrispondono al mancato prelievo di una delle confezioni. In queste ruotine si esegue nuovamente il prelievo della confezione dalla linea. Se si superano i quattro tentativi il robot genera un segnale di svuotamento della linea e uno stop delle movimentazioni.
Durante lo sviluppo della stazione robotica e della tesi ci si è imbattuti in problemi di singolarità, analizzati nel quarto capitolo. All'interno del capitolo si è affrontato il problema cercando vari metodi per risolverle come algoritmi che potenziano le capacità della CPU, anche se si appesantiscono la quantità di calcoli. Il problema delle singolarità nasce quando il braccio meccanico ha almeno 6 assi, in quanto, per risolvere la cinematica del robot e trovare la posa del robot, è necessario risolvere un sistema che dipende dai gradi di libertà nello spazio e dal numero di assi del robot. Si può dire che la singolarità equivale ad una incertezza del posizionamento e quindi del controllo della posa del robot.
L'alternativa più semplice è definire nell'intorno della zona di singolarità, alcuni punti per definire una traiettoria definita che non includa il punto, nel caso in cui il punto non sia obbligato. L'alternativa a questo metodo è l'ausilio di algoritmi offline come *Damped Least Squares* oppure come l'*on-line task modification method*. Gli algoritmi sono basati sulla teoria del controllo ottimo e grazie all'utilizzo di indici si cerca di minimizzarli o massimizzarli in base al loro significato.
La tesi copre una piccola parte del mondo della robotica collaborativa e di come i robot interagiscono con l'operatore. Non è stato possibile approfondire aspetti come la cinematica inversa e come il controller risolva tale problema, anche se si ipotizzi che venga utilizzato un metodo iterativo per arrivare alla soluzione. Altro aspetto interessante da valutare è l'applicazione dello YuMi con sistemi di visione e riconoscimento di oggetti.
Le strade che la robotica collaborativa ha aperto sono molteplici e le applicazioni sono innumerevoli, infatti, basta seguire sui social network le varie case costruttrici di cobot, si possono vedere svariate applicazioni nate dalla fantasia dell'uomo.

Appendice A

In questa appendice è riportato lo script sviluppato nel modulo principale richiamato dal controller. Per aiutare la lettura del programma viene proposta una legenda:

- PLC_### segnale digitale input, proveniente dal PLC, program logic computer
- ROBOT_### segnale digitale di output, inviato al PLC
- R_### indica un robtarget per il braccio destro
- Altre scritte variabili create per la logica del robot

Per quanto riguarda i colori:

- Nero variabili e segnali
- Blu definizioni proprie del linguaggio Rapid
- Verde commenti

Di seguito è stato riportato il programma sviluppato per gestire il braccio destro dello YuMi. Il braccio sinistro ha il medesimo programma, i punti definiti come robtarget sono simmetrizzati rispetto al piano xz del sistema di riferimento del robot.

```
MODULE MainModule
PROC Main()
   Svincola;
   GotoHome;
   IDmovimento:=0;

!GESTONE MANCATO PRELIEVO -> IL SEGNALE DAL PLC VIENE COLLEGATO AD UN INTERRUPT VIRTUALE CHE AVVIA LA TRAP ROUTINE
   IDelete sig1int;
   IDelete sig2int;
   IDelete sig3int;
   CONNECT sig1int WITH RitentaPrelievo;
   ISignalDI PLC_NonPrelevato,1,sig1int;
   CONNECT sig2int WITH RitentaPrelievoDX;
        ISignalDI PLC_NonPrelevatoDX, 1, sig2int;
   CONNECT sig3int WITH RitentaPrelievoSX;
        ISignalDI PLC_NonPrelevatoSX, 1, sig3int;

!INIZIA IL CICLO DI LAVORO CON L'ANALISI DELLE VARIE CONDIZIONI
```

```
WHILE PLC_WORKING=high DO
 !Definisco parametri ciclo: tipo scatolone, punto di deposito ,dimensioni delle confezioni
IF PLC_TipoScatolone=high THEN      !Se TipoScatolone=high -> Scatolone 1
   hScatola:=Scatola1;
   R_BaseDep:=R_BaseDep1;
   R_SopraDep:=R_SopraDep1;
   AltezzaScatola:=HScatola1;
   LarghezzaScatola:=LScatola1;
ELSEIF PLC_TipoScatolone=low THEN   !Se TipoScatolone=high -> Scatolone 2
   hScatola:=Scatola2;
   R_BaseDep:=R_BaseDep2;
   R_SopraDep:=R_SopraDep2;
   AltezzaScatola:=HScatola2;
   LarghezzaScatola:=LScatola2;
ENDIF

!GESTIONE POSIZIONAMENTO SCATOLONE
   P0:
        IF PLC_RichiestaScatolone=high THEN
                SetDO ROBOT_CaricoCompletato,0;                    !Resetto
                !Attendo comando da PLC
                WaitDI PLC_GoToHome,1\MaxTime:=1\TimeFlag:=bmaxtime{0};
       IF bmaxtime{0}=TRUE THEN
          bmaxtime{0}:=FALSE;
                    GOTO P0;
             ENDIF
                MoveL R_Home,Speed,z10,tool0\WObj:=wobj0;
     SetDO ROBOT_InHome,1;
                !Attendo che lo scatolone sia in posizione
                WaitDI PLC_ScatoloneInPosizione,1\MaxTime:=2\TimeFlag:=bmaxtime{1};
       IF bmaxtime{1}=TRUE THEN
          bmaxtime{1}:=FALSE;
                    GOTO P0;
             ENDIF
     SetDO ROBOT_InHome,0;
        ENDIF
```

```
FOR indexZ FROM 0 TO hScatola DO
FOR indexX FROM 0 TO 2 DO


    P1:
    !CONTROLLO FUNZIONAMENTO DELLA LINEA
    IF PLC_LineaInFunzione=high AND PLC_ScatoloneInPosizione=high AND
PLC_ConfezioniInPosizione=high THEN
                CicloCarico indexZ,indexX;

    ELSEIF PLC_LineaInFunzione=high AND PLC_ScatoloneInPosizione=high AND
PLC_ConfezioniInPosizione=low THEN
        GotoHome;
        WaitDI PLC_ConfezioniInPosizione,high\MaxTime:=1\TimeFlag:=bmaxtime{2};
        IF bmaxtime{2}=TRUE THEN
            bmaxtime{2}:=FALSE;
                GOTO P1;
            ENDIF
        GOTO P1;
        ENDIF

!INVIO SEGNALE DI CARICO COMPLETATO
IF PLC_TipoScatolone=high AND indexX=2 AND indexZ=2 THEN      !Scatola tipo 1
        SetDO Robot_CaricoCompletato,1;
ELSEIF PLC_TipoScatolone=low  AND indexX=2 AND indexZ=3 THEN    !Scatola tipo 2
        SetDO Robot_CaricoCompletato,1;
ENDIF
ENDFOR
ENDFOR

  !GESTIONE SCATOLONE CARICO
  P2:
        IF Robot_CaricoCompletato=high THEN
            WaitDI PLC_GoToHome, high\MaxTime:=1\TimeFlag:=bmaxtime{3};
            IF bmaxtime{3}=TRUE THEN
                bmaxtime{3}:=FALSE;
                GOTO P2;
```

```
            ENDIF
        MoveL R_Home,Speed,z10,tool0\WObj:=wobj0;
        SetDO ROBOT_InHome,1;
        WaitDI PLC_ScatoloneInFineLinea,high\MaxTime:=1\TimeFlag:=bmaxtime{4};
            IF bmaxtime{4}=TRUE THEN
                bmaxtime{4}:=FALSE;
                GOTO P2;
            ENDIF
            SetDO Robot_CaricoCompletato, 0;
        ENDIF

ENDWHILE


!GESTIONE ERRORE DI COLLISIONE
ERROR
StorePath;
StopMove;
  IF ROBOT_Collisione_DX=high AND ROBOT_Collisione_SX=high THEN
    !Diminuisco la velocità e recupero la posizione corrente
    StartMove;
    AccSet 50,50;
    pTempR:=CRobT(\Tool:=tool0 \WObj:=wobj0);
    !Muovo indietro e ritento il passaggio
    TEST IDmovimento
      CASE 0,10,50,90:
      MoveL RelTool (Offs(pTempR,0,0,-10),0,0,0),v500,fine, tool0\WObj:=wobj0;
      CASE 20:   !SopraPrel
      MoveL RelTool (Offs(pTempR,-30,0,20),0,0,0),v500,fine, tool0\WObj:=wobj0;
      CASE 30,31:   !Prel
      MoveL RelTool (Offs(pTempR,0,0,30),0,0,0),v500,fine, tool0\WObj:=wobj0;
      CASE 40:   !Prel
      MoveL RelTool (Offs(pTempR,0,0,10),0,0,0),v500,fine, tool0\WObj:=wobj0;
      CASE 60,80:   !SopraDep
      MoveL RelTool (Offs(pTempR,0,0,25),0,0,0),v500,fine, tool0\WObj:=wobj0;
      CASE 70,71:   !Dep
      MoveL RelTool (Offs(pTempR,0,0,30),0,0,0),v500,fine, tool0\WObj:=wobj0;
```

```
            DEFAULT:
            Stop;
        ENDTEST

    ELSEIF ROBOT_Collisione_DX=high THEN
        Robot_Collisione_DX_OK:=FALSE;
        !Diminuisco la velocità e recupero la posizione corrente
        StartMove;
        AccSet 50,50;
        pTempR:=CRobT(\Tool:=tool0 \WObj:=wobj0);
        !Muovo indietro e ritento il passaggio
        TEST IDmovimento
            CASE 0,10,50,90:
            MoveL RelTool (Offs(pTempR,0,0,-10),0,0,0),v500,fine, tool0\WObj:=wobj0;
            CASE 20:   !SopraPrel
            MoveL RelTool (Offs(pTempR,-30,0,20),0,0,0),v500,fine, tool0\WObj:=wobj0;
            CASE 30,31:   !Prel
            MoveL RelTool (Offs(pTempR,0,0,30),0,0,0),v500,fine, tool0\WObj:=wobj0;
            CASE 40:   !Prel
            MoveL RelTool (Offs(pTempR,0,0,10),0,0,0),v500,fine, tool0\WObj:=wobj0;
            CASE 60,80:   !SopraDep
            MoveL RelTool (Offs(pTempR,0,0,25),0,0,0),v500,fine, tool0\WObj:=wobj0;
            CASE 70,71:   !Dep
            MoveL RelTool (Offs(pTempR,0,0,30),0,0,0),v500,fine, tool0\WObj:=wobj0;
            DEFAULT:
            Stop;
        ENDTEST
        Robot_Collisione_DX_OK:=TRUE;
    ELSEIF ROBOT_Collisione_SX=high THEN
        WaitUntil ROBOT_Collisione_DX_ok;
    ENDIF
    RestoPath;
    AccSet 100,100;
    StartMove;
    RETRY; !Necessario per resettare l'errore di collisione
ENDPROC
```

```
PROC GotoHome()
    VAR speeddata mySpeed;
    mySpeed:=v200;
    mySpeed.v_ori:=100;
    MoveJ R_Home,mySpeed,fine,tool0;
    Stop;
ENDPROC

PROC Svincola()
  !Resetto segnali
  Reset ROBOT_CaricoCompletato;
  Reset ROBOT_InHome;
  Reset ROBOT_InPrelievo;
  Reset ROBOT_RitentaPrelievo;
  Reset ROBOT_ControlloPrelievo;
  Reset ROBOT_Collisione_DX;
  Reset ROBOT_Collisione_SX;

  !Resetto Elettrovalvole
  TriggIO tgEV_VuotoON,AnticipoEV_Vuoto\DOp:=custom_DO_1,1;          !EV vuoto
  TriggIO tgEV_VuotoOFF,AnticipoResetEV_Vuoto\DOp:=custom_DO_1,0;  !EV vuoto
  ENDPROC


  PROC CicloCarico(num Zindex,num Xindex)
    !Attivazione sincronizzazione tra i due bracci del robot
    SyncMoveOn Task_1, task_list;

    !Mi posiziono in Home
    MoveJ R_Home\ID:=10,Speed,z50,tool0\WObj:=wobj0;
    IDmovimento:=10;

    !Prelievo
    MoveL R_SopraPrel\ID:=20,Speed,z10,tool0\WObj:=wobj0;
    IDmovimento:=20;
    !Attivo l'elettrovalvola
    TriggL Reltool(R_Prel,0,0,10)\ID:=30,speedP,tgEV_VuotoON,z10, tool0\WObj:=wobj0;
```

```
IDmovimento:=30;
MoveL R_Prel\ID:=31, SpeedP, fine, tool0\WObj:=wobj0;
IDmovimento:=31;
waittime 0.1;
MoveL R_SopraPrel\ID:=40,Speed,z10,tool0\WObj:=wobj0;
IDmovimento:=40;

!Passaggio per Home
AccSet 50,50;
      MoveL R_Home\ID:=50,Speed,z10,tool0\WObj:=wobj0;
IDmovimento:=50;

!SetDO Robot_ControlloPrelievo,1;
!WaitDI PLC_ControlloPrelievo,high\MaxTime:=2\TimeFlag:=bmaxtime{6};
!IF bmaxtime{6}=TRUE THEN
!   bmaxtime{6}:=FALSE;
!   SetDO Robot_ControlloPrelievo,0;
      !ENDIF
!SetDO Robot_ControlloPrelievo,0;

MoveL R_SopraDep\ID:=60,Speed,z10,tool0\WObj:=wobj0;
IDmovimento:=60;
AccSet 100,100;

!Calcolo Offset per Deposito
OffsetX:=Xindex*LarghezzaScatola;
OffsetZ:=Zindex*AltezzaScatola;

!Deposito
TriggJ      Reltool(OFFS(R_BaseDep,OffsetX,0,OffsetZ),0,0,0)\ID:=71,Speed,tgEV_VuotoOFF,fine,
tool0\WObj:=wobj0;
IDmovimento:=71;
WaitTime 0.1;
MoveJ R_SopraDep\ID:=80,SpeedD,z10,tool0\WObj:=wobj0;
IDmovimento:=80;

!Ritorno in home
```

```
    MoveJ R_Home\ID:=90,Speed,z10,tool0\WObj:=wobj0;
    IDmovimento:=90;

    SyncMoveOff Task_2;
ENDPROC

TRAP RitentaPrelievo
    StorePath;
    !Prelievo
    MoveL R_SopraPrel,Speed,z10,tool0\WObj:=wobj0;
    !Attivo l'elettrovalvola
    TriggL Reltool(R_Prel,0,0,10),speedP,tgEV_VuotoON,fine, tool0\WObj:=wobj0;
    MoveL R_Prel, SpeedP, z10, tool0\WObj:=wobj0;
    waittime 0.5;
    MoveL R_SopraPrel,Speed,z10,tool0\WObj:=wobj0;
    !Passaggio per Home
    AccSet 50,50;
        MoveL R_Home,Speed,z10,tool0\WObj:=wobj0;
    RestoPath;
    StartMove;
ENDTRAP

TRAP RitentaPrelievoDX
    PrelievoDX_OK:=FALSE;
    StorePath;
    MoveJ R_Home,Speed,z10,tool0\WObj:=wobj0;
    MoveL R_SopraPrel,Speed,z10,tool0\WObj:=wobj0;
    !Attivo l'elettrovalvola
    TriggL Reltool(R_Prel,0,0,30),speedP,tgEV_VuotoON,fine, tool0\WObj:=wobj0;
    MoveL R_Prel, SpeedP, z10, tool0\WObj:=wobj0;
    waittime 0.5;
    MoveL R_SopraPrel,Speed,z10,tool0\WObj:=wobj0;
    AccSet 50,50;
        MoveL R_Home,Speed,z10,tool0\WObj:=wobj0;
    !controllo prelievo
    PrelievoDX_OK:=TRUE;
    RestoPath;
```

```
    StartMove;
  ENDTRAP

  TRAP RitentaPrelievoSX
    StorePath;
    WaitUntil PrelievoSX_OK=TRUE\MaxTime:=20\TimeFlag:=bmaxtime{5};
    IF bmaxtime{5}=TRUE THEN
      TPWrite "Errore Tempo Massimo per recuperare confezione superato";
      bmaxtime{5}:=FALSE;
        ENDIF
    RestoPath;
    StartMove;
  ENDTRAP
ENDMODULE
```

INDICE DELLE FIGURE

Figura 1 - Robot collaborativo che condivide lo spazio di lavoro con operatori umani 13
Figura 2 - Modelli CR-7iA e CR-7iA/L 17
Figura 3 - A sinistra modelli UR3, UR5, UR10 di Universal Robot, a destra modello HC10 della Yaskawa.... 18
Figura 4 - Kuka LBRiiwa 19
Figura 5 - Modelli Sawyer eBaxter della Rethink 20
Figura 6 - Aura, il robot collaborativo di Comau 22
Figura 7 - ABB IRB14050 Single Arm Yumi e area di lavoro 23
Figura 8 - ABB YuMi IRB 14000 YuMi 28
Figura 9 -Esploso del braccio IRB 14000 e dimensioni dei link 29
Figura 10 - Alcuni componenti dei bracci, in ordine: motore, imbottitura e blocco meccanico 30
Figura 11 - La figura a sinistra mostra i contrassegni utili per la calibrazione del cobot. Il cobot si trova in posizione di calibratura, come è visibile, in modo migliore, nella figura a destra 32
Figura 12 - A sinistra funzionamento di massima del sensore ad effetto Hall. 33
Figura 13 - Vista dettagliata della flangia e dei pin 35
Figura 14 - Dettaglio della pinza ABB e varie combinazioni di grippers realizzate da ABB per l'IRB14000 35
Figura 15 - Dimensioni della pinza e assi di riferimento del TC 36
Figura 16 - Diagramma di payload senza pinza e con pinza 36
Figura 17 - Controller integrato nello YuMi. A destra visione dei componenti principali del controller 37
Figura 18 - Controller IRB 14000 a sinistra visto dall'alto, a destra visto dal retro 38
Figura 19 - Flex pendant con sistema RobotWare 39
Figura 20 - Tasks selezionabili da FlexPendant, ogni task è riferito ad una unità meccanica 40
Figura 21 - Schermata della flex pendant, definita "Movimento Manuale" in cui è possibile definite come muoversi. 41
Figura 22 - Schermata del software RobotStudio con vista delle varie schede. 42
Figura 23 - Sistema MultiMove con controller collegati 45
Figura 24 - Rappresentazione minima del braccio dell'IRB14000, gli assi sono disallineati. I valori sono espressi in millimetri. 48
Figura 25 - Layout della linea di confezionamento 51
Figura 26 - Visione del simulatore, rappresentazione semplificata della stazione con analisi di movimentazione del cobot. 53
Figura 27 - Segnali scambiati tra controller robot e PLC 54
Figura 28 - Limiti di forza in funzione del tempo 55

Figura 29 – Schermate del software RobotStudio dove si può vedere i due segnali definiti per ogni task e come vengono creati 56
Figura 30 - Inizializzazione di un segnale di interrupt e collegamento con trap routine 66
Figura 31 - Reset segnale di inteerupt alla fine della trap routine associata per poterlo riutilizzare 66
Figura 32 - Controllo del tempo di attesa di un segnale proveniente da PLC 67
Figura 33 - Relazione spazio operativo e spazio giunti 72
Figura 34 - Singolarità di Polso (1), di gomito (2) e di spalla (3) 74
Figura 35 - Esempio di manipolatore planare ridondante che acquisisce la stessa posizione dell'end effector 75
Figura 36 - Esempio della gestione degli errori all'interno di una funzione 86
Figura 37 - Parte dell'error handler sviluppato per il caso applicativo 87
Figura 38 - Definizione del segnale di sistema di collisione 89

INDICE DELLE TABELLE

Tabella 1 -Confronto caratteristiche dei cobot attualmente in commercio 24
Tabella 2 - Tabella dei limiti biomeccanici 31
Tabella 3 - Tabella indicante i gradi per ogni asse nella corretta posizione di calibrazione per i bracci dello YuMi 32
Tabella 4 - Parametri di Denavit - Hartenberg per il braccio destro dello YuMi 48

Bibliografia

ABB, Application manual – *Technical reference manual MultiMove*, 2017

ABB, Application manual - *Technical reference manual Rapid Kernel*, 3HAC16585-1, 2013

ABB, Application manual – *Technical reference manual Panoramica Rapid***,** 3HAC16580-7, 2009

ABB, Application manual - *Technical reference manual, Guida introduttiva, IRC5 e RobotStudio*, Id documento 3HAC027097-007, 2008

ABB, Application manual - *Technical reference manual Robotware*, Id documento 3HAC050961-007, 2016

ABB, Application manual - *Technical reference manual Controller IRC5,* Id documento 3HAC047136-007, 2016

ABB, Application manual - *Technical reference* manual Controller Software IRC5, Id documento 3HAC048264-007, 2014

ABB, Application manual - *Technical reference manual, IRB14000*, Id documento 3HAC052983-007, 2016

ABB, Application manual - *Technical reference manual, Dati tecnici del prodotto IRB 14000*, Id documento 3HAC052982, 2018

ABB, Application manual - *Technical reference manual, Sicurezza funzionale e SafeMove2*, Id documento 3HAC052610, 2016

ABB, Application manual - *Technical reference manual SofMove*, Id documento 3HAC050977-001, 2016

Elin A. Topp, Mathias Haage, Jacek Malec, *Knowledge for Synchronized Dual-Arm Robot Programming*, Artificial Intelligence for Human-Robot Interaction, AAAI Technical Report, 2017.

Tang W., *Redundant manipulator infinity-norm joint torque optimization with actuator constraints using a recurrent neural network*, IEEE, 2001.

Tan J., Xi N., Wang Y. Houghton, *A singularity-free motion*, Automatica, 2004

Qiu C., Cao Q. Miao S. Shanghai, *An on-line task modification method for singularity avoidance of robot manipulators,* Robotica, 2008.

Duleba I., Sasiadek J. Z. Wroclaw , *Modified Jacobian Method of Transversal Passing through Singularities of Nonredundant Manipulators*, IEEE, 2002.

Mayorga R.V., Wong A. K. C. Waterloo, *A singularities Preservation Approach for Redundant Robot Manipulators*, IEEE, 1990.

Tcho'n K. Muszy'nski. Wroclaw, *Singular inverse kinematic problem for robotic manipulators: A normal form approach*, IEEE, 1998.

Contour Tracking of a Redundant Robot Using Integral Variable Structure Control with Output Feedback. Lin C., Lee K. Taipei : J Intell Robot Syst, 2011.

Choi H., Lee S., Lee J. Gwangju , *Minimum Infinity-norm Joint Velocity Solutions for Singularity-robust Inverse Kinematics*, International Journal of precision engineering and manifacturing, 2011

Sitografia

Origine e storia del termine robot https://it.wikipedia.org/wiki/Robot

La robotica industriale http://www.automazionenews.it/la-robotica-industriale/

Robot collaborativi https://www.robotiko.it/robot-collaborativi-cobot/

"Yumi - creating an automated future together," https://new.abb.com/products/robotics/industrial-robots/irb-14000-yumi

"Abb robotics - the flexpendant hmi," https://www.youtube.com/watch?v=KTKAsZaEmjs

Libreria ABB, http://search.abb.com/library/Download.aspx?DocumentID=9AKK106354A3256&LanguageCode=en&DocumentPartId=&Action=Launch