Progetto del corso “Real Time and Distributed Systems”

Simulazione del movimento di un braccio robotico

A cura degli studenti: Luca Ricci, Elia Zuccaro 24/01/2020

Matricole: 543990, 608029

Email: luca.ricci.dox@gmail.com, eliazuccaro@gmail.com

Requisiti

L’obiettivo di questo progetto è quello di realizzare un’interfaccia grafica che simuli la presenza di un braccio robotico in ambiente tridimensionale, con possibilità di azionarne i movimenti tramite input dell’utente.

Il braccio in questione è fissato al terreno, ed è composto da quattro link ed altrettanti giunti rotoidali. Non sono presenti giunti prismatici, quindi non sono stati implementati moti traslazionali, ma solo rotazionali, pressappoco come nella meccanica di un braccio umano.

Il programma è stato scritto interamente in linguaggio C, ed è stato fatto uso delle librerie “Allegro” per l’implementazione delle componenti grafiche e “pthread” per la gestione dei task, quindi per la realizzazione vera e propria della componente in tempo reale.

Oltre al movimento del braccio, il programma prevede anche funzioni di monitoraggio delle posizioni di tutti i giunti e dell’*end-effector*, e un contatore delle *deadline miss* avvenute durante l’esecuzione, il tutto in tempo reale.

I movimenti dei giunti sono gestiti simulando il comportamento di motori reali, quindi controllandone l’attuazione con un controllore PID per ciascun giunto azionato.

Architettura software

Il programma è stato organizzato in otto file:

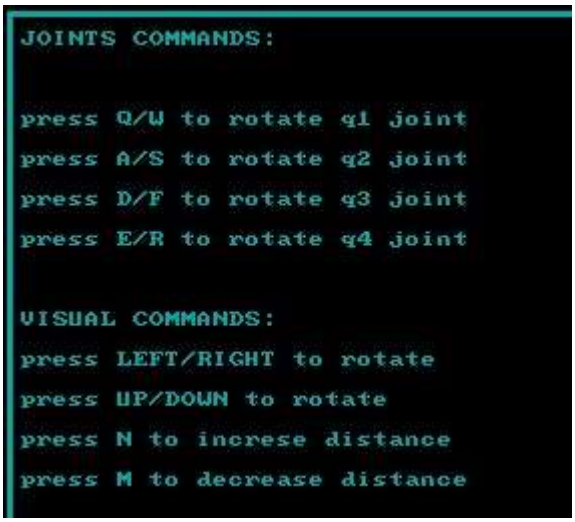
1. *robot_demo.c* è il punto di avvio del programma. Contiene il main, al cui interno un ciclo do-while richiama i task, e si occupa anche della chiusura dello stesso e della distruzione della bitmap all'uscita.
2. *inputread_task.h* è il file che contiene il task per la lettura di input da tastiera. Si occupa di stare in ascolto per l'arrivo degli input e di associare ogni tasto disponibile a un'azione a schermo. Come vedremo nella parte relativa alla meccanica del braccio, a ogni giunto sono associati due movimenti, rispettivamente per incrementare e decrementare l'angolo di giunto. In figura un esempio di implementazione di questa coppia di rotazioni
3. *visual_task.h* è il task che si occupa di aggiornare in tempo reale i parametri spaziali e di diagnostica presenti nella finestra di sinistra. Questi vengono gestiti con delle variabili booleane (come Dflag per le deadline miss e camflag per i movimenti della telecamera), che contengono informazioni sulle eventualità in cui ci siano state delle variazioni nei campi coinvolti. Se una di queste variabili risulta vera, allora il programma si attiva per andare a vedere il tipo di modifiche effettuate, senza essere costantemente in ascolto per tutte quante.
4. *compute_task.h* è il task che si occupa dei calcoli più complessi, che conviene disaccoppiare dagli altri task e gestire in modo compatto, come ad esempio i calcoli delle coordinate dei giunti nello spazio, le loro proiezioni 2D sullo schermo, oppure i calcoli relativi al controllo PID.
5. *support_lib.h* è un contenitore di funzioni, di vario tipo e per vari scopi. Ogni task richiama le funzioni di suo interesse, evitando così di accumulare codice nei singoli file e rendendo i file dei task e del main il più semplici e leggibili possibile. Si va dalle funzioni più complesse, il cui nome è esplicativo, come drow_GUI, refresh_dmiss, refresh_points_value, refresh_ambient, a quelle più semplici, come quelle di conversione gradi-radianti.
6. *ptask.h* è il file per le variabili e le funzioni riguardanti i task, in cui vengono assegnati e calcolati periodo, deadline, priorità, ecc. di ogni task
7. *global_constants.h* contiene le constanti globali

8. *global_variables.h* contiene le variabili globali

Interfaccia utente

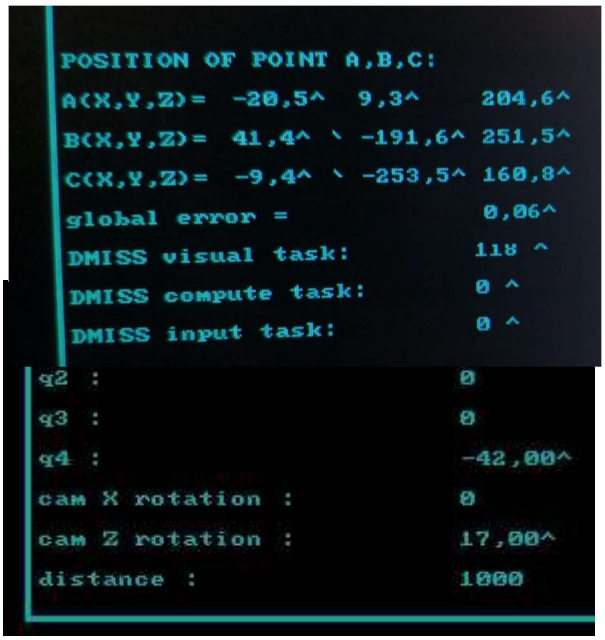
All'avvio del programma, dopo una schermata di introduzione di pochi secondi, si accede immediatamente all'interfaccia utente. Questa è composta da due frame distinti:

- Sulla destra troviamo una raccolta delle informazioni sull'utilizzo e sulla diagnostica del programma in esecuzione. Dall'alto verso il basso sono presenti una legenda dei comandi a disposizione e dei tasti ad essi associati, le posizioni in tempo reale dei giunti e le *deadline miss* registrate fino a quel momento, anch'esse in tempo reale.



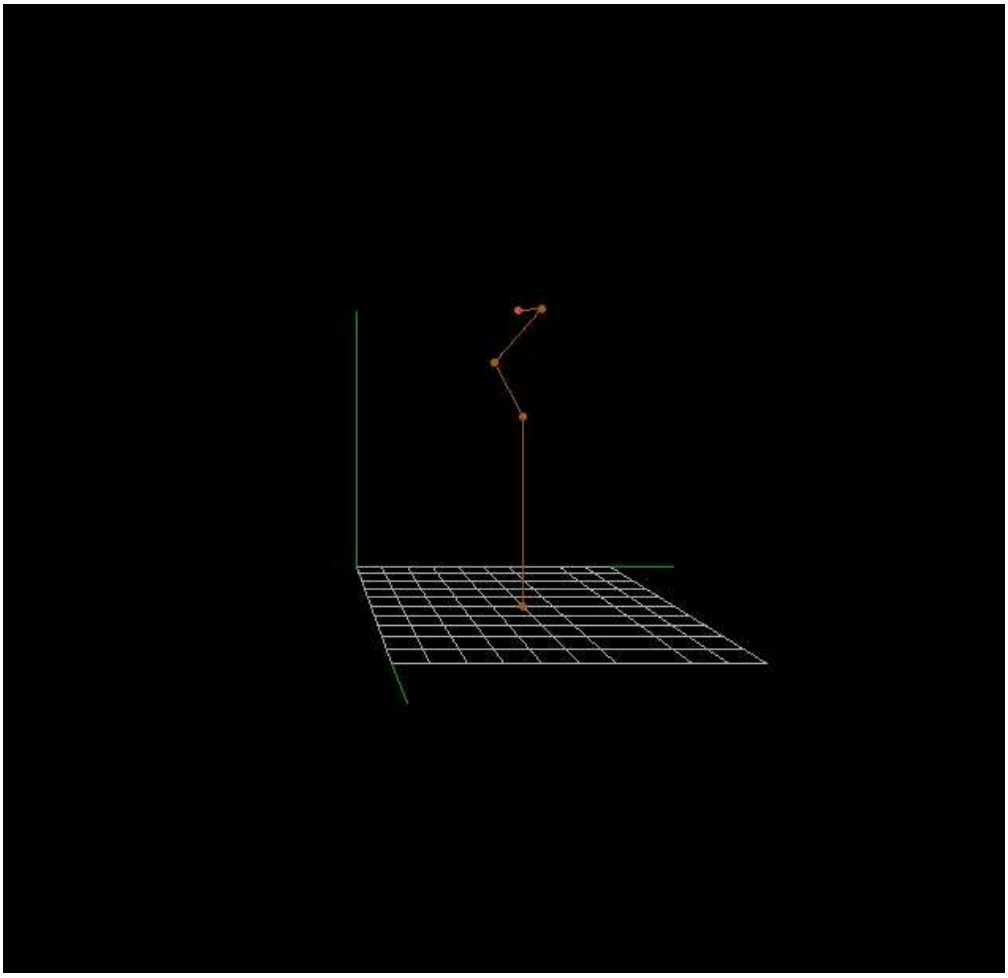
Nella prima parte della legenda troviamo la lista dei tasti e delle azioni ad essi associati, a partire dai comandi di giunto, quattro coppie di rotazioni (per angoli q positivi/negativi) per un totale di otto movimenti elementari.

Abbiamo poi sei comandi per la gestione della visuale: due coppie di rotazioni per movimenti solidali a meridiani e paralleli e zoom in/out



Nella seconda parte troviamo i dati per la localizzazione dei punti e per la diagnostica, a cominciare dalle coordinate cartesiane dei punti A, B e C. Abbiamo poi informazioni sull'errore globale (per la sua definizione vedere il paragrafo "Controllo in retroazione") e sulle deadline miss dei tre task. Infine abbiamo i valori degli angoli di giunto e i parametri visibili a schermo, cioè gli angoli di giunto, gli angoli della camera e la distanza dell'osservatore dal braccio.

- Sulla sinistra abbiamo il frame contenente il braccio e l'ambiente in cui è inserito, entrambi al centro della finestra e osservabili da diverse angolazioni, tramite i comandi di cambio visuale a disposizione.



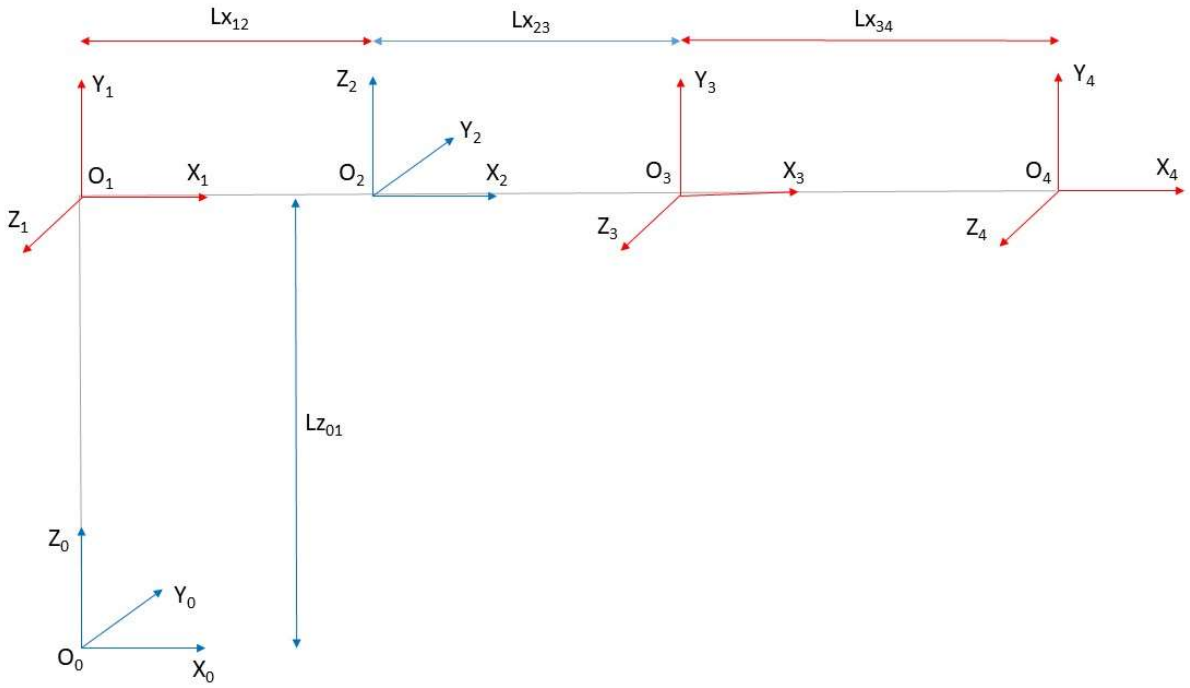
Come da figura, l'ambiente di lavoro è costituito dai tre assi cartesiani (in verde), e da una griglia bianca, la cui origine coincide con quella degli assi, e che giace sul piano XY. Dal momento che la terna di assi è cartesiana, la presenza del piano è sufficiente a identificare gli assi X, Y e Z. Il punto di contatto tra piano e robot è localizzato al centro della griglia, con il primo link solidale all'asse Z. I successivi punti e segmenti sono rispettivamente i giunti e link del braccio, con l'ultimo punto a rappresentare l'end-effector.

Progettazione meccanica

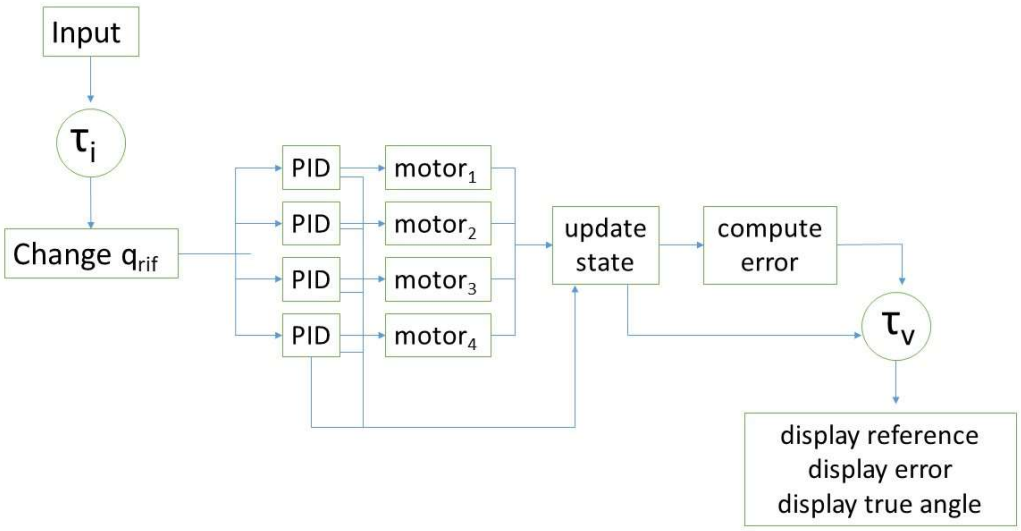
Il modello meccanico del braccio è stato ricavato seguendo il metodo di parametrizzazione di Denavit-Hartenberg (d'ora in avanti DH). Tale metodo consiste nell'assegnazione di un indice a ogni coppia giunto-link, a partire dalla coppia giunto di base – primo link (numerati con 0), e proseguendo in modo tale che a ogni indice corrisponda un giunto e il link a lui successivo. Questa rappresentazione si esplica nella tabella DH, raffigurata nella figura sottostante. I parametri a e d rappresentano le distanze tra un giunto e il suo precedente, sia nell'asse X ($\mathbf{a}_i$) che Z ($\mathbf{d}_i$). Gli angoli $\boldsymbol{\alpha}_i$ occorrono per adattare il sistema di riferimento di un giunto rispetto a quello a lui precedente. Gli angoli $\boldsymbol{\theta}_i$ rappresentano gli angoli di giunto veri e propri.

	$\mathbf{a_i}$	$\boldsymbol{\alpha_i}$	$\mathbf{d_i}$	$\boldsymbol{\theta_i}$
S_0S_1	0	$\pi/2$	L_{z12}	q_1
S_1S_2	L_{x12}	$-\pi/2$	0	q_2
S_2S_3	L_{x23}	$\pi/2$	0	q_3
S_3S_4	L_{x34}	$-\pi/2$	0	q_4

Nella figura seguente è riportato lo schema di DH, con i riferimenti di origine, giunti, link e distanze utilizzati nella tabella DH.



Controllo in retroazione



La parte di attuazione dei giunti è stata sviluppata simulando la presenza di motori veri e propri, con una variabile di controllo e un anello di retroazione con cui gestire l'errore. Nella figura abbiamo lo schema di controllo.

All'interno del programma la variabile errore è definita come differenza tra l'angolo di giunto desiderato e quello attuale. Sulla base di questo si differenziano due tipi di errori: l'errore legato all'angolo del singolo giunto, e l'errore globale, definito come la somma degli errori di tutti e quattro i giunti. Sulla base dell'errore globale si stabilisce una soglia minima sotto la quale scendere per considerare il movimento andato a buon fine. Fino a quando tale valore non è stato raggiunto, il software continua il loop di controllo, calcolando di volta in volta l'input da dare ai motori, come da schema classico di controllo PID.

Il modello matematico utilizzato si basa sulle equazioni alle differenze. Per questo motivo è necessario memorizzare in delle variabili globali alcuni dei valori precedenti degli angoli di giunto e delle velocità, da usare per ricavare le differenze, ed è necessario stabilire un tempo di campionamento di tali grandezze, anch'esso stabilito tra le variabili globali.

Visualizzazione robot a schermo

Nel momento in cui si è stata costruita la schermata di sinistra, col frame per la visualizzazione del robot, sono stati presi alcuni accorgimenti per migliorare l'interazione da parte dell'utente.

In primis è stato utilizzato un buffer per la raccolta dei punti del braccio a schermo, trasmessi poi con l'istruzione blit, in modo da evitare pesanti effetti di sfarfallio che di fatto impedivano di individuare correttamente il braccio nello spazio.

In secondo luogo è stato simulato un sistema di una telecamera in movimento intorno al robot. Nella fattispecie, a ogni input "di visualizzazione" (ossia LEFT, RIGHT, UP, DOWN, N, M) corrisponde un'azione del robot associata all'input trasmesso: ad esempio una rotazione oraria "della telecamera" corrisponde a una rotazione antioraria del robot, oppure uno zoom in corrisponde a una estensione di scala del robot stesso.

Per dare l'idea della tridimensionalità a schermo sono state usate delle funzioni di proiezione, reperibili nel file *support_lib.h* con il nome di *project_x* e *project_y*.